DOI: 10.15439/2018F93
ISSN 2300-5963 ACSIS, Vol. 17

Unintended effects of dependencies in source code on the flexibility of IT in organizations

Debbie Tarenskeen
HAN University of Applied Sciences, Arnhem, The Netherlands
Email: debbie.tarenskeen@han.nl

Rogier van de Wetering
Open University, Heerlen, The Netherlands
Email: Rogier.vandeWetering@ou.nl

René Bakker
HAN University of Applied Sciences, Arnhem, The Netherlands
Email: Rene.Bakker@han.nl

Abstract—**This study links business requirements and adaptability of existing software systems. Organizations expect flexibility of IT with regard to business requirements. We hypothesize that the flexibility of business requirements is difficult in IT systems, because of software dependencies in the way domain knowledge is implemented. In this paper, we, therefore, explore how Business requirements have been implemented in the source code of three open source healthcare systems. Outcomes suggest that a tight interdependency of business terminology and functionality in source code hides business requirements from view and thereby hinders IT flexibility on higher levels.**

I. Introduction

SCHOLARS investigate strategic alignment of business and information technology (IT) for more than three decades within the information systems (IS) community. Recently, the importance of the role of flexible IT infrastructures for strategic alignment has been demonstrated anew for deployment, innovation, and evolution of IT systems in firms that operate in turbulent industries, including healthcare [1-4]. In the software architecture domain, software adaptability is seen as a quality attribute of software in general. Thereby meaning, for instance, the applicability of technological innovations or new technical features. Software adaptability is not explicitly aimed at changes in the business domain [5-7]. Expectations of the business do value general adaptability of systems, but also assume adaptability regarding business requirements. This current study focuses on changes in the business domain and business requirements and its consequences for software. We examine adaptability of IT systems regarding business terminology and business requirements.

II. IT Flexibility in enterprise architecture

Although Enterprise architecture methods such as TOGAF focus on high-level business requirements, in the Business architecture, they are meant to show relations between high-level requirements and supporting architectures. We use definitions of TOGAF, because TOGAF aims at describing all levels of the IT infrastructure. TOGAF describes the supporting IT as data architecture, application architecture, and technology architecture. The definition of architecture in this paper follows TOGAF's: ''The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.'' Based on ISO/IEC 42010 according to TOGAF.

III. Related research on software evolution

Adaptability of software in empirical research can be positioned in the domain of research of Evolution of software. Within this domain, we notice that the evolution of business requirements is only marginally addressed.

Lehman strongly influences the research field of software evolution. The Laws of software evolution have been stated and evaluated during more than a decade of research [9, 10]. Research in this field has made no explicit distinction between the evolution of systems based on requirements in general, and evolution of systems based on new business requirements.

Numerous studies have emphasized the complexity of source code changes after the initial system has been realized, for example, see [11-13]. Studies that examine the relation of source code to IS architectures have a different focus than this research. They, e.g., aim at developing frameworks for software architecture evolution knowledge [14], or frameworks for classifying architecture-centric software evolution research [15], or on automatically updating architecture documents based on software changes [16].

IV. Axiomatic design and conceptual independence

To present our point of view, we start by explaining the theoretical basis for adaptable and flexible low-level software components in an IT architecture. The theoretical views focus specifically on the adaptability of business requirements instead of on adaptability of software in general. The theoretical principles of Conceptual independence (CI), and the independence of functional requirements such as described in Axiomatic design (AD) [17-19] will be researched in this study in real-life software.

We report on a code mining study of open source code for Healthcare organizations for electronic health record systems

(EHR), to examine the way business terminology is applied. Then we will argue that there is a direct link of adaptability on the source code level to IT flexibility as expected by the Business architects.

V. Research questions

We explore three selected open source software systems to find out if a separation of business terminology and application code has been effectuated, to create flexibility in the source code (CI). Hence, our first question concerns description of the software systems by the developers:

RQ1: Are indications of CI found in the documentation?

Then, we question the interdependency of the data model and the application source code. We implicitly assume that the data in the database model will represent the data that will be persistently stored.

RQ2: Does code demonstrate interdependency of table names and source code?

Next, we want to examine the flexibility of the specific healthcare terminology in the software system, based on CI. Thus, we define:

RQ3: Is CI applied in the software application?

Next, for AD functional requirements are primary. So we define:

RQ4: Does the source code of the system show different components that are related to separate Functional requirements?

The remainder of this paper is structured as follows. First, we highlight the theoretical aspects relevant to this study. Next, we present our methods section which is followed by the results section. We then discuss our findings and end with concluding remarks and some suggestions for future research.

VI. Theoretical background

A. Axiomatic design

The principles of AD are explained by Suh [19]. He explains how a design method should account for the independence of functional requirements and a low information density in different design parameters of the system. He calls these characteristics the Independence axiom and the Information axiom. The systems he describes are industrial systems, but he emphasizes that these principles can be applied to IT [20]. We interpret design parameters as design components of an IT system. The objective of AD is to realize systems that are flexible and understandable. With AD, the designs are iteratively developed and have the domain of customer needs, demands and requirements as a point of departure. From customer needs, functional requirements are inferred. In the design, the functional requirements are formulated independently from each other and can be changed in the design without affecting the rest of the design. The principle of Independence of functional requirements is at the fundament of AD and can be compared to patterns in software engineering [21], such as separation of concerns. However, realizing independent functional requirements is not a priority in software engineering [22].

B. Conceptual independence

We advocate CI, the decoupling of healthcare terminology and domain models from software code to be able to alter healthcare terminology or domain models flexibly. We ground this choice on previous case studies that argue that AD principles are hard to implement in software systems, because of the interdependence of data models and the behavior of the system [23]. The interdependency of data models and application code has been extensively studied in IS research [24-27].

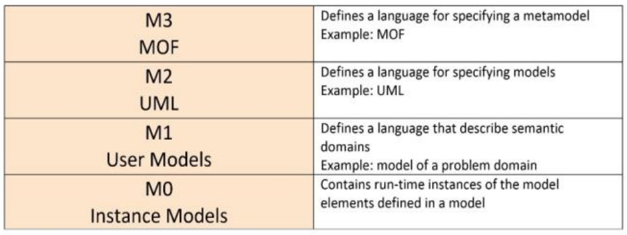
McGinnes points to the interdependence of data models and software application code as Conceptual dependence. He advises the decoupling of the data model and application functionality, meaning the behavior of the application [18]. McGinnes defines the Conceptual model as the structure of the information that is used in the business. Comparing the Conceptual model to the Domain model in UML, we find that in UML often behavior is added to the classes in the Domain, this is not the case in McGinnes' Conceptual model.

McGinnes adds behavior to Concepts by ordering Concepts in Archetypical categories, that applications can access. The applications have a responsibility to interpret the Archetypical categories. The applications add the behavior based on the specific Archetypical category. For instance, for "Location" the application knows that the instances of this category can be presented on a map.

C. Relation of Conceptual model to model-driven development

McGinnes describes the conceptual model as a (business) data structure that is used by the application. The meta-model, of the conceptual model, is fixed, the content is variable. These structures are comparable to MDD described by the OMG, Object management Group [28]. There are four levels of models, each function as a meta-level of the lower level. These levels are M0 to M3. McGinnes positions the conceptual model itself on level M1 as data.

We will address the meaning of these levels briefly.



M3 MOF	Defines a language for specifying a metamodel Example: MOF
M2 UML	Defines a language for specifying models Example: UML
M1 User Models	Defines a language that describe semantic domains Example: model of a problem domain
M0 Instance Models	Contains run-time instances of the model elements defined in a model

Fig. 1 Diagram of Modeling levels of OMG

The M1 level is the most important and most discussed modeling level in practice of software engineering. It shows categories or classes and the associations between them. The content consists of terms, for example, Person, Product, Order. The level is comparable with table names in RDBMSs. The model M0, on the lowest level, contains the instances of categories M1 stored. It is comparable to records in RDBMSs or instances of objects in programming languages. See Figure 1, published in a whitepaper explaining the different levels of Modeling of OMG [29].

The M2 level contains the description of model elements in a modeling approach; this is the meta-level of the description of, e.g., UML-models. The highest level (M3) contains a description of all (possible) modeling approaches. It is intended for comparing different modeling approaches [29].

The description of McGinnes of the model is at the M2 level, the model that concerns business concept types is an M1 level-model [30]. In this paper, we will not explore the similarities of OMG and McGinnes further. We think the challenge in system development lies in separating Conceptual models from behavior.

We argue based on ideas behind Conceptual dependence that CI is a prerequisite to being able to separate the different functional requirements from each other in the behavior part of the application [23].

D. Ampersand as illustration

We first, will describe a prototype system called Ampersand[2], based on a requirements specification language with relational semantics to illustrate the feasibility of implementing principles in source code [31]. Ampersand relies on model-driven development (MDD) to generate systems entirely defined by its domain model and business rules.

E. Ampersand applies CI and relation algebra for AD

We will explain the workings of Ampersand to demonstrate that flexibility of business functional requirements is feasible on source code level. This example is added for technical readers to explain the low-level code involved in separating business terms from application code. It also demonstrates with low-level code that a possibility to separate the business requirements from each other can be accomplished. The system Ampersand has separated the conceptual model from behavior. It, therefore, conforms to CI. It is based on relation algebra and has a mathematical structure [31, 32]. The conceptual model in Ampersand consists of concepts and relations between concepts. All information about concepts and relations is described in an Ampersand script (typically a .txt file). There is no extra information of the business hidden in the software system. Behavior is described and defined in invariant (or declarative) business rules. The behavior is only applicable to the concepts and relations in the script. A script contains one Context that is entirely separate from other Contexts that can be defined in Ampersand. Ampersand applies rules as a way to connect the conceptual model structure to behavior. Rules can also be defined to check the consistency of data. Ampersand applies Rule checking behavior to define the software behavior. Rule checking is applied to Concepts and Relations in the Ampersand model. Examples from rules in healthcare can be: Diagnoses must have a relation to a Medical doctor, Diagnoses must have a date, a Patient cannot receive medication without the consent of the MD.

Each rule must be independent of the other rules in Ampersand, and therefore, behavior can be defined according to independent business functions.

F. Ampersand Runtime

The Ampersand Runtime can read, parse the script and import the Conceptual model, data, and rules. The script contains models on level M1 and M0. After reading this, a Rule engine checks business rules and signals violations. It operates on any script that conforms to the syntax and constraints of the Ampersand approach (On level M2).

G. Example Ampersand script

The following description of the Ampersand script is the model in natural language on level M2 of the OMG. Here we describe constraints and model elements (categories) that can be present in the script.

The first term in the Ampersand script is the word: CONTEXT. It signals the beginning of the script. ENDCONTEXT signals the ending. Then a PATTERN is presented consisting of CONCEPTS and RELATIONS.

After the pattern, the word ENDPATTERN closes this part, and in the script, PROCESSs can be defined regarding Ampersand RULES.

Summarizing, we can state that Ampersand follows the principles of CI by providing flexibility for the structure and naming of the data model. There are two different methods for keeping the conceptual model separate from the application code in Ampersand. First, the Ampersand Runtime works directly with the script and does not know about the domain in the script. Second, the script can be used for MDD. The Ampersand system conforms to the Independence axiom of AD, at least as far as functional requirements are concerned that can be defined in rules.

We have explained the workings of Ampersand in detail to demonstrate that flexibility of business functional requirements is feasible on source code level.

[2] Named after the ampersand symbol (&). According to Michels et al. the name refers to getting the best from both business and IT, i.e., achieving results from theory and practice alike, and realizing the desired results effectively and more efficiently than ever before.

VII. Research Method

A. Data collection procedure

We report the outcomes of code analysis of three systems. Two of these systems are frequently used in international health practice. The third system implements the standard of openEHR; a development we see more often these days. The latter claims to support different kinds of models for medical data. The research data are downloaded systems from GitHub. These were run locally to assess runtime dependencies and check if the source code is complete. Then, we have analyzed the documentation and the source code. We classified source code in types, the source code for libraries, the source code for initializing the database, source code for user interface frameworks and source code for business and other functionality in the software system. Only the last type of files have been examined in RQ2.

Since the idea of the paper is to evaluate open source systems in healthcare for application of CI and AD, we have searched for open source systems with an active community. The systems have been included in the references (websites and date) [33-36]. All of these are web applications that were run by us with an apache or tomcat server. Cabolabs is written in grails, openEMR in PHP and openMRS in java. All could operate with a MySQL database. The cabolabs openEHR download consists of 1351 files with 48 different extensions. The openEMR consisted of 12118 files with 130 extensions. The openMRS software has two downloads, the standalone consists of 723 files with 56 extensions. Because we also wanted to analyze the java code, we have also downloaded the core of openMRS with 1623 files with 40 different extensions.

B. A multistep approach

For each system, we applied a multistep approach including the following action: (1) running the systems locally, (2) analyzing all relevant and available documentation, (3) analyzing the directory structure, (4) extracting the data model from the MySQL database, (5) analyzing specific healthcare terms (see chapter VIII), (6) analyzing if database tables are hardcoded or generated for the particular system and (7) selecting of source code with (business) functionality (manually).

We incorporate specific methodological considerations and action (per research question) into the results sections.

VIII. Separate Analysis of healthcare terms

We wanted to assess if application code depended upon hardcoded names in software code derived from healthcare. In the source code distinguishing between names that refer to healthcare terms and names that are necessary to follow the technical program flow, is difficult. Since we do not have expertise on healthcare terminology, but we wanted to signal the terms that were healthcare terms, we have asked independent reviewers and one healthcare professional (psychiatrist) to review the names of database tables and list the names derived from or partly related to healthcare.

We have asked four reviewers that are not researchers or in any way related to the case studies. We have asked them to evaluate every table name in the three databases.

Table I.
Evaluation of healthcare related terms in table

		openEHR	openEMR	openMRS
Number of tables		59	212	148
Number of tables with Health care related names	According to 3 of 4 reviewers	1	30	16
	According to Health care professional	2	62	50
Proportion of tables with Health care related names	According to 3 of 4 reviewers	0,02	0,14	0,11
	According to Health care professional	0,03	0,29	0,34

The scores of the healthcare professional have been registered separately also. For every open source program, there were two scores: the percentage of table names that three of the four reviewers labeled as healthcare related. Moreover, the separate score of the healthcare professional.

In Table I we find the number and percentage of table names with recognizable healthcare terminology parts. The healthcare professional classified more names as healthcare related than the other persons, but all the table names that the 3 out of 4 persons listed were a subset of the names that the healthcare professional listed. We are now able to assess interconnectedness of application code to hardcoded healthcare terms.

IX. Results Conceptual independence and Axiomatic design in source code

A. Conceptual independence in the documentation

This section addresses RQ1. We have extensively read the associated documentation and searched for indications that the system is adaptable based on healthcare terminology. Through our analyses, and also based on our review of the Information model of openEHR (on http://www.openehr.org/), we conclude that openEHR indicates CI. A quote from documentation of openEHR confirms this view [37]: "*Your EHR system does not need to know a priori about any of the clinical data it will process, such as vital signs, diagnoses or orders. Models for those things are developed separately. Models for data sets and forms are also developed separately, and UI form components are generated from these definitions.*"

The data structure is said to be very flexible and can support transformations to other healthcare terminology standards.

In openEMR, there exists no reference to a model, but we find a description of the Database structure [38]. There is some variability for the conceptual model, by which we mean, the user can define categories in one table. Thus, openEMR is partly flexible concerning terminology, but not concerning models of healthcare data. Finally, openMRS shows signs of CI and AD. To highlight this particular view, we quote from the wiki documentation of openMRS [39]: *"At the heart of OpenMRS is a concept dictionary. This dictionary, much like a typical dictionary, defines all of the unique concepts (both questions and answers) used throughout the system."*

The software itself, openMRS, in essence, is constructed to support 'modules.' Implementations can modify the behavior of the system to meet local requirements using these modules. Because changes can be added to the Conceptual model, it is not necessary for everyone having to agree on a single approach.

B. *Interdependency of table names and source code*

We now report the results of for RQ2. These results consist of totals of code mining results.

We have defined two indications of the interdependency of code and data, i.e., I) hardcoded use of table names in the source code of more than 80 percent of the tables and II) hardcoded use of table names in the source code of more than 80 percent of the source code.
We did not find the expected first indication in every system. We would have expected the use of all the tables in the source code. If table names are missing, they are not used. Since the named applications are the only applications that use the database tables, this needs further investigation.

The second indication has been signaled in the source code files. If names of database tables in source files are hardcoded, then any particular change in table names implies changes in the source code. With table names spread over different files, then changes in table names lead to changes in multiple maybe interdependent files, leading to unpredictable behavior of the software application.

Why would a table name change? If ideas about the Conceptual model change or if other functionality needs other data concepts or maybe extra attributes (columns in tables) then the table names and columns will change. Adaptations of concepts and table names or columns are frequent in the evolution of code [25]. In this empirical study of Qiu, it was found that adding tables, columns and changing names of tables and columns frequently appear. We focus here, specifically, on the names that are healthcare related, because we signal a relation between business terms and source code files.

Our method can be described in the following way: we have extracted all table names from the applied database management system and have counted the number of times these names are hardcoded in software source code. We have calculated the percentage of database table names that were found in the source code files. We have counted the number of source code files that access database table names directly, or use Class names that are derived from table names, for instance by removing the dash. In Figure 2 the relations of the source code files to table names are visually presented. We also have calculated the percentage of source code files, that access table names directly.

Counting will demonstrate the relation.

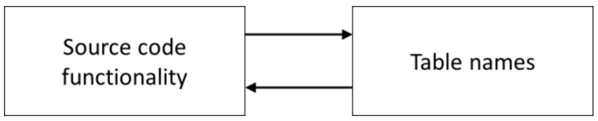


Fig. 2 The existence of relations between source code and table names

Concerning openEHR, 46% of table names have been found in the programming code, but only two of those are marked as Healthcare related. The names are doctor-proxy and patient-proxy, but no table names are related to medical knowledge. The groovy files with table names accounted for 99% of 176 files, but these were not marked as Healthcare related, exception above. In groovy files, 69% class names have been found, that are derived from table names. Groovy files with these class names accounted for 69% of the groovy files.

In the openEMR download, 82% of table names are found hardcoded in PHP-code. Including 29 of 30 with Healthcare-related table names. In the PHP-files 94% of 5401 files have access to hard-coded table names.

In the openMRS-core download, we have found 57% of table names are hardcoded in java-source code. Including almost all table names (14 of the 16), that have been marked Healthcare related. In the Java-files in the openMRS-core, 100% of 1019 files access hardcoded table names.

Based on the second indication, we find an extensive interdependency between source code and database table names in all three systems.

X. Is Conceptual independence applied in the software application?

This paragraph reports results for RQ3. The indications below are derived from characteristics of CI:

- *Indication: No hardcoded use of healthcare-related table names in the source code.*
- *Indication: A presence of a separate structured model for healthcare terms, in the source code for generating database tables.*
- *Indication: A presence of a separate structured model for healthcare terms, in a separate file.*

We expect that when the healthcare-related table names are found in source code files, then changes in healthcare terminology directly affect the source code.

So for the first indication of RQ3, we have to mark database table names that have a direct reference to the medical terminology in the system. For this indication, we had first to classify all table names in "Unknown name" and "Healthcare terminology name." See Chapter VIII. Then we searched for occurrences of healthcare-related table names in source code files. Two systems: openEMR and openMRS, applied hardcoded healthcare-related table names in the source code.

The other two indications, above, are meant to demonstrate a separation of the conceptual model and the application code, as is a characteristic of CI. In detail, we have found that openEMR and openMRS have applied frameworks for separating business domain terms (the Conceptual model) and business logic from the rest of the application code. The frameworks used are Zend for openEMR and Hibernate for java in openMRS. These frameworks and the related source code of the systems have been analyzed. The frameworks use script code for defining the (Business) Conceptual model. They do not separate the Conceptual model from the behavior of the software. Therefore these do not comply with CI. The frameworks aid the developers with building and partly generating source code. Hibernate helps developers in separating database management systems from source code but does not aid in decoupling business terminology from source code. The framework script code then becomes part of the source code. We cannot directly extract the applied Conceptual models.

The software of openEHR contains separable Conceptual models apart from application logic. We confirm the existence of separate Conceptual models because we also find "parsers" and "indexers" in the source code.

For the last indication, we have counted the number of times table names can be found in one file, to search for indications of a definition file for the Conceptual model. In openEHR, we found the "opt-file" and "adl-file,", in which the M1 model is included as data. They comply with the M2 model of openEHR. Therefore it can be used for separating the Conceptual model from the behavior of the software.

In the openMRS source, the liquibase tool is applied for updating tables based on changes in the database for new modules. With the liquibase functionality updates on database structure and data can be automated with liquibase.xml-files. The M1 model is input as xml-data, but no M2 model can be found.

In openEMR, only an .sql file was found that contained 188 of the 212 tables. The healthcare related terms are not included as data but are hardcoded in sql. It cannot be used as an M1 model, because changing it will break the source code and no M2 model can be found.

Concluding: In the source code of Cabolabs openEHR server system all three indications have been found. Several files with the Conceptual model and its instances (M1 and M0) have been signaled. These files with extensions .adl and .opt can be reused by other openEHR standard based systems. Cabolabs openEHR-server applies CI.

In the other systems frameworks such as Hibernate and Zend have been used, for partly separating the model from the application code. Further, the frameworks do not distinguish business terms from application code classes. The consequence is that the current application of frameworks involves code programmers for adaptation of business logic and business terminology.

XI. Axiomatic design applied in source code

In this paragraph, a report of RQ4 is given. For AD, functional requirements are primary. In AD it is required that the software system can be divided into components that are related to functional requirements. Systems based on AD will be adaptable based on changing functional requirements because business IT architects can pinpoint specific source files where changes are necessary.

RQ4 will lead to demarcation lines in the Runtime components or demarcation lines in the source code, which has different independent functional requirements.

- *Indication: Existence of directory structures in the source code that show Functional requirements*
- *Indication: Existence of runtime modules that can be added and deleted for Functional requirements behavior that is executed*

The indications for Axiomatic design will be studied in detail in future research. In this overall check of the source code, it is found that openMRS contains a directory structure for separate modules. We find complementary functionality in the openMRS runtime application because modules with Business functionality can be turned off and on. With the openEHR server software, tooling is under construction that can generate User interfaces based on the opt-files. Moreover, thus separation of high-level functional requirements can be realized.

XII. Conclusion and discussions

In this study, we have explored the adaptability of source code concerning the business requirements and changes in the business domain terminology. Interdependency of the data model and the application code can make systems hard to change, this is seen in the literature and in our investigation of open source healthcare systems.

However, the dependency of the application source code on healthcare terms can be avoided by separating these terms in a separate model as input for the application. We demonstrate this with the Ampersand prototype, where indications for CI and AD can be located in the source code.

We have explored how business terms and business functionality appear in application source code in three open source systems actively used in healthcare. We have shown that CI, separating the business terms from the application software, can be applied and is applied in openEHR and partly in openMRS. AD has not been studied extensively in this case, but indications for AD are found in openMRS.

We conclude that because of the extensive interdependence of the data model and application source code in openEMR and openMRS, business terminology becomes part of the source code and cannot be adapted without radically changing the source code. So we conclude that in these systems the flexibility of business terminology is obstructed if the business terminology is not explicitly separated from the application source code.

Despite this studies contributions, there are several limitations that future research should address. The researchers remark that an alternative to separating the conceptual model from application source code would be to use tooling for source code editing based on business requirements. Moreover, currently, some indications for this kind of tools were present in the source code that is examined. Frameworks help to separate the Conceptual model from the application code, but in the end, Conceptual models become an integrated part of the source code. When the Conceptual model is included in the source code, then it will depend on professional skills or discipline of the programmer(s), to check that the Conceptual model will stay separated from application code. Since frameworks do not distinguish health care terms from software application classes, medical expertise is necessary to locate these.

In this paper, we have only studied software architecture for a limited number of applications and components. Therefore it can be questioned if a full-scale application of this principle can be implemented in enterprise architectures. We are currently researching the design and implementation of a separate (conceptual model-layer) data layer in a large scale healthcare IT architecture.

REFERENCES

[1] J. C. Henderson and H. Venkatraman "Strategic alignment: Leveraging information technology for transforming organizations," Ibm Systems Journal, 1, (1993), 32, pp. 472-484, doi: 10.1147/sj.382.0472

[2] R. Van de Wetering, P. Mikalef and A. Pateli "A strategic alignment model for IT flexibility and dynamic capabilities: toward an assessment tool," Proceedings of the 25th European Conference on Information Systems (ECIS), Guimarães, Portugal(2017), pp. 1468-1485, doi:

[3] R. van de Wetering, P. Mikalef and R. Helms "Driving organizational sustainability-oriented innovation capabilities: a complex adaptive systems perspective," Current Opinion in Environmental Sustainability(2017), 28, pp. 71-79, doi: http://dx.doi.org/10.1016/j.cosust.2017.08.006.

[4] R. van de Wetering, J. Versendaal and P. Walraven "Examining the relationship between a hospital's IT infrastructure capability and digital capabilities: a resource-based perspective, doi:

[5] L. Bass, P. Clements and R. Kazman Software architecture in practice. Addison-Wesley Professional, Upper Saddle River, NJ, 2012.

[6] F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord and J. Stafford Documenting Software Architectures: Views and Beyond. Addison-Wesley Professional, 2011.

[7] J. Tyree and A. Akerman "Architecture decisions: Demystifying architecture," Ieee Software, 2, (2005), 22, pp. 19-+, doi: 10.1109/ms.2005.27.

[8] TheOpenGroup TOGAF Version 9.1 Evaluation copy. The Open Group, 2011.

[9] M. M. Lehman "Laws of software evolution revisited," European Workshop on Software Process Technology(1996), pp. 108-124, doi: https://doi.org/10.1007/BFb0017737.

[10] I. Herraiz, D. Rodriguez, G. Robles and J. M. Gonzalez-Barahona "The evolution of the laws of software evolution: A discussion based on a systematic literature review," ACM Computing Surveys (CSUR), 2, (2013), 46, pp. 28, doi: 10.1145/2543581.2543595.

[11] N. Ajienka, A. Capiluppi and S. Counsell. "Managing Hidden Dependencies in OO Software: a Study Based on Open Source Projects." In Proceedings of the Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on, IEEE, 2017, pp. 141-150, doi: 10.1109/ESEM.2017.21.

[12] H. Kagdi, M. L. Collard and J. I. Maletic "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," Journal of Software: Evolution and Process, 2, (2007), 19, pp. 77-131, doi: 10.1002/smr.344.

[13] H. Kagdi and D. Poshyvanyk "Who can help me with this change request?," Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on(2009), pp. 273-277, doi: 10.1109/ICPC.2009.5090056.

[14] A. Ahmad, P. Jamshidi and C. Pahl "A framework for acquisition and application of software architecture evolution knowledge: 14," ACM SIGSOFT Software Engineering Notes, 5, (2013), 38, pp. 1-7, doi: 10.1145/2507288.2507301.

[15] P. Jamshidi, M. Ghafari, A. Ahmad and C. Pahl. "A framework for classifying and comparing architecture-centric software evolution research." In Proceedings of the Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, IEEE, 2013, pp. 305-314, doi:

[16] T. Haitzer, E. Navarro and U. Zdun "Reconciling software architecture and source code in support of software evolution," J Syst Softw(2017), 123, pp. 119-144, doi: https://doi.org/10.1016/j.jss.2016.10.012.

[17] S. McGinnes. "The Problem of Conceptual Incompatibility." In Proceedings of the International Conference on Availability, Reliability, and Security, Springer, 2011, pp. 69-81, doi:

[18] S. McGinnes and E. Kapros "Conceptual independence: A design principle for the construction of adaptive information systems," Information Systems(2015), 47, pp. 33-50, doi: https://doi.org/10.1016/j.is.2014.06.001.

[19] N. P. Suh Axiomatic Design: Advances and Applications (The Oxford Series on Advanced Manufacturing). Oxford University Press, New York Oxford, 2001.

[20] N. P. Suh "Fundamentals of Design and Deployment of Large Complex Systems: OLEV, MH, and Mixalloy," Journal of Integrated Design & Process Science, 3, (2012), 16, pp. 7-28, doi: 10.3233/jid-2012-0001.

[21] C. Larman Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development. Prentice Hall PTR Upper Saddle River, N.J., 2005.

[22] F. Buschmann, K. Henney and D. Schmidt Pattern-oriented Software Architecture: on patterns and pattern language. John wiley & sons, 2007.

[23] D. Tarenskeen and R. Bakker. "Applying Axiomatic design and Conceptual independence in the domain of IT systems." In Proceedings of the ICAD 2017 International Conference on Axiomatic Design, Iasi Romania, 2017, doi: https://doi.org/10.1051/matecconf/201712701006.

[24] A. Aryani, F. Perin, M. Lungu, A. N. Mahmood and O. Nierstrasz. "Can we predict dependencies using domain information?." In Proceedings of the Reverse Engineering (WCRE), 2011 18th Working

Conference on, IEEE, 2011, pp. 55-64, doi: http://doi.ieeecomputersociety.org/10.1109/WCRE.2011.17.

[25] D. Qiu, B. Li and Z. Su. "An empirical analysis of the co-evolution of schema and code in database applications." In Proceedings of the Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 125-135, doi: 10.1145/2491411.2491431.

[26] A. Cleve, M. Gobert, L. Meurice, J. Maes and J. Weber "Understanding database schema evolution: A case study," Sci. Comput. Program.(2015), 97, pp. 113-121, doi: https://doi.org/10.1016/j.scico.2013.11.025.

[27] T. Mens, L. Meurice, M. Goeminne, C. Nagy, A. Decan and A. Cleve "Analyzing the Evolution of Database Usage in Data-Intensive Software Systems, October 14, 2017, (2017), 2017, doi: [28]

[28] I. Object Management Group Meta Object FacilityTM (MOFTM) Core 2.5.1. 2016. http://www.omg.org/spec/MOF/. Retrieved October 9, 2017, Accessed in 2017.

[29] I. Object Management Group Meta-Modeling and the OMG Meta Object Facility (MOF) . 2017. www.omg.org/ocup-2/documents/Meta-ModelingAndtheMOF.pdf. Retrieved October 9, 2017, Accessed in 2017.

[30] J. Bézivin and O. Gerbé. "Towards a precise definition of the OMG/MDA framework." In Proceedings of the Automated Software Engineering. (ASE 2001). 16th Annual International IEEE, 2001, pp. 273-280, doi: 10.1109/ASE.2001.989813.

[31] G. Michels, S. Joosten, J. van der Woude and S. Joosten. "Ampersand." In Proceedings of the International Conference on Relational and Algebraic Methods in Computer Science, Springer Verlag, 2011, pp. 280-293, doi: DOI https://doi.org/10.1007/978-3-642-21070-9_21.

[32] G. Michels, S. Joosten, J. v. d. Woude and S. Joosten. "Ampersand applying relation algebra in practice." In Proceedings of the Proceedings of the 12th international conference on Relational and algebraic methods in computer science, Rotterdam, The Netherlands, Springer-Verlag, 2011, pp. 280-293

[33] P. Pazos openEHR cabolabs server-v0.9. 2017. https://github.com/ppazos/cabolabs-ehrserver. Retrieved 03/05/2017, Accessed in 2017.

[34] openEMR openEMR-v5.0.0. 2017. https://github.com/openmrs/openmrs-standalone. Retrieved 03/03/2017, Accessed in 2017.

[35] openMRS openMRS Core-v4.0.0. 2017. https://github.com/openmrs/openmrs-core. Retrieved 03/04/2017, Accessed in 2017.

[36] openMRS openMRS Standalone 2.5. 2017. https://github.com/openmrs/openmrs-standalone. Retrieved 03/03/2017, Accessed in 2017.

[37] What is openEHR? 2017. http://www.openehr.org/what_is_openehr#. Retrieved October 14, 2017, Accessed in 2017.

[38] openEMR Database structure openEMR. 2014. http://www.openemr.org/wiki/index.php/Database_Structure. Retrieved October 14, Accessed in 2017.

[39] B. Mamlin and S. Jindal Introduction to OpenMRS. 2017. https://wiki.openmrs.org/display/docs/Introduction+to+OpenMRS. Retrieved October 14, 2017, Accessed in 2017.