Understanding the Successes and Challenges of Model-Driven Software Engineering - A Comprehensive Systematic Mapping

Thiago Gottardi
University of São Paulo
Institute of Mathematics and Computer Sciences
P.O. Box 668 – 13.566-590 – São Carlos, São Paulo, Brazil
Email: gottardi@icmc.usp.br

Rosana Teresinha Vaccare Braga
University of São Paulo
Institute of Mathematics and Computer Sciences
P.O. Box 668 – 13.566-590 – São Carlos, São Paulo, Brazil
Email: rtvb@icmc.usp.br

***Abstract*—Model-Driven Software Engineering (MDSE) is a development method in which models are used to generate software. Despite documented advantages, projects employing MDSE may fail due to development challenges. In this paper, we study and document these challenges by conducting an up-to-date systematic mapping that goes beyond reviewing MDSE studies: we also include two derived paradigms (Model-Oriented Programming and Models at Run-time). Therefore, we present a systematic mapping with two objectives: The first objective was to identify specific domains in which MDSE is successful, while the second objective was to identify what are the challenges to apply this methodology to general purpose development processes. Following the review of 4859 studies (3727 are unique), we have identified the application and technological domains in which MDSE projects are more likely to succeed. We also discuss challenges presented by 17 primary studies. The analysis of the results indicate that MDSE application is consolidated in specific domains. A common feature identified among studies related to general purpose MDSE processes is that, initially, authors reported lack of proper methods and training. After new techniques have risen, it has been pointed that MDSE projects still face maintenance problems that can discourage their usage in other domains.**

***Index Terms*—Systematic Mapping, Review, Model-Driven Software Engineering, Secondary Study, Software Domains, Model-Oriented, Models at Run-time, Domains, Challenges**

I. Introduction

Model-Driven Software Engineering (MDSE) is a specific case of Model Driven Development (MDD) applied to software engineering. In these development methods, models are active artifacts during the software engineering process. Therefore, these models are not only used to describe design and concepts, they can also drive the development [1].

In order to develop software using MDSE, it is necessary to use modeling languages that allow the creation of machine readable models, which, in turn, can be executed or transformed into the final software. In this manner, it is possible to replace the source-code by models that may represent the software in higher abstraction levels, tightening the semantic gap between the problem and its software solution [2].

Besides adequate modeling languages, development teams also require specific tools to edit, validate and transform models, which may be specific to the project or domain. It is possible to categorize development teams that use MDSE into two groups. The first group is composed by specific domains that have suitable tools to use MDSE since the inception of development projects. However, the second group is related to domains with no previous tool definition, therefore, forcing developers to either adopt general purpose or develop their own MDSE tools by defining new modeling abstractions, for instance, by using MDA (Model-Driven Architecture) [3].

We have conducted a secondary study with the objective of identifying success cases in both domain specific and general purpose MDSE approaches. Therefore, the objective was to answer questions that relate to both of these groups.

Despite knowing that MDSE has been used successfully in several domains, we believe that the state-of-the art should be made much clearer. The identified related secondary studies by other authors are not recent and may not be based on systematic review approaches. Therefore, the main aim of our work is to capture the distribution of application and technological domains and discuss challenges related to MDSE.

Our review goes beyond searching MDSE studies, we have identified Model-Oriented Programming [4] and Models at Run-time [5] as software paradigms that were developed upon MDSE principles. Therefore, studies related to these concepts were reviewed exhaustively using the available search engines.

This work contributes to the study of MDSE approaches by providing a comprehensive secondary study in which we discuss a range of thirty-three years of publications after reviewing 4859 studies. We categorize success cases and summarize a set of identified challenges.

The remainder of this paper is organized as follows: Works related specifically to this paper are cited in Section II. In Section III, systematic mapping (SM) concepts are presented, as well as the activities done during the conduction of the SM. The quantitative results gathered from the SM process are listed in Section IV. These results are discussed in Section V, which is done by also including qualitative data. Limitations and threats to validity of this study are described in Section VI. Finally, the conclusions for the study are in Section VII.

II. Related Works

Model-Driven Development is not a new topic in software engineering. We have found 19 secondary studies among the search results. It is worth mentioning that the work by Asadi and Ramsin [6] is a secondary study closely related to this one. However, their work is not a systematic mapping but a literature survey. Also, their work is specific to MDA while we intended to identify every MDSE related approach, including non MDA-based.

In summary, there were no updated systematic mappings among the secondary studies. The work by Gottardi and Braga [7] includes a related systematic mapping but it was only focused on MDSE and last updated in 2015.

Whittle *et al.* [8] have put forward a survey applied on professional software developers in order to identify their problems when using MDSE in practice. For example, some authors have pointed that MDSE tools would evolve significantly and solve most of challenges [9]. However, in another work, Whittle *et al.* [8] have gathered evidences that indicate that this conception is not accurate, as the evolution of these tools should be more focused on the developers. In their study, they have identified that despite improvement of tools, there are still difficulties faced by developers, related to lack of methods, lack of training and misconceptions about the usage of models. The main similarity of this work is that it also investigates challenges related to MDSE application and we also argument that these difficulties could be all related to the lack of adequate methodology.

III. Systematic Mapping Execution

A Systematic Mapping (SM) is a specific method of literature review. It allows identifying and quantifying primary studies relevant to research questions in a specific knowledge area [10]. This SM was executed according to the guidelines by Kitchenham and Charters [11], which were defined in order to establish a systematic and repeatable literature review process. They recommend three phases for the executed process: planning; conducting and reporting.

A. Planning

The Planning Phase is the first phase. It contains activities in which the researchers develop a document named as "Protocol", which is shown in Subsection III-D. During the Planning Phase, "Data Extraction Plan" and "Quality Criteria Definition" are also developed. By defining the execution procedure in a review protocol, this process instance becomes controlled and repeatable, which is one of the primary objectives of following a systematic approach.

B. Conduction

The Conduction Phase is composed by the "Selection" and "Extraction" activities. These activities must be performed by following the established protocol.

The extraction activity was extended in this SM and deserves a further explanation. Its objective is to extract data and fill the extraction form ("Data Extraction"), which in our case is composed by categories planned as described in the protocol. The extracted data is employed to distribute the studies into the established quality criteria ("Quality Distribution"). For all studies we identified one or more categories and checked whether they included validation. This validation was later employed for quality distribution and for the challenges discussion.

C. Reporting

The last phase is the Reporting, with activities related to the data summarization. The first activity involves performing statistical analysis on the quantitative data ("Statistical Analysis"). Then, these results are summarized into text and plots. It also involves discussing the results in the effort of identifying new insights related to the study objects.

D. Protocol

An excerpt of the protocol definition for the SM performed is visible on Table I. This table contains two columns, arranged into field name and value pairs that include the objective, questions, intervention, control, results, source selection criteria, and study selection criteria. Among these fields, the questions and study selection criteria are frequently referenced during the results of the conduction phase presented in Section IV.

The most important item of the protocol is its objective. By intending to identify MDSE success/failure cases and challenges, two questions were devised. The first question aims to identify the success cases and the failures in specific domains. The results of this question are important because the search for challenges in MDSE outside these domains is also part of the objectives. If the success cases became too ubiquitous, then one could argue that the secondary question is irrelevant. This is because the secondary question is related to challenges encountered when applying MDSE into domains that do not have visible success cases. Without visible success cases, there would be a lack of tools or methodology, thus making the challenges more apparent.

Another important item of the protocol is the set of inclusion and exclusion criteria. Inclusion criteria "I1" and "I2" were created to respond the primary question, while "I3" is related to the secondary question. The exclusion criteria are employed to remove the unrelated studies and other results that are not primary studies.

E. Search Strategy

The searches were divided into different sessions, which were conducted to collect three different categories of studies.

The first category is focused on the software development approaches using modelling, i.e. not limited to MDSE. The second category is related to any study that involves model orientation, whereas the third category is related to any study involving models at run-time.

Therefore, three search strings were created, one for each category. It is worth mentioning that these search strings have been constructed by joining the basic keywords defined on Table II, complemented with synonyms and related terms.

TABLE I
PROTOCOL DEFINITION

Protocol Item	Item Description
Objective	The objective of this literature review is to identify success cases and challenges in MDSE approaches.
Primary Question	What are the specific domains in which developers have achieved success by employing MDSE?
Secondary Question	What are the the general purpose MDSE approaches and what are the challenges to create such approaches? Does MDA solve these challenges?
Intervention	Studies related to MDSE approaches and their challenges must be identified and categorized.
Control	The search results must involve a list of studies related to the questions that are known by the researchers. This list includes articles and books by Pastor, Whittle and Czarnecki.
Results	Quantitative data on approach frequency distribution within domain categories. Qualitative data on reported challenges.
Source selection criteria:	Source must index studies on Computer Science, Mathematics or Engineering. Source must allow Boolean operators. Source must be accessible by the researchers.
Selection Criteria:	**Inclusion:** • I1 - Primary studies that present a success case of MDD, MDSE, DSL or MDA in a specific domain; • I2 - Primary studies that present a non success case of MDD, MDSE, DSL or MDA in a specific domain; • I3 - Primary studies that present challenges of applying MDD, MDSE, DSL or MDA in general purpose projects; **Exclusion:** • E1 - Unrelated to MDD, MDSE, DSL or MDA. • E2 - Not a primary study.

TABLE II
MDSE SOFTWARE DEVELOPMENT APPROACH SEARCH STRING DEFINITION

Identifier	Keyword	Synonyms and Related terms
A	Software Development	• software development; • software engineering;
B	Approach	• approach; • process;
C	Support	• tool; • support;
D	MDSE and MDD	• mdd; • development; • mde; • engineering; • software; • mda; • model-driven architecture; • model driven architecture; • Model-Driven; • model; • driven; • model-driven. • model-oriented • model oriented.

TABLE III
MODEL ORIENTATION SEARCH STRING DEFINITION

Identifier	Keyword	Synonyms and Related
E	Model Orientation	• model-orientation; • model orientation; • model-oriented • model oriented.

The keywords shown on Table II were used as basis to create the first category search string. The final search string is obtained after a conjunction operation (represented by "$\wedge$") applied to the table rows and a disjunction operation applied among the synonyms of each keyword. Therefore, the final search string is (A) $\wedge$ (B) $\wedge$ (C) $\wedge$ (D).

For the second category, the keywords shown on Table III were used. Since there is only one row, the disjunction operation is applied among the synonyms of the keyword, with the intent of capturing any study related to this keyword.

The third category was also defined to capture any study related to the specified keyword. Therefore the keyword shown on Table IV was used. Since there is only one row, the disjunction operation is applied among its synonyms.

The searches were planned to be carried out through the following search engines: ACM Digital Library [1], IEEE Xplore [2], Engineering Village Compendex[3], Wiley Digital Library[4], Web of Science[5], Science Direct[6], Elsevier Scopus[7], Springer Link[8] and Google Scholar[9],

However, some search engines were canceled after a few search sessions, since it was not possible to calibrate or to complete the selection completely.

The search was then concluded through the following search engines: ACM Digital Library (DL), Engineering Village Compendex (EV), IEEE Xplore (IEEE) and Elsevier Scopus (Scopus).

Their update dates may vary, as specified in Table V. The initial search sessions were executed on May 30^{th}, 2014 by collecting studies from all reported search engines and then updated systematically until January 1^{st}, 2018, effectively reaching 4859 studies. It is important to point that due to the broad nature of this search, this is a continuous work of literature review that would never be finished.

Despite not updating all engines completely, these searches returned more studies from past years that were not collected during the searches done on 2014, effectively exceeding the number of studies found during initial searches, since they did

[1]http://dl.acm.org
[2]http://ieeexplore.ieee.org/
[3]http://engineeringvillage.com
[4]http://onlinelibrary.wiley.com
[5]http://wokinfo.com/
[6]http://sciencedirect.com
[7]http://scopus.com
[8]http://link.springer.com
[9]http://scholar.google.com

TABLE IV
MODELS AT RUNTIME SEARCH STRING DEFINITION

Identifier	Keyword	Synonyms and Related Terms
F	Models At Runtime	• models at runtime; • models at run-time; • models at run.time; • models at run time; • model at runtime; • model at run-time; • model at run.time; • model at run time; • models @ runtime; • models @ run-time; • models @ run.time; • models @ run time; • model @ runtime; • model @ run-time; • model @ run.time; • model @ run time; • models@runtime; • models@run-time; • models@run.time; • models@run time; • model@runtime; • model@run-time; • model@run.time; • model@run time;

TABLE V
SEARCH SESSIONS AND UPDATES

Search Category	Search Engine	Last Update	Returned Study Count
MDSE	ACM DL	May 30th, 2014	26
	IEEExplore	September 6th, 2017	405
	EV	May 30th, 2014	655
	Scopus	September 8th, 2017	2805
MO	IEEExplore	January 1st, 2018	18
	EV	January 1st, 2018	37
	Scopus	January 1st, 2018	42
MRT	IEEExplore	December 9th, 2017	65
	EV	December 9th, 2017	389
	Scopus	December 9th, 2017	417
All	ACM DL		26
	IEEExplore		488
	EV		1081
	Scopus		3264
	Total		4859

not include other categories besides MDSE. Nevertheless, all studies returned by engines were thoroughly evaluated according to the processes established previously in this section.

F. Study Selection

After ensuring that there were no duplicated studies, the review was conducted by analyzing studies returned by each search engine. The search engine priority was set by reviewing first the engine that returned the higher number of previously known studies.

The data extraction form used for all studies contains fields that must be filled during the extraction phase. The planned form contains three fields:

1) Identified Success Case Domain;
2) Identified Failure Case Domain;
3) Identified MDSE problem or challenge.

The valid values for each of the enumerated fields are presented on Table VI. These valid values are defined as nominal sets, i.e., groupings of enumerated and named items that represent categories important to our study.

TABLE VI
VALID VALUES FOR DATA EXTRACTION FIELDS

Field Number	Field Name	Field Type	Cardinality	Nominal Set
1	Identified Success Case Domain	Subset of Nominals	zero to many	Domain Set
2	Identified Failure Case Domain	Subset of Nominals	zero to many	Domain Set
3	Identified MDSE Problem or Challenge	Subset of Nominals	zero to many	Problem Set
4	Identified Validation Type	Subset of Nominals	zero to many	Validation Set
5	Presents Solution for an MDSE challenge	One Nominal item	one	Boolean Set

The first nominal set is named as Domain Set, which contains 57 nominals, including domains related to Web, Embedded Systems, Business Information Systems, Telecommunications and Networking, Industrial Control Systems, Military, Parallelism, Simulation, Computer Aided Design, Education and Computer Games. A complete list of these nominals is shown on Table VII.

It is important to point that these nominals were defined to allow hierarchical analysis. For example, every nominal which starts with "Embedded System" is considered as a child of the first "Embedded System" nominal, then, upon counting the numbers of studies that are related to this domain, the studies marked with any child nominal are also counted along with their parents. The domains were distributed into application and technological domains during results analysis.

The second nominal set is the Problem Set, which contains the nominals "Methodology Problem", "Maintenance Problem", "Testing or Validation Problem" and "Tools Problem". It is important to point that throughout the study the focus on "Tools Problem" was diminished since it is unclear to define whether the study exposes MDSE problems or the authors were simply encouraged to create new tools. Moreover, this issue was already covered in the literature [12].

TABLE VII
DOMAIN SET NOMINALS

Application	Technological
Tourism	Autonomous Mobile Robotics
Telecom	SoS (System-of-Systems)
E-commerce	System Virtualization
Farming	Ubiqua/Pervasive
Automotive	Cloud/Data Warehouse
Government	Robotics
CAD (Computer-Aided Design) Tool	Large Scale
CPS (Cyber-Physical System)	Middleware
Game	User-Interface
Academic	Mobile
BioInformatics	Service-Oriented/Web Service
Multimedia	Parallelism
Math/Theoretical Scientific	Web - Cloud Computing
Military/Defense/Aerospace	Hardware
Control System	Fault Tolerance / Adaptative
Health	Software Architecture
Human Interaction	Embedded System
Industrial	Web
Simulation	Network
Education	
Communication	
Business Information Systems	

The third nominal set is the Validation Set, which contains the nominals "Case Study", "Experimental or Empirical Study", "Experience Report", "Feasibility Study", and "Proof or Demonstration". The Boolean set contains two nominals, true and false. This set is used to indicate whether the study presents a solution to the challenges or problems identified in MDSE.

G. Study Quality Criteria

The quality criteria were divided into two groups. The first group is used to select the studies related to domain successes and failures. In this case, we have defined that these studies must not present the domain simply as a case study, that is, to avoid papers that use a domain as a validation without providing a practical success or failure report.

Considering the second group, it is related to the studies that present challenges or problems related to MDSE. We have planned strategies for defining if these studies are relevant for our secondary study. Therefore, we have defined that only the studies that either have a solution for the challenges or contain a validation should be carried into the discussion activity.

H. Employed Tools

A custom set of tools was developed by the author to be employed during the conduction phase. Their requirements involved supporting the hierarchical nominals and allowing the researchers to cooperate on the same review and to provide real-time reports about the review evolution and preliminary summarization via a web-page that features graphics and descriptive statistics. More details of these tools are also available within the packing documents.

IV. SECONDARY STUDY RESULTS

This section contains the summarization of results, which was carried out after conducting the extraction phase. Qualitative and quantitative data that were used to respond the research questions are provided.

A. Search Results

The aim of this subsection is to provide information regarding the results returned by the search engines. The searches returned 4859 studies, in which, 3727 were not duplicated. The extraction phase was split into two, one for each research question: 2628 studies were selected for domain extraction while 106 were selected for MDSE challenges extraction.

Figure 1 contains a plot that was created to allow a general visualization of the distribution of collected studies. The graph was designed as a stacked bar plot, which allows the viewer to compare the portion of duplicated and unique results returned from each search engine.

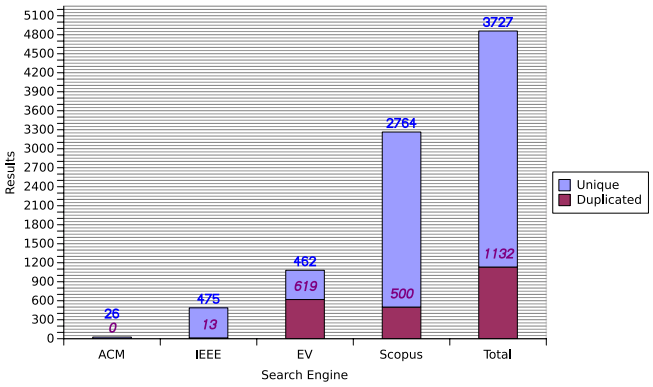

Fig. 1. Source Distribution

The columns of Figure 1 are named as "ACM", "IEEE", "Scopus" and "EV" since they represent, respectively, the search engines ACM Digital Library, IEEE Xplore, Elsevier Scopus and Elsevier/Engineering Village Compendex. In order to improve the visualization of the plots here shown, it was established that they would start from one year before the year that got the oldest results. In every bar plot, the vertical axis contains the number of studies, whereas the horizontal axis may represent categories, process phases or years.

In this review, studies were not filtered by their publication year. The oldest study that passed the selection phase was published in 1985 and written by Hoffnagle and Beregi [13]. Their study is related to automated software generation, however, since there is no explicit model as input, it was not categorized as MDSE.

Alkadi and Carver [14] wrote the oldest study that was considered as related to MDSE by this review process. It was published in 1998. They created an approach which employs models for test case generation.

B. Domain Categorization: Results

The aim of categorizing the studies was to identify software solution domains in which MDSE is successful. These categories are divided into application domains and technological domains.

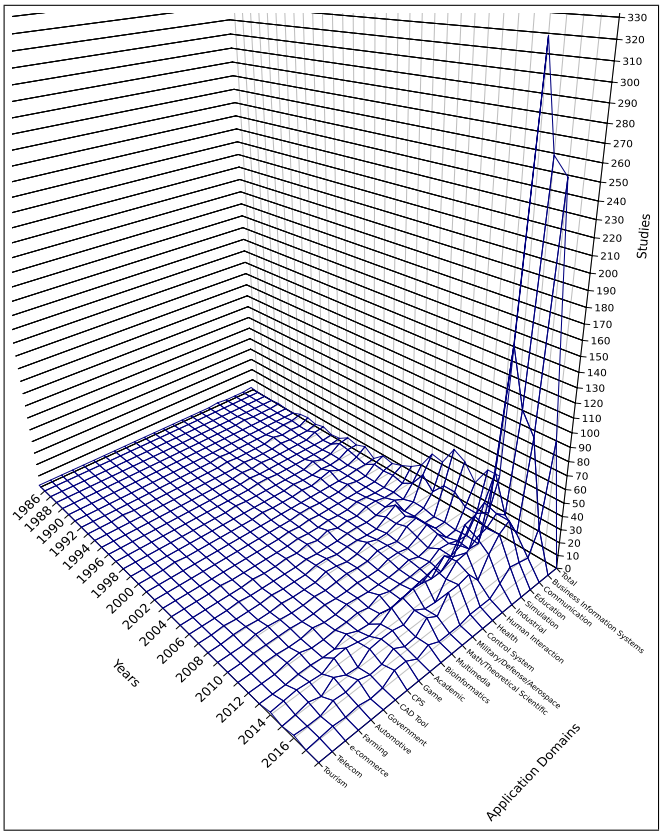

Fig. 2. Evolution of Most Common Application Domains

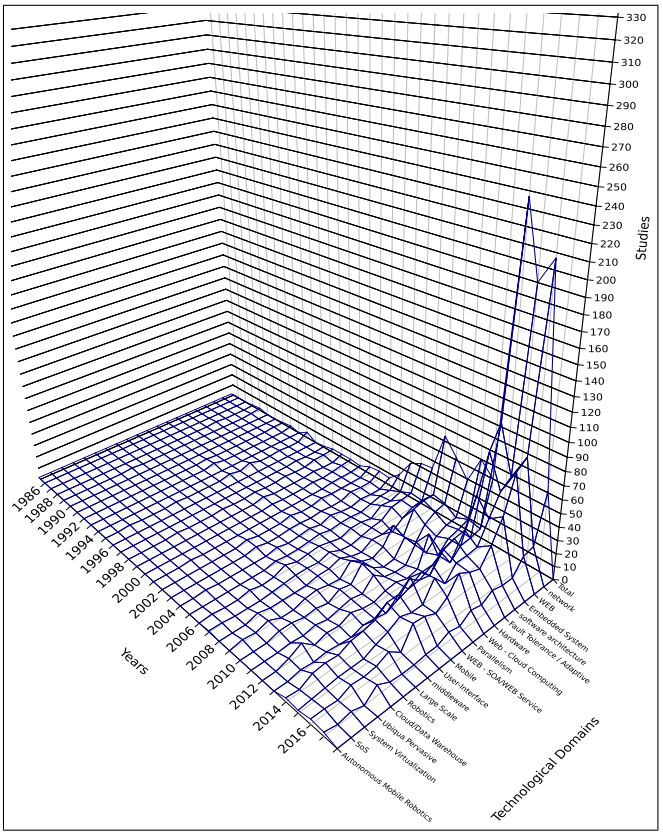

Fig. 3. Evolution of Most Common Technological Domains

The application domains are visible in the plot of Figure 2. This plot contains the years spanning from 1985 to 2017 while showing the study count per application domain (presented on Table VII). These domains are ordered by maximum count. The first item shows the total count per year. Therefore, the graph shows that the top five application domains are Business Information Systems, Communication, Education, Simulation and Industrial.

In the same sense, technological domains are shown in a similar plot in Figure 3. This plot also includes the years and study count per domain from Table VII. The first element shown is the total per year. The top five domains that have been identified are: Networks, Web, Embedded Systems, Software Architecture (design and generation), and Adaptive Systems.

Although not planned prior to execution, it has been identified that the use of MDSE in Embedded Systems Domain is very scattered among different subdomains. To avoid concerns related to how broad is the categorization of this domain, we have identified Embedded Systems subdomains which employ MDSE. These subdomains are presented in Figure 4, which contains a plot showing the evolution of number of studies in these domains. The subdomains "Avionics and Aviation", "Robotics", "Agriculture", "Cruise Control", "Home Automation", "Sensors and Actuators" and "Road Vehicle" are distributed per year. It is important to point that the "Total" bar is also the total count of all studies related to Embedded Systems with or without a specific subdomain.

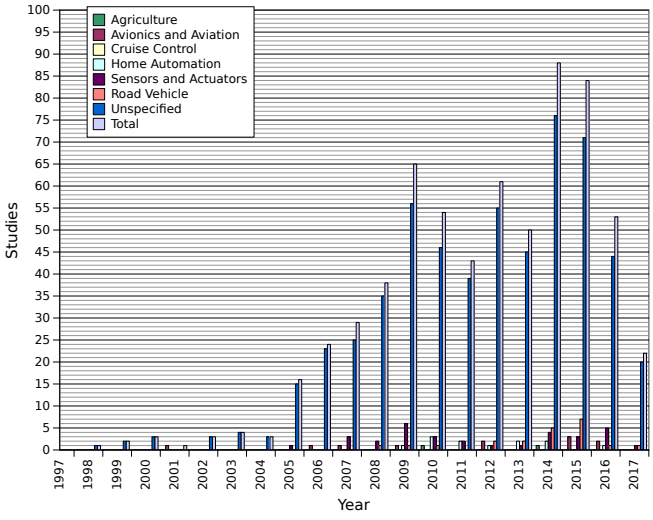

Fig. 4. Evolution of Embedded Systems Application Domains

C. MDSE Challenges: Results

The goal of this subsection is to provide quantitative results regarding the number of studies that present challenges on applying MDSE into general purpose projects.

After the review process, eight studies that contain references to maintenance challenges were identified. In the same

manner, nine studies that discuss methodology challenges were identified. Considering these studies, there is the total number of seventeen unique studies, since there are two studies which are common in both listed categories. Further discussion regarding this data is presented in Section V.

V. Results Discussion

This section presents a discussion about the SM results, along with further qualitative data. It includes discussion on the identified software domains, MDSE remaining challenges and limitations of the study.

A. Domain Distribution

The results helped to confirm our previous expectations about MDSE tools intended for Business Information Systems, Web and Embedded Systems. By reading the related studies, we believe that a constant concern among these approaches is to accelerate the implementation of data entities and their manipulation.

However, we did not expect that the network domain would achieve high frequency in the identified distribution. Most of the MDSE studies in the network domain are related to the development of parallel distribution of software execution.

B. MDSE Remaining Challenges

The General Purpose Challenge is related to the secondary question of this SM. This question was planned because our research team is investigating the application of MDSE in non-conventional domains and in general purpose projects.

Another result of this SM is that the number of primary and secondary studies related to challenges in unconventional domains and in general purpose methodologies of MDSE application is far below the expected, i.e., 2 in 4859 studies. The search results were carried out exhaustively, which means that all studies were checked for challenge discussion.

During the conduction we have identified nine works that point methodology problems in MDSE besides the proposal of MDA [15]–[23]. We have also found eight studies that contain information on maintenance problems that occur in projects that employ MDSE [23]–[30]. It is important to note that there are 2 studies which are common to both categories. Consequently, there are seventeen studies presented in this subsection.

The summarization of these studies is presented on Table VIII. This table is ordered chronologically and contains the reference number of each study, authors, title, problem type and study type. The problem types are split into two columns: methodology and maintenance, which represent, respectively, methodology problems and maintenance problems. Therefore, the study lines that deal with specific problem types have their respective cells shaded to indicate that they relate to the problem type.

Chitforoush *et al.* [17] are among the researchers who identified that MDSE lacks methodology, processes and guidelines to instruct when developers should use each model in a MDA-based project. In their attempt to provide a solution to this problem, they have defined a general methodology framework based on MDA. These authors claim that their framework is flexible enough to be adapted to various processes and needs. They also compared their framework to similar approaches. However, they do not provide validation on the efficiency of their approach. They have suggested this validation as future work but we could not find it published.

Asadi *et al.* [16] are from the same research department and created another solution to the problem identified by Chitforoush *et al.* [17]. They have defined a MDSE development life-cycle, which has as main advantage to propose more specific process definitions to guide developers using MDA. However, the stricter definition could also affect flexibility. The authors compared their approach to related approaches and we could not find a validation study. It is worth mentioning that their work was published after a survey on MDA problems by the same author [6].

Both of these studies have identified problems in the methodology of previous works. Despite this identification, it was not possible to find any validation in these studies pointing whether the problem was completely solved. It is also worth citing the analytic survey by the same authors [6], which provides the theoretical foundation used to create their new approach.

TABLE VIII
MDSE Challenges Summarization

Ref.	Title	Method	Maint.	Study Type
[24]	Engineering long-lived applications using MDA			Experience Report
[17]	Methodology support for the model driven architecture			Method Proposal
[16]	An MDA-based system development lifecycle			Method Proposal
[26]	MDA Tool Components: A proposal for packaging know-how in model driven development			Model Management Model
[18]	Adapting Software Development Process towards the Model Driven Architecture			Method for Process
[20]	Integration of domain-specific models into a MDA framework for time-critical embedded systems			Method for Process
[19]	Integration of MDA framework into the model of traditional software development			Method for Process
[23]	Challenges in Deployment of Model Driven Development			Experience Report
[15]	An outline of conceptual framework for certification of MDA tools			Method Proposal
[30]	From aspect-oriented models to aspect-oriented code? The maintenance perspective			Experimental Assessment
[25]	MoDSEL: Model-driven software evolution language			Language Proposal
[27]	Merging of EMF models: Formal foundations			Algorithm for Maintenance
[28]	From model-driven software development processes to problem diagnoses at runtime			Generator Dev. Method
[22]	Using software categories for the development of generative software			Generator Dev. Method
[29]	Co-evolving meta-models and their instance models: A formal approach based on graph transformation			Algorithm for Maintenance
[21]	A model-based workflow from specification until validation of timing requirements in embedded software systems			Method Proposal
[31]	Model-Oriented Web Services			Method Proposal

Nikulsins and Nikiforova [18] have described the need for customized processes. They have studied how to adapt Rational Unified Process and Microsoft solutions framework to support MDA. This need for customized process is further described in a more recent work [19], in which the authors point that MDA provides no guidelines for activities, roles, phases and responsibilities.

While facing a similar issue, Sanchez *et al.* [20] and Noyer *et al.* [21] have noticed the lack of a model-based process for embedded systems development.

Cernickins *et al.* [15] also have described a methodological problem, however, their focus is on tool certification. Thus, their certification framework contains guidelines that can be employed to identify if a tool set or a project is lacking an important activity or feature which they claim to be necessary.

Besides these studies, Nazari and Rumpe [22] have devised instructions specifically for developing software generators, which is a specific activity related to MDSE projects.

Seifert *et al.* [24] have written an experience report where they point that the use of MDSE increases dependency of the tool chain. They argue that this problem is not only limited to custom made tool chains, because tools may be updated and become incompatible to the older model instances. This is caused by changes on the language definition. For instance, new UML definition versions are made available and tool developers may follow the new definitions and break backward compatibility [24].

Bendraou *et al.* [26] have discussed the need for packaging metadata about the artifacts used within projects that employ MDA principles. This packaging would then support the maintenance activities.

Hovsepyan *et al.* [30] have studied the impact of code-generation on software maintenance. The most important contribution of this study, when compared to others presented herein, is the in depth statistical analysis of the maintenance impact using metrics to compare the results in a quantitative manner. However, it focuses on specific models for a programming paradigm [30].

Er and Tekinerdogan [25] describe a language named "MoDSEL" (Model-Driven Software Evolution Language). They claim that this language can be used to compare models, track their changes and identify maintenance impacts. Therefore, this is a solution that deals with the maintenance problems that may be found in software projects employing MDSE [25].

Westfechtel [27] and Mantz *et al.* [29] have discussed the need for formal foundations to merge models. These foundations are employed to implement tools to handle maintenance issues faced by developers when dealing with version conflicts that may arise during development and maintenance.

Yu *el al.* [28] have discussed the problem that arises when it is required to provide maintenance to code generators. In their study, they have devised a tool to help debugging software produced by model-driven development.

The studies shown on Table VIII span eleven years of publications. Roughly in the middle of these years would be 2009. Prior to 2009, 5 out of 7 studies were focused on the methodology problem. After 2009, 6 out of 8 studies were focused on the maintenance problem, which could indicate a trend on the research efforts.

The first study that covers both categories was published in 2009. It was written by Teppola *et al.* [23]. It contains an experience report created by applying surveys on software development companies. These authors have described that software developers using MDSE perceive both maintenance and lack of methodology issues. They claim that despite the advantages that have been experienced by the developers while using MDSE, there are still several issues to be treated. The authors conclude that the developers are optimistic hoping that these issues are solved because they approve the use of MDSE regardless of its current limitations [23].

Similarly to Asadi and Ramsin [6], Gottardi and Braga have also published a secondary study [7] and then devised a method proposal. In their proposal, they have suggested to evolve MDSE into a new development method, including a development method with maintenance flexibility. While this suggestion could be interesting outcome for the method, their evolution attempt can be tracked back to Model-Oriented Programming. In this paradigm, a programming language has been defined to tighten the gap between design and code [32]. This allows the developers to derive code from design and design from code in a round-trip engineering method that avoids losing semantics from either kind of artifact throughout the development phases. According to this paradigm, a set of studies in this context have been reviewed, however, they have not added further discussion to the identified challenges.

Models at Run-time can also be argued as a paradigm derived from MDSE. Aßmann *et al.* have discussed that this paradigm puts a step forward on reflective programming, allowing software to reach higher adaptability levels [5].

As we discuss the span of studies on MDSE and related paradigms, it is clear that MDSE has reached a productivity level for specific domains in which it achieved successes. It is suggested that the legacy of MDSE research could be employed into new advanced paradigms to increase productivity, software quality and adaptability.

It is not part of this study to discuss other secondary studies, although it is important to list related studies. A related systematic mapping has been performed [7], but it lacks updates after 2015 and only reviewed 1651 studies compared to 4859 of this study, which also includes paradigms derived from MDSE.

An analytic survey, which is part of the work described by Asadi and Ramsin [6], should be cited as an initial discussion of limitations of MDA and MDSE. It is worth mentioning that there are other two works by Whittle *et al.* [8], [12] that were collected as control for this secondary study, and were not added to Table VIII. These studies also indicate that the evolution of MDSE tools should focus more on the human aspects of developers, who still face difficulties when trying to use them. Further details on related works are presented within Section II.

VI. Study Limitations and Threats to Validity

The aim of this subsection is to provide details on the identified limitations and what we have done to mitigate them. The limitations were categorized by their origin, which include "Search Strategy", "Study Selection", "Data Extraction", and "Researcher Bias".

Search Limitations: There are a few limitations related to our search strategy. The first of them is regarding to the search string. After calibrating the string in order to achieve most relevant results, it is possible that the string lost part of the original intended semantics and may fail to return some of the intended studies. To mitigate this problem, we have identified which search engines indexed the preliminary known primary study defined as "control" in the protocol. Then we confirmed that the search string was enough to cause the search engine to return the indexed known studies.

The number of search engines is less than the initially planned. It was originally intended to include the databases by ISI Web of Knowledge, Science Direct, Wiley InterScience and Google Scholar. The decision to cancel conduction of results carried by Springer Link was late in our process. This was decided because this search engine provided a small relative number of relevant studies and the high number of results was affecting the review time. After these issues, we believe that it is important to plan the execution phases by reviewing the studies from each database sequentially in a "shortest job first" strategy instead of our adopted sequence.

Study Selection: The guidelines by Kitchenham and Charters [11] have been defined considering that the selection activity should only be used to define which studies should proceed to the extraction activity. However, the authors preferred to categorize every study since the selection phase.

The impact of this approach is unknown. In our strategy, the selection phase became longer, however, we believe that it was positive to avoid another limitation of this phase: it was not possible to reject studies during the selection phase without providing general categories to the study, which is an evidence that no studies could have been rejected without proper reading of its abstract.

Data Extraction: The main item analyzed during data extraction was identifying the set of categories in which each study should be linked to. This also includes the application domains.

A constant concern during the execution was to provide an exhaustive categorization of every returned study. Several categories were planned before the conduction. After discovering more specific studies, the authors then decided to create an hierarchical category definition interactively in order to provide an in depth report.

Since it was not possible to define if the interactively defined categories should be applied to studies reviewed prior to their definition, only a few categories were selected, and the review was restarted considering the new category set.

Considering that only the studies that provided information related to problems to apply MDSE were selected for discussion, the rate of discussed studies has became much inferior than the expected numbers. However, as the primary question dealt in this study was to provide a systematic mapping related to the most common domains, this issue should not affect its credibility.

Researcher Bias: Since this work was carried out by two researchers, the risk of researcher bias affecting the results is considerably high.

In order to mitigate this threat, we have defined keywords for each category and established systematic approaches to carry the review as impartially as possible.

All the studies that present information regarding MDSE challenges were reviewed extensively. However, the studies unrelated to the challenges topic were not fully read during the process.

VII. Conclusion

A secondary study has been presented in this paper. This study is a comprehensive systematic mapping with the intent of identifying the successes and failure cases of MDSE. Another objective was to identify the challenges of MDSE while discussing whether they have been solved.

After reviewing a total of 4859 studies, the most common domains have been identified as well as discussions related to challenges developers face while attempting to apply MDSE to projects dealing with uncommon or too specific domains.

As part of results summarizing, we have identified that the MDSE success domains are clustered into application and technological domains. This data was presented quantitatively considering the success cases that were not only used as case studies. The success cases indicate that MDSE has reached production levels for specific domains. In this manner, it is suggested that MDSE is recommended for specific domains, involving both academia and software industry.

During our searches, we could not find a report on a failure case, still, we identified challenges and presented these qualitatively in a discussion section.

There are 17 identified studies which are related to challenges in employing MDSE. This discussion involved challenges related to software maintenance and methodology issues. The studies have also been categorized and summarized.

The studies with challenge discussion encourage new approaches to cope with the existing issues related to MDSE. In the context of methods and maintenance, it is possible that new processes, techniques, tools and developer training could mitigate both of these issues. As the recent development of new tools has taken place, in this thesis, we discuss how other approaches could be employed besides creating new tools.

This systematic mapping presented the review of studies from 1985 to 2018, however, studies related to MDSE challenges are a scarce. Therefore, it is difficult to point whether a challenge has been dealt with in the recent years.

Moreover, we could find no evidences that the proposed solutions were in fact used. Considering the maintenance problems, we argue that these issues could rise in any project and they should be mitigated since its beginning.

After discussing the successes and challenges, we also point how paradigms derived from MDSE suggest that the MDSE research legacy could lead to new methods and techniques to improve software quality and adaptability in ways beyond the original proposal of code generation.

ACKNOWLEDGMENT

Many thanks to Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP – process number 2016/05129-0) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES – process numbers DS-8428398/D and BEX 3482-15-4) for funding received during the development of this work.

REFERENCES

[1] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven Software Engineering in Practice*, ser. G - Reference, Information and Interdisciplinary Subjects Series. Morgan & Claypool, 2012. [Online]. Available: http://books.google.com.br/books?id=2tVu-wC4XkAC

[2] R. France and B. Rumpe, "Model-driven development of complex software: A research roadmap," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 37–54. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.14

[3] OMG, *Overview and guide to* OMG*'s Model Driven Architecture*, http://www.omg.org/cgi-bin/doc?omg/03-06-01, Object Management Group Std., Rev. 2.3, May 2010. [Online]. Available: http://www.omg.org/cgi-bin/doc?omg/03-06-01

[4] O. Badreddin, A. Forward, and T. Lethbridge, "Exploring a model-oriented and executable syntax for uml attributes," *Studies in Computational Intelligence*, vol. 496, pp. 33–53, 2014, cited By 6. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84958544467&doi=10.1007%2f978-3-319-00948-3_3&partnerID=40&md5=04f8343e3081a1567898e8875b3aebe3

[5] U. Amann, S. Götz, J.-M. Jézéquel, B. Morin, and M. Trapp, *A Reference Architecture and Roadmap for Models@run.time Systems*. Cham: Springer International Publishing, 2014, pp. 1–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08915-7_1

[6] M. Asadi and R. Ramsin, "Mda-based methodologies: An analytical survey," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5095 LNCS, pp. 419–431, 2008.

[7] T. Gottardi and R. T. V. Braga, "Model driven development success cases for domain-specific and general purpose approaches: A systematic mapping," in *XVIII CIbSE*, URP,SPC,UCSP. Lima-Peru: UCSP, April 2015, pp. 432–445.

[8] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "Industrial adoption of model-driven engineering: Are the tools really the problem?" in *Model-Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. Clarke, Eds. Springer Berlin Heidelberg, 2013, vol. 8107, pp. 1–17.

[9] O. Pastor and J. C. Molina, *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Secaucus, NJ, USA: Springer-Verlag New York, 2007.

[10] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering*, vol. 17, 2008, p. 1.

[11] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, UK, Tech. Rep. EBSE 2007-001, 2007. [Online]. Available: http://www.dur.ac.uk/ebse/resources/guidelines/Systematic-reviews-5-8.pdf

[12] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "Industrial adoption of model-driven engineering: Are the tools really the problem?" in *MoDELS*, ser. Lecture Notes in Computer Science, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, Eds., vol. 8107. Springer, 2013, pp. 1–17.

[13] G. F. Hoffnagle and W. E. Beregi, "Automating the software development process," *IBM Systems Journal*, vol. 24, no. 2, pp. 102–120, 1985.

[14] I. Alkadi and D. Carver, "A testing assistant for object-oriented programs," in *Aerospace Conference, 1998 IEEE*, vol. 4, Mar 1998, pp. 149–158 vol.4.

[15] A. Cernickins, O. Nikiforova, K. Ozols, and J. Sejans, "An outline of conceptual framework for certification of mda tools," in *Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modelling Theory-Driven Development, MDA and MTDD 2010, in Conjunction with ENASE 2010*, Athens, 2010, pp. 60–69.

[16] M. Asadi, M. Ravakhah, and R. Ramsin, "An mda-based system development lifecycle," in *Proceedings - 2nd Asia International Conference on Modelling and Simulation, AMS 2008*, Kuala Lumpur, 2008, pp. 836–842.

[17] F. Chitforoush, M. Yazdandoost, and R. Ramsin, "Methodology support for the model driven architecture," in *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, Dec 2007, pp. 454–461.

[18] V. Nikulsins and O. Nikiforova, "Adapting software development process towards the model driven architecture," in *2008 The Third International Conference on Software Engineering Advances*, Oct 2008, pp. 394–399.

[19] O. Nikiforova, V. Nikulsins, and U. Sukovskis, "Integration of mda framework into the model of traditional software development," *Frontiers in Artificial Intelligence and Applications*, vol. 187, no. 1, pp. 229–239, 2009. [Online]. Available: http://dx.doi.org/10.3233/978-1-58603-939-4-229

[20] P. Sanchez, J. Barreda, and J. Ocon, "Integration of domain-specific models into a mda framework for time-critical embedded systems," in *2008 International Workshop on Intelligent Solutions in Embedded Systems*, July 2008, pp. 1–15.

[21] A. Noyer, P. Iyenghar, E. Pulvermueller, J. Engelhardt, F. Pramme, and G. Bikker, "A model-based workflow from specification until validation of timing requirements in embedded software systems." Institute of Electrical and Electronics Engineers Inc., 2015, pp. 166–169. [Online]. Available: http://dx.doi.org/10.1109/SIES.2015.7185056

[22] P. M. S. Nazari and B. Rumpe, "Using software categories for the development of generative software," in *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, Feb 2015, pp. 498–503.

[23] S. Teppola, P. Parviainen, and J. Takalo, "Challenges in deployment of model driven development," in *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, Sept 2009, pp. 15–20.

[24] T. Seifert, G. Beneken, and N. Baehr, "Engineering long-lived applications using mda," in *Proceedings of the Eighth IASTED International Conference on Software Engineering and Applications*, 2004, pp. 241–246.

[25] E. Er and B. Tekinerdogan, "Modsel: Model-driven software evolution language," in *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*, M. Mernik, Ed., 2012, pp. 572–594.

[26] R. Bendraou, P. Desfray, M.-P. c. Gervais, and A. Muller, "Mda tool components: A proposal for packaging know-how in model driven development," *Software and Systems Modeling*, vol. 7, no. 3, pp. 329–343, 2008.

[27] B. Westfechtel, "Merging of emf models: Formal foundations," *Software and Systems Modeling*, vol. 13, no. 2, pp. 757–788, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10270-012-0279-3

[28] Y. Yu, T. T. Tun, A. K. Bandara, T. Zhang, and B. Nuseibeh, "From model-driven software development processes to problem diagnoses at runtime," vol. 8378 LNCS, 2014, pp. 188 – 207. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08915-7_7

[29] F. Mantz, G. Taentzer, Y. Lamo, and U. Wolter, "Co-evolving meta-models and their instance models: A formal approach based on graph transformation," *Science of Computer Programming*, vol. 104, no. 1, pp. 2–43, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2015.01.002

[30] A. Hovsepyan, R. Scandariato, S. Van Baelen, Y. Berbers, and W. Joosen, "From aspect-oriented models to aspect-oriented code? the maintenance perspective," in *AOSD.10 - 9th International Conference on Aspect-Oriented Software Development*, Rennes and Saint-Malo, France, 2010, pp. 85 – 96.

[31] T. Gottardi and R. T. V. Braga, "Model-oriented web services," in *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, March 2016, pp. 14–23.

[32] A. Forward, O. Badreddin, and T. C. Lethbridge, "Umple: Towards combining model driven with prototype driven system development," Fairfax, VA, United states, 2010. [Online]. Available: http://dx.doi.org/10.1109/RSP.2010.5656338