A Taxonomy for Software Modernization Tools

- An Architecture-Driven Modernization perspective -

Bruno Marinho Santos[1], Ignacio García Rodríguez de Guzmán[2], Mario Gerardo Piattini Velthuis[2], Valter Vieira de Camargo[1].

–

Bruno.Santos@ufscar.br, Ignacio.GRodriguez@uclm.es, Mario.Piattini@uclm.es, ValterVCamargo@ufscar.br

–

[1]Departamento de Computação, Universidade Federal de São Carlos, São Carlos-SP, Brazil.

–

[2]Escuela Superior de Informática, Universidad Castilla-La Mancha. Ciudad Real, Spain.

São Carlos, Brazil
July 14, 2019

Version Control			
Version	**Date**	**Name**	**Description**
0.1	07/20/2018	Bruno Santos	Documentation of methodology, information sources and requirements.
0.2	09/15/2018	Bruno Santos	Taxonomy design and description.
0.3	11/14/2018	Bruno Santos	Document update after UCLM partners contribution.
0.4	04/14/2019	Bruno Santos	Document update after UCLM partners contribution.

Contents

Chapter 1

Requirements

1.1 Methodology

This document present a taxonomy for Modernization Tools in the context of Architecture-Driven Modernization (ADM). The advent of ADM and its intention of turning KDM a de-facto metamodel into Modernization Tools has demanded a study around the terms in this context. Thus, the taxonomy presented here aims at supporting the understanding and correct classification of modernization tools in the context of ADM.

The main difference between existing approaches for classifying reengineering tools [3, 5, 1] is that our taxonomy is focused on classifying internal properties such as metamodels, abstraction level and purpose which can be applied in all sets of modernization tools. Existing approaches are more focused on the functionalities or in scenarios where the tools can be applied. For instance, Bourque and Abran [5] presents a taxonomy for software reengineering tools that contains classifications such as component extraction, design recovery and migration between software

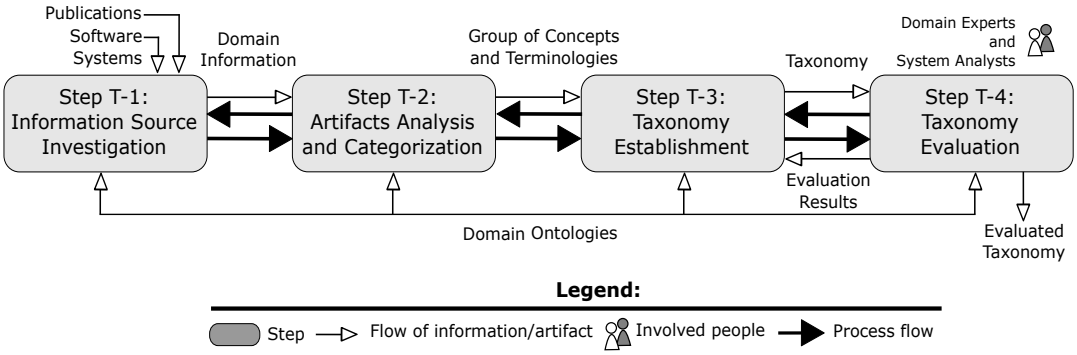

Figure 1.1: Methodology to establish a Taxonomy

engineering tools. Klain [1] describes categories for business process reengineering tools such as project management, modelling and business process analysis.

Our motivations with this taxonomy is (i) to assist software engineers in classifying existing software modernization tools and also (ii) helping modernization engineers in the design process of modernization tools.

In order to reuse existing tools it is important to the software engineers to know some information about these tools. However, sometimes it is hard to categorize modernization tools since there is a wide set of existing characteristics, thus it is crucial to know the minimum of important information in order to choose the set of tools to be reused.

Another point that can be mentioned as a motivation is when modernization engineers are developing modernization tools, a set of decisions have to be made so the tool's design could fulfill its purpose. With our taxonomy, we provide the set of initial decisions that need to be taken to secure design this kind of tool. We claim that with our taxonomy the problems mentioned before can be mitigated.

In order to establish our taxonomy, we adapted a methodology followed by Oliveira et al. [6] that was designed to elaborate a taxonomy of Services for Developing Service-Oriented Robotic Systems. Figure 1.1 presents the steps of the methodology. In short, to establish our taxonomy we select and investigate sources information (in Step T-1) in order to identify concepts and terminologies about the domain (in Step T-2). After that, the taxonomy and its description is established (in Step T-3) and the evaluation of this taxonomy is conducted (in Step T-4). Following, the steps important to build the taxonomy are presented in more details.

1.2 Step T-1: Information Source Investigation

The first step to establish our taxonomy was to identify the sources information to be used in the elicitation of concepts and terminologies. Different sources were considered, mainly related to ADM domain.

These sources can be classified into two sets: Set 1 - Formal specifications, glossary and white papers about ADM; Set 2 - Existing Modernization Tools identified in the literature. In order to identify the concepts and terminologies about modernization tools available in the literature we carried out a deep search on OMG[1] website, more specifically in the ADM part. In addition, a systematic mapping was also considered to identify information sources [4]. Following, each set of information sources is described in detail:

- **Set 1 - Formal specifications, glossary and white papers about ADM**: An important source of information is the set of artifacts provided by OMG about

[1]Object Management Group (OMG) is dedicated to bringing together its international membership of end-users, vendors, government agencies, universities and research institutions to develop and revise standards as technologies change throughout the years. https://www.omg.org/

ADM blueprint. We analyzed the ADM website, 4 white papers, 1 glossary and formal specifications about software modernization[2]. As a result, we identified that since ADM involves the concepts of software reengineering and MDA[3] we should consider the terms: (i) Reverse Engineering; (ii) Restructuring; (iii) Forward Engineering; (iv) Platform Specific Model (PSM); (v) Platform Independent Model (PIM) and (vi) Computation Independent Model (CIM). We considered these terms as concepts associated to software modernization in the context of ADM to help us building out taxonomy.

- **Set 2 - Existing Modernization Tools identified in the literature**: In order to see how the concepts of ADM were being implemented in real world we analyzed existing modernization tools in the literature and in the industry. We analyzed the modernization tools presented in a systematic mapping on ADM[4] and two books [9, 7]. The OMG/ADM make available a list of vendors[4] that somehow employs the ADM concepts in its tools or they supported the OMG/ADM in its consolidation process. We performed an analysis of these tools in order to understand its behavior and it role in the modernization process in order to support our taxonomy. As a result, we identified that modernization tools are directly related to the usage of metamodel not only to representing legacy knowledge through reverse engineering but also metamodels that supports the analysis of source code and the other steps of software modernization advocated by ADM. Another important concept found by analyzing the books is the concept of Modernization Scenarios that were took in consideration while composing our taxonomy.

1.3 Step T-2: Artifacts Analysis and Categorization

In this step we analyzed all the information sources found in Step T-1 and as a result we elicited the concepts presented in Table 1.1. These concepts represent the terms that are the base of our taxonomy but not all of them are explicit in the diagram that represents our taxonomy presented in Section 2.1.

Based on the analysis performed in order to find these concepts in the previous step we also elicited a set of constraints to help us in the taxonomy elaboration.

- **Constraint 1 - Purpose**: Modernization Tools should be categorized about

[2]This page provides a summary of OMG specifications that have either been formally published or are in the finalization process about software modernization. https://www.omg.org/spec/category/software-modernization

[3]Model Driven Architecture (MDA) is an approach to software design, development and implementation spearheaded by the OMG that provides guidelines for structuring software specifications that are expressed as models.

[4]https://www.omg.org/adm/directory.htm

Table 1.1: Concepts elicited from Step T-1

Concept	Name	Source (Derived from...)
C1	Reengineering	Set 1 and 2 of Step T-1
C2	Reverse Engineering	C1
C3	Restructuring	C1
C4	Forward Engineering	C1
C5	Model Driven Architecture	Set 1 and 2 of Step T-1
C6	PSM	C5
C7	PIM	C5
C8	CIM	C5
C9	Modernization Tools	Set 1 and 2 of Step T-1
C10	Modernization Scenarios	Set 2 of Step T-1
C11	Standard Metamodel	Set 1 and 2 of Step T-1
C12	Proprietary Metamodel	Set 2 of Step T-1
C13	Modernization Environment	C9 and C10
C14	Code-level	C5 and C6
C15	Model-level	C5, C7 and C8
C16	ADM-based metamodels	C11
C17	Computational Support	C9 and C13
C18	Technical Architecture	c10
C19	Application/Data Architecture	c10
C20	Business Architecture	c10
C21	Architecture-Driven Modernization	Set 1 of Step T-1

the Reegineering (C1) step that it represents, i.e. its purpose that could be Reverse Engineering (C2), Restructuring (C3) or Forward Engineering (C4);

- **Constraint 2 - Abstraction Level**: Modernization tools should also be categorized about its abstraction level according to MDA (C5) which could be represented by PSM (C6) or Code-level (C14) and also PIM/CIM (C7 and C8) or Model-level (C15);
- **Constraint 3 - Modernization Concepts**: It should be specified in the taxonomy how the modernization concepts (Tool and Scenario) presented in Table 1.1 are related to each other;
- **Constraint 4 - ADM standards**: While considering standard metamodels (C11), It should be implicit that there are specific metamodels that were designed by OMG to work with the ADM blueprint (C16).

While formulating and organizing these constraints we felt the need of a con-

cept that could involved modernization tools (C9) and modernization scenarios (C10). Thus, we created the **Modernization Environment** (C13) and **Computational Support** (C17) that encapsulate these two terms.

Note that several of these concepts are well know in software reengineering and MDA fields. The main reason to include them in our taxonomy is due to they are also related to software modernization and are also important concepts while characterizing modernization tools.

In general, the Step T-2 Artifacts Analysis and Categorization were important not only to identify the concepts but also to guide us during the creation of our taxonomy once we had to check if all constraints and concepts were being fulfilled, correctly represented and described.

Chapter 2

A Taxonomy for Software Modernization Tools

2.1 Step T-3: Taxonomy Establishment

This section presents the taxonomy of Modernization Tools that we built based on the concepts and constraints identified in Step T-2. We are using the terms Reengineering and Modernization as synonyms, i.e., both refer to the process of understanding a legacy system, analyzing it and transforming it in an improved version so that the maintenance costs go back to acceptable levels.

Our taxonomy is graphically represented by the conceptual diagram shown in Figure 2.1. The grey concepts in Figure 2.1 represents the classification labels in our taxonomy, which are: Modernization Scenario (MScen), Type, Purpose, MTool's Metamodel and Metamodel's Abstraction Level. Some of these labels are used to classify tools, other to classify metamodels and another to classify modernization scenarios.

Also in Figure 2.1, there are four main terms which are Modernization Scenario, Computational Support (CompSu), Modernization Tool (MTool) and Modernization Environment (MEnv). As can be seen in figure, Modernization Scenario "uses" Computational Support, that by its turn has two types: Modernization Environment and Modernization Tool. Notice that CompSu is being used here in a generic context, this is, CompSu generalizes the terms MEnv and MTool. A Modernization Environment can be a composition of Computational Supports that contains one or more Modernization Tools. In the following paragraphs more explanations of each term that appears in the taxonomy are provided.

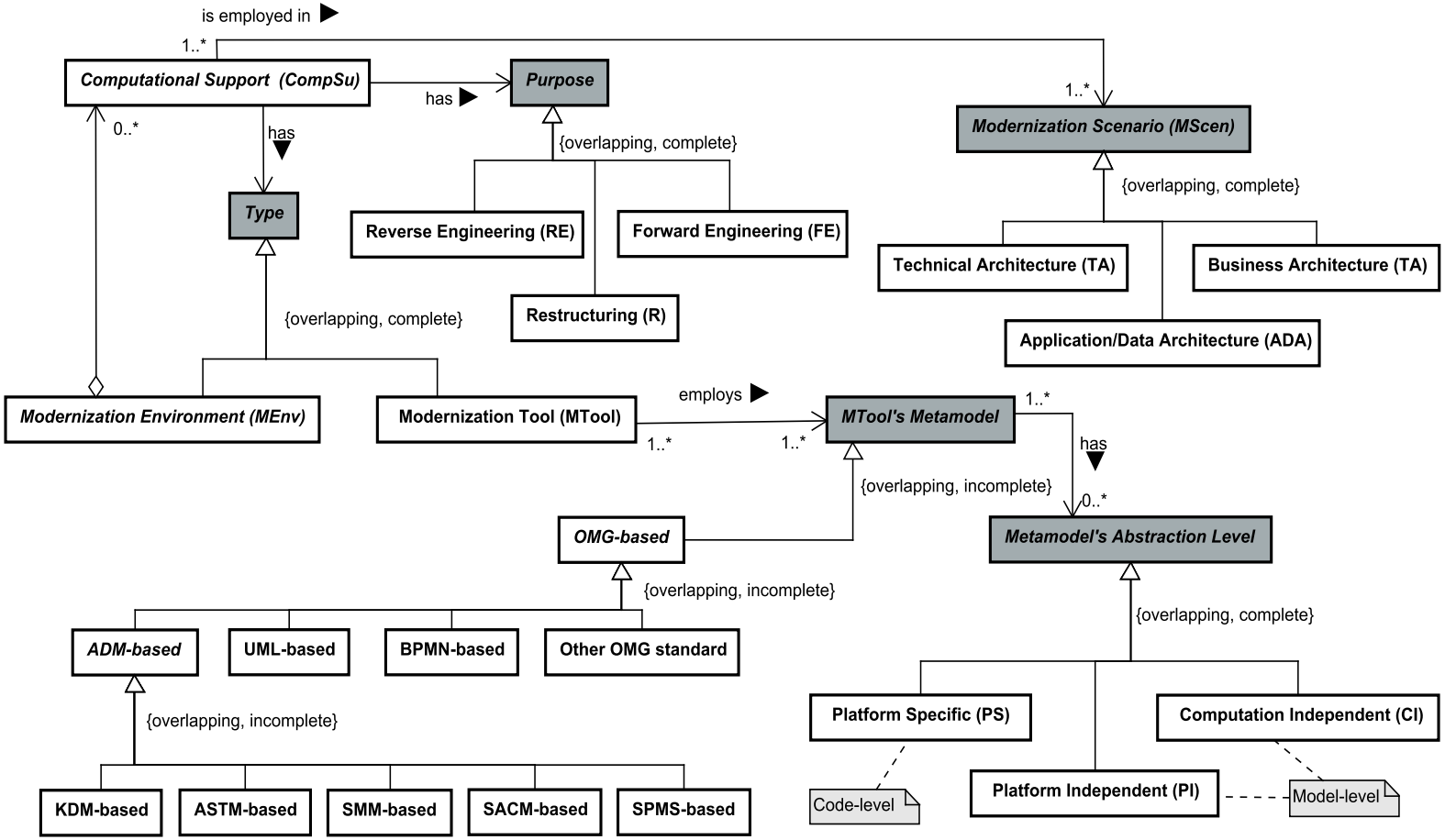

Figure 2.1: Modernization Tools Taxonomy - Conceptual Diagram

Modernization Scenarios are scenarios which show where and how a software system could be modernized and it also helps modernization teams on how to plan the modernization project [9]. MScens can act in three different architectural levels, which are: Technical, Application/Data, and Business Architecture. These architectural levels are important to define not only the set of MEnvs and MTools to be used but also the set of metamodels and their abstraction level that are going to be employed in the tools.

For instance, Language-to-Language Conversion is a Technical Architecture MScen that is responsible for the conversion of one or more software systems from one programming language to another. The main motivation for this scenarios could be the obsolescence of a language, or by the existence of a requirement to enhance a functionality not supported in the current language, and so others.

Computational Support is the most generical concept when talking about MEnvs or MTools. *In the context of software systems modernization, a CompSu is any computational solution that supports at least one of the reengineering processes*.

A CompSu has a **Type** and a **Purpose (grey concepts)**. Considering the Type, a CompSu can be a Modernization Environment or a Modernization Tool. About the Purpose, each CompSu has at least one and they correspond to the classical phases of reengineering processes, which are: Reverse Engineering, Restructuring and Forward Engineering. These purposes are needed in order to complete the whole modernization process.

Since CompSu supports the conduction of MScens, MEnvs and MTools should be carefully designed when choosing the metamodel once MScens demands specifics abstraction levels to deal with technical, application/data and business architecture modernizations. This is due to the fact that the metamodel choice has direct impact in the interoperability between tool of different purposes.

Other concept to be understood is **Modernization Environment**. *A MEnv is an aggregation of Computational Supports that by its turn can be MEnvs or MTools that support the conduction of* ***Modernization scenarios***.

For instance, MoDisco [2] is a MEnv that supports the reverse engineering phase by allowing the knowledge discovery of java source code in to different metamodel instances. With MoDisco is possible to recover, for instance, UML and KDM models. MoDisco is composed of several discoverers, that can be understood as MTools and each discoverer deals with a specific metamodel.

The most important term that needs to be understood is **Modernization Tool**. *MTools are tools that support the automation of any step of modernization processes*. As an example, a MTool can automatize (i) the generation of metamodel instances, (ii) the process of calculating metrics, and also (iii) the execution of model transformations/refactorings.

Modernization Tools can be classified according to its metamodels that they employ internally for representing and process the systems to be modernized. In

this taxonomy, only standard metamodels are being considered to provide the classification of MTools, once standard metamodels facilitate the interoperability of tools that use them.

The label ***MTool's Metamodel*** **(grey concept)** is used to classify MTools when considering its metamodels used internally by means of two main groups: ***OMG-based*** and ***ADM-based***. When a MTools uses an OMG standard it is possible to classify the tools as being **<*OMG-based*> MTools**. For instance, if a tool has algorithms that operate with BPMN the MTool will be classified as BPMN-based MTool. Which means that this BPMN-based MTool is prone to be interoperable with other BPMN-based MTools from other MEnvs.

Another possible classification for MTool considers the group of standards metamodels proposed by ADM which are the **ADM-based MTools**. ADM-based are MTools that use at least one of the standards metamodels that were proposed specifically to be employed in the ADM context. Thus, the classification of tools that use this kind of metamodel would be **<*ADM-based*> MTool**. For instance, a KDM-based MTool is a tool that uses the KDM metamodel in its internal mechanisms.

Considering the abstraction level of the metamodels employed in MTools, there is another classification, the **Metamodel's Abstraction Level** (grey concept). Knowing that MTools have algorithms to process metamodel instances, it is possible to classify the abstraction level that MTools operate. The three possible abstraction levels, according to MDA, are: Platform Specific, Platform Independent, and Computation Independent. Classifying the Metamodel's Abstraction Level of a MTool are directly linked to the metamodels that are being employed in the tool. Thus, it is possible to have more than one Metamodel's Abstraction Level in a single MTool.

Since platform specific models act in source code level we could also claim that these models are **code-level**. Now, considering the platform specific and the computation independent models they act in higher model levels if compared to source code, where some fine grained details can be omitted in order to provide a more abstract viewpoint of the same software system. Thus, we claim that they are **model-level**.

2.2 Classifying Computational Supports According to the Proposed Taxonomy

This subsection presents Table 2.1 that considers a set of CompSu and their classifications according to the proposed taxonomy. The first column contains the **Name** of the computational support and the second column has a short **Description** of each CompSu. The third, fourth and fifth columns classify computational supports according to theirs **Modernization Scenario**, **Type** and **Purpose**, respectively.

The last three columns classify more internally the **Composee MTools**. The

sixth column were added just to provide more information about **MTool's Function** in order to facilitate the understanding of the classification provided in seventh and eighth columns. The seventh column contains the classification of the tools that use OMG standards: **MTool's Metamodels**. Finally, the eighth column classify the MTool's Metamodels according to their abstraction level: **Metamodel's Abstraction Level**.

According to Table 2.1, KDM-RE acts in the **application/data architecture** MScen, it is as **modernization environment** with the purpose of **restructuring**. KDM-RE is composed of three modernization tools that uses OMG standards, two **KDM-based** and one **UML-based** which are **PI and CI**; and **PI**, respectively.

An important discussion here is on how to classify these CompSu as being MTools or MEnvs. To classify a CompSu it is necessary to undertand how it works internally according to its metamodels and its functionalities. For instance, in Table 2.1 we have Arch-KDM [8] that works only with KDM metamodel and a DCL for KDM. The approach that Arch-KDM is inserted has several steps and functionalities such as Planned Architecture Specification, Current Architecture Extraction and Architecture Comparison that return the found architectural drifts. Thus, we classify Arch-KDM as a MEnv since it involves more than one functionality.

The author of MoDisco [2] mention this tool as being a "A model driven reverse engineering framework", this is due to the fact that MoDisco has several discoverers that act independently according to user usage. In this sense, each discoverer acts as an independent tool, however the set of discoverers/tools are encapsulated and interoperable in a MEnv that they call as being a model driven framework. Thus, according to the proposed taxonomy, MoDisco is a **MEnv** that is composed of several **MTools** that has the **purpose** of **reverse engineering**.

There are MTools that supports completely a specific MEnv's purpose such as CloudMIG and RUTE-K2J and others that supports partially such as KDM-AO. Thus we claim that CloudMIG and RUTE-K2J are MEnv composed of MTools and KDM-AO is a MTool that can be a composee of a MEnv, since only by itself, it has no power to perform completely at least one of the MEnv's purposes, such as: Reverse Engineering, Restructuring or Forward Engineering.

In the other hand, it is completly possible to have a MEnv composed of MTools and other MEnvs from different purposes, for instance we could have a MEnv composed of MoDisco, KDM-RE and RUTE-K2J that would fulfill a complete modernization process that has all three CompSu purposes.

Table 2.1: Classification of Computational Support for Software Modernization

Name	Description	Classification according to					
					Composee MTools		
		MScen	Type	Purpose	MTool's Function	MTool's Meta-model	Metamodel's Abstraction Level
KDM-RE	It implements the Fowler's refactoring catalog and allow modernization engineers to apply them in KDM instances.	ADA	MEnv	R	Refactoring Application	KDM-based	PI and CI
					Refactoring Propagator	KDM-based	PI and CI
					As-is visualizer	UML-based	PI
RUTE-K2J	It takes a KDM instance as input and automatically generates a Java model from it.	ADA and BA	MTool	FE	Transform KDM instances in Java model	KDM-based	PI and CI
MoDisco	It is able to retrieve information from legacy source code and databases and represent them as KDM and other metamodel instances.	TA and ADA	MEnv	RE	ASTM Discoverer	ASTM-based	PS
					UML Discoverer	UML-based	PI
					Metrics Application	SMM-based	PS, PI and CI
					KDM Discoverer	KDM-based	PI and CI
GAFEMO	It is a framework for the modernization of legacy systems using a model-driven and service-oriented approach using the features provided by gap-analysis techniques.	TA, ADA and BA	MEnv	RE, R and FE	Obtention of logical model	ASTM-based	PS
					Refinement of logical model	KDM-based	PI and CI
					Refinement of business model	SBVR-based	CI
CloudMIG	It supports a semi-automatic legacy systems migration to cloud	TA and ADA	MEnv	RE	Metrics Calculation	SMM-based	PS, PI and CI
					Recovers KDM instances	KDM-based	PI and CI
MARBLE	It retrieves the business processes from existing systems using a set of model transformations.	TA, ADA and BA	MEnv	RE	Recovers KDM instances	KDM-based	PI and CI
					Recovers BPMN intances	BPMN-based	CI
CCKDM	It identifes crosscutting concerns in KDM instances. To do so the tool uses a combination of a concern library and a modified clustering algorithm.	ADA	MTool	RE	Analyzes and marks KDM instances	KDM-based	PI and CI
KDM-AO	It implements a light and a heavyweight extensions that enables the instantiation of aspect-oriented concepts in KDM instances.	ADA	MTool	RE	Enables the aspect oriented concepts instantiation	KDM-based	PI and CI
Arch-KDM	It helps in the conduction of Architecture-Conformance Checking (ACC) employing exclusively the KDM.	ADA	MEnv	RE	Planned Architecture Specification	KDM-based	PI and CI
					Current Architecture Extraction	KDM-based	PI and CI
					Architecture Comparison	KDM-based	PI and CI

Bibliography

[1] Pierre Bourque and Alain Abran. "An Innovative Software Reengineering Tools Workshop&Mdash;a Test of Market Maturity and Lessons Learned". In: *SIGSOFT Softw. Eng. Notes* 19.3 (July 1994), pp. 30–34. ISSN: 0163-5948. DOI: 10.1145/182824.182829. URL: http://doi.acm.org/10.1145/182824.182829.

[2] Hugo Brunelière et al. "MoDisco: A model driven reverse engineering framework". In: *Information and Software Technology* 56.8 (2014), pp. 1012 –1032. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2014.04.007. URL: http://www.sciencedirect.com/science/article/pii/S0950584914000883.

[3] E. J. Chikofsky and J. H. Cross. "Reverse engineering and design recovery: a taxonomy". In: *IEEE Software* 7.1 (1990), pp. 13–17. ISSN: 0740-7459. DOI: 10.1109/52.43044.

[4] Rafael S. Durelli et al. "A mapping study on architecture-driven modernization". In: *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*. 2014, pp. 577–584. DOI: 10.1109/IRI.2014.7051941.

[5] Mark M. Klein. "Reengineering methodologies and tools a prescription for enhancing succes". In: *Information Systems Management - ISM* 11 (Apr. 1994), pp. 30–35. DOI: 10.1080/10580539408964633.

[6] Lucas Bueno Ruas de Oliveira et al. "Towards a Taxonomy of Services for Developing Service-Oriented Robotic Systems". In: *The 26th International Conference on Software Engineering and Knowledge Engineering, Hyatt Regency, Vancouver, BC, Canada, July 1-3, 2013.* 2014, pp. 344–349.

[7] Ricardo Pérez-Castillo, I. G. R. de Guzmán, and M. Piattini. "Architecture-Driven Modernization". In: *Modern software engineering concepts and practices: Advanced approaches*. Jan. 2011, pp. 75–103. DOI: 10.4018/978-1-60960-215-4.

[8] A. d. S. Landi et al. "Supporting the Specification and Serialization of Planned Architectures in Architecture-Driven Modernization Context". In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. 2017, pp. 327–336. DOI: 10.1109/COMPSAC.2017.225.

[9] William M. Ulrich and Philip H. Newcomb, eds. The MK/OMG Press. Boston: Morgan Kaufmann, 2010, pp. 419 –429. ISBN: 978-0-12-374913-0. DOI: https://doi.org/10.1016/B978-0-12-374913-0.00025-1. URL: http://www.sciencedirect.com/science/article/pii/B9780123749130000251.