Journal of Universal Computer Science, vol. 19, no. 2 (2013), 186-206
submitted: 7/5/12, accepted: 31/12/12, appeared: 28/1/13

MDD Adoption in a Small Company: Risk Management and Stakeholders' Acceptance

Federico Tomassetti, Marco Torchiano
(Politecnico di Torino, Torino, Italy
federico.tomassetti, marco.torchianopolito.it)

Lorenzo Bazzani
(Trim srl, Torino, Italy
l.bazzanitrim.it)

Abstract: This article presents the knowledge and experience acquired trough the process of establishing MDD practices within a small Italian company. Special attention has been devoted to project constraints, perceived risks, and relative mitigation strategies. Moreover the article evaluates how the introduction of the MDD approach was received by different stakeholders. In particular a structured questionnaire was the instrument used to reveal and collect the perceptions by different roles involved in the MDD adoption process. The case study considered development of applications conforming to a prescriptive architectural framework, which addresses a complex multi-tier architecture; the solution aims at describing component composition while avoiding both repeating tasks and writing awkward configurations.
Key Words: Mode-Driven Development, Small Companies, Experience Report, Risk Management
Category: H.5.5

1 Introduction

The idea of developing software based on models has been part of several approaches and methodologies proposed in the last 20 years [Booch, 1991]. Model-driven development (MDD) leverages detailed models by generating automatically the application code [Jiang and Hu, 2008] or interpreting them at runtime [Zeng et al., 2005]. Such approach has been proposed both for general purpose development [Jacobson et al., 1999] and for specific domains (e.g., real-time applications [Selic et al., 1994]).

The MDD family of approaches has been widely adopted in the industry with different gradations and we have reports of the experience acquired in large organizations (e.g., [Baker et al., 2005; Hutchinson et al., 2011a]). As far as small companies are concerned little empirical or even just anecdotal evidence is available in the literature.

Our goal is to investigate the reasons for success or failure for MDD adoption in small industrial contexts. In particular we focused our attention on two aspects: (i) the potential risks and the associated mitigation strategies, (ii) the overall perception by the different stakeholders. In this work we are specifically considering the initial reception and the effects of deploying for the first time an MDD solution in a small company.

As far as the risks about adopting MDD are concerned, from a single case study it is not possible to derive a statistically based picture, although by relying on the managers experience we were able to get a clear idea of the perceived risks. It is important to focus on the perceived risks because they, even though possibly not empirically grounded, are typically responsible for early rejection of MDD approaches. Moreover while the promises of MDD are known, we could not find in the literature, despite of our best effort, an analysis of risks and motivations that cause companies to fail or abandon adopting MDD. A report about the perceived risks emerged while designing and implementing a MDD solution and the lesson learned about coping with them, apparently represent an important contribution to the field.

Furthermore the success of MDD practices adoption depends significantly on how different roles subjectively appreciate them, because participants' commitment represent a key factor. A questionnaire (see Table 1) administered to all the participants to the project allowed us to assess the appreciation and perception from different stakeholders' perspective.

The research we present here was conducted in the context of a technology transfer initiative involving the Softeng group at Politecnico di Torino and a small ICT company (Trim srl). The main goal of the collaboration was to design and implement a MDD solution based on a Domain Specific Language (DSL) and the supporting tools. In particular the DSL is used to describe the structure of enterprise applications, realized in conformance with a prescriptive reference architecture composed by seven different layers. The DSL allows designing how the different components are linked together, in order to realize the application, and defining the data structures exchanged between the layers. By using instruments from the Eclipse Modeling project we were able to realize the supporting tools in a short period and with a small effort (circa 64K lines of Java code developed in 5 months of full-time work by one of the academical authors of the paper). Moreover, we realized an integrated environment by combining such tools with the Eclipse Java IDE used at the company.

The main contributions of the paper are: (i) an examination of the main risks perceived by SMEs in adopting an MDD solution, (ii) an analysis of the risk mitigation strategies adopted, (iii) an investigation of the acceptance of the MDD solution by different stakeholders. The contents of this paper are an extension to a previous short article [Tomassetti et al., 2010].

In the remainder of the paper Section 2 first presents the project considered for the case study, then Section 3 gives an overview of the devised MDD solution, Section 4 presents a risk management perspective on the MDD adoption, Section 5 discusses how different stakeholders accepted the new approach, Section 6 summarizes and compares related works, and finally in Section 7 we draw the conclusions.

2 Case Study background

Today common enterprise applications comprise many layers [Singh et al., 2002]. At the bottom we almost always find a relational database and the code needed to deal with it (Data Access Objects or DAOs). At the top lays the presentation layer (i.e., JSPs when the chosen technology is Java). In the middle we can find a variable number of layers composing elementary services, dealing with transactions, playing the Model or the Controller roles in the Model-View-Controller (MVC) pattern. The prescriptive architecture (named Financial Value Chain or FVC) involved in this work is no exception being composed by seven different layers. Those layers are reported in Figure 1.

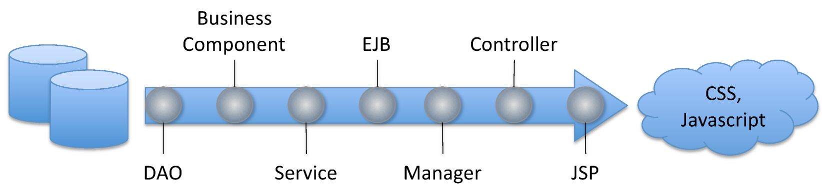

Figure 1: Layers of the case study architecture

Each layer brings into the project a different technology. In addition specific technologies are required to let the layers communicate while preserving abstraction. A common choice in a Java-based technology stack is to adopt Spring to manage dependency injection. Knowledge of several different technologies is required to deal with a such varying spectrum of libraries and frameworks. In a small company, counting a small number of developers, it could be difficult to either find or build this knowledge. There is a certain amount of code and configuration that does not implement any business logic but it is required just to realize the technological infrastructure, e.g., the composition of different elements and the communication between layers. We refer to it as architectural code as opposed to the business logic code. The architectural code and the related configuration often happens to be repetitive and as consequence error-prone [Sutcliffe et al., 1999].

2.1 Motivations for MDD

In the context of the project the company had to use a complex multi-layer architecture. Development for this architecture involved a large amount of repetitive and error prone tasks which demovatived the developers, moreover only developers with a large skillset were able to work on this architecture. For these reasons we decided to encapsulate technological details related to the architectural code in the transformations. In this way developers could focus on the business logic.

2.2 Project constraints

The design and implementation of the MDD solution we carried on was tailored for the specific needs of a company, therefore it had to confront with a set of different constraints, the most relevant being: i) since the software development project, where the approach had to be put in trial, involved several companies, it required the integration of components developed by 3-rd parties; ii) the client imposed quite strict rules for source code and configuration files that the generation phase had to adhere to; iii) the company management preferred the use of protected region with respect to other technical solutions.

Concerning the protected regions, the preference was mainly due to the fact it allowed to obtain generated code very similar to manually-written code. Such similarity, in case the MDD solution be abandoned, would make the fallback option to traditional development techniques easier and less expensive. An alternative to protected regions was to adopt specific strategies to separate generated code from manually written code (e.g., aspect oriented programming). These solutions while offering advantages would produce code different from the one produced by the standard development method. While for mature adopters of MDD the usage of protected-region could be considered an anti-pattern, in this case it was considered to be a pragmatic compromise increasing the confidence of the adopters.

2.3 Perceived risks

Due to the lack of previous experience in MDD at the company, we ended up debating a set of perceived risks, that are quite common among new MDD adopters. We conducted an open discussion session with the managers of the project aimed to elicit such risks; we wrote and refined a list of risks and eventually we validate the list back with the managers. As a result we could summarize four main risks:

R1) Tool rigidity: it is very likely that new development's best practices appear in the project lifetime, the tool could not be able to evolve to address them.

R2) Lack of developer adoption: the tool and the change in the development process could not be accepted by developers. They could feel like they are losing control or their skills are considered less valuable because of the MDD adoption.

R3) Solution lock-in: the company, at its first MDD experience, in case of failure could not be able to switch back to the previous (MDD-less) development process for the whole project or for the development of a single component with a limited effort.

R4) Application evolutionary inertia: the context of the application is doomed to rapidly evolve, both in terms of development technologies and as new domain and application requirements; while the company knows how to cope with such evolution using standard development, the fear is that MDD could make it more difficult. The request is that adopting the tool should not make reaction times longer but equal or possibly shorter than conventional development.

2.4 Scope of the solution

For the development of the lowest layer of the prescriptive architecture (FVC) an Object-Relational Mapping (ORM) was already in use. In particular the ORM used was iBatis and it was used with a companion tool, iBator, which analyzed the database schema and generated DAOs and configuration for iBatis.

The results obtained using iBatis were satisfactory so we decided to design an MDD solution that did not involve the database and DAO layers but instead was able to integrate with those layers as implemented using iBatis and iBator. Being the first MDD adoption trial at the company we wanted to focus only on the most problematic layers, where it was easier to obtain a significant improvement.

The company preferred to not adopt MDD for the presentation layer because they wanted their developers to maintain full control on this particular aspect of the application. Therefore the MDD solution was designed to cover five out of the seven layers of the FVC architecture.

The design of the MDD solution was driven first of all by pragmatism: while in an environment with experience in MDD usage we would have suggested a different approach in this case we had to look for a compromise between best practices and risk perceived. For this reason on one side we chose to focus on a subset of the layers and in particular on the ones where we could provide a more profound improvement, on the other hand we accepted the usage of protected regions which are considered an anti-pattern from more advanced users. In this particular case instead they helped to increase the confidence of the adopters because they had the possibility to insert custom code as it was needed. We advocate that the MDD approach has to be tailored on the base of the maturity of practitioners.

3 Case Study solution

The MDD approach adopted for this project is summarized in Figure 2: it is based upon three different model levels. The first one, the domain model, is defined by the user by means of a textual DSL. The second one, the intermediate model, is built reworking information extracted from the first model and from binary components using reverse engineering. Finally the third model is made up of the set of Java source files and XML configuration files generated.

The user defines a model of the architecture, using the DSL syntax. In the model he specify how components in the various layers are structured. Such a model is then translated by means of an automatic model-to-model (M2M) transformation to an intermediate model. From this intermediate model another transformation occurs, this time a model-to-text (M2T) transformation, producing as output a set of textual files: Java source code and XML configuration files.

The goal is to describe the domain model through a flexible DSL, so that the language can be used to describe a broad range of situations and at the same time be particularly concise resorting to common cases when it is possible. Conversely flexibility is not a goal for the intermediate model since it is used as the basis for the generation and it is never modified manually. Considering that the DSL has to be evolved to provide a better user-experience and the M2T transformation could have to be adapted to address technological changes the intermediate model is useful to somewhat absorbe those changes without them being propagated to the other extreme (i.e., a change to the DSL could in the worst case affect the intermediate model but not the M2T transformation, while a modification to the M2T transformation would not affect the DSL). Moreover the intermediate model is designed to contain completely specified information, which simplify the M2T transformation.

The solution devised combines tested practices and techniques into a customized toolset. It represents, we believe, a typical case of MDD adoption in a small company, so it lends itself as an exemplar case study.

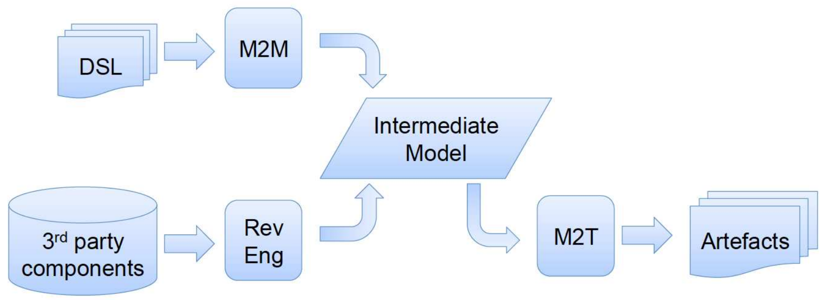

Figure 2: Models and transformations

3.1 Domain model

We opted to support a textual notation for our DSL and not a graphical one due to ease of development of tools with advanced features [Voelter, 2009].

One of our goals was to achieve a cost of switching as low as possible for the Java programmers already involved in the project; the use of a textual java-like syntax promised to be easier to learn than a graphical one. We considered that one of the main cost factors is the time needed for developers to learn the new environment and become proficient with it, consequently we decided for a notation as close as possible to the notations already used by the clients of the proposed MDD solution, who are mainly Java developers. In addition we considered concurrent engineering an important feature and while there are several tools supporting concurrent development for a textual notation (i.e., tools to execute differences analysis, automatic merge and so on), the adoption of a graphical notation would be more problematic [Leveque et al., 2009; Voelter, 2009].

The DSL language we designed is used to define the structure of a stack of components adhering to the FVC architecture and it is called FVCS (for FVC-Stack). Its syntax has been defined in agreement with two driving principles: i) the whole solution is designed to be immediately familiar to its intended users, which are Java programmers, so we designed the DSL syntax to resemble Java as much as possible. ii) The language resorts on some very common cases and for this reason a set of ”shortcuts” are provided. We report a small script to give the feeling of the language. In the project circa 1.0K lines of DSL code were written.

```
// import binary components
jar "path/binaryComponents.jar"
package it.trim {
   data in Abc {
      string name;
      date start;
      date end;
   }
   data out ADataOut {
      int total;
   }
   // implicitiyly refers to data in Abc
   businesscmp Abc {
      dao importedDao
      out ADataOut
   }
   service wrap Abc
   ejb MyEjb {
      Abc as "getAllXyz"
   }
   model MyModel {
      allfrom ImportedDao
      SomeOtherDao *;
   }
}
```

3.2 The intermediate meta-model

The role of the intermediate model as a basis for the final generation of the application model (represented by the generated artefacts) requires managing the different cases for every declaration kind and making explicit all the information provided implicitly at the domain model level through the DSL. The overall generation can be split in two transformation phases: i) a M2M transformation aiming to explicit information, ii) a M2T transformation which has to translate information to the specific target technology.

By adopting the intermediate meta-model we achieved greater flexibility and simplified the M2T transformation [Voelter, 2009]. The intermediate meta-model is composed by roughly one sub-meta-model for each component kind. We report in Figure 3 the portion of the intermediate meta-model used to represent one of the layer of FVC: Business Components.

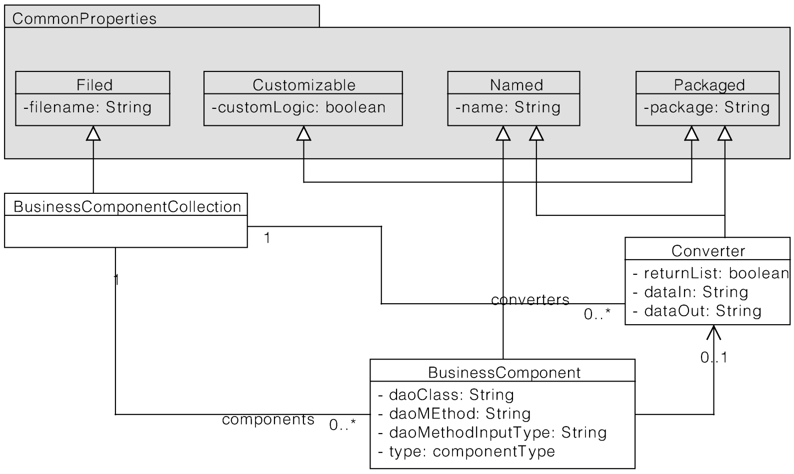

Figure 3: Excerpt of the intermediate Model

3.3 Generated artefacts

From the intermediate model by means of a M2T transformation two different kinds of artefacts are generated: Java source files and Spring XML configuration files. Java source files represent the implementation of the components defined using the DSL. In general a single component may give rise to one or more Java files. For some layers of the FVC architecture a configuration file may also be required to provide information relative to the whole set of components pertaining to that layer. Depending on the nature of the generated component it can be completely generated or it can be partially generated. Source files partially generated contains all the architectural code and let the user insert the custom logic into well-defined protected regions.

Configuration files define how different components are related and their scope is project-wide. There are three different configuration files, each of them contains configuration relative to a subset of the FVC layers. They are Spring XML configuration files: one is related to business components, the second one to services, the last one contains managers and controllers definitions.

Given the definition of a Business Component the application model consists of the following files:

- a Java class for the Business Component. It contains the class corresponding to the business component with the setter for the DAO and the field to hold its reference, the process method that could be completely specified (for wrapping Business Components) or just the skeleton containing a protected region in the general case;
- a Java class for the related Converter. The declared class contains methods to convert an ORM to a Data In and a Data Out to an ORM. ORMs are classes representing data in the database layer in a OO context;
- a portion of an XML configuration file for the Business Component and the related Converter providing information used by Spring to perform dependency injection.

3.4 Supporting tools

To implement our MDD solution we used Eclipse Modeling [Steinberg et al., 2008]. In particular we used these components of the bundle: the Eclipse modeling framework (EMF), Xtext, Xtend, Xpand.

EMF is a framework to define data models and meta-models, the structures of data models. In particular meta-models can be defined using XML, Java classes or modeling tools. The framework permits not only to define the structure of the models but also constraints that can be used to validate the model. EMF provides also a language called Ecore to define meta-models. Ecore is also a meta-model expressed in the means of itself. EMF is the technology that permits inter-operation between the different tools in the Eclipse Modeling Project. Every other tool in the project is able to operate with EMF models and meta-models.

XText is a DSL framework. It has to be fed with a DSL's syntax definition using an extended Backus-Naur Form (BNF). Starting from this syntax definition XText produces a parser, an EMF meta-model, and the skeleton of an editor. The parser is able to read a text file conforming to the DSL syntax provided and create from it a model referring to the generated meta-model. The editor will be specifically targeted at the DSL defined and will be provided as an Eclipse plug-in. Both the parser and the editor though fully working need to be customized by specifying validation rules, how to perform auto-completion, and many others refinements.

XPand is a template language able to consume EMF models to produce text files. XTend permits to create reusable definitions performing simple manipulations of EMF data. XTend is often used together with XPand to modularize some definitions that are reused many times in an XPand template.

One of the requirements was the possibility to re-use components developed with traditional techniques according to the FVC architecture but not using the designed

MDD facility we developed. The first reason is that we want to able to re-use components developed by other companies that did not adopt MDD techniques. This is important because Trim (the company involved in the case study) had to cooperate with other companies working on common projects. As second point in this way we are not tied up to develop every component of every level with the MDD solution. If a specific case arises that is hard to model with the current solution and the effort necessary to enrich the solution is a lot higher than the outcome that very specific component can be developed outside with traditionally techniques and then be referred in a FVCS file, i.e., to build a component in the upper layer. To satisfy the requirement the solution is designed to be able to re-use directly compiled components consisting of both a directory containing class files or a JAR.

The module devoted to reverse engineering is able to deal with both cases (a directory of class files or a JAR). This module analyses classes looking for implementation of components of a certain FVC layer by looking at the class name, at the interfaces implemented and at the superclass of the examined element.

4 Risks management

The development of the MDD solution for the FVC architecture, through a trial and error process, allowed us to better understand the key factors and the resulting benefits in developing and deploying such kind of solution in a small company with no prior experience about MDD. We first present lessons learned and later we discuss how they can be used to mitigate risks presented in Section 2.2.

4.1 Lessons learned

L1) need for intermediate level: we realized quite early in the project the necessity to introduce the intermediate meta-model. It proved to be very useful to prevent changes happening at one of the ends of the MDD solution to spread across the stages of the solution itself and affect the other end. This was important because during the development of the project we introduced many changes in the DSL syntax that were made as consequence of feedback from users. In most situations changes did not affect the intermediate meta-model but they were absorbed by the M2M transformation. Due to some changes we needed occasionally to slightly adapt the meta-model for some components but we never needed to adapt the M2T transformation due to changes in the syntax. The insulation role of the intermediate meta-model was also standing for changes coming from the technology side. As consequence both the syntax and the technology side were able to be improved and adapted separately so that we were free to let the language evolve to become more concise and at the same time we could adapt the way we produced the artifacts to meet the changing technical requirements.

L2) Convention over configuration: to adopt the principle known as ”convention over configuration” lead to concise scripts. Concise scripts are good for many reasons:

they are faster to write and they are faster to understand, which is also more important. They contain no redundant information so the reader can concentrate just on the particular cases, where something is not acting as usual.

L3) 3rd party integration: the ability to include components already developed in the MDD solution is critically important in industry environment. First of all it permits to not waste precedent investments and it permits a gradual transition from previous techniques to MDD. There is also another advantage: it brings confidence to be not locked in the MDD solution. If it is always possible to build components with traditional techniques and then integrate them in the models it will reduce greatly the risks involved in MDD adoption. These lessons confirm the findings presented in [Hermans et al., 2009], where several respondents wished the opportunity of being able to import preexisting models. It also permits a gradual adoption of the MDD approach, one of the important factors reported by [Hutchinson et al., 2011a].

L4) DSL flexibility: the MDD solution and in particular the DSL have to be flexible enough. In every technique involving modeling a certain amount of rigidity is assumed, MDD solutions needs to rely on archetypes and repeated patterns. Usually designers of MDD solutions tend to enforce too much, to limit what developers can do using the tools they create. A common pitfall is to envision in the small details how the whole solutions will be used and how the developers will have to implement the applications. This brings developers to feel unable to adapt their work tools to their way to execute the job. There is a tendency to consider developers as just another tool of the MDD solution. It proved to be more successful to think the other way around: developers are professionals and MDD designers just provide better tools that have to fit their needs and their own way to organize their work.

L5) Developer's involvement: developer's commitment is essential to obtain a successful and concrete adoption of the MDD solution. Our experience suggests that in order to obtain that developers have to be involved in the design phase considering their feedback as valuable. If developers feel that MDD adoption is going to be forced or that the solution is not flexible enough to adapt to their needs they are likely going to misuse it, causing to reduce or void the benefits provided.

4.2 Risk mitigation

The potential risk **R1 (tool rigidity)** has been mitigated by both L1 (need for intermediate level) and L2 (convention over configuration). The presence of an intermediate level allows adjusting the domain model and the code generation independently, thus achieving an high evolvability of the tool. Moreover the extensive use of conventions loosen the coupling among the modules of the tool, the result being a more maintainable system.

The possible **lack of developer adoption (R2)** was one of the main concerns during the development of the MDD approach. We learned that it can be mitigated in different ways: (i) relying on convention instead of configuring the utmost detail (L2) provides conciseness and relieves developers from writing repetitive patterns, (ii) the flexibility of the DSL (L4) allows the developers to easily work with the language, and (iii) the involvement of the developers (L5) in the design of the MDD solution ensured a high rate of adoption.

One of the fears of the company was switching to MDD, and then realizing it was not suitable for the project, and eventually ending up **locked in the solution (R3)**. The ability to integrate 3rd party components (L3) mitigated significantly this risk: it is alway possible to develop directly in Java any ”problematic” component and integrate it into the system.

An important competitive advantage in software development consists in being able to keep the pace with technological evolution. It was not clear whether MDD could jeopardize this ability of the company; the risk is to increase the inertia toward the evolution of the application to **adopt new technologies (R4)**. The existence of an intermediate level (L1) let the generation phase to evolve easily. Moreover the extensive use of conventions (L2) avoids over-specified models by introducing technology-dependent information, so mitigating the risk of evolutionary inertia.

5 Acceptance assessment

The reception of the MDD approach was evaluated by means of a questionnaire which was addressed to the different roles involved in the project who are employed at Trim. The goal was not to evaluate our solution but how participants in different roles reacted to MDD. The questionnaire was carefully developed by the two academical authors of this paper.

This section presents first the questionnaire, later the answers divided by topic are analysed.

5.1 Questionnaire definition

The survey was addressed to the Trim’s personnel involved in the project. The MDD solution was developed at the Politecnico di Torino with the constant feedback of the company. Later the tool was used and maintained at Trim. In particular there were five Trim’s workers involved and all of them accepted to fill in a survey tailored for the specific role they covered in the project. One of the workers is the manager (who is also a co-author of this paper) who contributed to develop the initial design of the MDD solution. A second one is a software architect with a broad and deep technical culture: he contributed with technical comments to the initial design and he was appointed with the duty to maintain and evolve the tools. The third worker is the project manager of the projects which used the tools since the first pilot and later in production. Finally there are two developers involved in those projects. The objectives of the questionnaire are to evaluate the results and in particular to ascertain possible deficiencies of the proposed approach, thus providing insights about the reception from the different point of views.

We adopted the participant-observer case study [Yin, 2002] because we needed the feedback from the personnel involved. Because we are not evaluating our approach to demonstrate its efficacy we consider the potential bias caused by the interest in the project to be not relevant.

Questions are divided into three groups which focus on different aspects of the project. The first one is about results evaluation (questions R1 through R11), the second one is about acceptance (questions A1 to A5) and finally the third one is about development process changes (questions P1 to P6). Overall we formulated 22 questions that are reported in the leftmost column of table 1. While the questionnaire was originally written in Italian, we present here the corresponding English translation. The questionnaire included both closed (C) and open (O) ended questions; the second column of the table describes the type of question. The close questions have been formulated in the form of assertions with which the respondent had to express his agreement according to a five point Likert scale [Oppenheim, 1992]: *Strongly disagree* (1), *Disagree* (2), *Neither agree nor disagree* (3), *Agree* (4), and *Strongly agree* (5). The typical encoding, used also in our analysis, is the integer number reported in parenthesis.

5.2 Discussion of responses

Table 1 reports, for each question, the answers provided by the different roles. Since the set of questions administered varied according to the role, there are combinations of role and question that have no response, such lack of answer is represented by void cells . For close-ended questions the cell contains the level of agreement encoded with an integer from 1 to 5 as shown above. The check mark, for open-ended questions, indicates that the question was administered to that role, who provided an answer (the full text of the responses is available in Appendix A). When the respondent did not provide any answer we report ”NA”. In particular developers used the possibility to not give an answer in two different cases because they were less involved into the project with respect to other roles and the changes due to the adoption of the MDD solution affected them marginally as we will report while analyzing answers on process changes. They were less involved because they were working mainly on the presentation layer, which is the layer excluded by the MDD solution. Anyway their role requires to use components generated by our solution and by means of specific questions we wanted to verify that its adoption is almost transparent to them.

5.3 Results evaluation

Typically when introducing modeling solutions there is the concern to cause rigidity to the development process: we rest reassured that the tool was in no-way a limit during the development (R1). The tool managed to reduce repetitive work (R2) although not to completely eliminate it. Anyway the reduction was sufficient to shorten significantly the development times (R3). The solution was considered also a tool useful to

Gr.		Question	O/C	Role M	Role SA	Role PM	Role D
Results	R1	The tool constituted a limit in any way during the development	C			1	1,1
	R2	Tool usage reduced repetitive work	C			4	4,4
	R3	Tool usage reduced development time	C	4	4	4	NA,NA
	R4	Tool usage reduced the number of defects in the developed applications	C	NA	3	3	NA,NA
	R5	The FvcGen approach supports the design phase and provides an overview of the system	C	5		4	
	R6	The approach improved maintainability	C		3	3	
	R7	The tool is easy to use	C			4	
	R8	The tool requires a quick learning phase	C			4	
	R9	The Fvcs DLS syntax is easy to learn	C			4	
	R10	Which aspects would you consider as critical to realize a similar project?	O		✓		
	R11	How could the solution be improved?	O			✓	
Acceptance	A1	Using the tool is professionally stimulating	C		4	4	4,4
	A2	Which were the major resistances to the adoption?	O	✓			
	A3	What could be done to favor the tool's acceptance?	O	✓			
	A4	Which are the most critical aspects for the adoption of a MDD solution in an organization?	O	✓			
	A5	How do you evaluate the adoption for the solution's users?	O		✓		
Process	P1	The adoption of FvcGen changed the way you work	C				1,1
	P2	The transition to the new development process has been quick	C	4			
	P3	The transition to the new development process has been very easy	C	3			
	P4	Which are the critical aspects for the adoption of FvcGen?	O				✓
	P5	Was the development project changed by the introduction of FvcGen? How?	O	✓			
	P6	Introducing FvcGen do the competences required for developers change?	O	✓			

Table 1: Answers to acceptance assessment questionnaire for different roles: M - Manager, SA - Software architect, PM - Project Manager, D - Developer. O/C = Open/Closed question. The numbers reported refer to a Likert scale ranging from 1 (Strongly disagree) to 5 (Strongly agree).

design globally the system and maintain an overview of the whole application (R5). As far as defect reduction is concerned (R4) the answers indicate that defects were not reduced by the adoption of the solution. First we note that respondents were mainly concerned with post-release defects, and we interpret the result as an indication that architectural errors were not reduced. We believe that low-level errors (typically those caused by misconfiguration or mistypes) quite common in hand-written code are eliminated by our solution. Such defects are show-stoppers and are routinely fixed before release causing a slow-down in the development and annoyance for developers. Though the company does not collect measures at this level and therefore no evidence is available to support this hypothesis. Apparently there is not a particular advantage in terms of maintainability (R6). This outcome can be explained by observing that the project has not entered the maintenance phase yet; therefore the answer to the specific question was a conservative one. There is a set of questions (R7, R8, R9) specifically addressed to the principal user of the tools (the Project Manager) and in particular of the FVCS language. The answers confirm that the tools and the language were practically usable and quite easy to learn. These characteristics were of course a key factor for the choice of Trim to adopt the solution. Finally analysing answers to the last questions we can get more general advice. The Software Architect considered the approach used (R10) to be correct and repeatable for similar applications, although he expresses doubts about the applicability of a similar approach to architectures that do not exhibit such a rigid multi-tier structure. The Project Manager instead points out that the solution could be improved (R11) by making it more easily deployable and reducing dependencies issues. As a side note he indicated, also during the development of the project, the necessity to increase flexibility e.g. by making the destination of generated artefacts for different components and the definition of Java package names directly configurable.

5.4 Acceptance

Because we consider the solution acceptance from all roles working at the company an absolute key-factor for the success of the project we realized a set of questions specific for this topic. We first asked the technical figures involved to find out if the considered this experience as professionally interesting (A1). We obtained positive answers by all roles. Then we asked the Manager and the Software Architect for their considerations. The Manager considers an hindrance to adoption (A2) the fact the developers tend to dislike technical solution proposed by external subjects. In this case they accepted it because they could see concrete benefits early. As a second point some issues in the tools deployment to different Eclipse installations were considered annoying. He suggested in order to favor the adoption (A3) to organize lessons on the use of the solution. Moreover we asked for advices on the adoption not specifically tailored to this experience. The Manager affirmed that the MDD solution should be to reused across other projects (A4). He specify that could be not always possible because, while the customer of this project would appreciate the adoption of a similar solution, other customers could ask the company to use a specific development approach not based on the MDD solution. The Software Architect considered the reaction of the solution users (A5) positive but he suggested that it could be subject to personal tastes; he elaborated further that while the people involved in the project appreciated it, it could be the case that developers prefer to develop manually most of the code in order to maintain strict control.

5.5 Process changes

We also wanted to evaluate how the development process was affected by the introduction of MDD in a small company for its first time and which were the changes. We found that transition was quite smooth: quick (P2) and with no particular difficulty (P3). As far as developers are concerned, since their work is essentially centered in the presentation layer, the adoption of the MDD approach is almost transparent to them (P4) and therefore their job is not affected (P1). A major difference emerged for the Project Manager: he is now able to do on his own a greater amount of work; in particular he can develop, by means of the tools, the whole back-end of the applications (P5). This is partially justified by the fact that he is "shielded" by many technical details related to the specific technologies involved and as result he is less concentrated on technological aspects and can spend more attention on the business logic. Finally, prerequisite on technical competences for the developers involved in the project can be relaxed (P6). This represents an important benefit for the company that can hire workers that are not necessarily expert in every single technology of the architectural stack.

6 Related work

Insights on the diffusion of MDE are reported by a large study from Hutchinson et al. [Hutchinson et al., 2011b] conducted by means of questionnaires and interviews. Another study about MDE diffusion (limited to Italian companies) was co-authored by the academical authors of this paper [Tomassetti et al., 2012; Torchiano et al., 2011]. Those studies are interesting to understand the phenomenon at large but do not reach the level of detail that case studies and experience reports permit to achieve.

While the literature includes several experience reports on MDD adoption, most if not all of them concern studies in the context of large industrial setups and companies with a medium to long experience in the field.

A notable example is Baker et al. [Baker et al., 2005] who report on the competencies developed at Motorola after 15 years since the adoption of MDE. In their experience the major obstacles in adopting MDE stem to the lack of a well-defined process, lack of necessary skills and inflexibility in changing the existing culture. Another account from a 5-years project at the same company can be found in [Foustok, 2007].

Fleurey et al. [Fleurey et al., 2007] report on the 10-years' experience with MDE developed at Sodifrance; the focus is on migration projects, where the benefits w.r.t. conventional techniques can be observed after an initial period, e.g. the first code could be delivered only after 10 months from project's beginning. In addition they present a cost-benefit analysis and suggest the presence of profitability threshold in terms of project size.

Hen-Tov et al. [Hen-Tov et al., 2009] describe a project with enterprise software; while the software category is similar to our case study, their approach requires an initial development effort of 10 man-years.

Hutchinson et al. [Hutchinson et al., 2011a] report lesson learned from adoption of MDE in three large multinational companies (a printer company, a vehicle manufacturer and a manufacturer of electronic systems). They conclude that important enablers for the succes of MDE in those large companies were: adopting a progressive and iterative approach, obtaining organizational commitment, motivating users, organization flexibility from the whole company and the presence of a business focus motivating the adoption of MDE.

The above reports concern MDD solutions that require a very long setup time; one article from MacDonald [MacDonald et al., 2005], which analyzes development through MDD of a component for a legacy systems, describes an approach requiring a low initial development effort. We think that small-companies have to choose a different way to MDD adoption than large ones and it seems to be neglected by the current research trend. For this reason we try to adopt some research approaches used in large-companies and adapt them to a case study undertook in a small one.

There are other works that studied MDD through a survey as we did. For instance, a survey on the success factors of DSLs adoption conducted on a large set of projects spread among many different companies [Hermans et al., 2009]. The scale factor difference between our work and that one was a crucial aspect affecting the way we designed our questionnaire and how we analyzed the results. Due to reduced number of people involved in the project we could not perform statistical analysis. Moreover our work differentiate the people involved in the survey by their roles while [Hermans et al., 2009] does not. Another work is from Staron [Staron, 2006] who proposed a questionnaire to personnel of two companies considering MDD for adoption, one having already undertook a pilot, the other not yet. It emerges that the three most important factors influencing the decision for adoption are i) availability of modeling tools, ii) cost of introducing the modeling technique to the process, iii) cost of creating models during software development. Once again companies considered are really large organizations. Shirtz et al. [Shirtz et al., 2007] reports considerations about successful way of convincing management to adopt MDD, their considerations on this topic are not the central part of the paper and are anyway related just to large companies.

Finally a review of experiences on MDE applications from Mohagheghi and Dehlen [Mohagheghi and Dehlen, 2010] contains considerations on effects on code quality (but not supported by data) and productivity while we are trying to perform an analysis of the effects on the development process and reactions by different roles involved.

7 Conclusions

In this paper we discussed the adoption of a MDD solution in a small-company with no prior experience with such techniques. We described the specific context and reported and motivated the principles applied to design the solution. In particular we stressed the importance to deliver high-quality supporting tools in order to guarantee an acceptable productivity. In a small company it is particularly important to achieve early benefits as result of small initial investments. In order to attain such goal we strived to (i) build an environment as familiar as possible for the prospective users and (ii) avoid any rigidity that could hinder the reactivity to requirements or process changes, which is one of the competitive advantages of small companies in respect to larger ones.

We investigated the perception of the solution by the personnel involved in the project. We could confirm the good results obtained and we gained considerable insight about deficiencies and ways to improve our approach. The respondents emphasized two main key success factors: very high-quality tools and flexibility. While these aspects were considered satisfying in this case study, they are so important to need still more attention.

Given the cross-sectional nature of our study, we did not consider long term effects. In particular no evidence could be found regarding the impact of the proposed MDD solution on maintainability and defect reduction: we plan to investigate more these aspects in future research.

As future work, we plan to evaluate again MDD approaches in small companies in the context of different kinds of applications investigating more in depth process changes.

Acknowledgments

We want to thank the personnel at Trim who took time to answer the questionnaire. We are also grateful to the anonymous reviewers for their comments and suggestions which helped us to improve this paper.

References

[Baker et al., 2005] Baker, P. ; Loh, S. ; Weil, F.: Model-driven engineering in a large industrial context - motorola case study. In: *MoDELS '05*, Springer-Verlag, 2005, pp. 476 –491

[Booch, 1991] Booch, Grady: *Object Oriented Design With Applications*. Addison-Wesley, 1991

[Fleurey et al., 2007] Fleurey, F. ; Breton, E. ; Baudry, B. ; Nicolas, A. ; Jezequel, J.: Model-Driven Engineering for Software Migration in a Large Industrial Context. In: *MoDELS '07*, Springer-Verlag, 2007, pp. 482 –497

[Foustok, 2007] Foustok, M.: Experiences in Large-Scale, Component Based, Model-Driven Software Development. In: *Systems Conf., 2007 IEEE*, april 2007, pp. 1 –8

[Hen-Tov et al., 2009] Hen-Tov, A. ; Lorenz, D.H. ; Pinhasi, A. ; Schachter, L.: ModelTalk: When Everything Is a Domain-Specific Language. In: *Software, IEEE* 26 (2009), july-aug., No. 4, pp. 39 –46. – ISSN 0740-7459

[Hermans et al., 2009] Hermans, Felienne ; Pinzger, Martin ; Deursen, Arie van: Domain-Specific Languages in Practice: A User Study on the Success Factors. In: *MoDELS*, 2009, pp. 423–437

[Hutchinson et al., 2011a] Hutchinson, J. ; Rouncefield, M. ; Whittle, J.: Model-driven engineering practices in industry. In: *Software Engineering (ICSE), 2011 33rd International Conference on*, may 2011, pp. 633 –642. – ISSN 0270-5257

[Hutchinson et al., 2011b] Hutchinson, J. ; Whittle, J. ; Rouncefield, M. ; Kristoffersen, S.: Empirical assessment of MDE in industry. In: *Software Engineering (ICSE), 2011 33rd International Conference on*, may 2011, pp. 471 –480. – ISSN 0270-5257

[Jacobson et al., 1999] Jacobson, Ivan ; Booch, Grady ; Rumbaugh, James: *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999

[Jiang and Hu, 2008] Jiang, Derong ; Hu, Jianfeng: Research of Model-Based Code Automatic Generation of Management Systems. In: *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, oct. 2008, pp. 1 –4

[Leveque et al., 2009] Leveque, Thomas ; Estublier, Jacky ; Vega, German: Extensibility and Modularity for Model Driven Engineering Environments. In: *ECBS '09*, IEEE Computer Society, 2009, pp. 305–314. – ISBN 978-0-7695-3602-6

[MacDonald et al., 2005] MacDonald, A. ; Russell, D. ; Atchison, B.: Model-driven development within a legacy system: an industry experience report. In: *Software Engineering Conference, 2005. Proceedings. 2005 Australian*, 29 2005, pp. 14 – 22. – ISSN 1530-0803

[Mohagheghi and Dehlen, 2010] Mohagheghi, Parastoo ; Dehlen, Vegard: Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. In: *Model Driven Architecture Foundations and Applications* Ed. 5095, Springer Berlin / Heidelberg, 2010, pp. 432–443. – URL `http://www.springerlink.com/content/n008724135474004/`. – ISBN 978-3-540-69095-5

[Oppenheim, 1992] Oppenheim, A. N.: *Questionnaire Design, Interviewing and Attitude Measurement*. London : Pinter, 1992

[Selic et al., 1994] Selic, Bran ; Gullekson, Garth ; Ward, Paul T.: *Real-Time Object-Oriented Modeling*. John Wiley & Sons, 1994. – 560 p

[Shirtz et al., 2007] Shirtz, Dov ; Kazakov, Michael ; Shaham-Gafni, Yael: Adopting Model Driven Development in a Large Financial Organization. In: *Model Driven Architecture- Foundations and Applications* Ed. 4530, Springer, 2007, pp. 172–183. – URL `http://www.springerlink.com/content/j3840236386r8074`. – ISBN 978-3-540-72900-6

[Singh et al., 2002] Singh, Inderjeet ; Stearns, Beth ; Johnson, Mark: *Designing Enterprise Applications with the J2EE(TM) Platform (2nd Edition)*. Prentice Hall, 2002

[Staron, 2006] Staron, Miroslaw: Adopting Model Driven Software Development in Industry A Case Study at Two Companies. In: *Model Driven Engineering Languages and Systems* Ed. 4199, Springer, 2006, pp. 57–72. – URL `http://www.springerlink.com/content/05n8w2176843p862/`. – ISBN 978-3-540-45772-5

[Steinberg et al., 2008] Steinberg, Dave ; Budinsky, Frank ; Paternostro, Marcelo ; Merks, Ed: *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2008

[Sutcliffe et al., 1999] Sutcliffe, Alistair ; Galliers, Julia ; Minocha, Shailey: Human

Errors and System Requirements. In: *Requirements Engineering, IEEE International Conference on* (1999), pp. 23. – ISSN 1090-705X

[Tomassetti et al., 2010] Tomassetti, Federico ; Torchiano, Marco ; Bazzani, Lorenzo: Applying MDA to complex multi-tier enterprise architectures. In: *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA : ACM, 2010 (ESEM '10), pp. 61:1–61:1. – URL http://doi.acm.org/10.1145/1852786.1852863. – ISBN 978-1-4503-0039-1

[Tomassetti et al., 2012] Tomassetti, Federico ; Torchiano, Marco ; Tiso, Alessandro ; Ricca, Filippo ; Reggio, Gianna: Maturity of software modelling and model driven engineering: A survey in the Italian industry. In: *Evaluation Assessment in Software Engineering (EASE 2012), 16th International Conference on*, may 2012, pp. 91 –100

[Torchiano et al., 2011] Torchiano, Marco ; Tomassetti, Federico ; Ricca, Filippo ; Tiso, Alessandro ; Reggio, Gianna: Preliminary Findings from a Survey on the MD State of the Practice. In: *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement*. Washington, DC, USA : IEEE Computer Society, 2011 (ESEM '11), pp. 372–375. – URL http://dx.doi.org/10.1109/ESEM.2011.51. – ISBN 978-0-7695-4604-9

[Voelter, 2009] Voelter, Markus: Best Practices for DSLs and Model-Driven Development. In: *Journal of Object Technology* 8 (2009), No. 6, pp. 79–102. – ISSN 1660-1769

[Yin, 2002] Yin, Robert K.: *Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series, Vol 5*. 3rd. Sage Publications, Inc, December 2002. – URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0761925538. – ISBN 0761925538

[Zeng et al., 2005] Zeng, L. ; Lei, H. ; Dikun, M. ; Chang, H. ; Bhaskaran, K. ; Frank, J.: Model-driven business performance management. In: *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*, oct. 2005, pp. 295 –304

A Appendix - Responses to open ended items

Item Role	Question Response
R10	*Which aspects would you consider as critical to realize a similar project?*
SA	The current version is suitable for the development of multi-tier applications, therefore for such kind of projects FvcGen does not bring any criticality. On the other hand, if the architecture were less layered then the use of the instrument *as-is* could introduce some issues. A further re-engineering of the tool could be required.
R11	*How could the solution be improved?*
PM	Right now it appears to be too constrained both in terms of generated code and used libraries. It was very hard to make it compatible with current work instruments (e.g. different versions of Eclipse and Ant)
A2	*Which were the major resistances to the adoption?*
M	In general developers are not likely to accept technical decisions taken by others. In this case, though, who used the tool immediately gained real benefits. The problem, which could limit its adoption in future projects, is the complexity of the installation procedure, due also to the several version of Eclipse.
A3	*What could be done to favor the tools acceptance?*
M	No idea about the tools. Though we could organize training sessions.
A4	*Which are the most critical aspects for the adoption of a MDD solution in an organization?*
M	For sure process and technologies ought to be standardized. For a company like ours, this is very difficult since customers often impose their techniques. In a context such as our current banking customer I see the adoption of this technique as very feasible without particular criticalities (but those related to natural resistance towards change).
A5	*How do you evaluate the adoption for the solutions users?*
SA	Positively, although it is heavily dependent on the person that use it (some are more productive when they work according their habits, others are more open to innovation)
P4	*Which are the critical aspects for the adoption of FvcGen?*
D	I did not use it directly (*both developers replied in this way*).
P5	*Was the development project changed by the introduction of FvcGen? How?*
M	Changing the process was exactly the goal of this collaboration. In particular we wanted the team to be less focused on the technological aspects and closer to the business issues. The solution actually collapsed on a single person the development of the generated components.
P6	*Introducing FvcGen do the competences required for developers change?*
M	Yes. All the generated portion hides precisely the technological complexity.

Table 2: Responses to open ended items (translated from Italian into English)