

Scientific Papers
of the Polish Information
Processing Society
Scientific Council

Software Engineering: Improving Practice through Research

Scientific Editor
Bogumiła Hnatkowska, Michał Śmiałek



SCIENTIFIC COUNCIL
POLISH INFORMATION
PROCESSING SOCIETY



Software Engineering: Improving Practice through Research

Scientific Editors

Bogumiła Hnatkowska, Michał Śmiałek

The Polish Information Processing Society Scientific Council

prof. dr hab. Zdzisław Szyjewski – *Chairman*

dr hab. prof. PW Zygmunt Mazur – *Vice-Chairman*

dr hab. inż. prof. PG Cezary Orłowski – *Vice-Chairman*

dr hab. Jakub Swacha – *Secretary*

prof. dr hab. Zbigniew Huzar

prof. dr hab. inż. Janusz Kacprzyk

prof. dr hab. inż. Marian Noga

prof. dr hab. inż. Ryszard Tadeusiewicz

dr hab. Tadeusz Gospodarek

dr hab. Leszek Maciaszek

dr hab. inż. Lech Madeyski

dr hab. Zenon Sosnowski

dr inż. Adrian Kapczyński

dr inż. Andrzej Romanowski

dr inż. Marek Valenta

Authors

Wiktor Zychla – **CHAPTER 1**, Aneta Ponieszewska-Marañda, Patryk Wójcik – **CHAPTER 2**, Piotr Jeruszka, Andrzej Grosser – **CHAPTER 3**, Adrian Najczuk, Artur Wilczek – **CHAPTER 4**, Adrian Najczuk, Artur Wilczek – **CHAPTER 5**, Krzysztof Miśtał, Ziemowit Nowak – **CHAPTER 6**, Ferenc Attila Somogyi, Mark Asztalos – **CHAPTER 7**, Agnieszka Patalas, Wojciech Cichowski, Michał Malinka, Wojciech Stępiak, Piotr Maćkowiak, Lech Madeyski – **CHAPTER 8**, Tomasz Gawęda, Ewa Nestorowicz, Oskar Wólk, Lech Madeyski, Marek Majchrzak – **CHAPTER 9**, Jarosław Hryszko, Lech Madeyski, Marta Dąbrowska, Piotr Konopka – **CHAPTER 10**, Lucjan Stapp, Adam Roman, Maciej Chmielarz – **CHAPTER 11**, Grzegorz Kochański – **CHAPTER 12**, Cezary Orłowski, Tomasz Deręgowski, Miłosz Kurzawski, Artur Ziółkowski, Bartosz Chrabski – **CHAPTER 13**, Cezary Orłowski, Bartosz Chrabski, Artur Ziółkowski, Tomasz Deręgowski, Miłosz Kurzawski – **CHAPTER 14**, Cezary Orłowski, Tomasz Deręgowski, Miłosz Kurzawski, Artur Ziółkowski, Bartosz Chrabski – **CHAPTER 15**, Tomasz Wala, Marek Majchrzak, Jolanta Wrzuszczak-Noga – **CHAPTER 16**

Reviewers

Zbigniew Banaszak, Włodzimierz Bielecki, Miklós Biró, Ilona Bluemke, Přemek Brada, Krzysztof Cetnarowicz, Zbigniew Czech, Włodzimierz Dąbrowski, Iwona Dubielewicz, Mariusz Flasiński, Józef Goetz, Janusz Górski, Adam Grzech, Piotr Habela, Bogumiła Hnatkowska, Zbigniew Huzar, Stanisław Jarząbek, Hermann Kaindl, Audris Kalnins, Piotr Kosiuczenko, Kevin Lano, Laszlo Lengyel, Leszek Maciaszek, Lech Madeyski, Marek Majchrzak, Erika Nazaruka, Ngoc-Thanh Nguyen, Mirosław Ochodek, Aneta Ponieszewska-Marañda, Krzysztof Sacha, Bartosz Sawicki, Klaus Schmid, Mirosław Staron, Andrzej Stasiak, Krzysztof Stencel, Jakub Swacha, Tomasz Szumac, Michał Śmiałek, Lech Tuzinkiewicz, Bartosz Walter, Andrzej Wąsowski, Krzysztof Wnuk, Janusz Zalewski, Jaroslav Zendulka, Adrian Żurkiewicz

Scientific Editors

Bogumiła Hnatkowska, Michał Śmiałek

Copyright by the Polish Information Processing Society, Warszawa 2016

ISBN: 978-83-943248-2-7

Edition: I. Copies: 100. Publishing sheets: 13,2. Print sheets: 17,1.

Publisher, print and branding: WESTGRAPH,

Przeclaw 96c/5, 72-005 Przeclaw, www.westgraph.pl

Contents

PREFACE.....	7
I. SOFTWARE ARCHITECTURE AND MODELLING	
1. HETEROGENEOUS SYSTEM ARCHITECTURE IN EDUCATION MANAGEMENT SOFTWARE	13
2. SERVICE-ORIENTED ARCHITECTURE FOR INTEGRATION OF INFORMATION SYSTEMS AT DATA LEVEL.....	31
3. THE USE OF WEB SERVICES ARCHITECTURE FOR ENGINEERING CALCULATIONS BASED ON THE WEBMES PLATFORM.....	49
4. DESIGNING A DATA WAREHOUSE FOR CHANGES WITH DATA VAULT.....	65
5. TOWARD AGILE DATA WAREHOUSING.....	81
6. PERFORMANCE ANALYSIS OF WEB APPLICATION USING MYSQL, MONGODB AND REDIS DATABASES	97
7. MERGING TEXTUAL REPRESENTATIONS OF SOFTWARE MODELS – A PRACTICAL APPROACH	113
II. SOFTWARE MAINTENANCE	
8. SOFTWARE METRICS IN BOA LARGE-SCALE SOFTWARE MINING INFRASTRUCTURE: CHALLENGES AND SOLUTIONS	131
9. HOW TO IMPROVE LINKING BETWEEN ISSUES AND COMMITS FOR THE SAKE OF SOFTWARE DEFECT PREDICTION?.....	147
10. DEFECT PREDICTION WITH BAD SMELLS IN CODE	163
11. POSTGRADUATE STUDIES ON SOFTWARE TESTING IN POLAND.....	177
12. DATA FLOW ANALYSIS FOR CODE CHANGE PROPAGATION IN JAVA PROGRAMS	187

III. AGILE TRANSFORMATIONS

13. TRIGGER-BASED MODEL TO ASSESS THE READINESS OF IT ORGANIZATIONS TO AGILE TRANSFORMATION.....207

14. THE REFERENCE MODEL OF TOOLS ADAPTATION IN THE PERSPECTIVE OF TECHNOLOGICAL AGILE TRANSFORMATION IN IT ORGANIZATIONS223

15. BUILDING PROJECT AND PROJECT TEAM CHARACTERISTIC FOR CREATING HYBRID MANAGEMENT PROCESSES241

16. EXPERIENCE REPORT: PROCESS OF INTRODUCTION OF DEVOPS INTO PRODUCTION SYSTEM257

Preface

Progress in software engineering is strongly related to advancements in the software industry. It would be ideal if this progress depended on challenges emerging during actual software projects and new solutions coming from academic research. Successes and failures in software projects should stimulate directions of academic research; industrial practice should validate academic results.

Unfortunately, everyday practice is not ideal. Thus the KKIO Software Engineering Conference series serves as a platform to promote cooperation between industry and academia right from its beginning. This book is based on materials prepared for the 18th edition of the conference held under the motto: “Better software = more efficient enterprise: challenges and solutions”.

The book relates to cooperation between universities and software industry mainly in Central Europe, with the emphasis on Poland. We leave to the Reader the assessment of the degree to which the materials reflect the current situation and its quality. Nevertheless, comparing this book to the volumes from previous conferences we can notice some progress. Hopefully, this trend will continue in the future. We also hope that the book becomes a contribution to current discussions on the development of universities in Central Europe and Poland that emphasise the importance of linking academic research with practice.

The book is split into three parts. Part I is devoted to Software Architecture and Modelling. In the first chapter, Wiktor Zychla presents a dynamic architecture suitable for large-scale education management systems. The architecture features such quality factors as flexibility, security, and scalability. In the second chapter, Aneta Ponieszewska-Marańda and Patryk Wójcik address the problem of integrating heterogeneous applications into a consistent homogenous system by reusing the existing components and ensuring transparency of exchanged data. They concentrate on the use of Service Oriented Architecture (SOA) – mainly micro-services – for this purpose. The pros and cons of the proposed solution are discussed. In the third chapter, Piotr Jeruszka, and Andrzej Grosser present the architecture of the webMES system

that supports numerical simulations. The system is service-oriented, and it collaborates with existing engineering software (e.g. NuscaS). The chapter presents the basic network services and data needed to perform calculations. The next two chapters, written by Adrian Najczuk and Artur Wilczek, treat the topic of data warehouse modelling. The first one researches the Data Vault architecture to find it better adjusted for changes than the alternatives. The second one presents outcomes of applying selected agile practices to warehouse modelling and development. In the sixth chapter, Krzysztof Miśtał, and Ziemowit Nowak conduct performance analysis where a web application communicates with either a non-relational database (MongoDB, Redis) or a relational one (MySQL). The experiments show that the non-relational solution is more efficient, especially under heavy traffic. The last chapter in Part I concentrates on modelling. The authors, Ferenc Attila Somogyi and Mark Asztalos, present the details of a method that aims at merging textual representation of software models. The method works on abstract syntax tree representations of merged models and addresses different conflicts that can appear during the integration process.

Part II of this monograph is devoted to Software Maintenance. Software testing and defect prediction are its central concerns. In the eight chapter, a group of authors from the Wrocław University of Science and Technology (Agnieszka Patalas, Wojciech Cichowski, Michał Malinka, Wojciech Stępnia, Piotr Maćkowiak, and Lech Madeyski) present how to use a large-scale software repository mining platform called Boa in practice. Specifically they explain how to retrieve interesting metrics of open-source projects. These metrics can be used – for example – to build software defect prediction models. Chapter 9 presents descriptions of three algorithms (ReLink, MLink, and RCLinker) that aim at linking issues reported in issue trackers with commits in versioning repositories. After an analysis, the authors (Tomasz Gawęda, Ewa Nestorowicz, Oskar Wołk, Lech Madeyski and Marek Majchrzak) have implemented one of them (RCLinker), and checked the results on open-source projects first, and next – on a commercial project provided by Capgemini. In chapter ten, the authors (Jarosław Hryszko, Lech Madeyski, Marta Dąbrowska, and Piotr Konopka) answer the question of whether code smell metrics, added to the basic metric set, can improve defect prediction in an industrial project. The main finding of their research is that the impact of code bad smells

metrics on defect prediction is negligibly small. Chapter 11 is about postgraduate studies on software testing in Poland. The authors (Lucjan Stapp, Adam Roman, and Maciej Chmielarz) present the profile, experiences and expectations of postgraduate students as well as the curricula of studies at four universities and colleges. In the twelfth chapter, Grzegorz Kočański proposes an approach for managing code change propagation in weakly typed languages. Such solutions are crucial in maintenance processes where a change in one place can influence many other places. The author defines a simple formalism for describing various effects of change propagation as well as for defining stack transformations and operations for manipulating local variables and object fields. The notation covers typical constructs like conditional statements, iterations, arrays and exceptions.

The last part of the book (III) addresses problems of IT Agile Transformations. Three first papers in this part form a logically connected sequence written by the same authors: Cezary Orłowski, Tomasz Deręowski, Miłosz Kurzawski, Artur Ziółkowski and Bartosz Chrabski. The first paper presents key factors and enablers in IT company's agile transformation process. Authors define the stages of readiness to agile transformations and the elements that should be analysed. The second paper proposes a reference model of tools adaptation in the perspective of agile technological transformation. The model was validated in two business cases. In the third paper, authors propose a method that allows for building a characteristic of the project and project team. Such a characteristic can be used for many purposes, like selecting the proper software development methodology among others. In the last chapter, Tomasz Wala, Marek Majchrzak and Jolanta Wrzuszczak-Noga, describe a process to introduce some elements of the Development and Operations (DevOps) practices to a production system in the Capgemini company. The system was originally developed using a waterfall methodology which was then transformed into Scrum. The DevOps pillars were accompanied by the "continuous everything" approach.

I. Software Architecture and Modelling

Chapter 1

Heterogeneous System Architecture in Education Management Software

1. Introduction

1.1. The need for a versatile architecture

Designing and implementing education management software was always a challenge but the Internet introduced a whole new era by giving a lot of new possibilities to both users and software providers. Systems slowly turned from local desktop applications focusing on selected areas and used by a few users to multiple web applications from different vendors, developed using different technologies and integrated together and thus forming large ecosystems of services for pupils, their guardians, teachers and managers.

In recent years, more and more local governments call for tenders ([1–5]) for implementing and deploying integrated web-based systems usually covering multiple different areas of educational management, including:

- school management – grades, attendance, web and mobile access for pupils,
- guardians and teachers,
- supervision of compulsory schooling,
- computer-aided school enrolment,
- online e-Learning systems,
- timetable planning,
- organizational charts planning,
- HR and accounting,
- public information newsletters and other kind of content management.

Unfortunately, not only so such systems have to be large in scale and must fulfil a detailed list of functional and non-functional requirements but also requirements vary from call to call. For example, a grade/attendance is often (but not always!) listed but other areas of educational managements are more or less optional. Another difficulty stems from the fact that while there are at least few software publishers in Poland who offer most services that could possibly be called for a detailed list of requirements in a specific call usually introduces such new features that are not present in the already implemented software. This leads to a situation where each single call for tender introduces new challenges and direct competitors in one call (where a single publisher can possibly deliver all software components alone) can closely cooperate in another call (where in order to fulfil caller's requirements, a system has to be composed of multiple applications from different publisher; moreover, a similar component requirement in two different public procurements can possibly be fulfilled by two different applications from two different publishers!).

Currently, the largest existing deployment consists of about 25 different modules and the number of different software publishers in a single deployment varies from 2 to 5.

Ultimately, each such educational management system can possibly be composed of unknown number of different applications from different publishers – an exact number of applications that form a system and an exact number of publishers who form a consortium which answers the call usually differ from call to call.

There is then an interesting challenge, a challenge of **flexible and scalable architecture** behind each such project. An architecture that would create a solid foundation and would quickly adapt to any similar set of requirements in the future. An open architecture focusing on interoperability so that different software platforms used by different publishers would talk to each other giving a seamless experience for users expecting a single, integrated solution rather than a set of different applications.

1.2. The challenge taken

The challenge for architecture of such heterogeneous integrated education management system was taken early 2011. A core architecture was successfully reused later and currently there are over a dozen of deployments: a single shared cloud deployment and multiple on-premise deployments on client's infrastructure. Table 1 is a summary of selected major deployments from 2011–2015 including approximate number of actual user accounts.

There are a few **Common Principles** behind. CPs are **requirements** which have been identified as common in most public procurements and in rare cases where a requirement was not explicitly stated in the original specification, it has always been acknowledged by contracting authorities during consultation phases of technical dialogues.

Table 1. Selected major deployments

Installation	Local government	Year	User accounts in thousands (January 2016)
Shared	NA	2010 and later	1200
GPE	Gdańsk	2012	110
ZSZO	Radom	2014	100
Resman2	Rzeszów	2013	100
Portal edukacyjny	Szczecin	2012	90
Opolska e-Szkoła	Opole	2011	90
EduNet II	Tarnów	2014	70
ESZO	Koszalin	2012	40
EduS@cz	Nowy Sącz	2012	20

Single Source of Information (SSI)

It is expected that the information is reused throughout the system and there is no need to re-enter the information to any module of the system assuming it has already been entered somewhere in the system.

Automatic Account Provisioning (AAP)

It is expected that user accounts are provisioned automatically which means that an account with corresponding set of permissions has to be created/updated as soon as the person registry entry is created/updated in an application which is the source of information about the group of users the user belongs to. This will be further discussed in Section 4.

Single Sign-On (SSO)

It is expected that all users gain access to all modules of the system upon a logging once. It is also expected that a global user registry doesn't only authenticate but also authorize users (which leads to a Claims-Based-Security-driven approach). It is also expected that a single sign-out operation is sufficient to terminate user's current session which means that all modules become inaccessible instantly.

The single sign-on will be further discussed in Section 4.1.

Data Integration (DI)

It is expected that there is a way of distributing the information between various modules of the system. This distribution is assumed to be user-assisted or automatic, depending on the specific business cases. For example, while the account provisioning (AAP) is expected to be automatic, importing pupil data from an enrolment module to a school management module should only be performed manually and on purpose.

The data integration will be further discussed in Section 3.

The goal of this submission is to show how these requirements are fulfilled while maintaining the scalability and flexibility of the overall architecture and keeping it secure in terms of identity and data integration.

2. Overview of the System Architecture

The overview of the system architecture is presented in Figure 1.

The frontend layer is composed of modules which are available to users and cover the functional requirements of the system. Since modules are

developed by multiple software publishers, various software platforms are used in the development (Microsoft .NET, Java EE, PHP, others) and various operating systems are used to host web-based services (Windows Server platforms and different distribution of Linux-based systems).

The backend layer – the one we focus on in this paper – consists of three components described in more details in the following sections.

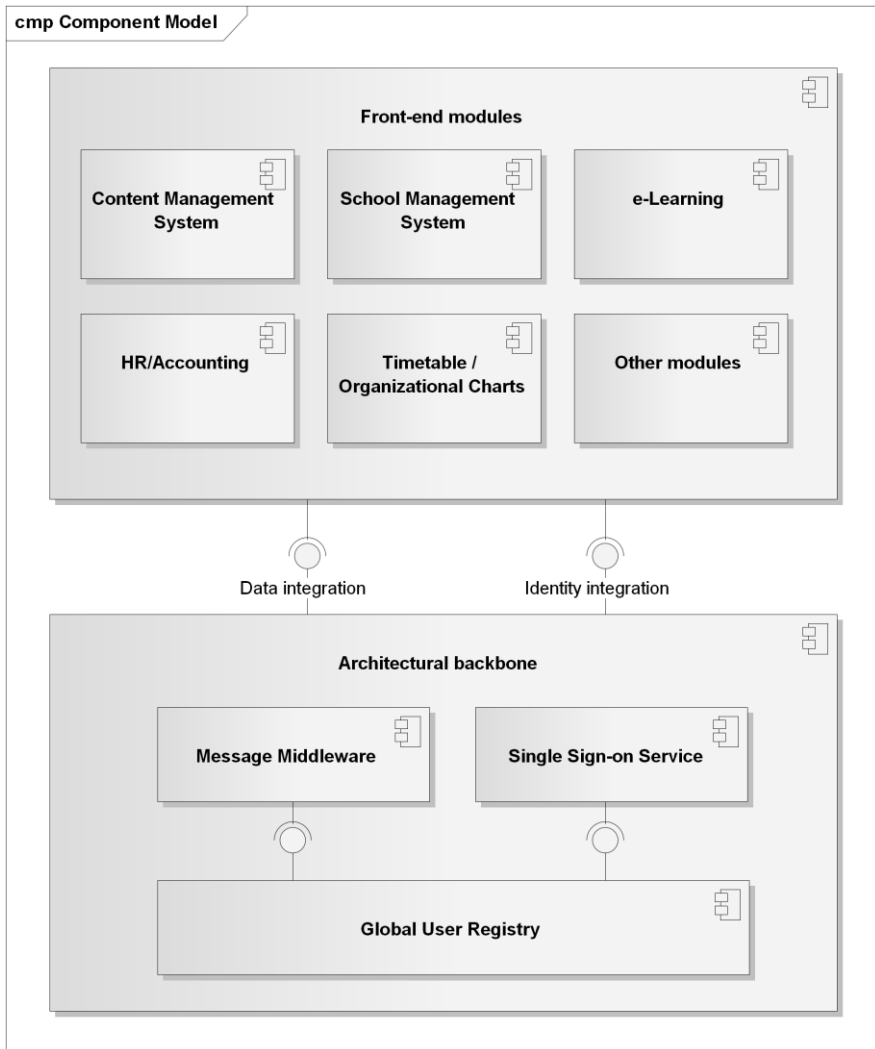


Figure 1. Overview of the System Architecture

Messaging Middleware

The Messaging Middleware is implemented as a Data Service Bus which enables system-wide publication/subscription communication based on WSDL/ HTTPs protocols.

Global User Registry

The Global User Registry implements a source of information about users and their roles as well as organizational units which are part of the system (schools and local governments).

Single Sign-On Service

The Single Sign-On Service implements several active and passive flows of enterprise SSO protocols and what is interesting here is that it acts as a **Protocol Normalization Service** – it is both a proxy and an adapter, hiding implementation details of several protocols and exposing a narrower but predictable set of services.

3. Message-oriented Middleware

The data integration across the system is based on the Service-Oriented Architecture pattern ([6, 7, 8]) specifically, the Event-Driven Architecture approach [9].

The conceptual diagram of the Data Service Bus component is presented in Figure 2. The component depends on two elements:

- **Domain Language** – A common Domain Language [10] is developed so that all modules across the system are able to both send and receive notifications.
- **Queuing Subsystem** – A reliable Queuing Subsystem [8] is used for publish/subscribe pattern implementation.

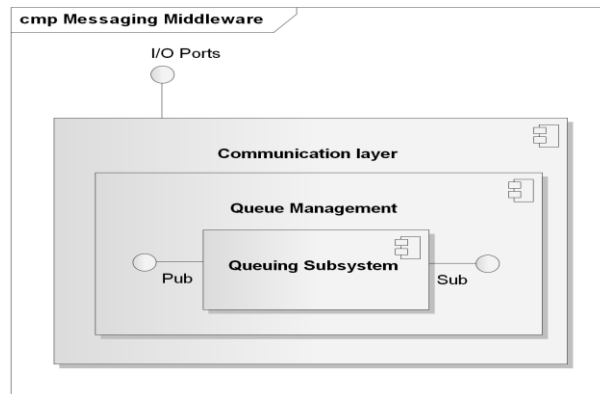


Figure 2. Simplified architecture of the messaging middleware

3.1. Domain Language

The Domain Language is one of the most important parts of the architecture and its design takes significant amount of time. It requires a solid business background and validation.

The Domain Language of the system is constantly evolving. Starting from 4 types of entities, it currently consists of about **100 different domain types** in several categories:

- **school management system** pupil data, guardian data, teacher data, groups, assignments, subjects, attendance, grades
- **accounting** budget plans, plan items, sections and positions
- **HR** work positions, salary tables, salary table items
- **organizational data** school data, branches, divisions, local government data
- **shared dictionaries** Educational Information System dictionaries [11]

Because of the strict legal requirement [12], the specification is XSD/WSDL based which means that for every single type, its XSD schema is publicly available as a part of the specification.

And example notification is presented in Figure 3 and its specification in Figure 4.

```

<?xml version="1.0" encoding="utf-16" ?>
<EnvelopedMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Id>4bf4f78a-010b-426c-9cfe-2163ac68a10f</Id>
  <Date>2016-04-06T14:52:39.8521016+02:00</Date>
  <Sender>CN=[example certificate]</Sender>
  <CorrelationId>00000000-0000-0000-000000000000</CorrelationId>
  <MessageTypeName>Vulcan.eSzkoła.Services.Model.Uczniowie.Uczen_v2
</MessageTypeName>
  <MessageXml>
    <Uczen_v2 xmlns="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
      <Uid>33092828874</Uid>
      <Context>Wroclaw</Context>
      <DaneOsobowe>
        <TypUid>PESEL</TypUid>
        <Imie>Poniktof</Imie>
        <Nazwisko>Cinikola</Nazwisko>
        <DataUrodzenia>1933-09-28</DataUrodzenia>
        ...
      </DaneOsobowe>
      <Uid_Jednostka>13159058-435a-48b7-9804-088d32b1fc02</Uid_Jednostka>
      <Kod_Jednostka>Oncyliusz</Kod_Jednostka>
      ...
      <Status>Uczen</Status>
      <KsiegaEwidencji>
        <MiejsceNauki>Pawiko</MiejsceNauki>
        <PodlegaObowiazkowi>true</PodlegaObowiazkowi>
        <Numer>341</Numer>
        ...
      </KsiegaEwidencji>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <CanonicalizationMethod Algorithm=
"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          <SignatureMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#rsa-sha1"/>
          <Reference URI="">
            <Transforms>
              <Transform Algorithm=
"http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#sha1"/>
            <DigestValue>Vph2gr9IEm471TLVnRcNNILDft4=</DigestValue>
          </Reference>
        </SignedInfo>
        <SignatureValue>Z8rtEZRUPrXNsnMcAPqVOVL4do7oHHLzDlhhZMUTsk...
</SignatureValue>
        <KeyInfo>
          <X509Data>
            <X509Certificate>MIICNzCCAaCgAwIBAgIQAMmclB12C4DM4fGTa...
          </X509Certificate>
          </X509Data>
        </KeyInfo>
      </Signature>
    </Uczen_v2>
  </MessageXml>
</EnvelopedMessage>

```

Figure 3. Example pupil data notification

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema elementFormDefault="qualified" >
  <xs:element name="Uczen_v2" nillable="true"
    type="Uczen_v2" />
  <xs:complexType name="Uczen_v2">
    <xs:complexContent mixed="false">
      <xs:extension base="ModelBase">
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1"
            name="DaneOsobowe" type="DaneOsoboweDS" />
          <xs:element minOccurs="0" maxOccurs="1"
            name="Uid_Jednostka" type="xs:string" />
          <xs:element minOccurs="0" maxOccurs="1"
            name="Kod_Jednostka" type="xs:string" />
          <xs:element minOccurs="1" maxOccurs="1" name="Od"
            nillable="true" type="xs:date" />
          <xs:element minOccurs="0"
            maxOccurs="1"
            name="KsiegaEwidencji"
            type="KsiegaEwidencji" />
          ...
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Figure 4. Part of the pupil data notification specification

Beside a formal interoperable specification which makes it possible to integrate applications developed with various software platforms, software development kits (SDKs) are provided for selected software platforms, namely the .NET (SDK is production ready) and Java EE (SDK is under development).

The SDK provides a complete object model for the Domain Language and numerous APIs which make it easier to implement an interoperable code. The SDK also contains a custom **synchronous broker** which acts as a development-time replacement of the fully-fledged service bus used in production environment. The broker is able to send, receive and pipeline messages but provides a basic set of debugging operations, including message inspection and modification. We have found that using the broker for early integration tests of applications which publish/subscribe notifications, significantly reduces the time spent later on acceptance tests. On the other hand, the broker is a lightweight desktop application and can be distributed and configured much easier than the actual service bus used in production environment.

3.2. *Queuing subsystem*

The queuing subsystem is used internally by the messaging middleware to implement basic publish/subscribe scenarios. At the moment the RabbitMQ/AMQP [13] is used as it achieves the throughput of up to **20k m/s** on a single server and can be easily scaled-out horizontally. In a few deployments, the MSMQ is used as a queuing subsystem [14] with noticeably lower throughput (about 1k m/s).

The bare queuing subsystem is extended with support of message serialization, exceptions and retries and the control of subscription processing throughput. Also, since the AMQP protocol is much more low-level than the required level of abstraction of the interoperable communication protocol, a set of input/output ports is implemented supporting various kinds of authentication/authorization, notably:

- **WSDL/HTTPs** the port that supports a symmetric pub/sub contract and XMLDSIG based signatures ([16, 17]) involving the shared X509 infrastructure and following the **Webservice Gateway** pattern ([8], [9])
- **DBMS** the port that relies on the DBMS authentication/authorization
- **file system** the port that relies on file system ACLs and easier maintenance

The overall throughput of the messaging middleware, including the overhead from signature validation on input ports and resigning XML notification on output ports is about **1.5k m/s** and is scaled horizontally if necessary.

4. Identity Management

The Global User Registry is the source of information on all users and their roles as well as organizational units throughout the system. There are two implementations of the registry, the DBMS implementation and the Directory Service implementation (the latter is recommended when there are desktop applications used by some users and integrated, Kerberos-based authentication is required for these applications). A web-based SSO component is used to deliver active/passive web-based authentication scenarios.

From the Data Integration perspective, the Global User Registry is one of components that subscribe to notifications about users and it updates the registry accordingly. These notifications are published by Authoritative Information Sources. An AIS's business responsibility is to maintain user registry for selected groups of users. The introduction of AISes is a direct implementation of previously mentioned AAP and SSI principles.

There are obvious candidates for information sources of some groups of users. For example, pupil data and guardian data is maintained by School Management Systems, specifically, School Secretary's Office staff is responsible for keeping the record up-to-date. Teacher and other personnel data come from HR systems. The flow of information about users from designated data sources to the Global Registry is presented in Figure 5.

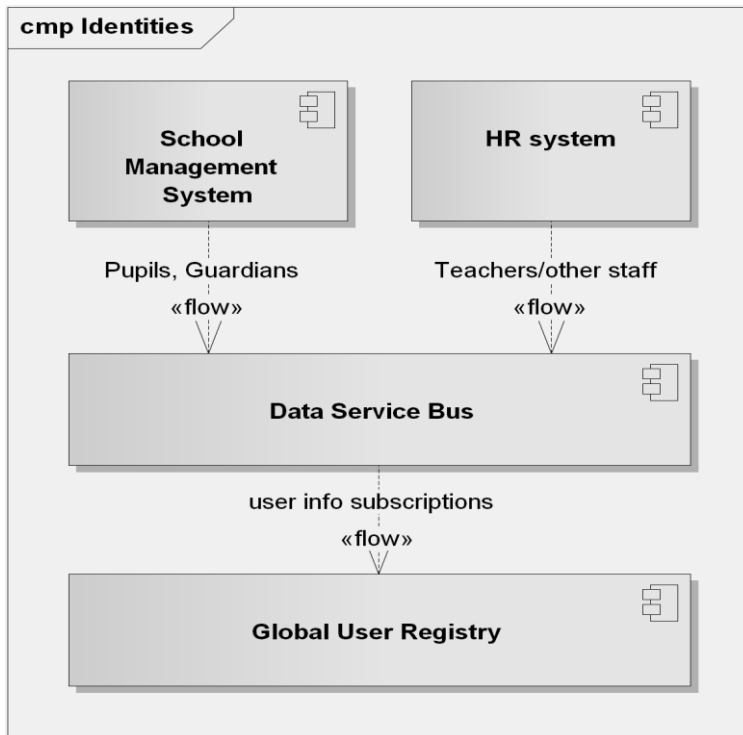


Figure 5. Information flow from data sources to the Global Registry

There are interesting cases when the very same user data is registered independently by two or more AISes. The case is common – two notable

scenarios are pupils who graduate a lower level school and move to a higher level one and guardians who need to access their pupil information in two or more different schools.

There are two ways to handle such cases:

- **enforcing global identification** – this approach is possible for pupil and teacher data. The idea is to use the Polish national identification number (PESEL) to identify the same person data published by two or more different AISes. There are no legal impediments on using PESEL for such identification.
- **no global identification** – this approach is used for guardian data. In this case there is a legal conflict between the requirement to create a single account for a single person and the requirement to follow personal data protection obligations. The conflict is resolved by Local Government authorities who take legal responsibility for their decision: they could either require guardians to provide the identification number (thus providing a single account throughout the system) or require the system to create a **identification substitute** each time the guardian data is entered to the system somewhere (thus following a strict legal obligation but creating possible duplicates in the Global Registry).

4.1. Single sign-on

The architecture of the Single Sign-on component introduces an interesting idea of Protocol Normalization. The idea is presented in Figure 6.

There is an important motivation behind this idea – although the SSO component should offer a reliable and well-defined contract, an actual identity provider sometimes doesn't implement one for various business and technical reasons. Therefore, the SSO component "normalizes" actual SSO protocols by acting as a Relying Identity Provider (R-IdP) ([16]):

- to a relying party (an actual client application) the SSO component looks like a regular identity provider which implements both WS-Federation [18] and OAuth2 [19] protocols,

- to an actual identity provider the SSO component looks like a regular client application which follows the actual provider's protocol specification.

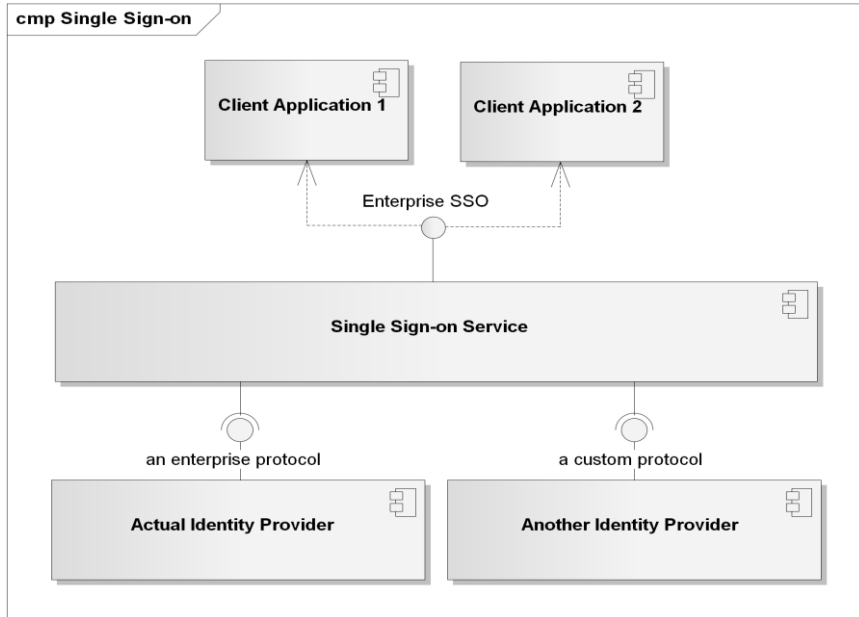


Figure 6. Normalization of SSO protocols

As an example, we present how actual normalization works in a scenario when the client expects the identity provider implements the OAuth2 protocol but the actual provider implements the WS-Federation protocol. The flow of the federated normalization is presented in Figure 7 and the request/reply sequence is as follows:

- 1) An unauthenticated user initiates a session with a client application. The application redirects the request back (HTTP 302) to the identity provider, the R-IdP in this case.
- 2) The browser initiates an OAuth2 authorization code flow sequence with the identity provider.
- 3) (this is where the normalization starts) Instead of just authenticating the user and returning the OAuth2 authorization code, the R-IdP initiates a WS-Federation authentication sequence with its identity provider, the

- IdP. The identity provider performs an actual authentication and returns a WS-Federation SAML token back to the browser.
- 4) The WS-Federation flow continues – the SAML token is POSTed back to the RIdP. The R-IdP establishes a local session and **caches** incoming Claims so that they can be reused later, when an OAuth2 graph API query will be issued by the client application. The WS-Federation flow is complete. The R-IdP issues the OAuth2 authorization code back to the application.
 - 5) The application continues the OAuth2 flow, it asks for exchanging the code for a token.
 - 6) The application uses the OAuth2 token to query the R-IdP's graph API for actual user data. This is where the claim cache is used and the information is returned back to the application.

The above example session of OAuth2 =>WS-Federation normalization is used quite commonly.

Another common scenario is a regular WS-Federation=>WS-Federation relying where the SSO component just delegates the authentication to another provider. The latter scenario is used in the multitenant, shared deployment (see Section 1.2) where different tenants are configured to use their own actual identity providers.

5. Conclusions and Future Work

Considering its scale and existing deployments, we have built one of the largest and widely used architectural foundation for heterogeneous education management systems in Poland. And while some elements are specific to the area of education (the Domain Language), other ideas could potentially be used in different context (the SSO normalization). The system meets the requirements (Common Principles) but focusing on flexibility, scalability and security lets us easily use it as an architectural backbone of about 20 different implementations (with a few other pending).

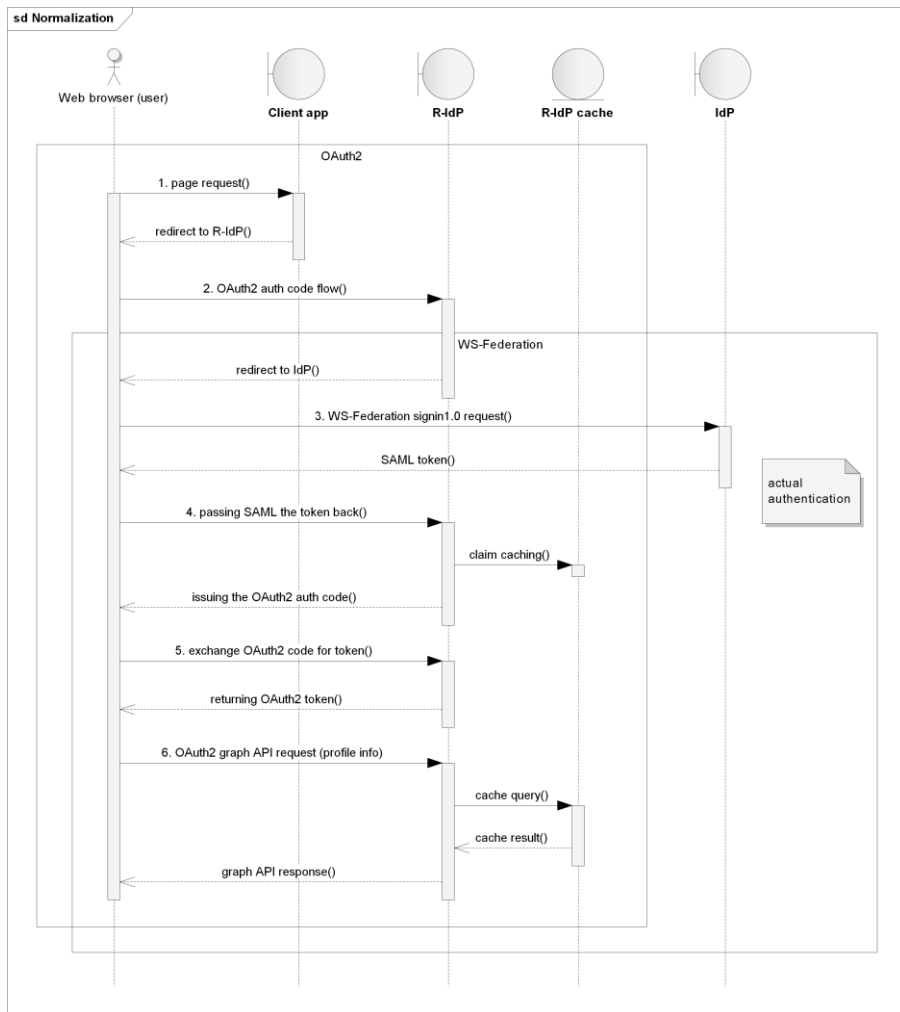


Figure 7. Example OAuth2 =>WS-Federation normalization sequence

While the design has been partially based on the School Interoperability Framework (SIF) specification [20], our contribution introduces:

- the Domain Language for Polish educational system,
- the XML/WSDL/HTTPs-based communication with XMLDSIG-based signatures,
- the specification of the SSO infrastructure with protocol normalization.

The Domain Language is constantly evolving and needs to be updated to follow current legal obligations. There is a versioning policy enforced and while there is an agreement on "extreme cause in changing/updating the Domain Language" (as each change potentially involves many systems that already implement the language), the general rule of the versioning policy is: **if a type needs to be changed, a new type is created**. This rule, together with explicit translation rules between types, so far lets us safely expand the language while keeping old systems compatible.

There are also features planned for future in the area of Single Sign-on. For example, there are other enterprise SSO protocols used by some identity providers which are not currently supported by the SSO component – the SAML2 [21] and Shibboleth [22].

Also, due to the distributed nature of architectural components, there is work to be done in the monitoring/auditing area. While it is currently possible to monitor/audit components separately, there is no easy way of combining audit traces from different components and it is mostly performed manually. A high-level auditing mash-up could be implemented with possible warning/alert triggering.

References

- [1] Elektroniczny System Zarządzania Oświatą, Call for tenders, <http://ted.europa.eu/udl?uri=TED:NOTICE:229491-2011:TEXT:PL:HTML>, Koszalin, 2011.
- [2] Gdańska Platforma Edukacyjna, Call for tenders, <http://www.gdansk.pl/bip/zamowienia-publiczne,825,19214.html>, Gdańsk, 2011.
- [3] EduS@cz, Call for tenders, http://www.nowysacz.pl/content/resources/przetargi/urzed_miasta/2012/0174_12/siwz.pdf, Nowy Sącz, 2012.
- [4] Zintegrowany System Zarządzania Oświatą, Call for tenders, <http://bip.radom.pl/download/69/40737/SIWZ1552.pdf>, Radom, 2013.
- [5] Edunet, etap II, <http://bip.malopolska.pl/umtarnow/Article/get/id,906097.html>, Tarnów, 2014.
- [6] M. Fowler. Patterns of enterprise application architecture. Addison-Wesley Longman Publishing Co., Inc., 2002.
- [7] G. Hohpe, B. Woolf. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional, 2004.
- [8] A. Rotem-Gal-Oz. SOA patterns. Manning, 2012.

-
- [9] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Kroghdahl, M. Luo. Newling T. Patterns: serviceoriented architecture and web services. IBM Corporation, International Technical Support Organization, 2004.
 - [10] E. Evans. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.
 - [11] Ustawa o systemie informacji oświatowej, Dz. U. 2011 Nr 139 poz 814.
 - [12] Krajowe Ramy Interoperacyjności Dz.U. 2012 poz. 526.
 - [13] AMQP 0.9.1 Explained, <https://www.rabbitmq.com/tutorials/amqp-concepts.html>
 - [14] Microsoft Message Queuing, [https://msdn.microsoft.com/en-us/library/ms711472\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx).
 - [15] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, P. Sommerlad. Security Patterns: Integrating security and systems engineering. John Wiley & Sons, 2013.
 - [16] D. Baier, V. Bertocci, K. Brown, M. Woloski, E. Pace. A Guide to Claims-Based Identity and Access Control: Patterns & Practices. Microsoft Press, 2010.
 - [17] XML Signature Specification, <https://www.w3.org/Signature/>, RFC 3275, 2002.
 - [18] WS-Federation Protocol Specification, <http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html>, OASIS, 2009.
 - [19] OAuth2 Protocol Specification, <http://www.rfc-base.org/rfc-6749.html>, RFC 6749, 2012.
 - [20] SIF Implementation Specification, <https://www.sifassociation.org/Specification/Pages/United-Kingdom.aspx>, 2007-2014.
 - [21] SAML2 Protocol Specification, <http://saml.xml.org/saml-specifications>, OASIS, 2005.
 - [22] M. Erdos, S. Cantor. Shibboleth architecture draft v05. Internet2/MACE, May, 2002.

Chapter 2

Service-Oriented Architecture for Integration of Information Systems at Data Level

1. Introduction

The complexity of information technologies (IT) has increased significantly over the last 20 years, introducing to the enterprises more and more applications to improve the efficiency of operation and business management. Business is constantly evolving through new markets, trends, mergers or development strategies. The main challenge with which the IT has to measure now is to keep pace with this development. Unfortunately, complex, inflexible, heterogeneous systems inefficiently used are often the result of such rapid growth.

Users of information systems are demanding immediate access to current information which the organization or organizations cooperating with it have, regardless of the system in which this information is stored. The need to adapt the systems for their mutual operation and integration is linked to it precisely. Manually updating of information between the systems would entail too much effort and the risk of inaccuracies or inconsistencies of updated data.

The integration of heterogeneous applications in consistent homogeneous systems using existing components while maintaining the transparency of the exchange of data between them has become a challenge for software developers [3]. The main problems faced by the existing solutions are low scalability, high costs, low flexibility and speed of implementation of existing solutions, as well as the need for significant interference in the source code of existing applications.

The integration of information systems can be implemented both at the application source code level by re-implementations of functions or applications, but also at the level of exchange of data or business logic through the formalization of services descriptors and the access to applications. Increasingly used web services with the popularization of Service-Oriented Architecture (SOA) change the paradigm by introducing the service-based communication between heterogeneous systems and enterprises [1, 2, 10, 15]. It avoids many common problems faced by other approaches.

The problem presented in the paper concerns the possibility of integrating existing heterogeneous systems for data exchange in accordance with the concept of service-oriented architecture. It was achieved according to SOA, mainly on the basis of network services.

The presented paper is structured as follows: section 2 gives the outline of types and methods of information systems integration. Section 3 deals with the concepts of service-oriented architecture and micro-services. Section 4 presents the proposed model of systems integration based on micro-services, while section 5 describes the architecture of integration model, showing its elements, their responsibilities and functionalities.

2. Integration of Information Systems

Normally one to several less or more complex information systems exist in companies of different size. Most of them have completely different functions that putting in one set of software would be much troublesome. However, despite the differences these systems often operate on the same or dependent data.

Integration is essential in order to ensure the functioning of business processes and data exchange between systems, both within the company and between the companies. It has to provide a spectacular, reliable and secure exchange of data between different parts of the system or systems, regardless of language, data format or platform on which these systems are based.

2.1. Types of integration

Designing and creating the integration solutions enabled the authors of [4] to identify six types of integration projects:

- information portals,
- data replication,
- shared business functionality,
- service-oriented architecture,
- distributed business logic,
- B2B (Business to Business) integration.

Information portals allow an access to all required for this purpose functions from a single location. The simplest of them operates on the principle of "tiles" which dividing the screen display the functions of individual systems or subsystems.

Many information systems require an access to the same data sets. Most of systems store the necessary data in their own databases. In a situation where the same data is required by many systems, each of them must have their independent copy. In case of change of the part of common data, the data in all dependent systems must be updated to maintain the consistency. The solution to such problems is the integration based on *data replication*.

Shared logic allows, depending on the requirements and control that we have over the system, a significant reduction of redundancy both data and business logic itself.

Service oriented architecture blurs the boundaries between integration and fragmentation of business logic of information systems. New business functionalities can be created through the use of ready-made functions or components, supplied from existing systems in the form of extracted services.

One of the key elements of system integration is the fact that business functionality is often based on many different systems. In most cases, all functions necessary for its course are contained in existing applications. Element missing in this case is the coordination of these applications/functions.

However, in many cases the business functionalities are also used by other companies cooperating with the company (suppliers or business partners), while talking about the bilateral dependence. These and similar incidents are defined as *B2B (Business to Business) integration*.

2.2. Integration methods of information systems

There are many criteria to be considered when designing and developing the integration solutions on information systems and applications: dependences between applications, interference with the existing implementation, coherence and data format, type of communication, data transport and security.

There is no universal method that would include and cope equally well with all aspects/criteria of integration. A multitude of solutions has led the authors [4] to separate four main groups of integration methods:

- methods based on exchange of data files,
- methods based on shared databases,
- methods based on remote procedure call (RPC),
- methods based on exchange of messages.

In integration based on exchange of data files one party (the application) produces the file containing the information required by the other party (another application). The file is stored in a format supported by the application that created it. For other applications able to read the file, it is required to bring it to a comprehensible format. Therefore, before the data is sent to the waiting applications, the data is transferred to the module or separate integration software, whose task is to convert data to a form understandable to all applications. After processing of file to the 'standard' format it can be read by other applications.

The basic mechanism of methods based on sharing databases is the central data storehouse available for all applications, making it possible to share the required data. A requirement here is handling the database for each application. If any application does not support by default a shared database, it is necessary to use an appropriate adapter.

Remote procedure call, RPC is a mechanism for synchronous remote calls, where one of the programs (client) calls the second (remote) program (server) [5]. The basic assumption of RPC is providing by server the procedures defined by the *Interface Definition Language, IDL*. Based on the created definitions the code fragments are generated respectively for the customer to allow a remote procedure call, and for the server that are required to implement the remote procedure.

The *methods based on exchange of messages* allow the integration of applications belonging to the systems of a company or companies, through the exchange of information in the form of messages. Systems allowing this type of communication are often determined as *MOM* (Message-Oriented Middleware) [6]. In the context of *MOM*, the message is understood as an independent package of business data supplemented by the corresponding headers containing primarily the data about routing of messages and often other useful information.

In order to ensure the integration between systems in enterprises, a lot of products using the exchange of messages were created. Both the older ones, such as IBM MQSeries, Microsoft MSMQ, TIBCO Rendezvous or Open Horizon Ambrosia and newer, such as SonicMQ or FioranoMQ were created in order to provide reliable and efficient connection of company systems through a network.

3. Service-Oriented Architecture

Service-Oriented Architecture, SOA is an architectural style designed for the construction of information systems based on loosely coupled, granular and autonomous components, called *services*. Services are provided by *service providers*, through registration in the *service registries*. Each service describes its processes and "behaviour" by so-called *contracts* that are used by external components, called *service consumers* or *service users* that execute the services found by service registry [8, 9].

In order the system is consistent with the idea of SOA, it should adhere to the following principles [9]:

- contracts of all services should be subordinated to a common standard,
- services should not be dependent on the surrounding environment (including from each other)
- contract should contain only essential information and the services should be defined only by their contracts,
- services should be independent of the use context and usable in many different contexts,

- services should be autonomous, i.e. their runtime environment should not be shared with other services,
- services should be autonomous, i.e. they should not keep the communication status and the consumed resources should only grow in case of calling them,
- services should enable the effective location in the network via so-called *service discovery*,
- services should allow for easy composition, i.e. creation of new systems and services based on existing ones.

The basic elements of SOA are the services with contracts describing them. In the model representing SOA from the technical side, usually there are three main elements: *vendors*, *consumers* and *services register* as well as three operations performed by them: *registration*, *discovery* and *calling* of services [7, 11] (Figure 1).

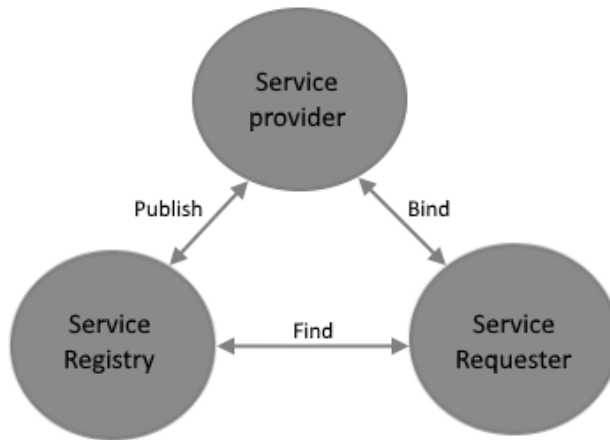


Figure 1. Schema of application functioning based on SOA with its elements

Service Component Architecture, *SCA* is a set of specifications describing a model for building the applications or systems based on SOA [12, 13]. *SCA* extends and complements the other realization modes of services, creating an open standard.

The *micro-services* are an architectural pattern that is the "light" subset of SOA. It was created in order to avoid the problems associated with implementation and deployment of monolithic systems and applications. In assumptions it has to draw the best ideas from SOA, while being deprived of its main

drawbacks, which are the costly, complicated implementation and amount of time needed to implement the architecture [14].

Figure 2 presents certain assumptions arising from the use of micro-services pattern. The main idea is the division into business modules, not the technical ones – the components of monolithic architecture presented on the left side of the figure map the application layers from the technical point of view.

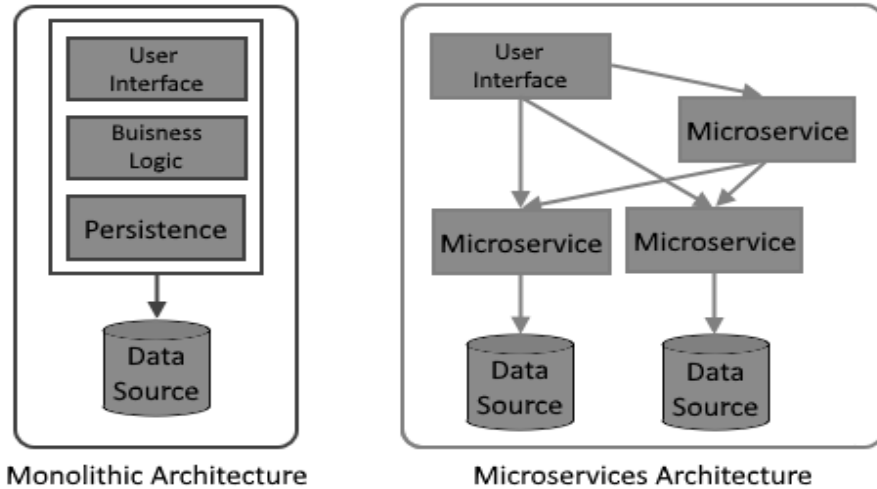


Figure 2. Differences between monolithic application and approach using micro-services

4. Model of Information Systems Integration Based on Micro-services Architecture

To solve the problems of information systems integration presented above the architecture of integration solution, consistent with the principles of service-oriented architecture, was developed. It uses the micro-services pattern to integrate the communication between the heterogeneous information systems. The proposed solution is a method based on exchange of messages.

In order to realize the integration with the developed solution, two pieces of heterogeneous systems were modelled in the form of models of independent applications, which were then extended with integration modules, implemented in accordance with the proposed solution, enabling the exchange

of data between applications, while maintaining the established requirements and assumptions.

The resulting models of systems were called *System Model A* and *System Model B*, and the whole developed integration solution were determined as *Microserviced Integration Broker, MIB*.

The main assumption, required to explore the possibility of using the proposed solution, is the heterogeneity of prepared information systems models. In this case, it is based primarily on the absence of direct relationships between the mock-ups, i.e. the lack of sharing by them any data types or formats, or implementation fragments. In addition, both models provide different communication interfaces and for further stress of the heterogeneity they were implemented in different programming languages.

Another assumption concerning the solution implementation is the fact that the modelled systems exist in the same local network. It can be therefore assumed that the part of the company structure was only modelled. It leads to the deployment of integration solution locally bypassing the security aspects related to for example the using of Internet as the transmission medium (Figure 3).

The basic functional requirement of *System Model B* is to enable the operator to call from the user interface level the method using the data of integrated *System Model A*. As a result of realization of such call the data received by *MIB* should be stored into internal database of *System Model B* and then presented to the operator.

The functional requirement of *System Model A* is listening for connections incoming on a specific communication interface and the service of commands received from connected client applications, whose format and content are consistent with the commands implemented in the system. In response to the command, the system should read the data from its internal database and then return it in certain form as a response to the query sender.

The most important functional requirements from the point of view of this solution are those relating to *MIB*. The basic requirement of *MIB* is to receive the requests incoming from the sender (*System Model B*) and forwarding them to the recipient (*System Model A*). Since *MIB* is a distributed system consisting of several micro-services, it should have the possibility to explore their effectiveness. In addition, an important requirement implemented by *MIB*

is the translation of data passing between the systems as appropriate for their types and formats.

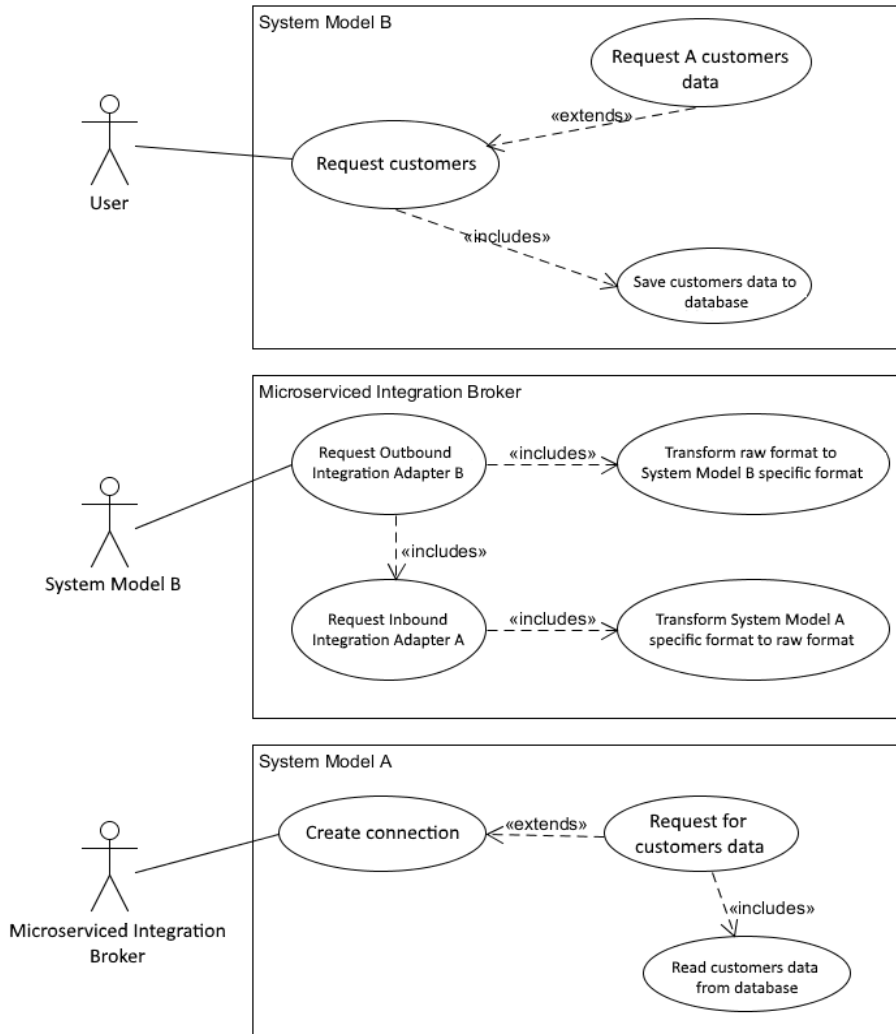


Figure 3. Basic functionality of integration solution model

Apart from the above functional requirements, also some non-functional requirements were determined. The first one is to provide the integration of System Model A with System Model B without any modifications of System Model A implementations, which would be related to integration of these systems.

The second requirement is a distributed solution architecture of MIB, which enables an easy replacement of individual modules and the extension of integration of new systems with a minimum of changes to the implemented solutions. In addition, MIB should implement the common data type for the certain functionality, which allows the independent changes in native data types, derived from System Model A and System Model B, and also an easy use of the same data in the next integrated systems.

5. Architecture of Information Systems Integration Model

Integration solution *MIB* was based entirely on the Java Enterprise Edition platform. All micro-services included in MIB and their registers were implemented with the use of number of modules integrated into *Spring Boot*. Models of *System Model A* and *System Model B* were made as separate applications. System Model A is a simple application made in client-server architecture in PHP language. System Model B like MIB solution was implemented in Spring Boot. To communicate with the end user the *REST web services* were used. The architecture of developed solution is presented in Figure 4.

System Model A was developed as server part of *client-server* architecture. The application listens the incoming calls on a specific TCP port. When a customer calls (e.g. by telnet) the server waits for a message incoming from the client. Incoming message is validated on the server side. Validation is to determine whether the received message is supported by the server command. If the server receives a supported command, it starts the execution of the related business logic. Otherwise, the server responds returning the appropriate message about unrecognised or unsupported command, and then returns to listening. For the needs of MIB solution, the System Model A supports the command, which business logic is to connect to a database, execute the query and then return the result to the client.

System Model B is a simple application that provides REST service using the MIB solution to retrieve the data from System Model A and then recording it in database embedded in memory and returning the stored data to the author of the request. Moreover, the application has a built-in client allowing the execution of requests from the web browser level.

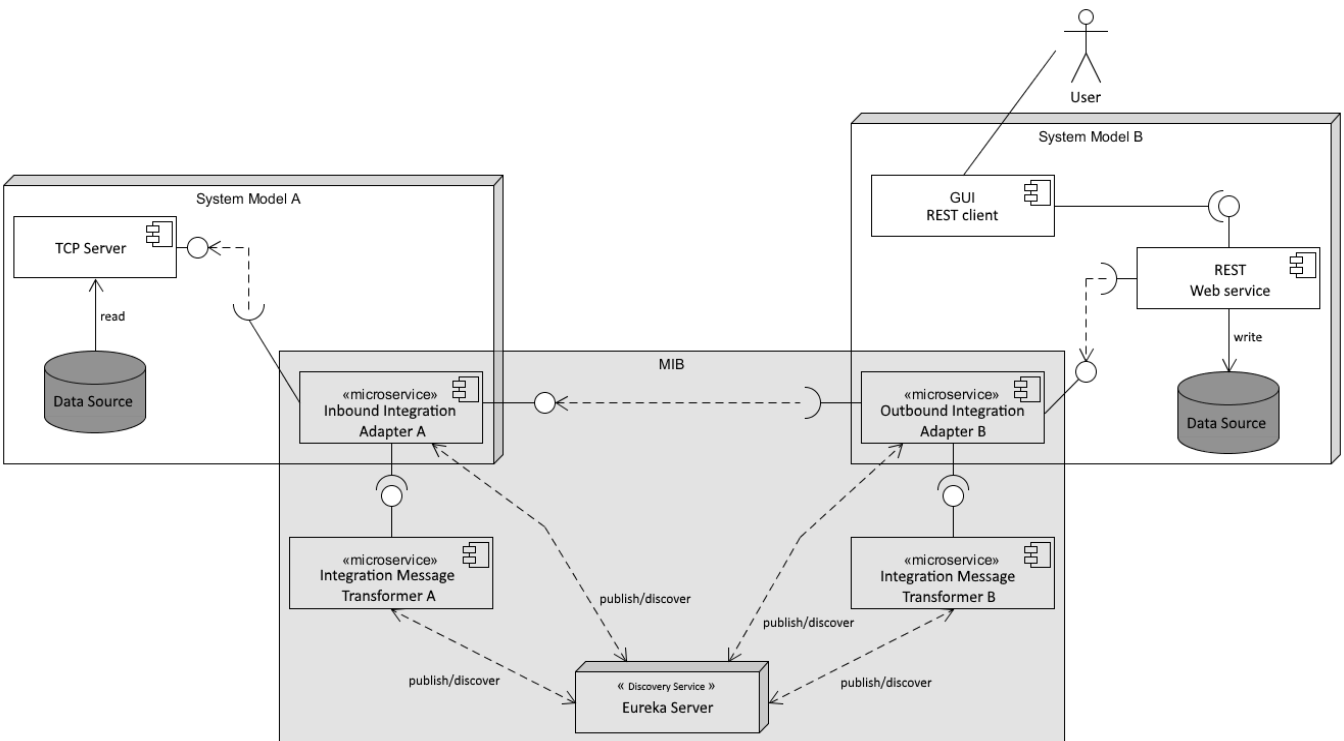


Figure 4. Architecture of integration solution MIB with integrated models of systems

The developed integration solution is used to communicate the System Model B with System Model A. Because it works as the intermediary between systems designed in micro-services architecture, it was called *Micro-service Integration Broker, MIB*.

Micro-service Integration Broker was implemented in accordance with the concept of *service-oriented architecture*. However, in contrast to the classical approach, it is devoid of its main drawbacks such as increased complexity of the system, poor resistance to crashes or difficult testability. Use of micro-services makes that the whole MIB solution consists of five autonomous micro-services that do not affect the existing part of the system and they can be managed, changed or tested individually. The resiliency is ensured by the register and the so-called *load balancer* that monitors and manages the instances and the redirections to particular micro-services.

MIB may consist of a different number of modules depending on how much systems have to communicate with each other. In the framework of communication between *System Model A* and *System Model B* four micro-services were implemented recording to a single services registry. It is possible to distinguish two types of micro-services in MIB: input/output adapters and data/messages transformers. As the models of systems A and B, have been created with the assumption that *System Model B* will request the data from *System Model A*, two adapters were prepared respectively: *micro-serviced outbound integration adapter of system B, MOIA-B* and *micro-serviced inbound integration adapter, MIIA-A*. In addition, two transformers have been implemented: *micro-serviced integration message transformer, MIMT*, respectively for system A and for system B. The last module of MIB is *Eureka server*, developed by Netflix micro-service, which is a services register providing: discovery of services, load balancing and service of failures. The main task of Eureka service is to monitor the services registered in it and to provide its location based on the names under which they were recorded.

MIB is the most important element of implementing the proposed integration model. It consists of five applications (written in Java language using Spring Boot) that, as five independent micro-services, together form the integration solution, used to communicate *System Model A* with *System Model B*. The central point of MIB is micro-service "Eureka Server" acting as a registry of other micro-services.

Next micro-service within MIB is *Microserviced Outbound Integration Adapter*, *MOIA* cooperating directly with System Model B. Activity of this adapter was focused in controller mapping the functionality made available by *adapter of incoming messages* of System Model A. It is based on sending the request received from System Model B to the adapter of System Model A, found in the registry. Then the received message is translated on the format compatible with System Model B using the transformer found in the registry.

A request that goes from *MOIA* adapter to the adapter of incoming messages of System Model A (*Micro-serviced Inbound Integration Adapter*, *MIIA*) is supported by the controller, whose logic is to connect through protocol (native to System Model A), sending commands with parameters received in the request and then the transformation of the response to general form and returning it to *MOIA* adapter.

To make the data extracted from one system be understood for the second one, it is necessary to translate them. It is implemented using two micro-services, acting as transformers. Using a common data format allowed for implementation of transformers in such a way that they are not aware of each other – it makes data format of System Model A independent on data format of System Model B.

The integration of communication using *MIB* proceeds as presented in Figure 5. The user sends via *SwaggerUI* interface a request to *System Model B*. In the framework of service of user's request, System Model B through its own REST client sends a request to *B output adapter*.

B output adapter maps all of the functionalities offered by other input adapters. Thanks to it, *System Model B* may send a request in the form identical, which will reach the target system, differing only in the recipient's address.

B output adapter within the service of received request, send it to *A input adapter* associated with a given functionality, using to it the Eureka client oriented on Eureka register. Thanks to it *B output adapter* does not to know the address of input adapter – just it knows the name of the system, which the request has to reach. On the basis of the system name and the prefix identifying the input adapters, the running instance of *A input adapter* is found in Eureka register.

When a request sent from *B output adapter* reaches the *A input adapter*, the operation logic associated with its service is performed. *A input adapter* connects to System Model A, sending the command native for A, by communication protocol, supported by System Model A. *System Model A* returns the data associated with this command to the sender, i.e. to *A input adapter*.

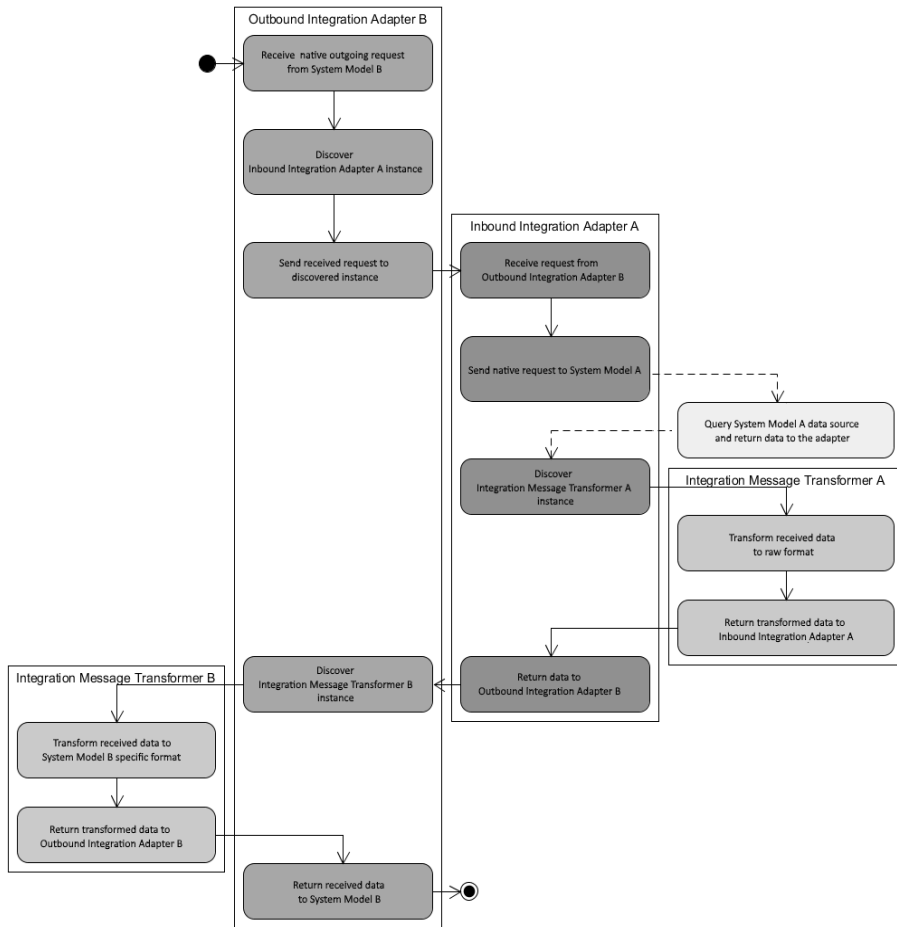


Figure 5. Schema of MIB system working on the example of data transmitted between System Model A and System Model B

After receiving the data in a format compatible with System Model A, the A input adapter searches for the instances of A messages transformer in Eureka register.

Transformers are the only elements of the system, knowing the formats supported by the systems. A *input adapter* sends the data obtained from System Model A in order to transform them to the agreed general format. The data in general format returns back to A input adapter where they are immediately sent back to direct request sender, i.e. to B output adapter.

Finally, *B input adapter* performs the analogous reference to the relevant transformer, that translates the data from general format to format specific for *System Model B*. After returning the data by transformer, they are immediately returned by the *B output adapter* to the original author of request, i.e. to *System Model B*.

The use of general format well-known for all transformers allows the independent modification of data specific for the system without the necessity of modifications of all transformers, translating the data to and from the format of the system.

6. Conclusions

Service-oriented architecture has introduced the new possibilities for communication between heterogeneous systems or applications, largely due to the popularization of network services. Micro-services, as an architecture derived from SOA, but fully consistent with it, try to solve the typical problems of pure service-oriented architecture. Therefore, this architectural pattern as well as the technologies strongly associated with it, has been selected for the main objective of presented work.

The described assumptions and requirements fully address the problems associated with SOA, so that the proposed solution minimizes their occurrence. Designed solution architecture allowed performing the optimal implementation, and the presented models of information systems have enabled the mapping of conditions of solution deployment for existing heterogeneous systems.

Conducted simulation tests showed the correct operation of implemented integration model, the use of which has enabled seamless communication between prepared mock-ups of system.

The use of micro-services is very flexible and universal approach for integration of information systems. Thanks to its flexibility, the created integration solution can be further developed. It is scalable and can be used for the integration of more than two systems. The potential directions of development should focus first on expanding and optimizing of existing modules, for example by more universal system of message translations or by developing of more universal way of functionalities mapping between various adapters of incoming and outgoing communication. Moreover, the security aspect and the ability to customize the integration solution prepared to cooperate with more systems represent the future direction of presented works.

There is probably no software architecture without drawbacks. There is also no universal approach that will work in each information system. SOA is not an ideal creation, applicable in every situation. However, its main advantages are:

- thanks to well definition and separation of elements, the software can be easily modified – flexibility of architecture allows to quickly add new *services*,
- good design and implementation of applications/systems in accordance with SOA principles allows to reuse the services in other systems or projects,
- SOA provides the separation of logic, enabling the developers to work independently on services, not blocking to each other,
- due to SOA, the creation of new functionalities does not affect the other functions, and if the contract was not affected, the customer does not have to make any changes to his system,
- through standardized communication of services in SOA, the easiest integration of services and applications is possible,
- due to high pressure on the business aspects, SOA contributes to the creation of software better meets the user requirements.
- SOA is platform independent and it does not exclude the cooperation between applications written in different programming languages.

References

- [1] M. P. Papazoglou and W. J. Heuvel. Service oriented architectures: approaches, technologies and research issues, *VLDB Journal*, 16 (3), 389–415, 2007.
- [2] G. Alonso, F. Casati, H. Kuno and V. Machiraju. *Web Services – Concepts, Architectures and Applications*, 2004.
- [3] H. Paulheim, *Ontology-based Application Integration*, Springer Science+Buisness Media, LLC, 2011.
- [4] G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley Longman Publishing, 2004.
- [5] C. Bussler. *B2B Integration: Concepts and Architecture*, Springer-Verlag Berlin Heidelberg, 2003.
- [6] R. Monson-Haefel and D. A. Chappell. *Java Message Service, Second Edition*, O’Reilly, 2009.
- [7] X. Pan, W. Pan and X. Cong. SOA-based Enterprise Application Integration, *proc. of 2nd International Conference on Computer Engineering and Technology*, 7, 564–568, 2010.
- [8] A. Rotem-Gal-O. *SOA Patterns*, Manning Publications Co, 2012.
- [9] T. Erl. *SOA Design Patterns*, SOA Systems Inc., 2009.
- [10] T. Erl. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, 2005.
- [11] S. Kumari and S. Kumar Rath. Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration, *Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1656–1660, 2015.
- [12] A. Karmarkar and M. Edwards. *Assembly of Business Systems Using Service Component Architecture*, Springer–Verlag Bering Heidelberg, 529–539, 2006.
- [13] D. Du, J. Liu and H. Cao. A Rigorous Model of Contract-based Service Component Architecture, *Proc. of International Conference on Computer Science and Software Engineering*, 409–412, 2008.
- [14] M. Villamizar et all. Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud, *IEEE*, 583–590, 2015.
- [15] L. Chomatek and A. Poniszewska-Maranda. Multi-agent systems in SOA architecture, *Proc. of 16th Conference Networks and Information Systems*, 2008.

Chapter 3

The Use of Web Services Architecture for Engineering Calculations Based on the webMES Platform

1. Introduction

The increase in computing power and the evolution of numerical algorithms (approximating the solution of a physical problem) influenced the development of engineering calculations. Physical phenomena, described by differential equations, with given boundary conditions may be simulated by means of computer calculations.

The analytical solution of these equations is in many cases difficult or even impossible. Therefore, the researchers use numerical methods, using transformed differential equations (and other data about the problem, such as material properties and boundary conditions) to create a system of linear equations. In case of large problems, when the entire region is divided into a number of discrete components, the system of equations may consist of millions of unknowns (the searched physical values) that can be calculated only by a digital machine [1].

The calculation of the abovementioned large systems of equations and the use of other methods to solve a given physical problem consume a part of resources of the computing unit. In addition, the implementation of the simulator involves writing a code in one of programming languages, which requires knowledge and experience in this field. On the other hand, the use of customized engineering software, as a module in a larger system, is associated with the development of an interface between the individual modules. All these elements mean that there may be a need for delegating these tasks to

a separate, perhaps remote software, which will be able to communicate with any system because of the universal interface.

An important aspect during the tests in numerical experiments is the ability to monitor the calculations. Computing performance of modern personal computers allows performing simulations in time, in which a given calculating unit is not used for other tasks – for example at night. The remote monitoring of calculations seems to be interesting when it comes to the use of web services. It allows to detect the interruption of calculations by external factors (e.g. a computer failure), which will later contribute to the prevention of similar occurrences. The following paper presents a simple client program that allows checking the status of calculations.

1.1. Web services

One of the possible architectures to be used in the design and implementation of the system is a service-oriented architecture. The system, divided into service, can be treated as a collection of objects that communicate with other objects via the interface.

This concept, which is close to the concept of interface as a “contract” [2], allows for the multiple use of one module in a single system, as shown in Figure 1.

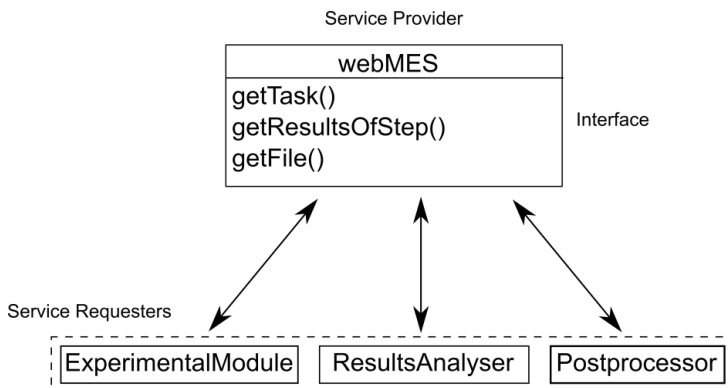


Figure 1. The division of the information system as a set of services. The webMES module provides an interface for modules ExperimentalModule, ResultsAnalyser and Postprocessor

The services, mentioned above as “objects”, are not necessarily objects included within the local system. They can be treated as distributed objects that can communicate with a remote computing unit, taking only the output data. The implementation and technology is completely indifferent, while the important things are data transmitted within a given interface. The article describes the use of web services – services whose communication medium is the Internet.

1.2. Literature review

Following a literature review, the authors could list several works that have tried to implement similar functions. The work [3] proposed a complete web server in which pre- and post-processing are performed on the website, while the calculations are performed by the computing server (back-end). Unfortunately, the results are not available via web services, and hence – the integration of its software with a calculation module is not possible without the inconvenience of page parsing.

On the other hand, the article [4] describes a web application form of Vine Toolkit which is useful in the construction of scientific portals that provide an integrated set of tools, applications and data (science gateway). This software, by providing an extensible, unified and modular application programming interface, allows access to grid technologies and disk resources. Vine Toolkit also enables the integration of Adobe Flex and BlazeDS, which allow creating powerful web applications.

The paper [5] presents an interesting concept of using web services to generate finite element meshes. The main emphasis was placed on the refinement of services, so that they could be widely used, e.g. in mobile applications.

The abovementioned works have contributed to the creation of a project and prototype of web services, which was named webMES. The efforts were focused on the creation of appropriate data structures and adapting the used technology to work with engineering software. The work [6] presented mainly theoretical issues associated with the performance of numerical simulations and a physical project of the system, including the description of classes.

1.3. Purposes

The purpose of this work is to create a server of web services, working with existing engineering software. The software presented in the subsequent part of the article differs in terms of requirements and programming dependencies. The user must provide an appropriate environment and libraries. The concept of web services allows providing a unified interface, regardless of the employed simulation software (solver). Additional benefits will be presented in Section 5.

One of the problems, outlined in Section 3, is to use the solvers that were not designed to work with web services. Their communication with the user comes down to generating output files. This hinders the interaction between applications (the server of web services and solver). This Section proposed a solution to this problem without having to modify the source code of the proposed engineering software.

Section 4 presented a client program, which is the first tool that works with the webMES system.

2. Engineering Software

The examined engineering software (abbreviated to ES) will be treated as software, having the following characteristics:

- it is run by a single program on a system with a console shell;
- the user enters input data (e.g. the file names) using command-line arguments;
- the program saves the results to the output files;
- each file represents one time step (which narrows the studied phenomenon to the initial value problems).

Consequently, the works do not include large simulation programs (e.g. AutoCAD) that display a real-time simulations. The web services presented in the subsequent part of the article use software that operates according to the above characteristics – which converge all of the software to the batch software architecture [7].

2.1. Used software

The concept of the abovementioned engineering software was initiated after analysing the NuscaS software [8]. NuscaS is a program for engineering calculations on the basis of the finite element method. In order to operate it requires data on the mesh (the original format `mesh`, used by the built-in generator of meshes) and task. The task is represented by a series of files:

- `bc` – description of the boundary conditions,
- `ic` – description of initial conditions,
- `m` – material properties,
- `cp` – task properties (time step, number of steps, etc.).

The launch of NuscaS with the above data results in the generation of output files in `rlt` format. This is the file in which each node (identified by a number) is assigned to the value of the desired variable (or variables).

The work on theoretical issues of the finite element method and the analysis of heat conduction resulted in the creation of jMES. jMES is a heat conduction simulator, which uses the same format of the input data as NuscaS. However, the output data format is different. JSON format was picked because of the availability of parsers in different programming languages, making it easy to upload data to other system.

The authors are also designing and implementing a different simulation system, using the TalyFEM library [9]. This library provides a set of classes that make it easier to create software using the finite element method. Applications developed with the use of this library can run faster than other programs due to the scalability and parallelism. One of the tasks during the work on webMES will be the integration of web services with the software developed on the basis of TalyFEM.

3. Design and Implementation of Web Services

The web services described in this section are implemented using the concept of REST API [10]. This concept allows treating services as a set of API (Application Programming Interface) to communicate with engineering software. The HTTP technology was used as the data exchange protocol, and

the JSON format as the language of messages. The subsection 3.2 presented the employed technology for creating web services. The presented concepts can be applied using a variety of technologies, oriented on programming of web applications (e.g. Java EE or .NET), which would therefore contribute to the better description of the problems associated with the combination of available engineering software and the server of web services.

One of the problems was sending messages from the software to the server of web services. The existing and employed programs are not designed to send messages, they do not allow access to their own mechanisms and interfaces to inform about their status.

3.1. Integration with the existing ES

Two possible ways of communication between software and web services were specified during the analysis:

- the code of engineering software informs the server of web services about its status (Figure 2a);
- the server of web services based on the analysis of engineering software sets its status (Figure 2b).

It appears that the first method, from the perspective of software design and software engineering, is better than the other for future expansion. If one connected it to the possibility of communication between engineering software (available through its API) and the server of web services, the simulator itself can be regarded as a service, in which a separate server of web services acts as a proxy between the client and engineering software.

Unfortunately, the programs prepared and employed to simulate physical phenomena do not have the ability to inform about their status with their modules (written using, for example, the observer pattern [11]), nor do they give the possibility of polling by the remote software using the prepared API. While in the case of jMES and TalyFEM adding communication with separate modules is a matter of supplementing their projects, it is not possible in the case of NuscaS, as the project is no longer being developed. From the beginning, NuscaS was envisioned to be a single station program, which does not have any ability to communicate with the outside world.

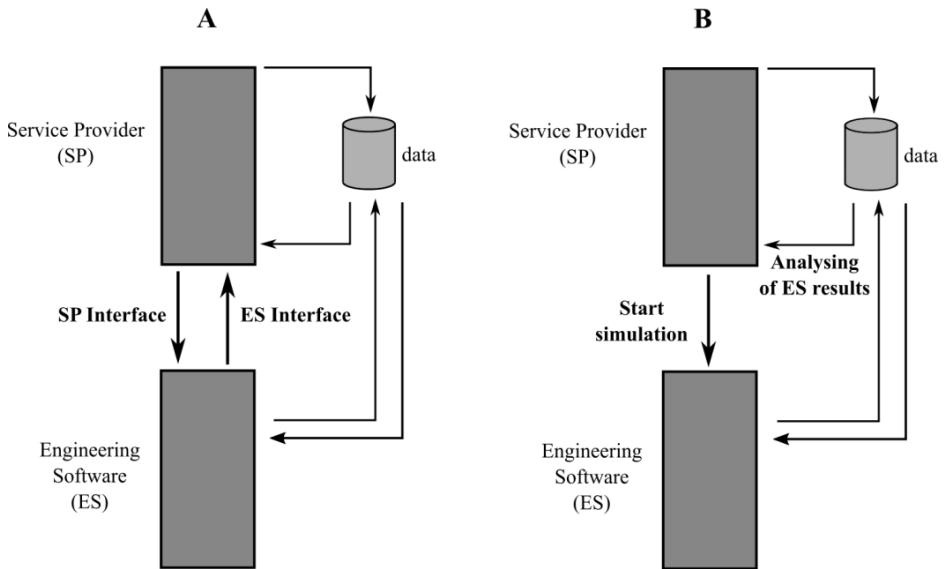


Figure 2a and 2b. The concepts of communication between engineering software and the server of web services
 a) with the use of API simulator
 b) without the use of API simulator (analysis results)

The behaviour that can be used to monitor the calculations was observed when analysing the operation of the above software. Each of the solvers after calculating a time step can save the result into a file. Knowing the number of steps (due to the file with data on the task), one can assess the percentage of completed tasks (CT) using the following formula:

$$CT = \frac{OF}{NS} * 100\%, \quad (1)$$

where OF is the number of output files, and NS is the number of all the steps. The appointment of CT allows monitoring the task. To calculate the number of files that meet certain criteria for the name (each task generates output files whose names contain the appropriate prefix) one is required to use libraries to manage file systems. Such libraries have virtually all popular programming languages. In this way, one of the possibilities of the described system – monitoring calculations – can be done without interfering in the code of ES.

The next task – downloading the results – can be accomplished in a very similar way, without the need to perform any software modifications. If the calculation of each time step results in recognition of an appropriate file, then it is enough to return the file as a result of the calculation. The output files contain an appropriate suffix, which is the step number. To manage this task it is necessary to have a library to operate the file system. It should not be problematic to load and return the file content.

It should be noted that returning the file content in the form of HTTP response is not entirely optimal, and in the case of larger meshes it will not be possible (due to expiring session – time-out). Nevertheless, this method, in which the file content is perceived as a check box in JSON, is only a temporary solution.

3.2. Used technologies

Web services need tools that enable to create a program which allows reacting in a deterministic way to incoming requests. Since the HTTP protocol is based on the transmission of character strings, the whole is possible to implement in any programming language and any technology that enables listening on the appropriate port and sending a reply to the remote client. From the perspective of web application development it seems that the use of one of the dedicated web technologies will create appropriate network services in the shortest time and with integrated solutions ensuring high-quality software.

Due to the authors' experience, the server software was written in Python using the Django framework [12]. Django allows to create a web application, using the built-in object-relational mapper (and database support), the system that generates content based on templates or automatically-generated admin panel. This particular panel is used to run tasks. The main task of any programmer is to define models (classes) and views (functions that return the appropriate HTTP request). Due to the fact that the web services, in the current stage of development, do not require generating websites, it was not necessary to implement templates.

Each view should return a `Response` object. The `HTTPResponse` object, which can return the appropriately generated response with the use of HTML, is the most frequently used object. If the only returned response is in

JSON format, there is a need to manually set the appropriate type of resource and add a return information. The creators of Django simplified the return of responses in the appropriate format, providing a `JSONResponse` object, requiring only a dictionary object (`dict` from Python). The object itself and the framework set the appropriate type of response, and other metadata, while the dictionary is converted to JSON format. This allowed for a significant acceleration of works on services, hiding metadata of the HTTP protocol (Figure 3).

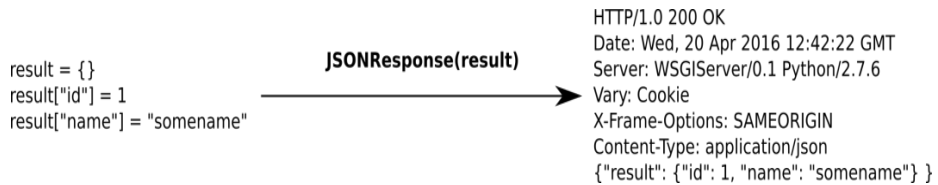


Figure 3. The use of a `JSONResponse` object, turning the dictionary object in the HTTP response with the appropriate object in JSON format

3.3. System layers

In the current version of the system, running an engineering simulation can be represented as a sequence of actions, initiated by the admin panel. The user must activate the appropriate check box to run the simulation. It should be emphasised that this is a temporary solution, easy to implement, and the launch of calculations is possible in the future by the appropriate web services.

Launching the simulation begins with the creation of a thread (`NscThread`) using the Python interpreter and notifying the system (database) about any change in the calculation. Then, the process is launched through the interaction with a system layer (`subprocess` module) – the executable file of an engineering application with appropriate arguments. As described previously, the engineering application does not inform about its status, but the process itself will finish its operation when the engineering application is fully executed. Full information is again entered into the database, ending the `NscThread` thread. The results are stored as files, which can be accessed through the web services described in the next section. The entire scheme of calculations, including the layers, technologies and Python modules is shown in Figure 4.

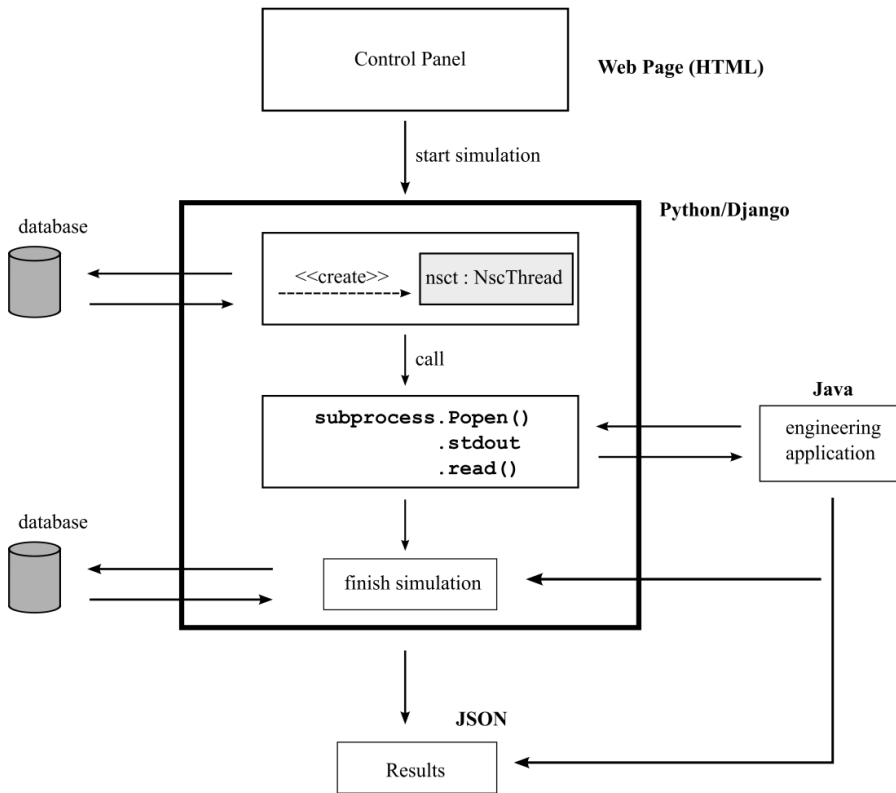


Figure 4. Layers of the calculation module of the webMES system

4. Available Services

The web services operating within the described system are services that do not change the system status or influence the formation of new calculations. These services are “read only” and take the results of all calculations. The creation of new tasks is not currently possible using only web services. A temporary solution is to use an admin panel to place tasks on the server. In the future, it is envisaged to use appropriate requests, along with the development of a client application.

At the time of work, webMES provides the following web services:

- `get_tasks` (returning information about all tasks),
- `get_computed_steps` (the completion level of tasks),

- `get_computed_step` (results in a given step),
- `get_task_properties` (task properties loaded from the cp file),
- `get_files` (information about identifiers and types of all uploaded files),
- `get_file_content` (returning the file content).

4.1. Arguments of web services

The GET method is used to transmit the parameters of requests. It was decided to use it because of the relatively small number of data transferred, the ease of testing and the use of this method to the controller handling requests in Django. The parameters can be placed in a URL address, which does not use the query string. Instead, it uses a semantic URL.

Using the query string one can provide an argument name and its value, which are appended to the resource identifier, such as e.g.:

```
http://page/tasks?task_id=1&step=20
```

The whole request (address) is processed by the data controller of web services. Therefore, it is possible to modify the controller by the independently analysed HTTP request. In Django it is achieved through the URL Dispatcher that consists of a list of supported URLs, and corresponding views. Using the semantic URL, such a request can be written as:

```
http://page/tasks/1/20
```

and the URL dispatcher triggers an appropriate method, including the transfer of appropriate arguments to the view.

4.2. Implementation of services

The implementation of services comes down to writing appropriate methods (views) that return responses. Django allows, as mentioned in Section 3, to return a dictionary object in JSON format as a response to a request. Therefore, the task of the programmer is to create an appropriate dictionary object, which will be returned as JSON.

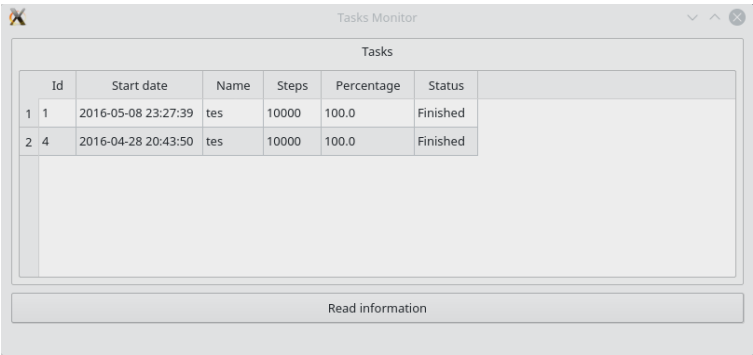
The currently used solution is the manual creation of a dictionary based on the available data. The models, whose content is returned in web services,

have a defined `toJSON` method that creates a dictionary (not a JSON object) and returns it.

This is not the target solution. Each code refactoring of models (changing fields, adding new models etc.) makes it necessary to manually modify the appropriate methods, generating dictionaries based on the data. In addition, it is required to monitor the controller to include the added methods in the URL Dispatcher module. The solution used in the future will be the use of library, generating basic web services based on the content of models – it can be compared to the use of serialization in Java. One of the proposed libraries is REST Django, which is intended to be implemented in the system. Some web services (which require calculations and operations on files, such as returning the status of calculations) cannot be automatically generated and there is still a need for manual management.

4.3. Communication with webMES

A client software was developed to monitor the running tasks using the system. It was written with the use of Python and PyQt (application form), making it possible to check the status of calculations on the most popular operating systems, including Windows and Linux.



	Id	Start date	Name	Steps	Percentage	Status
1	1	2016-05-08 23:27:39	tes	10000	100.0	Finished
2	4	2016-04-28 20:43:50	tes	10000	100.0	Finished

Figure 5. The client of webMES

The application, based on the JSON files provided by services, fills in all information about the running tasks (Figure 5). This way one can obtain data about the task name, the time of its commencement and status

(completed, running, stopped), the number of steps concerning the performed calculations and percentage of completed tasks (CT).

5. An Example of How the System Works

A simple example of the initial problem of heat flow was used to present the results of the webMES system. The two-dimensional area in the size of 2×2 is divided into triangular finite elements. One side contains a boundary condition of the third type (heat exchange with the surroundings), and the opposite side contains a boundary condition of the second type (constant heat flux). Such a problem can simulate the process in which certain body is heated from “below”, and the top exchanges heat with the environment.

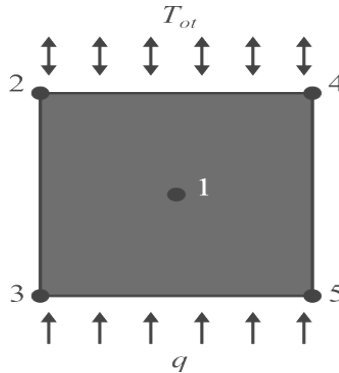


Figure 6. The examined task area. The q stream works from the bottom, the heat exchange with the environment occurs from the top. The selected nodes: 1 (the centre), 2, 3, 4, 5 (the corners)

5.1. The input data and results

The jMES software uses input files that are compatible with NuscaS. Grid files, initial conditions, boundary conditions, material properties and task properties were created. The time step was set to 0.01 s, the number of steps to 1000. The starting temperature was 400K, the ambient temperature 300K. The remaining properties of the task do not affect the duration of web services, and were therefore omitted in the description.

The results, obtained in the JSON format for the 1000th step (after 10 seconds in the simulation), can be obtained using the following address:

```
http://page/tasks/4/1000
```

These results (the number of dimensions, the number of nodes, etc. were omitted for readability) are as follows:

```
{
  "errors": "",
  "result": {
    "vals": [
      [...]
      {
        "id": 1,
        "val": 400.0
      },
      {
        "id": 2,
        "val": 300.0182570430909
      },
      {
        "id": 3,
        "val": 411.4285714285529
      },
      {
        "id": 4,
        "val": 300.0182570430909
      },
      {
        "id": 5,
        "val": 411.4285714285529
      }
    ],
    "desc": ""
  }
}
```

The temperature in the 3rd and 5th node increased (the heat flow of low power), and it decreased in the 2nd and 4th node (very high heat transfer coefficient). The further work will optimize the results. Currently, the model of output file concerning the NuscaS system, the resulting file contains a lot of redundant information (task data, the numbers of nodes) which can be removed.

The speed of the system was not analysed during calculations. It depends on the calculation time performed on the engineering software. As part

of the further work, using a scalable simulation software, it will be possible to carry out comparative studies between different solvers.

5.2. Benefits of using webMES

The use of the created jMES simulator would require the user to use the proper preparation of a computing environment. The programmer would have to manage the needed software (Java Virtual Machine, computing libraries) and data. WebMES allowed hiding the necessary software components from the user. This is a big advantage if one wishes to use the NuscaS program – its console version runs only on Linux.

The web services allow the programmer-client to delegate tasks to the selected calculation program. This has not been implemented in any work, but after the release of the system it will be possible to use its web services for numerical calculations.

6. Conclusion and Plans for Further Work

Web services are increasingly used as a method for exchanging data between the server and the client (e.g. a mobile application). The use of this concept allows for a remote analysis of engineering calculations, including systems that do not allow access to their API. As shown in the work, it is possible with the use of Django framework and the relevant test behaviour of the software, and in particular the generated output data. A simple client program that communicates with the server was designed and written for the purposes of this study.

In future it is planned to develop the system as a web service. Additionally, the project should be refactorized using the REST Django library to automatically generate CRUD services for individual models. The problems of launching calculations and transmitting input data to the server should also be studied.

An important element of the work will be the analysis and use of scalable engineering simulators. The person responsible for the simulation will be able to choose to run calculations in a multi-core application when defining the

task. The simulators of physical phenomena, with the ability to work in a distributed environment are in the implementation phase. WebMES can be used as a medium to compare the current solutions with the new, scalable simulation systems.

References

- [1] O. C. Zienkiewicz. *Finite Element Method*, McGraw Hill, 1977.
- [2] B. Eckel. *Thinking in Java*. Wydanie IV, Helion, Gliwice, 2006.
- [3] H.-M. Chen, Y.-C Lin. Web-FEM: An internet-based finite-element analysis framework with 3D graphics and parallel computing environment. *Advances in Engineering Software*, 1, pp. 55–68, 2008.
- [4] P. Dziubecki, P. Grabowski, M. Krysiński, T. Kuczyński, K. Kurowski, D. Szejnfeld. Modern Portal Tools And Solutions with Vine Toolkit for Science Gateways. In: *Proceedings of the 3rd International Workshop on Science Gateways for Life Sciences London, United Kingdom*, 2011.
- [5] N. Szczygiol, J. Mikoda, A. Wawszczak. Web service for finite element mesh generator. *Computer Methods in Material Sciences*, 10, pp. 176–180, 2010.
- [6] P. Jeruszka. webMES – projekt usług sieciowych do realizacji symulacji inżynierskich, *Studia Informatica* 119, pp. 169–179, 2015,
- [7] A. S. Tanenbaum, H. Bos. *Systemy operacyjne*. Wydanie IV, Helion, Gliwice, 2015.
- [8] N. Szczygiol, A. Nagórka, G. Szwarc. NuscaS – autorski program komputerowy do modelowania zjawisk termomechanicznych krzepnięcia. *Polska metalurgia w latach 1998–2002*, pp. 243–249, 2002.
- [9] H.K. Kodali, B. Ganapathysubramanian. A computational framework to investigate charge transport in heterogeneous organic photovoltaic devices. *Computer Methods in Applied Mechanics and Engineering*, 247, pp. 113–129, 2012.
- [10] Learn REST: A RESTful Tutorial, <http://www.restapitutorial.com/>, accessed: May 11, 2016.
- [11] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Inżynieria oprogramowania: Wzorce projektowe*, WNT, Warszawa, 2008.
- [12] Django. The web framework for perfectionists with deadlines, <https://www.djangoproject.com/>, accessed: May 11, 2016.

Chapter 4

Designing a Data Warehouse for Changes with Data Vault

1. Introduction

Designing a data warehouse may appear to be quite a simple task, especially at the conceptual level because it's often treated as a database which main purpose is to leverage complex analytical queries efficiency and ability to store and audit historical data. Nowadays there are few popular data warehouse data models and there is a vibrant debate among data warehouse architects and engineers over which model is better. Authors of this paper having a strong commercial experience in delivering data warehouse projects are aware that finding an optimal approach is rather impossible, because every data model has its pros and cons in particular scenarios. During implementations in BPO companies (business process outsourcing) such as call centres frequent changes in data structures, business rules and requirements are triggered with almost every new project. It forced us to look for an appropriate data modelling and processing techniques.

There are lots of works related to data warehouse implementation and design starting with traditional waterfall models with requirements first approach [9], dedicated agile methods proposed by [2] and [5], or quite opposite implementation first approaches as proposed by [3]. Although the development methodology is not a topic of this paper it is important to mention that whatever approach you will choose it's highly possible that changes of different nature may occur during implementation process, such as:

- changes of source data structures;
- business rules modification;

- data updates and deletions;
- source systems migrations;
- ETL jobs modifications.

The paper is organized as follows: Section 2 contains an overview of existing data warehouse models and methodologies. In Section 3 the resistance to changes within popular models will be discussed. Data Vault automation techniques in introduction will be in the scope of Section 4. Last Section is a short conclusion of the topics covered in this paper.

2. Background Introduction to Main Architectures

In this section we will introduce main ideas regarding data warehouse architecture and implementation methodologies. We are going to compare Data Vault to the models and methodologies from different schools we find it important to show the basic differences between them.

2.1. Enterprise Data Warehouse (EDW)

Most of the data warehouse architectures consist of few core layers as presented in Figure 1. Each of them has a different purpose (see Table 1), but the main area of discussion among mentors such as Kimball, Inmon or Linstedt concentrates on Enterprise Data Warehouse layer.

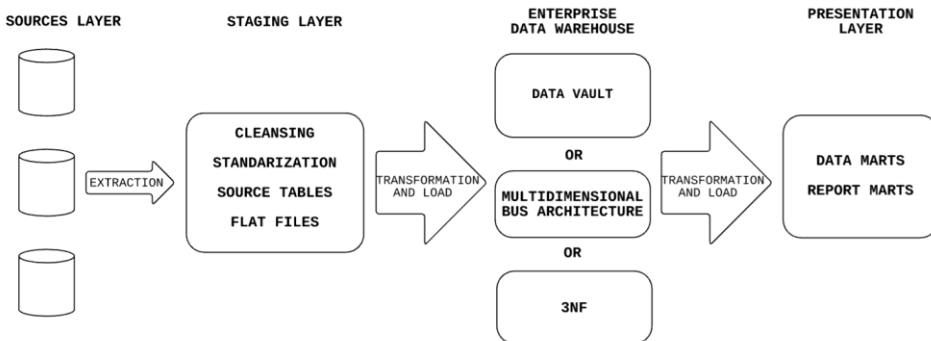


Figure 1. Data warehouse multi-layer architecture

Table 1. Data warehouse layers description

Layer	Description
Sources layer	Conceptual layer representing all required data sources such as relational databases, files, APIs, enterprise service bus and others.
Staging layer	Within this layer raw data from sources is stored for further processing. It can be treated as preparation area where data is processed, cleansed and standardized and not available for user queries.
Enterprise Data Warehouse	Cleansed and integrated data is stored in order to allow creation of specific subject data structures within following presentation layer.
Presentation layer	This is the final layer in which data is prepared for specific usage (e.g. for financial department) usually in multidimensional data marts or recalculated structures called report marts.

There are two main philosophies regarding EDW which mostly differ in terms of end-user access, design and implementation approach which are described briefly in Table 2. The top-down approach recommended by [4, 8] and gives some overhead connected with the need of loading most of the required source data into 3NF form which is time-consuming task and then deriving dimensional data in form of data marts from it, which is quite straightforward as denoted by Kimball [5]. On the other hand Kimball is an advocate of bottom-up approach [6] and uses the concept of Enterprise Bus Architecture where data marts are created and loaded in the area of EDW and then during further implementation they create more and more complement data warehouse by using conformed dimensions. The third approach represented by Linstedts Data Vault (DV) [8] may be treated as the hybrid approach because data is loaded first into EDW layer with lowest available granularity (no aggregates, which are allowed in Kimball's approach) and then loaded into data marts or report marts. Another interesting thing about DV is that it does not have 3NF or dimensional model but unique hub and spoke architecture which will be discussed within next section.

Table 2. EDW approaches description

Approach	User access	Data model/design	Implementation
Top-down	End users do not have access to EDW layer, it is used for storage of consistent and normalized/semi-normalized data.	3NF or Data Vault model.	Data is loaded with original granularity into EDW, from there it is later loaded into specialized data marts.
Bottom-up	EDW layer is connected with Presentation layer. Data marts as a part of a complement data warehouse.	Enterprise Bus Architecture (dimensional model).	Data marts are loaded based on users requirements. Loads of aggregated data is acceptable.

2.2. Bottom-up Enterprise Bus approach

Bus Architecture may be treated as a generalization of dimensional data model. There are two main types of tables, fact and dimension tables. Without going into details fact tables focused on events and measures connected with them and dimension tables gives us more descriptive and context information about entities connected with that fact. With bottom-up approach it is advised to create data marts (facts and dimension tables covering some business area) of business required granularity and with help of conformed dimensions (such as date dimension) these data marts should create complementary data source as implementation proceeds. The resulting data warehouse will be a set of connected data marts which is called Enterprise Bus.

2.3. Top-down Data Vault and CIF approach

Opposite to the bottom-up approach there are two popular top-down approaches. In both of them data is loaded into EDW layer first. It is important to notice that this layer is generally not available for end-users. There are two leading data models within this layer which are described in Table 3.

Data is consumed by analytical applications or query tools after it is loaded into data marts, known from bottom-up approach.

Table 3. Components of 3NF and DV models

Data model	Components
3NF – normalized	There are no distinguished components within this model. Data is stored in the way that minimizes redundancy and required operations during updates and deletes.
Data Vault – denormalized	There are 3 main types of tables: hubs (storing information about entities business keys/natural keys), satellites (tables with descriptive attributes concerning exactly one hub or link table) and links (many to many tables) denoting relations such as transactions, associations, hierarchies between two or more entities (hubs).

3. Handling Changes within EDW Layer

In this section, we compare sample solutions when handling mentioned types of changes in Data Vault 2.0, multidimensional and 3NF data model. For simplification we only focus on changes within Enterprise Data Warehouse area, mentioning influence on the presentation layer.

3.1. Data structure changes

To examine different models behaviour when changes in source data structure appear we define a scenario in which primary key switched from INT to CHAR, such situation may happen after application upgrade or system migration as shown on Figure 2. This change may require different modifications in EDW area. Let us discuss how this change can be handled.

With Data Vault we may decide to change the type of customer_id business key from INT to CHAR, as long as integers may be easily casted to char values. Satellites and link tables are connected with the hub table based on hub_customer_hkey attribute which is calculated using hash function (e.g. MD5, SHA) from business key. If not only the type of business key would

change but the values as well (e.g. ID=234 to ID=AD_33) then the recommended practice would be to create another hub table and leaving the old one inactive for audit purposes (Figure 3). The same strategy would apply to the satellite table when new descriptive columns such as the second_name would be added.

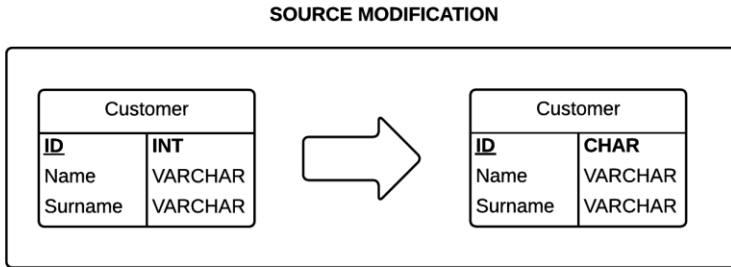


Figure 2. Data warehouse multi-layer architecture

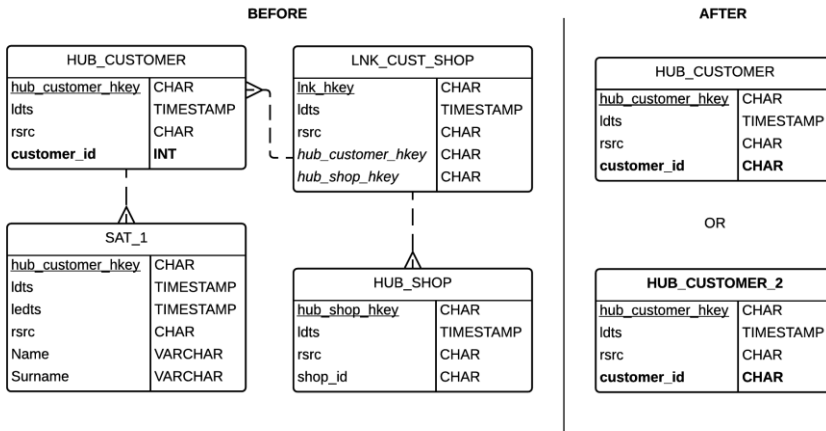


Figure 3. Recommended approaches for key column change with Data Vault

The dimensional model will behave similarly as Data Vault, only one column would have to change and if the change would also affect values it would require either overwriting old ones or creating, archiving them or creating complete new set of facts connected with a new dimension. Normalized EDW layer will require changing of the key type in every related table what makes it the most demanding model in this scenario (see Figure 4).

In this scenario both dimensional and Data Vault models seem to cause less trouble because only one attribute has to be changed opposite to normalized model. If the change would also affect previous attribute values all the models would face similar challenges.

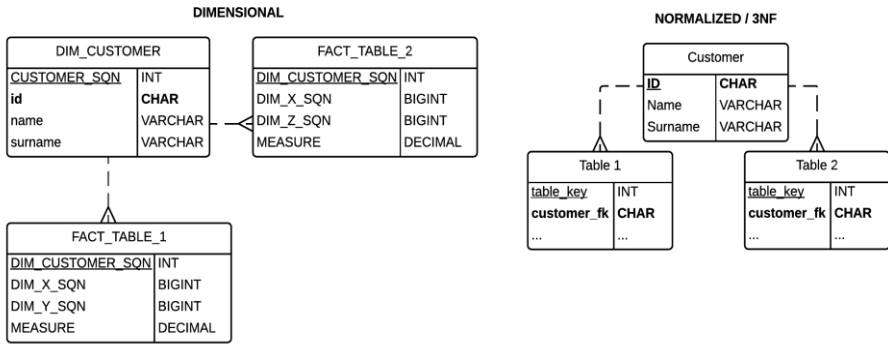


Figure 4. Structure changes with dimensional and normalized EDW model

3.2. Business rules modification

When working with data sooner or later one will come across non atomic attributes such as codes with business meaning. Let's discuss hypothetical change in retail shop category code meaning change which may occur after company decides to change its profile from food only to general store (Figure 5).

cat_code	meaning
DRY_LONG	dry goods, long EXP
DRY_SHORT	dry goods, short EXP
LOO_LONG	loose food, long EXP
....	...

➔

cat_code	meaning
FOOD_OWN_LOW	food, own brand, low margin
TEX_OUT_HIGH	textiles, producer brand, low margin
TEX_OUT_LOW	textiles, producer brand, low margin
...	...

Figure 5. Category code before and after change

After this transition old product category changes were not only expanded but completely changed in order to better analyze new business, food type and expiration types were replaced with a product, brand and margin types. This

type of change is a revolution not only for the chain owners but also for the previously designed data warehouse. Required modifications regarding respective models will be discussed based on Figure 6.

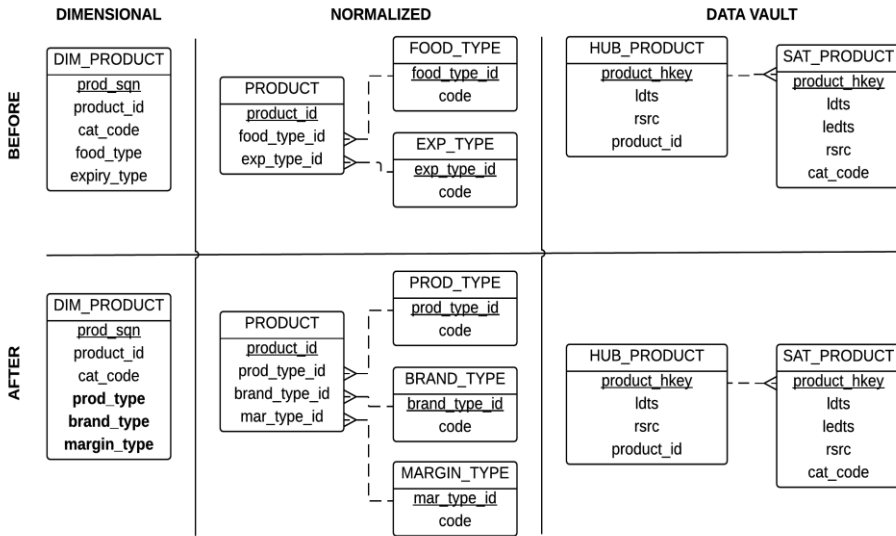


Figure 6. Changes in EDW after business rules modification

Data vault recommended technique is to do nothing as applying business rules on EDW layer is inadvisable. The way in which cat_code could be decomposed into different attributes is by every means a business rule, that is why aren't present. Satellite decomposed cat_code columns such as food_type are not the default element and may be added only in a special form of the model called Business Data Vault.

The dimensional model will require adding a new fields to the product dimension table. One may consider adding completely new dimension but this in turn would require modification of related facts, which should be treated as a drawback as well as a possibility for sqn sequence keys to change after complete dimension reloads, thus would require related facts reloads as well.

When using normalized model product new type tables should be added, old food and expiry type tables may be removed or left as they were, but new products will have null values in corresponding food_type_id and exp_type_id attributes.

Data Vault proves again to handle changes well as it do not require to reload any existing data what minimizes risk of serious failure, especially when historical data is not available. Moreover moving transformations such as cat_code decomposition into another layer appears to be a future proof technique.

3.3. Historical data handling

Attribute values change is a frequent event within transactional databases. Discussed models do have proved recommendations for keeping track of them so we will discuss them in classical case of surname change. This kind of change may be treated as slow changes as opposed to frequent changes such as e.g. call centre agent stage.

We can assume that after surname update the value was simply overwritten in the source system. It is usually the data warehouse responsibility to keep track of these changes what usually happens at reload interval and is connected with the notion of wrinkle-of-time [3]. Standard ways of handling these changes within EDW layer is demonstrated on Figure 7.

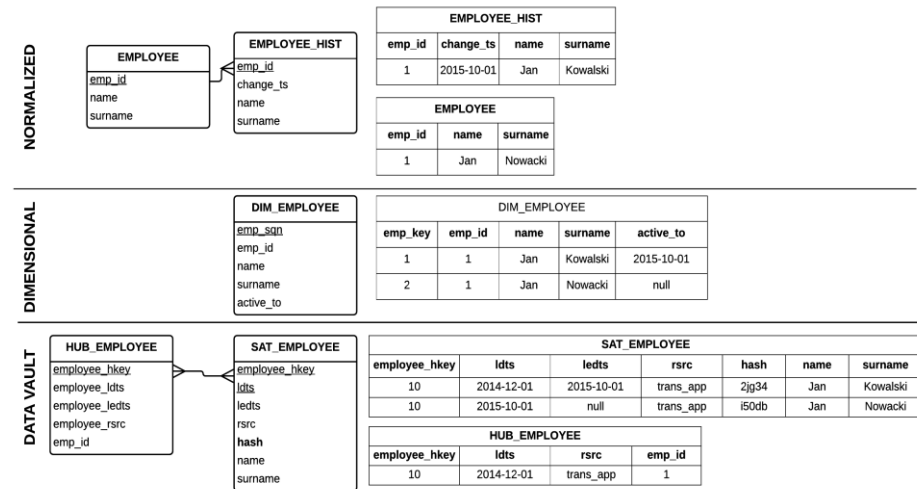


Figure 7. Handling surname change from Kowalski to Nowacki within EDW layer

Data Vaults fundamental demand is to enable changes tracking so that restoring the sources at a given point in time to be available. That is why this

model supports versioning in every satellite table similarly to type 2 slowly changing dimensions in dimensional model. After the change is discovered a new record with load timestamp is inserted. To make update lookup operation faster *hash* column is introduced. Hash can be generated from source data in a load time and then matched with existing hash eliminating the need for complex join conditions.

With normalized model additional history table might be used and record state could be tracked after every change with help of *change_ts* timestamp attribute, which stores information about update moment. This approach is useful if one wants to have a quick access to the current data via employee table but some historical analysis are also available thanks to *employee_history* table.

Dimensional model supports data versioning with several techniques of which the most popular are slowly changing dimensions (SCD) [6]. In discussed scenario we may decide to go with SCD type 2 adding a new row whenever defined attributes changes. In this model no attribute hash was introduced but it may be used as some optimization. Nevertheless there will still be the requirement for matching facts that appeared at given time with the dimension state at that time what is doubtlessly time consuming (additional joins required).

Data Vault proves itself a good model for versioning from both computational and design perspective. Thanks to use of hash keys in its tables and satellite attribute hashes it is easy and fast to load.

3.4. Data cleansing and consistency

It can be commonly heard that data in the data warehouse should be clean and consistent, but such approach may cause some problems when project evolves. Firstly Data Vault author spotlights the risk connected with applying business rules on EDW layer what was discussed in section 3.2. Operations such as: data standardization (e.g. same format for addresses), formatting (e.g. upper case names notation), cleansing (e.g. deleting invalid phone numbers) are completely ignored by Data Vault. This approach has pros and cons but it is convincing that using non universal ways of cleaning data may lead to

such modification that it will be impossible to track the real state of source systems.

Moreover some data can be incorrectly and irreversibly changed due to wrong cleansing techniques. Let's examine Figure 8 with example of name cleansing with fuzzy match, if lookup dictionary won't contain given name e.g. Leo it could be transformed into Leon and loaded as such into data warehouse. Such operation would be irreversible in some cases, and as EDW should be a base for further layers it seems reasonable not to clean data extensively there.

The same applies to the consistency; let's assume that after data load it appears that the same bill was assigned to two customers which is against expected source data model. Enforcing one-to-one relationship within EDW area would require applying business rule for choosing the right assignment, but as we already know it might not be a good idea. Though Data Vault makes architects not to do so because link tables that models relationships are many-to-many tables by default which in turn allows modelling any relationship without having to duplicate bill entity. With dimensional model it would be required to create two bill fact entries, one for each customer and normalized model would have to handle two bills with the same business key. Once again Data Vault approach appears to be a reasonable choice for cleansing and consistency.

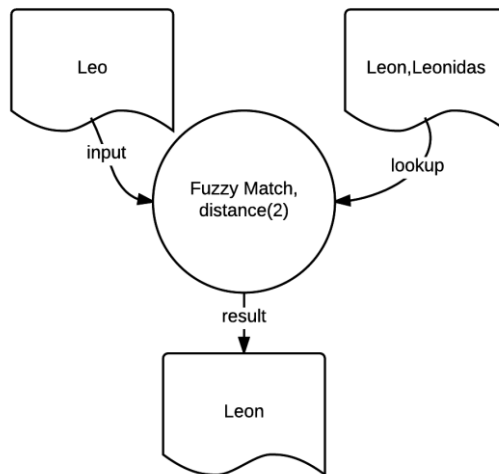


Figure 8. Example of incorrect name cleansing

4. ELT Automation for Faster Changes

Another important elements of data warehouse landscape are the ETL/ELT processes. Usually when the project starts it is hard to predict the nature of production environment. Let's consider the impact of the decision that all new data has to be loaded into the data warehouse directly from dump files instead of direct insert commands. This example is quite likely to happen when using services such as Amazon Redshift where inserting new data from S3 files via COPY command is a recommended technique opposite to inserting it row by row [1]. Modifying all existing ETL jobs would be a time consuming and vulnerable to errors. This is why it appears convenient to minimize the number of jobs what can be achieved by generalizing them. Based on our positive experience with Data Vault and it's standardized structure we propose a semi-automatic framework (Figure 9) for loading staging data into Data Vault EDW area. This framework is based on two general requirements:

- source data is extracted into staging area first;
- staging area tables reside within the same database as Data Vault tables.

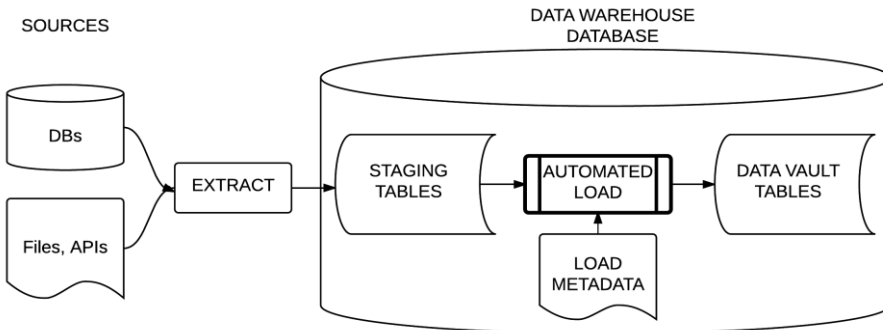


Figure 9. Semi-automated Data Vault loading framework

This framework is capable of loading all Data Vault objects i.e. hubs, links and satellites. Data is extracted from sources using independent jobs into the staging area within target database. After loading staging area generic load mechanisms start to generate required SQL queries in order to perform steps demonstrated on Figure 10 for each Data Vault's table type i.e.:

- generate queries to fetch staging data;

- generate and execute insert command using generated queries;
- generate and execute queries deactivating updated or deleted records.

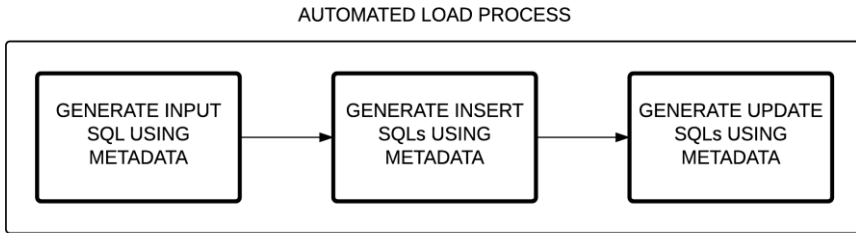


Figure 10. Data Vault generic load process

First step of generating input queries is acquired by using queries and location metadata provided by developer. This approach enables high level of flexibility regarding usage of staging data.

Within the insert generation step new rows are identified by computing and comparing hash keys generated from source business keys (in case of hub and link tables) against hash keys (*hkey*) that already exist within Data Vault. In case of satellite tables hash keys are also computed but apart from them attribute hash column *hash* is computed and used for faster change detection, so if computed *hkey* and *hash* do not match any record existing in target satellite new row is inserted.

In the last step updates are acquired by using OLAP LEAD function to identify if any newer record with the same *hkey* appeared within specified satellite table if so the last seen date attribute (*lsd* column or *ledts* as used in our implementation) is set to the current load timestamp as proposed by [7]. If record has been deleted from the source database the same *ledts* attribute value is set but it can be only detected by comparing hash keys present in a Data Vault and those computed from data source.

The details of meta data tables and their content won't be further discussed within this paper. It is important though that such framework allows handling Data Vault load using only 3 generic jobs what have a positive impact on time required to modify and optimize loading mechanisms.

5. Conclusion

The use of change resistant model and architecture within the Enterprise Data Warehouse layer has a positive impact on data warehouse maintenance and development process. Individual elements, case studies and recommendation presented in this paper were the result of a review of available literature and the experience gained during the successful commercial deployments. Positive attributes of Data Vault seen during discussed case studies prove that it can be treated as a serious competitor to more popular Inmon's and Kimball's models and methodologies. More detailed paper about successful usage of the discussed model in a business environment will be covered in a separate paper. The further work may aim at development on open source tool for Data Vault automation as well as comparison of the load times of popular EDW models.

References

- [1] Amazon Web Services using a copy command to load data – amazon redshift. http://docs.aws.amazon.com/redshift/latest/dg/t_Loading_tables_with_the_COPY_command.html, accessed: April 17, 2016.
- [2] R. Hughes. *Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013.
- [3] W. H. Inmon. *Building the Data Warehouse*. QED Information Sciences, Inc., Wellesley, MA, USA, 1992.
- [4] W. H. Inmon, C. Imhoff, and R. Sousa. *Corporate Information Factory*. John Wiley & Sons, Inc., New York, NY, USA, 1997.
- [5] R. Kimball, L. Reeves, W. Thornthwaite, M. Ross, and W. Thornwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998.
- [6] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2002.
- [7] D. Linstedt and K. Graziano. *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault*. Createspace Independent Pub, 2011.
- [8] D. Linstedt and M. Olschimke. *Building a Scalable Data Warehouse with Data Vault 2.0*. Elsevier Science, 2015.
- [9] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software*

Engineering, ICSE'87, pp. 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

Chapter 5

Toward Agile Data Warehousing

1. Introduction

Following the digitalization of our environment the amount of data that can be used to improve processes, services and products increase. It is growing every year exponentially and according to IDC [1] in 2020 will be 50 times higher than in 2009 and 4 times higher than in 2015. Analytical needs of enterprises and institutions are constantly growing and aim now far beyond the analysis on aggregated level of data based on some predefined dimensions. It is desirable to know not only about the sources of existing trends, but also to predict these trends in the future. To meet the needs of analytical systems, usually implemented on data warehouses, a must have is a high degree of flexibility, very short terms of implementation together with low granulation of data. Those expectations are hard to fulfil without agile data warehousing.

Traditional software development methodology known as waterfall model is already to a major extent replaced by the agile methodologies. This trend stems from the fact that the use of a waterfall model is time-consuming both in the analysis and implementation, which does not allow for a quick response to the changes. Despite the fact that one of the most popular agile framework i.e. SCRUM [12] was founded in 1995, the first work on the application of these methodologies in data warehouse projects appear only in the second decade of the new millennium [2, 6, 8].

In the face of these challenges it becomes important to find efficient project management practices as well as the suitable architecture of data warehouse. We try to explore both areas with the emphases on chosen elements. Our main contributions are as follows:

- We present chosen agile project management techniques that have proven their robustness in several projects which we had a chance to take a part in.
- We conduct the survey-based evaluation of the techniques and discuss the results.

The paper is organized as follows: Section 2 contains a overview of project management methodology adopted for data warehouse projects. In Section 3 the idea of *Data Vault* is presented, as the basis of logical data warehouse architecture together with best physical architecture practices. Survey-based evaluation of presented techniques is the scope of the Section 4. Last Section is a short conclusion of the topics covered in this paper.

2. Agile Project Management

Agile methodologies are popular in software development for many years effectively replacing waterfall methodology. This is still not the case with respect to data warehouses. The direct application of agile methodologies in data warehousing projects often fails due to:

- frequent changes of data sources over time;
- the constantly increasing volume of data;
- long service life of data warehouses;
- requirements for compatibility with existing processes;
- multiple user groups with conflicting requirements and priorities.

In this section we will try to explore the reasons for difficulties in applying agile methods for data warehousing and describe the methods we are using successfully in every day work.

2.1. Stakeholders conflicts in requirements

Analysis of business requirements is crucial for both agile and waterfall methodologies. A correct understanding of the business needs is required to implement appropriate functionality. Compared with the implementation and development of transactional applications, the main difference in case of data warehousing projects is the significant presence of conflicts of interests of

individual stakeholders. A good sample would be the approach to data analysis of financial and operational departments regarding projects efficiency. Finance departments are mostly focused on the analysis of the profitability of individual projects, which results in erecting requirements for accessing data with high granularity, but the periods of updating the data are not crucial and the updates may occur on a weekly or even monthly basis. On the other hand, for the operating divisions important is the ability to conduct analyses at least daily and in the case of certain processes even close to real time.

The conflicting requirements together with usually much more complex architecture with many dependencies between the system components have significant impact on the effectiveness of the project management methods giving the agile methods the field to prove their supremacy. The first very important aspect in proper project segmentation.

2.2. Project segmentation based on processes

Measuring the success of the data warehouse implementation with the number of cubes in data model is not a great idea. You should focus more on whether the data model allows for rapid implementation of the new reports and support improvement regarding decisions made based on the presented information. It is advisable to gradually cover the business processes, not focusing on individual reports and their different variations.

Hughes in the book [2] highlighted the need to formulate additional criteria of business requirements, according to which a top-down approach should be working, dividing the business needs into independent smaller data models. The aim of this approach is the development of functional data warehouse divided into prioritized processes occurring in the enterprise as opposed to the prioritization of individual stories. Such approach was applied for the project covered by a survey.

2.3. Data granularity

Another important element allowing increasing the efficiency of the development of data warehouse is the right interpretation of the requirements with respect to granularity of data. When implementing the reporting

requirements it is always worth analysing the highly possible future demands. Having for instance the requirement coming from a user story to provide sales report per projects, it is worth to analyse if in the near future similar tasks may appear, taking into account more granular data e.g. sales per team, employee or product.

Starting the development of the data warehouse for particular business process should focus on analysis of the process with the lowest possible granulation of data. Instead of creating a data mart with aggregated sales data per project with respect to user story, you should rather create a data model, which will allow for the subsequent materialized aggregation to the required level of granularity or even will get the aggregation working at the level of the database query for the report without materializing it.

2.4. Project team

The often underestimated and not carefully taken aspect is the splitting of project roles between the project team members and external players. In this subsection we describe our recommended practices for organizing data warehouse implementation team based on different roles and responsibilities. The external players like the *Business Sponsor* play in case of data warehouse project similar role like for the other software projects. In case of data warehouse it is usually easier to gain the attention from higher level management because results of the project usually address them directly. In case of development team we advice to split it into front-end (reporting) and back-end (data modelling) team in order to gain the proper separation on concerns.

The front-end team will be more evolved in the communication with end users to build reports or dashboards reflecting end users expectations and way of working. It should be less involved in building the data model and ETL processes what will be the domain of back-end team. The front-end team should have very good knowledge of the business process being supported by a data warehouse. The introduction of a separate front-end team together with the usage of self-service tools, as proposed in the next section, provides to far more user-friendly reporting layer of the application built also in sustainable less time.

The reporting team can consist partly of future key-users of application from customer side, which gradually learning the reporting tools in development phase can then educate end-users being usually a part of the same team. This saves a lot of time for breaking the mental barriers between software developers and end users. The leader of the reporting team should cooperate closely with the *Product owner* especially regarding the critical decisions about the implemented functional requirements. Neither the less he should also support the back-end team in defining the proper data model.

The understandable data model of the data warehouse becomes crucial for projects strongly relying on self-service tools. Making the data model clear for key and end-users is a big challenge, but if successful ensures the huge reduction of future application maintenance costs. The users understanding the data presented through the data model can and, as our experience tells, really are inventive about the possible new reporting and analytical requirements that can also to the great extent be implemented directly by the users in with self-service tools.

The back-end team will have more technical tasks regarding building the data model in chosen database or data warehousing system and creating the ETL processes to fill in the data. In the following section we will focus on the technical architecture of the solution that supports the reduction of implementation tasks needed to build fully functional data warehouse application. Never the less the overall success of the data warehousing project is depended on the results of development. The key problems to be coupled are the development team are already described like proper project segmentation, coupling with conflicting requirements and taking into account the data granularity. The technical tasks of the development team are not much different in case of data warehouse projects like in other software project whereby the same agile principles apply.

2.5. Kanban

The Kanban methodology is the agile project management technique that was adopted by us for data warehouse projects. The planning in Kanban is different than the planning process in Scrum methodology where it takes place before each iteration. In Kanban the planning is more general and can be done

at intervals of several weeks. Kanban also works better with managing ad-hoc tasks as it does not require stopping current sprint cycle. During the project meetings the role of the development team is to inform the Product Owner about the technical aspects of the planned functionality influencing the project task. The specific tasks that will go to the product backlog should not be discussed in detail instead the meeting should be treated as a brainstorming session allowing optimally choose the next steps with a balanced perspective for both technical and business expectations.

Kanban based implementation should be started with creating understandable process consisting of clearly described steps. The following table shows the proposed stages of the implementation based on cooperation between the reporting and development team. During implementations we worked out an optimal process described in Table 1.

Table 1. Phases on Kanban board

Phase	Name	Description
Phase 1	Backlog	Tasks in the form of stories commissioned by the Product Owner. The tasks of the highest priority at the top of the list.
Phase 2	In progress	Tasks in progress. One member of the team can deal with up to one task at a time.
Phase 3	Review	The tasks waiting for the review of correctness and performance in accordance with accepted standards (e.g. Is the code readable? Are there attached the relevant descriptions?)
Phase 4	Test	Performing the functional tests by a dedicated developer or tester, before passing to verify by the Product Owner
Phase 5	Business Verification	Product Owner alone or with a reporting team verifies compliance with the requirements.
Phase 6	Done	In this stage are tasks considered to be done, after a defined time they can be archived.

It is worth noting that in the proposed process the verification step is present two times which is contrary to the Kanban principle of the simplification of the process. We suggest however to do it, due to the fact that in case of errors in the source system the values which for technical persons seem

plausible for the business may look abnormal at first glance. Such situation happened many times in our project and helped to detect discrepancies.

3. Agile Data Warehouse Architecture

The data warehouse architecture can benefit from using modern technologies at both logical and technical perspectives. We will first discuss the logical architecture where the data model plays the central role and then move to the considerations about using self-service and in-memory/columnar databases to reduce the complexity of technical system components.

We tried to compare the nowadays most popular multidimensional models with the Data Vault model in the face of common challenges occurring in the development and maintenance of data warehouse. The aim was finding the best data model for implementation, bearing in mind the fact that the iterative growth of requirements in agile methodologies introduces the risk of frequent changes of requirements.

3.1. Agile data modelling

The already classical approach for multidimensional modelling popularized mostly by Kimball and Inmon [4, 5] and being a part of the Common Warehouse Metamodel [9] is the de facto standard of data warehouse implementation. The new propositions like Data Vault [3] have then a tough opponent to beat and are forced to bear with developers habits. Nevertheless we advice at least considering of trying to invest some effort in a proof of concept to collect some experience and have a possibility to look and the data model from different perspective. We have done it and the change became not only possible but also profitable.

The Data Vault modelling emphasizes several aspects of data modelling which are also addressed by the other methods in different way:

- Data should be traceable i.e. the source of a data needs to be recorded.
- A single version of the facts should be stored i.e. both correct and incorrect data with respect to business rules and not only the correct and cleaned so called single version of the truth.

There are also important non-functional aspects taken into consideration which are rather out of the scope for the classical models like enabling of parallel loading and the guaranty of being resilient to change in the business environment.

The Data Vault architecture was developed by Dan Lindstedt [7]. It is a denormalized model but looking at the sample ERD diagram you get the impression that you see the 3NF model (see Figure 1).

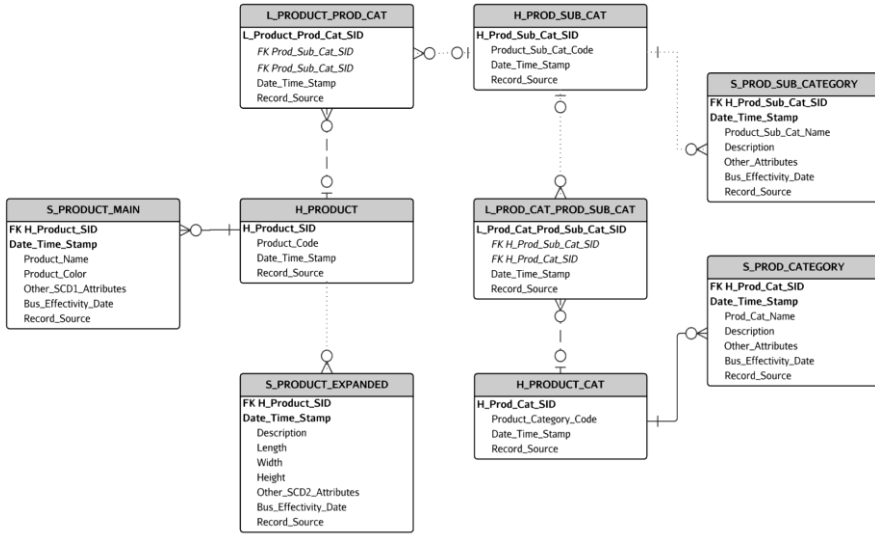


Figure 1. Sample Data Vault model

Dan Lindstedt distinguishes three types of tables: hubs (with the names starting from H on the sample diagram), links (L) and satellites (S), which have well-defined functions. Currently there are two variants of the model available: Data Vault (DV) and Data Vault 2.0 (DV 2.0).

Hubs are tables, which in a minimalist way represent objects from the business environment such as employee, product or store (like dimensions). The structure of the hub table is based on the object key coming from the source and is enriched with additional system attributes such as date of loading, data source and a timestamp specifying when the object was last seen in the source.

Links realize the function of determining the relationships between the objects represented by the hubs. They represent the relation a many-to-many

(like facts) because they indicate that at a certain time relationship between objects was valid. This dependence can be association, hierarchical relationship or the transaction. An example of the link table may be a table connecting worker hub and product hub, which can mean that the employee sold the product. A link can connect multiple hubs.

In both types of tables already presented there is no space for a descriptive data such as employee name. Here come the satellites tables, whose main role is to store all descriptive attributes of both hubs and links. Satellites also perform similar functions as slowly changing dimensions in the dimensional model with recording the history of attributes.

The Data Vault model is highly resilient to changes what is the crucial feature in agile data modelling. For sample change where the type of customer ID has changed the changes needed to other tables are limited to changing the column type in hub table and creating new satellite with a new set of descriptive fields or adding columns to an existing satellite.

We invite the reader especially interested in the features of Data Vault model to our separate paper dedicated only for analyses of Data Vault properties (to appear shortly).

3.2. Agile technical architecture

The technical architecture of the data warehouse can greatly support the overall agile development process. We would like to emphasize two elements that had mostly influenced the projects we were working on: in-memory/columnar databases together with self-serviced tools.

The first originating in columnar data representation proposed by prof. Stonebraker [11] and widely adopted also by commercial systems [10] allows the high simplification of data model. The significant performance improvement encourages the elimination of old fashioned physical performance improvements like materialized aggregation of data. The maintenance of aggregated data incises highly the complexity of the data model and data loading processes thus eliminating of aggregates with still better overall performance is one of the main benefits of using the new columnar/in-memory databases. The simplification of the data model allows reorganizing of ETL processes that can afterwards be run more frequently or gather more granular data.

The idea presented in the second section regarding building separate reporting team is extremely important for the implementation of agile process. The basic functionality of a self-service business intelligence tools is the ability to define complex data models of tables, views, or aggregated OLAP cubes. This model is then displayed to the user and allows him to generate reports and graphs. The useful feature of self-service tools is the ability to combine data from different sources in a single model and the ability to carry out initial transformation. Such a system should be able to connect to many database management systems and define the structure of collected data easy.

The self-service tools increase the awareness of how a data warehouse was build because users use the data model directly and not only the made-ready and pre-defined reports. Such a situation makes the user start thinking in terms of multi-dimensional model, which greatly facilitates the work of the development team. Thus also the stories are becoming friendlier to the implementation team and minimize the risk of errors.

Another important benefit of the self-service system is the ability to create user's own reports. This minimizes the distance between the user and the implemented data warehouse, eliminating the need to build all reports by the development team. In a short time BI platform becomes the central point of information for all the key departments in the organization, which eliminates the risk of remaining with the old practice of reporting and increase the return on investment of data warehouse project.

4. Survey-Based Evaluation

The agile techniques proposed in this paper where used by the authors in the several commercial project implemented with stable development and reporting team. Before moving toward agile techniques the team was working basically using waterfall methods. Before the evaluation the team gained significant agile experience. The survey was issued after almost two years of cooperation and three fully completed minor sized projects (about 400 man days each). The development team consisted of 5 persons and the reporting team of 6. The survey questions were prepared for evaluating the decisions made regarding the project management and the system architecture.

4.1. Agile Project Management

The questions regarding the agile project management techniques were both general like the question about the overall opinion about the Kanban method introduction (Figure 2) and more detail like the question about the most observable changes in the work space i.e. the Kanban boards (Figure 3). The positive impact of agile methods is significantly observable regarding the organization of work. In this case, 55% of persons noticed a slight improvement, with 36% significant. You can deduce that the proposed process requires the adjustments in order to better fit the development team.

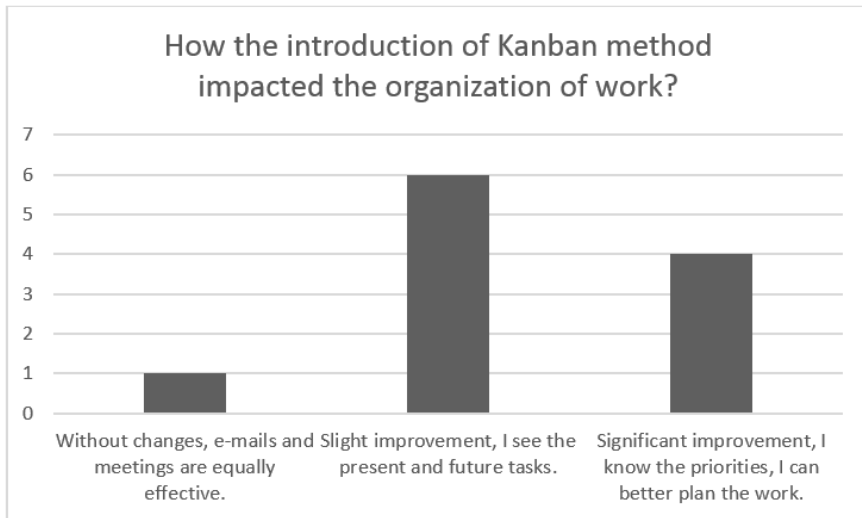


Figure 2. Work organization with Kanban

The quality of delivered artefacts after introducing tools supporting agile methods like Jira and Kanban boards, was assessed unambiguously positively. All respondents noticed an improvement of as many as 73% reported a significant improvement of the quality.

The improvement of communication within the development team as stated in another question (Figure 4) also supports the overall positive assessment of Kanban implementation. In addition, it is worth mentioning that the communication of requirements through stories was not working smoothly. Unfortunately, the requirements were not always provided in a standardized

way, more dutifulness on this issue would bring the better results. As you can see the use of Kanban brought improvement in quality, work organization and communication. It can be considered that this methodology was applied successfully for agile data warehouse implementations in this particular team.

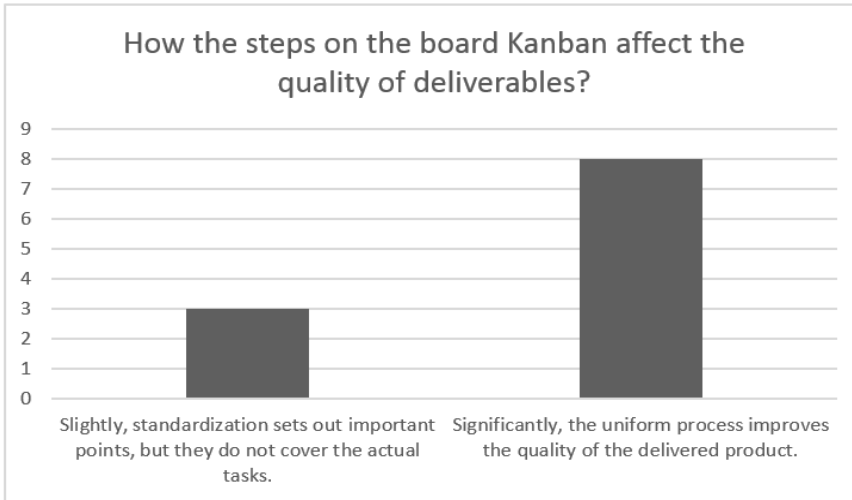


Figure 3. The quality of deliverables with Kanban

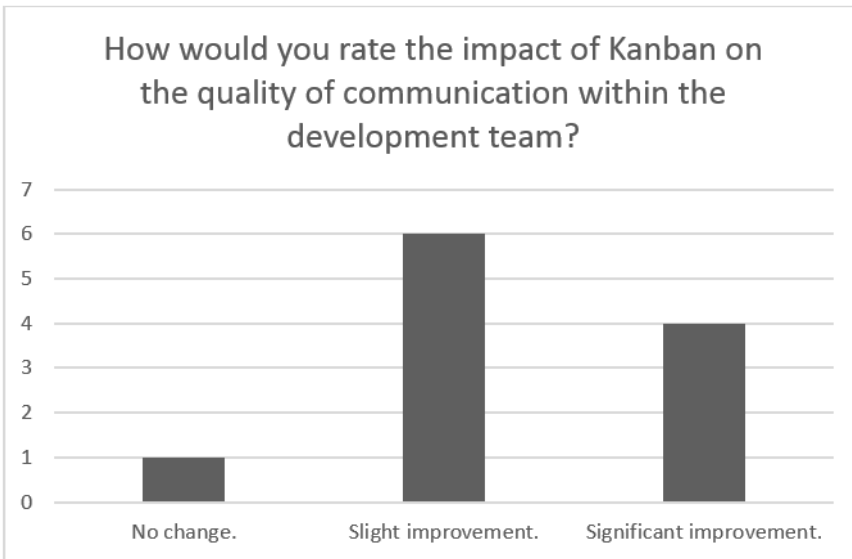


Figure 4. The quality of communication with Kanban

4.2. Agile Data Warehouse architecture

The questions regarding expected improvements in the data warehouse architecture were concentrated on evaluation of Data Vault model and the usage of self-service tools as the over proposed improvements like the usage of in-memory solutions combined with columnar data storage have more technical nature and the improvement can be measured and evaluated using the efficiency metrics.

We will present the results only for evaluation of usage the Data Vault model as it is in our opinion much less popular in data warehouses. As you can see on the results of question about the overall impact of introducing this model (Figure 5) the 60% of the members of the development team were giving the positive grades to the application of this model especially in the context of resistance to changes in requirements compared to the previously applied multidimensional model. 20% of the team noticed no change and another 20% felt the change make the things worse.

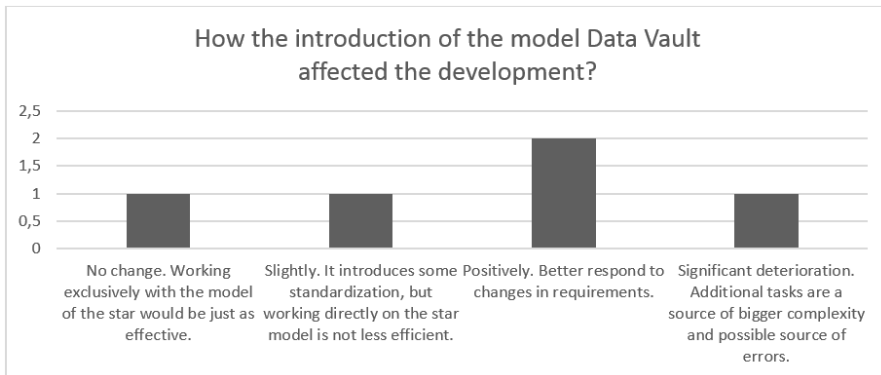


Figure 5. The development with Data Vault

The reasons for those results could be traced to the difficulties associated with having to learn a new data model from scratch and need to adapt to it the knowledge collected so far about design of data warehouse. The significantly positive answers prove the high level of acceptance for innovation in the development team. The more surprising can be the result on a question addressed to the reporting team consisted of key-users rather than developers (Figure 6). The results show clearly that this model was extremely difficult to

use for only one person in the team. It pictures probably the less conservative nature of the user perspective to particular technical implementation.

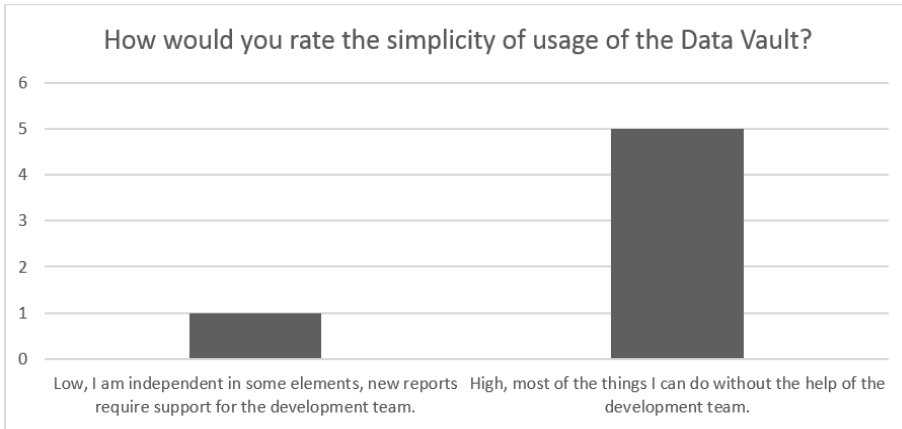


Figure 6. The ease of usage of the Data Vault

The reporting key-users were also asked about their opinion of self-service tools introduced in the projects. The assessment was unambiguously positive without any complains among team members. The self-serviced tools have greatly improved the possibility to reduce the effort of implementation of small changes in the reporting and give the reporting users the possibility to adjust the reporting layer according to their way of working.

5. Conclusion

The use of agile methodologies in the implementation of the data warehouse requires a comprehensive process for the management team, requirements analysis, and the use of appropriate technological practice. Individual elements and recommendations presented in this paper were the result of a review of available literature and the experience gained during the successful commercial deployments. The positive ratings of discussed techniques are supported by an anonymous survey-based evaluation among the development and reporting team involved in several projects. These projects were successful in the opinion of the business sponsor and have brought benefits to the enterprise, allowing quicker access to the data describing crucial business

processes. The further work may aim at development of the complete methodology based on the presented elements to form a framework to be applied in data warehousing projects.

References

- [1] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future, 2007:1–16, 2012.
- [2] R. Hughes. Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum. Newnes, 2012.
- [3] H. Hultgren. Modeling the Agile Data Warehouse with Data Vault. New Hamilton, 2012.
- [4] W. H. Inmon, D. Strauss, and G. Neushloss. DW2.0: The architecture for the next generation of data warehousing: The architecture for the next generation of data warehousing. Morgan Kaufmann, 2010.
- [5] R. Kimball and M. Ross. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons, 2011.
- [6] T. Knabke and S. Olbrich. Towards agile bi: applying in-memory technology to data warehouse architectures. In IMDM, pp. 101–114, 2011.
- [7] D. Lindstedt and K. Graziano. Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace, 2011.
- [8] M. Muntean and T. Surcel. Agile bi-the future of bi. Informatica Economica, 17(3):114, 2013.
- [9] J. Poole, D. Chang, D. Tolbert, and D. Mellor. Common warehouse metamodel, Vol. 20. John Wiley & Sons, 2002.
- [10] V. Sikka, F. Färber, A. Goel, and W. Lehner. Sap hana: The evolution from a modern main-memory data platform to an enterprise application platform. Proceedings of the VLDB Endowment, 6(11):1184–1185, 2013.
- [11] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O’Neil, et al. C-store: a column-oriented dbms. In Proceedings of the 31st international conference on Very large data bases, pp. 553–564. VLDB Endowment, 2005.
- [12] J. V. Sutherland and K. Schwaber. The scrum methodology. In Business object design and implementation: OOPSLA workshop, 1995.

Chapter 6

Performance Analysis of Web Application Using MySQL, MongoDB and Redis Databases

1. Introduction

NoSQL abbreviation means "*Not only SQL*". It is a term that covers a wide and constantly growing subject of non-relational databases. It owes its popularity, among other things, to the growing importance of *Big data* solutions. Relational databases were becoming difficult to operate and maintain with the vast amounts of data, whose pattern was subject to frequent changes. Non-relational databases, thanks to horizontal scalability and high data availability, proved to be an excellent solution, not offered by relational databases. They are perfect for real-time web applications and in *Big data* solutions. Unfortunately, the opportunities they offer lead to certain consequences which must be considered when designing IT solutions.

The most important consequence is the lack of fulfilment of the ACID principle. In return, the compromise solution known as BASE [1] is offered. The first element (*basically available*) determines that the base has high availability of data in accordance with the CAP theory (**BA**). A break in the system operation can lead to non-parallel change of data between nodes. *Eventual consistency* (**E**) is responsible for the re-synchronization when connectivity between the nodes of the database has been restored. The last element of the BASE rule is *Soft state* (**S**), which is a derivative of **E** element. It ensures data synchronization after some time, but it leaves resolving differences, conflicts related to the change in the database to the implementation of the database engine.

Breaking the ACID rule and replacing it with the BASE principle means that in practice it is much easier to implement transactional systems using relational databases. The transactional nature of the system is often a very important requirement for many applications.

Non-relational databases can typically have lower efficiency compared to the conventional ones, when the engine is addressed with a large number of queries that require searching or aggregating.

Unlike relational databases, most of which use dialects of SQL queries, in NoSQL databases there is no standardization concerned with the query language. Each of them has its own methods of data access and working on the database, often very different from each other.

An important, but often overlooked aspect is much less support of the developers' community. This can be seen very well by checking the number of query results in the Stack Overflow service. For "NoSQL" query only 22,680 entries were found, while for "sql" query as many as 303,515 entries were located. For non-relational databases, there is still a small number of tools to work with, despite an ever increasing interest.

Based on the above characteristics it can be easily inferred that NoSQL databases have their own particular group of applications and prior to implementing them in an application the consequences they bring should be taken into account.

2. Similar Work

NoSQL Databases are associated mainly with the lowest layer of Big data application type. This is due to the fact that they scale horizontally very well. Gandini and others implemented the three most popular NoSQL databases in the Amazon EC2 cloud: Cassandra, MongoDB and HBase and studied the impact of different configurations on database performance. They showed that with the help of queuing networks it is possible to model performance characteristics of the database at high loads [2].

Schmid, Galicz and Reinhardt analyzed the performance of PostgreSQL and MongoDB databases for their use in WMS (Web Map Server). In terms of

points MongoDB database was faster than PostgreSQL. As for lines and polygons MongoDB was slower than PostgreSQL [3].

Choi, Jeon and Yoon showed that in the case when efficiency is more important than reliability, MongoDB database brings significantly better results than Oracle. The research was carried out on a single entity [4].

Gupta and Narsimha compared the capabilities of MySQL and Cassandra databases. It turned out that when saving data a badly configured Cassandra may be less efficient than MySQL [5].

Barbierato, Gribaudo and Iacono conducted evaluation of the performance of Big data applications using NoSQL databases. The research was conducted on the MapReduce Apache Hadoop platform. They showed how *Big data* application developer can evaluate the performance of applications that use Hive NoSQL query language [6].

Freire and others on the basis of 4.2 million records of health care data from a relational database (MySQL) generated XML and JSON documents and then imported them to three individual XML database systems (BaseX, eXistdb and Berkeley DB XML) and to dispersed NoSQL MapReduce Couchbase database system. XML databases were much slower and required much more space than Couchbase. NoSQL Couchbase had better response times than MySQL, especially in the case of larger sets of data [7].

3. MongoDB

3.1. The most important features

MongoDB is a multiplatform database where data is stored in the form of documents whose style is similar to that used in JSON files. Files are saved in BSON format which was designed specifically to be able to support additional data types, coding, and increase the speed of searching their contents.

MongoDB database does not offer data bonds at queries and complex transactions.

3.2. Query language

MongoDB, like most non-relational databases, has specially developed query language to manipulate the database and the data contained therein. It was designed as a clear and intuitive language for the programmer. Queries involve sending commands to the database of orders in the following format [8]:

```
databaseObject.collectionName.request(  
  {listOfArguments}).modifier(listOfArguments)
```

The following example shows a simple query which selects all the documents from the user collection:

```
db.user.find()
```

Commands can be directed to the database using JavaScript programming language or the API available for a particular programming language on the part of the server.

3.3. Performance

One of the main objectives of MongoDB is the ease of scaling horizontally and ensuring high productivity at the expense of breaking the ACID principle. With the ability to scale horizontally the cost of maintenance of database servers and their load are considerably reduced. High productivity was confirmed by tests of performance. United Software Associates researched the database operation in the Yahoo's Cloud Serving Benchmark environment. MongoDB database showed significantly better performance than the Couchbase Server or Cassandra solutions in all the experiments carried out. In some cases the database was even 25 times faster [9].

3.4. Data model

MongoDB stores data in the form of documents. They are a set of attributes (keys) and their values. The documents are stored as files in BSON

format. In its structure, it resembles JSON, but it is a highly optimized format, adapted to store complex data types.

The database structure is divided into four components (consecutive, included within the preceding one) [10]:

- database,
- collections (equivalent of a table),
- documents (equivalent of a single record),
- keys and their values (equivalent of columns in a table).

The documents are able to store simple data types, such as: *string* or *int*, but also the complex types, such as objects, tables, other documents (nesting).

MongoDB also provides two methods for mapping relationships between documents. The first of these are references. Those are links pointing to a specific document. The other method involves using the opportunities of the BSON format that is nesting documents. There is the inclusion relationship and nested documents are always processed in the context of superior documents. This can significantly reduce the number of queries.

4. Redis

4.1. *The most important features*

Redis is a non-relational database written in C. It is characterized by high performance, but at the expense of high demand on storage resources. It works entirely within the computer's memory (RAM) [11]. To some extent it has the handling of transactions implemented. They are treated as a method to perform bulk actions with the option of their cancelling. Most of the basic operations on the data are of the atomic type. Redis also supports data bonds.

4.2. *Query language*

Redis, similarly to MongoDB, has its own language used to perform operations on data contained in the database. All activities are performed using

simple text commands typed at the terminal or remotely transmitted to the base, according to the format [12]:

```
OPERATION KeyName Value
```

The following example shows the assignment of the text "TEST" to the key "testInscription" in the "inscriptions" group:

```
SET inscriptions:testInscription "TEST"
```

4.3. Performance

When the permanence of data is not high priority, Redis as a base functioning in the operating memory is an extremely efficient solution in comparison with the solutions that save the result of each operation to the disk. Redis writes and reads operations performed with symmetrical speed and acts as a individual, single-threaded process.

4.4. Data model

Redis is a database working on the basis of the key – value data model. The database may store a limited number of types. These are:

- a list of string variables,
- a collection (*set*) of string variables,
- an ordered collection (*sorted set*) of string variables,
- associative hashed arrays (*hash*),
- bitmaps,
- HyperLogLogs.

The type of value assigned to the key defines a set of methods available for data manipulation. The key can be any object (string, Java object, etc.). The key type is not limited while the specification defines that it should not exceed 1 kB of memory. An additional possibility is setting the validity period for the key, as in the *Cookies* mechanism. After a set time the database automatically removes a value.

5. Research position

It was decided that for the research purposes a real application will be used that helps to support a patient in a medical outpatient clinic, namely the numbering system used in primary healthcare institutions. In many outpatient clinics at the time of registration a number that is valid on the day for which the appointment was made is assigned to a patient. The patient often spends many hours in the waiting room not to miss their number, because the times of visits are irregular and the time of entry into the doctor's office cannot be planned.

The main tasks of the application includes providing basic information about the clinic, such as name, address, phone number, a list of doctors and the times of their duties, or the list of additional services. However, its most important feature is informing patients about the number currently admitted by the respective doctors.

The implementation of the application functionality requires frequent queries performing all the basic operations on the data stored in them (CRUD). When a large number of clinics is registered, high availability of data becomes an important aspect. Lack of data pattern also allows restricting the disk surface occupied by the base in the event of failure to provide certain data by a clinic or a user. At the time of implementing the solution on a large scale (clinics from all over the province), the ability to scale the database horizontally will be very important to be enable handling a large number of queries.

The project was carried out to according to MVC (Model–View–Controller) pattern. Due to the simplicity of operation and the opportunities offered, it is now one of the most popular architectural patterns used in modern web applications. Simple MVC framework used in the application provides MVC architecture and a number of tools facilitating the development and operation of the program. The passive model is used in the above framework. It means that it cannot itself change the status of the application without request from the controller or view. This solution is sufficient for the proper operation of the whole platform.

5.1. Database support in the application

A characteristic feature of NoSQL databases is the lack of a common standard of query language. Each solution provides a set of methods available within the API database. To be able to perform queries from PHP language level, installing an appropriate driver to support each database was required. Communication with the MySQL database is an exception, where queries can be sent using the inbuilt PHP module PHP Data Objects (PDO).

5.2. Database structures and test data (seed)

Creating a separate database for the time of developing and testing an application is common practice. On the basis of the existing projects of databases, Seed class was developed in the code responsible for automatic creation of appropriate storing structures and for inserting test data.

The above class is somewhat different for each database engine. It builds various structures suitable for each of database, but it enters the same test data. This allows the application to function using different databases at the same time having the same data in each version. In the case of both non-relational databases, data storage objects are created automatically when inserting a record or trying to refer to it (if it did not previously exist). The programmer does not have to build the data pattern in the database first. It is contrariwise when developing structures from the code level in a relational database. The first step involves building an appropriate pattern and then records are entered.

In all databases there were the data required to build the application and test the correctness of its actions, such as: user accounts with different authorization levels (administrator, physician, patient), clinics, identifiers of currently admitted patients, links of users and clinics, and the like.

5.3. MySQL

The framework used in the application incorporates a set of methods to work with MySQL database using PDO mechanism. These methods only allow working with the data within the existing data structure, and therefore the

embedded solution was replaced by the Eloquent ORM module [13]. This is an element derived from the Laravel framework. It has very complex mechanisms for working with data as well as building the database pattern from the application code level. In contrast to the MongoDB and Redis MySQL databases, structures within the database for storing data needed to be created first. The physical model of MySQL database is shown in Figure 1.

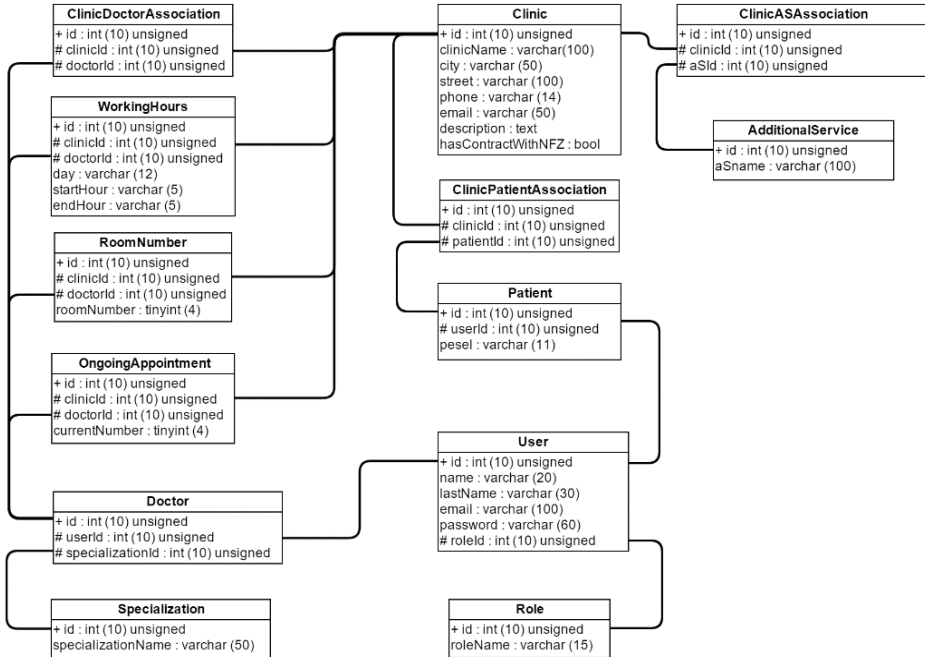


Figure 1. The physical model of MySQL database

5.4. MongoDB

To query the MongoDB database in an object-oriented way, PHP MongoDB driver was installed [14]. It provides a set of classes that allow connecting to the database, performing data manipulation and handling data types specific to the database (for example MongoId, MongoDate). To facilitate the work, the wrap class was developed, supporting CRUD method for the database. Creating collections and documents is performed automatically when

data is inserted or with a download attempt (even if no data exist in the database).

The model of the database implemented for use in the application is shown in Figure 2. The dotted lines represent the relationships occurring among the collections.

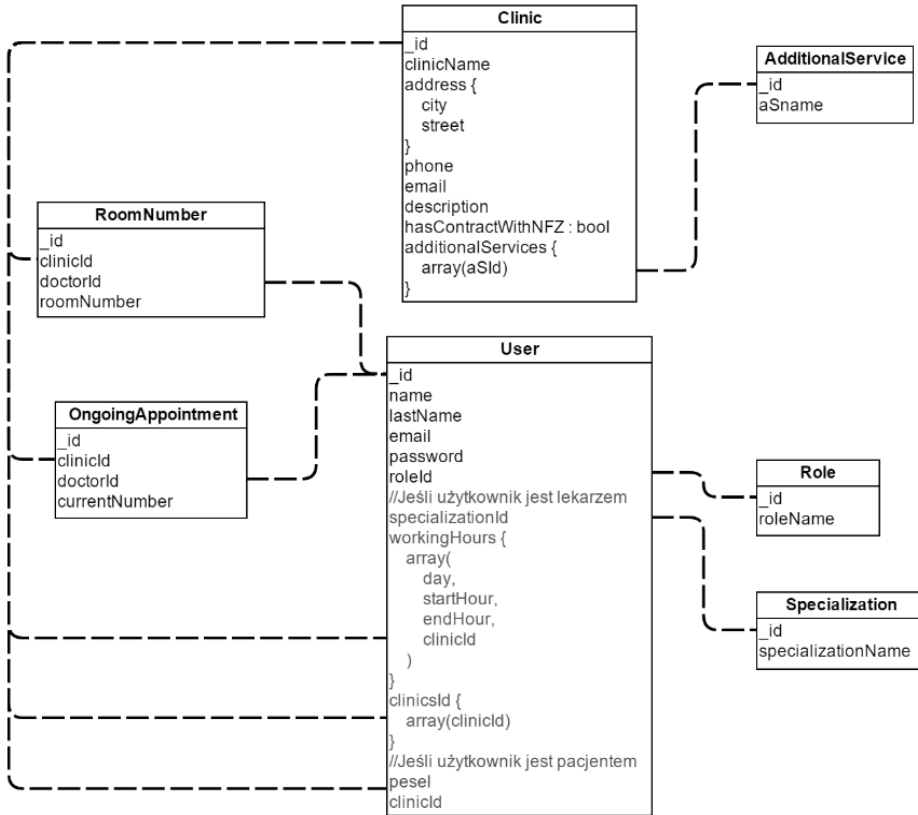


Figure 2. The physical model of MongoDB database

5.5. Redis

To operate the Redis database from the PHP language level the installation of PhpRedis extension was necessary [15]. The project is open source available within the Github hosting repository. The driver provides a set of

classes that allow connecting to the database and performing operations on it. The keys are created by assigning data to them.

Figure 3 shows the physical model of Redis database. The dotted lines represent the relationships occurring among objects. For each hash an extra key is created that stores the number of objects from the same group. The key is created according to the formula: *HashName:Count*.

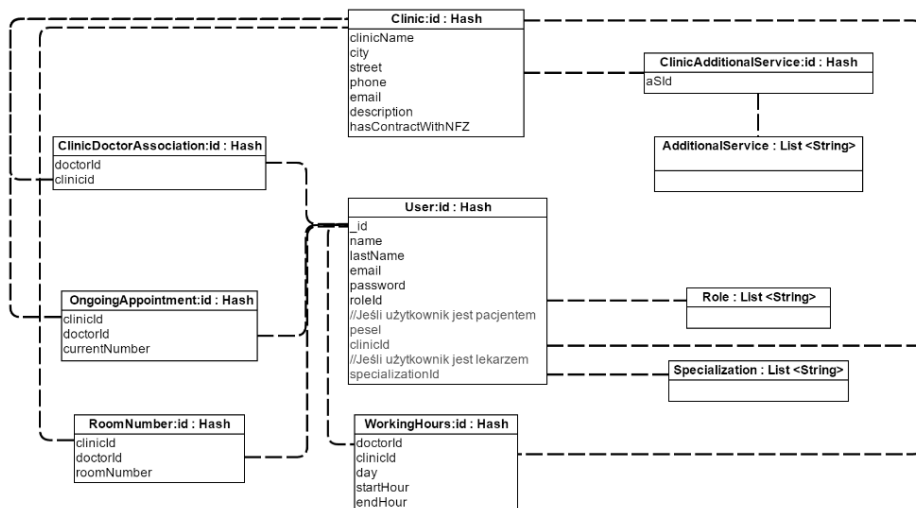


Figure 3. The physical model of Redis database

6. Experiments and Discussion of Results

The experiments were performed on a computer with an Intel Core i5 2450M (2.5 GHz) and 8 GB of RAM (DDR3), running on Linux Ubuntu. Apache Benchmark software[16] was used to carry out the experiments. This is an extension of Apache server that allows you to check the performance of an application by simulating the set traffic on the server. It is a simple HTTP client that periodically generates requests to the server for specific resources. Its effect can be parameterized by calling it from the command line. The syntax of the sample command is as follows:

```
ab -n 100 -c 10 http://example.com:80/
```

where n means the total number of requests to perform, and c is the number of simultaneous queries (competitiveness).

During the experiments those cases of use were tested that are important due to the characteristics of each of the applied databases. They were implemented through the following views:

- searching for a clinic,
- currently admitted patients,
- doctor's panel (changes of the number of the patient admitted).

First of all the time to build structures in the databases and insert test data in them (*seed*) was measured a hundred times.

Table 1. The summary of seed execution times

Database	MongoDB	Redis	MySQL
Time [s]	60.487	38.412	133.59

Table 1 summarizes the durations of the hundredfold time of building structures and placing test data for each database. In the case of MongoDB and Redis there is no need to create structures in the database before sending data to it. In MySQL database it is necessary to create the tables that will store data first. This is the cause of much shorter duration time of the experiment on NoSQL databases.

The second stage was a gradual increase in the number of queries about the above mentioned views. The following issues were taken considered: handling time for a series of requests, the number of requests per unit of time and the number of queries handled incorrectly.

The load was as follows:

- 100 requests (10 users),
- 1,000 queries (100 users),
- 10,000 requests (1000 users).

The search clinic view was burdened first. The view of currently admitted patients is the critical element from the end-user perspective. It is exposed to the largest load in the system. The view of changing the number of the currently admitted patient (doctor's panel) is linked to the previous view. The change of number in this view takes place within 15 seconds after the number

is changed by a doctor. This view works under considerably less burden than the previous views.

Figure 4 shows a graph of the experiment duration, depending on the number of requests sent for each view. Figure 5 presents the average number of requests handled per second for each view. The graph in Figure 6 shows the number of requests handled incorrectly, depending on the system load for each view.

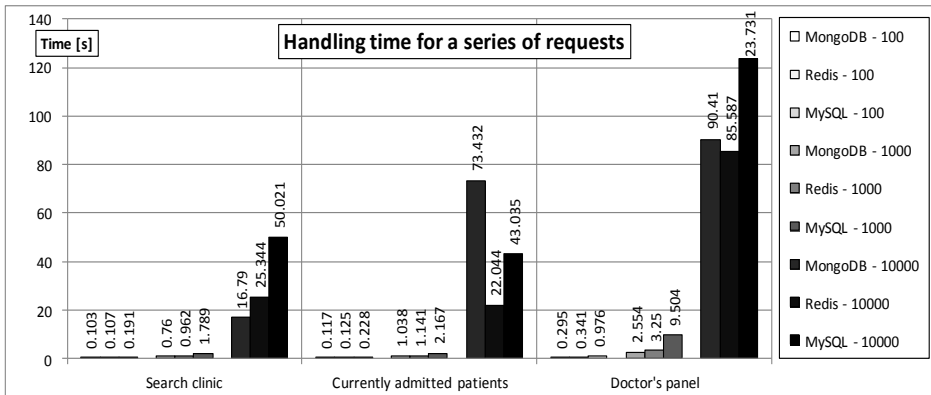


Figure 4. The chart of handling time for a series of requests depending on the number of requests sent for individual views

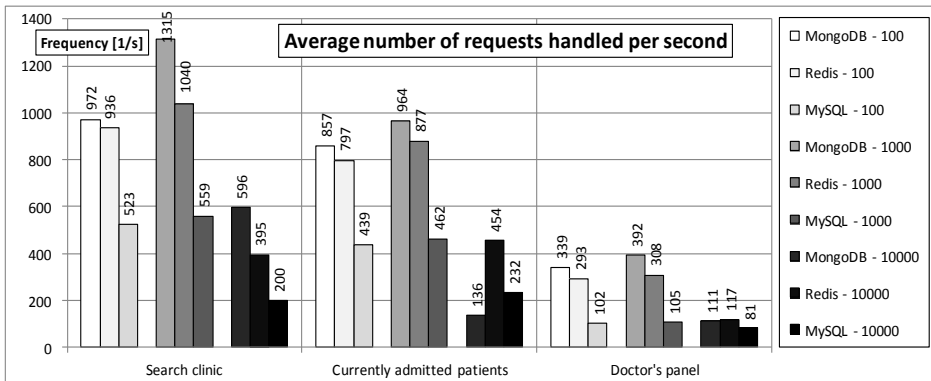


Figure 5. The chart of the average number of requests handled per second for individual views

The measurement results clearly show that the performance of all databases with little or periodically increased traffic is comparable. The differences are the order of hundredths or tenths of a second. It was only at the time of a

very large number of simultaneous queries that differences in performance and proper operation are revealed.

Both the non-relational databases reveal a much higher efficiency of operation in comparison with MySQL. In some cases the processing times are nearly three times shorter and there are nearly four times more properly handled queries.

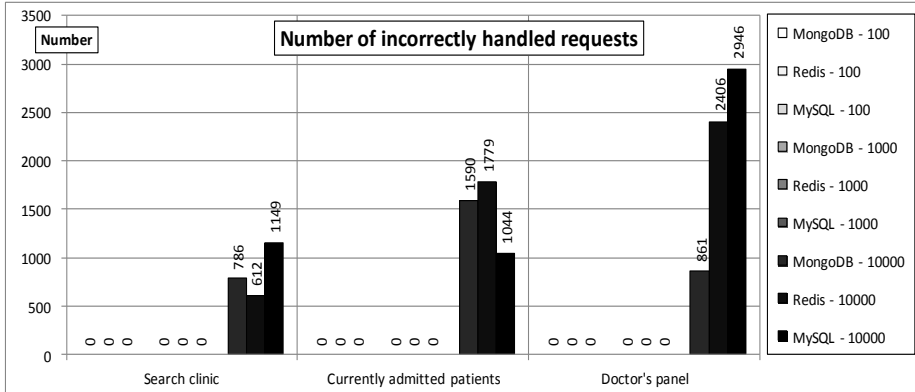


Figure 6. The chart of the number of incorrectly handled requests for individual views

The weaker results of Redis in comparison with MongoDB are a result of the database characteristics. Access to elements is carried out only by means of the primary key. It is not possible to select objects based on the value. This often creates the need to retrieve all objects and iterate through them to find the right ones. This feature is considered a great advantage as well as a major disadvantage of Redis database. Performance improvement can be achieved by dispersing the database and changing the design of the structure.

The results of MySQL database under heavy traffic applications are clearly weaker than the results of MongoDB and Redis. However, under normal system load, the loading time of views is comparable and does not affect the quality of work with the application.

7. Summary

In order to perform the aforementioned analysis, an application was built that implemented the same set of tasks based on different database

engines: MongoDB, Redis and MySQL. The application was designed to get the most out of each of the databases. The aim was also to present the opportunities and differences in working with NoSQL databases and to show their advantages and disadvantages. The application was created in PHP, and to service the databases, special packages (drivers) providing sets of methods to communicate with databases were used.

The application was built using Simple PHP MVC framework. As a result the architecture in line with the MVC architectural pattern was provided.

The experiments were performed on a local Apache server in the environment of the Linux Ubuntu operating system. The tool used to carry out the stress test was Apache Benchmark, which generated requests of different intensity to the application. For views which implemented the most important functionalities experiments were carried out in stages. At each stage the number of simultaneous requests to the server was increased. Despite the heavy load, non-relational databases showed high data availability fulfilling all the queries even three times faster than MySQL.

References

- [1] P. Zieliński, Wprowadzenie do NoSQL, część I, *Microsoft Developer Network*, <https://msdn.microsoft.com/pl-pl/dn912483.aspx>.
- [2] A. Gandini, M. Gribaudo, W.J. Knottenbelt, R. Osman, P. Piazzolla. Performance Evaluation of NoSQL Databases, *Computer Performance Engineering*, LNCS 8721, pp. 16–29, 2014.
- [3] S. Schmid, E. Galicz, W. Reinhardt. WMS performance of selected SQL and NoSQL databases, *Military Technologies (ICMT)*, IEEE 7153736, pp. 1–6, 2015.
- [4] Y. Choi, W. Jeon, S. Yoon, Improving Database System Performance by Applying NoSQL, *Journal of Information Processing Systems*, Vol.10, No. 3, pp. 355–364, 2014.
- [5] S. Gupta, N. Narsimha. Performance Evaluation of Nosql-Cassandra over Relational Data Store-Mysql for Bigdata, *International Journal of Technology* 6(4):640, pp. 640–649, 2015.
- [6] E. Barbierato, M. Gribaudo, M. Iacono. Performance evaluation of NoSQL big-data applications using multi-formalism models, *Future Generation Computer Systems* 37, pp. 345–353, 2014.
- [7] S.M. Freire, D. Teodoro, F. Wei-Kleiner, E. Sundvall, D. Karlsson, P. Lambrix. Comparing the Performance of NoSQL Approaches for Managing Archetype-Based Electronic Health Record Data, *PLoS ONE* 11(3), pp. 1–20, 2016.

- [8] The MongoDB Manual, MongoDB Inc., 2015, <https://docs.mongodb.com/manual/>
- [9] High Performance Benchmarking: MongoDB and NoSQL Systems, United Software Associates, 2014, http://info-mongodb-com.s3.amazonaws.com/-High%2BPerformance%2BBenchmark%2BWhite%2BPaper_final.pdf
- [10] R. Chamot. MongoDB Tutorial, <http://student.agh.edu.pl/~chamot/bazy/>
- [11] M. Boruta. Redis – Wprowadzenie, Silesian Ruby User Group, 2011, <https://srug.pl/assets/miroslaw-boruta-redis-wprowadzenie.pdf>
- [12] The Redis Documentation, <http://redis.io/documentation>
- [13] T. Otwell. Eloquent ORM, <http://laravel.com/docs/5.0/eloquent>
- [14] PHP Group. MongoDB driver, <http://php.net/manual/en/book.mongo.php>
- [15] M. Grunder, [phpredis/phpredis](https://github.com/phpredis/phpredis), <https://github.com/phpredis/phpredis>
- [16] B. Skvorc. Stress-test your PHP App with ApacheBench, Sitepoint, 2014 <http://www.sitepoint.com/stress-test-php-app-apachebench/>

Chapter 7

Merging Textual Representations of Software Models – a Practical Approach

1. Introduction

Model-driven engineering is becoming increasingly important in software engineering. By models in this paper, we refer to graph-based software models. The most common example for this is the Unified Modelling Language (UML [1]), but our focus is on domain-specific modelling (like EMF [2], GME [3], VMTS [4] etc.).

Displaying and editing these models can be done in several ways: using a graphical notation, a textual notation or using both simultaneously. The main difference and advantage of textual notations over graphical notations lies in the fact that even though a graphical notation is usually easier to understand, editing the text is often more efficient than editing the model via a graphical interface. This is especially true in the case of large or complex models. If a development environment with convenience features (e.g. syntax highlighting, code completion) is also available, then the efficiency of the editing process is further increased. There are other papers that further elaborate on the relevance and the advantages and disadvantages of the textual approach [5].

To coordinate and optimize teamwork in source code-based development, version control systems (VCS [6]) are used. The same concept can be carried over to MDE as well. Texts are easier to compare and merge than the models themselves [5]. There are no prevalent version control systems yet that support text-based modelling. The motivation of our research is to create the foundation for conflict handling in such a VCS.

There are three main approaches to comparing the textual representations of models: (1) comparing the raw texts; (2) comparing the abstract syntax trees (AST-s [7]); (3) creating a specific comparator for the language that describes the model. The third approach is the most accurate and efficient one, but it requires a concrete implementation for every language. The first two approaches are more general, but usually less accurate. In this paper, accuracy means that the approach identifies as many correct differences and as few incorrect differences as possible. The formal definition and analysis of accuracy in the presented method are parts of future work. The second approach is the main focus of our work. Models are transformed into textual form with the aid of domain-specific languages (DSL [7]). During the reverse operation, the parser of the language builds an AST from the text, which can then be easily parsed in order to get the structure of the model from the text. Using the AST-s in addition to the raw texts during the comparison achieves more accurate and easily understandable results.

The goal of our research is to develop a method for comparing and merging the textual representations of models. The method can be applied to handle conflicts in a VCS that supports text-based modelling. The inputs of this method are two textual representations. The output is the merged text that contains every solved difference between the two texts. Using the AST-s in addition to the raw texts during the comparison gives more accurate results. In the examples of this paper, we are using a theoretical language (Simple Modelling Language or SMDL for future reference) that describes our sample theoretical models. A theoretical model can only contain nodes and the nodes can only contain typed attributes. Edges and other model elements are handled the same way, they are excluded for the sake of simplicity. It is also worth mentioning that while the examples in this paper are very simple and small in size, preliminary experiments show that the method can be applied to more complex models as well. Future work is planned to further study and improve the performance of the method.

The differences between raw text-based and AST-based comparison is illustrated in the example in Figure 1. When using a traditional text comparing method [8] to compare the different states of our model, we cannot get any information on which model elements these changes are related to, whereas using the abstract syntax trees as basis of the comparison, we can associate the

changes with model elements. In our example, the order of nodes A and B are reversed, and the type of attribute A1 is different. When comparing the raw texts, we cannot tell which model elements are different, we only see the difference in text. When comparing the AST-s, the structure of the trees contains more information and we can determine the elements that were changed. In this example, the change in the attribute type of A1 can be discovered once we compare the two texts that belong to the two A1 trees. Therefore, although traditional text comparison is still needed, AST-based comparison is the core of the method presented here.

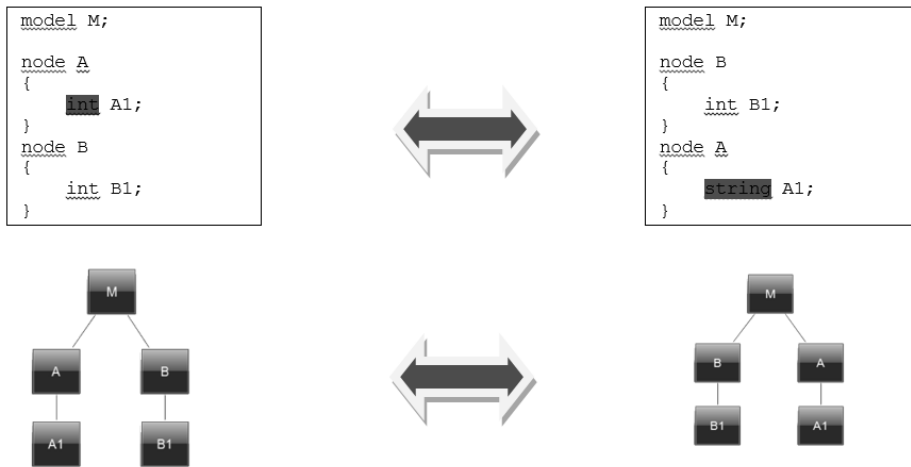


Figure 1. Differences between raw text-based and AST-based comparison

In previous work [9], we have introduced the concept of the previously described method. The crucial steps of the method were outlined, with the focus being on applicability and the differences between other, already existing approaches to model comparing and merging. In this paper, we take a practical approach and describe the method in greater detail. Following the principles described in this paper, a concrete implementation of the method can be realized.

The structure of this paper is organized as follows. In Section 2, we present related work and briefly summarize the results of previous work, with the focus being on the generality of the method, the differences with source code comparison and the contrasting between other existing approaches. In Section 3, the method is described in detail along with examples for each step. The

main contribution of this paper can be found in this section. The algorithms and techniques detailed here do not describe a particular implementation, rather they are fundamentals that can be expanded to implement the method. There exists a prototype implementation [10] of the method, which is presented in more detail in Section 4 along with the evaluation of the applicability of our method. Finally, in Section 5, we summarize the paper.

2. Background and Related Work

In this section, we summarize the concept described in previous work [9] in addition to presenting related work. We first take a look at other, already existing model comparing or merging approaches. After that, we talk about differences between the method and source code comparison. Finally, we touch upon the generality of the method.

There are very few existing approaches that can compare or merge models based on their textual representations. There are methods that specialize in comparing UML [1] models [11, 12]. These methods, however, do not handle textual representations, instead they use the inner structure of the models as basis of the comparison. Approaches that are prevalent in the industry, like EMF Diff/Merge [13] also tend to focus on comparing the models based on their structures instead of using textual representations. TMDIFF [14] is an algorithm that is used to compare the textual representations of a model. It uses a reference graph-based matching algorithm and does not focus solely on the AST that is parsed from the text. This approach is different from our method as it is more suited for behavioural models and has trouble with moved elements in the text.

Now we summarize the concept of our method detailed in previous work. Comparing and merging the textual representations of models is similar, yet different from source code comparison. The difference lies in the fact that we can demand that the models, and therefore their textual representations should be both syntactically and semantically correct, whereas semantic correctness cannot be decided in the case of a single source code file. This policy

is reinforced by the fact that most modelling environments [2, 3, 4] ensure that models edited through their graphical interface remain semantically correct.

Our method aims to be both accurate and general. As we have seen before, the accuracy comes from the AST-based comparison. When certain criteria are met, the method can also be used as a general model comparing and merging method in addition to being the foundation for conflict handling in a VCS. In our case, generality means that the method can be used with any modelling environment or language, provided that the parser of the language delegates certain operations to our method. These operations are the following:

- Parsing the text and building the AST.
- Syntactic and semantic verification of the text.
- Deciding if two trees represent the same element.
- Deciding if the difference between the texts of two trees concerns only the format.

With the use of these operations, our method is independent of the language. Therefore, there is an additional input of our method – the parser of the language. In addition, since we are using the AST-s as basis of our comparison instead of the structure of the model (as it was demonstrated in Section 1), our method can also be applied to any modelling environment. Thus, the method can be considered general with the added restriction that the parser of the language must be modifiable to delegate these operations to our method. It is worth mentioning that the method is not applicable when the parser of the language does not support delegating the operations.

3. Contributions

In this section, we present the details of our approach to comparing and merging the textual representations of models. The goal of the method is to compare the two textual representations based on the AST-s built from the texts, and create the merged text that contains solutions to every difference between the two textual representations of the model. The main steps of this process are as follows: (1) matching the trees to identify matching elements in the two texts; (2) discovering the differences between the two texts; (3) solving the differences and creating the merged text.

3.1. Tree matching

The first step of the method is comparing the two AST-s that are built from the textual representations, and matching the trees that represent the same elements with each other. This is the foundation of the comparison, the later steps of the method all build upon the result of the matching process. It is worth mentioning that although in SMDL, every tree is mapped to an actual model element, this is usually not the case with real languages. In practice, the AST also contains trees that serve only to represent the syntax of the language. In addition, a certain model element can also consist of multiple trees at once. The mapping between the AST and the model is realized by the parser of the language. Therefore, when we are speaking about model elements and trees, in reality, those trees are not always mapped to a model element directly, but the principles of the method stay the same.

The matching of the AST-s is done with the help of the parser of the language. One of the operations the parser has to delegate to our method is deciding if two trees represent the same element. Using this operation, we can easily identify which trees we need to match with each other. These matched trees are then stored for later use. We also make note of the trees that had no matching pair in the other AST.

The algorithm that matches the trees is very simple. We try to match the root trees with each other, and if they match (the parser of the language decides this), then we repeat the process with the children of the root tree in the first AST. For every child tree in the first AST, we try to match it with every child tree in the other AST. The process is then repeated for the children of a matched tree, and so on. At the end of the procedure, we try to match every tree that does not have a matching pair. Although it is not the focus of our research, matching the trees in this way instead of comparing every tree with every other tree in the other AST has substantial performance benefits.

Figure 2 describes the tree matching process in a simple example. In SMDL, two trees represent the same element if and only if they have the same name, therefore the parser decides if they are the same based on their names. The first step of the matching process is to try and match the root tree M, which is successful. After that, we try to match the nodes A, B and C in the first AST with B and A in the second one. Nodes A and B are successfully

matched, but C is left without a matching pair. Next, the children of A and B are matched, and both A1 and B1 are matched successfully. Node C in the right AST is left without a matching pair. Finally, we try to match the unmatched trees. These are the two C nodes, which have the same name, therefore every element is matched successfully in our example. It is worth mentioning that in practice, node C being the child of node A in the second AST is usually a containment relationship.

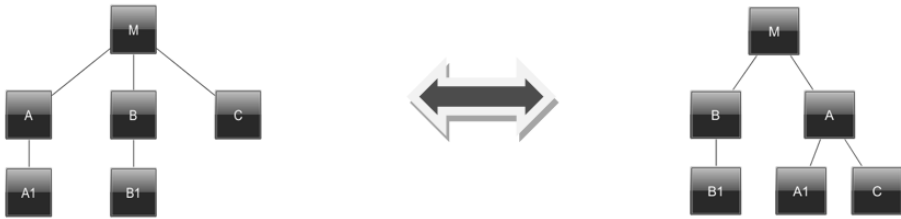


Figure 2. Tree matching

3.2. Conflict handling

During the comparison of the textual representations, our goal is to find every difference between the two texts. To discover these difference, we are of course using the AST-s in addition to the matched tree lists that we got as the result of the tree matching process. In this paper, one such difference is called a *conflict*. Conflicts are assigned to the innermost tree in which they occur. In the case of SMDL, this means that a conflict between two attributes is assigned to the attribute trees instead of the node trees. In this section, we take a look at the different types of conflicts that can occur between the two textual representations. These conflict types cover every possible conflict between the two texts. We also examine how these conflicts can be solved and discovered.

Conflict Solving. In order to create the merged text, we have to solve every conflict between the two textual representations. We would like to automate the process as much as possible in order to limit user interaction. We cannot eliminate user interaction altogether for the same reasons that traditional version control systems cannot eliminate it either. Solving a conflict means we assign a solution to it. A solution is a simple text that replaces the text of the tree associated with the conflict, with the text of the solution.

Next, we examine the different types of conflicts that can occur. For each type, we take a look at its description, possible solutions and a way to discover the conflict.

Different Text Conflict (DTC). This conflict type occurs when two trees are matched – meaning they represent the same element – but the texts associated with the trees are different. One of the delegated operations of the parser tells us if the difference only concerns the format of the text or if the difference is semantic.

To discover this conflict type, we have to use traditional text comparing methods to compare the texts of every matching pair of trees. Afterwards, we have to ask the parser of the language whether the difference concerns only the format of the text or not.

If the difference is semantic, we cannot decide which solution would be more suited for automation, since we cannot ask the parser to decide which one is the better solution. Therefore we cannot automate the solving process in this case. If the difference concerns only the format of the text, then as a best practice, the longer text should be chosen as the automatic solution. The reason for this is that in most languages, the longer text usually contains the various non-semantic elements, like comments, for example. This, of course is very much dependent on the particular language, thus there is no correct solution, best practices can be applied instead.

An example of a DTC is illustrated in Figure 3. The AST-s are omitted from the figure because in this case, there is no difference between them. When we compare the texts belonging to the A nodes, we recognize that there is a conflict. After asking the parser of the language, we know that it only concerns the format of the text as there is a comment in the first text. Therefore, we can automate the solution of this conflict. We also find another conflict between the two texts related to the type of attribute B1. This is a semantic difference, which means that the solution of this conflict cannot be automated.

New Tree Conflict (NTC). This conflict type occurs when there are trees that are unmatched after the tree matching process. For each unmatched tree, an NTC is created. This type represents that there is a new tree in one of the AST-s that is not present in the other one. Discovering NTC-s is rather easy, as we only have to check if there are any unmatched trees after the tree matching process.

We also have to decide where in the merged text the new tree is placed. One possible solution is that we insert the new tree at the end of the merged text. Another solution is that we insert it after its previous tree in the corresponding AST. This requires storing the previous tree.

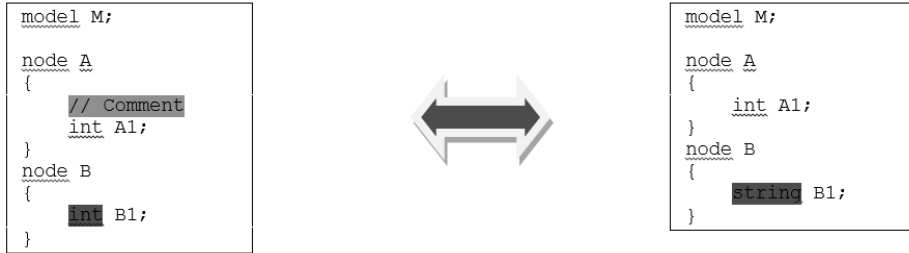


Figure 3. DTC

There are no universally correct solutions for solving NTC-s. One applicable best practice is that most of the time we would like to see the new tree appear in the merged text. Therefore, we can automate the solution by automatically inserting the text belonging to the new tree in the merged text. An example NTC is illustrated in Figure 4. The texts are omitted from the figure as they are not relevant in this case. In the example, we have 3 trees that are unmatched after the tree matching process: B2, C and C1. Attribute B2 and node C both get an NTC associated with them. In the case of C1, however, it is unnecessary to create a new conflict as C1 is the child of C, because the conflict associated with C contains C1 as well.

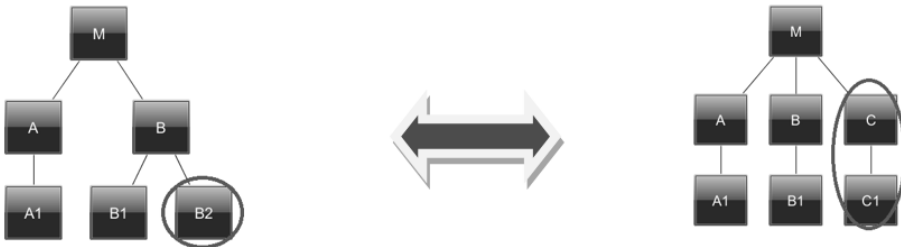


Figure 4. NTC

Handling the order of elements can be done in a variety of ways, as there are several best practices, but no universally correct solution. The solution described here is one such best practice that is based on the fact that cases

where two or three consecutive elements have their order changed are very common in practice. In the examples described in this section, we are using a different notation for the AST-s. The sub trees of the first AST are located on the left side of the figure, while the sub trees of the second AST are located on the right side. We examine this solution through an example illustrated in Figure 5. In this solution for the order problem, we compare the list of trees that are located after the specific tree in one AST and the ones that are located after the specific tree in the other AST. For example, node A has B, C, D, E and F located after it in the first AST and D, F and E in the second. We conclude that A-B and A-C are conflicts in the order of trees as marked in Figure 5. The same can be done for the rest of the trees, after which we find out that B-C and E-F are also conflicts.

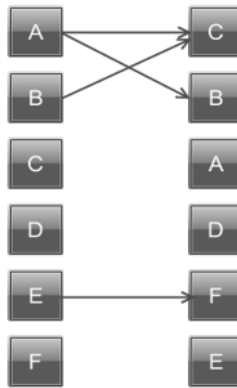


Figure 5. Discovering OC-s

We can see that there are two order conflicts in the example: $EF \Leftrightarrow FE$ and $ABC \Leftrightarrow CBA$. Therefore, we have to identify which patterns resemble cases where the order of two or three consecutive trees is different, and handle the rest of the cases as movement in the text. These patterns are illustrated in Figure 6.

In textual form, the patterns are the following: (1) A-B; (2) A-B, A-C, B-C; (3) A-C, B-C; (4) A-B, A-C. Since there is no universally correct solution as to which order is better and order difference is usually not a semantic difference in the model, we can automatically solve OC-s with one of the orders. Similarly to new tree conflicts, we have to store the previous trees for every tree present in the order.

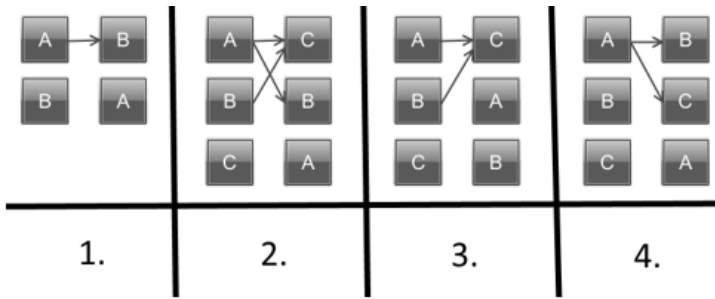


Figure 6. Notable OC patterns

3.3. The merging process

The merging process begins after every conflict has been discovered. In this section, we first take a look at the two phases of the merging process. After that, an example is presented, in which we demonstrate the merging process and also speak about the interactions the different conflict types can have with each other. Some details of the merging process are omitted due to the limits of this paper.

The merging process consists of two phases: the automatic phase and the incremental phase. During the automatic phase, a merged text is generated that contains the automatic solution of every conflict. For conflicts that cannot be solved automatically, placeholder text that is prominent to the user should be used instead. During the incremental phase, the user can change the solutions assigned to the conflicts, similar to traditional version control systems. In previous sections, it was mentioned that every conflict is assigned to the innermost tree that it belongs to. Keeping track of the conflicts like this is applicable as using the automatic solutions always provides a syntactically and semantically correct merged text. However, once the user changes the solution of a conflict in the incremental phase, we cannot be sure that the resulting text, and therefore the AST built from it is still correct. This is mostly due to the fact that it is recommended in practice to allow users to arbitrarily change the solution of a conflict in order to avoid possible errors. The same situation arises if a conflict has placeholder text associated with it as a solution. Thus, tracking the conflicts during the merging process has to be done by maintaining their absolute positions in the merged text.

Figure 7 depicts the end of the automatic phase of the merging process in an example. The order conflict ($ABC \Leftrightarrow CBA$) is automatically solved by using the left order. The new tree D1 is automatically inserted into the merged tree, right after node D in the text. The attribute type difference of A1 is a semantic difference, therefore we cannot solve it automatically, and instead we insert a placeholder into the merged text. The end of the merging process can only be started by the user once every conflict is solved. At the end of the process, we have to ensure that the merged text is both syntactically and semantically correct, thus a valid model can be parsed from the text.

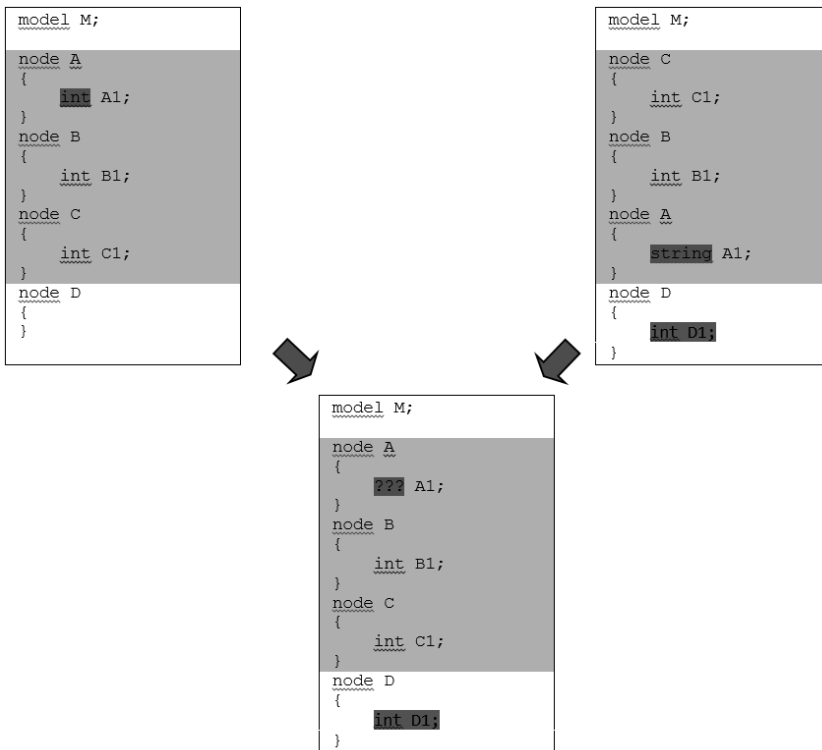


Figure 7. The result of the automatic phase of an example merging process

To summarize, we take a look at the interactions between the conflict types that can affect the tracking of conflicts by their absolute positions in the text. We examine what happens in our example described in Figure 7 if the user changes the solution assigned to our order conflict ($ABC \Leftrightarrow CBA$). Changing the order conflict also changes the absolute position of the tree of

attribute A1, thus the conflict related to A1 also changes its position, which we have to take into account. Finally, Table 1 summarizes the interactions that the different conflict types can have on each other. The cells of the table indicate the objects that need their offset updated when the specific conflict is changed.

Table 1. Interactions between the different conflict types.

		Interacting Conflict Type		
		Different Text	New Tree	Order
Changed Conflict Type	Different Text	Position	Position and Previous Tree	Position and Previous Trees
	New Tree	Position	Position and Previous Tree	Position and Previous Trees
	Order	Position	Position and Previous Tree (varies)	Position and Previous Trees

4. Evaluation of Results and Future Work

The AST-based comparison used by the method is an accurate and general comparison that the rest of the method builds upon. Therefore, the method can not only be used as a foundation of version control systems for the textual representations of models, it also serves as a general and accurate approach to comparing and merging models when the discussed criteria are met. These criteria are as follows: (1) the model must have a formal language that describes its textual form; (2) the parser of the language has to delegate certain operations to our method.

Visual Modelling and Transformation System (VMTS [4]) is a graph-based, domain-specific meta-modelling and model processing framework. Visual Model Definition Language (VMDL) is a textual language that is used to textually describe models in VMTS. An integrated development environment (IDE) with advanced features such as automated code completion and syntax highlighting is also available for VMDL. There is a prototype implementation [10] of the method presented in this paper. The prototype is based on the fundamentals presented in this paper. It was successfully integrated

with VMTS and VMDL, but it can be integrated with other modelling environments, thus generality can be accomplished in practice as well, with the mentioned restrictions. After loading the textual representations, the application matches the AST-s, discovers every conflict and generates the merged text while automatically solving as many conflicts as possible. The user can then override the solution of each conflict either manually or choosing from the list of alternative solutions, and finish the merging process. Practical experience in using the prototype with VMTS shows that the application can be suitably used in the case of small and medium-sized models.

The goal of our research was to develop an accurate and general method for comparing and merging the textual representations of models. Performance was not the focus during development, but our goal for future work is to optimize the different steps of the method. Testing and measuring the performance of the prototype implementation is also our goal. We talked about how the different conflict types presented in this paper cover every difference between two textual representations. While preliminary experiments lead us to believe that this is the case, another major topic of future work is to formally prove this, thus further elaborate on the accuracy of the method.

5. Conclusions

The main motivation behind our research was to create the foundation for conflict handling in version control systems that work with the textual representations of models. Our aim was for the comparison to be as accurate as possible while also aiming for generality as best we can. The goal of our research was to develop a method for this purpose using the abstract syntax trees (AST-s) built from the texts in addition to the texts themselves as basis of the comparison.

In this paper, we presented the method in greater detail. Using the fundamentals described here, a concrete implementation of this method can be realized, with the prototype implementation being proof of this. The method consists of three main steps. During the tree matching process, AST-based comparison is used, which is more accurate and general than raw text-based comparison, thus its results can be efficiently used in the later steps of the

method. The conflicts between the two textual representations are categorized into three different conflict types (DTC, NTC, OC). These types categorize the differences that can occur during the comparison. Every conflict type has their own possible solutions and ways of discovery. Every conflict must be discovered in order to start the merging process. The merging process is similar to traditional version control systems as they both have two distinct phases. During the automatic phase, user involvement is not required as the conflicts are solved automatically. In the incremental phase, the user can review the automatic process and make changes as they see fit. During the merging process, we have to handle the interactions that the different conflict types can have with each other.

We established that in addition to being the foundation for conflict handling in version control systems, the method can also be an accurate and general approach to comparing or merging models when the discussed criteria are met. We also briefly presented a prototype implementation of the method that was realized using the fundamentals presented in this paper.

References

- [1] Unified Modeling Language, <http://www.omg.org/spec/UML/>
- [2] Eclipse Modeling Framework, <https://eclipse.org/modeling/emf/>
- [3] Generic Modeling Environment, <http://www.isis.vanderbilt.edu/projects/gme/>
- [4] Visual Modeling and Transformation System, <https://www.aut.bme.hu/Pages/Research/VMTS/Introduction>
- [5] H. Grönniger, H. Krahn, R. Bernhard, S. Martin, V. Steven. Text-based Modeling, In: Proceedings of the 4th International Workshop on Software Language Engineering (ateM 2007), Nashville, TN, USA, Informatik-Bericht Nr. 4/2007, Johannes-Gutenberg-Universität Mainz, 2007.
- [6] W. Swierstra, A. Löh. The Semantics of Version Control, In: Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, Portland, Oregon, USA, 2014.
- [7] M. Fowler. Domain Specific Languages, Addison-Wesley Professional, 2010.
- [8] GNU Diff Utils, <https://www.gnu.org/software/diffutils/>
- [9] F. A. Somogyi. Merging textual representations of software models, Multi-Science – microCAD International Multidisciplinary Scientific Conference, Miskolc, Hungary, 2016.
- [10] Prototype implementation of the method, <https://www.aut.bme.hu/Upload/Pages/Research/VMTS/Papers/TextualModelComparer.zip>

- [11] Z. Xing, E. Stroulia. UMLDiff: an algorithm for object-oriented design differencing, In: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (ASE '05). ACM, New York, NY, USA, 2005, 54–65.
- [12] U. Kelter, J. Wehren, J. Niere. A Generic Difference Algorithm for UML Models, P. Liggesmeyer, K. Pohl & M. Goedicke (eds.), Software Engineering, pp. 105–116: GI. ISBN: 3-88579-393-8, 2005.
- [13] EMF Diff/Merge, <http://www.eclipse.org/diffmerge/>
- [14] R. van Rozen, T. van der Storm, Model Differencing for Textual DSLs, BE-NEVOL 2014-Proceedings of the Belgian-Netherlands Evaluation Workshop, 2014.

II. Software Maintenance

Chapter 8

Software Metrics in Boa Large-Scale Software Mining Infrastructure: Challenges and Solutions

1. Introduction

Boa is a tool that can be used for data mining repositories of open-source projects. It contains the full history of a repository—from every revision’s date and author, data on added, deleted and modified files to the complete state of the repository at the moment of commit. All data can be obtained by using the dedicated language. Boa provides a set of functions, which can be used for advanced data filtering [1, 2].

Boa has already been used for a variety of studies, including developers’ willingness to adapt new Java features [3] or the licenses used in open-source projects [4]. So far they have not been metrics-oriented, even though the tool is intended to be used this way, as implicated by the inclusion of appropriate examples in the documentation of Boa [5] (e.g., *What are the number of attributes (NOA), per-project and per-type?*, *What are the number of public methods (NPM), per-project and per-type?*).

In this paper we focus on using Boa infrastructure to answer three research questions:

- 1) Which of the classic, widely known, software engineering metrics can be implemented in Boa? The implementation of classic software engineering metrics in Boa and publication of calculation scripts will make it easier to extend existing small-scale empirical software engineering research using software metrics, performed usually on a small number of projects, to a large-scale research.

- 2) What new metrics, that take advantage of the Boa's unique infrastructure, can be proposed? This paper will serve as a guide, for other researchers and practitioners, who shows how to implement new software metrics taking into account the unique features, as well as limitations, of the Boa large-scale software repository mining platform.
- 3) What is the feasibility of defect prediction models based on large number of projects data obtained from Boa data sets? According to our knowledge, this is one of the first attempts (if not the first) to build large-scale software defect prediction models based on a very large number of projects. Existing software defect prediction models usually base on a very limited number of projects.

Presented study refers to state of Boa framework during October 2015 – January 2016 period – when the source material was gathered.

2. Research methodology

In this section we introduce briefly into the following topics: how we selected projects for further investigation (see Section 2.1), how we implemented software metric scripts using the Boa language (see Section 2.2), and how we built software defect prediction models using software metrics from Boa (see Section 2.3), including also how we obtained data from the Boa output files (see Section 2.4).

2.1. Projects selection

Boa source code described in this paper has been developed and tested on two Boa data sets: September 2015 GitHub, and September 2013 SourceForge. A special filtering has been applied to select projects passing some entry criteria. The software projects explored in our study had to pass the following criteria:

- 1) **They have to have a code repository with revisions.** The *2013 September/SourceForge* data set consists of 700k projects. Our analysis with Boa queries has shown that 30% of them have no code repository [6, Section 2.1]. Out of remaining 489k (amount close to this stated by

Boa developers – 494,158 [7]) 4,767 projects have two repositories. Repositories in those projects have common history of revisions [6, Section 1.1]. In case of projects with multiple repositories, only the first of them is considered during study to avoid data duplication. Out of 489k projects with one code repository, 423k of them had no code revisions (commits) [6, Section 1.2]. It is difficult to determine whatever or not Boa is missing some data—the data sets have been defined for a given month in a given year, and current state of the repository might be different. The *2015 September/GitHub* data set has 7.83 million projects. 95% of them have no code repository in the Boa framework, even though the majority of them is available from the GitHub website. They are active and public, but most of them have had no commits since 2013 [6, Section 1.3]. From 380k projects with repository, only 2486 of them had commits in 2015 [6, Section 1.4]. Out of the entire GitHub dataset, 4% of projects have code repositories with revisions [6, Section 1.5].

- 2) **They have to have over 100 commits.** The projects picked should be mature enough for metrics calculation. A larger number of commits usually means a larger number of *fixing revisions*, which are in turn used for development of software defect prediction models.
- 3) **They have to be written in Java.** Java has been picked for this research due to being a mature, object-oriented language, popular among developers. It is also worth mentioning that Boa is written in Java, as well as provides extra Java-specific options, such as recognizing Java source files with and without parsing errors.

The Boa language implementation of filters to select projects fulfilling the above mentioned criteria is presented in Listing 1.

```
before node: Project -> {
  # They have to be written in Java.
  ifall (i: int; !match(`^java$`,
    lowercase(node.programming_languages[i]))) stop;

  # They have to have a code repository with revisions.
  if(len(node.code_repositories) > 0) {
    visit(node.code_repositories[0]);
```

```

    }
    stop;
}
before node: CodeRepository -> {
  # They have to have over 100 commits.
  if(len(node.revisions) < 100) stop;
  ...
}

```

Listing 1. Implementation of filters

The final number of projects that passed our entry criteria is presented in Table 1.

Table 1. Data sets

Dataset	All projects	Accepted projects
GH small	7,988	29
GH medium	783,982	2485
GH large	7,830,023	25307
SF small	7,029	50
SF medium	69,735	666
SF large	699,331	7407

2.2. Implementation of SE metrics

All of the metrics are calculated for classes. Each of the metric is implemented as a different Boa query, and is run on all Boa data sets mentioned in Section 2.

Due to long execution time, only data from GH small and SF small data sets are used for creating prediction models later on.

The output file of a query has to have the following data:

- the ID of the project
- the ID of the class
- the value of the calculated metric or the expected value.

This approach makes it possible to effortlessly merge all values gathered as the outputs of Boa queries, so they can be used as an input data set for a prediction model.

2.3. Defect prediction model

Software defect prediction model is aiming to find the classes that cause the most defects. A simple strategy to find them is searching for the classes that had been fixed most frequently.

2.3.1. Expected value – NCFIX

The expected value in our defect prediction model is Number of Class Fixes. Based on Boa's abilities, it is assumed the class has been fixed, if the two following conditions have been met:

- the file containing the class has been modified in a revision;
- the revision is marked as a fixing revision by the Boa's function *isfixingrevision* [1].

The list of classes and their fixes is obtained by the following algorithm:

- 1) Create an empty key-value collection for storing respectively: files in projects, number of fixing revisions for each file.
- 2) Visit a project's repository revision.
- 3) Check if it's a fixing revision.
- 4) Investigate the files changed in this revision.
 - (a) If a file is marked as deleted, remove it from the collection.
 - (b) If a file is added to the project in the current revision, add it to the collection:
 - i. with a value of 1 if the revision is a fixing one;
 - ii. with a value of 0 otherwise.
 - (c) If a file is modified in the current revision, update it in the collection
 - i. increment the number of fixes by one, if the revision is a fixing one;
 - ii. leave it otherwise.
- 5) Repeat steps 2-4 until you reach the most recent revision and there is no more revisions to check.
- 6) For all files stored in the collection, select only the ones that declare classes. Return the identifiers of the classes, and numbers of fixes corresponding to their files as the output.

The algorithm is inspired by the *getsnapshot* function implemented by Boa [1], which returns the state of the repository at given time stamp.

2.4. The use of Boa API and Weka

To allow easy management of Boa jobs and connecting job outputs with development of defect prediction models, a simple Java program [8] has been written. The software uses Boa Java API [9] release 0.1.0 to run jobs. Data from Boa is transformed into *.arff* file of following format:

```
@RELATION classes
@ATTRIBUTE class ID string
@ATTRIBUTE M_1 NUMERIC
. . .
@ATTRIBUTE M_N NUMERIC
@ATTRIBUTE fixingRevisions NUMERIC
```

where *classID* is an identifier of a studied class; *M_1 ... M_N* is a vector of calculated metrics for a class from latest repository SNAPSHOT; *fixingRevisions* attribute is the expected value described in Section 2.3.1.

3. Results

In this section three kinds of contribution are discussed, related to implementation of classic and new software metrics in Boa, as well as development of software defect prediction models on a basis of very large number of software projects. The latter can be seen as a way to address external validity threats common for most of the empirical studies focused on software defect prediction. All metrics' implementations are available to download via links provided in appendix [6, Section 3].

3.1. Implementation of classic software engineering metrics

This section presents how to implement scripts to collect some of the well-known, classic software metrics [10] in Boa. The metrics were chosen based on their popularity and Boa's limitations.

3.1.1. Obtaining classes

Using *getsnapshot* function implemented in Boa, all files available in the most recent revision of the project are gathered. Then, they are filtered so that only the files containing classes are taken into consideration. The data stored in the *Declaration* [1] and its attributes are used for calculating the value of a metric.

3.1.2. Inheritance issue

Each declaration (class or interface) node in Boa has its array of parents [1]. However, those parents are presented only as *Types*, meaning, they only have *TypeKind* (determining if it's a class, interface, or something else) and name, without its full package path or any other identifier. If two classes or interfaces in a project have the same name, but they are in different packages, it is impossible to determine which one is the ancestor of a given declaration. Therefore, all metrics using inheritance (such as all of the MOOD metrics [11], Depth of Inheritance Tree, Number of Children and Coupling between Object Classes [10]) had to be unfortunately, excluded from the study.

3.1.3. Metrics obtained directly from the Declaration node

Weighted Methods per Class (WMC) in its base version—the sum of methods in a class, Number of Fields (NoF) and Number of Nested Declarations (NoND), presented in Table 2, have been successfully implemented using the structure of the *Declaration* node alone.

Table 2. Declaration attributes and associated metrics

Attribute	Metric
methods	WMC
fields	NoF
nested_declarations	NoND

For each of those metrics, the value is a length of the attribute array. The execution time for those metrics is relatively small, up to 10 minutes for the

biggest data sets, which clearly shows the advantages of using Boa and the approach to calculate metrics using the structure of the *Declaration* node, presented in this paper.

3.1.4. Response For a Class (RFC)

The RFC metric was implemented as a number of methods in the class, added to number of remote methods directly called by methods of the class. The issue with the implementation of this metric is that Boa makes it difficult to recognize the difference between class' inner method and method of the external classes of the same identifier. For example: the method *getId()* of class *A*, called in class *B*, is seen as the same as method *getId()* in class *B*. If class *A* called two methods of the same name from different classes (class *B* and class *C*), those would be indistinguishable as well. There is no direct method that would allow to instantly determine the types of called methods' arguments [1] as well as the type of instance of variable from which the method was called [6, Section 1.6]. Such information can be obtained only by deeper analysis of Boa's AST tree, to the level of single *Statements*.

The simplified version of the metric, that ignores this nuance, has been successfully implemented and ran for both Boa's data sets.

3.2. Implementation of new software metrics

The metrics presented below have been developed by us upon learning more about the Boa architecture and its tree structure.

3.2.1. Number of Statements in Methods

The NoSiM metric is calculated as a sum of all statements in class methods. The nodes calculated are of the Boa type *Statement*. For studied Java classes, those nodes are either blocks of code marked by '{ }' or single code expressions. The implementation of this metric is a starting point for implementation of a Lines of Code (LoC) metric. To achieve the LoC metric, all class' fields, number of methods, and such, would have to be added.

3.2.2. Maximum Depth of Declaration Nesting

MDoDN is the maximum level of class nesting in a class. For the following code:

```
class A {  
    class B {  
        class C {}  
    }  
    class D {}  
}
```

the result for class *A* would be 3 (the depth of *C* class). The metric is not calculated for nested classes (in the example: *B*, *C*, and *D*). For implementation of this metric, Boa's stack functions are used. Every time the node of a nested *Declaration* is entered, it is pushed onto the stack. The metric value is the stack's element count.

3.2.3. Number of Anonymous Declarations

NoAD for Java is a sum of all anonymous children classes in the parent class. To calculate this metric, the Expression Boa node is tested for having a Declaration with a parameter of *ANONYMOUS* type.

3.2.4. Cumulative metrics

Metrics NoM, NoF, NoSiM, NoAD and NoND have been also successfully implemented in cumulative versions (CNoM [6, Section 1.7], CNoF [6, Section 1.8], CNoSiM [6, Section 1.9], CNoAD [6, Section 1.10], CNoND [6, Section 1.11]), where calculated value is a sum of metric for not only a class, but also all its nested and local classes.

3.3. Defect prediction model

The defect prediction model presented below is a single defect prediction model calculated for a high number of Boa projects. This is different from

a traditional approach, with a single, or several projects used to develop defect prediction models.

Data obtained from the Boa output files (described in Section 2.4) is randomly separated into training set and testing set (in 9:1 proportion). The *fixingRevisions* attribute in the testing set is nulled out, so it can be calculated using prediction model.

We used Random Forest to build defect prediction model. Random Forest generates a lot of random samples which are the subsets of training data set. A decision tree is generated for each of the samples [12]. The parameters listed below have been determined experimentally:

- number of trees: 200,
- max depth: 12,
- number of features: 12,
- cross-validation folds: 10,
- random seed: 1

The results of 10-fold cross-validation are presented in Table 3. Pearson product-moment correlation coefficient r shows a low correlation between the results from defect prediction model and real values, with high error ratio. Those results are further analyzed in Section 4.

Table 3. Results of evaluation of the prediction model

Evaluation attribute	GH 2015 (small)	SF 2013 (small)
Correlation coefficient (R)	0.215	0.244
Mean absolute error (MAE)	2.16	0.603
Root mean squared error (RMSE)	9.96	1.32
Relative absolute error (RAE)	102%	93.3%
Root relative squared error (RRSE)	100%	97.8%

3.4. Reference values of software metrics

The subsequent goal was to characterize a large number of open source projects available from Boa by means of software metrics in order to create reference values of software metrics. Table 4 presents descriptive statistics for each of calculated metrics among the data sets.

4. Discussion

The presented prediction model was tested on small data sets, but with correct resources it can be easily scaled to use full data sets with up to 25k subjects. This use case would be, to the best of our knowledge, the first attempt to create a large scale defect prediction model, as other examples from literature show prediction models developed using less than 200 projects [13, 14, 15].

The performance of the prediction model is poor due to the fact that a majority of classes studied has zero fixing revisions and therefore input data is highly unbalanced, see Table 5. However, the quality of prediction model and employing methods to deal with the class imbalance problem are not the main objectives of the study. Our aim was to show that it is possible to collect all the data necessary to build a large-scale software defect prediction model using the Boa platform.

Results from Table 4 show that not for all metrics standard deviation is lower for filtered datasets. This can be caused by the nature of metrics (such as NoND, NoAD, MDoDN), which are unlikely to have a high mean value in majority of projects.

4.1. Further research

It is worth to look at the way the fix in the revision is identified. Boa-provided function *isfixingrevision* is based only on the commit message text analysis. We assume this function is not ideal and integrating Boa API with outside software, such as bug tracking systems, can be a better solution to determine existing bugs in code revisions.

The data used for building prediction models in our study has big disproportions. Applying different filters and criteria (more mature projects, different languages and so on) could provide better data set for analysis, with more fixing revisions.

An interesting path of further research are process metrics [15, 16], which reflect changes over time and are becoming the crucial ingredients of software defect prediction models.

Table 4. Mean, median and standard deviation for metrics calculated in the study.

Metric	GH2015 (all)			GH2015 (fixes > 0)			SF2013 (all)			SF2013 (fixes > 0)		
	μ	\tilde{x}	σ	μ	\tilde{x}	σ	μ	\tilde{x}	σ	μ	\tilde{x}	σ
NOAD	0.12	0	0.86	0.15	0	1.05	0.17	0	1.31	0.34	0	2.40
CNOAD	0.13	0	0.93	0.17	0	1.13	0.19	0	1.39	0.36	0	2.51
NOND	0.32	0	1.20	0.32	0	1.28	0.14	0	0.83	0.22	0	1.00
NOF	2.98	1	15.64	3.10	1	9.29	3.75	2	8.51	4.36	2	11.43
CNOM	7.10	3	18.69	7.50	4	13.84	8.97	5	14.21	11.48	6	19.27
MDODN	0.20	0	0.44	0.22	0	0.47	0.14	0	0.38	0.20	0	0.45
CNOSIM	40.33	13	104.15	48.51	16	126.33	65.84	26	177.32	90.83	33	221.83
NOSIM	36.80	13	92.57	44.43	15	116.30	61.58	24	170.26	83.28	31	208.54
NOM	6.40	3	16.85	6.67	3	11.64	8.15	5	12.41	10.12	5	17.09
CNOF	7.10	3	18.69	7.50	4	13.84	8.97	5	14.21	11.48	6	19.27
CNOND	0.33	0	1.26	0.34	0	1.37	0.14	0	0.87	0.23	0	1.05
RFC	13.03	7	23.27	15.19	8	22.55	19.06	12	24.10	24.81	15	32.75

Table 5. Number of classes with zero and more than zero fixes in datasets

Amount of class fixes	GH 2015(small)	SF 2013 (small)
0	13296 (58.9%)	30244 (80.1%)
>0	9260 (41.1%)	7504 (19.9%)

5. Conclusions

Overall, the goal of the research, as described with research questions – implementation of software metrics in Boa and collecting data sets from a large number of projects, e.g., for the sake of prediction models – has been achieved.

We were able to implement some of the classic software engineering metrics using Boa, we presented some Boa-specific metrics, and we made an attempt to create a defect prediction model with the data we gathered. This proves that Boa can be a useful tool for data mining analysis in this particular field, as well as for creating sophisticated queries regarding its data sets. However, Boa is still a new framework that comes with a few disadvantages, and some of the metrics and operations were impossible to implement at the moment. In the following sections, the challenges met and our solutions are presented.

5.1. Challenges

Boa uses visitor pattern – one of Boa’s greatest strengths – which sometimes might provide unexpected results if queries are not written properly.

5.1.1. Local and nested classes

One of the first issues we encountered creating Boa queries was a different size of output jobs. For our metrics, we gathered all classes from all projects. Therefore, for the same data set, all queries should return the same number of rows. As it turned out, the difference was caused by the behaviour of the visitor pattern, used by Boa. When source code contains a local class (class defined inside one of the methods) or a nested class (a class declared inside of another class), this class is visited by the visitor pattern before the analysis of the class containing it ends. Upon returning to the class-container, some of its metrics and calculations had been assigned to the local or nested class.

Solution: Boa offers implementation of stacks, which we started using while visiting local and nested classes. We took advantage of this solution implementing the Maximum Depth of Declaration Nesting metric described in Section 3.2.2.

5.1.2. Boa code compilers

Boa uses two different code compilers for SourceForge and GitHub data sets. As the framework is still in early development, sometimes the same query acts differently depending on the data set used.

Example: One of Boa sample queries "How many committers are there for each project?" [17] works fine in SF [6, Section 1.12], but causes compilation error in GH [6, Section 1.13]. In that case, a small change in the code notation solved the issue [6, Section 1.14]:

- Code resulting with error:

```
committers [p. code_repositories[i]. revisions[j]].  
  committer.username] = true;
```


- Code resulting with success:

```
username : string = p. code_repositories[i].  
    revisions[j].committer.username ;  
    committers[username] = true ;
```

This example shows that a person creating queries with Boa might run into different issues depending on the data set picked.

During our research, we often used Boa dictionaries. Dictionaries are defined by Boa as *map[key_type]* of *[value_type]*. Boa returns an error, if *int* is used as a *value_type*. We must have stored our integer values as strings, which resulted in converting value to integer each time it was used in calculations, and then back to string to update the map.

5.1.3. Debugging process

The errors reported by Boa are often lacking any sort of description. The debugging process comes down to commenting out parts of queries to check which fragments are causing errors. Each code test takes about a minute (and then some follow-up time to check if the output data is correct), and sometimes multiple tests are required to find the source of an error. There is no way of tracking the execution of the queries.

Solution: All variables used during the debugging process have to be initiated, by defining its type and aggregation method, and then returned in the output file.

5.2. Contribution

The paper describes our experience with using Boa platform for implementing software engineering metrics and defect prediction models. Our findings can be useful for both researchers – with solutions presented in Section 5.1 and provided source codes for metrics we implemented – as well as developer teams and project managers, providing an example for obtaining large-scale SE metrics for projects of particular profile (i.e. number of commits, used programming language and so on). The metric implementations proposed

by us are scalable – calculated for classes, but could be as well implemented for packages or projects.

Based on our findings, we confirm that Boa can be a powerful data mining tool, which can be used for a variety of research, alone and with usage of other software, like Weka, as demonstrated in Section 2.4.

References

- [1] Iowa State University of Science and Technology: The Boa Programming Guide. <http://boa.cs.iastate.edu/docs/>, 2015, accessed: October 18, 2015.
- [2] R. Dyer, H. A. Nguyen, H. Rajan, T. N. Nguyen. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 422–431, IEEE Press, 2013.
- [3] R. Dyer, H. Rajan, H. A. Nguyen, T. N. Nguyen. Mining billions of fast nodes to study actual and potential usage of java language features. In: Proceedings of the 36th International Conference on Software Engineering. pp. 779–790, ACM, 2014.
- [4] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, D. Poshyanyk. License usage and changes: A largescale study of java projects on github. In: The 23rd IEEE International Conference on Program Comprehension, ICPC, 2015.
- [5] Iowa State University of Science and Technology: Example Boa Programs, <http://boa.cs.iastate.edu/examples/>, 2015, accessed: October 11, 2015.
- [6] A. Patalas, W. Cichowski, M. Malinka, W. Stepniak, P. Mackowiak, L. Madeyski. Appendix to Software Metrics in Boa Large-Scale Software Mining Infrastructure: Challenges and Solutions, 2016, <http://madeyski.e-informatyka.pl/download/PatalasEtAl16Appendix.pdf>
- [7] Iowa State University of Science and Technology: Boa. Mining Ultra-Large-Scale Software Repositories. Dataset Statistics, <http://boa.cs.iastate.edu/stats/>, 2015, accessed: October 18, 2015.
- [8] Java research software, source code for metrics and statistical tests, <https://github.com/Aknilam/metrics-research-software>
- [9] Iowa State University of Science and Technology: Boa. Mining Ultra-Large-Scale Software Repositories. Client API, <http://boa.cs.iastate.edu/api/>, 2015, accessed: October 18, 2015.
- [10] S. R. Chidamber, C. F. Kemerer. A metrics suite for object oriented design, IEEE Transactions on Software Engineering 20(6), pp. 476–493, 1994.
- [11] F.B. e Abreu. Design quality metrics for object-oriented software systems. ER-CIM News 23, 1995.
- [12] L. Breiman. Random forests. Machine Learning 45(1), pp. 5–32, 2001.

- [13] M. Jureczko, L. Madeyski. Towards Identifying Software Project Clusters with Regard to Defect Prediction. In: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. pp. 9:1–9:10. PROMISE '10, ACM, New York, USA, 2010.
- [14] M. Jureczko, L. Madeyski. Cross-project defect prediction with respect to code ownership model: An empirical study. *e-Informatica Software Engineering Journal* 9(1), pp. 21–35, 2015.
- [15] L. Madeyski, M. Jureczko. Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study. *Software Quality Journal* 23(3), pp. 393–422, 2015.
- [16] M. Jureczko, L. Madeyski. A review of process metrics in defect prediction studies. *Metody Informatyki Stosowanej* 30(5), pp. 133–145, 2011.
- [17] Iowa State University of Science and Technology: Example Boa Programs, <http://boa.cs.iastate.edu/examples/>, 2015, accessed: October 18, 2015.

Chapter 9

How to Improve Linking Between Issues and Commits for the Sake of Software Defect Prediction?

1. Introduction

Bug predictions and defect predictions can save a lot of money which otherwise would be spent on bug fixes. Commit logs and bug reports are very often not linked with each other [1] although those links can provide very valuable information which can help in software defect predictions and project evolution. According to bugs prone – by counting the number of bug reports that are matched with them. For the project evolution, those links can generate defect data – for example the number of defects related to various classes in a project. As a result, it is possible to develop defect prediction models for software projects, e.g., [3, 4, 5].

The other problem is misclassification between bugs and non-bugs – many issues which are classified as 'bugs' refer to maintenance, refactoring or enhancements. Regarding to one of the reports regarding how misclassification impacts bug prediction [1] this problem is very common. Authors of the article [1] have conducted a manual examination of more than 700 issue reports of five open source projects. Their result revealed that 33,8% bug reports were misclassified – being not code fixes, but rather new features, refactorings or documentation updates. There are many simple approaches in matching links with issues, mainly based on simple textual matching. However, there are also three promising approaches (ReLink, MLink and RCLinker) which are based on repository changes and features extraction from bugs and issue trackers

metadata. One of the approaches (RCLinker) is based on the machine learning tool which is described in the section below.

In Section 2 we have described those three approaches to link issues with commits. Defect Prediction in Software Systems (DePress) [6] Extensible Framework allows building workflows in graphical manner. DePress is based on the KNIME project. The main aim of the DePress Framework is the support for empirical software analysis. It allows collecting, combining and analysing data from various data sources like software repositories or software metrics.

Our work provides the following contributions:

- 1) We wanted to find a way to link effectively commit logs and bug reports. That is why we have decided to use modified by us RCLinker approach. To achieve this, we have decided to use Defect Prediction in Software Systems and implement our approach of the RCLinker algorithm as a new node in the workflow. We used machine learning tool.
- 2) To validate and check our approach we have tested it on the three projects – two open source projects by Pivotal Software: Spring Data Redis, Spring OSGi and the proprietary commercial project provided by Capgemini.
- 3) The RCLinker approach uses metadata and textual features. We have proposed new features based on the JIRA metadata to check if they will improve the results.

2. Related Work

There are many approaches to linking issues with commit. We have reviewed literature using snowball sampling, described by Wohlin in [7]. Descriptions of the relevant articles are presented in the subsequent subsections.

2.1. ReLink: recovering links between bugs and changes

ReLink [8] is the simplest algorithm on which we will base our work. Traditional approaches for linking issues with commits presuppose that developers are on three properties:

- 1) Time interval – it is a time difference between commit date and issue modification date. After each fix, developer must update issue in tracking system, so the time difference will be small.
- 2) Issue owner and commit author – if issue owner is the same as commit author, this issue is probably connected with the commit
- 3) Text similarity – commit message should be similar to the issue description if they are linked. To normalize text in issue and commit message, ReLink uses the following techniques: removing stop-words, stemming and using synonyms – for example, change "additional" to "extra".

ReLink has a "learning" phase. To learn its model we must follow these steps:

- 1) Assign a very small value to time interval.
 - (a) Assign a very small value to the text similarity threshold.
 - (b) Discover links with traditional heuristics for given time interval and similarity threshold, then count number of discovered links and then calculate F-measure using metrics like "Percent of commits that fixes bugs" (more possibilities are below).
 - (c) Increase text similarity threshold a little bit
 - (d) Repeat steps 3 to 4 until we reach a maximum value of threshold
- 2) Increase time interval a little bit.
- 3) Repeat steps 3 to 6 until we reach a maximum value of time interval.
- 4) Choose threshold for two "properties" with the highest F-measure.
- 5) Return threshold and time interval.

To start discovering new connections, we must run two algorithms to get proper criteria and then ReLink will "learn" these criteria. After that, ReLink checks links that fulfils criteria. After checking all links, ReLink returns its list.

ReLink discovers up to 26% more links than the traditional approach [8]. It is often used with the following metrics: percent of commits that fixes bugs, percent of files with defects and average time of bug fixing.

2.2. MLink: multi-layered approach for recovering links between bug reports and Fixes

MLink [9] is a multi-layered approach to automatically recover issue links. In comparison to ReLink, it is not only based on the terms-linking method but it checks the changes in the code repository too and tries to link them with the issues metadata.

MLink uses cascading layers – each layer has a detector with its own set of textual and code features. The layers' input is a filtered set of the candidate links which comes from the previous layer – it means that each detector can be used as a filter. It reduces the amount of the links and passes the set to the next layer. Layers which have filters with higher levels of confidence on accurate detection are applied at earlier levels.

This model consists of six detectors:

- 1) **Pattern-based detector** – this is similar to ReLink approach – issues metadata and commits logs messages are checked if they contain some typical patterns such as 'fix the issue ...', 'fix the bug ID...' etc.
- 2) **Filtering layer** – the remaining links from the previous layer are analyzed if they violate time constraint – it means that the commit time for the fix must be between open and close time of the corresponding issue.
- 3) **Patch-based detector** – it extracts the patch code recommended by the bug reporters or people who have mentioned it in the issues comments.
- 4) **Name-based detector** – it detects if the entities or other components mentioned in the issue are the same as these which are in the commit log.
- 5) **Text-based detector** – it is similar to the previous layer but extracts comments in the changed code to and tries to link them with the issue metadata.
- 6) **Association-based detector** – it is the last layer which is used if the text used in the texts or entities names cannot be matched with the issue (the texts are not similar). It uses association strengths between the terms in the issue and the entity names.

MLink is better than ReLink because it checks and compares not only terms but changes in the code repository too. It achieves high accuracy level: F-score: 87-93%, recall: 85-90%, precision: 82-97% as outlined in [8].

2.3. RCLinker: Automated Linking of Issue Reports and Commits Leveraging Rich Contextual Information

RCLinker's [2] authors discovered, that many commits are not containing relevant information in commit messages. It means that if we want to improve linking issues with commits, we must use other contextual information.

RCLinker uses ChangeScribe [10] to generate additional messages about commits. ChangeScribe adds information in following format (real example from [2]):

This change set is mainly composed of:

1. Changes to package org.apache.solr.common.cloud:

(a) Modifications to ClusterState.java:

i. Remove an unused functionality to get shared

Messages, created by ChangeScribe, are then appended to each commit message. RCLinker also uses other contextual information like commit date, issue update date, issue comments' date.

RCLinker uses machine learning – trained Random Forest. Authors defined 9 text features (which are basing mostly on cosine distance between texts) and 11 metadata features (which are basing mostly on issue, commit and comments dates). We use this features to train Random Forest.

Usage of RCLinker is divided into two phases:

- 1) Learn phase – extending commit messages with ChangeScribe, extracting features (T1 – T9 and M1 – M11) and training model with i.e. Weka implementation of Random Forest.
- 2) Production phase – extending commit message with ChangeScribe, extracting features (T1 – T9 and M1 – M11) and using on created model to choose proper issues for commit.

RCLinker is much better in case of very poor developers' commit messages. It has approximately 136 % better results of F-measure than MLink, however precision is lower than in MLink.

We have also checked articles that are citing [2, 8] or [9]. Most of them are not related with linking issues with commits.

Empirical Evaluation of Bug Linking [11] is an empirical evaluation with benchmark of ReLink algorithm. It does not propose any new tool. However this article shows that usage of ReLink is reasonable.

In *The Missing Links: Bugs and Bug-fix Commits* [12] there is an analysis of problems with issue-commit linking. Authors used Linkster tool and expert knowledge to check 493 commits and link them to issues. Despite this work did not propose any new tool or algorithm, it is a good article to understand problems in linking issues with commits.

2.4. When do changes induce fixes?

This article [13] describes one of the simplest algorithms which we use in our approach. In this case it is described how to link bugs from the bug database with commits. This method is quite simple – every commit message is split into a stream of tokens (syntactic analysis). Each token could be one of the items: bug number (it is based on a simple regex), a plain number, a keyword such as fixed, defects etc. and a word. After that, the syntactic confidence is being counted – it is always an integer number between values 0 to 2.

This linking method is based on a semantic analysis too. There is also a score if some of the following conditions were resolved: the bug has been resolved as fixed at least once, the bug description is used in the commit message, the author of the commits has been assigned to it or one or more files affected by the commit has been attached to the bug.

In our approach we use pattern matching and semantic analysis too.

3. Experimental Setup

In this section we want to describe why we have decided to use RCLinker approach. According to *MLink* article [9], MLink is better than ReLink by 6-11% in F-score, 4-13% in recall, and 5-8% in precision. In *RCLinker* article [2] authors sustain that RCLinker has gained far much better results in F-measure by 138.66% in comparison to MLink. That is why we have decided to use RCLinker approach.

3.1. Research questions

- RQ1: How effective RCLinker is in recovering missing links between issues and commits? *In this RQ we will check how effective solutions from literature are.*
- RQ2: Is it possible to pick versatile machine learning settings giving good effects for all kinds of projects? *In this RQ we will check how RCLinker's results can be changed when we adjust classifiers' settings. We will try to check if it is possible to gain better results than original authors.*
- RQ3: Will RCLinker will be effective on projects with various difficulty levels? *We will run defect prediction on datasets that vary on number of issues that could be matched with commit logs.*
- RQ4: How to improve RCLinker algorithm? *We will add new metrics based on Jira metadata and evaluate on various machine learning classifiers.*

3.2. Datasets

The research will be performed using two open source projects by Pivotal Software: Spring Data Redis, Spring OSGi and the proprietary commercial project provided by Capgemini.

To check commits and issue linking we have looked through Git history of selected projects to find out if they contain Jira ticket numbers.

Outlined projects were chosen by discovering Spring's projects catalogue. We chose projects which were created recently. The projects were compared with each other mindng commits coverage with Jira tags and overall commits number. This dataset will be split into two parts. First one will be used as training set, second one will have its Jira tags removed and used for validation of output.

Spring OSGi will be used as dataset with higher complexity. This data set is bigger and not fully tagged with Jira issues. Specific thing for Spring OSGi commit history is tagging multiple commits with the same Jira issue number. It is two times bigger than Spring Data Redis – consists of over two thousands commits, while Spring Data Redis of around one thousand.

Commercial project provided by Capgemini will be used in final development of the algorithm. The data set used for the research comes from a system produced for one of the biggest automotive companies in Europe and it covers all aspects of car purchasing. The project is developed using agile methodologies. Support and bug fixing is hierarchically organised using Kanban technique as described in [14].

3.3. *Knime workflow description*

Our main goal is to implement a new DePress plugin. To supply datasets for the plugin (issues from JIRA and commit logs from GitHub) we needed to prepare workspace and provide links to JIRA and GitHub repository. More detailed information about reproduction (i.e. detailed steps of installation) can be found in Appendix A.

3.4. *Metrics – model input*

We will use two categories of metrics: based on commit message and based on commit metadata. Metrics are similar to those used in *RCLinker* article [2]. We have used them as a model input – independent predictors. As dependent variable we predict if the given pair commit-issue is a true link or not. For a list of notations, used in metrics table, please see Table 1. We will use metrics described in Table 2.

Metrics J1a, J1b, J1c, J1d, J2a, J2b, J3a, J3b, J3c are metrics designed by us, which are based on JIRA changes and metadata.

Model input consists of two additional indicators: *realLink* – valued as 1 if given pair of commit-issue exists in golden set, otherwise 0, and *undersampled RealLink*, which is output of undersampling process described in *RCLinker* article [2]. Golden set is extracted from version control system repository by traversing all the existing commit descriptions and matching issue tracking IDs in them. If such ID is found in description of commit, it is considered as a part of golden set.

During first phase of experiment, we have learned model using Random Forest. Next, we have tried to improve results using also the following classifiers from Weka library: MultilayerPerceptron, BayesNet, NaiveBayes, SGD,

AdaBoostM1, RealAdaBoost and changing default parameters' values to get better results. Unfortunately it did not improve the results, so we have decided to use Random Forest classifier.

Table 1. List of notations used in metrics description

Msg	Human-written commit message
Csmg	Commit message generated by ChangeScribe
cmtDate	Commit date
summary	Summary of an issue
Desc	Description of an issue
Prio	Priority of an issue
noCom	Number of comments in issue
com _i	Comment (1 ≤ com _i ≤ noCom)
words(D)	Number of distinct words in document D
+	text concatenation
reportedDate	Report date of an issue
updatedAt	Last update date of an issue
date (commi)	Date of i th comment in issue

3.5. Prediction model and measures

During experiment we have used a model to predict if the given issue should be linked with the given commit. For each pair (issue, commit) we have analysed if there is a link between them or not.

As an output of program returns a list of linked issues with commits – pairs (*issue*, *commit*). We have evaluated the result in the matter of measures such as precision, recall and F-measure.

4. Results

In this section are presented results which we have achieved by using RCLinker approach without the ChangeScribe tool. In the first two tables can be found results for open source projects, in the last one – for the commercial Cag Gemini project.

Table 2. Used textual and metadata metrics

Text features	
T ₁	$\cos(\text{summary} + \text{desc} + \text{COM}_i, \text{msg} + \text{csmmsg})$
T ₂	$\text{average}(\cos(I_a, C_b)), I_a \in \{\text{summary}, \text{desc}, \text{COM}_i\}, C_b \in \{\text{msg}, \text{csmmsg}\}$
T ₃	$\text{max}(\cos(I_a, C_b)), I_a \in \{\text{summary}, \text{desc}, \text{COM}_i\}, C_b \in \{\text{msg}, \text{csmmsg}\}$
T ₄	T_2/T_3
T ₅	T_2/T_1
T ₆	$ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) \cap \text{words}(\text{msg} + \text{csmmsg}) $
T ₇	$\frac{ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) \cap \text{words}(\text{msg} + \text{csmmsg}) }{ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) \cup \text{words}(\text{msg} + \text{csmmsg}) }$
T ₈	$\frac{ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) \cap \text{words}(\text{msg} + \text{csmmsg}) }{ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) }$
T ₉	$\frac{ \text{words}(\text{summary} + \text{desc} + \sum \text{COM}_i) \cap \text{words}(\text{msg} + \text{csmmsg}) }{ \text{words}(\text{msg} + \text{csmmsg}) }$
Metadata features	
M ₁	Number of changed files in the commit that are mentioned in the issue text
M ₂	$\frac{M_1}{\text{noCom} + \text{prio}}$
M ₃	1 if the issue reporter is also the committer. Otherwise, 0
M ₄	1 if the committer posts comments in the issue. Otherwise 0
M ₅	1 if the commit date is between the report date and the last updated date of the issue. Otherwise, 0
M ₆	$\text{cmtDate} - \text{reportedDate}$
M ₇	$\text{updatedDate} - \text{cmtDate}$
M ₈	M_6/M_7
M ₉	$\min_{1 \dots \text{noCom}} \text{cmtDate} - \text{date}(\text{com}_i) $
M ₁₀	$ \text{cmtDate} - \text{date}(\text{COM}_{\text{noCom}}) $
M ₁₁	$ \text{cmtDate} - \text{date}(\text{COM}_{\text{noCom}-1}) $
JIRA features	
J _{1a}	1 if issue status is RESOLVED or CLOSED. Otherwise 0
J _{1b}	1 if issue status is IN PROGRESS. Otherwise 0
J _{1e}	1 if issue status is REOPENED. Otherwise 0
J _{1d}	1 if issue status is OPEN. Otherwise 0
J _{2a}	1 if commit and issue are similar (classes from commit and from issue description). Otherwise 0
J _{2b}	1 if commit and comment are similar (classes from commit and classes from comment - for example stacktraces etc.). Otherwise 0
J _{3a}	1 if assignee from the issue is the same as the committer. Otherwise 0
J _{3b}	1 if issue reporter is the same as the committer. Otherwise 0
J _{3e}	1 if author of the issue comment is the same as the committer. Otherwise 0

4.1. Spring Data Redis

Results for Spring Data Redis project are presented in Table 3.

Table 3. Spring Data Redis evaluation results

Description	Recall	Precision	F-measure
no cross validation, 2 nearest neighbours	0.63	0.17	0.27
10 iterational cross validation, 10 nearest neighbours	0.52	0.10	0.16
10 iterational cross validation, 2 nearest neighbours	0.48	0.38	0.42
15 iterational cross validation, 2 nearest neighbours	0.46	0.37	0.40
10 iterational cross validation, 1 nearest neighbour	0.41	0.54	0.47
10 iterational cross validation, 0 nearest neighbours	0.38	0.79	0.51

The best achieved results processing Spring Data Redis project were recall: 0.38, precision 0.79 giving F-measure at point of 0.51.

During evaluation it turned out that producing nearest neighbours actually does not impact results in the positive way. It makes recall slightly rise, but with cost of huge precision drop.

Using cross validation instead of random splitting data set into two fixed-size subsets improved the result. When no cross validation was used, with two nearest neighbours generated, there were F-measure equal to 0.27. With the same nearest neighbour setting and cross validation used, the F-measure raised to level of 0.42.

Increasing number of cross validation iterations did not bring significant improvement comparing to extended computation time needed to process data. Adding 5 iterations enhanced F-measure by 0.02.

4.2. Spring OSGi

Results for Spring OSGi project are presented in Table 4.

Table 4. Spring OSGi evaluation results

Description	Recall	Precision	F-measure
10 iterational cross validation, 1 nearest neighbour	0.19	0.22	0.20
10 iterational cross validation, 0 nearest neighbours	0.17	0.58	0.26
15 iterational cross validation, 0 nearest neighbours	0.19	0.60	0.29

In comparison to Spring Data Redis, Spring OSGi has far more less correct commits descriptions. While Spring Data Redis has almost all of them well described, Spring OSGi has more or less 50%. That is why, the results are much worse. Slightly better result we got by increasing the number of iterations.

4.3. Capgemini project

Results for the commercial Capgemini project are presented in Table 5.

Table 5. Capgemini project evaluation results

Description	Recall	Precision	F-measure
10 iterational cross validation, 2 nearest neighbours	0.58	0.41	0.48
10 iterational cross validation, 1 nearest neighbour	0.54	0.56	0.55
15 iterational cross validation, 0 nearest neighbours	0.49	0.86	0.62
10 iterational cross validation, 0 nearest neighbours	0.49	0.88	0.63

Proprietary commercial project provided by Capgemini has quite good results. First of all the dataset with commits and issues was not too big – this was a period of 6 months. The other thing why results are good is caused by good described commits' messages – about 95% has good commit description.

In comparison to Spring Data Redis – the best results were achieved when nearest neighbours equalled 0. The recall and F-measure raised significantly: recall from 0,56 to 0,88 and F-measure from 0,55 to 0,63.

5. Discussion

In this section, we have described why we have not used ChangeScribe in our implementation of the RCLinker algorithm. ChangeScribe caused many performance and implementation problems which are described below.

5.1. General discussion

As we can see in the results, RCLinker algorithm performs the best in Capgemini proprietary project. Also results for the Spring Data Redis are still quite good, however they are much worse than in the original RCLinker [2] approach.

5.2. Problems

During implementation of RCLinker algorithm we have encountered many performance and implementation problems.

First problem was with executing ChangeScribe in non-eclipse environment. ChangeScribe was not describing properly all changes. We wrote to ChangeScribe's authors and created an issue on the GitHub repository. They helped us and gave access to special, modified version of ChangeDistiller.

Second problem was memory complexity of ChangeDistiller. We have tried to run application with various heap sizes, however even 15 GB of RAM was not enough.

5.3. Validity threats

Golden set is extracted from version control system repository by pattern matching potential issue IDs in commits' description. This is a threat to validity since there may be some mistakenly tagged descriptions and the

golden set acquired this way may not be full. There is no other fully credible way of achieving such a golden set in projects evaluated by us. Manual creation of golden set would be extremely time consuming and would not plausibility of it would be arguable as well.

6. Conclusions

Connections between issues and commits are very valuable in defect prediction. Unfortunately commit logs are often missing clear disclosure of these links.

Our implementation of RCLinker was able to achieve results with F-measure equal to 0.62 on the commercial project. This is promising result, but is not enough for enterprise use of this tool. The result indicates need for further development of the algorithm itself.

Due to problems with ChangeScribe results are inconclusive. Comparing to the RCLinker paper our implementation achieves significantly worse results. There is a possibility, that using ChangeScribe, the results would be comparable to original RCLinker's evaluation.

We wanted to check which features are significant and important for the results. T1, T2 and T3 were good indicators when it comes to textual relevance between commits and issues. Features T4 and T5 are normalized forms of T1-T3 respectively and that is why we supposed that they may be not very essential — we have checked this assumption using different classifiers described below. Features T6-T7 were used to compute the number of common words between issue and commit bringing new information to the classifier so they are relevant for classifier. Because T8 and T9 are the ratio of T6 to the number of distinct word in issue and commit we consider them as not useful. Metadata features M9, M10, M11 were based on the comments and dates between them and did not provide valuable information for machine learning algorithms. After evaluation we consider JIRA features J1a, J1b, J1c, J1d, J2a, J2b, J3a, J3b, J3c as not relevant, as they were not improving result of machine learning.

Concluding the revision we decided to leave significant metrics T1, T2, T3, T6, T7, M1-M8 and not use features: T4, T5, T8, T9, M9, M10, M11, J1a,

J1b, J1c, J1d, J2a, J2b, J3a, J3b, J3c. We have tested it on the Spring Data Redis dataset. The results were comparable: Recall: 0,36, Precision: 0,79, F-measure: 0,49. With the previous features set we got: Recall: 0,38, Precision: 0,79 and F-measure: 0,51.

We have also tested our implementation using the following classifiers from Weka library: MultilayerPerceptron, BayesNet, NaiveBayes, SGD, AdaBoostM1, RealAdaBoost and Random Forest. However, Random Forest, used in original paper [2], gave us the best results. We have checked how various parameters will change the results. For Random Forest, the best parameter set is: Max Depth: unlimited, number of trees: 10.

7. Future works

We were not able to gain such a good results as described in the article about RCLinker [2]. We suppose that a tool which will be similar to ChangeScribe can improve the results. Additional features which were based on JIRA metadata did not improve the results. It is likely that a tool which generates additional messages about commits will give significant information about changes in the repository code — it will be possible to create a new set of features. The decision to create a new tool instead of using ChangeScribe is associated with the problems described in the subsection 5.2.

References

- [1] K. Herzig, S. Just, and A. Zeller. It's Not a Bug, It's a Feature: How Misclassification Impacts Bug Prediction, In: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, Piscataway, NJ, USA, pp. 392–401, IEEE Press, 2013.
- [2] T.-D. B. Le, M. Linares-Vásquez, D. Lo, and D. Poshyvanyk, RCLinker. Automated Linking of Issue Reports and Commits Leveraging Rich Contextual Information, In: Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, ICPC '15, Piscataway, NJ, USA, pp. 36–47, IEEE Press, 2015.
- [3] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu. A General Software Defect-Proneness Prediction Framework, IEEE Transactions in Software Engineering, Vol. 37, pp. 356–370, 2011.

- [4] L. Madeyski and M. Jureczko. Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study, *Software Quality Journal*, Vol. 23, no. 3, pp. 393–422, 2015.
- [5] M. Jureczko and L. Madeyski. Cross-project defect prediction with respect to code ownership model: An empirical study, *e-Informatica Software Engineering Journal*, Vol. 9, no. 1, pp. 21–35, 2015.
- [6] L. Madeyski and M. Majchrzak. Software Measurement and Defect Prediction with De-Press Extensible Framework, *Foundations of Computing and Decision Sciences*, Vol. 39, no. 4, pp. 249–270, 2014.
- [7] C. Wohlin. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering, In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14*, New York, NY, USA, pp. 38:1–38:10, ACM, 2014.
- [8] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung. ReLink: Recovering Links Between Bugs and Changes, In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11*, New York, NY, USA, pp. 15–25, ACM, 2011.
- [9] A. T. Nguyen, T. T. Nguyen, H. A. Nguyen, and T. N. Nguyen. Multi-layered Approach for Recovering Links Between Bug Reports and Fixes, In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE '12*, New York, NY, USA, pp. 63:1–63:11, ACM, 2012.
- [10] ChangeScribe, <https://github.com/SEMERU-WM/ChangeScribe>, 2016.
- [11] T. F. Bissyandé, F. Thung, S. Wang, D. Lo, L. Jiang, and L. Réveillère. Empirical Evaluation of Bug Linking, In: *17th European Conference on Software Maintenance and Reengineering (CSMR)*, pp. 89–98, 2013.
- [12] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein. The Missing Links: Bugs and Bug-fix Commits, In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '10*, New York, NY, USA, pp. 97–106, ACM, 2010.
- [13] J. Sliwerski, T. Zimmermann, and A. Zeller. When Do Changes Induce Fixes?, In: *Proceedings of the 2005 International Workshop on Mining Software Repositories, MSR '05*, New York, NY, USA, pp. 1–5, ACM, 2005.
- [14] M. Majchrzak and L. Stilger. Experience Report: Introducing Kanban Into Automotive Software Project, From Requirements to Software: Research and Practice, *Scientific Papers of the Polish Information Processing Society Scientific Council*, pp. 15–32, 2015.

Chapter 10

Defect Prediction with Bad Smells in Code

1. Introduction

Among different aspects of software defect prediction process, one of the key elements is proper selection of metrics for training and verification dataset preparation. Most popular data is source code metrics [6, 11], but also different types of metrics are considered effective in term of defect prediction, such as design metrics [24], change metrics [21], mining metrics [22] or process metrics [18, 13].

1.1. Related work and goal

Separate group of design metrics are metrics based on code smells, also known as bad smells or code bad smells. The term was formulated by Kent Beck in 2006 [1]. The concept was popularized by Martin Fowler in his book *Refactoring. Improving the structure of existing code* [5]. Kent Beck was a co-author of the chapter on code smells.

Kent Beck on his website explains the idea of code smells [1]:

Note that a Code Smell is a hint that something might be wrong, not a certainty. A perfectly good idiom may be considered a Code Smell because it's often misused, or because there's a simpler alternative that works in most cases. Calling something a Code Smell is not an attack; it's simply a sign that a closer look is warranted.

Due to nature of code smells described above, there is ongoing discussion if code smells could be used effectively in quality assurance in code

development [27, 26]. Major motivation for this research was to investigate, if code smells can improve software defect prediction.

In industrial software development, only Holschuh et al. investigated code smells metrics effectiveness in defect prediction process for Java programming language [7]. No code smells metrics for defect prediction in .NET oriented industrial software projects are known to authors. Thus, we decided use long-term defect prediction research project run in Volvo Group [9, 10] as an occasion for conducting an experiment with introduction of bad smells based metrics to prediction process and observe the results, if they improved prediction effectiveness or not:

RQ: How Code Bad Smells based metrics impact defect prediction in industrial software development project?

1.2. Research environment: Industrial software development project

Project, on which the study was conducted, is a software development of critical industry system used in Volvo Group vehicle factories called PROSIT+. It is created based on client-server architecture. The main functionality of PROSIT+ system is: programming, testing, calibration and electrical assembly verification of Electronic Control Units (ECUs) in Volvo's vehicle production process.

PROSIT+ system consists of few coexisting applications. The most important one, desktop application “PROSIT Operator”, communicates in real time with a mobile application, located on palmtop computer used by vehicle factory workers to transfer all production related information to a local server. The server is responsible for storage and distribution of configuration-, system- and product-related data. Such communication can generate extremely heavy data transfer loads in large factories, when more than 100 mobile applications are used. Other application include: “PROSIT Designer”, “PROSIT Factory Manager” and web application “PROSIT Viewer”. All of them are also connected to the same server.

Development of each PROSIT+ version lasts one year. After this period software is released to the end-user. As this period of time is connected to factory production cycle it cannot be fastened or postponed.

All applications within PROSIT+ system were developed using Microsoft .NET technology and Microsoft Visual Studio as the integrated development environment. For version control purposes, Microsoft Team Foundation Server was used. Before release of version 11 of the PROSIT+ system, IBM ClearQuest was used for software defect management. Until the development of version 11, Team Foundation Server was used for defect tracking.

Project lacks of bottlenecks described by Hryszko and Madeyski [8], which could hinder or prevent from applying defect prediction process. However, we observed relatively high number of naming issues in the project. Main reason of that situation we consider high maturity of the software system – over the time, naming conventions have changed. We consider naming issues as negligible problem and we will exclude them from the further investigation.

2. Research Process

Defect prediction was already an ongoing process in investigated project. It used SourceMonitor software as metric source and as prediction tool – KNIME-based DePress Extensible Framework proposed by Madeyski and Majchrzak [19]. This tool, based on KNIME [17], provides with a wide range of data-mining techniques, including defects prediction, in various IT projects, independently of technology and programming language used. We will also use KNIME/DePress for purpose of our research.

To investigate the possible impact of code-smell metrics on defect prediction, we developed the following plan to follow:

- 1) Generate metrics from SourceMonitor;
- 2) Generate code smells metrics from CodeAnalysis;
- 3) Parse results from CodeAnalysis and merge them with metrics from Source-Monitor.
- 4) Link check-ins to defects;
- 5) Link classes from check-ins to defects (the assumption is that if a class was changed while fixing a defect, that class was partially or fully responsible for that defect);

- 6) Merge list of classes with merged metrics from CodeAnalysis and SourceMonitor;
- 7) Use different software defect prediction approaches combinations to select optimal prediction set-up for evaluation purposes;
- 8) Divide PROSIT+ code into 20 sub-modules and run prediction model training and evaluation using data from each module separately;
- 9) Collect and interpret the results.

2.1. SourceMonitor as basic metrics source

Defect prediction process in PROSIT+ is based on metrics that are gathered using SourceMonitor tool [12]. That tool performs static computer code analysis on complete files and extracts 24 different kinds of metrics. Example metrics extracted are:

- Lines of code,
- Methods per class,
- Percentage of comments,
- Maximum Block Depth,
- Average Block Depth.

2.2. CodeAnalysis tool as code smells metrics source

In our experiment, we decided to use Microsoft CodeAnalysis tool to gather code smells metrics. Primary deciding factor was cost: CodeAnalysis tool is delivered as a part of Microsoft Visual Studio software development suite for .NET based projects. Thus, there were no additional costs of introduction of this tool into the investigated software development project.

CodeAnalysis for managed code analyzes managed assemblies and reports information about the assemblies, such as violations of the programming and design rules set forth in the Microsoft .NET Framework Design Guidelines [20].

According to documentation, there are approximately two hundred rules in CodeAnalysis [20], triggering 11 kinds of warnings (Table 1). Tool can be

run from command line and results are then stored in an .xml file, that can be later parsed and analyzed further.

Table 1. Bad smell warnings in CodeAnalysis

Bad smell warning	Area covered
Design	Correct library design as specified by the .NET Framework Design Guidelines
Globalization	World-ready libraries and applications
Interoperability	Interaction with COM clients
Maintainability	Library and application maintenance
Mobility	Efficient power usage
Naming	Adherence to the naming conventions of the .NET Framework Design Guidelines
Performance	High-performance libraries and applications
Portability	Portability across different platforms
Reliability	Library and application reliability, such as correct memory and thread usage
Security	Safer libraries and applications
Usage	Appropriate usage of the .NET Framework

3. Results

We conducted our experiment by following the plan presented in previous section. Here we present the results.

3.1 Automatically generated code: observed anomaly, cause and solution

After analyzing the relation between numbers of reported code smells issues and file length metrics for complete software system, in datasets prepared basing on CodeAnalysis and SourceMonitor tools, we observed that different numbers of issues are reported for the same, large file length values (Figure 1). As considered software contains only small number of large files, we interpreted that as an anomaly: different total number of code bad smell issues were reported for the same files. After investigation, we found that in investigated system files with more than 1000 lines of code (LOC) are in most

cases generated automatically and contain more than one class for a file, while CodeAnalysis tool calculates number of issues metric per class. That discrepancy resulted in abnormal number of issue per file length relation: different number of issues values were collected for the same LOC values, because numbers of issues values were calculated for different classes located in the same files, identified by the same LOC value.

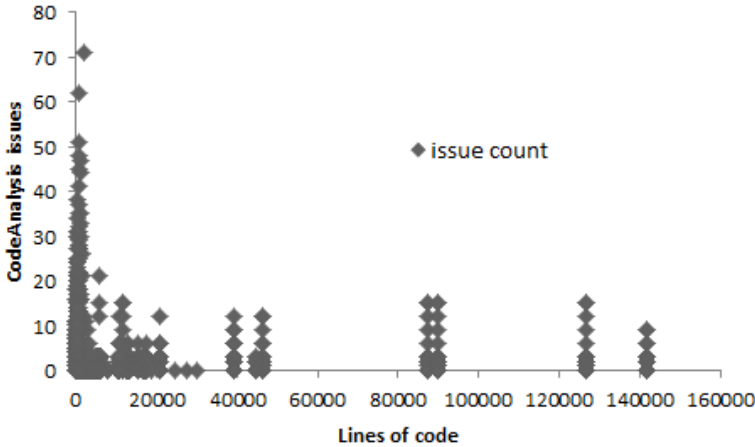


Figure 1. Anomalies in number of issues metric per file length (measured in LOC) relation, introduced by automatically generated code, later removed from analysis

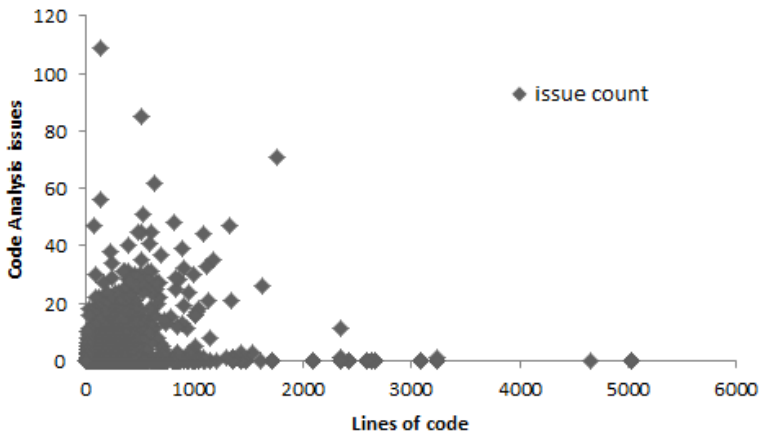


Figure 2. Number of issues metric per file length (measured in LOC) relation for investigated software, with automatically generated code removed

As automatically generated code files exist only for installation and deployment purposes and are not covered by tests and are not reachable for end-users of the system, we decided to consider them as a source of information noise and **we removed them from further analysis**. Number of issue per file length relation improved after that step (Figure 2).

3.2 Metrics breakdown difference: problem and solution

After a thorough investigation of the above problem, we found that different values of issue number metric for the same LOC metric was caused by the different metrics breakdown used by two tools selected for metric datasets generation: CodeAnalysis gathers data for every class while SourceMonitor for every file. When results from two tools were merged into single dataset, SourceMonitor metrics, fixed for each file, were artificially divided per each class in the file (Table 2).

Table 2. Example of dataset from first approach: single class per record (SourceMonitor metrics are artificially divided per each class in file)

File	Class	SourceMonitor LOC	CodeAnalysis Issues
File1.cs	Class1	33	3
File1.cs	Class2	33	20
File1.cs	Class3	33	6
File2.cs	Class4	30	15

To counteract against metric anomalies described in section 3.1, as well as against possible introduction of informational noise into the training dataset, we decided to change the approach and rearrange the datasets into single file metrics per record layout. To achieve this, metrics gathered by CodeAnalysis had to be aggregated (added; Table 3).

Table 3. Example of dataset from second approach: single file per record (CodeAnalysis metrics are artificially added)

File	Class	SourceMonitor LOC	Code Analysis Issues
File1.cs	Class1...3	100	29
File2.cs	Class4	30	15

3.3 Optimal prediction mechanism selection

To choose optimal prediction mechanism, we decided to test combination of different classifiers, feature selection and balance algorithms (Table 4) against two datasets: with- and without code bad smells metrics collected by CodeAnalysis tool.

Table 4. Combinations of different approaches

Classifier	Feature Selection	SMOTE	Bad smells metrics?
Naive Bayes	None	With	Present
Random Forest	Elimination	Without	Absent
PNN	Simulated Annealing		

We used SMOTE algorithm [4] to balance classes with defects and without them.

To select most important metrics from all available, as some of them should have seemingly little impact on the presence of true software defects, e.g. Efficient power usage warning (Table 1), we decided to use in our research two feature selection algorithms: KNIME's build-in reversed elimination greedy algorithm [16] and simulated annealing meta-heuristic algorithm by Kirkpatrick et al. [15] in form proposed by Brownlee [3].

As classifier, we used popular in defect prediction studies [6, 21, 14, 23] Naïve Bayes classifier and Probabilistic Neural Network (PNN), as well as Random Forest [2] classifier.

Results of testing combinations of above machine learning elements in favour of best prediction results are presented in Table 5. Two datasets – with and without code bad smells metrics included, were divided using stratified sampling method into two equal subsets, for training and evaluation purpose. Prediction models were evaluated using F-measure [25].

Highest F-measure value (0.9713) was observed for dataset with code bad smells used, when SMOTE algorithm and reversed elimination feature selection mechanism was used to select optimal subset for training and evaluation of Random Forest classifier. And such combination was selected for final evaluation of usage of code smells based metrics in defect prediction process.

Table 5. Results for optimal prediction set-up selection (defect-prone class)

Classifier	SMOTE	Feature selection	Bad smells metrics included?									
			No					Yes				
			F-meas.	TP	FP	TN	FN	F-meas.	TP	FP	TN	FN
Naive Bayes	NO	Annealing	0.1318	23	161	3330	142	0.1149	15	81	3410	150
		Elimination	0	0	1	3490	165	0	0	0	3491	165
		None	0.1314	31	276	3215	134	0.1389	40	371	3120	125
	YES	Annealing	0.5474	1497	482	3008	1994	0.586	1695	599	2891	1796
		Elimination	0.5961	1736	598	2892	1755	0.6106	1807	621	2869	1684
		None	0.5424	1476	475	3015	2015	0.5775	1657	591	2899	1834
PNN	NO	Annealing	0	0	0	3491	165	0	0	0	3491	165
		Elimination	0	0	0	3491	165	0	0	0	3491	165
		None	0	0	0	3491	165	0	0	0	3491	165
	YES	Annealing	0.7313	2187	303	3187	1304	0.7582	2335	333	3157	1156
		Elimination	0	0	0	3491	165	0.8051	2568	320	3170	923
		None	0.7253	2147	282	3208	1344	0.7333	2204	316	3174	1287
Random Forest	NO	Annealing	0.0963	9	13	3478	156	0.1538	15	15	3476	150
		Elimination	0.1587	15	9	3482	150	0.1405	13	7	3484	152
		None	0.1429	14	17	3474	151	0.1538	15	15	3476	150
	YES	Annealing	0.9551	3364	189	3301	127	0.9696	3407	130	3360	84
		Elimination	0.9654	3390	142	3348	101	0.9713	3435	147	3343	56
		None	0.9601	3383	173	3317	108	0.9696	3407	130	3360	84

3.4 Datasets evaluation: CodeAnalysis (bad smells metrics) against SourceMonitor

For final evaluation, if code bad smells-based metrics could be valuable for defect prediction purposes, we divided all available code, in considered industrial software development project, into 20 smaller, similar in size sub-modules (ca. 700 records after SMOTE oversampling). Greater fragmentation of system's code was not technically possible. For each sub-module we collected metrics using SourceMonitor or/and CodeAnalysis, to create different datasets:

- 20 datasets of SourceMonitor metrics only;
- 20 datasets of CodeAnalysis (code smells) metrics only;
- 20 datasets of combined metric: SourceMonitor + CodeAnalysis.

Additionally, each kind of datasets we decided to test against feature selection (FS) process. During the evaluation, we collected Accuracy and Cohen's kappa measures for overall results (Table 6), and F-measure and Recall for defect-prone classes (Table 7).

Table 6. Final results of datasets evaluation

Dataset	Measure	Mean	Std. deviation
SourceMonitor without FS	Accuracy	0.9422	0.0187
	Cohen's kappa	0.8844	0.0374
CodeAnalysis without FS	Accuracy	0.676	0.0451
	Cohen's kappa	0.3518	0.0904
SourceMonitor + CodeAnaly- sis w/o FS	Accuracy	0.9487	0.0226
	Cohen's kappa	0.8973	0.0453
SourceMonitor with FS	Accuracy	0.97	0.0122
	Cohen's kappa	0.9399	0.0245
CodeAnalysis with FS	Accuracy	0.8249	0.059
	Cohen's kappa	0.6497	0.1180
SourceMonitor + CodeAnaly- sis with FS	Accuracy	0.9791	0.0135
	Cohen's kappa	0.9582	0.027

3.5 Threads to validity

Conclusion validity. In our research, we tested 20 datasets collected from different software modules. More research using larger data set, collected from different sources is needed to confirm our findings.

Internal validity. We have used aggregation of CodeAnalysis metrics for each file, by adding metrics collected for each class. Such solution was introduced to solve metrics breakdown difference problem and make combination of two metric sources possible, however it could impact the final result of our research.

External validity. Our research is based only on metrics gathered from one software development project. Despite the fact, that we were able to collect 34 different metric kinds for 20 different program modules, we were still constrained by single environment: development team and its programming habits, programming language, tools used, etc. Because of this fact, more research is needed to verify our findings in other software development environments (contexts).

Table 7. Measures for records marked as defect-prone

Dataset	Measure	Mean	Std. deviation
SourceMonitor without FS	Recall	0.9608	0.0278
	F-measure	0.9433	0.0188
CodeAnalysis without FS	Recall	0.666	0.2961
	F-measure	0.6447	0.1157
SourceMonitor + CodeAnalysis w/o FS	Recall	0.9637	0.0303
	F-measure	0.9494	0.0228
SourceMonitor with FS	Recall	0.9824	0.0146
	F-measure	0.9704	0.012
CodeAnalysis with FS	Recall	0.8424	0.0542
	F-measure	0.8286	0.0559
SourceMonitor + CodeAnalysis with FS	Recall	0.9859	0.0206
	F-measure	0.9792	0.0136

4. Discussion

When selecting optimal defect prediction set-up for further verification if code smell-based metrics can improve prediction results, we observed that best result was achieved for dataset with bad smell metrics included (F-measure = 0.9713). However, for the same setup, but without code smells metrics, F-measure value was only by 0.0059 lower (Table 5) what makes the difference between SourceMonitor and CodeAnalysis results negligible. Final results collected from 20 different software sub-modules confirmed that statement: Average accuracy value for prediction based on dataset constructed basing on both sources was only by 0.0091 better than result for SourceMonitor-only based metrics (Average F-measure value difference = 0.0088), while standard deviation value was 0.0136. Worth noticing is drop of CodeAnalysis – only based prediction results, when feature selection (FS) process was removed from the experimental setup.

Results of our experiment of using code smells metrics in software defect prediction, show irrelevant – in our opinion – impact on effectiveness of the process, when basic dataset (SourceMonitor-based) was extended by CodeAnalysis metrics. because even if prediction effectiveness measures are

slightly higher, they stay within the limits of error. But when only CodeAnalysis-based metrics were used for prediction (without basic set of SourceMonitor-based metrics), such process resulted with high accuracy (0.8249) and F-measure (0.8286) results.

Thus, answering the research question: *How Code Bad Smells based metrics impact defect prediction in industrial software development project?* We want to state, that in industrial environment, such as PROSIT+ software development project, impact of code bad smells based metrics is negligibly small, and usage of CodeAnalysis-based metrics should not be considered useful, due to fact that additional effort needed for introducing code smell-based metrics to software defect prediction process is not compensated by relatively high increase of prediction effectiveness.

However, we observed surprisingly high effectiveness of prediction, when dataset based on CodeAnalysis only was used. Authors believe, that code bad smells can be effectively used for defect prediction process especially there, where other metrics are not available, or computing power is insufficient to handle large sets of different metrics (for example 24 kinds of metrics for SourceMonitor), while CodeAnalysis metrics set, used in our research, contained only 11 different kinds of metrics. Due these promising results, aspects of using code bad smells only based metrics in defect prediction processes should be investigated further.

References

- [1] K. Beck. Code Smell (2016), <http://c2.com/cgi/wiki?CodeSmell>, accessed: May 8, 2016.
- [2] L. Breiman. Random Forests. *Machine Learning* pp. 5–32, 2001.
- [3] J. Brownlee. *Clever Algorithms. Nature-Inspired Programming Recipes*, Jason Brownlee, 2011.
- [4] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence* pp. 321–357, 2002.
- [5] M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2006.
- [6] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell. A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering* 38(6), pp. 1276–1304, 2012.

-
- [7] T. Holschuh, M. Pauser, K. Herzig, T. Zimmermann, R. Premraj, A. Zeller. Predicting defects in SAP Java code: An experience report. In: ICSE-Companion 2009, 31st International Conference on Software Engineering, pp. 172–181, 2009.
- [8] J. Hryszko, L. Madeyski. Bottlenecks in Software Defect Prediction Implementation in Industrial Projects. *Foundations and Computing and Decision Sciences* 40(1), pp. 17–33, 2015, <http://dx.doi.org/10.1515/fcds-2015-0002>
- [9] J. Hryszko, L. Madeyski. Assessment of the Software Defect Prediction Cost Effectiveness in an Industrial Project. *Advances in Intelligent Systems and Computing* (accepted), 2016.
- [10] J. Hryszko, L. Madeyski, R. Samlik. Application of Defect Prediction-Driven Quality Assurance Methodology in Industrial Software Development Project, pre-print, 2016
- [11] N. Jaechang. Survey on Software Defect Prediction (2014), hKUST PhD Qualifying Examination
- [12] J.Holmes: SourceMonitor Site, 2016, <http://www.campwoodsw.com/sourcemonitor.html>, accessed: 2016.05.06
- [13] M. Jureczko, L. Madeyski. A Review of Process Metrics in Defect Prediction Studies. *Metody Informatyki Stosowanej* 30(5), pp. 133–145, 2011, <http://madeyski.e-informatyka.pl/download/Madeyski11.pdf>
- [14] T. M. Khoshgoftaar, A. S. Pandya, D. L. Lanning. Application of Neural Networks for Predicting Faults. *Annals of Software Engineering* 1(1), pp. 141–154, 1995.
- [15] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimization by Simulated Annealing. *Science* 220(13), pp. 671–680, 1983.
- [16] KNIME.COM AG: Backward Feature Elimination, 2016, https://www.knime.org/files/nodedetails/_mining_meta_mining_features_Backward_Feature_Elimination_Start_1_1_.html, accessed: June 28, 2016.
- [17] KNIME.COM AG: KNIME Framework Documentation, 2016, <https://tech.knime.org/documentation/>, accessed: May 6, 2016.
- [18] L. Madeyski, M. Jureczko. Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study. *Software Quality Journal* 23(3), pp. 393–422, 2015, <http://dx.doi.org/10.1007/s11219-014-9241-7>
- [19] L. Madeyski, M. Majchrzak. Software Measurement and Defect Prediction with Depress Extensible Framework. *Foundations of Computing and Decision Sciences*, pp. 249–270, 2014.
- [20] Microsoft: Code Analysis for Managed Code Overview, 2016, <https://msdn.microsoft.com/en-us/library/3z0aeatx.aspx>, accessed: May 6, 2016.
- [21] R. Moser, W. Pedrycz, G. Succi. A Comparative Analysis of The Efficiency of Change Metrics and Static Code Attributes for Defect Prediction. In: *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on*. pp. 181–190, 2008.
- [22] N. Nagappan, T. Ball, A. Zeller. Mining Metrics to Predict Component Failures. In: *Proceedings of the 28th International Conference on Software Engineering*. pp. 452–461, 2006.

- [23] R. W. Selby, A. Porter. Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis. *IEEE Transactions on Software Engineering* 14(12), pp. 1743–1756, 1988.
- [24] G. Succi, W. Pedrycz, M. Stefanovic, J. Miller. Practical Assessment of the Models for Identification of Defect-Prone Classes in Object-Oriented Commercial Systems Using Design Metrics. *Journal of Systems and Software* 65(1), pp. 1–12, 2003.
- [25] I. H. Witten, E. Frank, M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
- [26] M. Zhang, T. Hall, N. Baddoo. Code Bad Smells: a review of current knowledge. *Journal of Software Maintenance and Evolution: Research and Practice*, pp. 179–202, 2011.
- [27] M. Zhang, T. Hall, N. Baddoo, P. Wernick. Do bad smells indicate "trouble" in code? In: *DEFECTS '08 Proceedings of the 2008 workshop on Defects in large software systems*, pp. 43–44, ACM, 2008.

Chapter 11

Postgraduate Studies on Software Testing in Poland

1. Introduction: Tester as an IT Profession in Poland

IT managers constantly have a problem with testing. They do not know exactly what testing means, why it is essential in the software lifecycle and why it is worth to invest in it. Till now some IT people, when thinking about "testers", have in mind a low educated person who – by more or less randomly clicking on different fields in applications – tries to find some errors.

In the last few years, however, things become to change. More and more managers are aware that testing is not "clicking", but a formal, systematic process based on the mathematical background (see for example [1]). They understand that testing tries to answer a very important question: what is the actual risk related to the application under test. Managers start to treat testing as a technological investigation which allows us to gain some information about the quality of System Under Test. Technological – because it heavily relies on methods such as mathematics, graph theory, logic, tools (special programs), theoretical computer science, models, measuring etc. Investigation – because testers do it in a formal way.

Observing the market one can find that many organizations in Poland are looking for qualified testers – mostly in big cities (see Figure 1). It is interesting to observe the requirements for this position (see Figure 2). It is obvious that employers are looking for persons with good English knowledge (typically on a B2 knowledge level, standard for graduates in Poland). Most of these organizations are international, most IT documentation is in English. Moreover, English nowadays is a lingua franca in the IT world. We can also

understand that – especially regarding higher positions (Senior Tester, Test Manager) – employers are interested in some experience, but mostly in basic testing process, some tools like bug trackers, etc. From the data one can find that ISTQB certificate is an important basis which confirms the candidate's testing knowledge.

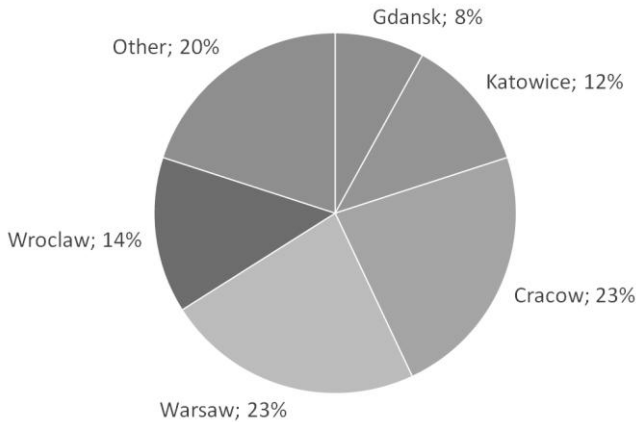


Figure 1. Demand for testers in different cities in Poland – based on Pracuj.pl portal; February 2016 (155 job advertisements)

It is very interesting that the minority of employers requires the education in technology (computer science, electronics) (38%). In our opinion it is because the organizations are aware of the fact that graduates with technical education prefer to work as developers, as they are usually paid much better than testers. Hence they try to omit this problem, replacing the requirement of „education” by requirement of “experience”. It may also be a result of – invalid in our opinion – belief that the main task of testers is just to act as a user, who is not necessarily technically educated. What is also interesting, there is a relatively small amount of requirements (a bit over 50%) for programming skills. In our opinion it comes from the fact that people with good programming skills prefer to work as developers than testers.

But computer science graduates may be reluctant in developing their careers in the quality area also for other reason. Namely, computer science curricula for undergraduate and graduate studies usually do not pay much attention to software testing or quality assurance. Much of the hours devoted to software development focus on programming (programming languages, data

structures, and algorithms). Usually there is only one course on software engineering (typically ca. 15-30 hours) covering all SE areas (requirements, software development lifecycles, software design, architecture, UX, documentation, IT project management etc.). Therefore, there are only one or two lectures on software testing. In result, the graduates have only a very limited knowledge on software quality. Their testing skills are usually related to the developers' tasks (frameworks for unit testing, Test Driven Development, Behaviour Driven Development etc.).

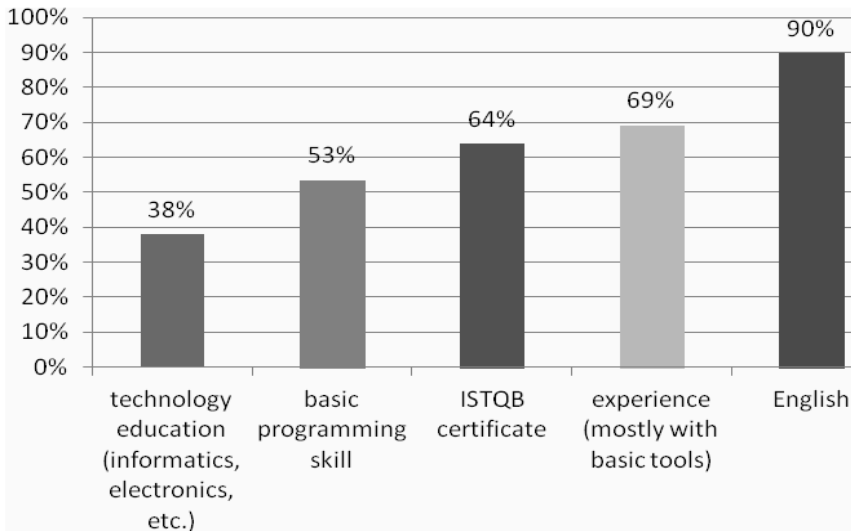


Figure 2. Requirements for tester positions – based on Pracuj.pl, February 2016 (155 job advertisements)

Jagiellonian University in Cracow is the first Polish university that offers – constantly since 2008 – a full, 60h course on software testing for CS undergrad students (30h lectures + 30h labs). This course is obligatory for Software Engineering track and elective for other CS tracks. Every year ca. 120 3rd year students are enrolled on the course. The curriculum includes most topics covered by ISTQB Advanced Level syllabi. Moreover, several lectures are devoted to software quality assurance.

The similar course – however only as elective – is also offered to students at Warsaw University of Technology. Typically, about 15 students each year are enrolled on the course.

The common lack of knowledge on software testing is noticeable when studying the contents of the job positions descriptions. Many employers do not distinguish between software quality assurance and software testing, which can be classified as a software quality control activity. Software testing is therefore a proper subset of QA, but companies looking for testers frequently call these positions like "Quality assurance engineer", "QA engineer" etc. having a tester role in mind.

But even more striking for us are the requirements for tester positions. We analyzed several job offers for tester positions published on *pracuj.pl* – the most popular job seeking portal in Poland. Most of the job descriptions focus on the skills in test execution automation (that is, some knowledge on tools and programming languages) or on the knowledge of different SDLCs, Agile being the most popular. However, almost none of the job descriptions require:

- analytical skills,
- ability to read technical documents as test basis,
- good communication skills,
- inquisitiveness, exactitude, preciseness,
- knowledge on test design techniques,
- knowledge on different test approaches, like risk-based approach,
- skills on effective incident reporting.

Many companies require from the candidate the ability to manually or automatically executing test scenarios/test scripts. But even if scripts are to be designed by the candidate herself, there is no requirement about the ability to effectively design the tests that will be executed by these scripts.

We think that it is more than obvious that there is a great need to promote professional knowledge on software testing and software quality assurance. In the next sections we present the testing education in Polish universities, a general characterization of students and the analysis of curricula of several postgraduate studies in software testing.

2. Testing Education in Polish Universities

Postgraduate studies are the most specific form of the formal software testing education in Poland. This idea is specific to Poland and – according to

our knowledge – rather uncommon anywhere else¹. Curricula of regular (undergraduate and graduate) IT studies at Polish universities, technical universities and colleges typically contain a few hours devoted to software testing, usually as part of a Software Engineering course. But the postgraduate studies are becoming noticeably more and more popular. At the moment there are about 150-170 students taking such postgraduate courses at several colleges located in the biggest Polish cities (Warsaw, Gdansk, Cracow, and Wroclaw).

What is very interesting, women are the majority of the students, which is rather unexpected, because typically most of the students in technology area are men.

Table 1. Basic characteristics of the postgraduate studies in software testing in Poland

University	Total number of graduated /women	Number of students/ woman	Year of establishing
Copernicus, Wroclaw	91/45	32/25	2010/2011
Gdansk School of Banking	100/56	62/27	2012/2013
Vistula, Warsaw	25/10	25/7	2014/2015
Jagiellonian University, Cracow	39/20	39/20	2015/2016

3. Students: Their Profile, Experience and Expectations

Students interested in this kind of education can be divided into two groups. The first one are people familiar with the overall high quality of the IT jobs (salaries, benefits, work environment) that we sometimes figuratively call "programmers' wives". It does not necessarily mean that most of them are female, but they typically fulfil the following conditions: they believe that software testing is a good job with rather high salary and with relatively low technical entry level, among their close relatives or friends there is somebody working in the IT industry, they have some – typically very basic – knowledge on programming and advanced computer usage.

¹ According to the research program ISTQB Academia survey (internal materials)

A good example of the student expectation is a quote from the enrolment questionnaire, being an answer to the question about the motivation to undertake such studies at Jagiellonian University:

"I want to change my job and start to make progress; my expectation is to gain knowledge that would be enough to start as Junior Quality Tester."

The second group consists of people who have just started their careers as Junior Testers and found out that they need some systematized knowledge. From the same questionnaire:

"I want to learn how to effectively prepare tests, how to test software manually and with the usage of tools."

Candidates of the first edition of studies at Jagiellonian University filled out a questionnaire about their experience in software testing and IT in general. This data was used to split the students into two groups: less advanced and more advanced, but the results of this survey also shed some light on who is interested in this kind of studies and what background these people have. The results of the survey are presented in Tab. 2. Students were to perform a self-assessment in the following areas:

- knowledge on Windows systems,
- knowledge on Linux systems,
- knowledge on computer networks,
- knowledge on databases,
- knowledge on script programming,
- knowledge on C++,
- knowledge on Java,
- knowledge on C#.

Each skill was assessed with the 0-5 scale, where 0 means "absolutely no knowledge" and 5 means "I have an advanced knowledge in this area". The students might add other possessed skills. We also asked them about their expectations related to the studies. We received 56 answers.

Students also mentioned other skills, such as: HTML/HTML5/XML/CSS (7), Python (5), Selenium (2), AutoCAD/CAD systems (2), MS Office (2), Javascript (1), php (1), Matlab (1), MathCAD (1), jQuery (1), GIT

(1), Robot Framework (1), JMeter (1), JUnit (1), TestNG (1), SVN (1), DOS (1).

Table 2. Survey among Jagiellonian postgraduate studies candidates

Skill	Number of responses for a given grade						Mean
	0	1	2	3	4	5	
Windows systems	5	3	10	10	22	6	3.1
Linux systems	33	8	11	1	3	0	0.8
Networks	18	19	7	7	5	0	1.3
Databases	22	12	12	9	1	0	1.2
Script programming	32	12	7	4	1	0	0.8
Programming language (C++)	37	11	5	3	0	0	0.5
JAVA	34	8	9	4	1	0	0.8
C#	44	7	4	1	0	0	0.3

22 out of 56 persons indicated an IT company as their place of work. As for the expectations, two main groups are clearly visible: 24 said that they want to change their career path (so probably they have little or no initial knowledge on testing), 25 responded that they want to develop their careers in the field of software testing. 2 persons wanted to undertake the studies to prepare for the ISTQB Foundation Level exam.

23 responders indicated that during the studies they want to become familiar with widely understood automation (test automation tools like Selenium, script languages, programming languages), 6 wanted to gain more experience in test management, 5 wanted to learn about the test design techniques.

4. Curricula of Postgraduate Studies in Software Testing

The curriculum of software testing studies must take into account the two above-mentioned groups of students and their expectations. There are some differences between locations and universities, but basically the programs are more or less consistent with the ISTQB syllabi (CTFL and – partially – Agile). There are also some basic and levelling courses such as:

- programming (rather very basic), focusing mostly on writing simple scripts to solve problems automatically and on writing code for popular test automation frameworks,
- database systems, to allow for effective testing of software that uses them,
- Internet technologies that are used for websites and webservices,
- some introduction to operating systems, with special focus on Unix/Linux because they are widely used in the industry.

Curricula of Cracow, Gdansk, Warsaw and Wroclaw programs are presented in Table 3.

During the lectures related directly to software testing the main ideas from ISTQB CTFI syllabus are presented. Lecturers at postgraduate studies are mostly professionals with a rich IT industry experience, hence the emphasis on a practical and real-life approach; also many references to real-life projects and situations are given:

- although models are good to define, not all organizations follow them strictly,
- extensive planning is a good idea, but in many projects tester needs to manage her/his project with a limited plan; some hints are presented,
- because of popularity of Agile, the ideas related to the tester's role and the way of cooperation in the Agile environment are presented.

Because the test automation is on the top nowadays, much attention is put on the testing tools. As it can be observed in Table 3, about 20-25% of the curricula is devoted to test automation. At the moment typically the following tools are discussed:

- Selenium: IDE, Webdriver and Grid,
- some tools from the xUnit family,
- JMeter as a tool for performance testing,
- SoapUI for testing API & webservices,
- Robotium as a tool for mobile testing.

Students have an opportunity to use these tools during the workshops in laboratories. They also test previously prepared programs with some defects injected.

Table 3. Programs of the postgraduate studies in software testing

Subject	Number of hours			
	Jagiellonian University (Cracow)	Gdansk School of Banking	Vistula (Warsaw)	Copernicus (Wroclaw)
Software testing foundations, test design techniques	40	26	40	35
Test methodology				25
Test automation	40	50	50	50
Test management, test documentation	30		30	25
Networks	20	82	20	55
Databases	20		20	
Programming	30			
Operating systems	20		20	
Preparation to the ISTQB Foundation Level exam			20	
Soft skills in testing		14		10
Project seminar, additional trainings		28		
Total	200	200	200	200

Stowarzyszenie Jakości Systemów Informatycznych (Society for Quality of Information Systems, SJSI) supports these studies. Most of the lecturers are the members of SJSI, they are ISTQB certified testers and they try to share their knowledge with students studying in different locations. At the moment a new ISTQB portfolio is presented to the students to explain possible paths of their careers in software testing.

Students have an opportunity to take the ISTQB CTFL exams on special prices, much less than the market prices. Authors of the best graduate projects have this opportunity for free; they also have a possibility to present their papers on Testwarez – the biggest testing conference in Poland (about 300

participants each year) as guests of SJSI. All these opportunities aim to promote software testing studies in Poland.

5. Conclusions

In this paper we described the current state of the software testing post-graduate studies in Poland. We presented and analyzed the curricula and the candidates' profiles, experience and expectations. Some students attending software testing studies have an IT background, but most of them come from completely different fields. At present, salaries for software developers are usually higher than for testers or QA engineers. This is because quality assurance and quality control, in the employers' view, are still underrated. However, recently this trend has started to change. We believe that in a few years the quality assurance and quality control in Poland will become a mature and respected field of software development. Therefore, the salary disparity will start to vanish as one can observe in west European countries². As there will be a need for more experts in the field, postgraduate studies on software testing run at mathematics and computer science faculties are a big necessity. Such initiatives will have a direct influence on the quality of the software developed in Poland.

References

- [1] 2016 State of Testing Report, PractiTest & Tea Time with Testers, 2016, <http://qablog.practitest.com/wp-content/uploads/2016/04/StateofTesting2016.pdf>

² Private communication with senior testers from ISTQB German & French Testing Boards.

Chapter 12

Data Flow Analysis for Code Change Propagation in Java Programs

1. Introduction

The paper considers building a tool, based on well investigated the control and the data flow problems, which would take two versions of a program code and identifies operations which may produce different result. It is devoted to creating a model which allows comparing operations of two programs by the result they produce.

The tool requirement comes from the need of a maintenance of production systems which involves modification of its source code. Based on the scope of the change adequate testing is applied to maintain the designed software quality. Testing costs should balance between full, regression testing, and underestimated, selective testing. Any overestimation of testing increases the costs of software development. Performing a full regression tests on every program code change is often unacceptably expensive. On the other hand, underestimation of testing scope is even worse, introduces the risk of inconsistency of logic and data flow, degradation of safety, as well as performance, security breaches, etc. what translates to a number of bugs.

A proper code change scope estimation is essential from the quality and the cost point of view of a software project management. It determines business decisions: increasing the budget, rejecting the code change, selecting different solution, discovering workarounds, etc.

It is not a rare situation that a mission critical software in production for years requires relatively small change, but which is not applied because of a risk of introducing a bug which would involve serious consequences.

The planned budget for the change may directly impact the coding phase decisions: how many classes are going to be modified, how generic the implementation is going to be, which tools and libraries shall be used, etc.

When a change is implemented the situation is not much better, the scope of the change is still only estimation – a human error prone factor.

Issue tracking systems usually introduce severity, a characteristic to estimate the scope of change. However, grades like *tweak*, *minor*, *major*, *critical* are often confused by business users with *priority* and thus only occasionally used to report a change request and even more rarely used to manage or communicate the scope to the end user.

There is a need to formalize specification of how the code change can propagate through an application body. When the scope of change could be automatically generated the benefits would concern a coding process (greater awareness of the change consequences), a testing process (precise tests selection), a customer verification (limited end user verification effort to reduce risks of discrepancies), and so on.

The general idea of the paper is to create a tool which takes as an input two versions of a program code and outputs places of a program code which may behave differently between versions. The result of the tool should be comparable to an analysis performed by a programmer not knowing a specific domain. The paper demonstrates the capability of creating a tool but does not conclude practical results.

Section 2 extends the model proposed in [3]. Proposed model is oriented on comparison of operation results: a data flow technique is eliminated (operational stack, local variable, object's field) but operations through data flows are maintained.

The changed program code generates different responses which are propagated to other (not changed) parts of the program and they respond differently causing the change to be propagated. Changes can be populated explicitly – direct influence on a result, or implicitly – when a change approaches conditional statements with predicate that can influence control flow selection and, because of it, change the produced result. In particular, a notation is proposed in Section 3. Defining stack transformations and operations for manipulating local variables and objects' fields in terms of an abstract set of data flow relations is described in Section 4 to eliminate data propagation technique

selection impact (e.g. passing the value through the local variable, object's field or through the stack does not impact an expression result).

Section 5 shows how to deal with programming constructs like conditions, loops, exceptions, it is supported by a case study presented in section 6. The paper is summarized with a short review of the related work in section 7.

2. Code Change Propagation

This section describes how a tool could compare two program codes to identify operations which may produce different result.

The code change of a program implies some of its operations to produce different responses during execution. Such changed response flows to other, not-changed operations of a program and cause them to respond differently. The changed fragment of a program code propagates the change through the program body along its data flow paths. Deterministic operations may produce different results when they differs (e.g. operation IADD is changed into operation IMUL) and/or when data flowing to them differs.

Change Propagation Model. A propagation of a change is represented by a data flow model. A program body is converted into a data flow relation set. Two programs operations may be compared by appearance of the same elements in both sets. When all data which flow into certain operation are the same for two programs then it can be deducted that such operation responds the same result. Otherwise, a change is introduced, and its flow along data flow graph is observed (an operation which receives changed data as an input produces different result on an output).

Data definition and data usage operations are paired as data flow relations. Reading stack, using local variable, getting object's field value are connected to the corresponding push stack, assign local variable and setting object's field value operations. In the result, possible program code changes: moving part of a code to a method, or the opposite inlining the method body where it is used, extracting expression to a variable and field (or the opposite) may be automatically eliminated, as they do not introduce a change.

Precision of qualifying a change as substantial determines false-positive change propagation results. The paper is limited to comparing machine (byte-

code) operations but the model can accept more sophisticated comparison methods like resolving expressions or loops.

3. Notation Proposal

Data flow is preceded by control flow analysis. The control flow is defined as predecessor operation and successor operation relation set.

Each successor of a conditional operation (like if, switch, etc.) has a statically known value of the selection assigned to it. For example, it is known, for conditional operation IF_ICMPEQ L1 (comparison of two integer values) that if the condition is *true* (two integer values are the same) then a jump to the operation labelled by *L1* is performed. When the result is *false*, control flow is passed to the operation next to it. Let's introduce several useful notations.

Control Flow Relations. Control flow is defined as set F_c of control flow relations, represented by symbol ' \rightarrow ': relation $f_1 \rightarrow f_2$ denotes processing of operation f_1 passes the control flow to operation f_2 (operation f_1 is a predecessor of operation f_2). Sequence $f_1 \rightarrow f_2 \rightarrow f_3$ implies $f_1 \rightarrow f_2$; $f_2 \rightarrow f_3$ and is called a *path*.

Data Flow Relations. Data flow is defined as set F_d of data flow relations, represented by symbol ' \rightsquigarrow ': relation $(f_1) \rightsquigarrow f_a$ denotes that result of f_1 operation flows into f_a , where f_a is one argument operation (e.g. IFNULL, IFNE, INEG, I2B); relation $(f_1; f_2) \rightsquigarrow f_b$ denotes that results data of operations f_1 and f_2 flow into operation f_b , where f_b is two argument operation (e.g. IMUL, IADD, AALOAD, IF_ICMPEQ).

Conditional Data Flow. Relations Conditional data flow is modelled by suffixing a control flow relation with conditional symbol ' \triangleleft ' followed by condition selection set. Condition selection is a conditional operation (e.g. IFNE, IF_ICMPGE, TABLESWITCH) and statically known value (e.g. *true*, *false*) assigned to the control flow path selection.

Data flow relation $(f_1; f_2) \rightsquigarrow f_b \triangleleft \{f_c : v\}$ denotes that results of operations f_1 and f_2 flow into operation f_b only if the result of operation f_c is v . (Data Flow Relation is a particular case of Conditional Data Flow Relation where the condition selection set is empty).

4. Data Flow Considerations

A method for transforming the program data manipulation operations into the data flow relation set is proposed below.

4.1. Defining operations

Local variable value definition (value assignment) is handled in the bytecode by operations of the form xSTORE l (depending on data type there may be: ISTORE for an integer, LSTORE for a long, ASTORE for a pointer data type) for l-th variable. We say that a defining operation survives when there is no other defining operation on the control flow path. Otherwise, we say that it is *killed* (or does not *survive*).

An object's field value is defined by operation PUTFIELD <field name>. An array's element value is assigned by operations of form xASTORE (depending on data type: IASTORE for an integer, LASTORE for a long, AASTORE for a pointer data type, etc.).

Function P specified by Eq. (1) returns subset of defining operations set F which survives at operation f_a along their control flow path. In other words, P returns such operations of F from which there is a *path* to f_a and none of operations on the *path* is an element of F .

$$P(f_a; F; F_c) = \{f_x, f_x \in F \wedge \{f_x \rightarrow f_1; \dots, f_n \rightarrow f_a\} \subset F_c \wedge \{f_1; \dots, f_n\} \cap F = \emptyset\} \quad (1)$$

Let's consider control flow relation set $F_c = \{f_1 \rightarrow f_3; f_2 \rightarrow f_3; f_3 \rightarrow f_4\}$. For defining operations set $F_1 = \{f_1; f_2; f_3\}$ operation f_3 survives at operation f_4 : $P(f_4; F_1; F_c) = \{f_3\}$ (operations f_1 and f_2 are killed by operation f_3). But if we consider defining operations set $F_2 = \{f_1; f_2\}$ then both of defining operations survive: $P(f_4; F_2; F_c) = \{f_1; f_2\}$

4.2. Stack manipulation operations

The operational stack analysis relies on the fact that it is statically known for structured programs [3]. When a Java source code statement is finished the operational stack is empty. It is deduced that the operational stack

manipulation operations are not forked (since the first element is pushed till the last stack element is popped). This rule is not confirmed by a non-declarative exception throwing (see section 5.4). However, as the stack content is not accessible for an exception handler, this scenario is not required to be handled by the analyzer.

Analysis of the stack operations is performed in a similar way to how they are processed during execution on a machine. When operation type of push is analyzed then a result of it is pushed on the analyzer stack. For operations of statically known result like `ICONST_0`, `ICONST_1`, `BIPUSH 10` the values pushed on the stack are 0, 1, 10 accordingly. When operation type of `pop` is analyzed a value/s is/are popped from the analyzer stack and a data flow relation (between popped value/s and analyzed operation) is created. For arithmetical operations like `IADD`, `INEG` attribute values are popped from the analyzer stack, a data flow relation is created and the operations themselves are pushed on the stack.

Figure 1 demonstrates the analyzer stack for bytecode operations: `BIPUSH 101`, `ICONST_42`, `IMUL3`, `ISTORE 24` (corresponding to `int x = 10 * 2`; in Java). Analysis starts with empty stack – S_0 . Operation `BIPUSH 101` pushes value 10 on the stack – S_1 and `ICONST_42` pushes constant values 4 on the stack – S_2 . Operation `IMUL3` pops two values from the analyzer stack, the data flow relation $(10_1;4_2)$ `IMUL3` is created and the operation itself is pushed on the analyzer stack – S_3 . Operation `ISTORE 24` pops the value from the stack – S_4 and creates data flow relation element $(IMUL_3)$ `ISTORE 24`.

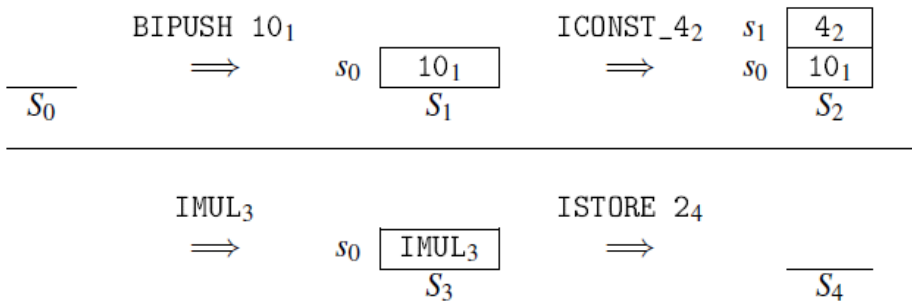


Figure 1. The analyzer stack operations example

4.3. Local variables

Local variable data flow analysis is that the analyzer resolves a local variable definition operation when a local variable usage operation is analyzed.

Let's introduce function $ldef(l; F_c)$ which returns all define operations of l -indexed local variable, such as: ISTORE 1, ASTORE 1, DSTORE 1, FSTORE 1, LSTORE 1, etc.

Function $lval$ specified by Eq. (2) defines a set of all possible l -th variable values at operation f_a . Possible values of the l -th variable are operations which flow into survived at operation f_a defining operations of that variable.

$$lval(f_a; l; F_c) = \{ v; (v) \rightsquigarrow f \wedge f \in P(f; ldef(l; F_c); F_c) \} \quad (2)$$

Let's consider an example of assigning 0-th variable the result of sum of integers and its usage further down the control flow: $F_c = \{IADD_n \rightarrow ISTORE\ 0_{n+1}; \dots; oper_{n+k} \rightarrow ILOAD\ 0_{n+k+1}\}$ (operations between $n+2$ and $n+k$ are not killing the 0-th variable). A value of the 0-th variable used in the $ILOAD\ 0_{n+k+1}$ operation is determined as single operation $IADD_n$: $lval(ILOAD\ 0_{n+k+1}; \{ISTORE\ 0_{n+1}\}; F_c) = \{IADD_n\}$.

Java compiler guarantees that a local variable must be explicitly given a value before it is used, by either initialization or assignment, in a way that can be verified using the rules for definite assignment [8]. It means that for every local variable usage operation (type of xLOAD 1) the corresponding variable value definition operation (type of xSTORE 1) exists and is identified by $lval$ function.

4.4. Object's identification

The analyzer identifies objects by operation NEW k (k is a class name). A reference to an object (a local variable or object's field) may point to object creation operation NEW k or *null* operation: ACONST_NULL. Inline method substitution causes that the same method called from different places is uniquely identified (NEW k operation is identified by the call stack).

4.5. Object's fields

An object's field value (a result of operation GETFIELD field_name) is determined by a corresponding defining field value operation PUTFIELD field_name. Let's introduce function $fdef$ ($field_name; F_c$) which returns all define operations of field field_name (all PUTFIELD field_name operations).

Function $fval$ specified by Eq. (3) defines a set of all possible field_name values of object's reference ref at operation f_a . It consists of values flowing into field defining operations.

$$fval(f_a; ref; field; F_c) = \{v; (ref; v) \rightsquigarrow f \wedge f \in P(f_a; fdef(field; F_c); F_c)\} \quad (3)$$

For operations from Listing 1 (corresponding to the source code: A a = new A(); a.field = null; if(a.field == a)) the following data flow relations are created: relation (NEW A₁) \rightsquigarrow ASTORE 1₄ defines value assignment of the 1st variable, relation (NEW A₁; ACONST_NULL₆) \rightsquigarrow PUTFIELD A.field₇ assigns *null* to field A.field of object identified by operation NEW A₁), and relation (ACONST_NULL₆; NEW A₁) \rightsquigarrow IF_ACOMPNE₁₁ which shows that values *null* and NEW are flowing into the comparison operation.

1 NEW A	5 ALOAD 1	10 ALOAD 1
2 DUP	6 ACONST_NULL	11 IF_ACOMPNE ...
3 INVOKESPECIAL A.<init>	7 PUTFIELD A.field	
4 ASTORE 1	8 ALOAD 1	
	9 GETFIELD A.field	

Listing 1. Java bytecode object definition example

The analyzer should be aware of missing Java object's fields definition requirement, since object's field value does not need to be explicitly defined [8]. In consequence, defining operation may not be found.

5. Implementation Considerations

Below we provide examples demonstrating how the proposed notation can be used to describe such programming constructs as conditional statements, loops, exceptions, method invocations; examples are based on the case study included further in the paper.

5.1. Conditional statements

Conditional operations of type IF like IF_ICMPEQ (comparison of integer values equality), IF_DCMPGT (comparison of double values, checks if the first value is bigger than the second), or type SWITCH like TABLESWITCH, LOOKUPSWITCH are represented as forks in a control flow graph.

Data flow analysis of conditional operations is performed in the same way like other operations. However, their possible results are attached to data flow relations as conditions.

1 ILOAD 0	4 ISTORE 1	8 ILOAD 1
2 IFEQ L6	5 GOTO L8	9 ISTORE 2
3 ICONST_1	6 ICONST_2	
	7 ISTORE 1	

Listing 2. Bytecode IF statement example

Operations from Listing 2 are reflecting the source code: `if (l0) l1 = 1; else l1 = 2; l2 = l1;` (operations are referenced in text by index: e.g. operation 1 is typed as ILOAD 0₁ or f_1). The control flow relations set is $F_c = \{f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \rightarrow f_8, f_2 \rightarrow f_6 \rightarrow f_7 \rightarrow f_8\}$, the 1st variable definitions at operation ILOAD 1₈ are $ldef(f_8;1;F_c) = \{ISTORE 1_4;ISTORE 1_7\}$. As both of them survive the possible values for the 1st variable are $lval(f_8;1;F_c) = \{1_3 \triangleleft \{IFEQ_2 : false\}; 2_6 \triangleleft \{IFEQ_2 : true\}\}$. Unconditional operation ISTORE 2₉ inherits conditions from the data flowing to it, and the final data flow relations are $\{(1_3) \rightsquigarrow ISTORE 2_9 \triangleleft \{IFEQ_2 : false\}; (2_6) \rightsquigarrow ISTORE 2_9 \triangleleft \{IFEQ_2 : true\}\}$.

5.2. Loops

The sequence of processing operations by the analyzer does not affect the result of the analysis. However, it is reasonable to do data flow analysis according to control flow graph. To determine possible values of data (a stack, a local variable, a field, an array's element) at a certain operation it is required to have calculated data flow to that operation back in the control flow graph. Instead of considering every possible control flow path, the analyzer would make usage of the knowledge that the program is structured and analyze structures one by one. For example, if-else statement is considered as analyzed

when all its control flow paths (for condition operation result true and false) are analyzed.

Loop analysis implies that the body of loop operations are visited many times by the analyzer. In the paper we use simple criteria of considering a loop as analyzed: when the next iteration of a loop body does not identify any new data flow relations. During the first analyzing iteration of the common loop: for (int i = 0; i < n; i++) initial value of local variable i is defined, data flow relations from that initial value to comparison i < n and to increment i++ are created. During the second analyzing iteration the next relations are created: from the increment operation to the comparison and to the increment operation. The third analyzing iteration does not identify new relations, analysis of the loop is finished.

5.3. Arrays

An array is supported in the same way as object's field with one difference, object's field name is statically known. When array's element is used (operations of type xALOAD) the analyzer pops two elements from the analyzer stack: reference to an array definition and array's element index and locates corresponding, defining operations (operations of type xASTORE). If an array definition (operation NEWARRAY) and element's index are statically known then the element definition is precisely selected. Otherwise, all array's elements are returned as a potential value.

5.4. Exception handling

Exceptions may be thrown declaratively by operation ATHROW. Java virtual machine can additionally create and throw a deterministic exception during execution of some bytecode operations, namely NullPointerException – during execution of operations INVOKEVIRTUAL, AALOAD, PUTFIELD when reference on which the operation is performed is null, ArrayIndexOutOfBoundsException – during execution operations of type xASTORE, xALOAD when index argument exceeds the size of an array and ClassCastException – during execution of operation CHECKCAST when type of the 1st argument reference cannot be cast to the 2nd argument type. The analyzer treats exception

vulnerable operations like conditional statements, namely if an exception condition is met then an exception is thrown. Otherwise, control flow is passed to the operation.

For throw exception operations the target catch block operations are identified as a control flow next operation (like for operations GOTO). If exception type is not statically known then the control flow may be passed to more catch blocks. For structured programs throwing exception may be considered as an exit point of multiple structures like: a loop, an if-else statement, a method, a try-catch block of a different exception type, etc.

5.5. Target of a virtual method

A data flow graph depends on data flow analysis and vice versa. The local control flow relation set (method scope) is built based on the method code. A virtual method call operation targets are selected based on data types of objects flown to invoke operation (on which the method call is invoked). Then operation INVOKEVIRTUAL is replaced with if-else statement of INVOKESPECIAL operations (based on class hierarchy analysis [2]).

5.6. Weak typed language

Java is a strong typed language, the presented approach does not make usage of this feature and may be adopted to weak typed languages like *JavaScript*, *Python* as well. The type of a local variable is not derived from the method signature but from the data definition (defining operation) flowing into that method.

6. Case Study

Data flow relation set building process is demonstrated on source code of Action class from Listing 3. The comments */*L*/* in Java source code points corresponding byte code operations of Listing 4. The result of the analysis, creation of data flow relations upon bytecode operations is presented in Table 1. Within this section operations are represented by indexes of bytecode

operations of Listing 4, e.g. 1, 2, 67, or by full names suffixed with this index: ICONST_0₁, STORE 2₂, ANEWARRAY Action₆₇ accordingly.

```

public abstract class Action<P>{
    static Action[] actions = new Action[] {
        new Action<Integer>() {
            int execute(Integer param) {
                return result +/*L57*/ param.intValue() +
                    /*L59*/ 1/*L58*/;
            }
        }, new Action<String>() {
            int execute(String param) {
                return param.length() +/*L64*/ 2/*L63*/;
            }
        };
        int result = 0/*L83*/;
        abstract int execute(P param);
        void process(P param) {
            result = execute(param);
        }
    }
    public static void main(String[] p) {
        int res;
        try {
            for (int i = 0/*L1*/; j; (j = i +/*L5*/ 1/*L4*/) </*L10*/
                p.length/*L9*/; i += 2/*L30*/) {
                switch (i %/*L13*/ 2/*L12*/)//*L14*/ {
                    case 0:
                        actions[0].process(new Integer(j));
                    case 1:
                        actions[1].process(p[j]);
                }
            }
            res = actions[0].result;
        } catch (NullPointerException e) {
            res = actions[1].result;
        } catch (IndexOutOfBoundsException e) {
            res = actions[2].result;
        }
        System.exit(res/*45*/);
    }
}

```

Listing 3. Action class source code

	26 ALOAD 0	53 ALOAD 0	80 PUTSTATIC Action.actions
Action.main	27 ILOAD 3	54 GETFIELD Action\$.result	81 RETURN
L1 L31 L38 NullPointerException	28 AALOAD	55 ALOAD 1	Action.<init>
1 ICONST_0	29 IVIRTUAL Action.process	56 IVIRTUAL Integer.intValue	82 ALOAD 0
2 ISTORE 2	30 IINC 2 2	57 IADD	83 ICONST_0
3 ILOAD 2	31 GOTO L3	58 ICONST_1	84 PUTFIELD Action.result
4 ICONST_1	32 GETSTATIC Action.actions	59 IADD	85 RETURN
5 IADD	33 CONST_0	60 IRETURN	Action\$.int\$.<init>
6 DUP	34 AALOAD	Action\$.execute	86 ALOAD 0
7 ISTORE 3	35 GETFIELD Action.result	61 ALOAD 1	87 ISPECIAL Action.<init>
8 ALOAD 0	36 ISTORE 1	62 IVIRTUAL String.length	88 RETURN
9 ARRAYLENGTH	37 GOTO L44	63 ICONST_2	Integer.<init>
10 IF_ICMPGE L32	38 ASTORE 2	64 IADD	89 ALOAD 0
11 ILOAD 2	39 GETSTATIC Action.actions	65 IRETURN	90 ILOAD 1
12 ICONST_2	40 ICONST_1	Action.<clinit>	91 PUTFIELD Integer.value
13 IREM	41 AALOAD	66 ICONST_2	92 RETURN
14 LOOKUPSWITCH	42 GETFIELD Action.result	67 ANEWARRAY Action	Integer.intValue
0:L15 1:L23	43 ISTORE 1	68 DUP	93 ALOAD 0
default: L30	44 ILOAD 1	69 ICONST_0	94 GETFIELD Integer.value
15 GETSTATIC Action.actions	45 ISTATIC System.exit	70 NEW Action\$1	95 IRETURN
16 ICONST_0	46 RETURN	71 DUP	String.length
17 AALOAD	Action.process	72 ISPECIAL Action\$.<init>	96 ALOAD 0
18 NEW Integer	47 ALOAD 0	73 AASTORE	97 GETFIELD String.value
19 DUP	48 ALOAD 0	74 DUP	98 ARRAYLENGTH
20 ILOAD 3	49 ALOAD 1	75 ICONST_1	99 IRETURN
21 ISPECIAL Integer.<init>	50 IVIRTUAL Action.execute	76 NEW Action\$2	
22 IVIRTUAL Action.process	51 PUTFIELD Action.result	77 DUP	
23 GETSTATIC Action.actions	52 RETURN	78 ISPECIAL Action\$.<init>	
24 ICONST_1	Action\$.int\$.execute	79 AASTORE	
25 AALOAD			

Listing 4. Action class bytecode

Bytecode in Listing 4 represents Java source code in Listing 3. Analysis of the Action.main method starts from static constructor Action.<clinit> which is called by Java virtual machine when the class is loaded, before the control flow is passed to method Action.main. The first two operations, ICONST_2₆₆

and ANEWARRAY Action₆₇ of method Action.<clinit> causes creation data flow relation (ICONST_2₆₆) \rightsquigarrow ANEWARRAY Action₆₇, represent instantiation of a new array of type Action with size 2. It can be denoted alternatively as (66) \rightsquigarrow 67 or 2₆₆ \rightsquigarrow 67. Method Action.<clinit> analysis yields the following set of relations: {66 \rightsquigarrow 67; (70;83) \rightsquigarrow 84; (67;69;70) \rightsquigarrow 73; (76;83) \rightsquigarrow 84; (67;75;76) \rightsquigarrow 79; (67) \rightsquigarrow 80}.

Table 1. Bytecode Action class analysis

Op.	Stack	Relations / Method Args.	Op.	Stack	Relations / Method Args.	Op.	Stack	Relations / Method Args.
1	{1}		21	{70, 18}		48	{76, 76}	
2	{}	(1) \rightsquigarrow 2	22	{}	(70,18)	49	{76, 76, 28}	
3	{1}		47	{70}		50	{76}	(76,28)
3 ₂	{30}		48	{70, 70}		61	{28}	
4	{1, 4}		49	{70, 70, 18}		62	{}	(28)
4 ₂	{30, 4}		50	{70}	(70,18)	96	{28}	
5	{5}	(1, 4) \rightsquigarrow 5	53	{70}		97	{97}	(28) \rightsquigarrow 97
5 ₂	{5}	(30, 4) \rightsquigarrow 5 \triangleleft {10: false}	54	{83}		98	{98}	(97) \rightsquigarrow 98
6	{5, 5}		54 ₂	{59}		62	{98}	
7	{5}	(5) \rightsquigarrow 7	55	{83, 18}		63	{98, 63}	
8	{5, p}		55 ₂	{59, 18}		64	{64}	(98, 63) \rightsquigarrow 64
9	{5, 9}	(p) \rightsquigarrow 9	56	{83}		50	{76, 64}	
10	{}	(5, 9) \rightsquigarrow 10	56 ₂	{59}		51	{}	(76, 64) \rightsquigarrow 51
		\triangleleft {10: false}	93	{18}				\triangleleft {10: false}
11	{1}		94	{5}		30	{}	(1) \rightsquigarrow 30 \triangleleft {10: false}
11 ₂	{30}		56	{83, 5}		30 ₂	{}	(30) \rightsquigarrow 30 \triangleleft {10: false}
12	{1, 12}		56 ₂	{89, 5}				\triangleleft {10: true}
12 ₂	{30, 12}		57	{57}	(83, 5) \rightsquigarrow 57	32	{67}	
13	{13}	(1, 12) \rightsquigarrow 13 \triangleleft {10: false}	57 ₂	{57}	(59, 5) \rightsquigarrow 57	33	{67, 33}	
13 ₂	{13}	(30, 12) \rightsquigarrow 13 \triangleleft {10: false}}	58	{57, 58}		34	{70}	
14	{}	(13) \rightsquigarrow 14 \triangleleft {10: false}	59	{59}	(57, 58) \rightsquigarrow 59	35	{59 83}	
		\triangleleft {10: false, 14: 0}	50	{70, 59}		36	{}	(59 83) \rightsquigarrow 36
15	{67}		51	{}	(70, 59) \rightsquigarrow 51			catch block
16	{67, 16}				\triangleleft {10: false, 14: 0}, \triangleleft {10: false, 14: 1}	38	{}	e \rightsquigarrow 38
17	{70}	(67, 16) \rightsquigarrow 17 \triangleleft {10: false, 14: 0}	23	{67}		39	{67}	
18	{70, 18}		24	{67, 24}		40	{67, 40}	
19	{70, 18, 18}		25	{76}		41	{76}	
20	{70, 18, 18, 5}		26	{76, p}		42	{64 83}	
21	{70, 18}	(18,5)	27	{76, p, 5}		43	{}	(64 83) \rightsquigarrow 43
89	{18}		28	{76, 28}	(p, 5) \rightsquigarrow 28	44	{59 83 64}	
90	{18, 5}		29	{}	(76, 28)	45		System.exit
91	{}	(18, 5) \rightsquigarrow 91 \triangleleft {10: false, 14: 0}	47	{76}				

Analysis of method Action.main is shown in Table 1. Operation 1 (ICONST_0₁) is pushed on the analyzer stack. Operation 2 (ISTORE 2₂) pops the element from the analyzer stack and data flow relation (1) \rightsquigarrow 2 is created. Operation 3 pushes operation 1 on the analyzer's stack. The alternate source of local variable (operation 30) is not known at this stage of the analysis (described below).

Below we describe in more detail the respective aspects of the analysis.

condition is not met: (5; $p \rightsquigarrow \text{ARRAYLENGTH}_9 \rightsquigarrow \text{IF_ICMPGE}_{10} : \text{false}$). Operations 97 and 98 may throw `NullPointerException` and operation 98 `ArrayIndexOutOfBoundsException`.

Loop handling. Operation 31 (`GOTO L331`) unconditional jump back in the code stands for the loop structure of operations 3-30. (Conditional operation 10 for value *true* becomes the loop exit condition). The analyzer follows that control flow relation as many times until no new data flow relation is created. During the second iteration analysis of loop 3-30 the value of local variable 2 comes additionally from operation 30 and new data flow relations are created (operations subscripted with 2 in Table 1). When operation 3 is visited the third time no new data flow relation is created, the analyzer leaves the loop analysis and continues with operation 32 (which is the target of loop exit condition 10: *true*).

Data flow handling. Operation `GETSTATIC Action.actions15` loads on the analyzer stack defining operation `PUTSTATIC Action.actions`: the only candidate is operation 80. According to `Action.<clinit>` data flow analysis result the data flowing to operation 80 is operation 67, what causes that operation 15 pushes operation 67 on the analyzer stack. Operation `AALOAD17` looks for defining operation `AASTORE` which takes as a first argument reference created by the operation 67 and indexed by `ICONST_016` and finds relation (67; `ICONST_069;70`) \rightsquigarrow `AASTORE73`. The result of `AALOAD17` analysis is operation 70 pushed on the analyzer stack.

Method invocation handling. Operation `INVOKEVIRTUAL Action.process22` pops two elements from the analyzer stack: operations 70 and 18. Operation 70 (`NEW Action$170`) is an object definition on which the `Action.process` method is invoked (and it is 0-th argument of this method call). Operation 18 is the 1st argument of this method call. As class `Action$1` is not overriding method `Action.process`, so the analysis is passed to method `process` of class `Action`. Afterwards, operation `INVOKEVIRTUAL Action.execute50` takes two elements from the analyzer stack: `NEW Action$170` and 18. As method `Action.execute` is overridden by `Action$1` class then the analysis is passed to it.

Change Propagation. Results of the analysis specified in Table 1 are presented graphically at Figure 2. Consider, for example, changing operation 83 (representing the source code of `int result = 0;`) to a different value (e.g. `int result = 1;`). It can be seen that the change is propagated to operations: 57, 59,

45. It implies that the change does not affect loop exit condition operation 10, does not impact on switch statement nor exception throwing. Let's consider this time changing operation 12 (representing number 2 in switch (i % 2)). This change has direct influence on operation 13 which impacts operation 14. Changed operation 14 implies the possible changes in control flow decisions, indirectly impacts operations: 28, 97, 98, 64, 57, 59, 45.

7. Related Work

The above exercise demonstrates the scope of analysis of a potential implementation. It may be seen from the above that a program code change analysis can be automated. Such an analyzer is currently under development by the author. It is intended to expand the analysis of data flow compared to the existing solutions characterized below.

In [1, 2] it is assumed that a target of a virtual call depends on a Class Hierarchy Analysis. A virtual call operation is replaced with if-else statement of static method bindings. This approximation is proper but inaccurate – note that resolving virtual call of `Object.toString()` in Java would involve all classes implementing this method! In the presented paper the class candidates are significantly limited only to classes which come from data flow analysis.

This paper approach is similar to [3] but extends the basic concept of data flow by specifying operations that the latter goes through. Additionally, it extends the accuracy of data sharing by distinguishing the statically known object instances.

Similarly to [9] the presented approach is context sensitive. However, methods are not preliminarily analyzed (implying preliminary assumption that method parameters do not *share*). The methods are inlined with the analyzed parameter possible values what introduces better precision at the expense of the computation complexity of the analysis.

It may be shown that for invocation `sh1.expand(sh1.next)`; of the *Sharing Example* from [9] the analyzed loop exit condition `while(cursor != null)` is never fulfilled, as the data flow into variable `cursor` never gets *null* value.

Sharing Analysis, Reachability Analysis, Cyclicity Analysis [7, 6, 4, 5] can be easily derived from the data flow analysis result as a presence of data flow relations.

References

- [1] D. Bacon and P. Sweeney. Fast static analysis of C++ virtual function calls, In: Proceeding OOPSLA '96 Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 324–341, ISBN: 0-89791-788-X, 1996.
- [2] J. Dean, D. Grove, and C. Chambers. Optimization of object-oriented programs using static class hierarchy analysis, In: Proceeding ECOOP '95 Proceedings of the 9th European Conference on Object-Oriented Programming, pp. 77–101, ISBN: 3-540-60160-0, 1995.
- [3] S. Genaim and F. Spoto. Information Flow Analysis for Java Bytecode, In: Proceeding VMCAI'05 Proceedings of the 6th international conference on Verification, Model Checking, and Abstract Interpretation, pp. 346–362, ISBN: 3-540-24297-X 978-3-540-24297-0, 2005.
- [4] S. Genaim and D. Zanardini, Reachability-based acyclicity analysis by Abstract Interpretation, *Theoretical Computer Science* 474, pp. 60–79, 2013, doi: 10.1016/j.tcs.2012.12.018.
- [5] D. Nikolić and F. Spoto. Definite Expression Aliasing Analysis for Java Bytecode, *Theoretical Aspects of Computing – ICTAC 2012* 7521, pp. 74–89, 2012.
- [6] D. Nikolić and F. Spoto. Reachability analysis of program variables, *ACM Transactions on Programming Languages and Systems (TOPLAS)* 35, Iss. 4, 2013, doi: 10.1145/2529990.
- [7] S. Secci and F. Spoto. Pair-sharing analysis of object-oriented programs, In: SAS'05 Proceedings of the 12th international conference on Static Analysis, pp. 320–335, ISBN: 3-540-28584-9 978-3-540-28584-7, 2005.
- [8] The Java Language Specification, Java SE 7 Edition, 2011, URL: <http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf>, accessed: February 20, 2016.
- [9] F. Spoto, F. Mesnard, and É. Payed. A Termination Analyser for Java Bytecode Based on Path-Length, *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 32, Iss. 3, 2010, doi: 10.1145/1709093.1709095.

III. Agile Transformations

Chapter 13

Trigger-based Model to Assess the Readiness of IT Organizations to Agile Transformation

1. Introduction

The agile manifesto, announced in 2001 was the pivotal event to promote the idea of agile approach in IT projects. Starting this year, the popularity of methodologies such as Scrum, XP or Kanban is continuously increasing [1] among the persons responsible for managing projects and also members of IT teams.

At the core of the agility idea is ability to adapt to customer expectations. Based on this statement, agility in IT projects should be understood as the ability of supplier to provide products (systems and services) consistent with the client's business needs.

This approach caused a shift from supplying ready-made, complex systems to products consistent with client's needs, changed the way of realization of IT projects over the last years. Created methodologies (frameworks) and techniques, which increase the supplier's ability to adapt to client's expectations begin to displace old, waterfall management models. Focus on the iterative and increment delivery dominated the earlier approach to the projects realization, based on baseline plan and providing product in accordance with requirements set out in the initial phase of projects [9].

Thanks to the popularity of agile methodologies and techniques, agile project management become so popular that either small or the world's biggest companies of the IT industry decided to change the management method to the agile. This popularity of agile approaches compels the IT organizations to change the way they operate – from the traditional, based on processes and

plans to agile – based on involvement the customer and supplier in the project. The process of changes caused a transition from the waterfall model of delivering to the iterative and incremental delivering methods called Agile Transformation (AT).

Nowadays the "agile transformation" becomes the key word, which describes the transition from waterfall method of software delivering to the new way – agile. However, whether the use by the supplier several agile techniques (e.g. daily meetings) may be sufficient to define supplier as agile organization? In this article authors attempt to determine the factors, which are important during the agile transformation processes. These factors were defined as a triggers or events giving rise to the decision to abandon classic, waterfall project management methods [6].

Due to the fact that agile transformation increasingly becomes more popular in IT industry, it seems it is reasonable to attempt to classify key concepts and to identify key factors, which contribute to the success of agile transformation.

This article consists of five main sections. The first one is the introduction to the agile transformation concept. Second point is the description of model of assessment of the IT organizations' concept. Third point contains the description of study method and obtained results. In point four authors described modified model, which was complemented by key elements. The last point is the summary of research.

2. Model to Assess the Readiness of IT Organization to Agile Transformation

Observations of agile transformations in IT organizations allowed to conclude that the level of readiness of organization for this change determines extent the probability of success of this process. IT organizations, which intend to change the project realization methods from classic, waterfall to agile should be properly prepared for this change.

Readiness of the organization to initiate transformation process can be described as a state, in which after evaluation, a chance to implement agile approaches with success is high. Starting agile transformation without prior

analysis and assessments of the organization makes these processes uncontrollable. In the initial phase of research, authors assumed that process of transformation must be considered from the perspective of Project – Supplier – Client [2]. During the described in this article research this approach has been changed to approach in which transformation process must be considered from the four perspectives. The next section describes these four perspectives.

2.1. Perspectives of agile transformation

Agile transformation process can be under consideration from the four, following perspectives:

- **Project perspective** – the scope of this perspective includes changes in project management methods and adaptation of agile methodologies and techniques.
- **Processes perspective** – the scope of this perspective includes changes in the functioning of the organization and internal procedures.
- **Organizational culture perspective** – the scope of this perspective includes changes in the mentality of people who manage of the organization and team members.
- **Technology perspective** – the scope of this perspective includes changes in technology use and the need to adapt to new working technologies.

Perspectives are the logical areas of functioning of organization, which change project management method from classic to agile. Between perspectives are some dependencies, i.e. project and processes perspective should be analysed before technology perspective. On the other hand, organizational culture perspective, due to its immeasurable character, should be subjected to transformation slowly, gradually and probably only after the full implementation of agile methodologies and techniques. Perspectives and relations between them are represented in the Figure 1.

Agile transformation perspectives allow for separation of the areas that should be assessed. At this stage of research, we focus on the detailed description of only one perspective – the processes. In the next subsection authors present a proposal for a model for assessing organizational readiness for AT

from processes perspectives. State of readiness and evaluation method is described in the next section of this article.

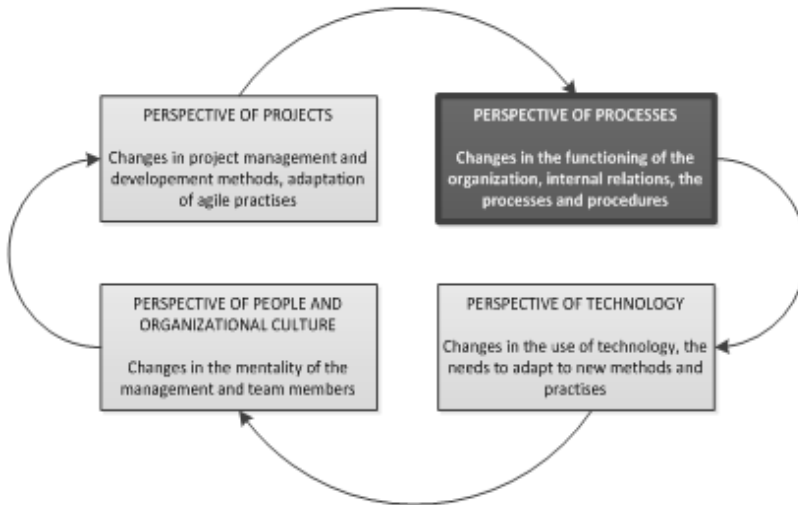


Figure 1. Agile transformation perspectives

2.2. Stages of agile transformation

In the processes of agile transformation, there are three following stages:

- **Before AT (initiation and planning stages)** this is the stage, where the organization is preparing for the transformation processes. Evaluation and activities to ensure that the point of readiness is achieved occurs in this phase. At this stage, decision is made and this is the reason why it is so important to determine whether the organization is at the point of readiness.
- **Agile transformation (execution stage)** – this is the stage where the agile transformation processes have been started. AT can be revolution for organization, which is characterized by the sudden abandonment of existing methods or it can be evolution for organization, where transition to agile takes place gradually.
- **After AT (improvement stage)** – this is the stage, where organization already achieves the end point of agile transformation process, which

means that classical methods have been abandoned but transformation process still goes one, in particular the evolution of processes and adaptation new agile processes.

Figure 2 presents stages of agile transformation.

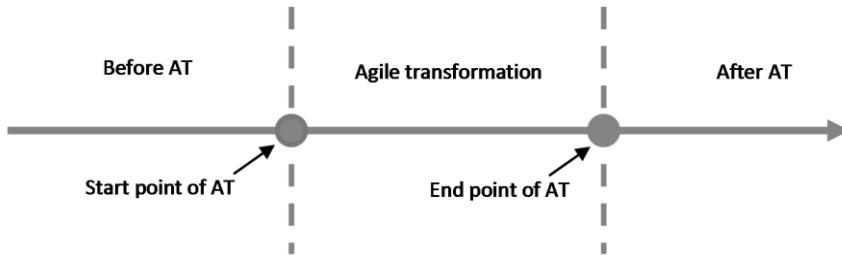


Figure 2. The stages of agile transformation

Evaluation process occurs in first phase and the key element is to identify if start point has been achieved. In the assessment process we can distinguish two states:

- **The point of readiness has not been achieved (Figure 3)** – this is the state in which decision has been taken prematurely and the risk of failure of agile transformation is high. Organizations, which not reach point of readiness, should analyse themselves and improve processes.

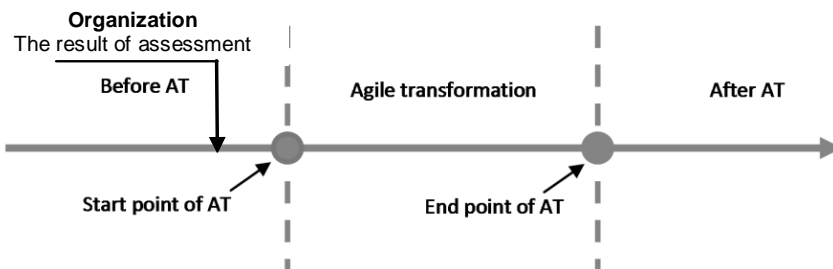


Figure 3.. The point of readiness has not been achieved

- **The point of readiness has been achieved (Figure 4)** – this is the state in which decision has been taken in proper time and the risk of failure of agile transformation is limited. The organization is in the start point of agile transformation.

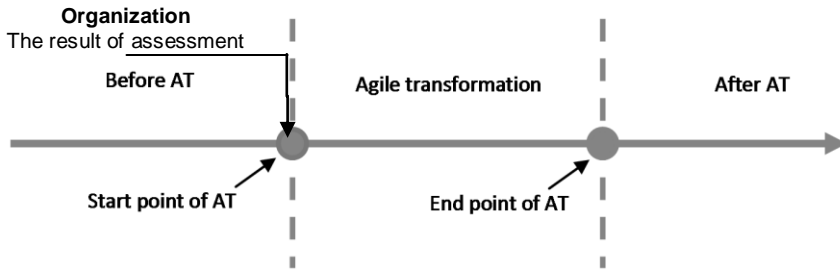


Figure 4. The point of readiness has been achieved

During the research, authors defined the two states of readiness, but in reality it happens those IT organizations, which have not reached the point of readiness, begin the transformation process.

2.3. Model to assess the readiness of IT organization to agile transformation

Figure 5 presents a proposal for model to assess the readiness of IT organization to agile transformation. Described model consist of three main layers:

- **Input** – input to the model is the measure instrument, which means the tool for evaluating IT organization. Measuring instrument consist of the questions about organization's characteristic and the occurrence and level of processes (levels definitions in accordance with CMMI [7]) in organization. Depending on the type of company (product-based or service-based company) questions are suitably selected.
- **Inference layer** – the key element of this layer is the inference technology. It uses processes described in two standards – Information Technology Infrastructure Library (ITIL) – 26 processes for service-based companies and Capability Maturity Model Integration (CMMI) – 22 processes [3] for product-based companies. In initial phase of research authors decided to use ready-made standards, however the list of processes will be extending during further research. In inference layer occurs selection process of relevant organization's characteristics. The next step is to assess the level of processes in organization

and to map processes to agile transformation processes (processes of target methodology) [1].

- **Output (results)** – at the output occurs an evaluation of organization's readiness to AT and as result the list of processes needed to improve or reorganization.

The layers and components of the model are illustrated in Figure 5.

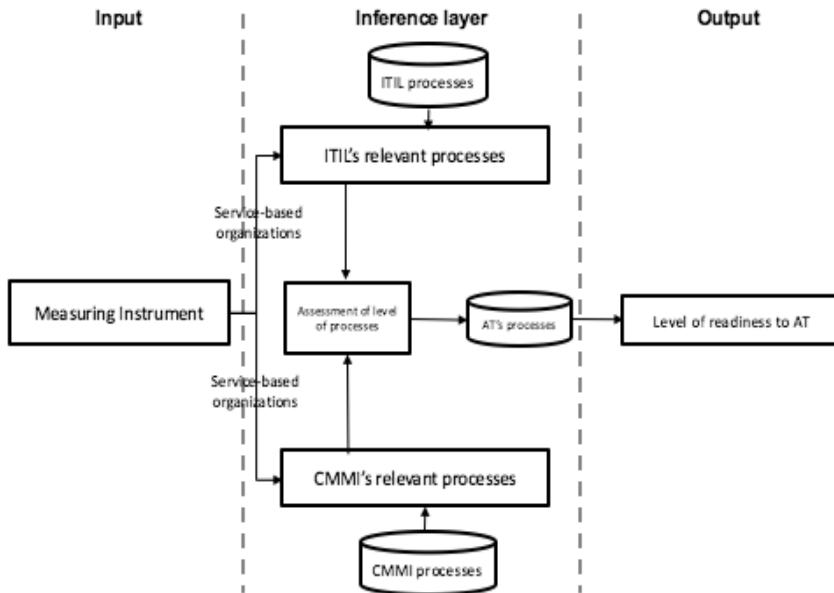


Figure 5. The layers and components of the model

3. Verification of the Concept Model

Presented in Figure 5 model contains introductory assumptions, which were verified during research of IT organizations. The method of research, obtained results and impact of results on model has been described in the next section of this paper. The structure of model presented in Figure 5 is based on the preliminary own observations and on the data from the ISBSG database. ISBSG – The International Software Benchmarking Standards Group Limited is database, which contains 1582 records [4], each record is the description of one project. Based on the analysis of 1582 cases and author's own

observations the preliminary concept model was developed (Figure 5). In further research authors prepare the questionnaire in order to verify the correctness of concept model. The questionnaire was directed to the members of the boards, directors, managers, team leaders and members of development teams. The study consisted of two parts: quantitative research, which was the online survey and qualitative research which were the standardized interviews with managers and team leaders. The last element of the research was to complement the concept model with new elements. Figure 6 present the flow of the research.

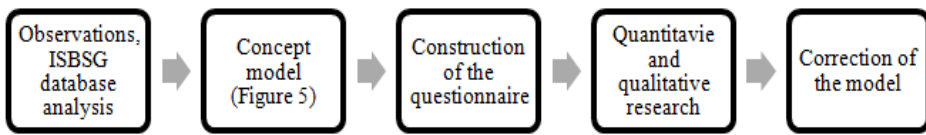


Figure 6. Flow of the research

The key element of the research was the qualitative and quantitative study, which allows for the complement the concept model with new elements. Detailed description of qualitative and quantitative research is presented in the next subsection of this publication.

3.1. Quantitative research

The scope of the quantitative research includes the preparation of preliminary questionnaire and directs it to the members of the boards, directors, managers, team leaders and members of development team. The survey was divided into seven main sections:

- **Organization** – this section includes questions about characteristics of organization i.e. structure, size, geographic spread, number and size of development team, work environment and also customer type.
- **Processes in the organization** – this section includes the questions about the level of descriptions of the processes in organization, whether there are processes definitions (is yes, at what level – general or specific), frequency and method of audits and also the level of processes standardization.

- **Type of organization** – the purpose of this section was to determine what is the type of organization, whether is product-based or service-based. Depending of the type of organization, in the next section were used respectively CMMI [7] or ITIL.
- **Processes** – in this section authors verified the level of implemented processes. The questions were about CMMI for product-based organizations [5] and about ITIL for service-based organizations. For each of process definition used a four point scale: processes are not defined, processes are defined but are not implemented, processes are defined and partially implemented, processes are defined and fully implemented.
- **Used methodology** – this section includes questions about the methodologies used in organization and about the state of agile transformation. Depending on the answer, respondent was directed to another section about agile transformation. In this section were distinguished six states of agile transformation: organization uses a classic methodology and does not plan to change it for agile, organization uses a classic methodology but plans to change it for agile, organization is during the agile transformation, the organization completed agile transformation but after a while returned to the state before transformation, the organization completed agile transformation but agile methodology is not implemented as intended (not in 100% according to the assumption of methodology), the organization completed agile transformation and agile methodology is implemented as intended.
- **Agile transformation** – this section contains questions about chosen agile methodology, the key factors in the transformation processes, about problems and about processes, which needed to be implemented or reorganized. This section includes also questions about the decision to change old methodologies for agile.
- **Demographics** – this section includes questions about respondent – age, job title and experience.

The survey shown that only 33% of IT organizations, before agile transformation, analysed the compatibility of processes with the chosen methodology, only 17% prepared a recovery plan in case of failure. As also shown 50% of organizations had to implement new processes for agile transformation,

33% defined already exist processes and only 17% did not have to organize and implement processes. The greatest problems during the agile transformation were the reluctance of team members (100% of respondents) and the lack of definition of processes (100%). These results showed how important is to assess the organization and accurate preparation to the agile transformation.

3.2. Qualitative research

Qualitative research in its scope included standardized interviews with project managers and team leaders. Questions concerned the key aspects in the evaluation of an organization's readiness to agile transformation. The result of interviews was to determine perspectives of agile transformation (Table 1) and key factors, which was important during the decision-make process.

Table 1 includes perspectives of agile transformation with defined key factors. The second key element in assessment of the organization is transformation triggers, which are described in the next section of article.

4. Transformation Triggers and Evolution of Concept Model

Factors that lead to the decision (e.g. by the members of the board) to change current, waterfall approach for agile approaches are called transformation triggers.

It is worth noting that the reasons for undertaking the decision to transformational change can vary substantially. Transformational change may indeed result from a desire to improve, but can also result from other causes, less related to the parameters of the project and more aspects of human.

Below are results of an attempt to classify the reasons for taking the decision to agile transformation. Such a classification is dictated observations on a dozen teams that have undergone a process of agile transformation and can be extended to other categories during the further research.

Depending on the observed reasons for the decision to agile transformation, four categories of triggers were isolated, presenting alongside examples

of events that lead to the decision to implement the changes. Categories of agile transformation triggers are summarized below.

Table 1. Perspectives of agile transformation

Processes perspective	<ul style="list-style-type: none"> - level of implemented processes - audits methods - compatibility between processes in organization with chosen methodology - ability to change existing processes - cost of changes of processes
Project perspective	<ul style="list-style-type: none"> - complexity and innovation of projects - client's characteristic - maturity level of customer - type of projects - typical duration of projects
Organizational culture perspective	<ul style="list-style-type: none"> - transformation triggers - size of organization - size of project teams - level of involvement of employees / management - geographical dispersion - level of knowledge of chosen methodology - competences of team members
Technology perspective	<ul style="list-style-type: none"> - required tools - cost of licences - cost and possibility of abandoning the existing tools

- **Effectiveness triggers** – this are the factors, which can be measured directly in the project. Project management methodologies distinguish [10] six of these aspects by including: schedule, budget, scope, quality, risk and benefits. In the case of these triggers we can talk about the need for agile transformation just to improve performance of efficiency in projects. Organizations decide to change their current way of working for agile so that to improve the performance of their projects (increase timely delivery, improve the quality of the delivered system, etc.). It should be noted that the parameters of efficiency can be early

warning signals – if the organization observes low efficiency, can in the early stages of start preparing for transformational change.

- **Forcing triggers** – this are the factors not resulting directly from the situation of supplier's organizations, but are dictated by other considerations. They can arise from i.e. adaptation to the customer's organization, which expects organizations to deliver products in frequent and incremental way. It can also mean a situation in which supplier during realization of the project interacts with other entities cooperating for the benefit of the customer and must adapt to a mode other entities.
- **Project triggers** – such internal factors of the project, which force organizations to use agile approaches and thus the transformation. If organization decides to implement i.e. research and development programs may find that the specificity of such projects will not allow the use of a waterfall approach. Hence, the use of agile approaches will somehow naturally, inherently inscribe in the specifics of the project. The premise of the merits may also be the level of information (entropy project) possessed at the time of starting the project as well as the state of maturity of customer. Each of these factors can influence decisions about the use of agile approaches. It is important that agile transformation can be scalable – involve only one team, program and the whole organization (depending on the substantive scope of the projects, orders, etc.).
- **Motivational triggers** – such outside non-formal factors that lead organization to decide on the agile transformation. In some situations, organizations decide to implement agile methodologies adapting to market trends and certain expectations of employees. Due to the fact that today there is a great worker's need for "Space" and freedom of action and the need for self-realization, self-organization in place of the rules imposed from above, agile transformation can meet these expectations. Motivational factors cause the need for change may also be due to the low morale of the teams working in the classical approaches. Motivational Triggers arise from the need to raise the synergy of teams, providing conditions for the conceptual, creative thinking and thus to teamwork.

It is important to keep in mind that agile transformation triggers are factors, which force the decision. The decisions to agile transformation are taken mostly on the level of company management. Of course, there exist cases of grassroots initiatives (as was the case in several companies where employees expect to change the way of work), but it is an example of specific triggers – motivational. The decision to change, however, takes, in most of cases, company's board. Making decision about transformation in the situation when the organization is not ready for it, may involve certain negative consequences and contribute to failure of transformational changes or at least to achieve a smaller benefit than expected as a result of the transformation. That is why it is important to possess tools to evaluate the organization and its readiness to agile transformation. As a result of research authors decided to add triggers to the model to assess the readiness of agile transformation. Transformation triggers prove to be an element with a high level of significance, for the assessment process. It is also important that there has been a change in assumptions. It has been shown that processes perspective is only one of the four, which should be reviewed. Model with the new elements is presented in Figure 7.

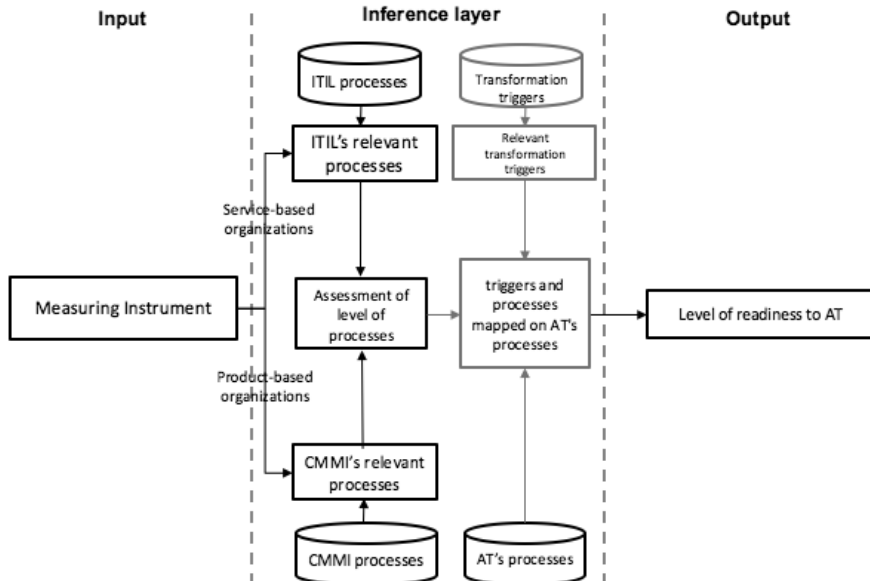


Figure 7. Based on triggers model to assess the readiness of IT organization to agile transformation

The result of the research is to complement the model for assessing readiness to agile transformation by the triggers. In the assessment process, in addition to assessing the level of processes and their level of adjustment to target methodology, must also be considered transformations triggers. During organization's assessment it is important to determine what factors were decisive in the decision-making process and this factors should be mapped to processes of agile transformation. This approach ensures inclusion of two key aspects – level of compatibility of processes and key factors that affect agile transformation.

5. Summary

This article presents concept of a model for assessing an organization's readiness for agile transformation. As a result of research demonstrated that the processes of transformation should be considered not as initially expected from perspective of Customer – Supplier – Project but from four perspectives: process, technology, people and organizational culture. Due to the complexity of each of the prospects the decision was made to embed of proposed model within the perspective of processes. In addition, four groups of transformation triggers were defined: effectiveness, forcing, project and motivational. During the study also demonstrated need to build such models, because only 33% of surveyed organizations made analyse their own processes for their compatibility with the chosen methodology, and only 17% had prepared a recovery plan in case of failure. As also shown 50% of organizations, which complete the agile transformation had to introduce a completely new processes, 33% had to reorganize processes already exist, and only 17% there was no need for them to reorganize. It should be noted that although this article presents a concept of model for assessing readiness to agile transformation, this does not mean that positive result of the evaluation is a guarantee of success. Agile transformation consists of four perspectives, which should be assessed. This article is part of a whole research on agile transformation and includes description of only one perspective. In the next stages of research on agile transformation authors wanted to build concept models for other perspectives and subjected it in to verification by means of quantitative and qualitative research. The result of the

research is to build a model to assess the readiness of agile transformation, taking into account all four perspectives.

References

- [1] C. Orlowski, T. Deregowski, M. Kurzawski, K. Ossowska, A. Ziolkowski. Wykorzystanie miar złożoności projektu do oceny stanu ewolucji organizacji informatycznej, *Innowacje w zarządzaniu i inżynierii produkcji Tom 2*, Oficyna Wydawnicza PTZP, Opole, 2016.
- [2] Z. Kowalczyk, C. Orlowski. *Advance Modeling of Management Processes in Information Technology*, Pomorskie Wydawnictwo Naukowo – Techniczne, 2012.
- [3] T. Deregowski, A. Ziolkowski, Hybrid approach in project management – mixing capability maturity model integration model with agile practices, *Social Science (Socialiniai Mokslai)*, Nr. 3 (85), 2014.
- [4] ISBSG Database, Release 12, February 2013.
- [5] C. Orlowski, T. Deregowski, M. Kurzawski, A. Ziolkowski. Model służący kształtowaniu procesu zarządzania projektem informatycznym dla podniesienia gotowości do zwinnej transformacji organizacji informatycznej, *Innowacje w zarządzaniu i inżynierii produkcji, Tom 2*, Oficyna Wydawnicza PTZP, Opole, 2016.
- [6] M. B. Chrissis, M. Konrad, S. Shrum. *CMMI for Development: guidelines for process integration and product improvement*, Third Edition, Pearson Education, p. 289, 2007.
- [7] CMMI Product Team, *CMMI for Acquisition*, Version 1.3, Carnegie Mellon University, p. 171, 2013.
- [8] D. Longstreet. *Fundamentals of Function Point Analysis*, Longstreet Consulting Inc., 2005.
- [9] The International Function Point Users Group, *Function Point Counting Practices Manual*, Release 4.1.1., 2000.
- [10] K. Koteswara Rao, G. S. V. P. Raju, T. V. Madhusudhana Rao. Effort Estimations Based on Lines of Code and Function Point in Software Project Management, *International Journal of Computer Science and Network Security*, Vol. 8 No.6, 2008.
- [11] Great Britain: Office of Government Commerce, *Managing Successful Projects with Prince2*, TSO, Norwich, United Kingdom, 2009.

Chapter 14

The Reference Model of Tools Adaptation in the Perspective of Technological Agile Transformation in IT Organizations

1. Introduction

The growing importance of project management in the computer industry organizations implies a number of changes in the functioning of these organizations. The classic approach in the management of these organizations, based on specialization of functions of individual departments persists more and more orientation on projects. It means, therefore, that the objectives of these organizations and their main business processes is not due to the efficient functioning of these departments, but to the effectiveness of various design programs and individual projects.

The evolution of the IT organization from the silo to the project is dictated by both the high dynamics of the macro-environment (changes in the law, the need to adapt to global trends, opportunities and expectations for tools) as well as the environment closer to these organizations (the intensity of competition in the markets, the need for faster introduction of new products, match up to customer expectations, increase of the importance of dedicated services – "special services").

The processes of evolution of the organization of information from the silo to the project (and service) enables these organizations to better adapt to the market changes, but also cause a number of internal problems resulting from this evolution [1]. Regular reports Standish Group [13, 14] showing that the percentage of completed projects with problems (delays, overruns, incomplete range) is still relatively high. It means that the evolution of the

organization to the project from silo-structure is still not a guarantee of success. It turns out that classically projects, based on the planning, specification of requirements and financial framework (called cascade) and assuming providing business value to the customer only after the completion of the project, do not meet customer expectations.

That problems connected to the classic approach to project management and the need of immediately delivery of business value (e.g. an efficient functionality of the system) have caused increase of the importance of "agile" approach in project management [2]. The foundation of this trend is to engage the customer to design work in such a way that the value of the business should be delivered as soon as possible, preferably in short cycles of developing based on incremental building system. Processes of change in the functioning of the PM organizations and industry of IT services are called agile transformation for at least 15 years. It means that organizations abandon cascade and start to use agile methodologies [9].

The phenomenon of agile transformation connected to a number of organizational changes in IT companies is a relatively new phenomenon which requires constant analysis and deep research. IT industry treats this phenomenon as a part of evolution, but the experience of many organizations shows that transformation processes can be uncontrolled and contribute to increase the level of project risk. Therefore, it is necessary to study the agile transformation processes to a better understanding of their impact on the functioning of the organization of the IT industry [10].

Current research conducted by the authors point to the need to analyze the agile transformation processes in several parallel areas called perspectives of agile transformation. Long-term observations of phenomena possible to separate such perspectives as: the perspective of projects, processes perspective, the perspective of tools and the perspective of organizational culture. Analysis of these key perspectives allows to better understand the processes of transformation, their importance for the functioning of the IT organization and manage of these processes in a way to minimize the risk arising from transformation changes and increase the effectiveness of projects in IT organizations.

The purpose of this paper is to demonstrate the possibility of use copyright model of technology selection in one of the identified agile

transformation's perspective. The selection of technologies reference model – developed to adapt the functionality to the needs of organizations implementing specific technologies – seems to be consistent with the assumptions of the management based on agile methods and tools aligned to the needs of the organization and project teams. This article presents the possibility of usage of that model in the perspective of the technological organization implementing the agile transformation.

2. Agile Transformation Perspectives

Implementation of agile transformation by the IT organization may face certain obstacles that can significantly prevent the implementation of agile processes. Publications which concern experience gained with the implementation of agile approach indicate mainly to the challenges of existing organizational structure, reluctance of people to make changes and processes restrictions. These issues are the primary constraint to the organizations which start the process of transformation or consider use of agile methodologies. Latest report [3] concerning the state of implementation of agile methodologies, published by the organization VersionOne, confirms observations documented in the literature. The report based on study [3] involves 3 880 organizations. 24% of organizations had more than 20 000 employees. The report mainly shows limitations of possibilities for introducing changes in the organization as the main barrier to the implementation of agile methodologies: because of cultural (55%), structure of the organization (42%). The following places were the lack of experience of the team (39%), lack of management support (38%) and the limitations resulting from the usage of the market regulatory compliance and standards (13%).

Processes of agile transformation entail a number of changes in the functioning of the IT organization and project teams in these organizations [4]. Necessity (in a relation to agile manifest) to provide contacts between people over process approach and tools and technology will shift the functioning of these organizations into new directions. Agile transformation means the necessity of another (new) method of verification of the progress of the projects, creation of conditions for cooperation with the customer and enable customer

to project teams and requires from project teams to being ready (maturity) to carry out work in a self-organizing, transparent and based on continuous improvement.

However, application of the principles of agile project management (agile projects), means that changes are not only in the organization but also in the technologies which are used by organization. So that, the readiness of the organization to agile transformation and forecasting the transformation process' course should be considered on the level of 4 perspectives. They are logical areas of the organization passing processes of agile transformation.

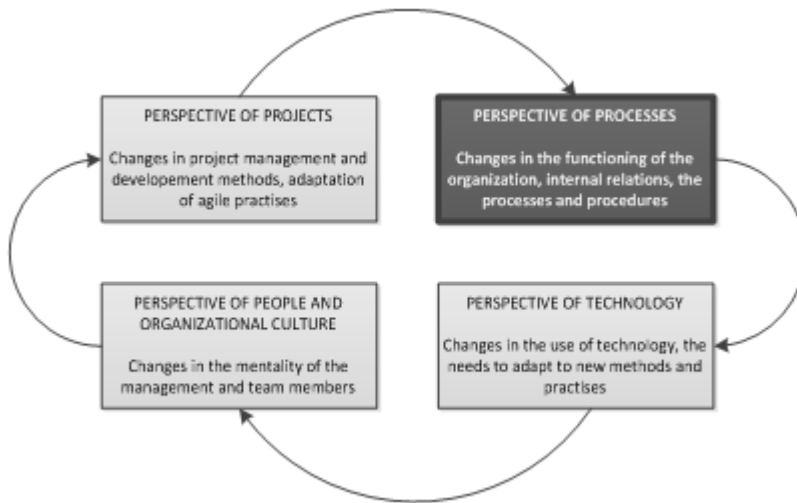


Figure 1. Perspectives of agile transformation

As indicated in the figure, between the perspectives it is possible to point at some time dependences. It means that the perspective of projects and processes should be analyzed before technologic perspective. On the other hand, the perspective of organizational culture, due to its immeasurable character, should be transformed slowly, gradually, and probably only after the full implementation of agile methodologies and technologies in support of agile work mode. However, in order to thoroughly study of the agile transformation, it can be needed to conduct independent studies about every perspective to define influence of each of the 4 areas on the success of agile transformation processes.

In this article authors focus on the perspective of technology, where organization which transforms itself should take into account the adaptation of appropriate tools, consistent with new principles of agile projects in the organization which implements the transformation processes [11].

2.1. Technologic perspective of agile transformation

One of the key actions in the technological perspective of agile transformation is to prepare the tool environment, which will support the chosen agile methodology or a set of good practices on the selected stage of the transformation process. The report clearly indicates the critical importance of technology for scaling agile methodologies to the needs of the organization which have a corporate nature. Consistency of process, tools, and practices (43 % of organizations [3]) and implementation of tools supporting the work of the teams are key factors for success in scaling agile approach to IT organizations.

Transformational changes in the technological perspective force generally change technology (in organization), which is used to support project management or the management of the developing process. In the case of a small single team, the choice of tools used is not important because of the low level of complexity of the project and direct communication in the team, which is promoted as a basis for effective work team. So that, organizations indicate on simple tools that are able to meet their need for the implementation of simple projects. There are a few which are mentioned most often: Microsoft Excel (60%), Atlassian JIRA (51%), Google Docs (18%) and Bugzilla (10%), what confirm the small scale and low complexity of projects which are implementing.

According to the studies [3, 14], organizations see a need to change these tools when they start the adaptation of methodologies tailored to the scale of their organization. It directly results from a discrepancy between tools listed as used with a list of recommended tools for professional use. The authors indicate here the lack of recommendations from the sites of popular processes (Scrum [5], XP [6], SAFe [12]) concerning the use of tools and functionality that they should have. Similar behaviour can be seen from the organization of developing them as Scrum.org, Scrum Alliance or IAS, which are not associated with any of the technologies.

Observed factors allow to the conclusion that the process of technology selection in organizations which are agile transforming play and will play an increasingly more important role [7]. Especially interesting seems to be matching the technology to the needs of the organization based on its expectations for the functionality of the tools or compliance of chosen methodology used by organization. The authors suggest the use of reference models as one of the possible solutions that can be adjusted to support the selection of the technology due to the needs and scale of the organization.

3. Selection of Technologies Reference Model for IT Organizations which Implement Process of Agile Transformation

In order to explore and deepen knowledge about technology's support for agile processes of transformation and selection tools for projects, authors conducted an analysis of the available literature and knowledge. The result of the study was to put forward the thesis that the processes of agile transformation should be supported by a properly chosen technology for the entire organization in the way to achieve specified at the beginning of the transformation of business goals.

For this purpose, the authors have created a concept of reference model according to the selection of tools for the execution of processes in projects and IT services. Model, according to the guidelines, should enable the selection of such tools, which fit to the developing processes implemented in IT organizations. Below there is discussed logical construction and usage of existing implementation of the model to the needs of technology selection. Such a model survey is crucial to demonstrate its role in the perspective of organizations implementing technological agile transformation processes.

3.1. The construction of the ALMRef reference model for IT organizations which implements process of agile transformation

The knowledge base constructed in accordance with this concept was supplemented with information about Open Source solutions (ex. Jenkins, Subversion, Selenium IDE, Eclipse, JMeter and many others) to make it more

independent from IBM and more complete. Authors considered and recommend including other vendors from the market including Atlassian, VersionOne or Rally Software. Due to complexity of the model structure those vendors were not included in early version on the model. The authors carried out their task by functional and non-functional mapping onto the reference tool. The decomposition was achieved by a detailed analysis of the tools the authors, and direct contact with those responsible for the products (Product Manager) from IBM. The functional analysis of the selected tools revealed the limitations of the model at the very beginning of the study. Relying only on the functional decomposition of tools, it becomes impossible to assess their influence on an organization, in terms of implementation, thus overlooking a significant argument influencing both the effort and cost. The developed concept allows the tools' function to be separated from the established formal or agile process. On the one hand, the model can be applied to methodologies with a high level of formalism by choosing a larger pool of functionality, or on the other hand, it can head towards the Scrum and eXtreme Programming [15] methodologies, by selecting only the required elements. This separation from established processes allows the universal application of the model. It can be used in any organization regardless of the implemented process, with only expectations towards development in mind.

After analyzing best practices, the authors decided to apply the concept of architectural views which was already used earlier for modelling IT systems. The use of perspectives allows for easier management of a multi-dimensional structure, such as the reference model, while at the same time providing relevant information to the interested parties. The concept developed by P. Kruchten introduces five perspectives [8] (4+1 Architectural View Model), which affect different layers and aspects of the constructed system. These include functionality, business processes, logic, infrastructure and the internal structure of the solution. The model approach to modelling the architecture of IT systems allows the collaboration of many interested parties of a project on those elements which are most important to them, at the appropriate level of abstraction.

The authors are adapting this approach to embed it in the reference model for software life cycle management tools. The structure of this reference model includes the functionality of the tools, the infrastructure for

implementation and integration, the life cycle disciplines with best practices, and the roles involved in the project. The model structure is dynamic and allows for the arbitrary selection of perspectives, so that a more or less detailed information model can be achieved.

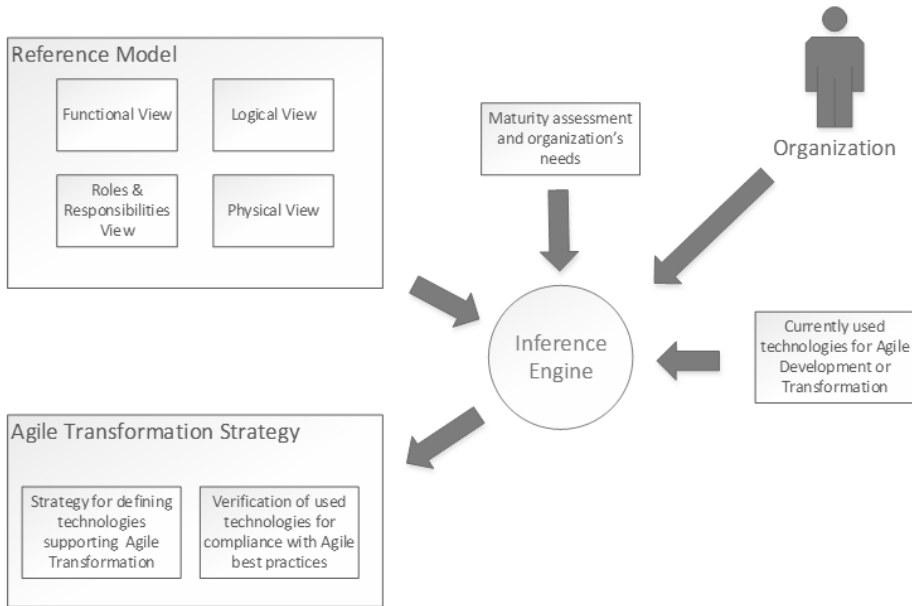


Figure 2. Reference model with inference engine for determining the technology aspects of Agile Transformation strategy

To specify the suggested perspectives in the reference model, elements of UML 2.4 were applied in the form of diagrams of use cases, implementation, packages and components. These diagrams fully support the concept of perspectives developed by P. Kruchten and they allow the structural description of this model. Reference model is supplied by information about popular tools that are used in processes of IT project management. This was helpful to define main areas of reference model that are presented on Figure 3.

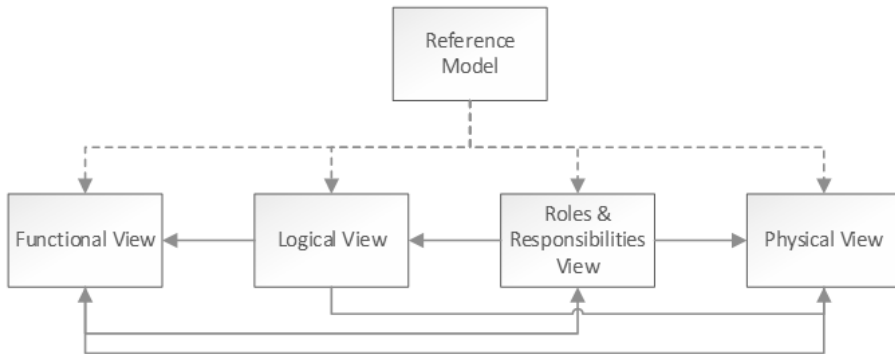


Figure 3. Structure of designed reference model for supporting technology determination in agile transformation projects

Functional perspective is basic architectural view allowing the analysis of functionalities associated with a particular branch of software engineering or a group of tools. The perspective shows the relationship between the functionalities provided in the form of use cases and the relationships between them. **Logical perspective** showing the relationship between the areas of developing processes proposed in the functional perspective and the relationship between Open Source and IBM tools which support those relationships. **Roles and responsibilities perspective** presenting relationships between the project roles and the functionalities assigned to them in tools and their functions in projects. The view makes it possible to verify which interested parties should be involved in the implementation of chosen software and who will be its target user. **Physical perspective** presents integration or its possibility between Open Source tools and IBM Rational. The perspective shows the already existing built-in data flows and indicates how much work must be done to combine the tools, if they are not yet integrated. An example of such a connection might be integration between the requirements management tools and the architecture management area.

4. Verification of ALMRef Reference Model in an Environment of Real IT Organization

Research on the developed reference model is verified on the basis of experiments (business cases) in the target environment of the IT project's customer, who plans to use a specific set of tools. The carried out research process includes verification of the model in both environments: the agile companies undergoing transformation and those that do not pass agile transformation, but they need the tools to adapt to the processes of software development. Verification in the first environment checks the application of organizational (management) of model. In the second environment it checks the importance of implementing and serves on tuning the model. Both verification environments are important for the technological perspective of agile transformation of IT organizations. Figure 4 shows the research process, from the construction of the model through its verification of corrective and improving actions.

Research on the developed reference model is verified on the basis of experiments (business cases) in the target environment of the IT project's customer, who plans to use a specific set of tools. The carried out research process includes verification of the model in both environments: the agile companies undergoing transformation and those that do not pass agile transformation, but they need the tools to adapt to the processes of software development. Verification in the first environment checks the application of organizational (management) of model. In the second environment it checks the importance of implementing and serves on tuning the model. Both verification environments are important for the technological perspective of agile transformation of IT organizations. Figure 4 shows the research process, from the construction of the model through its verification of corrective and improving actions.

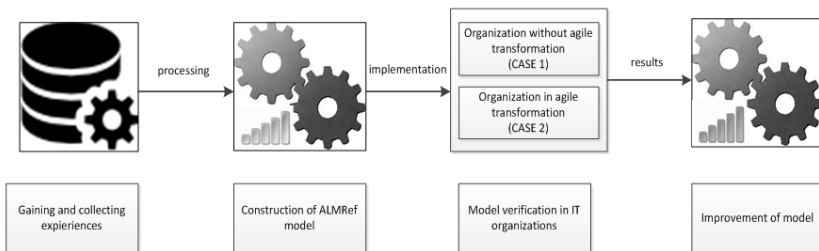


Figure 4. Process of the constructing the reference model

In order to verify the model there were chosen 2 environments of experiment. One of them concerned the organization which doesn't pass through the processes of agile transformation, but is very conscious of technological needs. So, according to verification, it is possible to check the correctness of the implementation of the model and prepare scenarios of its use in practice. On the other hand, in a second experiment, there was conducted research for organizations already passing agile transformation in order to verify if the reference model can help IT organizations in the technological perspective.

Both the organizations had been selected from a list of organizations, where the authors previously led advisory services and implemented a technology solution. Due to the time-consuming research and confidentiality of key processes of the organization, the conduct of research in an environment with limited access to information or resources would not be possible. Before carrying out the verification, there was a series of interviews made with the organizations in a way to prepare the research team to the verification process, the assessment of the maturity of the organization and severity in relation to agile transformation. Conducting research before starting research work was also aimed on selecting the scenario research, which will be best suited to the needs of the organization studied. According to the assumptions of the plan, information collected during the verification should evaluate the usefulness of the model in the real organization and be the basis for its modification or further improvement.

The environment of the first study was the project of implementing IBM Rational Collaborative Lifecycle Management (CLM) at the Chamber of Duty in Torun. As part of the program e-Customs, there are projects which will build the test environment in addition to the production environment of the system. Hence – that arose the need to standardize and improve the testing process SI SC by providing technical tools, organizational solutions and gather technical knowledge in the field of testing systems SI SC. In order to that tasks, there was appointed Project TestReg It includes e.g. project, realization, delivery, implementation and support for system TestReg maintenance. The context of the project and its completion at the Chamber of Duty in Torun has created ideal conditions for the verification of the created selection of the technologies reference model in the area of software development. Usefulness of model and its accuracy in the selection of tools under the previously defined

requirements and the compliance of the results of actually effected selection tool in the project were verified during conducted research. The result of the audit was to gather comments on its further improvement and development. The end of the first study and the introduction of the comments gathered during its implementation allowed to be ready to conduct further research in projects of greater complexity and scale. During the advisory work for one of the clients of IBM, there has been observed that it is the environment for the verification of the model and its support for the selection of technologies for adapting agile methodologies. In this case, the research environment was the organization of the public sector, which implemented the Scrum methodology, and wondered about the implementation of the SAFe approach. As the organization has invested in a set of tools to support the processes of software development, it wanted to assess whether currently held set of technologies will support their new choice – agile approach. The issue was a challenge for the organization both in the technology and further business development. The study was conducted to analyze existing research environment in relation to agile processes for which the organization wishes to migrate. Then there were compared the results obtained from the analysis of the consultants, which was carried out in parallel. The use of the model allowed to achieve results overlapped with the results obtained by a group of consultants, but in a much shorter time and with less work and required expertise of employees. The rest of this article presents the details of each of the studied business case.

4.1. Research experiment in the IT organization before starting the agile transformation process

Verification of reference model for the selection of the technology conducted in the framework of the project TestReg implemented for the Chamber of Duty was based on a survey shading level of maturity to carry out verification of business scenarios using the model. The study was aimed at selecting all potential business scenarios built in model which can be applied during the selection of technology in IT projects in the House of Customs. The conducted survey has allowed to the emergence of two scenarios: a selection of tools based on the fulfilment of functional requirements and the second – the statement of the tools which fully meet the activity of selected processes of

software development. The most time was devoted to the study of technology selection scenario based on the fulfilment of functional requirements, because in the opinion of the audited organization, it was best to show the reality of the implementation of previous projects. As part of the verification scenario, there introduced to the application all the functional requirements which have been defined in terms of the tools in the project TestReg. The experience was to check whether the application to verify the reference model will give the same recommendations for the selected tools. A set of studied tools narrowed there to the commercial products, which was compatible with the facts specified in the terms of reference.

Organizational requirements for tools published in the Terms of Reference (ToR) were the basis for the verification scenario. There is a need to introduce them into the application realizing business scenarios basing to reference model in a way to designation a set of tools that suits specific needs. In the case of the functionality defined by the Chamber of Duty, there was required mapping functionality for entries ToR collected in the reference model. The result was the coverage requirements for a set of features of the reference model in a one-to-one or one-to-many. There was conducted an analysis of selected features and then designated both a set of tools available to Open Source and commercial solutions. At the stage of application development for the analysis of the reference model there assumed that the solutions which meet at least 60% coverage requirements are recommended as a tool to implement. Chamber of Duty noted that in the case of the public sector, meeting the requirements must be close to 100%, and at 90-100% is acceptable. It highlighted that the parameter of acceptability should be a default with the possibility of modification at the stage of the analysis in the application. The results of the survey were summarized by the next survey, which assessed both prepared reference model and its application to the analysis. In the opinion of the Customs model usefulness is high and results are consistent with reality. Chamber of Duty pointed to the possibility of obtaining significant savings of time needed to analyze the areas for which tools are selected by using knowledge of the reference model and analysis applications. Important – from the perspective of the organization studied – was the possibility of the quick selection tool with specific features without the need to analyze the entire market available to suppliers. The completed surveys indicated a potential risk of

impact tool supplier for the content of data provided in the model and have influence on the generated recommendations. Significant from the point of view of the customer was to change the structure of the model to a more hierarchical, what allows to select functionality more easily. After the end of first of the scenarios, there was conducted similar research for other scenario, which was to prepare using a reference model compilation tools fully meet the activity of selected processes of software development implemented in the Chamber of Duty.

4.2. Research experiment in the IT organization passing agile transformation process

Another research experiment resulted directly from earlier research on reference model for the selection of technology in the organization. The organization has already used model to evaluate its usefulness in the simple case of selection tools based on the specified functionality. Trust to the reference model built thanks to the experience created in the organization interest in using the model in the process of evaluation of the currently existing tools supporting developing processes to support the currently implemented method SAFe (Scaled Agile Framework). Reference model created with the requesting application supports common methodology for the work of the team as Scrum and Extreme Programming, but at a stage when studies were conducted has not supported the methodology SAFe. As part of the experience there were created two groups of consultants, which were aimed at assessing the suitability of currently held tools (by the client), which were available in the context of support for the SAFe. The first group, in this case, used the model and the other based solely on their own experiences. In the case of the first groups, additional work was to complement in the reference model elements which were missing (in SAFe methodology that works at the level of teams is a process based on Scrum, already available in the model). As part of the experience in daily cycles, there were collected information from the teams' knowledge, the progress of the process of customer environmental analysis utility and readiness to issue a recommendation regarding support for the environmental tool from chosen by the customer methodology. The whole process of study was limited at the outset by the time up to 14 days and was not suppressible. For

the first group, which worked with the reference model, first week of work was dedicated to complement the reference model of the missing elements associated with the chosen by customer methodology. The level of progress on the evaluation of environmental utility and customer readiness to issue recommendations remained at very low levels. People working in this group clearly identified on the summary of the first week of work that in these areas have not made any progress. The second group working without a model, after the first week established the level of work on the analysis and assessment of the readiness issue as moderately severe and presented a much higher level of progress in relation to the first group. In the second week of the research, first group (because of the supplement has all the missing elements of the methodology Scaled Agile Framework) was able within the first two days of the week to reach the state of the second group at the end of the first week. Groups dedicated the next days to further analysis and preparation of recommendations. The results indicated and evaluated in much the same way the current status of held by the client tools and allow unequivocally make appropriate decisions related to the apply or not apply currently existing technologies. A significant difference between the prepared analysis was based on their level of detail and consultants' conviction for their confidence. The second group pointed to the low level of confidence, due to limited time and a small number of verified functionality. Group working with the model indicated a high level of certainty, due to the large number of functionality of the tools evaluated for compliance with the SAFE.

The experience made with the customer allowed to evaluate the use of the model in the process of agile transformation with additional adaptation of the model, which consisted of the introduction of the new methodology. Research of model for the designated by organization business scenario shown that it is possible to shorten the time required for the analysis of about 50 % at a time when the model is complete. Another important element identified by the organization is a significant higher level of assurance in the analyzes received, due to the amount of the assessed tools and functionality.

The audited organization has expressed interest in the possibility of using the model in other organization's agile transformation projects. Additionally, the studied organization pointed to a new path of development, which can help in the commercialization of the solution.

5. Changes to the Model Resulting from the Verification Process

Research conducted also at the Chamber of Duty in Torun clearly indicated the usefulness of the created reference model for the selection of technologies for both organizations undergoing transformation processes of agile, but also those still working in a more formalized way. In both cases, the reference model showed a significant benefit of reducing the time and improves the accuracy associated with the selection of technology for the organization.

During the study organizations drew attention to the need for continuous improvement model for its update to the current standing of available commercial products on the market and the changing trends in the world of agile methodologies. According to the surveyed organizations could represent a challenge for the project, which has a scientific nature. Another important element reported during the evaluation of the model was its independence from suppliers of technology, which is an important issue for organizations, which in Poland are subject to the processes of public procurement. The organizations reported concern in this case, that the model can be manipulated in such a way as to point to a specific provider of technology.

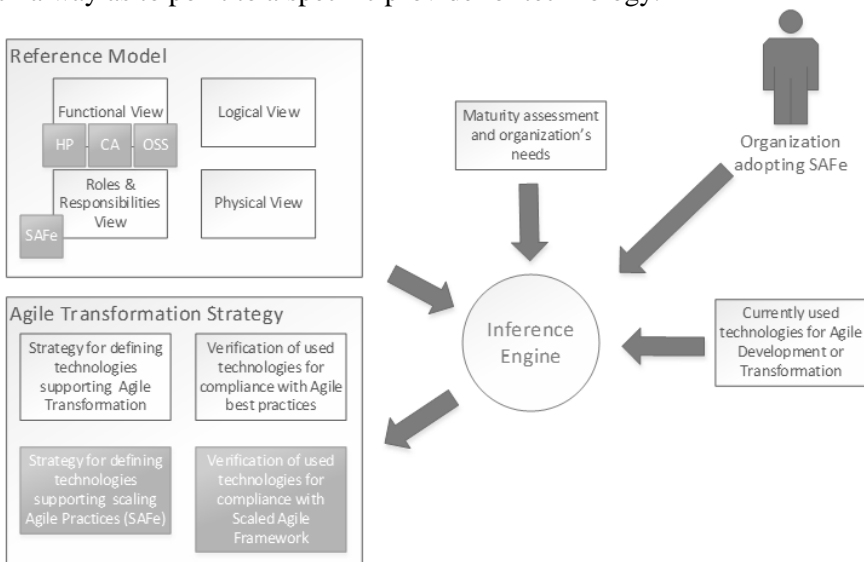


Figure 5. Structure of extended ALMRef reference model after applying outcomes of verification process

Both organizations indicated, even after additional analysis, that the openness of the model and its expansion can be a guarantee of its independence. Another element that was raised both the Chamber of Duty and the other of the surveyed organizations, was the potential to extend the model to other areas of technology selection, which went to the scope of processes of software development and also took into account the areas of project management, enterprise architecture and construction and maintenance services in based on the concepts of ITIL (Information Technology Infrastructure Library).

All of the identified areas of model development were analyzed by the authors and are valuable suggestions for the next steps of research and research directions for both the development of reference models, and the processes of agile transformation. The authors of the publication here draw attention to the wide interest in the organization of business character of this area of research as well as their active participation in the processes of verification and assessment of the current state of research. This allows to say that this area is not only an interesting research issue, but also a big business problem which IT organizations have not been able to cope yet. Conducted as part experiments studies confirmed posed at the beginning of the research thesis and allowed for complete verification of the reference model by using all available perspectives and available business scenarios provided for agile methodologies. Conducted experiments allow you to clearly determine the suitability of the created reference model in the processes of agile transformation on the basis of the results obtained.

References

- [1] R. Ryan Nelson. *IT Project Management: Infamous Failures, Classic Mistakes, and Best Practises*, MIS Quarterly Executive, Vol. 6 No. 2, University of Virginia, 2007.
- [2] R. Pichler. *Agile Product Management with Scrum. Creating Products that Customers Love*, Addison-Wesley Professional, 2010.
- [3] VersionOne, *VersionOne The 10th annual State of Agile Report*, 2016, <http://stateofagile.versionone.com/>
- [4] R. C. Martin. *Agile Software Development: Principles, Patterns and Practices*. s. 1, Prentice Hall, 2002.
- [5] K. Schwaber, M. Beedle. *Agile Software Development with Scrum*, Prentice Hall, 2001.

- [6] K. Beck. *Extreme Programming Explained: Embrace Change*, s. 1, Addison-Wesley, 1999.
- [7] B. Chrabski. *Integration and Dependency in Software Lifecycle based on Jazz platform, Problems of Dependability and Modelling*. Monographs of System Dependability, red. J. Mazurkiewicz, J. Sugier, T. Walkowiak, K. Michalska, Oficyna Wydawnicza Politechniki Wrocławskiej. Wrocław, 2011.
- [8] P. Kruchten. *Architectural Blueprints – The “4+1” View Model of Software Architecture*, IEEE Software 12 (6), 1995.
- [9] C. Orłowski, T. Deregowski, M. Kurzawski, K. Ossowska, A. Ziolkowski. *Wykorzystanie miar złożoności projektu do oceny stanu ewolucji organizacji informatycznej, Innowacje w zarządzaniu i inżynierii produkcji, Tom 2, Oficyna Wydawnicza PTZP, Opole, 2016.*
- [10] Z. Kowalczyk, C. Orłowski. *Advance Modeling of Management Processes in Information Technology*, Pomorskie Wydawnictwo Naukowo – Techniczne, 2012
- [11] T. Deregowski, A. Ziolkowski. *Hybrid approach in project management – mixing capability maturity model integration model with agile practices*, Social Science (Socialiniai Mokslai), Nr. 3 (85), 2014.
- [12] D. Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises*, Addison-Wesley Professional, 2007.
- [13] <http://standishgroup.com/>, accessed: May 28, 2016.
- [14] Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch, <https://www.infoq.com/articles/standish-chaos-2015>, accessed: May 28, 2016.
- [15] K. Beck, C. Andres. *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2004.

Chapter 15

Building Project and Project Team Characteristic For Creating Hybrid Management Processes

1. Introduction

To be able to manage constantly changing requirements and to deal with constantly changing environment IT organizations need to go through Agile Transformation process. Agile Transformation is understood as applying Agile mindset, which is not the same as using Agile Software Development Methodologies. Agile mindset is understood as constant adaptation to changing requirements, circumstances and environment.

Adaptation process should happen on various different levels. One of it is the choice of adequate project management processes and software development methodology for each project, which is about to start. Within the same organization there could be a need to deliver various projects, which differ on important aspects such as the amount of budget, project duration, number of team members, client specific and business domain in which client organization operates. Each of the projects requires dedicated set of processes and software development methodology adapted to its unique characteristic and needs, which will enable directing project work in a way, which increases the probability of project success.

Over the past few decades dozens of different standards and methodologies for supporting software development processes were developed. Many process improvement models, such as Capability Maturity Model Integration (CMMI) [5] have been developed. They define in details the full life cycle of software development project, from collecting requirements through design, development, testing, up to delivery and support. They are tools and methods

agnostic; they concentrate on what and why should be done, not defining how it should be done. In addition to models, there are many different project management methodologies, grouped in two main categories: Agile and Waterfall. Traditional approaches, often called Waterfall after their graphical representation, contain several independent phases such as requirements gathering, design, implementation or testing. Each phase is performed indecently; transition to the next phase is possible only if previous phase was completed. In opposition to traditional approaches, Agile methodologies use incremental and iterative approach. Project is divided into phases called Sprints. Each Sprint usually lasts two weeks. During single Sprint project team performs activities related to design, testing and coding. Each Sprint ends with increment of functionalities, which can be released and presented to the client. Agile methodologies were formed in opposition to traditional, sequential model. They are very informal and simple. In Agile methodologies practitioners make most of important decision during the project.

Described standards and methodologies, both Agile and Waterfall, were used to manage many different software development projects. There are many examples, where projects were managed with Agile methodologies and they succeeded. In many other cases projects failed despite using Agile approach. Similar situation is with waterfall methodologies, many projects, which used Waterfall approach succeed. There are also many examples of projects, which were managed with Waterfall approach and failed. It doesn't say anything about methodologies, models and their effectiveness. Project failure or success is a consequence of choosing proper methodology to manage project with particular specific.

The choice of Software Development Methodology is a key decision, which could change project result. Unfortunately existing knowledge does not provide method, which could help IT organizations with selecting proper, adjusted to project specific, methodology. There is no method which lets analyze project, client and delivery organization specific and basing on analysis results suggest the most adequate software development methodology.

This paper is an attempt to fill this gap. It contains a description of method, which allows building multifaceted project, client and delivery characteristic. Such characteristic can be used for many different purposes, including selecting adequate methodology for the project.

2. Client Characteristics in the Context of Model for Designing Hybrid Management Processes (DHMP)

Authors of this paper, in the course of their daily work related to project management, spotted a problem associated to lack of tools and methods, which allow choosing adequate, processes and methodologies, adjusted to unique needs of the project. This problem was described in one of earlier publications “Model for building Project Management processes as a way of increasing organization readiness for Agile transformation” [3]. Proposed in this article Model for Designing Hybrid Management Processes (DHMP) defines concept of building custom project management processes, which are adjusted to unique project needs. Processes are based on results of project, client and delivery organization analysis and they try to fill all the gaps in existing processes and address specific needs resulting from project characteristic.

Module, which is responsible for building the characteristic of project, client and delivery organization, is a key element of DHMP Model. Based on project characteristic generated with this module, DHMP Model is able to define which processes, project management methodologies, engineering and process optimization techniques are most suitable to the needs of analyzed project. In the last phase DHMP Model combines all selected processes and tools into consistent project management process.

Project management processes, created with help of DHMP Model, are recommended as most adequate and suitable process for managing particular, process, which could significantly increase the probability of project success.

Figure 1 presents an approach, which was used to build project characteristic.

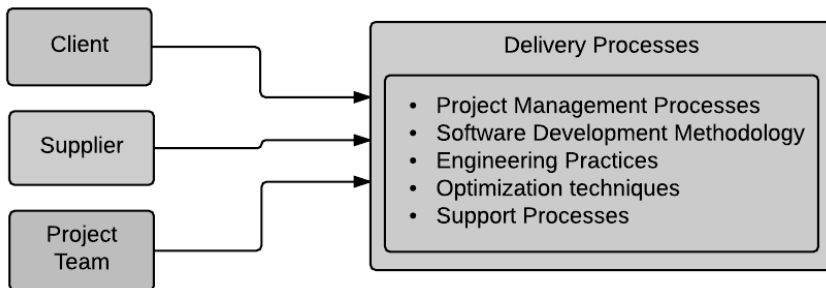


Figure 1. Characteristics of the client in the context of DHMP Model

In order to verify DHMP Model a survey was created. Survey is an integral part of DHMP Model and is responsible for creating project characteristic. Analyzed in the survey aspects include, among others, project team, management processes and tools used to support software development processes. Survey allows evaluation of multifaceted complexity of the project and project team. Project characteristic build with survey will be used as a source of data for Knowledge Base. Knowledge Base is another integral element of DHMP Model and is used to store information on completed projects. Based on information included in Knowledge Base, organization will be able to build a set of recommendations for new projects, which are about to start.

Preparing survey and conducting pilot study allowed identification of projects' key accepts which are crucial for the selection of software development methodologies. It also allowed refining and verifying DHMP Model.

Survey contains of two parts. First part is about analyzing the specific of project and project team. Second part is used to build characteristics of processes, methodologies and tools used by delivery organization. Survey and findings from pilot study are presented in later sections of this paper.

3. Study of IT Organizations with DHMP Model

Pilot study was carried out in a software development company. The company has more than 4,000 employees, distributed among different localizations such as United States, South America, Australia, Asia and Europe, including Poland. Solutions and products provided by company are related to multichannel marketing [1], data mining [15] and Big Data [11].

Pilot study had two main objectives. First goal was to verify survey, it's structure, aspects it measures and usefulness of collected data in preparing the characteristic of project and project team. Second goal was to verify DHMP Model, check if when provided with data collected through survey, it can be used to draw useful conclusions about projects and processes.

During pilot study six projects were analyzed. Projects have varied on size, budget, length and project phase. All projects were for different clients. One of the projects was for a client form publishing business, two projects were for clients from automotive industry, one for Tobacco Company. The last

two projects were about creating standard products, which were sold to different clients from different business sectors.

4. Analysis of the Project and Project Team

In the first part of the survey, three aspect of the project are examined. First of them is project, second is project team (delivery), third one is the result of the project. Each aspect is described with series of parameters; each parameter is examined through a series of questions. Answers to questions determine the complexity of each parameter. The complexity of the individual parameters builds up the complexity of related aspect. Complexity of parameter can be low, medium and high.

The structure of the questionnaire for building project and project team characteristic has been presented on Figure 2.

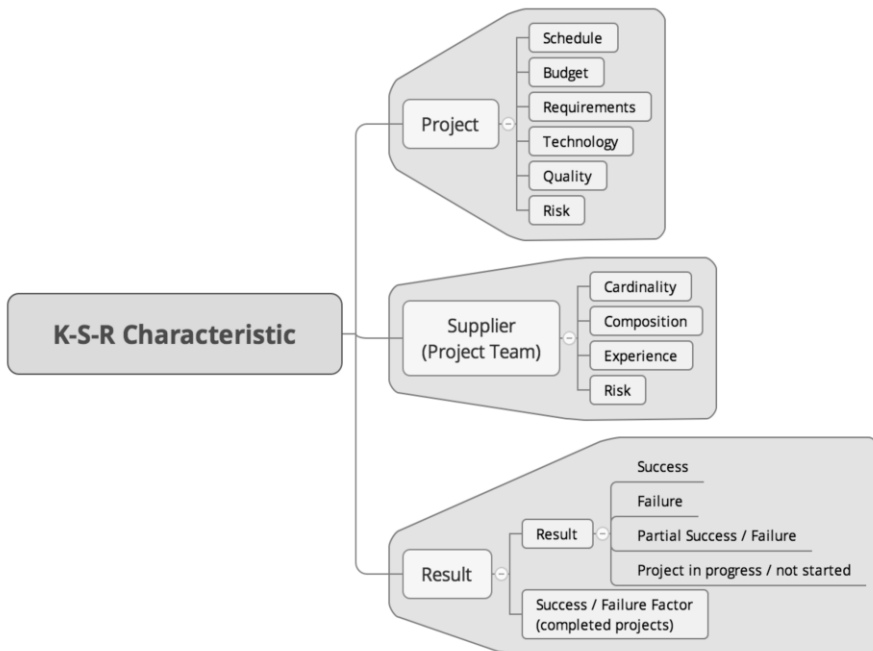


Figure 2. Aspects examined as a part of building project and project team characteristics

4.1. Project complexity

The complexity of the project is determined on the basis of six parameters: budget, schedule, requirements, technology, quality and risk. As was mentioned before, complexity of each parameter is evaluated through series of questions. Structure of questionnaire and factors, which determine parameters complexity, are described in this section.

The first of analyzed parameters is budget. Fact, if budget has been or has been not defined, has the biggest impact on budget complexity. If the budget was formulated, survey checks what is the total project cost (less than \$250k, between \$250K and \$750K, over \$750K). Higher costs mean higher complexity. Last factor, which determines project complexity and is measured through survey is an experience has in delivering projects with similar budgets.

Second aspect of the project, which is assessed as a part of determining project complexity, is schedule. At first survey checks if schedule for the project was defined. If answer to this question is yes, survey checks project length (it could be 6 months, between 6 an 12 months and over 12 months). The longer project last, the higher schedule complexity is. Last factor related to schedule complexity, which is measured by the survey, is how important analyzed project is for the delivery organization. If project has key meaning for delivery organization, it can be assumed that all necessary resources will be secured and provided. If project has medium priority, it means that it will need to compete about resources with other projects, with higher or the same priority. Finally, if project has low priority, there is a huge risk that common resources, which are shared with other projects, will be delivered only if other projects won't need them. In last two cases there is a significant risk that project will be delayed due to lack of important resources.

Next aspect, which has an impact on project complexity, is requirements. First and foremost it needs to be determined whether requirements for the project were defined. If they were, survey checks how flexible they are and what's the risk that they will change during the project. Requirements complexity is low, if they are well documented and formal process of managing requirements exists. It's even better for requirements complexity, if changes in requirements require contract renegotiation. In such case, it can be assumed that changes in requirements won't happen too often. Requirements complex-

ity will grow if they were defined, but delivery and client teams assume that they can be easily modified. The most complex situation with requirements is when they were defined on very basic level and assumption was made, that client will discover his needs during the project and requirements will be defined on the go.

The next aspect, which can significantly influence project complexity, is technology. One of the key factors, which could significantly increase technological complexity, is the need of integration with external systems, especially if team members don't have much experience in this area. The number of technologies used in the project will also shape technological complexity. Level of technological complexity will grow together with the number of technologies used in the project. Even single technology used in development project can be very complex. When there is a need to use many different technologies, when they start interacting with each other, level of complexity grows significantly. Another aspects related to technology, which needs to be taken into consideration, when analyzing project complexity, is knowledge and experience team members have with each technology used in the project.

Another aspect, which shapes project complexity, is quality. Survey checks if quality assurance is implemented in the project and in which project phases. It's crucial for project quality, whether the mechanisms of securing it exist on each project level or only during test phase. It's also important if project team uses well-known standard tools and techniques for increasing quality of the code during whole project lifecycle. Mentioned techniques are Continuous Integration [7], Unit Testing [2], Code Reviews [2], Static Code Analysis [4], Continuous Delivery [9] and Continuous Deployment [9].

The last project aspect, which is taken into consideration when evaluating project complexity, is risk. Survey checks if during the project risks are actively managed. Risk impact on project complexity is much lower when risks are identified and prioritized during the project and risks mitigation plan is created and maintained. It's also important if risk mitigation plan is part of project plan. Risk also depends on team experience in delivering projects of similar scale.

4.2. Project team complexity

Main factors, which determine Project Team Complexity and are evaluated as a part of survey, are: size of the team, its structure, experience and distribution.

Complexity of project team grows together with the number of team members. That is why survey measure the number of team members. Another important factor, which impacts team complexity is team structure. The experience of authors of the study and its participants suggests the team complexity is lower if team is build from client and delivery representatives. Complexity grows when team consists of delivery organization representatives and doesn't contain any client representatives. The most complex situation is when team contains external suppliers who act as contractors.

Another important factor, which shapes team complexity, is experience team members have in working together. The higher common experience is, the lover impact on team complexity.

The last factor, which has an impact on team complexity, is geographical distribution of the team. If team members are grouped in one localization, impact on project complexity is low. It grows, when team members are distributed but they work in similar time zones. Geographical distribution has highest impact on the project when time differences are greater than six hours. In case, when time overlap is two hours or less, work, as one team becomes a real challenge. I such case additional effort related to documentation and good collaboration tools are required.

4.3. Project result

The last factor, which is evaluated as a part of survey, is project result. As mentioned earlier in this paper, survey can be used to examine projects, which have ended, and these, which are about to start. In case of projects, which have ended, survey evaluates they result and factors, which decided about it.

Survey tries to determine main factors of project success or failure. Interviewed person is asked to provide four most important factors, which de-

cided about project result. List of possible factors is based on Processes Areas defined in CMMI for Development (CMMI-DEV) [5].

CMMI is a process improvement method, which integrates all process and procedures existing in an organization and identifies potential gaps. CMMI model is often used to assess maturity of process implemented in the organization. CMMI provides holistic approach, it does not concentrate on specific methodologies and tools, it is method and tool agnostic. CMMI defines which process should be implemented in an organization and why, but it doesn't define how they should be implemented. Thanks to that it can be used with all types of methodologies, models and techniques, both Agile and traditional. Within the CMMI-DEV model 22 process areas are defined. Each process area describes key aspects related to particular area of software development process.

In survey only 18 from 22 Process Areas are used. Remaining four Process Areas concentrate on measuring and improving processes on organization, not project, level and that is why they have been omitted.

Interviewed person, when choosing key indicators which determined project results, can choose from the following factors:

- Configuration Management
- Measurement and Analysis
- Project Monitoring and Control
- Project Planning
- Process and Product Quality Assurance
- Requirements Management
- Supplier Agreement Management
- Decision Analysis and Resolution
- Integrated Project Management
- Product Integration
- Requirements Development
- Risk Management
- Technical Solution
- Validation
- Verification

Defining factors, which determined project result, is crucial for building recommendations for new projects. Basing on historical data, which contains

information on project characteristic, project result and key factors which determined result, decision makers in delivery organization can gain knowledge on which processes are really important for projects with particular characteristic. They will know what needs to change in project management processes, which processes needs to be added and which needs to be removed, to significantly increase the probability of project success.

5. Analysis of Processes, Methodologies and Software Development Standards

The second part of survey focuses on the project management methodologies, processes and standards. Survey can be used to research projects, which have ended and those, which are about to start. Studies on finished projects focus on processes and standards, which were used to manage them. For projects, which are about to start, survey concentrates on processes and standards, which project team is going to use to manage the project.

They are two main goals of this part of survey. First goal is to determine which areas were (project completed) or will be (projects which are about to start) formally managed. Second aspect of processes, which is examined, is level of their capability. Capability level is determined by four factors: level of project documentation, information on trainings, which were performed in the past, the fact if processes and their products are audited and how well processes are standardized on project and organizational level.

Survey also examines which Software Development Methodologies, Engineering Practices and Optimizations Techniques were (will be) used to support the implementation of the project.

The structure of the second part of the survey, which was described in this section is shown in Figure 3.

The main goal of this part of survey is to determined, which processes were (will be) formally managed, as a part of official project management process.

Processes for which it is checked whether they are a part of software development process or not and what is their capability level, are identical to the CMMI-DEV processes described in section 3.1.3 (Configuration

Management, Measurement and Analysis, Project Monitoring and Control, Project Planning, Process and Product Quality Assurance, Requirements Management, Supplier Agreement Management, Decision Analysis and Resolution, Integrated Project Management, Product Integration, Requirements Development, Risk Management, Technical Solution, Validation, Verification).

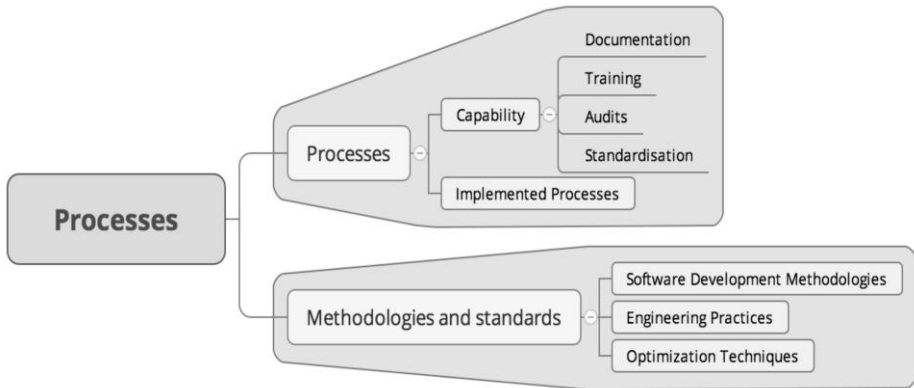


Figure 3. Process aspects examined as a part of building project and project team characteristic

As it was mentioned earlier in this section, apart from checking which processes are managed in a formal manner, survey also measures the level of their formalization. Level of process formalization depends on documentation, trainings, process audits and standardization level.

Processes and procedures used to manage project can be documented in details, briefly documented, or not documented at all. Apart from level of documentation, also the way in which processes were established may differ. Process could be introduced in a very formal manner, which includes trainings and workshops, supported by detailed documentation. Introducing new processes could be also limited to providing documentation. In the worst case process can be introduced without any organizational support.

As a part of survey it is also checked which processes are audited and how often audits are performed. The last factor, which is associated with process capability and is measured in the survey, is how well process is established in organization. In general, processes can be established on project level, organizational level or not established at all.

The last part of the survey allows determining which methods, engineering techniques, process improvements techniques and other IT standards were used (will be used) as a part of project management processes. Survey contains the following potential answers: PMBOK [10], CMMI [5], SixSigma [8], PRINCE2 [6], Scrum [14], Lean Development [13], Kanban [12], and XP [2].

Apart from information on which standards were incorporated, it's also important to know how well these standards are known to project team. Level of knowledge is determined by two factors: theoretical knowledge and practical experience (how long most of the team uses particular technology in their everyday work). Both factors are measured as a part of survey.

6. Impact of Pilot Study on DHMP Model

As a result of pilot study, the range of survey was reduced. Scope of survey was limited to these project and project team parameters that are crucial for building project characteristic. At the same time some new questions and sections were added to the survey. After changes survey concentrates more on these aspects of project and project team, which are crucial for the choice of right methodology for the project. Aspects, which don't influence the choice of processes supporting the management of the project, were eliminated from the survey.

Number of aspects, which are measured by the survey and level of details, is a compromise between the need of precise and detailed data and time needed to fill out questionnaire. Too general questionnaire won't deliver useful information. Too detailed questionnaire could significantly reduce the number of entities willing to take a part in survey.

Initial version of survey contained of sixty-two question and time required to fill it out was more than 90 minutes. After pilot study, number of questions and time needed to fill out questionnaire, were reduced by a half.

One of the main changes was removing a section related to building client characteristic. Client aspects, which are important for building requirements for project management processes, are analyzed as a part Project Team. According to feedback received from survey participants, client and delivery

should be examined as a part of the same entity: Project Team. Client shouldn't be analyzed in isolation from delivery.

From survey they were also removed questions that explored the state of client and delivery organizations. It was recognized that it is better to focus on individual projects than on whole organizations. Lessons from previous projects are more important for constructing project management process, then information on how the whole company is functioning.

Another effect of pilot study was changes in selection of project and project team key factors. In initial version of model, an assumption was made that there is a need to measure project and project team, but key parameters weren't specified. That has changed after pilot study. Basing on feedback received from study participants, key factors that will be used in the future studies to build project characteristic were designated. For project team these factors are: team cardinality, team composition and team experience. For project key aspects are schedule, budget, requirements, technology, quality and risk. Each aspect is measured with series of questions.

Another modification that was introduced as a consequence of pilot study, was adding section for measuring capability of software development methodologies, engineering practices, optimization techniques and support processes. Capability level defines how well each process and technique is established in an organization. Survey measures that by checking how well each processes, methodology and technique are documented, what types of trainings were performed in the past, if processes are regularly audited and if they are standardized on project and organizational level.

Important changes were also introduced in survey structure. Questions were grouped into logical section, which concentrate on key project aspects such as processes, methodologies and standards, project, project team and project result.

Modified DHMP Model structure is presented on Figure 4.

Changes in the type of the collected data and its structure impacted DHMP Model. In consequence of pilot study the need of storing and processing survey results was identified. To address it, new concepts of Knowledge Base and Rule Engine were introduced. The aim of knowledge base is to store survey results in a way, which enables analyzing organizations and its processes. Rules Engine is going to be a main tool used to analyze historical data

and to build set of recommendations for new projects. Both concepts will be developed in the course of further research.

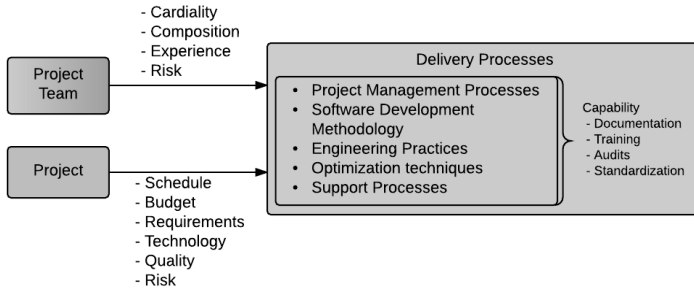


Figure 4. Evolution of DHMP Model and its module for building delivery characteristics

7. Conclusions

Ability to build custom management processes, which are tailored to the unique project needs, may be crucial for the success of IT organization. Such tailored, bespoke process needs to be based on data, which in details describes the needs of organization and its projects. Survey described in this paper is an attempt to introduce new method for collecting data on IT organizations, their clients and projects. Basing on survey results analysis the set of recommendations can be build. Recommendations could include suggestions on most adequate to project needs software development methodology, engineering techniques and processes and optimization standards.

In this paper authors focused on collecting data on projects, they specific and processes. In their future work they want to continue studies in this area but concentrate more on methods of analyzing collected data, drawing conclusions and building sets of recommendations. They intend to develop two concepts, which were introduced in this paper, the concept of Knowledge Base and Rule Engine. As a part of further work more IT organizations and projects will be examined with presented survey. Valuable recommendations need to be built on wider set of data.

References

- [1] A. Arikan. *Multichannel Marketing: Metrics and Methods for On and Offline Success*. Sybex, 2008.
- [2] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
- [3] C. Orłowski, T. Deregowski, M. Kurzawski, A. Ziolkowski. Model służący kształtowaniu procesu zarządzania projektem informatycznym dla podniesienia gotowości do zwinnej transformacji organizacji informatycznej, *Innowacje w zarządzaniu i inżynierii produkcji*, Tom 2, Oficyna Wydawnicza PTZP, Opole, 2016.
- [4] B. Chess. *Secure Programming with Static Analysis*. Addison-Wesley Professional, 2007.
- [5] M. B. Chrissis, M. Konrad, and S. Shrum. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Addison-Wesley Professional, 2011.
- [6] O. Commerce. *Managing Successful Projects with PRINCE2*. The Stationery Office, 2009.
- [7] P. M. Duvall. *Continuous Integration. Improving Software Quality and Reducing Risk*. Pearson Education, 2007.
- [8] S. L. Furterer. *Lean Six Sigma in Service. Applications and Case Studies*. CRC Press, 2009.
- [9] J. F. Humble,. *Continuous Delivery. Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, 2010.
- [10] *A Guide to the Project Management Body of Knowledge: PMBOK(R) Guide 5th Edition*. Project Management Institute, 2013.
- [11] N. Marz. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications, 2015.
- [12] T. Ohno. *The Toyota Production System: Beyond Large-Scale Production*. Productivity Press, 1988.
- [13] M. P. Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.
- [14] K. B. Schwaber. *Agile Software Development with Scrum*. Prentice Hall, 2001.
- [15] I. H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.

Chapter 16

Experience Report: Process of Introduction of DevOps into Production System

1. Introduction

The term DevOps (an acronym for Development and Operations) has gained much importance in the world of information processing in the recent years. DevOps lacks a formal definition but can be commonly understood as means to exchange feedback between developers and operations engineers to improve the software itself, the processes that all parties are following and the environment where the application is hosted.

The focus on automation and synergies that emerge from merging different IT aspects drives more and more business value to the customers and helps to save costs for the software houses. The aim of DevOps is to create a living and transparent ecosystem, where every phase of the software development follows each other quickly and automatically [16]. The main reason for applying DevOps in the development process is the enabled hyper-productivity, because of the ability to release very often (i.e. daily or even several times per day) [26]. There is a strong connection between the Agile and the DevOps (Figure 1) movements [3]: DevOps is helping what agile started. The agile technique introduces some flexibility to the development process and iterative approach in the collaboration between business and development (Figure 1). The DevOps technique tears down the walls between development and infrastructure.

The DevOps is based on four fundamental pillars (Figure 2). The people aspect concentrates groups of developers, testers, infrastructure administrators and clients together to collaborate with each other. The second and the third

aspects focus on automating the processes with the right tools to achieve maximum throughput in the development cycle. The last aspect – architecture – makes sure that the possibilities of many releases often, is used to its maximum extent. One of the most suitable architectures that fit great into DevOps is the microservice style, where small separately deployable software chunks that communicate with each other are implemented. The aim of the microservice architecture is to test easily and scale well [12, 5].

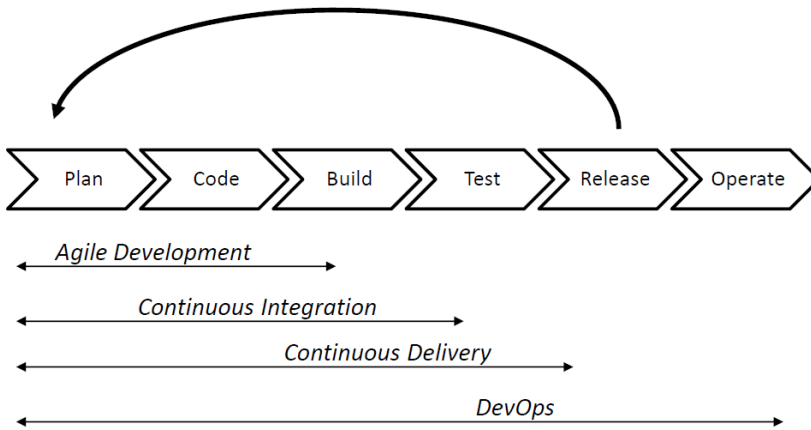


Figure 1. DevOps in development process

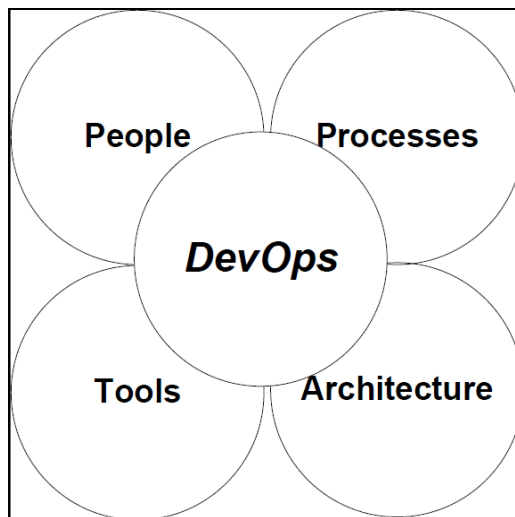


Figure 2. Main facets of DevOps

The main contribution of this paper is to present the process of the introduction of DevOps elements into the production system Automotive System. The system was designed using waterfall methodology and was transformed into Scrum. The DevOps pillars are accompanied by the continuous everything (continuous integration, continuous testing and continuous deployment) approach. The used tools and processes will be described. Some metrics will be proposed and the results will be discussed.

This paper is organised as follows: a related work is presented in the Section 2. Section 3 describes services concept in the organisation. Section 4 presents production system that used DevOps mechanisms and their metrics. General results and conclusions are discussed in Section 5.

2. Related Work

In the year 2013, Puppet Labs *State of DevOps Report* [25] benchmarked over 4000 IT organisations to better understand the companies at all stages of DevOps adoption. They noticed that the DevOps culture enables high performance by increasing agility and reliability. They claimed that the companies, which adopted DevOps techniques and culture, deploy code 30 times more often and have 50 percent fewer failures. They analysed and identified barriers to DevOps adoption. They noticed that the processes and tooling contribute to high performance, but without cultural change within the organisation, it will not be possible to implement the DevOps concept completely. The most common barriers to DevOps adoption are: lack of manager or team buy-in or the value of DevOps is not understood outside of a specific group.

In the *2014 State of the DevOps Report* [27] they additionally noticed the meaningful increase in the job satisfaction in companies adopting DevOps culture which in turn boost their performance. In the last report [26] they noticed growing performance of the companies adopting DevOps and additionally stated need of lean management and diversity in the companies as one of the important factors for DevOps culture adoption.

In the study *World Quality Report 2015-2016* [4] authors notice the strategic role of QA, testing automation and the ongoing DevOps adoption. Additionally to traditional QA tasks they state the need for the predictive analytics

usage in various stage of the project to shorten the lead time³, testing efforts and manage the risk before the production.

The company Paddy Powder [6] introduces continuous delivery and Kanban technique mainly to improve time to market. After the transition they noticed huge improvements in several areas: accelerated time to market, building the right product, increased productivity and efficiency, stable releases, improved quality, improved customer satisfaction. They shortened time to market every six months to at least once a week and the cycle from user story conception to production shortened from several months to several days. The number of open bugs decreased more than 90 percent. Bugs are treated the same as the new features and are added to the Kanban board. Hence, no dedicated bug fixing process is required. They recognised several changes: organisational (barriers between competing divisions), process-wise (delays due to approval boards) and technical (lack of a robust out-of-the-box³ solution).

Another example is the Etsy DevOps case study [9]. Jon Cowie emphasises that introducing DevOps approach in the enterprise is not one time transition. It requires the general business culture change in the long term. The DevOps way of working is especially hard when one tries to introduce DevOps culture in the large company, where different stakeholders on various levels do not want to give up their control to the team. To convince the management board, one has to show and emphasise business benefits of the new approach. According to the case study, they push changes to the production every 20 minutes with little impact on end users using the Chefs [7] technology.

In the [7] Neely and Stolt show the transition from time-boxed Scrum to the Scrum-ban [17] based continuous delivery process. They describe the two different views: from business and engineering perspective. Besides the technology and testing efforts, which have to be introduced to get automatisation, they once again emphasise the need for the business people commitment, especially in experimenting. They noticed that the monthly story throughput per developer increased and the number of bugs dropped and became predictable because of frequent release (no more bug peeks).

³ Lead Time – the time between the initiation and completion of a production process. Often used in term of lean management.

3. Services Concept in the Capgemini Organisation

One of the main issues in the development process is to keep the costs low. Much effort is put into increasing the productivity of the company as a whole. The Capgemini introduced the concept of services that are provided to the project teams whenever they need to use or to configure a new tool and / or process. The whole concept is based on the assumption that specialised units use the tools and processes more efficiently and at lower costs than the project teams that would have to master them through a long learning process. As the DevOps principles rely hugely on automation using many different tools, the services can potentially provide even more productivity to the project teams.

The services have been divided into two subcategories: the so-called shared services that concentrate on single tools or processes and so-called expert services that glue the tools together to form a functioning deployment pipeline [15] or to provide other specialised services like complete test automation for a project. The service approach helps to achieve maximum utilisation of the experts' knowledge as they are not exclusively reserved by some projects and their efforts can be redirected to the project that has the most needs at a given moment in time.

The shared services can help the projects save costs as their work has been structured into easily estimable processes and are provided in the Capgemini Righthore® [13, Chapter 1] delivery model.

4. Introducing DevOps Culture and Technique into the Automotive System

4.1. Background

The project is being developed for one of the biggest automotive companies in Europe, and it covers all aspects of car purchasing. The system is used only internally by the customer and manages the whole process of car renting, leasing and selling.

The system is made of 2 subsystems in two different technologies. Its first version was written in COBOL and is now to be replaced by a new

web-based application. Both subsystems are working in parallel, and the data is synchronised between them in real time. The client part of the system has been written using JSF (Java Server Faces) and JavaScript, while the server tier uses Spring Framework, Hibernate, Spring Batch and Spring WS. The database used for the project is IBM DB2. The project is manufactured using agile methodologies. Regular user stories are developed using Scrum framework. Support and bug fixing are hierarchically organised using Kanban technique as described in [22].

The team consists of 4 business as well as technology oriented sub-teams. The project is delivered from 3 different locations. This type of work has been organised in accordance with the Distributed Scrum concept as described by Majchrzak et al. [23].

4.2. A timeline

The old system has been developed since 1990 using pure waterfall life-cycle model. The new version of the systems was implemented at the beginning also using the waterfall software development model with the sharp division between business, development and operation organisations. First release after 16 months showed that we were not able to integrate with the old system and we did not cover minimal end user needs. Additionally, we were not able to establish effective communication with the operations team on the customer side and thus, all the changes and agreements were done in the conservative and document-driven manner which in turn had slowed our release process significantly and analysis of the production problems | mainly because of missing access to the infrastructure. In 2012 to improve at least some team and customer collaboration aspects, it was decided, that the project will be developed as one Scrum project. After final transition in 2013, the team, as well as the business, were using the Scrum framework – except the operations team.

Another important aspect which has been evolving since the project beginning is the focus on continuous project and test infrastructure improvement. We decided from the very start to invest in full test automation, continuous integration and clean code rules. While the development team requires only a few hours to deliver the new version of the system, e.g. after critical bug fix, the operations team is still the bottleneck in the process.

4.3. Engineering practices

To introduce DevOps principles we focus on the process automation (if possible), on XP techniques and simplicity.

Collaboration (Wiki, Issue Tracking System). Collaborative and agile approach determined the selection of tools. We decided to use leading agile tools like JIRA [1] and Conuence [1] to allow cooperation in the distributed environment.

Clean code and coding rules (Findbug, PMD). Each user story or task has to meet the Definition of Done (DoD), agreed both internally by the time and with the customer. We focus mainly on Clean Code rules proposed by Martin [19] and on lack of Findbug [11] and PMD [28] violations. The static analysis is executed centrally on Jenkins and in the case of the threshold of violations is reached the code corrections have to be immediately introduced.

Code Review, Pair Programming. All user stories and defects are being reviewed in the peer review process (Crucible [1]). In the case of crucial requirement or in the case of the job training we use Pair Programming [2] technique.

Continuous Integration. 2 Jenkins manages all integration tasks and run on several slaves. The extended configuration is required because we want to provide continuous feedback to the developers.

E2E Automation (xUnit, Selenium). We decided from the very beginning to invest in full test automation, in other words, all the business processes in our applications are tested automatically. We extended Selenium WD [29] framework to test Webservices and long running jobs. Instead of focusing on single use case we try to cover the whole business process to simulate real application use – even if the process takes, in reality, several days.

Automatic Deployment (Jenkins). We introduced automatic application deployment on all our tests systems including Tomcat and Websphere application servers.

4.4. DevOps introduction stages

Our way of working evolved slowly into DevOps approach and was initially driven by test automatisation and development methodology changes. To

improve the system architecture, infrastructure and processes we were extending temporary the team with DevOps experts from the Capgemini shared services. For instance(see Figure 3) in the two development teams we added DevOps experts. Even though they are only available in part-time to the project, the team can define the requirements in an efficient manner and together provide the required solution faster.

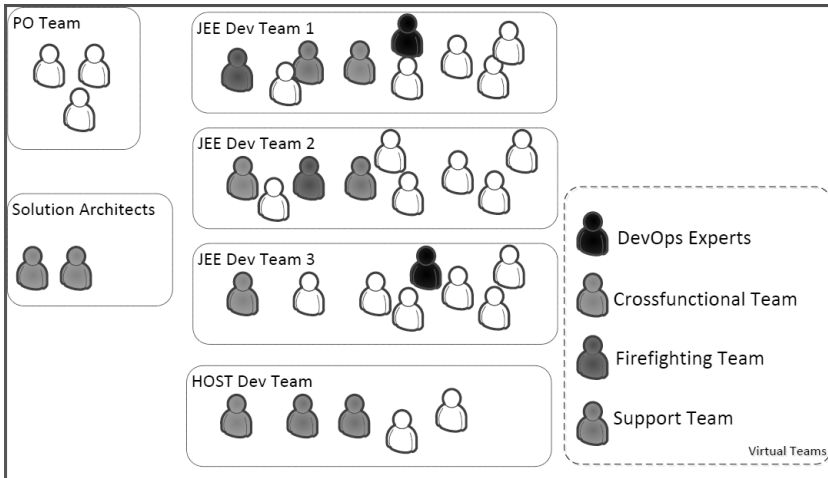


Figure 3. DevOps experts in the team structure

Integration. First step into IT automation was the continuous integration tools and the commit rules each developer has to fulfil to comply DoD. The tooling and infrastructure were installed by the external specialised team (shared service). However, the configuration was done internally by the team which in turn caused delays in the project schedule.

Automatic E2E. We introduced an automated testing soon after the first release. We decided to cover all use cases using the Selenium framework. Since then (exceptionally to joint experts service usage) testers are fully employed as team members responsible for test management and test automation. It is important to mention that test experts are preparing not all the tests. Developers and business analysts use test automation in their daily work as well.

Scrum. Even though we used several of XP practices from the early beginning of the project, the Scrum introduction was a significant change in the project constellation. First of all, we managed to break the wall between business people, development team and end users. Secondly, we introduced

continuous flow in our project and routines understandable for IT and non-IT peoples. Stepping into Scrum world required not only the process changes but also tools which let us work in a distributed environment and tools which allow efficient testing and integration.

Support and bug fixing process improvements. Introducing Scrum did not solve all the problems. We identified several bottlenecks in the project. Particularly in the support and bug fixing areas [22]. One of the improvements was the Kanban technique and the status visualisation of the tasks in the development pipeline.

Testing environment improvements. One of the obstacles we are facing now is the time required for end-2-end tests results. Thus, the ongoing improvement. We prepare the Docker [8] environment for managing test parallelization. Later on, the gathered knowledge will be used in the production environment so that we will be able to provide the continuous delivery approach to the customer.

4.5. Results

Figure 4 shows the development pipeline as well as the future improvements currently developed or considered. In general, the continuous delivery process could be divided into 4 group.

Inception. The phase is managed by Scrum process and Kanban technique in case of bugs or CRs. In reality, the process is more complex and consists of several agile ceremonies like planning, refinement, prioritisation meeting as well as Kanban meetings [22]. The output of this stage is the agreement of what has to be done, deadline and involved individuals.

Since the project start we increased the number of implemented user stories (Figure 5) and the team size (Figure 6). The project may be scaled depending on business needs without the organizational, technical or infrastructural problems.

Development. The team is cross-functional, consists of testers, developers, business analysts as well as of infrastructure specialists (part time as described in the previous sections). Despite the increasing number of developers (Figure 6), the team decided to keep the development environment (branching, configuration, and tagging) as simple as possible to avoid merging efforts. In

the past, we kept long living feature branches which in turn created significant merge debt. Now the team integrates the work several times per day on the main branch using (if needed) feature toggle approach [14]. The only branches we used are short living release branches. All the changes are continuously inspected and tested using quick unit and integration tests.

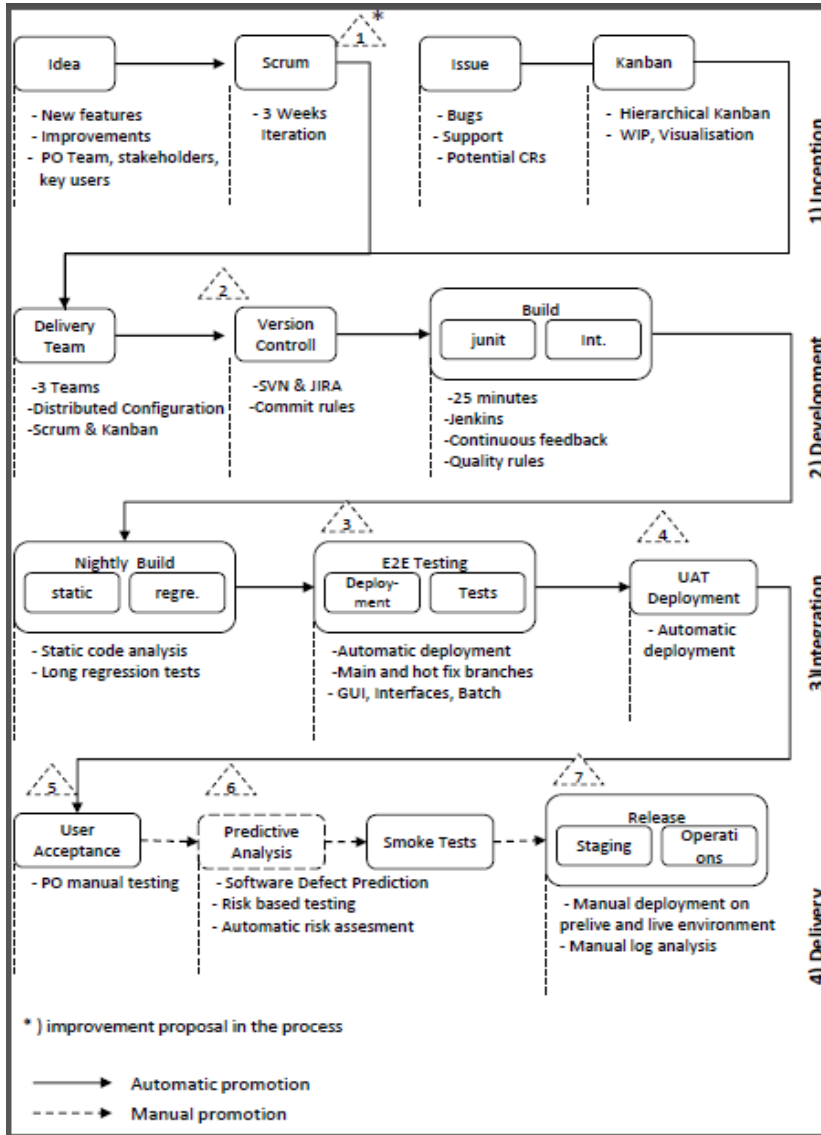


Figure 4. Development pipeline

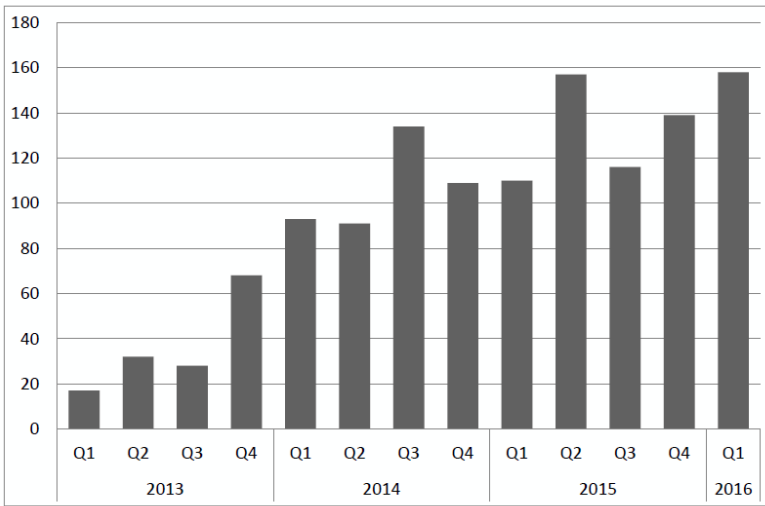


Figure 5. Number of user stories

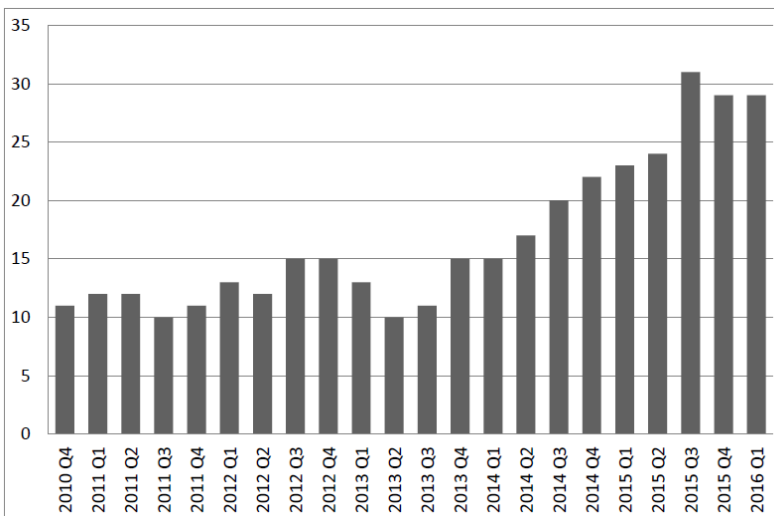


Figure 6. Number of active developers

Integration. The phase consists of two major activities:

- static code analysis and complete regression tests inclusive unit, long-running integration tests,
- automatic end-2-end tests.

Both processes are fully automated and test all business cases in the application during the night. All failures are collected and the comprehensive report (continuous feedback) is provided to the development team on an early morning. Additionally, every day we deliver the newest version of the system on UAT⁴ server automatically so that the PO-Team and key users can check the ongoing development results.

Delivery. Due to the different infrastructure provider than Capgemini, the customer did not decide to introduce the automatic deployment on the production (live) and pre-production (pre-live). The packages are automatically delivered, but the installation is manual. Before live installation, the manual tests (smoke tests) are executed to check whether the application was correctly configured during the deployment or not.

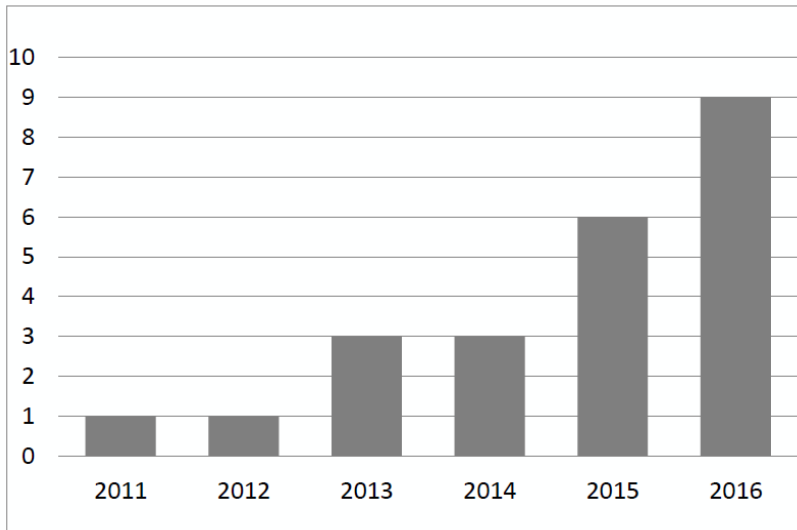


Figure 7. Number of releases per year

After introducing DevOps and continuous delivery principles, we have mainly shortened the release and test cycles so that we can deliver new software version after each sprint. We noticeably increased new installations on the production (Figure 7). We plan to provide at least 9 software versions to the live environment in 2016 (in the first quarter we delivered already 3).

⁴UAT stands for User Acceptance Testing.

If we consider the number of bugs reported in each quarter (Figure 8) we will see that since the number of releases has grown, we can predict, with the high likelihood, the number of found issues, so that we can plan the capacity of the fire-fighting team. Predicting the bug fixing efforts lets to avoid distribution of the sprint.

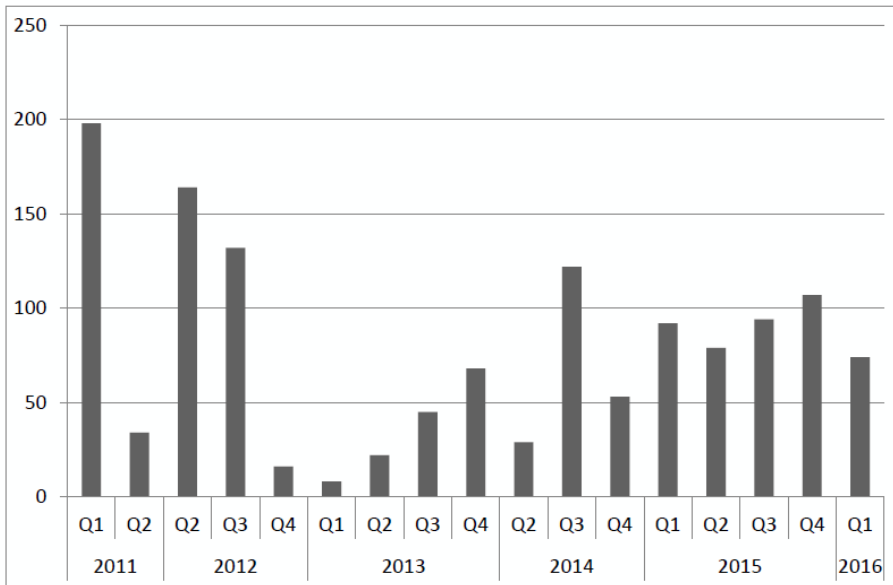


Figure 8. Number of pre-live and live bugs

4.6. Feature improvements

We changed the processes as well as the architecture of our system to support test automation and continuous integration goals. However, the transformation is still unfinished. We recognised several improvements which should or may be introduced to improve the process flow and shorten the delivery cycle. Some of the techniques like predictive analytics in software development are being recognized as leading improvement factors in the near future [4].

(1) *User Story estimation and risk assessment (predictive analytics)*. Each user story could be described with several characteristics like story point, product owner, development team, the number of tasks, business area,

technical component, key user or keywords. Base on the project history we would be able to build the analytic model which e.g. validates estimates or predict the risk of potential defects.

(2) *Version control improvement.* SVN has several drawbacks. Despite the fact we decided to keep only a few short living branches and keep the single branch for the main development, we see that using SVN is not effective and requires meaningful efforts for merging and code integration. Thus, we consider migration to Git.

(3) *Testing performance.* Despite the fact that our test environment is fully automated we see the need for faster feedback. In other words, we would like to run the tests in parallel and scale the environments on demand so that we would get results of full regression in one hour, which is crucial in case of urgent fixes. One of the options would be the use of Docker [8] concept.

(4,7) *Faster delivery of the application, pull instead of push approach.* One of the bottlenecks in the process is still the need for manual installation on the pre-live and live environment. We would like to introduce Docker-based installation and image registry. The registry stores Docker images and enables their distribution, so that the business side could decide (pull instead of push approach), whether the new version of the system should be introduced or not.

(5) *Automatic user acceptance testing.* We see that some of the user acceptance tests could be automated and provided in the test-first manner [18]. First probes showed that base on Behavioural Driven Development (BDD) [24] and Cucamber [10], we were able to deliver business-oriented testing framework, which in turn allowed writing tests by non-technical but business-oriented experts.

(6) *Software defect prediction (predictive analysis).* Not all tests could be executed manually, and not all test cases are identified because of the complexity. Thus, we propose to introduce risk-based testing approach where risk is a function of the probability of bug being found in the component or business process. The likelihood of the defect in given functionality could be identified by the application of some of the software defect prediction techniques. The first experiments conducted within the DePress [21] framework with the use of the process and product metrics, as proposed in [20], are promising.

(7) *Proactive monitoring (predictive analysis).* All the failures and events are stored in several log files. We think that some of the incidents (e.g.

memory problems) may be addressed in advance [30]. Thus, the central storage of the logs and the continues analysis is required.

5. Conclusions

The impact of DevOps on the IT world shall not be underestimated. Surely, not many of the companies that have a sizeable IT organisation are willing to automatize their deployment pipelines completely or to build DevOps teams, but our paper shows, that even moderate changes to the software development process in this direction bring measurable benefits, such as an increase in quality and flexibility. The available publications show a tendency to gain more and more market advantage to those who apply more and more DevOps principles. Having the ability to react fast to quality drops and to deliver new functionality more quickly than the competition may be a differentiating factor between staying in business or going bankrupt. There is still much to do in this field, but by doing the first steps in this direction – preparing an automated deployment pipeline, finding the new software architectures, refitting our software development processes and organising the teams accordingly, we make the push towards adopting the DevOps way on a broad front.

Acknowledgements

This work was supported by Capgemini Poland.

References

- [1] Atlassian. Atlassian Documentation. <https://confluence.atlassian.com>, accessed: May 10, 2016.
- [2] K. Beck and C. Andres. *Extreme Programming Explained: Embrace Change* (2Nd Edition). Addison-Wesley Professional, 2004.
- [3] S. Bellomo. *Architectural Implications of DevOps*. Technical report, Carnegie-Mellon University, 2014.
- [4] M. Buenen and A. Walgude. *World quality report 2015-2016, 7th edition*. Technical report, Capgemini, Sogeti and HP, 2016.

- [5] L. Bass, I. Weber, and L. Zhu. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 1st edition, 2015.
- [6] L. Chen. Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50-54, 2015.
- [7] Chef Software. All about Chef. <https://docs.chef.io/>, accessed: March 1, 2016.
- [8] Docker Inc. Docker Documents. <https://docs.docker.com/>, 2016.
- [9] Caroline Donnelly. What the enterprise can learn from online marketplace etsy's devops strategy. *Computer Weekly*, page 10, 2015.
- [10] I. Dees, M. Wynne, and A. Hellesoy. *Cucumber Recipes: Automate Anything with BDD Tools and Techniques*. Pragmatic Bookshelf, 2013.
- [11] FindBugs. <http://ndbugs.sourceforge.net/>.
- [12] J. Humble and J. Molesky. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal, Agile Product Management & Software Engineering Excellence*, 2011.
- [13] A. Hendel, W. Messner, F. Thun, A. Hendel, W. Messner, and F. Thun. *Rightshore!: Successfully Industrialize SAP Projects Offshore*. Springer-Verlag TELOS, Santa Clara, CA, USA, 1 edition, 2008.
- [14] P. Hodgson. Feature Toggles. <http://martinfowler.com/articles/feature-toggles.html>, February 2016.
- [15] J. Humble, Ch. Read, and D. North. The deployment production line. In: *Proceedings of the Conference on AGILE 2006, AGILE '06*, pp. 113–118, Washington, DC, USA, IEEE Computer Society, 2006.
- [16] M. Httermann. *DevOps for Developers*. Apress, Berkely, CA, USA, 1st edition, 2012.
- [17] C. Ladas. *Scrumban – Essays on Kanban Systems for Lean Software Development*. Modus Cooperandi Press, USA, 2009.
- [18] L. Madeyski. *Test-Driven Development: An Empirical Evaluation of Agile Practice*. Springer, Heidelberg, London, New York, 2010. <http://www.springer.com/978-3-642-04287-4>.
- [19] R. C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition, 2008.
- [20] L. Madeyski and M. Jureczko. Which process metrics can significantly improve defect prediction models? an empirical study. *Software Quality Journal*, 23(3):393-422, 2014.
- [21] L. Madeyski and M. Majchrzak. Software measurement and defect prediction with depress extensible framework. *Foundations of Computing and Decision Sciences*, 39:249-270, 12, 2014.
- [22] M. Majchrzak and L. Stilger. Experience Report: Introducing Kanban Into Automotive Software Project. In Piotr Kosiuczenko and Michał Śmiałek, eds., *From Requirements to Software: Research and Practice*, Scientific Papers of the Polish Information Processing Society Scientific Council, pp. 15–32. Polskie Towarzystwo Informatyczne, 2015.
- [23] M. Majchrzak, L. Stilger, and M. Matczak. Working with Agile in a Distributed Environment. In Lech Madeyski and Mirosław Ochodek, eds., *Software Engineering from Research and Practice Perspectives*, volume Scientific Papers of the

-
- Polish Information Processing Society Scientific Council, pp. 41–54. Polskie Towarzystwo Informatyczne, 2014.
- [24] D. North. Introducing BDD. <http://dannorth.net/introducing-bdd/>, March 2006.
 - [25] Puppet Labs and IT Revolutionary Press. 2013 State of DevOps Report. Technical report, Puppet Labs, 2013.
 - [26] Puppet Labs and IT Revolutionary Press. 2015 State of DevOps Report. Technical report, Puppet Labs, 2015.
 - [27] Puppet Labs, IT Revolutionary Press, and ThoughtWorks. 2014 State of DevOps Report. Technical report, Puppet Labs, 2014.
 - [28] PMD. <http://pmd.sourceforge.net/>.
 - [29] Selenium Project. Selenium Documentation. <http://www.seleniumhq.org/docs/>, March 2016.
 - [30] F. Salfner, M. Lenk, and M. Malek. A survey of online failure prediction methods. *ACM Comput. Surv.*, 42(3):10:1-10:42, March 2010.

Authors and affiliations

Mark ASZTALOS

*Department of Automation and Applied Informatics, Budapest University of
Technology and Economics, Hungary*

asztalos@aut.bme.hu

Maciej CHMIELARZ

Stowarzyszenie Jakości Systemów Informatycznych, Warsaw, Poland

m.chmielarz@sjsi.org

Bartosz CHRABSKI

IBM Polska, Warsaw, Poland

bchrabski@gmail.com

Wojciech CICHOWSKI

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland*

woj.cichy@gmail.com

Marta DĄBROWSKA

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland*

Volvo Group, Poland

marta.dabrowska.2@volvo.com

Tomasz DERĘGOWSKI

Axiom Corporation, Gdańsk, Poland

Tomasz.Deregowski@axiom.com

Tomasz GAWĘDA

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland*

tomasz.gaweda@outlook.com

Andrzej GROSSER

*Institute of Computer and Information Sciences, Czestochowa University of
Technology*

andrzej.grosser@icis.pcz.pl

Jarosław HRYSZKO

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland*

Volvo Group, Poland

jaroslaw.hryszko@pwr.edu.pl

Piotr JERUSZKA

*Institute of Computer and Information Sciences, Czestochowa University of
Technology, Poland*

piotr.jeruzska@icis.pcz.pl

Grzegorz KOCHAŃSKI

Smart4Aviation, Gdansk Science & Technology Park, Poland

grzegorz.kochanski@smart4aviation.aero

Piotr KONOPKA

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland*

Volvo Group, Poland

piotr.konopka@volvo.com

Miłosz KURZAWSKI

*Faculty of Management and Economics, Gdańsk University of Technology,
Gdansk, Poland*

mkurzawski@zie.pg.gda.pl

Piotr MAĆKOWIAK

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland,
piotr.mackowiakk@gmail.com*

Lech MADEYSKI

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
Lech.Madeyski@pwr.edu.pl*

Marek MAJCHRZAK

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland (Chapter 9),
marek.majchrzak@pwr.edu.pl
Capgemini Poland (Chapter 16),
marek.majchrzak@capgemini.com*

Michał MALINKA

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
mmmalinkaaa1@gmail.com*

Krzysztof MIŚTAŁ

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
203298@student.pwr.edu.pl*

Adrian NAJCZUK

*Wrocław University of Science and Technology, Wrocław, Poland
najczuk@gmail.com*

Ewa NESTOROWICZ

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
ewa.nestorowicz@gmail.com*

Ziemowit NOWAK

Faculty of Computer Science and Management, Wrocław University of Science and Technology, Wrocław, Poland
ziemowit.nowak@pwr.edu.pl

Cezary ORŁOWSKI

WSB University, Department of Information Technology Management, Gdansk, Poland
corlowski@wsb.gda.pl

Agnieszka PATALAS

Faculty of Computer Science and Management, Wrocław University of Science and Technology, Wrocław, Poland
a.m.patalas@gmail.com

Aneta PONISZEWSKA-MARAŃDA

Institute of Information Technology, Lodz University of Technology, Poland
aneta.poniszewska-maranda@p.lodz.pl

Adam ROMAN

Faculty of Mathematics and Computer Science, Jagiellonian University, Poland
Stowarzyszenie Jakości Systemów Informatycznych, Warsaw, Poland
roman@ii.uj.edu.pl

Ferenc Attila SOMOGYI

Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Hungary
sf1026@hszk.bme.hu

Lucjan STAPP

Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
Stowarzyszenie Jakości Systemów Informatycznych, Warsaw, Poland
l.stapp@mini.pw.edu.pl

Wojciech STĘPNIAK

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
wojtek@niom.pl*

Tomasz WALA

*Capgemini Poland
tomasz.wala@capgemini.com*

Artur WILCZEK

*Faculty of Computer Science and Management, Wrocław University of Sci-
ence and Technology, Wrocław, Poland
Artur.Wilczek@pwr.edu.pl*

Oskar WOŁK

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Wrocław, Poland
oskar.wolk@gmail.com*

Patryk WÓJCIK

*Institute of Information Technology, Lodz University of Technology, Poland
patryk.wojcik@outlook.com*

Jolanta WRZUSZCZAK-NOGA

*Faculty of Computer Science and Management, Wrocław University of
Science and Technology, Poland
jolanta.wrzuszczak-noga@pwr.edu.pl*

Artur ZIÓLKOWSKI

*WSB University, Department of Information Technology Management,
Gdansk, Poland
aziolkowski@wsb.gda.pl*

Wiktor ZYCHLA

University of Wrocław, Poland

VULCAN sp. z o.o., Wrocław, Poland
wzychla@ii.uni.wroc.pl



Scientific Council
of the Polish Information Processing Society
Solec St. 38/103
00-394 Warsaw, Poland
ISBN 978-83-943248-2-7