Accelerating the Development of Health and Social Services Systems Through Model-Driven Engineering

Two Case Studies

Ismaïl Khriss
Université du Québec à Rimouski
300, allée des Ursulines
Rimouski (Québec) G5L 3A1, Canada
1 (418) 723-1986 ext. 1455
ismail_khriss@uqar.ca

André McKibben
CISSS de la Gaspésie - PETRAAS
124 B, route 132 Ouest
Percé (Québec) G0C 2L0, Canada
1 (418) 782-2270 ext. 50343
andre.mckibben.chchandler@ssss.gouv.qc.ca

***Abstract*— The Integrated Center for Health and Social Services in Gaspesia, Québec (CIHSSG) needed new software systems in order to better serve its customers and offer new services. However, as a small organization, resources, both human and financial, are limited. This is why CIHSSG mandated us to come up with a solution that enables the development and acquisition of software systems with the lowest possible cost. We proposed the development of the new systems through the adoption of a model-driven engineering approach (MDE). In this paper, we present the case studies, our MDE approach and the results which show how our approach has accelerated the development of the new systems while reducing the cost of their development and maintenance.**

Keywords—model-driven engineering, model-driven architecture, enterprise patterns, design patterns, software framework, multi-layered architecture.

Type of the submission: Full/Regular Research Papers

Acronym of the symposium: CSCI-ISHI

I. Introduction

Organizations rely heavily on software systems to increase productivity of their employees and streamline their business processes. The Integrated Center for Health and Social Services in Gaspesia, Québec (free translation, CIHSSG in short), a regional Ministry of Health agency, does not come out of this reality. In order to better serve its customers and offer new services, CIHSSG needed new software systems. However, as a small organization, human and financial resources are limited. This is why CIHSSG mandated us to come up with a solution that enables the development and acquisition of software systems with the lowest possible cost. We proposed the development of the new systems through the adoption of a model driven engineering approach (MDE). The latter refers to the systematic use of models as primary engineering artifacts throughout the engineering lifecycle [1]. The Model-Driven Architecture (MDA) [2], an initiative proposed by the Object Management Group (OMG), is an example of MDE. As MDE, MDA is an approach of development where the principal elements are the models representing the various aspects of the software system at various levels of abstraction. The development process is based on successive models transformations resulting in the source code [2]. MDA distinguishes between two important levels of abstraction models: the platform independent model (PIM) and the platform specific model (PSM). The PIM describes the system independently of a platform of implementation. The PSM represents the elements of a PIM in a point of view, specific to an implementation platform. The interesting aspect of the MDA initiative is that the PSM can be obtained by a model to model (M2M) transformation of the PIM. This kind of transformation requires that the knowledge of the implementation platform is captured in a model called a platform description model (PDM). The source code of a system is easily obtained by a model to text (M2T) transformation of the PSM. Another case could be to obtain the source code directly from the PIM after application of a PDM through a M2T transformation. In this case, the PSM is obtained from the source code through a text to model (T2M) transformation.

Two new systems were developed through the adoption of an MDE approach. The two systems belong to two social service programs, both very different from the other. These programs are under the authority of CIHSSG. It may be relevant to say that, nevertheless the differences between the programs; they are managed by the same person, the co-author of the current paper, an employee of CIHSSG.

This paper is organized as follows. Section 2 presents the case studies (the two new systems). Section 3 gives an overview of our MDE approach. Section 4 presents some results that show our approach has accelerated the development of the new systems, while reducing the cost of their development and maintenance. Section 5 discusses some related works. Finally, section 6 provides some concluding remarks.

II. The Case Studies

In this section, we present the two social programs where originated the two new systems.

A. The Assessment, Treatment and Research Program For Authors of Sex Agresssion (ATRPASA)

The first program is a national program where persons convicted to sentences lower than two years for sexual crimes

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.25

are sent for assessment and treatment of their behavior. Delivering a program in such a format had never been done before. This is a short (26 weeks) but intensive and highly specialized program offered to persons coming from the 15 different correctional facilities in the province of Québec to participate in the program. Gathering data on offender like crime or victims' characteristics, attainment of treatment goals to be eventually associated with recidivism or non-recidivism, is an important goal of this program designed for this very heterogeneous population.

Issues of rapid screening in the correctional facilities of persons motivated for treatment, transferring them, validating the motivation at the arrival to the program and, on that basis, running assessment and treatment for a very short period of time while measuring the attainment of treatment goals are extremely important. Issues of making these data and information on this process available to the Ministry of Health and Correctional Services (under the authority of the Public Safety Ministry) are also significant.

The first participants arrived for the program on the 17th day of May 2009. The general characteristics and needs of the population had been identified and helped to define the process through which the participants would pass. Five phases were identified. For each phase, specific aspects of functioning of participants had been formalized on scales and screen tests to support the decisions to be made (refusal, admission or interruption of the participation) at each specific phase to ensure that the treatment would be offered and completed by participants who would collaborate correctly and attain the treatment goals. The actors involved in each phase (treatment operators, treatment director, correctional authorities, participant), the documents necessary to gather and transmit information between actors and, finally, the moments and the lines of communication had been built and were available.

B. General Psycho-Social Services program (GPSS)

The second program is a classic social services program based in the community, named General Psycho-Social Services (GPSS). Members of the community are free to come to receive help, support or counseling in order to address different personal, familial or social problems such as mood disorder resulting of the death of a close relationship, discipline problems with a child, anxiety associated to unemployment.

The main problem was that the methods used in the program to achieve the tasks and theoretical goals of this kind of programs can be qualified of highly-unstandardized methods. Professionals of different social sciences disciplines, of different ages, sex and experience, meet members of the community coming with various problems. The clinical data are commonly written on pen-and-paper notes from a narrative method. No prioritization is defined and no agreement is made about the relative value and the extent of information. The professionals, according to their subjectivity, consign data, not always relevant, presented without any prioritization. Significant amounts of data randomly constituted, highly subjective and “impressionist”, are introduced into the clinical files.

The planning of the interventions is then compromised and run through the same process. The problem itself of the person, the risk and protection factors, the goals of the intervention, the means available to the person and the strategies used by the professional to support the attainment of the goals are not standardized. The objective measurement of the person’s evolution on any aspect of his functioning is thus corrupted.

These numerous and deep bias lead to « errors » from a methodological standpoint, fragmentation on a clinical level and failures of the organizations to achieve their tasks. In reality, numerous notes are absent from the files or unreadable. Intervention plans are often « copy-paste versions » of anterior plans or irrelevant.

For these reasons, it has been decided to formalize on a continuum, ranging in levels from 1 to 5, four specific axis of functioning of the persons coming to the program.

It is important to specify that, before the implementation of the system, inter-judges agreement had been verified. It has been demonstrated that, using these scales, the professionals had a statistically significant tendency to score the axis equally.

It became reasonable to believe that « priority clinical constructs » were measured or, in other words, that what is to be changed to improve the situation by making the risk factors lower and improving the protection factors, enabling the person to move from an initial to a superior level, was formalized through objective indicators.

On this basis, consensuses have been established among professionals. Objectives, means available to the person, strategies to be used by the professionals to support positive changes have been specified and computerized. The system generates, from the score attributed by a professional to a person on one axis or another, a list of these objectives, means and strategies that are, by definition, less subjective and more consensual. This list is the intervention plan, presented on a formulary, to be signed by the person, according to the Law on Health and Social Services.

In summary, the measurement of the « clinical constructs » is more reliable and can be repeated on each axis to verify the evolution of individuals’ clinical situations and, by extension, the efficiency of the intervention plan. Fields open to research are numerous and the communication of clinical data is made easier. The system is also a useful instrument of training for new employees and supports the development of common language and work methods.

III. OUR MODEL-DRIVEN ENGINEERING APPROACH

A new system, called *ePetraas*, was developed to support the first program. Another new system, called *SRADC*, was implemented for the second program. We have adopted MDA, an MDE approach, for the development of the two new systems. Recall that MDA distinguishes between two important levels of abstraction models: the platform independent model (PIM) and the platform specific model (PSM). The PIM describes the system independently of a platform of implementation. The PSM represents the elements

of a PIM in a point of view, specific to an implementation platform. The PSM is easily obtained from the source code through a T2M transformation. Several UML (Unified Modeling Language [3]) modeling tools can perform this kind of transformations. The source code is obtained from the PIM through a M2T transformation. This kind of transformation requires that the knowledge of the implementation platform is captured in a model called a platform description model (PDM).

We have also decided to use the same implementation platform for the new systems. Figure 1 shows our MDE approach for the construction of the new systems.

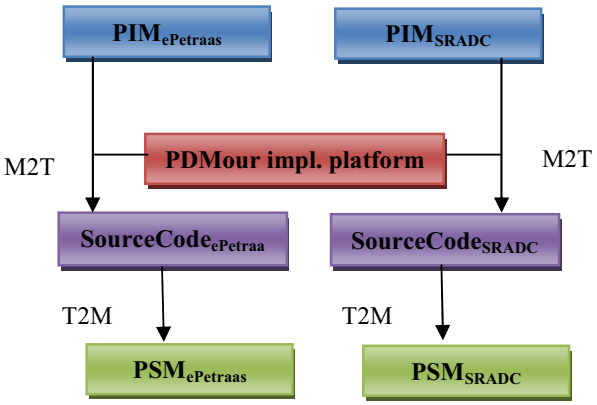

Fig. 1. Our MDE approach

In the rest of this section, we describe the components of our approach.

A. The platform independent models

Figure 2 shows an excerpt of the PIM of the system *ePetraas*. The PIM is captured in a UML class diagram and shows some aspects related to data reported by the infirmary. A participant has a medical card, which is regularly updated. Periodically, a staff member − a nurse − takes the vital signs of the participant. In case of hospitalization, a nurse fills out a hospitalization form that will contain some important information of the participant before and after hospitalization. When a participant leaves the prison of Percé for another prison, a medical transfer sheet is filled with relevant information on participant health.

Figure 3 shows an excerpt of the PIM of the system *SRADC* described in a UML class diagram. On a given date, a customer is assigned to a service program. A member of the therapeutic staff will play the intervening pivot of the customer. Several members of the therapeutic staff may be involved during treatments of the customer. A staff member may supervise other staff members.

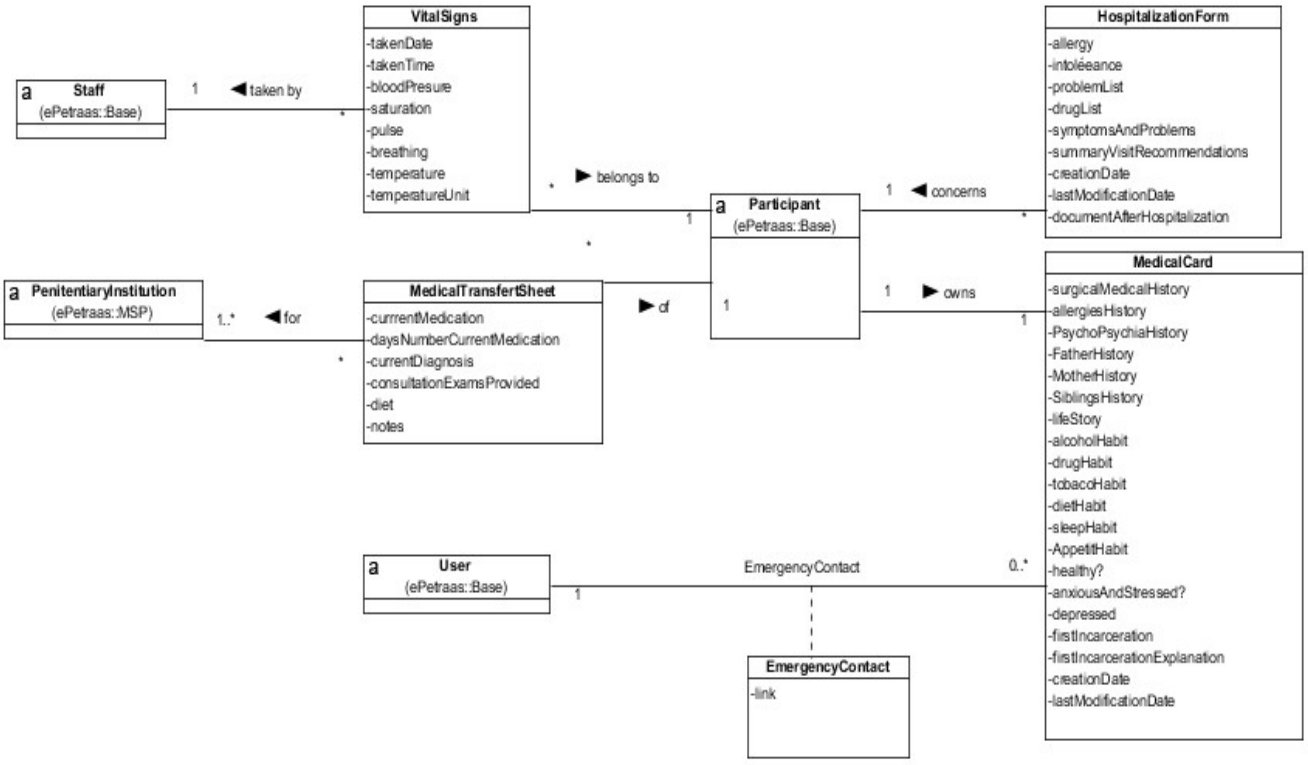

Fig. 2. Excerpt of the PIM of the ePetraas system (operations are not shown)

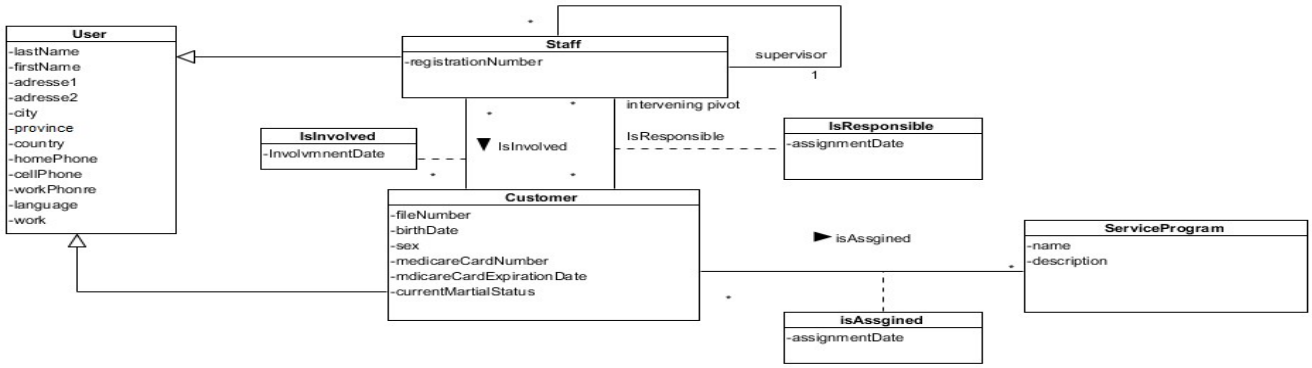

Fig. 3. Excerpt of the PIM of the SRADC system (perations are not shown)

B. The platform description model

We developed an in-house software framework to be our implementation platform for the two new systems. This framework uses a multilayered architecture: presentation, domain logic and data access.

The presentation layer is responsible for the delivery and formatting of information to the user. The domain logic captures the real-world business rules that determine how data can be created, displayed, stored, and updated. The data access layer (DAL) simplifies the access to data stored in databases. The layers have been implemented by using a set of enterprise patterns. Recall that an enterprise pattern is a design pattern which gives a reusable solution to a commonly occurring problem within a given context in the development of enterprise systems [4]. Examples of enterprise patterns are *Front Controller* and *Data Mapper*. The Front Controller pattern provides a centralized entry point for handling requests of users in web applications [4]. The Data Mapper pattern is a layer of correspondence that moves data between the database and the objects in the domain logic layer while keeping them independent of each other [4].

Our in-house framework was described in a platform description model. The latter consists of a set of XSLT templates which performs the M2T transformation from a system's PIM to its source code. Recall that XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents, or other formats such as HTML or plain text [5]. Each XSLT template captures a design concern in our framework. For instance, an XSLT template generates classes for the domain logic layers while other templates generate classes for the DAL by implementing the Data Mapper pattern. The use of XSLT as a language for transformations is possible since an UML model (such our PIMs and PSMs) can be captured in a XML format using the standard language XMI (XML Model Interchange [6]).

C. The platform specific models

Figure 4 shows an excerpt of the PSM of the system *ePetraas* captured in a UML class diagram. This excerpt shows the application of the Data Mapper enterprise pattern in our implementation platform. All domain logic classes inherit from a new class called *DomainObject*. This new class is the result of the use of the design pattern *Identity Field*. The *ID* attribute is used to maintain identity between an in-memory object and a database row [4]. Another pattern called *Foreign Key Mapping* is used to implement associations between domain logic classes.

The application of the Data Mapper results also in two kinds of new components: a set of interfaces and a set of classes (their names have the key word *Mapper* as a suffix). The interfaces give the abstract operations needed when accessing to a database. Those operations are implemented in the mapper classes. Their code is dependent to a certain technology. That is why those classes are in a package called *MSSQL* (for Microsoft SQL Server). The mapper classes have access to a façade class called *Database*. This class has a generic code to perform any SQL CRUD (Create, Read, Update and Delete) request.

Finally, note the presence of *DALFactory* and *MSSQLDataFactory* classes. These classes are the result of the application of the *Abstract Factory* design pattern [7].

Figure 5 shows a small excerpt of the PSM of the system *SRADC* expressed in a UML class diagram. This excerpt shows the application of the Front Controller enterprise pattern in our implementation platform. This pattern uses the *Command* design pattern [7]. The handler receives the HTTP Post or Get request from the Web server and retrieves relevant parameters from the request. The handler uses the parameters from the request to choose the correct command and then it transfers the control to the command for processing [4].

IV. THE RESULTS

The implementation platform uses the following technologies:

- C#, ASP.Net and JavaScript as programming languages.
- The database server is Microsoft SQL Server.

The size of the system *ePetraas* is 38851 lines of code (LOC) captured in 680 classes (see Table I). 19169 LOC

(respectively, 603 classes) were automatically generated which yield to 49.33% (respectively, 88.67%) of the total of the system.

The size of the system *SRADC* is 10231 lines of code (LOC) captured in 231 classes (see Table I). 5205 LOC (respectively, 197 classes) were automatically generated which yield to 50.87% (respectively, 85.28%) of the total of the system.

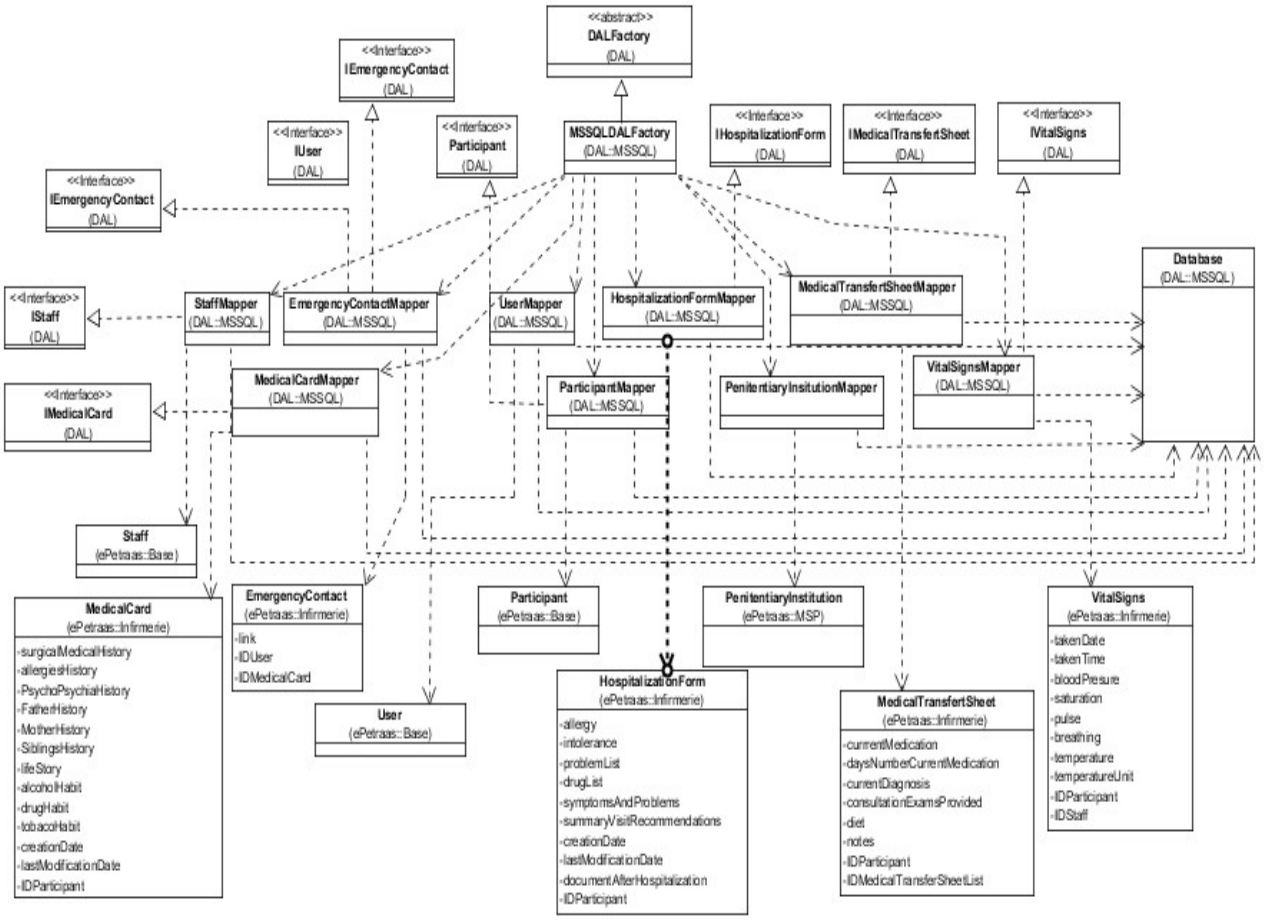

Fig. 4. Excerpt of the PSM of the ePetraas system (operations are not shown)

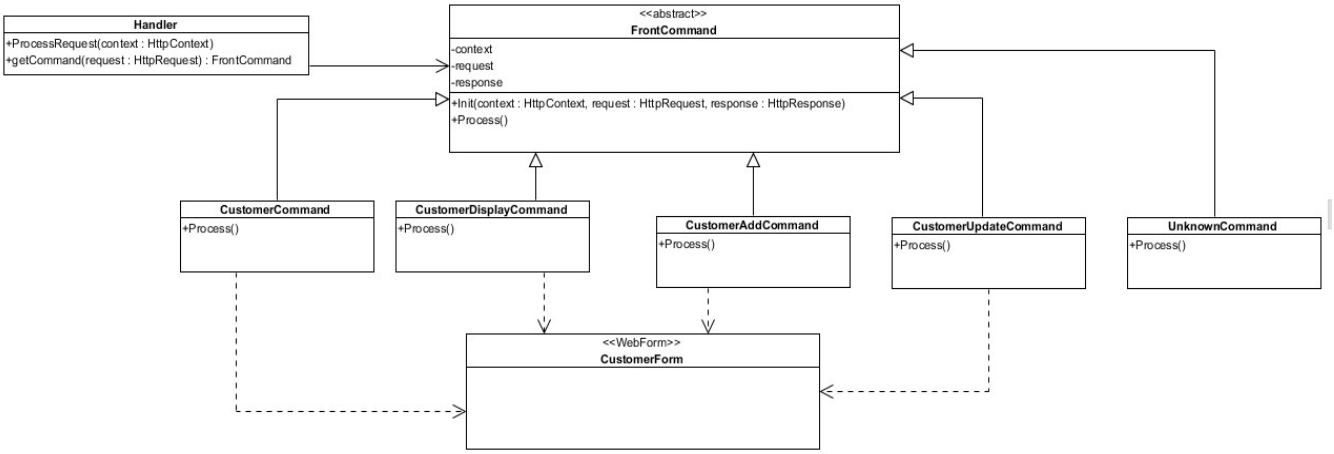

Fig. 5. Excerpt of the PSM of the SRADC system

TABLE I. SOME STATISTICS

	# of lines of code (LOC)	# of classes	# of generated lines of code (LOC)	% of generated code (%)	# of generated classes	% of generated classes (%)
ePetraas	38851	680	19169	49,33	603	88,67
SRADC	10231	231	5205	50,87	197	85,28

As we can observe, the classes automatically generated have in most cases a small size compared to those written manually. These are web form classes and therefore contain more code. Note that our results can be better if we chose to generate regular Web form classes. Only forms that have a complex user interface requires manual labor. By adopting MDE, CIHSSG has significantly reduced the cost for the development of these new systems. This economy is not only due to the degree of automatic source code generation, but also to the fact that the developed systems require very little maintenance effort. Our experience with the ePetraas system, in operation since early 2012, except for the first few weeks in the run-in period of the system, almost no intervention was required since then. The SRADC system, meanwhile, has been in operation since late 2013, has more required the implementation of new features than fixing existing bugs.

V. RELATED WORK

Despite the fact that MDE has an excellent reputation in the software engineering community, especially in academia, its acceptance and use in industry is very limited. Different studies have reported experiences of using MDE among both large companies (such as in Baker et al. [8], Staron [9] and in Hutchinson [10]) than in small and medium enterprises (such as in Cuadrado et al. [11], Kapteijns et al. [12] and in Clark and Muller [13]). Very few success stories have been reported. As stated in [14], studies have shown that the MDE adoption is facing "problems are as much to do with social and organizational factors as with tooling issues". The tooling issues are particularly the high cost of MDE commercial products, the need of learning a new language in order to develop the transformations and the cost of building the transformations. Reusing those transformations in more than one project is often required in order to have a return of investment.

In our experience, we have not faced the same problems as we have used the same implementation platform in the two case studies. We also chose XSLT as the language for building the transformations. Several free tools supporting the language are available. Learning a new language is not needed as XSLT is commonly used.

VI. CONCLUSION

In this paper, we presented two case studies reporting the adoption of a MDE approach in the development of two new systems. For each system, we presented its business requirements, an excerpt of its platform independent and platform specific models. A common implementation platform was used in both systems. Its model was also presented. Finally, we discussed the results obtained and showed how our approach has accelerated the development of the new systems, while reducing the cost of their development and maintenance.

REFERENCES

[1] D. C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," *Computer,* vol. 39, p. 25, 2006.

[2] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1," Object Management Group omg/2003-06-01, 2003.

[3] OMG, "Unifed Modeling Lanugage (UML) Specification : Infrastructure, Version 2.0," December 2003 2003.

[4] M. Fowler, *Patterns of Enterprise Application Architecture*: Addison-Wesley Longman Publishing Co., Inc., 2002.

[5] M. H. Kay, *XSLT 2.0 Programmer's Reference*: Wrox; 3rd edition, 2004.

[6] OMG, "XML Metadata Interchange (XMI), v2.1," 2005.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*: Addison-Wesley Longman Publishing Co., Inc., 1995.

[8] P. Baker, S. Loh, and F. Weil, "Model-Driven Engineering in a Large Industrial Context — Motorola Case Study," ed, 2005, pp. 476-491.

[9] M. Staron, "Adopting Model Driven Software Development in Industry – A Case Study at Two Companies," in *Model Driven Engineering Languages and Systems*. vol. 4199, O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 57-72.

[10] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming,* vol. 89, Part B, pp. 144-161, 9/1/ 2014.

[11] J. S. Cuadrado, J. L. C. Izquierdo, and J. G. Molina, "Applying model-driven engineering in small software enterprises," *Science of Computer Programming,* vol. 89, pp. 176-198, 2014.

[12] T. Kapteijns, S. Jansen, S. Brinkkemper, H. Houët, and R. Barendse, "A comparative case study of model driven development vs traditional development: The tortoise or the hare," *From code centric to model centric software engineering: Practices, Implications and ROI,* vol. 22, 2009.

[13] T. Clark and P.-A. Muller, "Exploiting model driven technology: a tale of two startups," *Software & Systems Modeling,* vol. 11, pp. 481-493, 2012/10/01 2012.

[14] 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*. vol. 8107, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. Clarke, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 1-17.