Generation of Context-Specific Electronic Patient Care Reports (ePCR) using Domain-Specific Modeling

Rohit Shenvi[1,2], Giovanni Mazza[1], Devashish Saini[1], Helmuth Orthner[1], Jeff Gray[2]

[1] Health Informatics Program, University of Alabama at Birmingham
Birmingham AL 35294-1170
{rshenvi, gmazza, dsaini, horthner} @uab.edu

[2] Dept. of Computer and Information Sciences, University of Alabama at Birmingham
Birmingham AL 35294-1170
gray@cis.uab.edu

Abstract. The creation of an electronic version of the Patient Care Report (ePCR) during the ambulance run remains a challenge for the pre-hospital environment. Current ePCR systems follow a "one-size-fits-all" approach. In this context, system components such as user interfaces and business models are developed without taking into account the specifics for different categories of patients or information needs of EMS agencies. These customizations involve extensive alteration of user interfaces, business objects and the data layer requiring significant investment of resources such as time and programming expertise. This paper describes an investigation into a model-driven approach to automate the generation of business objects and data layer classes for context-specific ePCRs. The workflow involves design of the metamodel, creation of domain models and generation of business classes based on the Component Based Scalable Logical Architecture (CSLA) framework.

1 Introduction to the EMS Environment

Emergency Medical Services (EMS) is defined as an organized system designed to transport sick or injured patients to the hospital [1]. EMS lies at the intersection of health care, public health and public safety, interacting with and carrying out the roles and responsibilities of all three [2]. With the almost universal acceptance to the single emergency number (e.g., 9-1-1 in the US), EMS is often the patients' first contact with the health care system. In the US, about 16 million patients are transported by EMS to Emergency Departments (EDs) every year [3].

EMS care is a collaborative effort between several organizations providing different levels or tiers of care. Most commonly, police or fire department personnel provide the first level of care [1]. These first responders are geographically distributed to allow early arrival at the scene of a medical emergency. The second level of care which involves providing interventions in the field and during transport varies widely

among EMS systems. Some systems have only a single level of care, most often trained to provide care at the ALS level (Advanced Level Service). Some rural agencies provide only BLS care (Basic Level Service) which all firefighters are trained to provide. Other systems deploy different levels of providers for care at scene and transport [1]. In the usual model of EMS care, patients are transferred to the closest ED. However, some systems make an active effort to identify the sickest patients and directly transfer them to a facility with available recourses for the appropriate level of care.

The use of technology in EMS for patient care documentation is growing. The EMS System Act of 1973 mandated that every EMS system "provide for a patient record system meeting appropriate standards" [4]. However, standards were difficult to define, and those that were, like the EMS Minimum Dataset (MDS), were not very widely used [5]. Electronic data collection in EMS began in earnest in the late 1980s and early 1990s as an attempt to collect information about patient care for quality improvement initiatives. Early efforts included manual entry into computers by data entry specialists and optically scanned 'lots-a-dots' forms [5-9]. In 1994, the National Highway Traffic Safety Administration USA (NHTSA) proposed a Uniform EMS Dataset consisting of 81 data elements [10], which has been further refined by the National EMS Information System (NEMSIS) project [11], and is currently in version 2.2.1 [12]. Several efforts toward the development of electronic patient care reports (ePCRs) have been reported [13-17] and several commercial products are available [18]. In the 200 largest US cities, 88% of agencies report that they are using some kind of electronic documentation. However, only 24% are able to input data at the patient's side [19]. Others have only paper-based documentation, or wait until they reach the hospital or base station to input the data.

1.1 The Need for an ePCR

Pre-hospital care documentation provides important information for EMS organizations and providers in the ED. This data can help improve patient care, allow for service reimbursements, and reduce exposure to litigation [20]. When the data collected is inaccurate or incomplete, there can be serious ramifications [20]. Typically, EMS providers collect information about the patient during a "run" using a paper-based Patient Care Report (PCR). The creation of an electronic version of this PCR, the ePCR, that can be filled during the run, remains a challenge. Very few versions of the ePCR that are usable for data entry during a run exist in the market. In most of the cases, data is documented in various formats and on unreliable media such as scratch pads and completed after the run. It is therefore easy to understand that in hectic emergency situations data may be missed, especially when multiple patients are involved.

Many states in the US require that PCR data be sent to a central state database. Normally, the EMS personnel returning from the field enter this data manually, which is a time consuming and error prone approach. Several EMS agencies also allow the PCR to be dictated whereby dedicated personnel document the information recorded during a run [20]. EMS agencies invest a significant amount of resources maintaining this non-streamlined workflow. Paper-based PCRs raise several Health Insurance

Portability and Accountability Act (HIPAA) concerns since proper procedures must in place and be maintained to control access to them [20].

The purpose of an ePCR is to ease the timely acquisition of accurate patient data. The collected data can offer insights into performance metrics that can be analyzed later and help to improve quality of patient care, thereby giving a competitive edge to the agency [20]. Various EMS agencies may need to collaborate to offer patient care in the event of mass casualties. Sharing patient information among active EMS units in the field can be achieved with the ePCR. Data security can be enforced using cryptographic techniques.

The clinical stabilization and patient care are the topmost priorities. For certain acute conditions such as heart attacks, stroke, diabetic coma, and certain trauma injuries, time is also critical. In these latter cases it is also important that the patient is transported to a clinical site that has the necessary resources available for immediate treatment. In these cases, the ED needs to be notified prior to the arrival of the patient so that these resources can be mobilized and ready when the patient arrives (e.g., an interventional cardiologist and a cardiac catheterization lab). Hence, the user interface of ePCRs should be straightforward for the capture and use of clinical data in the field.

Customizing features in the present ePCR cause a rippling effect across the entire system including the UI, business and the data layer. Deployment of new features requires re-building the entire application after inclusion of the desired feature. Modules are coupled in the sense that they cannot be deployed individually in the base application. Data bindings exist between elements in UI and the business objects and the data layer. A minor change such as addition or removal of a data element in any of the layers leads to a broken link. Section 2.2 explains these challenges in detail.

The paper is organized in the following sections. Section 2 explains the architecture of the present ePCR and the complexities involved in its development. Section 3 argues against the development of a common ePCR for every context and advocates the development of context-specific and feature-oriented ePCRs. Section 4 describes a domain-specific modeling approach to address the problems described in the earlier sections. Section 5 gives a brief description of similar work and Section 6 summarizes the paper.

2 Description of the ePatient System

The “Next Generation Emergency Responder” is a component of the Advanced Network Infrastructure for Health and Disaster Management (AdvNet) project funded by the National Library of Medicine (NLM) at NIH. The project’s focus is to study and develop a comprehensive infrastructure for the pre-hospital EMS environment starting with the 9-1-1 emergency medical call to the delivery of the patient to the appropriate ED for immediate treatment. The development of an ePCR that can receive patient information from the 9-1-1 dispatcher, acquire accurate clinical data in the field, and transmit this data to the ED before the patient arrives is a major component of our project.

The entire system for managing the PCR information flow is known as **ePatient**. The first iteration of the ePatient System was developed in collaboration with

Intergraph Corporation. We used Intergraph's Computer Aided Dispatch (9-1-1 CAD) system. This 9-1-1 medical dispatch system (I/Dispatch) was enhanced with optimized dispatch algorithms for acute cardiac and trauma incidents that could make use of critical patient data (if available electronically) and push this data with the new event data obtained by the dispatcher to the ePCR before the EMS provider arrived at the patient. The ePCR was also enhanced to communicate with the central ePatient database using several communication methods such as Verizon's CDMA EV-DO (a 2.5G cell phone technology), Wi-Max (IEEE 802.16), and Wi-FI (IEEE 802.11). Essentially, our testbed allows us to study the entire pre-hospital care process from the moment the 9-1-1 call is generated to the moment the patient is delivered to the hospital and handed over to the ED team. The ePatient System encompasses the following parts:

- The **ePCR** is the main GUI employed by the EMS providers to collect clinical information in the field about the patient and store in the local DB.
- The **eMonitor** is a component that is always running in the background and is used to send/receive information to/from the ambulance and the main database independently of the state of the network. If the network is not available for any reason it just keeps waiting until one of the three network transports is available. The network itself will choose the fastest transport if more than one are available.
- The **ePatientDB** is the Microsoft SQL database used to collect the patient information. A local copy of the database is carried by the mobile devices but limited to information for the current run. The synchronization of the local copy with the central database occurs when there is network connectivity.
- The **CADClient** is a GUI used to search the ePatientDB for information about the patient from previous runs that can then be sent to the ePCR in advance by the 9-1-1 medical dispatcher from the dispatch station.
- The **ePWebService** is a web service used to show summary information about the patient into the I/Netviewer system. I/Netviewer is the web interface of the Intergraph CAD I/Dispatch system.
- The **ePatientService** is a component used to relate the information collected with the ePCR system with the information collected by the Intergraph CAD system. In the field the CAD system uses I/Mobile GUI to collect and send information about the event.

The I/Mobile is a part of the I/Dispatch System used to monitor and send the GPS coordinates of the ambulance and collect information about the event. I/Mobile also receives and sends status information back and forth to the dispatch system from the field. The idea is to use I/Mobile and ePatient together to monitor and manage EMS events, including where the necessary clinical resources are available to treat a patient and provide patient information to the staff in the ED before the patient arrives. Figure 1 shows the main data flow among the different components of the ePatient System. Each rectangle denotes where the application is running at the moment of interaction in a single run.

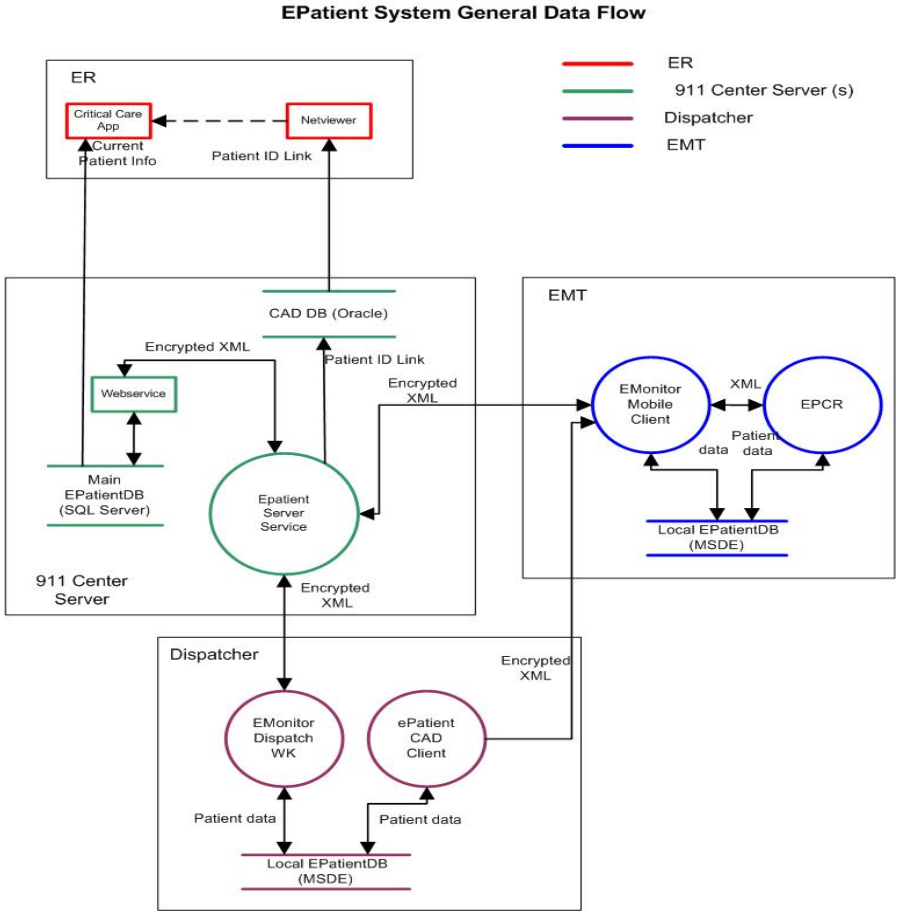

Fig. 1. ePatient Data Flow

2.1 ePCR Architecture

The ePCR is the GUI portion of the ePatient system implemented on the mobile units (Tablet PCs) and is used by Emergency Medical Technicians (EMTs) in the field. Figure 2 shows a snapshot of the ePCR. During the first iteration we used a classical Windows Forms Client/Server Architecture using C# with a local Microsoft SQL Database. In classical windows form design, the main form was loaded with code and the different portions of the ePCR were loaded in different Windows Controls that are loaded into the main form at run-time when required. This architecture provided flexibility but the business logic and the presentation logic were intertwined inside the forms code; thus, every modification and event manipulation required considerable code modifications in the main form. Initially, we thought these limitations were unimportant but when we received feedback from EMTs about feature requests (e.g., a speech recognition feature), the limitations of our architecture emerged and the need for a more flexible architecture became imperative.

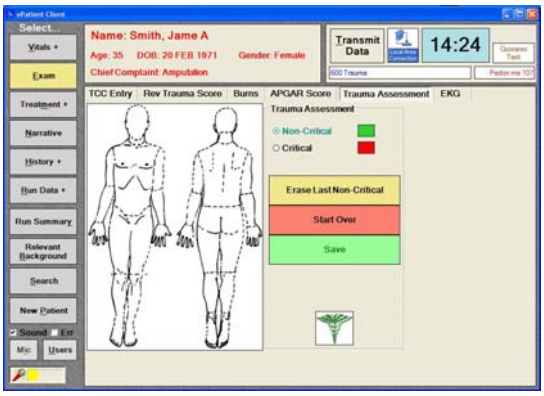

Fig. 2. ePCR User Interface

We adopted a new architecture proposed by the Microsoft Patterns & Practices team to develop smart client software which is called CAB (Composite User Interface Application Block) for the second iteration of ePatient. This model allows an application to be based on the concept of modules or plug-ins that are hosted in a main Windows Form called the shell. These plug-ins enable developers to build components that hide user interface complexity from the business logic development and facilitates development using patterns for loose coupling between modules. Each module acts as a distinct piece of software that also can be deployed separately and integrated at run-time into the main shell through XML configuration files. Figure 3 shows a general overview of the modularization of the main subsystems that constitute a CAB application. Each module as well as the shell can be developed by different teams as long as the interfaces and common models are maintained. The modules can be deployed separately and loaded when required. For instance, the speech recognition module can be deployed independently of the base application.

The separation of GUI logic from business logic is not a novel idea, but in CAB the GUI itself is composed at run-time using different modules that are independent. There is a global event handler in the shell that facilitates data sharing between business objects that are contained in different modules. In addition, each module may also have an internal event handler to manage events occurring within its scope.

For the creation of a new business logic layer, another framework has been chosen. This framework is called CSLA (Component-based, Scalable, Logical Architecture) [22], which assists in creating movable business objects that can be hosted on a central server or can move to a remote client. A CSLA business object contains a representation of the data and all the validation required by the business rules. Each business object has an independent data layer that contains the logic to preserve the object's state into the database. The data layer and the business layer can be integrated into the same object or be separated in different objects across address spaces. For the ePCR, the business object is separated from the data layer and exchanges information with the data layer using DTO (Data Transport Objects), which are simple objects that represent data. The data layers use ADO.Net Datasets to preserve the information into the local database.

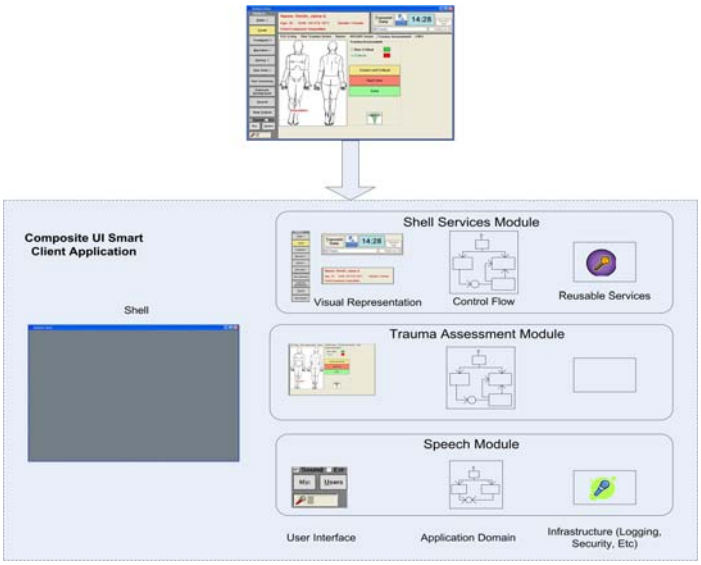

Fig. 3 Overview of the Composite UI Application Block (figure adapted from [21])

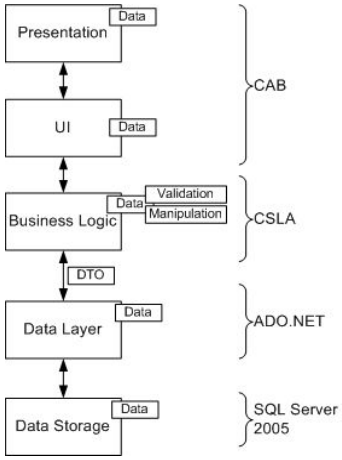

Fig. 4. Architecture for the ePCR GUI (figure adapted from [22]).

All of the overhead associated with this process (i.e., user interface interaction, validation of the business object, and movement of information to the data layer) allows for a more flexible and maintainable code but imposes a considerable amount of repetitive coding to create all the mapping. Coding is required in the following modules: from the ADO.Net Datasets to the DTOs in the data layer and later on from the business objects and the DTOs and finally the UI fields and the business object fields. To create all this mapping, the use of modeling tools offers an advantage that

makes the process straightforward by providing documentation as well as giving a graphical representation of the relation among the objects.

Figure 4 shows the architecture of the system using the model proposed by Lhotka [22]. Notice that all the business rule validations are delegated to the business objects and non-validation and manipulation are done in the other layers. DTOs are used to transfer data between the Data Layer and the Business Logic. Physically, these layers can be distributed in the network on different servers or reside on the same machine. CSLA allows streamlining the process by hiding the complexity of separating Business Logic from the Data Layer using multiple mechanisms of communication that are configurable at run-time via XML files. It is possible to use windows remoting, web services and even the new Window Communication Foundations (WCF) to move the business objects across the network so that the UI and presentation layer can use all the validation and manipulation logic locally.

The last modification to the architecture from the previous version was the incorporation of the WCF as the main vehicle to move data across the network for the eMonitor portion of the ePatient system. Our second iteration specifies services at both ends of the network to subscribe, notify and move XML messages based on the NEMSIS format to update the ePCR and query for data to the central database. WCF allows changing the mechanism of transport dynamically at run-time via XML configuration without changing the server.

2.2 Challenges of the ePCR Design

The main obstacle with the design of a multilayered system is that any aggregation or modification implies a change in other parts of the application. Given the fact that the requirement of information varies among EMS stations, the implementation requires that the data collection be flexible enough to incorporate different elements for customization needed for different workflows in different EMS stations. The incorporation of new features (e.g., speech recognition, automated data collection from medical devices, and patient specific-data entry) presents new adaptation challenges; for example, data entry should vary according to patient characteristics such a pediatric, adult, and geriatric. For this to work, new business objects need to be incorporated into the base ePCR system. The CAB model helps by allowing additional features to be introduced as new modules that integrate into the event model of the application.

One of the benefits that can be realized by adopting a modeling solution is streamlining the process of generating the skeleton for the CSLA objects. This would involve modifying the respective DTOs and the data layer interface so the incorporation of new business objects could be simplified, which would make the customization process for the ePCR more agile.

3 Exploring Design Alternatives for ePCR

In this section we expose the shortcomings of the present approach to designing ePCRs and argue for the adoption of context and feature-specific models to design a family of ePCRs to fulfill diverse needs for different stakeholders.

3.1 Context-Specific ePCRs

Current ePCRs are designed using "a one size fits all approach." In this context, system components such as user interfaces and business models are developed without taking into account the specifics for different categories of patients or information needs of EMS agencies. EMS agencies often need to capture additional information for purposes of quality assurance, billing, and resource usage analysis apart from the mandatory data elements specified by NEMSIS. Features such as voice recognition and clinical devices that record patient vitals may be required by some agencies. These customizations involve extensive alteration of user interfaces, business objects and the data layer, which require significant investment of resources such as time and programming expertise. Other customizations relate to patient specific information (e.g., pediatric, adult, or geriatric patients requiring different data elements) or relate to the patient's acute and chronic condition such as cardiac, stroke, and trauma. Each of these variations invariably results in changes to the business objects and their representation in the database. Also, a large part of the user interface needs to be altered to accommodate additional or different fields. Context-specific ePCRs eliminate the need to adapt a common ePCR to each specific context, thus reducing the maintenance effort. Details of a particular context are kept within the bounds of that specific version of the ePCR.

3.2 Feature-oriented ePCRs

A data acquisition system such as the ePCR should be designed as a family of systems with the members supporting different features. Features are product characteristics that identify family members within a product line [23]. Some agencies may opt for medical devices to record patient vitals and others would want a voice interface to the ePCR. Drug interaction modules are absent in current ePCRs. The primary reason for this is that drug interaction systems are standalone applications with their own knowledge base. Lack of integration with the ePCR and high costs discourage EMS agencies from adopting an interaction system. Embedding the interaction engine within the ePCR context would allow users to query for drug interactions without switching between applications and different data formats for each system. These are features that may be present or absent in an ePCR. EMS agencies can opt for the ePCR that best fulfils their business needs, thus reducing the costs involved in manual customization of the existing system. The base ePCR system contains the basic set of modules required by each EMS agency. Features are represented as domain constructs at the abstract layer (domain model). The design of the metamodel for ePCR is explained in detail in the next section.

4 Introduction to Model-Driven Engineering of ePCR Systems

Model-Driven Engineering (MDE) is one of the recent advances in software engineering that promises to improve the software development lifecycle. MDE elevates the level of abstraction above that of third generation programming languages by providing an environment that allows creation of models representing

domain objects rather than source code. End-users are typically domain experts who design the domain-specific model without the need of any programming knowledge. The models are specifications of the system that are independent of target platform details. Model Integrated Computing [24] is a variant of MDE that focuses on Domain-specific Modeling (DSM) [25], which is unlike traditional Unified Modeling Language (UML) models that are general purpose and apply to various domains. DSM leverages the model development process by providing a set of entities specifically tailored for a single domain.

Domain-specific Modeling Languages (DSMLs) can be text-based or visual. The first step in designing a DSML is identification of system elements that need to be modeled, which are specified in a metamodel that defines the language for a specific domain. This is done through iterative requirements-gathering phases with domain experts and various stakeholders of the system [26]. After the core entities for a domain are discovered, the environment designer maps each entity to the metamodeling constructs available in a modeling environment. A visual modeling environment allows customization of visual attributes using decorators [26].

End-users can use the language design models that obey the constraints specified in the metamodel. The final step in the DSML approach is the process of designing one or more model interpreters for each target platform. A model interpreter is a translator that accesses the model structure and offers services such as analysis, validation and code generation. An interpreter can be written in a general-purpose programming language, such as C++ or Java.

The Generic Modeling Environment (GME) is an open source visual modeling tool designed to construct domain-specific environments [27]. An environment designer initiates the workflow by constructing a metamodel, which is built using the visual idioms available in the GME. Metamodels contain the objects, attributes and relationships that represent the building blocks of the domain-specific model. The visual aspect of the different objects is also specified in the metamodel. Additionally, constraints are specified using the Object Constraint Language (OCL) to force the end-user to abide by the syntax of the target domain. Domain experts design the system's visual representation by using the various elements from the metamodel. After the domain-specific model is designed, a model interpreter generates source code for the target platform.

4.1 Designing a Metamodel for ePCR

Three main constructs are used to realize the initial ePCR metamodel. Model elements act as a container or a namespace for the ePCR modules represented by the atom construct. Relationships between the ePCR and its elements are represented using connections. Each modeling construct has several attributes that define the visualization and behavior of that element. Patient demographics, vital signs and medications, which are integral parts in every ePCR context, are defined as atom elements in the metamodel. The connections in an ePCR model represent a containment relationship between the atoms and the model. Each element described in this metamodel is a member (mandatory or optional) within the ePCR. A containment relationship is also defined between an object represented by an atom and its properties. Certain mandatory objects may be quite easily overlooked in the domain

model, which is an error. To enforce the inclusion of such objects, GME's object constraint manager can be used to define constraints on the creation of the domain model. A partial view of the metamodel defined for the ePCR domain is shown in Figure 5.

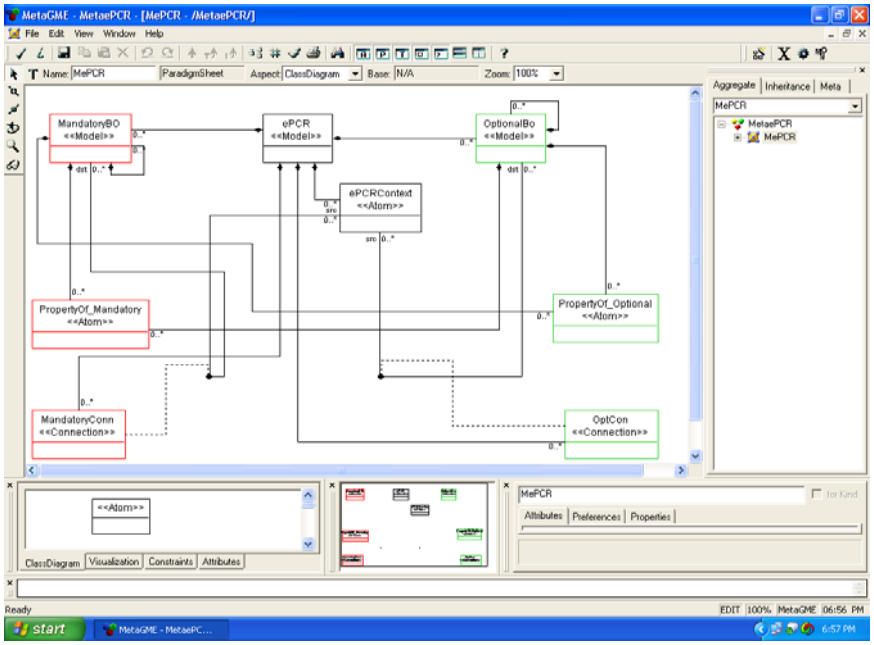

Fig. 5. Metamodel for ePCR

4.2 Using the Customized Environment

End-users of a domain-specific modeling language may be users with limited or no programming experience. However, because the purpose of this project is to automate the production of the business and data layer objects, the ePCR modeling environment is targeted for developers. The user starts with adding the top-level container, namely the ***ePCR*** model construct from the browser window in the design environment (see Figure 6). Business objects that are needed in the ePCR can be created by dragging the elements named ***MandatoryBO*** and ***OptionalBO***. Similarly, relation-ships are specified using one or multiple connection entities that were created in the metamodel. For example, the relationship between the vitals business object and the ePCR model element is created by using the mandatory connection element. Properties or fields are added to the business objects using the ***Propertyof_Mandatory and Propertyof_Optional*** elements. The added properties appear within the business object bounds.

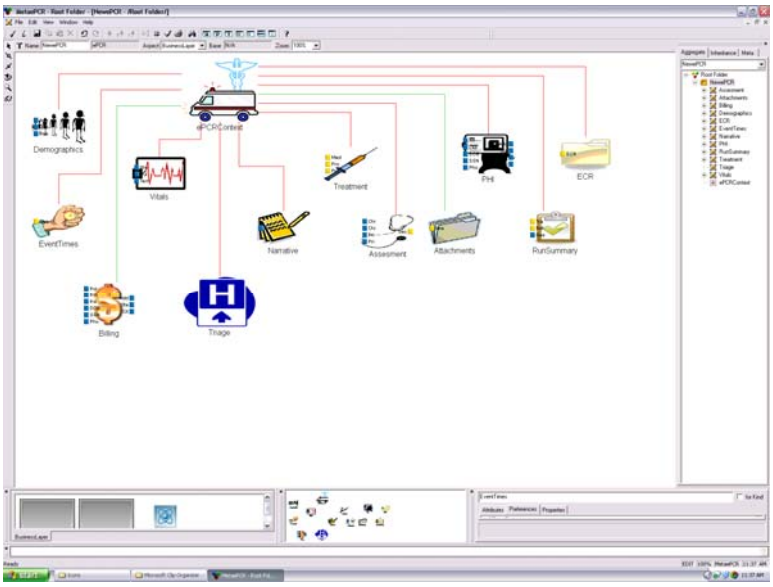

Fig. 6. Domain-specific Model of an ePCR Configuration.

4.3 A Customized Generator for ePCR

GME supports the use of the Builder Object Network (BON) framework for building model interpreters. The job of the BON is to instantiate C++ objects for each of the objects in the GME model tree. An interpreter can perform simple tasks such as printing out the model elements or complex jobs such as generating code for a particular platform. The model interpreter created for the ePCR domain traverses the internal structure of the model and generates CSLA classes for each of the business objects contained in the ePCR model. A business object in CSLA is represented as a ***BusinessClass*** type. The model interpreter is executed to generate the source files needed for the ePCR business layer. In addition to the business layer, data objects are required to store the captured data elements.

However, code generation for the data layer has additional challenges. Stored procedures are a popular mechanism of storing data in the database because they exist at the database level and the queries are optimized to provide better performance. Database operations are not just restricted to stored procedures. Transactions can be executed by running SQL statements embedded in the application code. Metadata for object properties are usually different than those of data base table fields. To address this challenge, a XML mapping file needs to be maintained that associates business object properties and database fields or stored procedure parameters. All properties in the business objects that need to be saved to the database need to be made serializable in order to ensure that they can be transported to a remote data layer in a distributed environment. Addition of new business objects into the deployed application requires corresponding data fields in the database and the data layer. The mapping file and the stored procedures existing at the database level need to be modified. These variations present a major hurdle in the generation of the data layer from the domain model.

5 Related Work

InterComponentWare (ICW) offers its partners the eHealth Framework, a powerful platform for the development of healthcare solutions. ICW has developed domain-specific languages that enable developers to express the domain objects for the applications in models that are independent of the platform and implementation technologies. The models, together with the architecture templates, serve as input to a code generator. The framework allows reuse of components existing at the modeling level, which greatly enhances the semantic consistency of the generated applications [28]. The automated generation of domain models to executable code increases efficiency and development speed and at the same time reduces errors [28]. Feature Oriented Programming (FOP) has been used to develop a product line of portlets [29], which shows how variations in a product line can be synthesized by composing features to create models and generate source code from these models. Mathematical properties of portlet synthesis are investigated to validate the correctness of the models, tools and the specifications [29].

6 Summary and Lessons Learned

This paper presents an initial investigation into a model-driven approach to automate the generation of the business and data layer for different ePCR contexts. This process involved designing a metamodel for a language that consisted of entities from the ePCR domain, applying the metamodel to create domain-specific models and creating a model interpreter to translate the abstract domain entities into CSLA business classes. The paper also addresses issues related to the generation of code for the data layer.

6.1 Lessons Learned

Business rules the govern majority of workflows within an organization. Business rules can be of the following types: Decision support; process flow, and data validation. The traditional approach intertwines business rules within application code often appearing in the GUI. Although this is a quick solution, it proves to be a maintenance nightmare because changing a business rule requires changing GUI code. Changing or adding business rules may affect one or more objects in several layers, which negatively affect scalability. It would be better if a business analysts can specify rules in a simpler language without the rigidity of a programming language.

Loose coupling between business rules and the business objects would be desirable because business rules change more frequently than business objects. Therefore, adoption of a business rule language becomes necessary. For example, in the EMS environment most clinical protocols are simple but change frequently. If we can find a method to specify clinical protocols unambiguously (e.g., a business language), one could take these specifications and translate them into executable code.

7 Future Work

Our current approach does not cover the issues involved in the adaptation of ePCR User Interface (UI) according to device characteristics (e.g., screen size). The complexity of our ePCR UI would require a substantial amount of design intelligence to be built into the model interpreter to accomplish this flexibility. Eventually, we hope, the domain-specific modeling tools will be powerful enough to create complex user interfaces required in the EMS domain.

The degree of automation achievable within the data layer needs to be further explored. Because quality and correctness of healthcare data play a critical role, unit tests are required for the generated business objects to ensure interpreter correctness. Because a significant amount of the system is generated through automation, the domain model (i.e., its properties) need to be evaluated using formal validation and verification techniques. This will assure that the preservation of properties in the generated source code [30].

Acknowledgements

This project has been funded by Federal funds from the National Library of Medicine, National Institutes of Health, under Contract No. N01-LM-3-3513 and the National Science Foundation, under CAREER grant CCF-0643725.

References

1 Pozner CN, Zane R, Nelson SJ, Levine M, "International EMS systems - The United States: Past, Present, and Future," *Resuscitation*, March 2004, vol. 60, no. 3, pp. 239-244.
2 Institute of Medicine, "Emergency medical services at the crossroads," *The National Academies Press*, Washington, D.C, 2007 [cited 2007 May 11], (http://www.nap.edu/catalog.php? record_id=11629)
3 Burt CW, McCaig LF, Valverde RH, "Analysis of ambulance transports and diversions among US emergency departments," *Annals of Emergency Medicine*, Apr 2006, vol. 47, no. 4, pp. 317-326.
4 *Emergency Medical Services Systems Act of 1973, Public Law 93-154*
5 Spaite D, Benoit R, Brown D, Cales R, Dawson D, Glass C, et al, "Uniform prehospital data elements and definitions: A report from the uniform prehospital emergency medical services data conference," *Annals of Emergency Medicine*, vol. 25, no. 4, pp. 525-534.
6 Joyce SM, Brown DE, "An optically scanned EMS reporting form and analysis system for statewide use: Development and five years' experience," *Annals of Emergency Medicine*, vol. 20, no. 12, pp. 1325-1330.
7 Spaite DW, Hanlon T, Criss EA, Valenzuela TD, Meislin HW, Ross J, "Prehospital data entry compliance by paramedics after institution of a comprehensive EMS data collection tool," *Annals of Emergency Medicine*, vol. 19, no. 11, pp. 1270-1273.
8 Stewart RD, Burgman J, Cannon GM, Paris PM, "A computer-assisted quality assurance system for an emergency medical service," *Annals of Emergency Medicine*, vol. 14, no. 1, pp. 25-29.
9 Swor RA, Hoelzer M, "A computer-assisted quality assurance audit in a multiprovider EMS system," *Annals of Emergency Medicine*, vol. 19, no. 3, pp. 286-290.

10 Mann NC, Dean JM, Mobasher H, Mears G, Ely M, "The use of national highway traffic safety administration uniform prehospital data elements in state emergency medical services data collection systems," *Prehospital Emergency Care*, vol. 8, no. 1, pp. 29-33.
11 Dawson DE. "National Emergency Medical Services Information System (NEMSIS)," *Prehospital Emergency Care*, vol. 10, no. 3, pp. 314-316.
12 *NHTSA uniform pre-hospital EMS dataset version 2.2.1: Final documentation and data dictionary*, 2006 [cited Mar 29 2007], (http://www.nemsis.org/media/pdf/NEMSIS_Data_Dictionary_v2.2.1.pdf)
13 *Battlefield Medical Information Systems Tactical - Joint (BMIST-J)*, [cited May 20 2007], (https://www.mc4.army.mil/BMIST-J.asp)
14 Anantharaman V, Swee Han L, "Hospital and emergency ambulance link: Using IT to enhance emergency pre-hospital care," *IJMI*, vol. 61, no. 2-3, pp. 147-161.
15 Killeen J, Chan TC, Buono C, Griswold WG, Lenert LA, "A wireless first responder handheld device for rapid triage, patient assessment and documentation during mass casualty incidents," *AMIA Annual Symposium Proceedings*, 2006,pp. 429-433.
16 Meislin HW, Spaite DW, Conroy C, Detwiler M, Valenzuela TD, "Development of an electronic emergency medical services patient care record," *Prehospital Emergency Care*, vol. 3, no. 1, pp. 54-59.
17 Pavlopoulos S, Kyriacou E, Berler A, Dembeyiotis S, Koutsouris D, "A novel emergency telemedicine system based on wireless communication technology—AMBULANCE," *IEEE Transactions on Information Technology inl Biomed*, vol. 2, no., 4, pp. 261-267.
18 *NEMSIS Compliant software*, [updated May 17 2007, cited May 20 2007], (http://www.nemsis.org/compliance/compliantSoftware.html)
19 Williams DM, 2005 "JEMS 200-city survey. A benchmark for the EMS industry," *JEMS*, vol. 31, no. 2, pp. 44-61, 100-101.
20 Zoll Data Systems, *RescueNet TabletPCR White Paper*, (www.zolldata.com/csite/NewPDFs/RescueNet%20TabletPCR%20White%20Paper 2006.pdf)
21 Microsoft Patterns and Practices,"Overview of Composite UI Application Block," (http://msdn2.microsoft.com/en-us/library/aa546409.aspx,)
22 Lhotka R, *Expert C# 2005 Business Object 2nd Springer-Verlag*, New York 2006.
23 Batory D, Sarvela JN, Rauschmayer A, "Scaling Step-Wise Refinement," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 355-371.
24 Sztipanovits J and Karsai G, "Model-Integrated Computing," *IEEE Computer*, April 1997, pp. 10-12.
25 Gray J, Tolvanen JP, Kelly S, Gokhale A, Neema S, and Sprinkle J, "Domain-Specific Modeling," *Handbook of Dynamic System Modeling*, (Paul Fishwick, ed.), CRC Press, 2007.
26 Balasubramanian K, Gokhale A, Karsai G, Sztipanovits J, and Neema S, "Developing Applications Using Model-driven Design Environments," *IEEE Computer*, vol. 39, no. 2, pp. 33-40.
27 Lédeczi A, Bakay A, Maroti M, Volgyesi P, Nordstrom G, Sprinkle J, and Karsai G, "Composing Domain-Specific Design Environments," *IEEE Computer*, vol. 34 no. 11, pp. 44-51.
28 InterComponentWare AG, "ICW eHealth Framework, Platform for Interoperable Health Care Solutions," pp. 2, 6 (www.us.icw-global.com/icw/content/e77/e172/e3112/ehealth-framework-en_usa.pdf)
29 Trujillo S, Batory D, Diaz O, "Feature Oriented Model Driven Development: A Case Study for Portlets," *International Conference on Software Engineering (ICSE)*, May 2007, Minneapolis, MN, pp. 44-53.
30 Jones VM, "Model Driven Development of m-Health Systems," *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, Washington, DC, pp. 580.