Designing and Implementing an HL7 Software Factory

Mauro Regio
Microsoft Corporation
One Microsoft Way
Redmond,WA 98052 USA
+1 (425) 705 3538
maurore@microsoft.com

Jack Greenfield
Microsoft Corporation
One Microsoft Way
Redmond,WA 98052 USA
+1 (425) 703 7575
jackgr@microsoft.com

ABSTRACT

In this paper, we share the experience gathered in designing and implementing a software factory for healthcare systems based on Health Level Seven standard. We discuss the long term vision and the scoped down proof of concept developed so far. We also outline the challenges encountered in our project and the opportunities to widen the scope of the approach to different industries and, more in general, to support business to business collaboration.

Keywords

Software Factories, Domain Specific Modeling, Healthcare, HL7

1. INTRODUCTION

The objective of this paper is to share the experience gathered in designing and implementing a software factory based on Health Level Seven (HL7), a standard for interoperability among healthcare organizations.

The work started almost one year ago with the high level specification of the factory, producing a first version of its schema and solution architecture, as detailed in [1].

In its initial phase, the factory targeted the design of HL7 collaboration ports, which are systems designed to i) be deployed at the edge of IT systems of healthcare organizations and ii) enable healthcare applications to collaborate in conformance with business and technical protocols standardized in HL7 Version 3, using a Web Service based communication infrastructure.

In the second phase, we implemented a first –scoped down– version of the HL7 Factory specified in the first phase. The focus of this version is the subset of HL7 collaboration port capabilities necessary to enable communication among healthcare applications through Web Service adapters [2], in conformance with HL7 Web Service profiles [3].

The full scope of the factory, as specified, also included development of enterprise application integration adapters to connect existing applications to Collaboration Ports, and orchestration of business message exchanges realizing a particular collaboration on behalf of line of business applications that were not designed to collaborate.

Our experience in designing and developing HL7 Factory has been valuable from two different perspectives.

In the developing the HL7 Factory we encountered some challenges in developing the factory schema, managing the factory configuration, understanding how domain specific languages would be used and leveraging the tools available at the time in the development environment.

At the same time, we realized that the factory's scope could be widened from collaboration among healthcare applications based on HL7 to a more generic notion of collaboration among applications based on standardized (or shared) specifications.

Therefore, we are currently in the process of generalizing the approach proven in the initial implementation of the HL7 Factory to design and build what we have called the Business Collaboration Factory.

2. SOFTWARE FACTORIES

Software factories use specific domain knowledge, solution architectures, tools and other reusable assets to help their users produce specific types of software solutions. A software factory is based on three key ideas: a software factory schema, a software factory template, and an extensible development environment.

A software factory configures an extensible development environment, such as Eclipse, Borland JBuilder, or Microsoft Visual Studio Team System (VSTS), using an installable package called a software factory template or guidance package. When configured in this way, the development environment becomes a specialized facility that accelerates the development of a specific type of software solution, such as a user interface or database access layer, or perhaps a whole application in a business domain like healthcare or homeland security. The software factory template is organized by a model called a software factory schema. The schema defines one or more viewpoints relevant to stakeholders in the production of the target software solutions. Each viewpoint defines the life cycle artifacts produced or consumed by its stakeholders, the activities they perform against those artifacts, and the reusable assets available to support them in performing those activities.

The software factory methodology [4] integrates model-driven development (MDD), component-based development (CBD) and agile development practices, including the use of patterns and pattern languages with models, frameworks and tools.

In order to leverage models effectively for various forms of automation, software factories make heavy use of domain-specific languages (DSLs). DSL technology is much newer than most of the other technologies used in software factories, and relies on families of extensible languages. DSL development tools and frameworks have been under development for some time in academic circles [7], however, and have recently started to appear in commercial form [5][8].

3. THE HL7 FACTORY

The HL7 factory automates the development of systems called collaboration ports, which enable interoperation among systems in the healthcare domain. Specifically, the solutions produced by the factory aim to:

- Realize Interactions defined by the HL7 standard as information exchanges that take place between Application Roles in response to Trigger Events. Collectively, these exchanges support the business goals of a specific use case, such as performing a Laboratory Observation. The factory automates the production of code that implements these interactions by mining information contained in the HL7 Reference Information Model (HL7 RIM).
- Enable application-to-application business collaboration expressed in terms of these interactions over an open standards–based Web Service infrastructure that is conformant to a subset of the HL7 V3 Web Service profiles [3], namely the Basic, Addressing, Security, and Reliable Messaging topics.
- Enable integration of new or existing applications that were *not* designed:
  1. For HL7 Version 3.
  2. To fulfill a business collaboration.
  3. To communicate over a Web Service infrastructure.

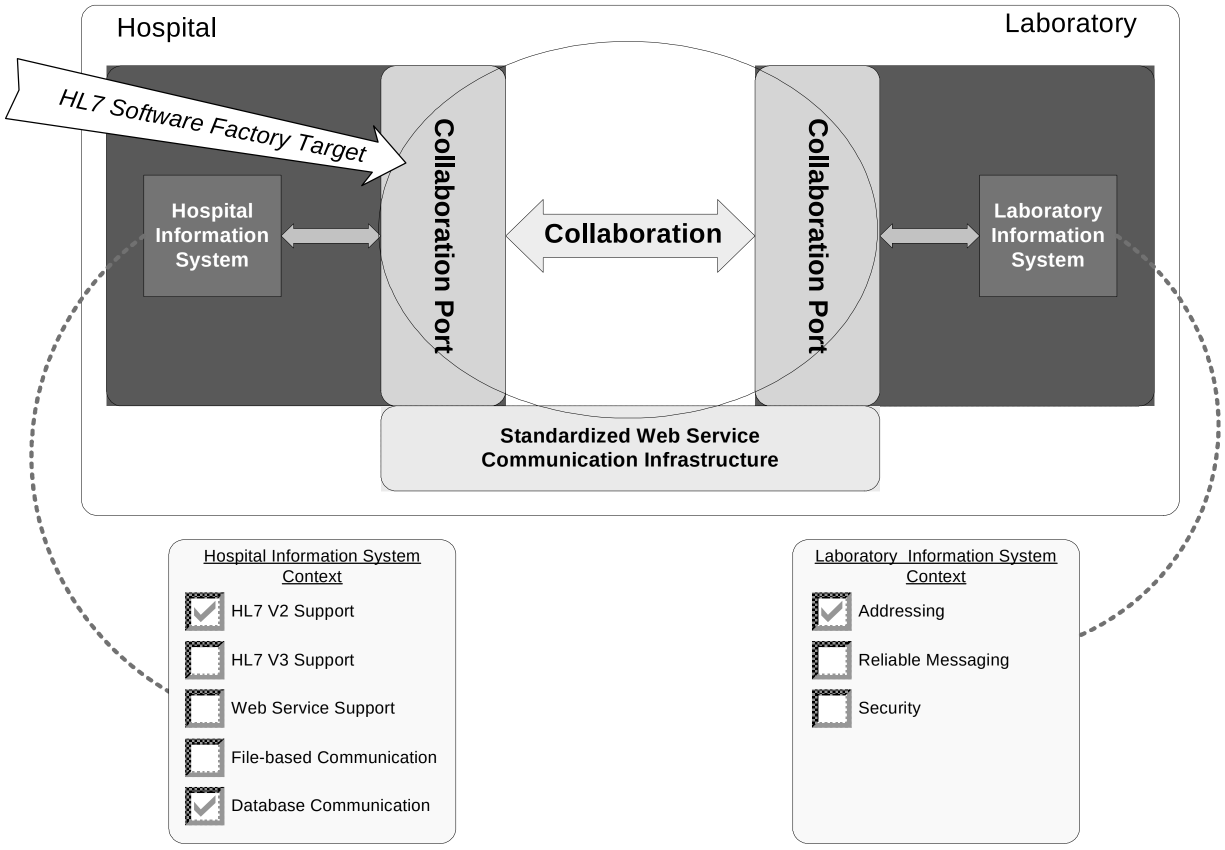
3.1 Context

It is important to understand the HL7 factory from two different perspectives: production and development.

3.1.1 Production Context

The factory end products are HL7 collaboration ports. These ports automatically enable disparate healthcare applications to collaborate behind and across the firewall using Web Services, provided that:

- At least one of the applications participating in the business collaboration will deploy an HL7 collaboration port. The other applications may participate through other meains



Hospital
Laboratory
HL7 Software Factory Target
Hospital Information System
Collaboration Port
Collaboration
Collaboration Port
Laboratory Information System
Standardized Web Service Communication Infrastructure
Hospital Information System Context
HL7 V2 Support
HL7 V3 Support
Web Service Support
File-based Communication
Database Communication
Laboratory Information System Context
Addressing
Reliable Messaging
Security

Figure 1 : HL7 Factory - Production Context

- All of the applications participating in the business collaboration will conform to HL7 V3 standards for message exchange, either natively, or with the help of an HL7collaboration port.

For a business collaboration between a hospital and a laboratory system, Figure 1 shows where HL7 collaboration ports sit in relation to the systems hosting the interoperating applications.

Note that collaboration ports are meant to be highly configurable to enable general dispatching, while also allowing complex message flow orchestration. Thus, ports like the ones shown in Figure 1 are configured both in terms of the technical details for a specific implementation and deployment and in terms of HL7 domain definitions and conformance levels.

3.1.2 Development context

The purpose of the software factory is to accelerate the specification and implementation of collaboration ports. As shown in figure 2, the factory combines problem domain knowledge supplied by the HL7 Reference Information Model and Web Service profiles, with knowledge of the platform technology, solution architecture and development process supplied by platform documentation and by the factory developers.

As suggested by the illustration, this knowledge is packaged into numerous assets, which collectively form the factory template. Simply stated, the factory template provides everything required to build an HL7 collaboration port, including reference data and artifacts, such as message schemas, tools, such as adapters generators, and process guidance.

The factory template must be installed into an Integrated Development Environment (IDE), namely Microsoft Visual Studio 2005 Team System, before it can be used to produce and deploy HL7 collaboration ports.

4. LEARNING

As noted above, the purpose of this paper is to describe the learning gained in specifying, designing, and implementing the HL7 factory. We have grouped the information into two categories. The first deals with lessons learned about factory development and usage in general. The second deals with insights gained regarding the target domain, and with how the factory might be generalized to address a broader set of target domains.

4.1 Challenges and opportunities

The most significant challenge from the inception of the project through to completion was the development and management of the factory schema.

Producing an initial version of the schema was relatively easy [1]. A grid based approach can be effectively used

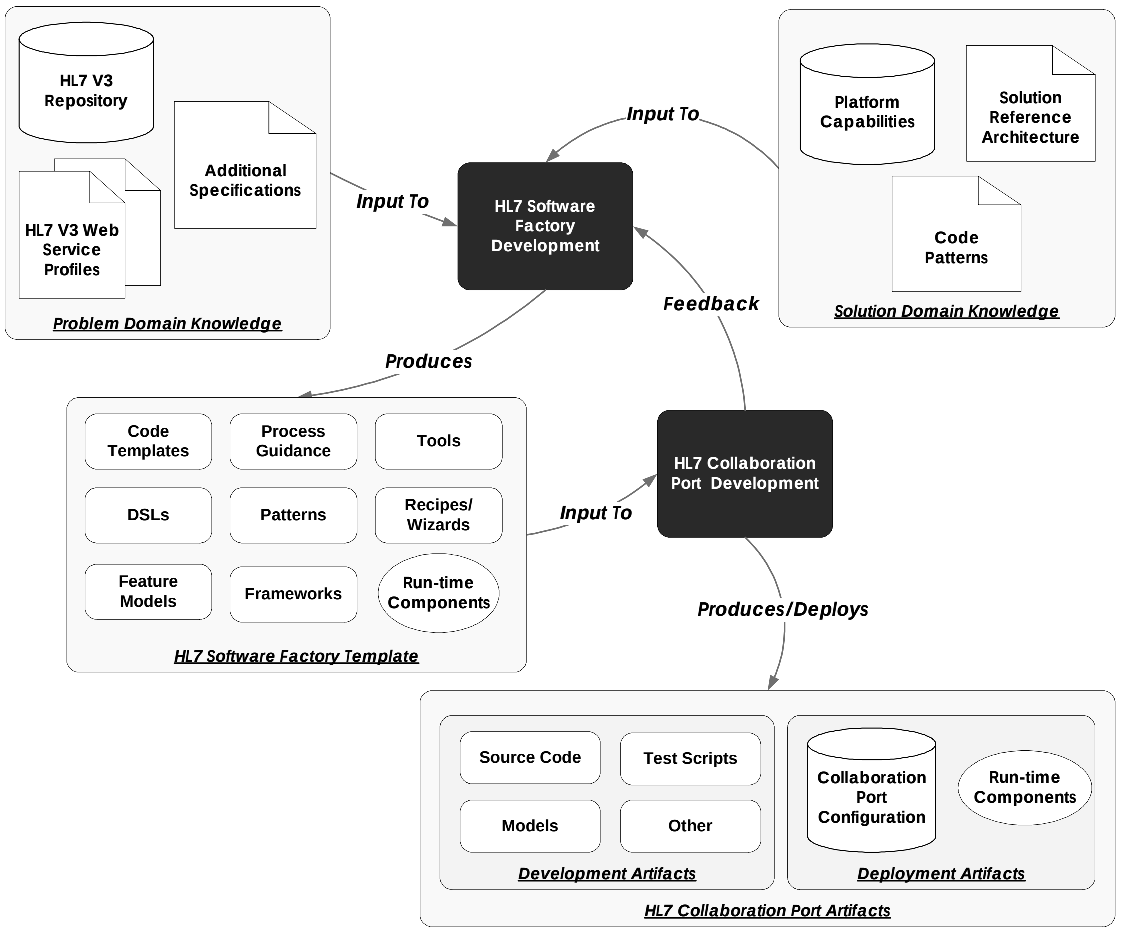


Figure 2 : HL7 Factory - Development Context

in this phase, organizing relevant *viewpoints* into a two dimensional matrix with level of abstraction on the vertical axis, and life cycle phase on the horizontal axis.

However, we quickly realized that the two-dimensional matrix was a fairly inadequate representation of the schema, especially for the fully scoped version of factory, because a) the schema was naturally multidimensional, b) a matrix representation does not capture relationships among non adjacent viewpoints, c) the graphs of viewpoints were of different types and depths, and unfold into nested graphs of different types and depths.

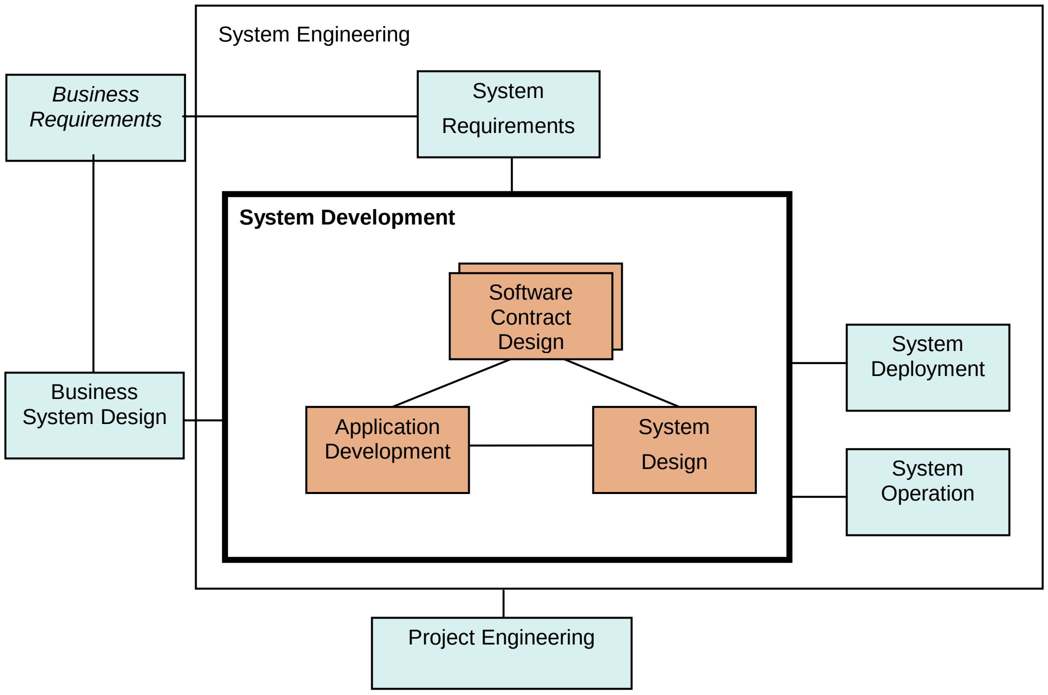


Figure 3 : System Development Viewpoint

Nonetheless, dealing with an amorphous graph-based representation of the schema required tools were not available. Therefore, we implemented the factory schema as a set of two dimensional projections of the relevant viewpoints. Each of these projections – a graph in its own right - details a specific aspect of the factory, effectively projecting it from the multidimensional schema in the same way that set of tuples is projected from a multidimensional data store to form a two dimensional view.

For example, Figure 3 and 4 show two viewpoint examples, namely the System Development viewpoint and one particular aspect of it: the Software Contract Design –at a lower abstraction level.

This approach allowed us to produce a version of the factory schema that we deemed complete, i.e., it comprehensively specified all the artifacts and tools required in the factory template to produce the products of the factory. However, it left verification of the schema to human inspection, and did not allow us to use the schema as metadata to drive the user experience within the IDE.

Configuration management was another challenging aspect of factory development, from two different perspectives.

First, we had to create a configuration XML schema, which would be complete in terms of allowing the expression of all valid combinations of supported features and/or implementation strategies. The XML schema was handcrafted and quickly became a maintenance burden, as it was highly sensitive to changes in the factory schema and template.

Second, we had no tools within the target development environment to support the configuration of a specific instance of the factory, or the validation of such a configuration, during product development.

Specification of the product development process was also a challenge in developing the Factory.

We had no satisfactory way to formally express the process in the factory template, and most importantly no way to inject the process as prescriptive guidance into the development environment.

Although the target development environment did support the creation of tasks, and the assignment of tasks to members of the development team, we had to rely upon natural language documents to describe the development process because we had not yet determined how to apply the task management features to a factory based product development process. In particular, we had not yet determined how to associate tasks with

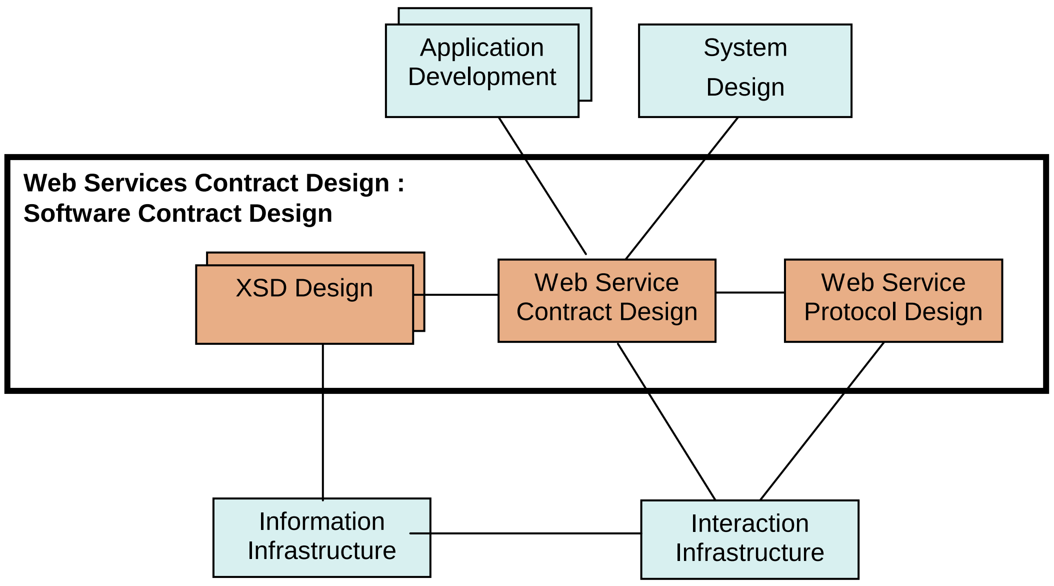

Application Development
System Design
Web Services Contract Design : Software Contract Design
XSD Design
Web Service Contract Design
Web Service Protocol Design
Information Infrastructure
Interaction Infrastructure

Figure 4 : Software Contract Design Viewpoint

specific assets supplied by the factory template, how to load the tasks from the factory template, or how to configure the tasks for a specific product.

We also found it quite hard to decide whether to provide or not a fully fledged domain specific language (DSL) for the requirements gathering phase of product development, especially given that Microsoft has published a set of DSL tools under the umbrella of its Software Factories Initiative.

Actually, we knew that a full DSL was not strictly necessary, because the relevant use case specifications were already available to us in HL7 repository. What we really needed was rather a sophisticated wizard that would help the user choose specific use cases (figure 5), application roles, service interaction patterns (figure 6), and other standard data elements, and navigate through the repository.

Also, the DSL designer technology from Microsoft was very immature when we started the project, and adopting it would have added significant risk to the project. Although we knew we were missing out on the opportunity to provide more sophisticated automation, and that the alternative, a wizard based user interface, would be not relevant outside the scope of the factory, we decided not to use the DSL technology in this phase of the project.

Given that we now have a much more consolidated and robust technology preview of DSL tools, we may start to experiment with DSLs in subsequent versions of the factory. Also, even if we did decide to create another wizard based user interface, we would very likely design and implement it using the DSL tools, instead of developing it from scratch.

The Guidance Automation Toolkit (GAT), another piece of enabling technology under the umbrella of the Software Factories Initiative at Microsoft, proved quite useful.

Simply stated, GAT is an extension to the development environment that makes it easy to create rich, integrated user experiences around reusable assets like frameworks, components and patterns. The resulting Guidance Packages are composed of templates, wizards and recipes, which help users build solutions in keeping with predefined architectural guidance.

In our project, we used GAT as the means of packaging and delivering the factory template. It provided an underlying model for the template that was much richer than what the development environment itself had to offer.

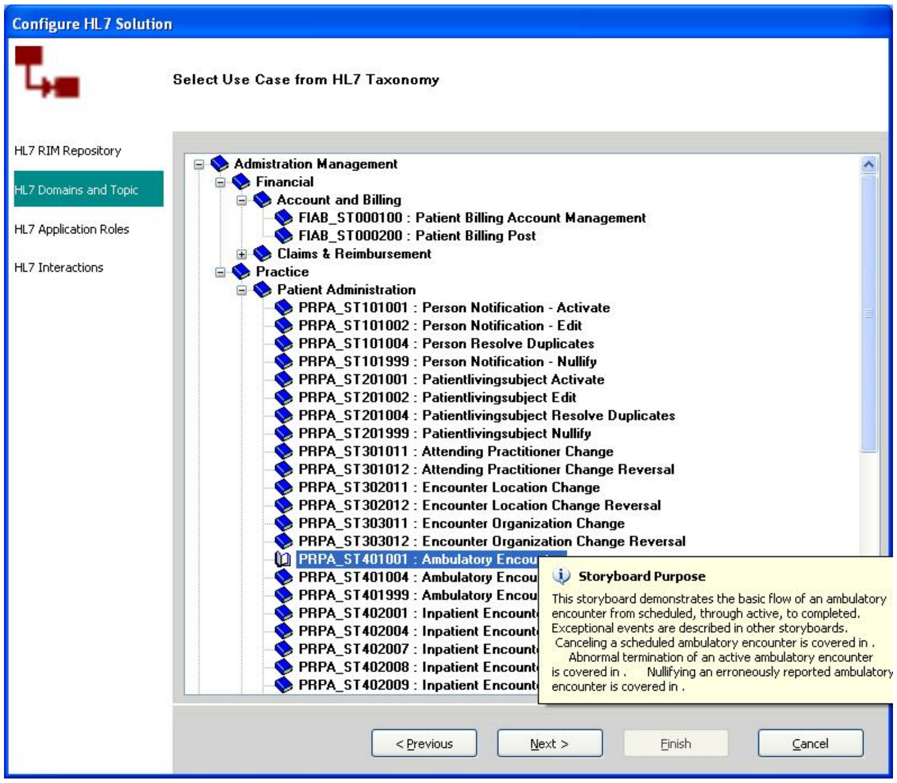
So, recipes have been provided for activities like: HL7 Web Services Adapter creation, execution of the configuration wizard, creation of Web Services contracts and automation of the code creation process.

Unfortunately, the GAT learning curve is quite steep. Its flexibility and customization options are limited and its integration with the IDE could have been better. Nevertheless, we believe that GAT adoption was a key decision that helped ensure project success and significantly reduced the factory development time.

4.2 The potential for generalization

Although we developed the factory to enable business to business collaboration among healthcare applications, it quickly became evident that we could apply such a factory to similar scenarios in other industries. We came to understand that the real scope of the factory should be business collaboration in general, using standardized or predefined specifications of interactions, application roles, events, Web Service profiles, and other domain elements, not just business collaboration in the context of HL7.

In retrospect, the reason we initially developed such a factory for HL7 was that the HL7 standard provided a complete set of easily accessible and well defined domain elements. Of course, many other standards organizations, such as RosettaNet and UN-CEFACT have also invested a great deal of effort in specifying domain elements to be used by businesses that want to collaborate using standardized protocols. Interestingly enough, as far as the



Configure HL7 Solution

Select Use Case from HL7 Taxonomy

- HL7 RIM Repository
- HL7 Domains and Topic
- HL7 Application Roles
- HL7 Interactions

- Admistration Management
  - Financial
    - Account and Billing
      - FIAB_ST000100 : Patient Billing Account Management
      - FIAB_ST000200 : Patient Billing Post
    - Claims & Reimbursement
  - Practice
    - Patient Administration
      - PRPA_ST101001 : Person Notification - Activate
      - PRPA_ST101002 : Person Notification - Edit
      - PRPA_ST101004 : Person Resolve Duplicates
      - PRPA_ST101999 : Person Notification - Nullify
      - PRPA_ST201001 : Patientlivingsubject Activate
      - PRPA_ST201002 : Patientlivingsubject Edit
      - PRPA_ST201004 : Patientlivingsubject Resolve Duplicates
      - PRPA_ST201999 : Patientlivingsubject Nullify
      - PRPA_ST301011 : Attending Practitioner Change
      - PRPA_ST301012 : Attending Practitioner Change Reversal
      - PRPA_ST302011 : Encounter Location Change
      - PRPA_ST302012 : Encounter Location Change Reversal
      - PRPA_ST303011 : Encounter Organization Change
      - PRPA_ST303012 : Encounter Organization Change Reversal
      - PRPA_ST401001 : Ambulatory Encou
      - PRPA_ST401004 : Ambulatory Encou
      - PRPA_ST401999 : Ambulatory Encou
      - PRPA_ST402001 : Inpatient Encount
      - PRPA_ST402004 : Inpatient Encount
      - PRPA_ST402007 : Inpatient Encount
      - PRPA_ST402008 : Inpatient Encount
      - PRPA_ST402009 : Inpatient Encount

Storyboard Purpose

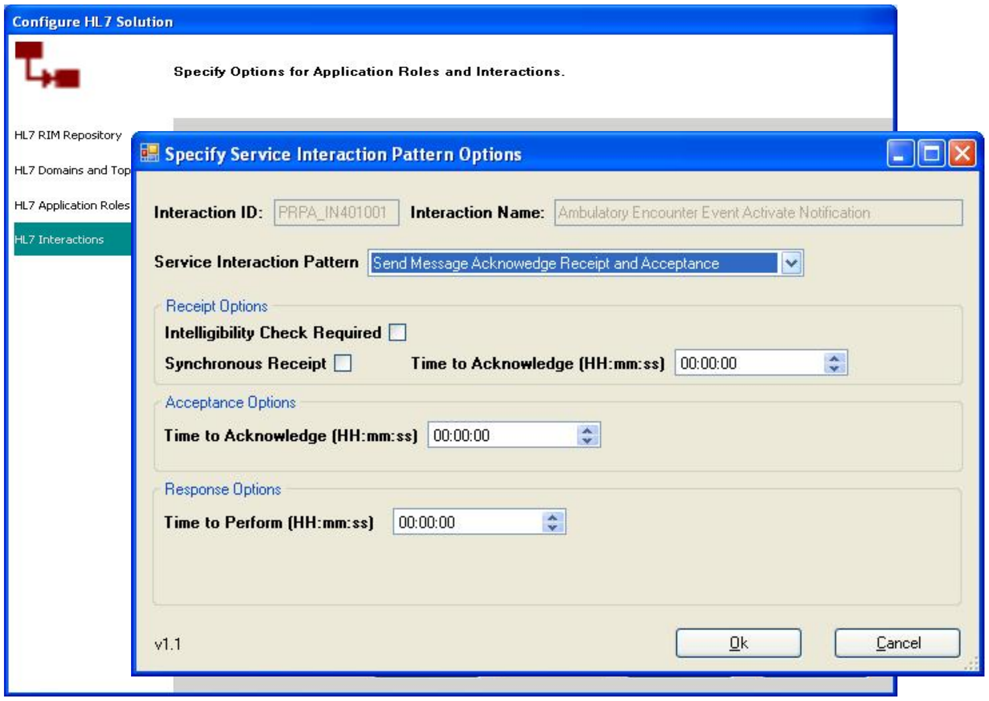
This storyboard demonstrates the basic flow of an ambulatory encounter from scheduled, through active, to completed. Exceptional events are described in other storyboards. Canceling a scheduled ambulatory encounter is covered in . Abnormal termination of an active ambulatory encounter is covered in . Nullifying an erroneously reported ambulatory encounter is covered in .

< Previous | Next > | Finish | Cancel

Figure 5 : Configuration Wizard - Use Case Selection

collaboration protocols and the information they exchange are concerned, these standards look very much alike.

We therefore concluded that it would be feasible not only to build standards based collaboration systems for other industries, but also to build a collaboration port factory that can



Configure HL7 Solution

Specify Options for Application Roles and Interactions.

- HL7 RIM Repository
- HL7 Domains and Top
- HL7 Application Roles
- HL7 Interactions

Specify Service Interaction Pattern Options

Interaction ID: PRPA_IN401001 Interaction Name: Ambulatory Encounter Event Activate Notification

Service Interaction Pattern: Send Message Acknowedge Receipt and Acceptance

Receipt Options

Intelligibility Check Required ☐

Synchronous Receipt ☐ Time to Acknowledge (HH:mm:ss) 00:00:00

Acceptance Options

Time to Acknowledge (HH:mm:ss) 00:00:00

Response Options

Time to Perform (HH:mm:ss) 00:00:00

v1.1

Ok | Cancel

Figure 6 : Configuration Wizard – Service Interaction Patterns Specification

be customized using business collaboration specifications for other industries. Of course, there will be a need for various import mechanisms, adapters and model conversions to deal with the different concepts used to describe business to business collaboration by different standards bodies. Nevertheless, we think a generic specification could be used to bridge these differences through configuration in a generic business collaboration factory.

At the same time, we recognize that a more generic factory may also be quite appealing to corporate developers constructing business collaboration systems inside the firewall, and who want a more formal approach to specifying and implementing those systems in order to guarantee better alignment between business goals and a portfolio of resulting IT services. However, in this case, we would need to provide enterprise architects with the models and tools necessary to support the specification process.

Finally, we realize that our factory must support the construction of collaboration ports for various technology platforms. We suspect that this could be accomplished using implementation patterns provided with the factory template to decouple the specification of the collaboration ports from platform specific implementation details.

4.2.1 Future Development Plans

Our plans are to exploit the opportunity described above to generalize the HL7 factory to form a Business Collaboration Factory. We think most of the work required to achieve this generalization will reside in the following areas:

- Provide models and tools to support the specification of business collaborations, focusing on information schema, business document/messages exchange protocols, and possibly business transactions;
- Define mappings between relevant industry standards and our internal model of business collaboration, and possibly tools for importing the domain elements they define;
- Introduce another level of configuration in the implementation of the collaboration port that will enable factory users to target a wider variety of technology platforms.

5. CONCLUSION

Our experience in developing a factory for HL7 collaboration ports has shown that we need to define better frameworks, tools and processes to specify the factory schema, to manage factory configuration in a flexible and extensible way, and to better understand how/when domain specific languages should be used. At the same time, initial implementations of extension mechanisms like the Guidance Automation Toolkit and DSL toolkit have proven their value, filling significant gaps in software factory infrastructure, and pointing to future innovation in that area.
We intend to continue to use and improve these tools in the next version of the HL7 Factory, as well as in the more generic – cross industry – version called the Business Collaboration Factory.

6. REFERENCES

[1] Mauro Regio, Jack Greenfield, Bernie Thuman - A Software Factory Approach To HL7 Version 3 Solutions http://www.microsoft.com/architecture/library.aspx?pid=think.integrate&abver=FEEB2E89-4412-4C58-A7F8-9B2CA0E0BDAC&id=http://msdn.microsoft.com/architecture/default.aspx?pull=/library/en-us/dnbda/html/HL7SoftFac.asp

[2] Mauro Regio - Web Services Enablement for Healthcare HL7 Applications - Web Services Basic Profile Reference Implementation; http://www.microsoft.com/architecture/library.aspx?pid=think.integrate&abver=E9A00024-3DC1-4B6A-BC20-22716E4D2FEA&id=msdn.microsoft.com/architecture/default.aspx?pull=/library/en-us/dnbda/html/HL7WebServApps.asp

[3] Roberto Ruggeri, Mauro Regio et al. - HL7 Advanced Web Service Profiles. http://www.hl7.org/v3ballot/html/welcome/environment/index.htm

[4] Greenfield and K. Short with S. Cook and S. Kent. - Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. Wiley. 2004

[5] Microsoft Enterprise Framework & Tools Group - Domain Specific Languages Toolkit http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx

[6] http://www.hl7.org

[7] http://www.isis.vanderbilt.edu/

[8] http://www.metacase.com/

[9] Microsoft Patterns and Practices Team - Guidance Automation Toolkit (GAT) http://lab.msdn.microsoft.com/teamsystem/workshop/gat/default.aspx