

LEVERAGING THE PIIM PROCESS TO ADVANCE HEALTH IT

**A ROADMAP FOR DEVELOPMENT:
THE PIIM CANONIC GUI MODEL SIMPLIFIES HL7 MESSAGING**

SOFTWARE DESIGN DESCRIPTION (VERSION 1.1)

PREPARED FOR:

**TELEMEDICINE AND ADVANCED TECHNOLOGY RESEARCH CENTER
(TATRC)**

DELIVERABLE UNDER AWARD NUMBER:

W81XWH-09-1-0456

AUTHOR:

MARINE KOSHKAKARYAN

REVIEWER:

NIMA BAHRAMI

DATE:

DECEMBER 21, 2012

Revision History

NAME	DESCRIPTION	DATE
Marine Koshkakaryan	Initial Draft	December 21, 2012
Nima Bahrami	Initial Review and Edit	December 21, 2012

Table of Contents

List of Tables & Figures.....	6
-------------------------------	---

I. INTRODUCTION

1.1 Purpose.....	7
1.2 Scope.....	7
1.3 References.....	7

II. DOCUMENT OVERVIEW

2.0 Document Overview	8
2.1 Section 3 — PIIM Canonical GUI Model (PCGM) Architectural Design.....	8
2.2 Section 4 — Common Data Access Layer (CAL).....	8
2.3 Section 5 — HL7 Version 3.0 (HL7 v3).....	8
2.4 Section 6 — PIIM Canonical GUI Model (PCGM).....	8
2.5 Section 7 — PCGM Java Plugin Prototype (PCGM-JPP).....	8

III. ARCHITECTURAL DESIGN

3.0 Architectural Design.....	9
3.1 Roles.....	9
3.1.1 CAL Developer (cal-developer).....	9
3.1.2 PCGM Model Developer (model-developer).....	9
3.1.3 PCGM IDE Plugin Developer (plugin-developer).....	9
3.1.4 GUI and PCGM Plugin User Developer(user-developer).....	9
3.2 Design-Time Use Case.....	9
3.3 Concept of Execution.....	10
3.4 Design-time Sequence Diagram.....	11
3.5 Run-time Sequence Diagram.....	12
3.6 Deployment Diagram.....	13

Table of Contents continued

IV. COMMON DATA ACCESS LAYER (CAL)

4.0	Common Data Access Layer (CAL).....	14
4.1	CAL Web Services (CAL-WS).....	14

V. HL7 V3 ANALYSIS

5.0	HL7 RIM Analysis.....	16
5.1	HL7 RIM Model Review.....	16
5.2	HL7 v3 Data Types.....	17

VI. PCGM DESIGN OVERVIEW

6.0	PCGM Design Overview.....	18
6.1	PCGM Design Considerations.....	19
6.1.1	Request Message Parameter Construction.....	19
6.1.2	Response Message Data Extraction.....	23
6.2	Data Restructure — PCGM DataType (PCGM-DT).....	24
6.3	The PCGM Configuration File.....	25

VII. PCGM JAVA PLUGIN PROTOTYPE (PCGM-JPP) IMPLEMENTATION EXAMPLE

7.0	PCGM Java Plugin Prototype (PCGM-JPP) Implementation Example.....	26
7.0.1	Creating a Web Services Client Project Using the NetBeans IDE Instructions.....	26
7.0.2	Creating a xJC Plugin, instructions.....	27
7.0.3	Java API sun.codemodel, API docs.....	27
7.0.4	The xJC Plugin task can be included in an ANT build, instructions. . .	27
7.1	PCGM Java Plugin Prototype (PCGM-JPP).....	27
7.2	Java Technologies Used by PCGM-JPP.....	27
7.2.1	JAX-WS Specifications.....	28

Table of Contents continued

7.2.2	XML Schema to Java Mapping.....	28
7.2.3	Schema-to-Java Mapping Data Types.....	32
7.3	PCGM-JPP Design.....	32
7.4	PCGM-JPP Code Explained.....	32
7.4.1	Request Parameter Object Construction.....	33
7.4.2	Response Parameter Data Extraction.....	33
7.5	PCGM JPP Results.....	34
VIII. MODEL FUTURE EXPANSION		
8.0	Model Future Expansion.....	36
8.1	User Interface Design — Plugin-Specific.....	36

List of Tables & Figures

TABLE 1	List of CAL Web Services Operations.	15
TABLE 2	Request Message Types with their Corresponding PCGM API Functions.	20
TABLE 3	Child Elements of Query Element.	21
TABLE 4	Child Elements of ControlActProcess Elements.	21
TABLE 5	Child Elements of QueryByParameter Element.	21
TABLE 6	Child Elements of ParameterList Elements.	22
TABLE 7	Child Elements of QueryByParameterPayload Element.	22
TABLE 8	Response Message Types with their Corresponding PCGM API Functions.	23
TABLE 9	PCGM Restructured Data Types.	24
TABLE 10	WSDL to Java Mapping.	28
TABLE 11	JAXB XML to Java.	28
TABLE 12	XML Schema and Java Data Types.	32
FIGURE 1	Design-time Use Case (link)	
FIGURE 2	Design-time Sequence Diagram (link)	
FIGURE 3	Run-time Sequence Diagram (link)	
FIGURE 4	High-level Deployment Diagram (link)	
FIGURE 5	High-level view of Common Data Access Layer Service (link)	
FIGURE 6	Components of Data Access Services (link)	
FIGURE 7	HL7 V3 RIM (link)	
FIGURE 8	PCGM (link)	
FIGURE 9	JAXBs XJC Compiler (link)	

1.0 Introduction

1.1 PURPOSE This document describes the design details of the Parsons Institute for Information Mapping Canonic GUI Model (PCGM).

1.2 SCOPE

The design of PCGM is derived from CAL Web Services (CAL-WS) and the Health Level 7 (HL7) Version 3.0 Reference Information Model (RIM). The main design goal of PCGM is to extend the CAL Web Services to suitable enterprise services for GUI usage on a variety of platform types (mobile, web, thick). This document also captures the development of a prototype plugin.

1.3 REFERENCES

Brull, Rob. "HL7 v3 RIM: Is It Really That Intimidating?" HL7standards.com. 31 May 2011. Web. July-Aug. 2012. <http://www.hl7standards.com/blog/2011/05/31/hl7-v3-rim-is-it-really-that-intimidating/>.

Chase, Nicholas. "Understanding Web Services, Part 2: Web Services Description Language (WSDL)." Understanding Web Services, Part 2: Web Services Description Language (WSDL). 7 July 2006. Web. Aug.-Sept. 2012. <http://www.ibm.com/developer-works/webservices/tutorials/ws-understand-web-services2/>.

Kulandai, Joseph. "JAXB Tutorial." JAXB Tutorial. 7 Aug. 2012. Web. Oct.-Nov. 2012. <http://javapapers.com/jee/jaxb-tutorial/>.

Pham, Kim. Data Access Service Software Design Description. Tech. no. 1.4. 2012. Print.

Van Luttikhuisen, Ronald. "Web Services Support in Oracle Enterprise Pack for Eclipse." Oracle.com. May 2009. Web. Aug. 2012. <http://www.oracle.com/technetwork/articles/java/oepe-web-services-support-166618.pdf>.

2.0 Document Overview

2.0 DOCUMENT OVERVIEW This document describes the design of a platform independent model PCGM, for a development acceleration tool that abstracts HL7. The Implementation Example section provides a detailed description of the development of a prototype that implements the model. This section summarizes what is covered in detail in later sections.

2.1 SECTION 3 — PIIM CANONICAL GUI MODEL (PCGM) ARCHITECTURAL DESIGN

Description of the system architecture depicts the components of the system and the interaction of the components through use case and sequence diagrams.

2.2 SECTION 4 — COMMON DATA ACCESS LAYER (CAL)

CAL provides a mechanism for accessing Electronic Health Record (EHR) data through the EHR API. CAL then converts the EHR data into HL7 Version 3.0 to be exchanged, through the CAL Web Services layer, with external entities. For further detail please see section 4 of this document and the Data Access Service Software Design Description document.

2.3 SECTION 5 — HL7 VERSION 3.0 (HL7 V3)

The HL7 V3 RIM was conceived as a universal reference model for healthcare interoperability covering the entire healthcare domain. As an essential part of the HL7 V3 object oriented development methodology, RIM provides an explicit representation of the semantic and lexical connections that exist between the information carried in the fields of HL7 messages. It specifies the grammar of the messages and, specifically, the basic building blocks of the language (nouns, verbs etc.), their permitted relationships and Data Types.

2.4 SECTION 6 — PIIM CANONICAL GUI MODEL (PCGM)

The PCGM is a roadmap for developing a tool, that auto-generates helper code based on the CAL implementation of HL7 messages. This section describes the tool in detail and provides a complete list of the PCGM API functions.

2.5 SECTION 7 — PCGM JAVA PLUGIN PROTOTYPE (PCGM-JPP)

Parallel to the model development a prototype implementing the PCGM was also underway for proof of concept, and further discovery that in turn informed the model.

3.0 Architectural Design

3.1 ROLES

Throughout the development of the model and the prototype, several developer roles emerged. These roles may change and evolve as the model matures. Although one individual may fulfill multiple roles, the following four separate areas of responsibility were identified.

3.1.1 CAL DEVELOPER (CAL-DEVELOPER)

The CAL developer is responsible for the development and maintenance of the CAL-WS component. He/she maintains the CAL-WS definition language (WSDL) and the configuration file required by PCGM.

3.1.2 PCGM MODEL DEVELOPER (MODEL-DEVELOPER)

The PCGM model developer is responsible for the development and the maintenance of the model described in this

document. He/she is responsible for new additions and changes to the model.

3.1.3 PCGM IDE PLUGIN DEVELOPER (PLUGIN-DEVELOPER)

The plugin developer's task is to develop and maintain the tool in a given IDE and platform. The tool may be an IDE Plugin and shall follow the architecture and design guidelines provided by the PCGM Model Developer.

3.1.4 GUI AND PCGM PLUGIN USER DEVELOPER (USER-DEVELOPER)

The user developer is the CAL-WS Client developer who is using the PCGM based tool courtesy of the plugin-developer to aid in connecting the GUI objects in the application he/she is developing to the data returned by the CAL-WS operation.

3.2 DESIGN-TIME USE CASE

The following use case provides a high-level view of the system components at design-time; presenting actor to component and component to component interaction (see figure 1).

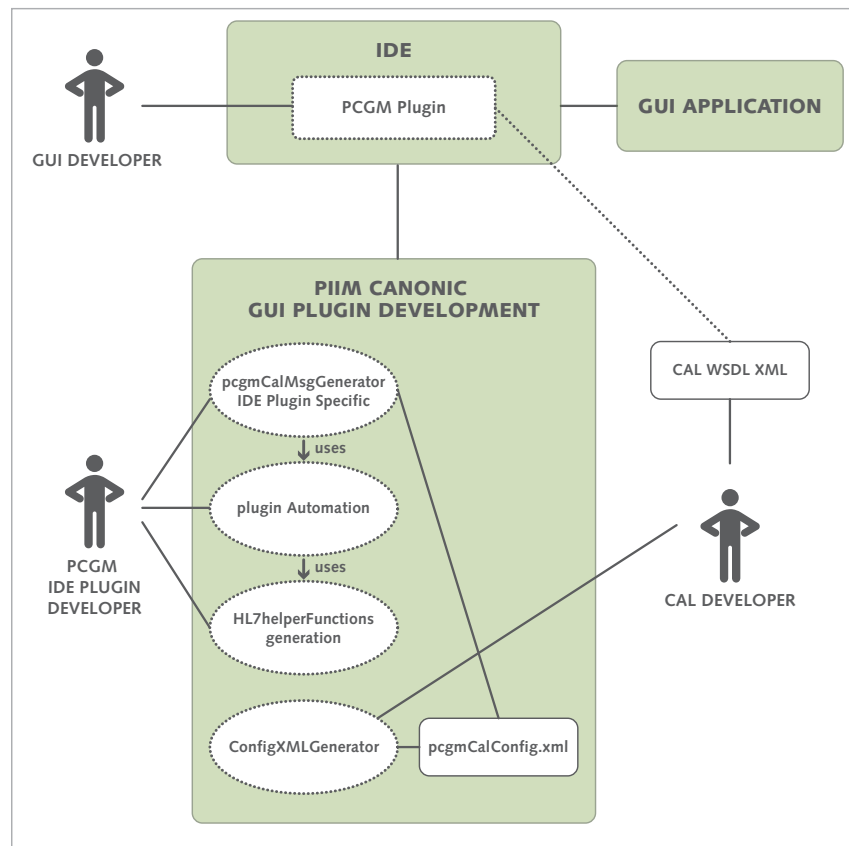


FIGURE 1:
Design-time
Use Case

3.3 CONCEPT OF EXECUTION

Process flows as follows:

1. The cal-developer provides the URL for the CAL WSDL.
2. The cal-developer includes the PCGM configuration file in a designated location on the CAL web server for the PCGM plugin to retrieve it.
3. The cal-developer includes the restructured PCGM data type schema with the CAL web services schema.
4. The user-developer invokes the PCGM Plugin.
5. The PCGM plugin includes a call to the IDE plugin function that handles Web Services (ws) client artifact generation; passing it the CAL WSDL URL.
6. The IDE ws plugin handles the artifact generation.
7. The PCGM plugin retrieves the PCGM configuration file stored on the CAL web server relative to the WSDL location.
8. The PCGM plugin loops through the CAL HL7 schema or java artifacts to obtain the class hierarchy of the objects.
9. The PCGM plugin references the PCGM configuration file to retrieve additional information about the HL7 objects to be constructed and the functions to generate for working with the objects.
10. The PCGM plugin uses the PCGM Data Types to construct return values for the helper functions it generates.
11. The PCGM plugin saves the helper classes it generates in the file system in relative to the location of the project it was invoked through.
12. The user-developer calls a PCGM Plugin helper function to construct an HL7 object of request type
13. The user-developer invokes a CAL-ws operation to request data from an underlying EHR
14. The user-developer passes the HL7 object from step 2 as a request message parameter
15. Common Data Layer Web Services (CAL-ws) returns the HL7 formatted outbound data to the calling entity
16. The user-developer passes the returned HL7 object to a PCGM Plugin function to extract data
17. The PCGM Plugin function returns restructured data to the user-developer

3.4 DESIGN-TIME SEQUENCE DIAGRAM

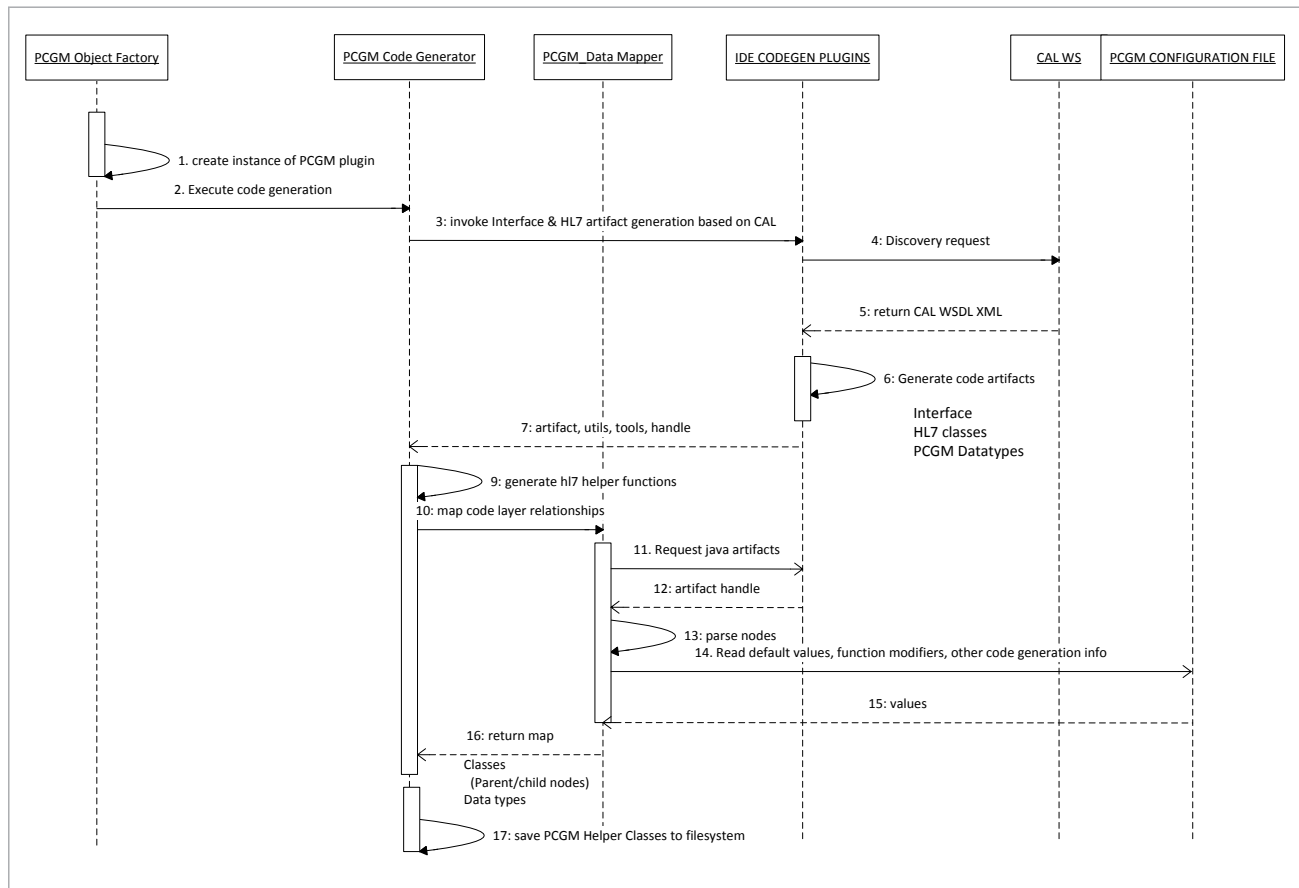


FIGURE 2: *Design-time Sequence Diagram*

This diagram depicts the process of creating an instance of the PCGM plugin in conjunction with the CAL-WS client.

3.5 RUN-TIME SEQUENCE DIAGRAM

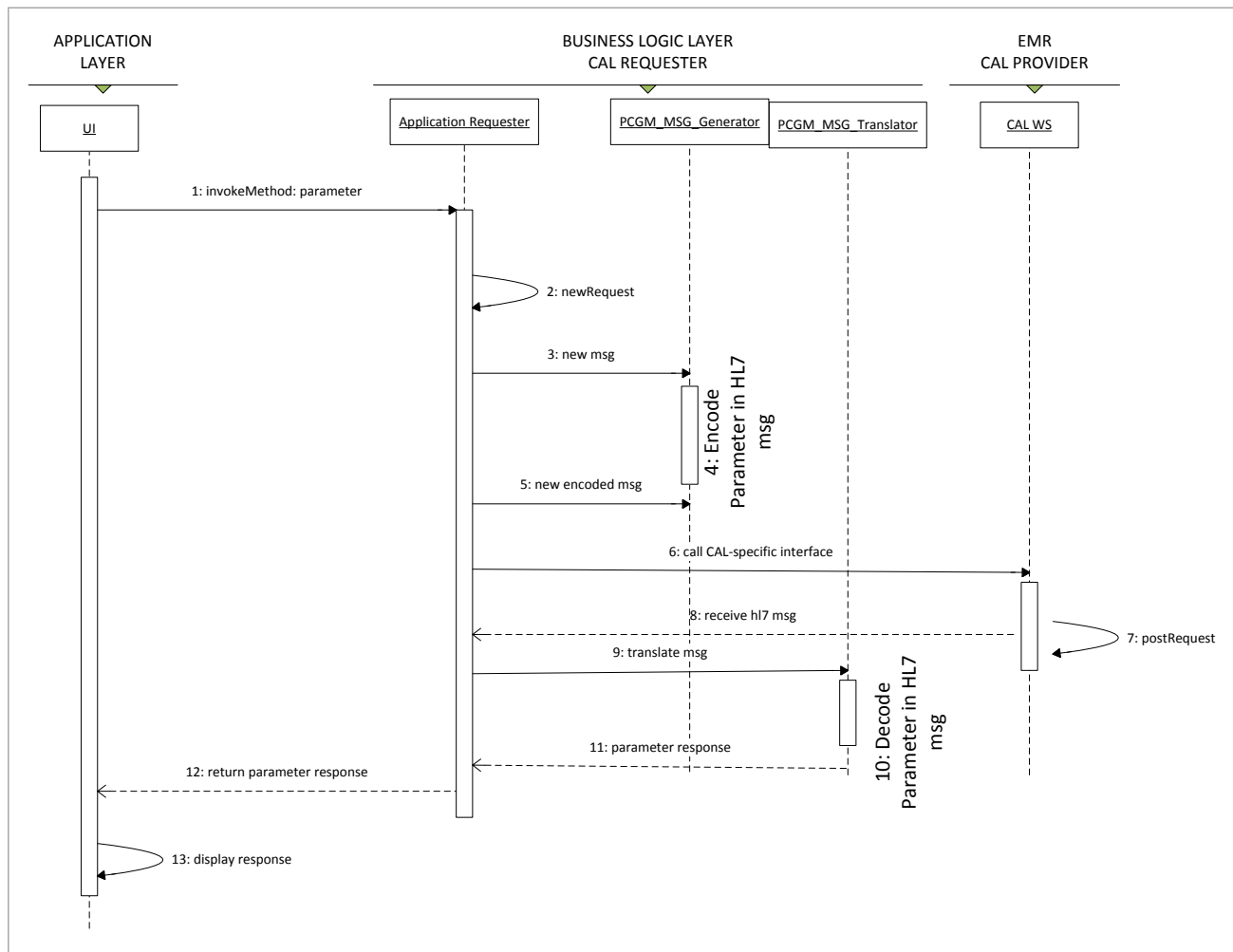


FIGURE 3: Run-time Sequence Diagram

The diagram, shown above, illustrates how a single message request and response moves through the three layers of the system.

3.6 DEPLOYMENT DIAGRAM

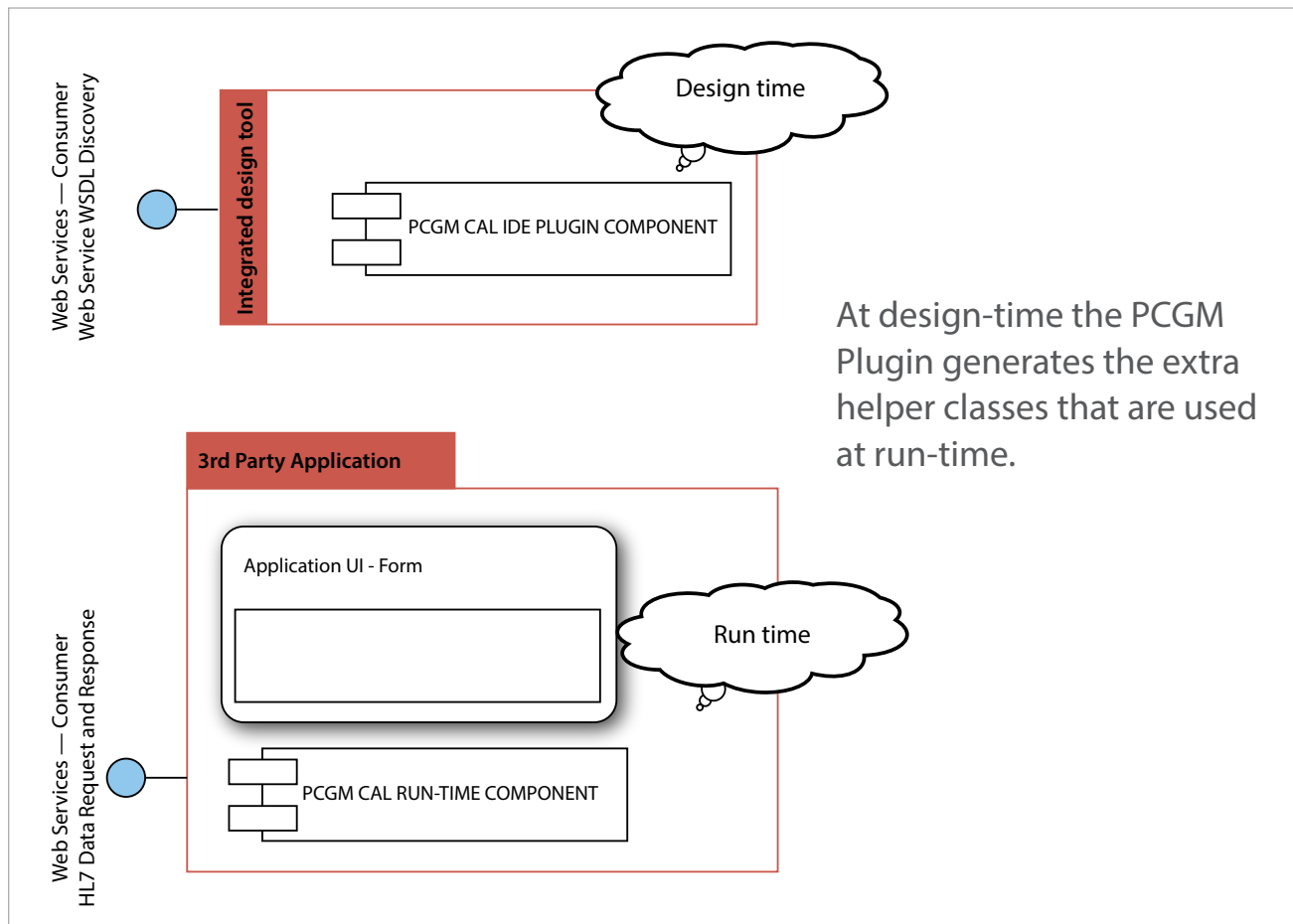


FIGURE 4: *High-level Deployment Diagram*

The deployment diagram depicts, at a high-level, where the PCGM IDE plugin resides in the larger system. It communicates the simple fact that the PCGM plugin generates helper code at design-time, which then is used by the user-developer at run-time.

4.0 Common Data Access Layer (CAL)

4.0 COMMON DATA ACCESS LAYER (CAL)

An entity, such as a front-end technology, may access data from any EHR connected to CAL through CAL's Web Services layer. CAL provides a more interoperable environment where data exchange and interpretation occurs by utilizing the HL7 standard.

- CAL provides access to the underlying EHR system(s) through their proprietary Open API's
- CAL maps EHR specific data to the appropriate HL7 RIM canonical model(s)
- CAL provides any terminologies and semantics translation as required

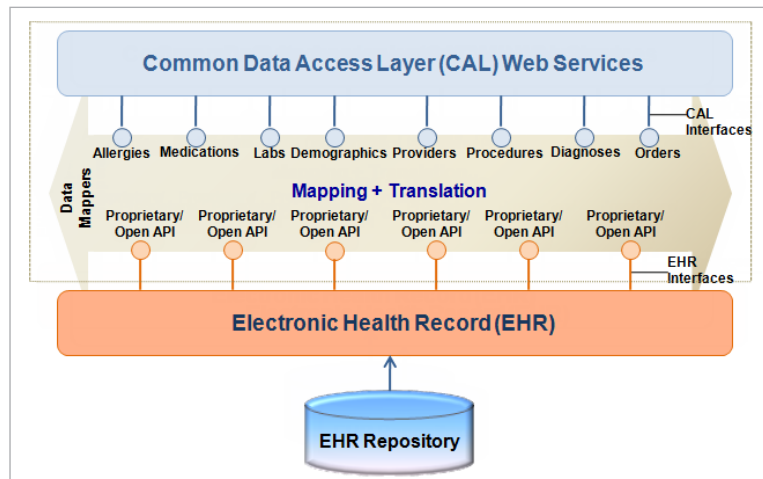


FIGURE 5: High-level of Common Data Access Layer Service.
SOURCE: Pham, p. 4 (2012).

CAL COMPONENTS:

COMPONENT	DESCRIPTION
Common Data Access Layer Web Services	A WSDL defining a set of web service operations available for external entities to request EHR clinical data and an API interface. A set of schemas identifying the structures of the input and output messages.
Data Mappers Library	A set of classes implementing data retrieval from EHR system and mediate its data into appropriate HL7 RIM messages for delivery to the calling entity. For example: An AHLTA data mapper class implemented Allergies domain will map allergies data retrieved from Clinical Data Repository (CDR) into a collection of HL7 RIM data elements and translated any terminologies from 3M coding system to the HL7 coding system(s).
Access Control	A security component handles any access control processing required by the EHR system.

FIGURE 6: Components of Data Access Service
SOURCE: Pham, p. 4 (2012).

4.1 CAL WEB SERVICES (CAL-WS)

The CAL Web Services Definition Language (WSDL) follows the W3C recommendation. It consists of the portType CommonDataLayer-PortType and operations with input and output messages. Each message has one part element that identifies the 'parameter' the message transports. CAL-WS WSDL:



The following table lists all the available CAL Web Services (CAL-WS) operations as of the date of this report, the message parameters as defined in the WSDL, and the corresponding HL7 data types mapped from the schema.

TABLE 1: LIST OF CAL WEB SERVICES OPERATIONS

CAL WS Operations	CAL Operation Request & Response Messages	CAL MessagesParameter Names	CAL Parameter Types
FindEncounters	INPUT	"FindEncountersRequest"	FindEncounters_PRPA_IN900300UV02Request
	OUTPUT	"FindEncountersResponse"	FindEncounters_PRPA_IN900350UV02Message
GetProcedures	INPUT	"GetProceduresRequest"	CareRecord_QUPC_IN043100UV01ProceduresRequest
	OUTPUT	"GetProceduresResponse"	CareRecord_QUPC_IN043200UV01Message
GetMedications	INPUT	"GetMedicationsRequest"	CareRecord_QUPC_IN043100UV01MedicationsRequest
	OUTPUT	"GetMedicationsResponse"	CareRecord_QUPC_IN043200UV01Message
GetAllergies	INPUT	"GetAllergiesRequest"	CareRecord_QUPC_IN043100UV01AllergiesRequest
	OUTPUT	"GetAllergiesResponse"	CareRecord_QUPC_IN043200UV01Message
GetVitals	INPUT	"GetVitalsRequest"	CareRecord_QUPC_IN043100UV01VitalsRequest
	OUTPUT	"GetVitalsResponse"	CareRecord_QUPC_IN043200UV01Message
GetProblems	INPUT	"GetProblemsRequest"	CareRecord_QUPC_IN043100UV01ProblemsRequest
	OUTPUT	"GetProblemsResponse"	CareRecord_QUPC_IN043200UV01Message
GetTestResults	INPUT	"GetTestResultsRequest"	CareRecord_QUPC_IN043100UV01TestResultsRequest
	OUTPUT	"GetTestResultsResponse"	CareRecord_QUPC_IN043200UV01Message
GetImagingResults	INPUT	"GetImagingResultsRequest"	CareRecord_QUPC_IN043100UV01ImagingResultsRequest
	OUTPUT	"GetImagingResultsResponse"	CareRecord_QUPC_IN043200UV01Message
GetImmunizations	INPUT	"GetImmunizationsRequest"	CareRecord_QUPC_IN043100UV01ImmunizationsRequest
	OUTPUT	"GetImmunizationsResponse"	CareRecord_QUPC_IN043200UV01Message
GetFamilyHistory	INPUT	"GetFamilyHistoryRequest"	CareRecord_QUPC_IN043100UV01FamilyHistoryRequest
	OUTPUT	"GetFamilyHistoryResponse"	CareRecord_QUPC_IN043200UV01Message
GetSocialHistory	INPUT	"GetSocialHistoryRequest"	CareRecord_QUPC_IN043100UV01SocialHistoryRequest
	OUTPUT	"GetSocialHistoryResponse"	CareRecord_QUPC_IN043200UV01Message
GetInsurances	INPUT	"GetInsurancesRequest"	CareRecord_QUPC_IN043100UV01Request
	OUTPUT	"GetInsurancesResponse"	CareRecord_QUPC_IN043200UV01Response
GetOrders	INPUT	"GetOrdersRequest"	CareRecord_QUPC_IN043100UV01OrdersRequest
	OUTPUT	"GetOrdersResponse"	CareRecord_QUPC_IN043200UV01Message
GetPatientInfo	INPUT	"GetPatientInfoRequest"	PatientDemographics_PRPA_IN201307UV02Request
	OUTPUT	"GetPatientInfoResponse"	PatientDemographics_PRPA_MT201303UVResponse
FindPatients	INPUT	"FindPatientsRequest"	FindPatients_PRPA_IN201305UVRequest
	OUTPUT	"FindPatientsResponse"	FindPatients_PRPA_MT201310UVResponse
FindProviders	INPUT	"FindProvidersRequest"	ProviderDetails_PRPM_IN306010UV1Request
	OUTPUT	"FindProvidersResponse"	ProviderDetails_PRPM_IN306011UV01Response
AppointmentSlotRequest	INPUT	"AppointmentBySlotRequest"	SlotRequest_PRSC_IN030101UVMessage
	OUTPUT	"AppointmentBySlotResponse"	SlotConfirmation_PRSC_IN030102UVMessage
FindResultEventQuery	INPUT	"FindResultEventQueryRequest"	ResultQuery_POLB_IN354000UV01Message
	OUTPUT	"FindResultEventQueryResponse"	ResultEvent_POLB_IN364000UV01Message
LabOrderRequest	INPUT	"CompositeLabOrderRequest"	CompositeOrder_POOR_IN200901UVLabRequest
	OUTPUT	"CompositeLabOrderResponse"	ActReference_POOR_IN200911UVMessage
GetAmbEncounterDetails	INPUT	"GetAmbEncounterDetailsRequest"	AMBCareRecord_QUPC_IN040100UV01Request
	OUTPUT	"GetAmbEncounterDetailsResponse"	CareRecord_QUPC_IN040200UV01Message
GetImpEncounterDetails	INPUT	"GetImpEncounterDetailsRequest"	IMPCareRecord_QUPC_IN040100UV01Request
	OUTPUT	"GetImpEncounterDetailsResponse"	CareRecord_QUPC_IN040200UV01Message
FindSlots	INPUT	"FindSlotsRequest"	SlotsQuery_PRSC_IN030101UVRequest
	OUTPUT	"FindSlotsResponse"	SlotsQueryResponse_PRSC_IN030102UVMessage
FindDocumentWithContent	INPUT	TBD	FindDocumentWithContent_RCMR_IN000031UV01Request
			tbd

The following document, created at the beginning of the PCGM project, assists developers become familiar with CAL-WS operations by mapping GUI user actions to the CAL message parameters utilized throughout the project.



5.0 HL7 V3 Analysis

5.0 HL7 V3 ANALYSIS This section provides a short introduction to the HL7 v3 RIM and Data Types. For further information on CAL's use of HL7 v3 please refer to Data Access Service Software Design Description document.

5.1 HL7 RIM MODEL REVIEW

The Reference Information Model (RIM) is an information model for health care data developed by Health Level 7

International (HL7). Based on the Unified Modeling Language (UML), the Reference Information Model consists of a generic set of classes from which more specific health classes are derived. For example, sub-classes of the class "act" include observation and procedure.

The RIM is a generic, abstract model that expresses information for the entire healthcare realm. The Domain Message Information Model (DMIM) is a subset of the RIM constrained to provide information relevant to a specific domain within healthcare, such as Patient Administration. The Refined Message Information Model (RMIM) is further constrained to define the contents of a specific message within the DMIM.

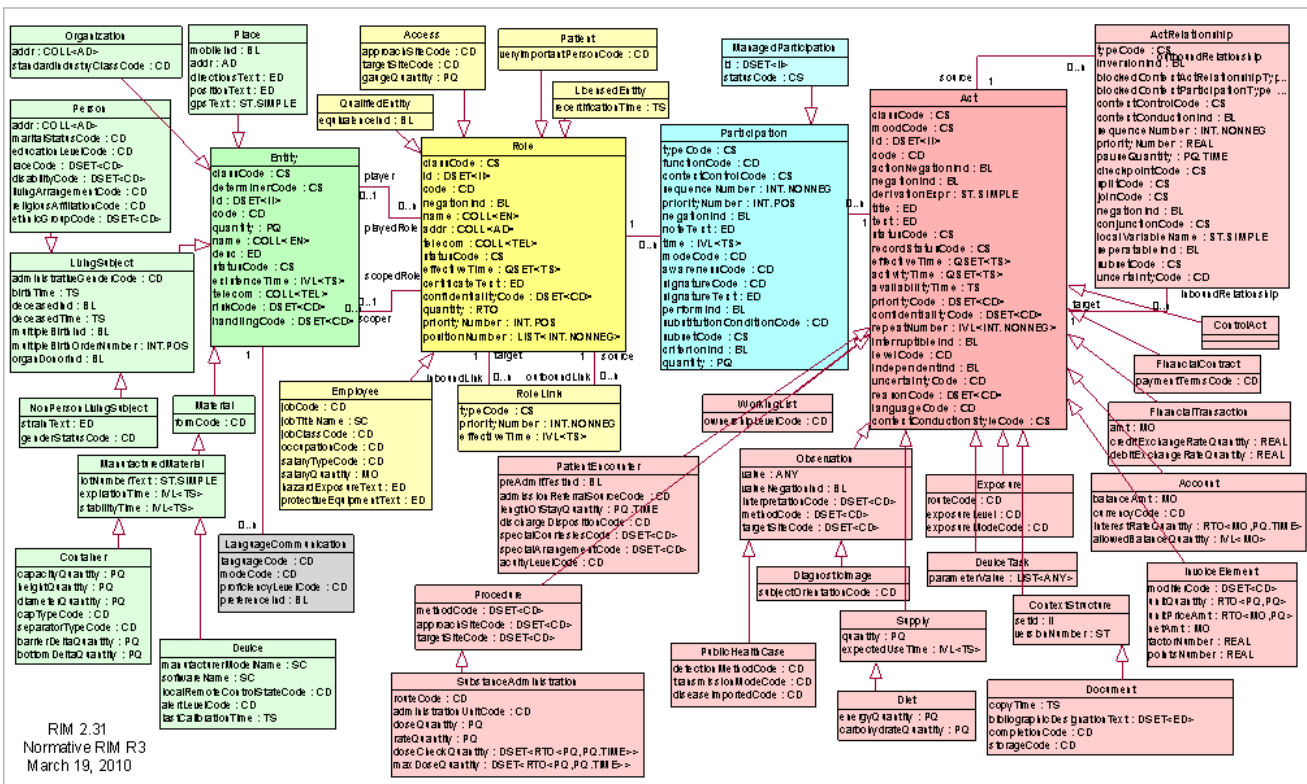


FIGURE 7: HL7 v3 RIM

SOURCE: Brull, (2011).

5.2 HL7 V3 DATA TYPES

New to v3 is Data Type Specification.

- Data types are the basic building blocks of attributes
- Data types define the meaning (semantics) of data values that can be assigned to a data element
- Meaningful exchange of data requires knowledge of the definition of values exchanged
- Every attribute in the RIM is associated with one and only one data type, and each data type is associated with zero or many attributes
- Data types in HL7 v3 are complex: Each data type has attributes and each data type attribute has a data type of its own

Coded data types are used heavily and are a good example of the complexity they introduce. They all extend the type ANY. CD extends ANY and most of the others, CE, CV, CS, CO restrict CD. CR adds a role-relationship qualifier. ANY does not have elements itself; it only offers the attribute 'null flavor'. To further demonstrate complexity, the CD data type has a child element called 'translation' which uses the type CD to define a translation of the data into a different code system

6. PCGM Design Overview

6. PCGM DESIGN OVERVIEW The PCGM design process began with review of exiting, best-in-class, design patterns used in the software industry for code generation and workflow automation. In parallel a prototype development project was designed to study the existing toolset for code generation. The combination of the two parallel efforts resulted in the creation of PCGM model's first draft. The following diagram represents this model.

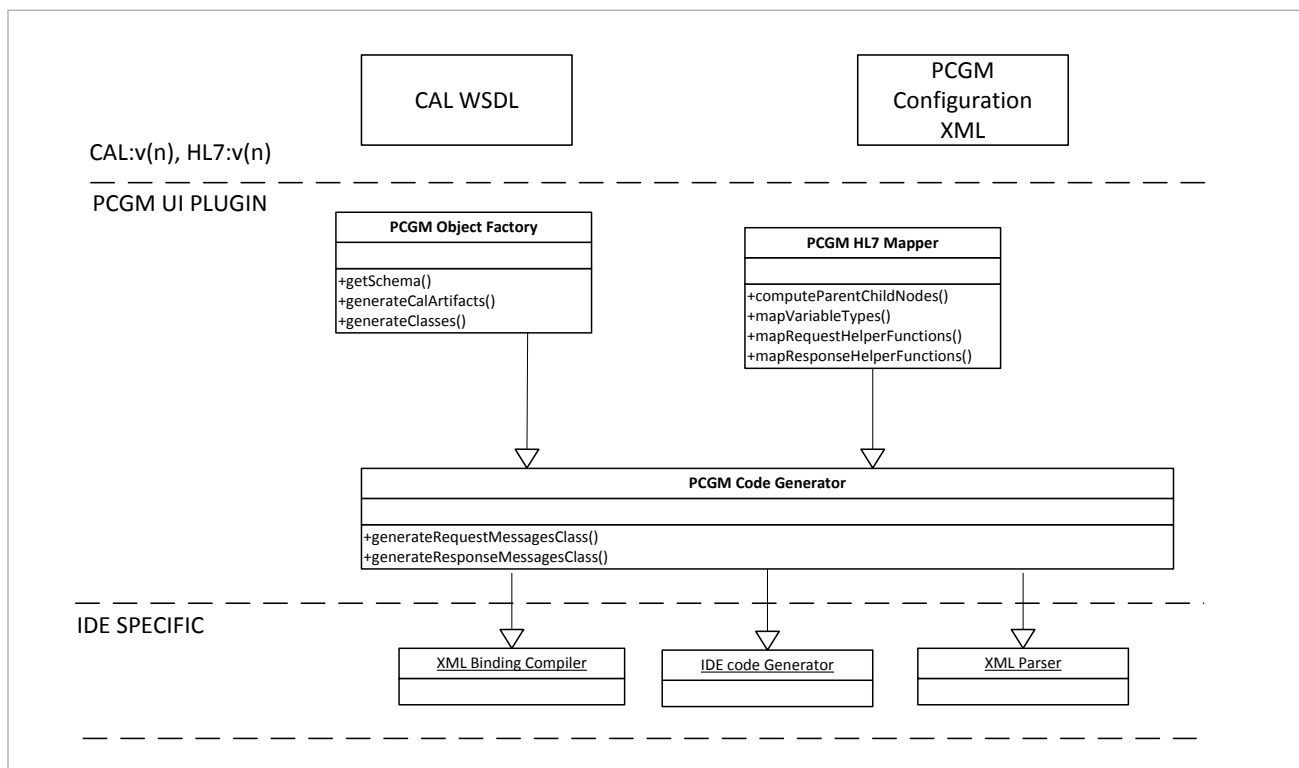


FIGURE 8: PCGM

The model consists of the following components:

1. IDE Specific Components

These components provide standard services such as web services binding, xml parsing, and code generation. All IDE's provide instances of such services and can be easily accessed within the IDE.

2. PCGM UI PLUGIN Components:

These components provide CAL and HL7 specific services for code generation. In the current version of the model, three classes have been identified: PCGM Object Factory, HL7 Mapper, and Code Generator.

PCGM Object Factory handles object instantiation, execution, and destruction. PCGM Code Generator provides services that assist in automatic generation of HL7 helper functions, and libraries required for abstraction of HL7 messaging. PCGM HL7 Mapper provides services that will assist in parsing and mapping artifacts necessary during execution. This mapping is vital in assisting code generation logic with creation of the right function and parameter hierarchies in real-time during execution.

3. CAL/HL7/PCGM Configuration files:

These provide essential meta-data in XML format that define interfaces, message classes, data types, and all other necessary content that assist in auto-generation of code artifacts, and helper functions. The model suggests the importance of versioning for these files. This will be subject of future expansion of the model and therefore out of scope for this phase of the study.

6.1 PCGM DESIGN CONSIDERATIONS

A developer, referred to as user-developer, creates a CAL-WS client and uses the PCGM based tool. The proposed tool is an IDE plugin.

The plugin generates the helper classes. A single class is generated with multiple helper functions to construct request message parameters. Many classes, one for each response type, are generated to [extract data from the response message parameter](#).

This section examines the CAL-WS operations and the elements within the messages; looking at each element in a request message and suggesting the appropriate source for the value. An analysis of the GUI relevance of each element in a response message, and a sample of suggested a restruc-

tured canonical data model is provided in section 6.2. The restructured model strips the data of the HL7 hierarchy, offering instead a flat representation with primitive types.

6.1.1 REQUEST MESSAGE PARAMETER CONSTRUCTION

The user-developer calls the helper functions and without any knowledge of the underlying HL7 infrastructure passes a PCGM required parameter value to create the HL7 object. He/she then passes the object returned by the helper function to the CAL-WS operation.

CAL uses various message types from the HL7 RIM domain context as request messages. A typical request message contains a long list of data elements, most of which do not need to be set dynamically since they are not unique to the message instance.

The following table contains the list of CAL-WS operations, the parameter (HL7 type) the input (request) message requires, and the final column displays the PCGM request helper functions that the user-developer can call to construct the corresponding parameter. The plugin-developer may use the table as a reference for what functions the PCGM request helper class should provide and what data the functions require to carry out the task. The user-developer may use the table to check what functions the PCGM request helper class provides and what data he needs to pass as parameter(s).

TABLE 2: REQUEST MESSAGE TYPES WITH THEIR CORRESPONDING PCGM API FUNCTIONS

CAL WS Operations	CAL RequestParameter Types	PCGM Request Helper Function Call
FindEncounters	(constructed and returned by the corresponding PCGM function) FindEncounters_PRPA_IN900300UV02RequestType	pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusAll", String "EncounterTypeAll") pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusAll", String "EncounterTypeAmbulatory") pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusAll", String "EncounterTypeEmergency") pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusAll", String "EncounterTypeField") pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusAll", String "EncounterTypeHome") pcgmHelperRequests.createRequestFindEncountersPRPAIN900300UV02RequestType (String patientId, String "EncounterStatusActiveOnly")
GetProcedures	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Procedures")
GetMedications	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "MedicationsAll") pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "MedicationsOnlyActive") pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "MedicationsDischarge") pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "MedicationsHistory")
GetAllergies	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Allergies")
GetVitals	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Vitals")
GetProblems	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Problems")
GetTestResults	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "TestResults")
GetImagingResults	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "ImagingResults")
GetImmunizations	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Immunizations")
GetFamilyHistory	tbd	tbd
GetSocialHistory	tbd	tbd
GetInsurances	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Insurances")
GetOrders	CareRecord_QUPC_IN043100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN043100UV01RequestType (String patientId, String "Orders")
GetPatientInfo	PatientDemographics_PRPA_IN201307UV02RequestType	pcgmHelperRequests.createRequestPatientDemographicsPRPAIN201307UV02RequestType (String patientId)
FindPatients	FindPatients_PRPA_IN201305UVRequestType	pcgmHelperRequests.createRequestFindPatientsPRPAIN201305UVRequestType (String patientLastName, String patientFirstName, String patientDateOfBirth, String patientGender, String patientZipCode, String patientHomePhone)
FindProviders	ProviderDetails_PRPM_IN306010UV1RequestType	pcgmHelperRequests.createRequestProviderDetailsPRPMIN306010UV1RequestType (String providerId, String providerLastName, String providerFirstName, String locationId)
AppointmentSlotRequest	SlotRequest_PRSC_IN030101UVMMessageTypes	pcgmHelperRequests.createRequestSlotRequestPRSCIN030101UVMMessageTypes (String appointmentSlotId)
FindResultEventQuery	ResultQuery_POLB_IN354000UV01MessageTypes	
LabOrderRequest	CompositeOrder_POOR_IN200901UVRequestType	
GetAmbEncounterDetails	CareRecord_QUPC_IN040100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN040100UV01RequestType (String patientId, String recordId)
GetImpEncounterDetails	CareRecord_QUPC_IN040100UV01RequestType	pcgmHelperRequests.createRequestCareRecordQUPCIN040100UV01RequestType (String patientId, String recordId)
FindSlots	SlotRequest_PRSC_IN030101UVMMessageTypes	pcgmHelperRequests.createRequestSlotQueryPRSCIN030101UVMMessageTypes (String providerId)
FindDocumentWithContent	tbd	tbd

PCGM defines three data sources for the values assigned to the elements of a given request message parameter:

1. The user-developer provides the parameters identified in the above table. This is the dynamic data that is specific to the message instance.
2. There is data set at run-time by the plugin code such as unique message id and time stamps.
3. The third source is the plugin configuration file that is provided with the CAL-WS WSDL. It can be

found on the CAL web server in a location relative to the CAL-WS WSDL. The configuration file provides the default data that may vary by message type but does not change per instance.

Following tables 3 to 7 list the elements of the request messages and what the data source for each element is.

Most request messages used by CAL-WS encapsulate the payload in the query element. The query element contains the child elements listed in table 3.

TABLE 3: CHILD ELEMENTS OF QUERY ELEMENT

ELEMENT	VALUE SOURCE
id	Plugin creates a unique identifier for message instance
creationTime	Plugin creates a timestamp for message instance
interactionId	Plugin configuration file contains the interaction identifier for the specific message type
processingCode	Plugin configuration file contains the default code specific for the message type
processingModeCode	Plugin configuration file contains the default code specific for the message type
acceptAckCode	Plugin configuration file contains the default code specific for the message type
receiver	Plugin configuration file contains the default values within this coded type, specific for the message type (good candidate for a global variable set by a plugin GUI wizard)
sender	Plugin configuration file contains the default values within this coded type, specific for the message type (good candidate for a global variable set by a plugin GUI wizard)
controlActProcess classCode="CACT" moodCode="RQO"	Plugin configuration file contains the default codes for the element attributes, specific for the message type See Table 4 for Child Elements of controlActProcess

Elements of type controlActProcess contain the child elements listed in table 4.

TABLE 4: CHILD ELEMENTS OF CONTROLACTPROCESS ELEMENT

ELEMENT	VALUE SOURCE
id	Not required – not used
code	Plugin configuration file contains the code specific for the message type
priorityCode	Plugin configuration file contains the default code specific for the message type
responsePriorityCode	Plugin configuration file contains the default code specific for the message type
queryByParameter OR <i>queryByParameterPayload</i>	See Table 5 for Child Elements of queryByParameter OR See Table 7 for Child Elements of queryByParameterPayload

TABLE 5: CHILD ELEMENTS OF QUERYBYPARAMETER ELEMENT

ELEMENT	VALUE SOURCE
queryId	Plugin helper creates a timestamp for message instance that may be used as the query identifier OR Plugin creates a unique identifier for message instance that may be used as the query identifier
statusCode	Plugin configuration file contains the default code specific for the message type
responseModalityCode	Not required – not used, however a default value may be added to the plugin configuration file
responsePriorityCode	Plugin configuration file contains the default code specific for the message type
<i>parameterList</i>	See Table 6 for Child Elements of parameterList

TABLE 6: CHILD ELEMENTS OF PARAMETERLIST ELEMENT

ELEMENT	VALUE SOURCE
careProvisionCode	Parameter – required, this value is passed to the pcgm helper function by the userdeveloper
careProvisionReason	Not used by PCGM
careRecordTimePeriod	Not used by PCGM
clinicalStatementTimePeriod	Not used by PCGM
maximumHistoryStatements	Not used by PCGM
patientAdministrativeGender	Not used by PCGM
patientBirthTime	Not used by PCGM
patientId	Parameter – required, this value is passed to the pcgm helper function by the userdeveloper
patientName	Not used by PCGM

TABLE 7: CHILD ELEMENTS OF QUERYBYPARAMETERPAYLOAD ELEMENT

ELEMENT	VALUE SOURCE
queryId	Plugin helper function creates a timestamp for message instance that may be used as the query identifier OR Plugin helper function creates a unique identifier for message instance that may be used as the query identifier
statusCode	Plugin configuration file contains the default code specific for the message type
responseModalityCode	Not required – not used, however a default value may be added to the plugin configuration file
responsePriorityCode	Plugin configuration file contains the default code specific for the message type
careEventId	Not used by PCGM
encounterStatus	Parameter – optional, plugin configuration file contains default
encounterTimeFrame	Not used by PCGM
patientId	Parameter – required, this value is passed to the PCGM helper function by the user-developer
patientLocationId	Not used by PCGM
responsibleOrganization	Not used by PCGM
typeOfEncounter	Parameter – optional, plugin configuration file contains default

Both queryByParameter and queryByParameterPayload contain optional elements listed in the tables above. These elements are used to constrain the query search. PCGM only uses the minimal constraint on the request search and only asks the user-developer to provide those values as parameters. This allows the user-developer to quickly query the underlying EHR with minimal information. Then once results are returned determine if further

constraints are necessary for a given use case. Any further filtering of the returned data can be done locally.

It is important to note that PCGM is designed to allow the model-developer (or the plugin-developer) to modify the signature of the generated function by choosing to add one of the optional elements as a required parameter. This is done through an addition of a parameter into the configuration file.

6.1.2 RESPONSE MESSAGE DATA EXTRACTION

Response Message Data Extraction: The user-developer uses the class specific to each parameter returned by a CAL-WS operation to extract the data exposed by the PCGM plugin. This allows the user-developer to quickly get to the particular data value in the HL7 object without all the code otherwise necessary to navigate the HL7 message hierarchy.

In the table below you can find the response messages returned by each CAL-WS operation, the corresponding PCGM plugin helper class and functions the user-developer can call to get the data available. The last column displays the PCGM plugin helper function return data type.

TABLE 8: RESPONSE MESSAGE TYPES WITH THEIR CORRESPONDING PCGM API FUNCTIONS

CAL WS Operations	CAL ResponseParameter Types	PCGM Response Helper Classes	PCGM Response Helper Functions	PCGM Return Type
	(passed to the PCGM function after received from the CAL web service)			
FindEncounters	FindEncounters_PRPA_IN900350UV02MessageType	pcgmHelperFindEncountersPRPAIN900350UV02MessageType	getEncounterList	List of EncounterType
GetProcedures	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getProcedureList	List of ProcedureType
GetMedications	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getMedicationList	List of MedicationType
GetAllergies	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getAllergyList	List of AllergyType
GetVitals	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getVitalList	List of VitalsType
GetProblems	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getProblemList	List of ProblemType
GetTestResults	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getTestResultList	List of TestResultType
GetImagingResults	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getImagingResultList	List of ImagingResultType
GetImmunizations	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getImmunizationList	List of ImmunizationType
GetFamilyHistory	TBD	TBD		
GetSocialHistory	TBD	TBD		
GetInsurances	CareRecord_QUPC_IN043200UV01ResponseType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getInsuranceList	List of InsuranceType
GetOrders	CareRecord_QUPC_IN043200UV01MessageType	pcgmHelperCareRecordQUPCIN043200UV01MessageType	getOrderList	List of OrderType
GetPatientInfo	PatientDemographics_PRPA_MT201303UVResponseType	pcgmHelperPatientDemographicsPRPAMT201303UVResponseType	getPatientLastName getPatientFirstName getPatientGender getPatientIdentifier getPatientDateOfBirth getPatientAddressStreetLine1 getPatientAddressCity getPatientAddressState getPatientAddressZip getPatientTelephoneHome getPatientTelephoneCell getPatientTelephoneWork getPatientEmergencyContactName getPatientEmergencyContactTelephone getPatientEmergencyContactRelationship	String String String String String String String String String String String String String String
FindPatients	FindPatients_PRPA_MT201310UVResponseType	pcgmHelperFindPatientsPRPAMT201310UVResponseType	getPatientList	List of PatientType
FindProviders	ProviderDetails_PRPM_IN306011UV01ResponseType	pcgmHelperProviderDetailsPRPMIN306011UV01ResponseType	getProviderList	List of ProviderType
AppointmentSlotRequest	SlotConfirmation_PRSC_IN030102UVMessageType SlotRejection_PRSC_IN030103UVMessageType	TBD TBD	TBD TBD	TBD TBD
FindResultEventQuery	ResultEvent_POLB_IN364000UV01MessageType	pcgmHelperResultEventPOLBIN364000UV01MessageType	getResultEventList	List of ResultEventType
LabOrderRequest	ActReference_POOR_IN200911UVMessageType	pcgmHelperActReferencePOORIN200911UVMessageType	getLabOrderList	List of LabOrderType
GetAmbEncounterDetails	CareComposition_REPC_IN040200UVMessageType	pcgmHelperCareCompositionREPCIN040200UVMessageType	getAmbulatoryEncounter	EncounterType
GetImpEncounterDetails	CareComposition_REPC_IN040200UVMessageType	pcgmHelperCareCompositionREPCIN040200UVMessageType	getEncounterDetails	EncounterType
FindSlots	SlotsQueryResponse_PRSC_IN030102UVMessageType	pcgmHelperSlotsQueryResponsePRSCIN030102UVMessageType	getSlotsList	List of SlotsType
GetDocumentWithContent	TBD	TBD	TBD	TBD

The CareRecord message returns a list of records and other operations may as well. New restructured data types representing these records can be (some have been) defined and the PCGM plugin function returns a list of the new data type.

6.2 DATA RESTRUCTURE – PCGM DATA TYPE (PCGM-DT)

The GUI relevant data types, referred to as PCGM-DTs, may be defined as java classes or xml schema documents and converted from one to the other using JAXB. The XML schema with PCGM-DTs should be included in the CAL-WS schema and processed the same way as the rest of the schema to generate the appropriate classes (POJOs) for later use. The following table shows a sample set of such data types.

TABLE 9: PCGM RESTRUCTURED DATA TYPES

PCGM Retrun Data Type	GUI Relevent FieldsDefined in Data Types	Element Location Path
		(...) = response->controlActProcess->subject->registrationEvent->subject2->careProvisionEvent
AllergyType	String recordId String patientId Date onsetDate String allergyType String allergyProductDesc String allergyReactionDesc String textComment String allergyInformationType "Severity" String allergyInformationValue String status String dataOwnerOrganizationName	(...)->pertinentInformation3->observation->id->extension (...)->recordTarget->patient->id->extension (...)->pertinentInformation3->observation->effectiveTime->value (...)->pertinentInformation3->observation->code->displayName >substanceAdministration->consumable->administerableMaterial->administerableMaterial->code- (...)->pertinentInformation3->observation->sourceOf->observation->value->displayName (...)->pertinentInformation3->observation->sourceOf->observation->text (...)->pertinentInformation3->observation->sourceOf->observation->sourceOf->observation->code->displayName >displayName (...)->pertinentInformation3->observation->statusCode->code >custodian->assignedEntity->assignedOrganization->name
VitalsType	String recordId String patientId String vitalMeasureTypeDesc String vitalMeasureTypeCode String vitalMeasrueTypeCodeSystem String vitalMeasurementDate String vitalMeasurementValue String vitalMeasurementUnit String vitalDataOwnerOrganizationName	(...)->pertinentInformation3->observation->id->extension (...)->recordTarget->patient->id->extension (...)->pertinentInformation3->observation->code->displayName (...)->pertinentInformation3->observation->code->code (...)->pertinentInformation3->observation->code->codeSystem (...)->pertinentInformation3->observation->effectiveTime->value (...)->pertinentInformation3->observation->value->value (...)->pertinentInformation3->observation->value->unit >custodian->assignedEntity->assignedOrganization->name
ProblemType	String recordId String patientId String problemType String problemDesc String textComment Date onsetDate Date resolutionDate Date recordDate String status Integer diagnosisPriority String dataOwnerOrganizationName String providerLastName String providerId	(...)->pertinentInformation3->observation->id->extension (...)->recordTarget->patient->id->extension (...)->pertinentInformation3->observation->code->displayName (...)->pertinentInformation3->observation->value->displayName (...)->pertinentInformation3->observation->text (...)->pertinentInformation3->observation->effectiveTime->value (...)->pertinentInformation3->observation->effectiveTime->value cal not used (...)->pertinentInformation3->observation->statusCode->code cal not used >custodian->assignedEntity->assignedOrganization->name (...)->pertinentInformation3->observation->performer->assignedEntity1->assignedPerson->name->family (...)->pertinentInformation3->observation->performer->assignedEntity1->id->extension
ImmunizationType	String recordId String patientId Date timeOfAdministration String site String route String doseValue String doseUnit String productName String productBrandName String productLotNumber String productManufacturerName String performerId String performerLastName String informationType ("Dose Number") String informationValue (seriesNumber) String adverseReaction ("") String adverseReactionDesc String dataOwnerOrganizationName Boolean refusalIndicator String refusalReason	(...)->pertinentInformation3->observation->id->extension (...)->recordTarget->patient->id->extension (...)->pertinentInformation3->substanceAdministration->effectiveTime->value (...)->pertinentInformation3->substanceAdministration->routeCode->displayName (...)->pertinentInformation3->substanceAdministration->approachSiteCode->displayName (...)->pertinentInformation3->substanceAdministration->doseQuantity->value (...)->pertinentInformation3->substanceAdministration->doseQuantity->unit (...)->pertinentInformation3->substanceAdministration >consumable->medication->administerableMedicine->code->displayName (...)->pertinentInformation3->substanceAdministration >consumable->medication->administerableMedicine->desc (...)->pertinentInformation3->substanceAdministration >consumable->medication->administerableMedicine->asMedicineManufacturer->manufacturer->name (...)->pertinentInformation3->substanceAdministration->performer->id->extension (...)->pertinentInformation3->substanceAdministration->performer->assignedPerson->name->family (...)->pertinentInformation3->substanceAdministration->sourceOf->observation->code->displayName (...)->pertinentInformation3->substanceAdministration->sourceOf->observation->value->value (...)->pertinentInformation3->substanceAdministration->sourceOf->observation->code->displayName (...)->pertinentInformation3->substanceAdministration->sourceOf->observation->value->value >custodian->assignedEntity->assignedOrganization->name (...)->pertinentInformation3->substanceAdministration->negationInd (...)->pertinentInformation3->substanceAdministration->sourceOf->code->displayName

TABLE 9: PCGM RESTRUCTURED DATA TYPES (CONTINUED)

	ImmunizationType	ProblemType
ProcedureType		
MedicationType		
TestResultType		
ImagingResultType		
InsuranceType		
OrderType		

The following sample XML file includes PCGM-DTs for ImmunizationType, ProblemType, and AllergyType.



AdapterCommonDataLayer_guiDataCareRecordTypes.xml

PCGM-DTs are derived from an analysis of GUI relevant data types, see file below for details.



EHR_GUI_DATA_FIELDS.pdf

6.3 THE PCGM CONFIGURATION FILE

The plugin developer creates the configuration file to provide additional information that is needed in the helper code generation process. The format of the file and it's content needs to mature along with the plugin code.

The element location path from table 9 can be used in creating the configuration file. It is the link between the HL7 elements in the schema and the file. Although PIIM has not explored it fully, it is possible to auto generate this configuration file, at least partly, using a sample message with default values.

The following sample configuration file is used in the PCGM implementation example. It includes information to generate helper code for a care record request type, care record message types, a patient demographic request type, and patient demographic response type.



pluginConfigFile.xml

Creation and optimization of this file is out of scope for this phase of development.

7.0 PCGM Java Plugin Prototype (PCGM-JPP) Implementation Example

7.0 PCGM JAVA PLUGIN PROTOTYPE (PCGM-JPP)

IMPLEMENTATION EXAMPLE This project was developed using the NetBeans integrated development environment (IDE) and the Java Framework. JAXB's XJC compiler Plugin and the sun.codemodel are central to the PCGM-JPP.

The plugin prototype is intended as a 'throwaway' prototype and not an 'evolutionary' prototype. A selection of the CAL-WS operations were applied for the development of the algorithm, therefore further development and analysis is required for a production ready solution. The prototype is a good source of reference for examples of sun.codemodel and XJC application, as it may be difficult to find sufficient documentation on these technologies.

The following steps describe the implementation example steps:

1. The CAL-WS WSDL URL was used to create a new web services client project in Netbeans. A test PAWS (AHLTAPI) implementation was used throughout this example.
2. The HL7 objects, that provide parameters to the CAL-WS operations, were manually constructed.
3. Once the response from the CAL-WS operations was returned, GUI relevant data from the HL7 object was extracted manually.
4. The manual code in the client project provided

direction on what parts could be auto-generated with the PCGM plugin.

5. An XJC plugin project was created. The JAXB subproject API, sun.codemodel, was utilized to auto-generate the code sections identified for auto-generation in previous step. This was only implemented for a select number of operations.

6. The auto-generated helper classes were inserted back into the original client project.

7. Using the auto-generated helper functions the hl7 objects were reconstructed as request parameters and data was extracted from the response.

The following sections provide further insight into the implementation details.

7.0.1 CREATING A WEB SERVICES CLIENT PROJECT USING THE NETBEANS IDE, INSTRUCTIONS

The instructions for creating a web services client and a servlet in the NetBeans IDE at:

<http://netbeans.org/kb/docs/websvc/client.html#exploringthefacilities>

CAL WSDL endpoint used throughout implementation:
<http://solutions-it.org:8080/CommonDataLayerService/AdapterCommonDataLayer?wsdl>

Patients available for you to query:

Randall Jones (99990069), Elizabeth Smith (99990070)

Note: As per the instructions, drag and drop the function calls to generate the code for creating the web service operations. The following image demonstrates one of the functions generated by this step.



The auto-generation creates code sections related to the port, the request, and calls operation on the port. The user developer initializes operational argument (param0). The arguments are of HL7 data type.

7.0.2 CREATING A XJC PLUGIN, INSTRUCTIONS

The following link provides detail instructions on creating XJC Plugin: http://weblogs.java.net/blog/kohsuke/archive/2005/06/writing_a_plugin.html

7.0.3 JAVA API SUN.CODEMODEL, API DOCS

The following link provides detail instructions codemodel API: <http://codemodel.java.net/nonav/apidocs/>

7.0.4 THE XJC PLUGIN TASK CAN BE INCLUDED IN AN ANT BUILD, INSTRUCTIONS

The following link provides insight into ANT build process as it relates to XJC and this example: <http://confluence.highsource.org/display/J2B/User+Guide>

7.1 PCGM JAVA PLUGIN PROTOTYPE (PCGM-JPP)

The Java Prototype Development Details:

JAXB processes the XML schema (xsd) files to generate the web services client Java code and artifacts; this is defined as the binding process. The JAXB XJC compiler plugin is used to access the outline of the CAL-WS schema available through this process. The plugin code steps through the outline and selects the classes that are used as a parameter for request operations and those used as a response return type to an operation. The outline object contains all the HL7 artifacts that will be converted from the XML schema to Java code. CAL implements a subset of these objects. For the purposes of speeding up the prototype development process, some of the schema was modified to more closely represent the CAL implementation, leaving out elements that are not used.

The request parameter generator will generate a helper class called the "pcgmHelperRequests". This will hold all the operational specific functions. This helper class contains all the functions corresponding to each request operation parameter. These functions, when called, create HL7 objects of request type. [Section 6.1.1](#) of this document provides design details for the generated helper class for request operations.

The response data extractor will generate a class for each response type found in the CA-WS WSDL. These classes contain private functions that work together internally on the extraction of data from HL7 objects. The public functions within the classes return the data to the user thereby hiding the details of the extraction.

[Section 6.1.2](#) of this document provides design details on the generate response helper classes.

7.2 JAVA TECHNOLOGIES USED BY PCGM-JPP

PCGM-JPP uses Java Architecture for XML Binding (JAXB). JAXB contains a fast and convenient way to bind between XML schemas and Java representations, making it easy for Java developers to incorporate XML data and processing functions in Java applications. The XJC schema compiler, also called the binding compiler, is an important part of a JAXB implementation. The compiler binds a source XML schema to a set of schema-derived program elements. The binding is described by an XML-based binding language in a binding file. The binding compiler produces a set of packages containing Java source files and JAXB property files.

- JAX WS is a Java API for XML Web Services it uses JAXB for XML binding
- XJC is JAXB's XML binding compiler tool

JAXB's XJC compiler creates Java classes based on an XML schema. Those classes and the JAXB runtime enable transformation of XML data to Java objects and vice versa. Once these types are generated they can be used as long as the schema does not change. A change in schema requires regeneration of the Java classes.

- XJC models the classes from the schema and generates the java ATS (abstract syntax tree) from the model
- XJC plugins can access the model and change the outline for the generated java classes

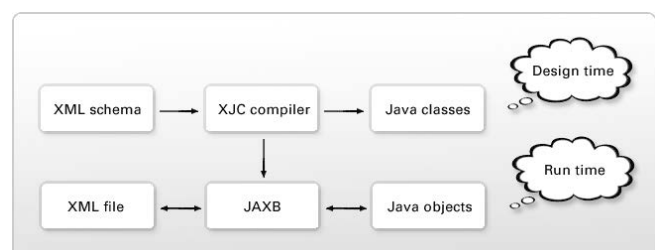


FIGURE 9: JAXB's XJC Compiler
SOURCE: Luttikhuizen, p. 1 (2009).

7.2.1 JAX-WS SPECIFICATIONS

Table 10 provides mapping relationships from WSDL specification to java. This information is indirectly related to PCGM Plugin design.

TABLE 10: WSDL TO JAVA MAPPING

WSDL	JAVA
xsd:complexType (wrapper) The method parameter signature typically is determined by the wsdl:message	Service Endpoint Interface method parameter signature Note: If a parameter is out or inout, a Holder class generates.
wsdl:message The method parameter signature typically is determined by the wsdl:message.	Service Endpoint Interface method signature
wsdl:portType	Service Endpoint Interface
wsdl:operation	Service Endpoint Interface method
wsdl:binding	Stub Note: The Stub and ServiceLocator classes are implementation specific.
wsdl:service	Service Interface and ServiceLocator Note: The Stub and ServiceLocator classes are implementation specific.
wsdl:port	Port accessor method in Service Interface

7.2.2 XML SCHEMA TO JAVA MAPPING

In this section the mapping of XML schema to java code is examined. Table 11 provides relational details between XML, Java and implementation details on how the PCGM Plugin generated code handles each. This is accomplished by providing code snippets representing code written for code generation. Then actual code generated will be provided for reader to examine.

TABLE 11: JAXB XML TO JAVA

WSDL	JAVA	PCGM-JPP HANDLING
xsd:complexType	Java Bean Class	JVar Note: request types also generate a function and response types and messages generate classes
nested xsd:element/xsd:attribute	Java Bean Property	JVar (via the corresponding complex type) Note: in most cases the type of the element leads us to another complexType
xsd:element attribute maxOccurs="unbounded"	List<> Java	Request: JVar + .get() + .add() Response: JForLoop
xsd:simpleType (enumeration)	Generics	TBD
<...mixed= "true" .../>	List<Serializable>	
<...nillable = true, minOccurs =0.../>	JAXBElement<>	See example 4 below

The following paragraphs contain example extracts of: CAL-WS (HL7) XML Schema, converted Java Code POJOs, PCGM Plugin code that produces the auto-generated helper code, and PCGM auto-generated helper code.

EXAMPLE 1:

XML SCHEMA:

```
<xsd:complexType name="CareRecord_QUPC_IN043100UV01RequestType">
<xsd:sequence>
    <xsd:element name="localDeviceId" type="xsd:string" />
    <xsd:element name="senderOID" type="xsd:string" />
    <xsd:element name="receiverOID" type="xsd:string" />
<xsd:element name="query" type="hl7:QUPC_IN043100UV01.MCCI_MT000100UV01.Message" />
</xsd:sequence>
</xsd:complexType>
```

Note: The sequence element specifies that the child elements must appear in a sequence. Each child element can occur from 0 to many number of times.

CONVERTS TO JAVA (ANNOTATED POJO):

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CareRecord_QUPC_IN043100UV01RequestType", propOrder = {
    "localDeviceId",
    "senderOID",
    "receiverOID",
    "query"
})
public class CareRecordQUPCIN043100UV01RequestType {
    @XmlElement(required = true)
    protected String localDeviceId;
    @XmlElement(required = true)
    protected String senderOID;
    @XmlElement(required = true)
    protected String receiverOID;
    @XmlElement(required = true)
    protected QUPCIN043100UV01MCCIMT000100UV01Message query;
    ...
}
```

PCGM PLUGIN CODE USING SUN.CODEMODEL:

The PCGM Plugin code generation uses sun.codemodel services for code auto-generation. The PCGM Plugin uses the 'outline' object to access java artifacts related to CAL and HL7 during execution. This allows the PCGM Plugin to generate classes, functions, and variables based on mapping established in the process. Sample code provided below demonstrates how variables are declared and initialized by PCGM Plugin during code generation.

Declare the variable jVariable.

```
JVar jVariable = codeBlock.decl(classType, variableName);
```

Initialize the variable.

```
jVariable.init(JExpr._new(classType));
```

PCGM GENERATED CODE:

PCGM Plugin uses the configuration file to extract information related to variables it creates. It will use the values specified in the configuration file to determine if a variable is set to a defined default value or if it is a parameter that was passed to the function by the user-developer.

For example, in the case of the 'localDeviceId', the value is obtained from the configuration file. Information about value sources can be found in [Section 6.1.1](#) of this document. The code is generated based on the value source. The following example demonstrates this for variable 'localDeviceId'. The default value for 'localDeviceId' was provided in the configuration file and therefore the code generated sets the value of the variable as such.

```
CareRecordQUPCIN043100UV01RequestType RqstMsg = new CareRecordQUPCIN043100UV01RequestType();
    java.lang.String RqstMsg_localDeviceId = ("1.1");
    RqstMsg.setLocalDeviceId(RqstMsg_localDeviceId);
    java.lang.String RqstMsg_senderOID = ("1.1");
    RqstMsg.setSenderOID(RqstMsg_senderOID);
    java.lang.String RqstMsg_receiverOID = ("1.1");
```

EXAMPLE 2:

Attributes convert to Java properties as well as elements show in previous example.

XML SCHEMA:

```
<xs:attribute name="classCode" type="ActClassControlAct" use="required" />
```

CONVERTS TO JAVA:

```
@XmlAttribute(name = "classCode", required = true)
protected ActClassControlAct classCode;
```

EXAMPLE 3:

The maxOccurs="unbounded" XML attribute converts to a Java List. When constructing the request in the PCGM helper code often one object in the list needs to be given a value.

XML SCHEMA:

```
<xs:element name="receiver" type="MCCI_MT000300UV01.Receiver" maxOccurs="unbounded" />
```

CONVERTS TO JAVA:

```
@XmlElement(required = true)
protected List<MCCIMT000100UV01Receiver> receiver;
```

PCGM PLUGIN CODE USING SUN.CODEMODEL:

For handling Lists, additional code is required that sets the values of the declared variable through the use of the get() and add() methods of the POJO.

If the current field is a List, then in addition to the code that declares the variable another statement is added to the code block. This produces the line of code which sets the value through the get() and add() methods of the POJO.

```
if (isListType){
    pcgmCodeBlock.directStatement(variableName+".get"+fieldName+"().add("+concatVariableName+");"); }
```

EXAMPLE 4

In some cases during the internal workings of the response helper class some functions need to

iterate through a list of objects using a for-loop, this can be done as follows.

XML SCHEMA:

```
<xs:complexType name="REPC_MT004000UV01.CareProvisionEvent">
...
<xs:element name="pertinentInformation3" type="REPC_MT004000UV01.PertinentInformation5" nillable="true"
minOccurs="0" maxOccurs="unbounded" />
...
</xs:complexType>
```

Converts to Java:

```
public class REPCMT004000UV01CareProvisionEvent {
...
@XmlElement(nillable = true)
protected List<REPCMT004000UV01PertinentInformation5> pertinentInformation3;
...}
```

PCGM PLUGIN CODE USING SUN.CODEMODEL:

Create the for-loop and add it to the PCGM block.

```
JForLoop forLoop = pcgmBlock._for();
Create the variable i, add it to the for-loop and initiate it to 0.
JVar i = forLoop.init("pcgmCodeModel".INT, "i", JExpr.lit(0));
Create a java expression je.
JExpression je = JExpr.direct(parentName+"List.size()");
Use the variable i and the java expression je as the for loop test.
forLoop.test(JOp.lt(i, je));
Update i.
forLoop.update(JExpr.assignPlus(i, JExpr.lit(1)));
Create the body of the for loop so that code can be added to it
JBlock loopBlock = forLoop.body();
...

```

PCGM Generated Code:

```
for (int i = 0; i<(pertinentInformation3List.size()); i += 1) {
...
}
```

At other times the private (internal) function simply returns the list to another function. See PCGM-JPP code at <https://github.com/piim/TATRC1-PCGM-Plugin-Prototype1> for details.

EXAMPLE 5:

When XML element information cannot be inferred by the derived Java representation of the XML content, a JAXBElement object is provided. This object has methods to get and set the object name and object value.

XML SCHEMA:

```
<xs:element name="queryByParameter" type="QUPC_IN043100UV01.QUQI_MT020001UV01.QueryByParameter"
nillable="true" minOccurs="0" />
```

CONVERTS TO JAVA:

```
@XmlElementRef(name = "queryByParameter", namespace = "urn:hl7-org:v3", type = JAXBElement.class)
protected JAXBElement<QUPCIN043100UV01QUQIMT020001UV01QueryByParameter> queryByParameter;
```

PCGM PLUGIN CODE USING SUN.CODEMODEL:

In addition to the code that declares the variable; the PCGM Plugin adds another statement to the code block which uses the set() method of the JAXBElement to set the value of the variable.

```
if (isJAXBType){
pcgmBlock.directStatement(varName+".set"+fieldName+"(factory.create"+nextType+fieldName+"("+concatVariableName+"))");}
```

PCGM GENERATED CODE:

Adds the QueryByParameter to the ControlActProcess.

```
RqstMsg_query_controlActProcess.setQueryByParameter(factory.createQUPCIN043100UV01QUQIMT020001UV01
ControlActProcessQueryByParameter(RqstMsg_query_controlActProcess_queryByParameter));
```

7.2.3 SCHEMA-TO-JAVA MAPPING DATA TYPES

The Java language provides a richer set of data types than the XML schema. The following table provides a mapping of XML data types to Java data types in JAXB.

TABLE 12: XML SCHEMA AND JAVA DATA TYPES

XML Schema Type	Java Data Type
xsd:string	java.lang.String
xsd:integer	java.math.BigInteger
xsd:int	int
xsd:long	long
xsd:short	short
xsd:decimal	java.math.BigDecimal
xsd:float	float
xsd:double	double
xsd:boolean	boolean
xsd:byte	byte
xsd:QName	javax.xml.namespace.QName
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]
xsd:unsignedInt	long
xsd:unsignedShort	int
xsd:unsignedByte	short
xsd:time	javax.xml.datatype.XMLGregorianCalendar
xsd:date	javax.xml.datatype.XMLGregorianCalendar
xsd:g	javax.xml.datatype.XMLGregorianCalendar
xsd:anySimpleType	java.lang.Object
xsd:anySimpleType	java.lang.String
xsd:duration	javax.xml.datatype.Duration
xsd:NOTATION	javax.xml.namespace.QName

As mentioned before, when XML element information cannot be inferred by the derived Java representation of the XML content, a `JAXBElement` object is provided. This object has methods to get and set the object name and object value.

7.3 PCGM-JPP DESIGN

The PCGM-JPP design is based on the PCGM design; see [Section 6](#) of this document for details.

7.4 PCGM-JPP CODE EXPLAINED

7.4.1 REQUEST PARAMETER OBJECT CONSTRUCTION

The “generateRequestClass” function is invoked from the run method. This begins the generation of the class that builds request objects. This class uses the outline object available through the XJC plugin as a parameter and using `sun.codemodel` it adds the “pcgmHelperRequests” class to a newly created codeModel called the “pcgmCodeModel”. The execution loops through the classes in the outline object. For each class of type Request it adds a method to the “pcgmCodeModel”. It also calls the function “generateRequestMethodBody” which will be described further in the next paragraph. The “generateRequestClass” function will also access the configuration file and create a global map of the list of the parameters and variables that each method in “pcgmCodeModel” requires. The “ParametersMap” is used to add the parameter signature to the method in the “pcgmCodeModel”. The configuration file gives the plugin developer the ability to add or omit parameters without changes to the plugin code.

The “generateRequestMethodBody” continues to add to the “pcgmCodeModel”. A variable is added to the “pcgmCodeModel” for each field that is also in one of the maps “VariablesMap” or “ParametersMap”. The first object is the request type object received as a parameter from the calling function. Thereafter, a recursive call is made for the fields within the current object.

7.4.2 RESPONSE PARAMETER DATA EXTRACTION

The “generateResponseClass” function is invoked from the run method. This begins the generation of the classes that extract data from the response objects passed to them.

The “generateResponseClass” in the PCGM-JPP takes the outline as a parameter and using `sun.codemodel` it defines and adds a new class to the “pcgmCodeModel” for every response/message type class in the outline. It also accesses the configuration file and creates local, global maps of the private (“privateFunctionsMapWholeList”) and public (“publicFunctionsMapWholeList”) functions for later reference. The generation of functions is started by invoking “generateResponseMethod”. The “generateResponseMethod” function makes decisions based on the modifier of the function; information in the configuration file drives these decisions and in turn the code generation. The public functions should correspond to the primitive data types (PCGM-DTs) returned to the user-developer.

Private functions handle stepping into the HL7 hierarchy by declaring, initializing, and calling the `get()` method of the current field/object being processed; then return that object to the next function. The next function generated calls the previous function to get the previous object on which it then calls its own `get()` method, and so on.

EXAMPLE GENERATED CODE:

```
private PRPAMT201303UV02Patient
pcgm_getSubject(PatientDemographicsPRPAMT201303UVResponseType param1)
{PRPAMT201303UV02Patient subject = new PRPAMT201303UV02Patient();
subject = (param1).getSubject().get(0);
return subject; }

private II pcgm_getId(PatientDemographicsPRPAMT201303UVResponseType param1) {
II id = new II();
id = pcgm_getSubject(param1).getId().get(0);
return id;}
```

If the current field processed is in the “privateFunctionsMapWholeList” global map, sun.codemodel is used in generating the private function described above. A JMethod called “pcgmMethod” is added to the “pcgmCodeModel”. There is further detail about the function in the configuration file that contributes to the return type and other considerations as the method signature and body are generated.

Public functions are responsible for returning data to the user-developer, this may be one piece of data in String or Date format, or it may be a [PCGM-DT](#). See also [Table 8](#) for the list of public functions and data types returned.

EXAMPLE GENERATED CODE:

```
public String getPatientGender(PatientDemographicsPRPAMT201303UVResponseType param1) {
String code;
CE administrativeGenderCode;
administrativeGenderCode = pcgm_getAdministrativeGenderCode(param1);
code=administrativeGenderCode.getCode();
return code; }
```

If the current field processed is in the “publicFunctionsMapWholeList” global map “createPublicClass” function is called from the “generateResponseMethod”. Based on several conditions from the configuration file and the previous class that was generated code generation takes different paths.

7.5 PCGM JPP RESULTS

PCGM JPP was used to rapid prototype implementation of PCGM model. This exercise was aimed to learn more about the environment and assist in design phase of the project. The results of this exercise have been summarized in this section.

FINDINGS:

1. JAXB became central to the java prototype implementation. It contained two elements that were extremely useful in this project. The sun.codemodel object provided code generation services while the xjc Plugin made available the outline of the java POJOS that represent the CAL-WS HL7 schema. It is important to note that the information needed from the outline can also be obtained by parsing the schema directly. The high-level handling of requests and responses were straight forward and easy to deduce. The details of the data types, however, proved to be more of a challenge.
2. The PCGM plugin generated helper functions significantly accelerated development. The abstraction of HL7 layers for the purpose of simplifying development is the most significant contributor to this acceleration.

CONSIDERATIONS:

1. The CAL implementation of the HL7 schema (and associated XSDs) seems extensive and all-encompassing and a large portion does not seem to be in use. However, existence of the extra content contributes to unnecessary complexity of auto-generated code. It may also impact execution times. Optimizing the CAL WSDL could result in optimization of auto-generated code, reduction of the code complexity, and reduction of design and implementation complexity of PCGM Plugin.
2. Coded Value Types: Often a data type in the schema is defined as ANY and this requires the message instance to specify the derived type by using the attribute “xsi:type” like such: ‘xsi:type=”hl7:CD”’. In general this is bad practice and it makes it difficult, maybe even impossible to auto-generate code based on the schema.
Future work: CAL implementation should specify the type and not use ANY.
3. Problem List chronicity was defined as a GUI relevant data item however the field was not included in the CAL Care Record message. There may be other such cases.

8.0 Model Future Expansion

8.0 MODEL FUTURE EXPANSION The PCGM Plugin's main goal is to accelerate Front-end development in specific to HL7 related communication to EMR backend. The model's main focus for this phase of the project was on reducing complexity of mapping HL7 messages manually by user-developer. The model has laid out a foundation for auto-generation and abstraction of middle layer code associated with HL7. The model also addresses roles and specific functions to help streamline workflow in the development process starting with creation of CAL-WS, PCGM Plugin development, and finally IDE development.

The approach was modeled based on existing, best-in-class, and proven technologies that are currently in utilization in the software industry. The model can be used in several areas for further automation. These areas include utility tools, IDE plugins, and testing.

8.1 USER INTERFACE DESIGN — PLUGIN-SPECIFIC

The area of User Interface (UI) development is an ideal candidate for model expansion. There are many auto-generation tools in existence that could be analyzed and used in enhancing rapid development features of PCGM Model. Modern IDEs are designed for expendability. They achieve this goal by allowing 3rd party plugins that are added to the IDE. PCGM is an ideal candidate to become an IDE UI Plugin. Under such scenario, the user-developer would use PCGM plugin much like a .net developer utilizes ADO UI plugin. A simple drag and drop of CAL PCGM component and definition of CAL-WS URL link would result in PCGM UI exposing all existing elements and messages within an EHR CAL implementation. This would allow the user-developer to simply drag and drop fields on the IDE UI forms without prior exposure to underlying complexity of HL7 messaging and connectivity.

The PCGM is the first step towards the creation of a larger toolset that would simplify software development and enhance EHR interoperability. PIIM hopes to see this realized through the continued development of this model and subsequent tools through the open source community.