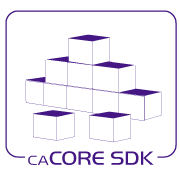
Next Generation caCORE Infrastructure Tools



Version: 0.1

Last Modified: 7/27/2010

Team : caCORE SDK

Client : National Cancer Institute -   
 Center for Bioinformatics and Information Technology,

National Institutes of Health,

US Department of Health and Human Services

Document History

Revision History

| **Version Number** | **Revision Date** | **Author** | **Summary of Changes** |
| --- | --- | --- | --- |
| 0.1 | 07/13/2010 | Prasad Konka, Satish Patel, Bediako George | Initial Draft |
| 0.2 | 07/27/2010 | Prasad Konka | Final Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Team/Role** | **Version** | **Date Reviewed** | **Reviewer Comments** |
| Satish Patel |  | 0.1 | 07/12/2010 | Multiple comments related to content and format |
| Bediako George |  | 0.1 | 07/12/2010 | Multiple comments related to content |
| S. Liu |  | 0.1 | 7/16/2010 | Multiple comments |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Related Documents

More information can be found in the following related documents:

|  |
| --- |
| **Document Name** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Table of Contents

1. Executive Summary 4

2. Background 5

3. Introduction 6

4. caCORE Enterprise Infrastructure 7

4.1 Enterprise Architecture 9

4.2 Semantic Infrastructure 9

4.3 Service Infrastructure 11

4.4 Security Infrastructure 13

4.5 Integration Infrastructure 14

4.6 Deployment Infrastructure 16

5. Achieving Enterprise-level Interoperability 17

5.1 ECCF 20

5.2 Artifacts Generation 20

5.3 Harmonized Data Types 21

5.4 Audit 22

5.5 Localizations 22

5.6 Transformations 22

5.7 Vocabulary/Terminology Framework 23

5.8 Behavioral Semantics 23

5.9 Automated Testing / Certification 23

5.10 Metadata Framework 24

5.11 Security Framework 24

5.12 Service Framework 25

5.13 Integration Framework 25

5.14 caCORE Tools 26

**5.14.1** **Semantic Infrastructure** 26

**5.14.2** **Syntactic Infrastructure** 29

6. Conclusion 35

7. References 36

8. APPENDIX A 37

# Executive Summary

caCORE tools and APIs are developed by the National Cancer Institute Center for Bioinformatics and Information Technology (NCI CBIIT) to provide the building blocks for development of interoperable information management systems. This enables data sharing meaningfully in a coordinated way from the scientific bench to the clinical bedside and back. As the caBIG® community continues to expand along with evolution of technologies, standards and policies – this document examines the need for next generation caCORE Infrastructure tools and processes to enable enterprise-grade, standards based syntactically and semantically interoperable applications in a federated collaborative environment. This document provides an outline on achieving computable semantic interoperability through next generation of caCORE Infrastructure services based tools, frameworks and processes.

The scope of this document is limited to exploring capabilities for next generation caCORE tools to assist the caBIG® community. This is document provides functional overview of each of caCORE next generation components and their associated tools. The description in the document is not intended to provide any design or implementation details of those components. A detailed exploration of each of these tools/components will be done as necessary.

As caCORE Infrastructure and its tools reached their limits, the importance of next generation tools to enable enterprise-grade interoperability between distributed systems based on semantically aware services has been vital. The tools would need to be process and standards driven, modular, loosely coupled, extensible, developer friendly and open source based to reach wider community within caBIG® program. Automation or semi-automation from a model to runtime including model creation, mapping, curation, annotation, transformation, code generation, metadata, composition, security, deployment, runtime execution, conformance assertion and testable integration should be supported. These tools will need to provide platform independent specifications along with reference implementations to support multiple technologies and platforms. With the adoption of ECCF, these tools should also be able to generate necessary artifacts to provide and assert bindings, traceability, conformance, compliance and governance.

# Background

caCORE Infrastructure and its tools have been actively developed and used since 2004 resulting stable deployments in production since 2006. At its inception, the caCORE Infrastructure was bound to the capabilities needed for the cancer community while adopting Grid computing, HL7 V2 messaging and data element standards and ISO 11179 repository standards. With the growing adoption of HL7 V2 standards across healthcare domain, it poses many limitations and challenges like:

* Lack of formal methodologies to model data elements and messages
* Lack of consistent application data model
* Lack of well-defined application and user roles
* Lack of precision in the standard
* Internationalization

To solve the interoperability challenge, CBIIT has chosen to use MDA because of its proven standards. The initial interoperability project involved three steps:

* Analyze what was needed and develop use cases
* Use UML® to standardize model representations and artifacts, often using class and sequence diagrams
* Use meta-models to generate code

The advantage to this approach is that MDA allowed CBIIT to develop or use available tools to automatically generate some of the code, while giving the limited flexibility to tailor it to specific needs. Enterprise service generation has been focused on specific model constructs, format, data type standards, technology with no or limited extensibility and IDE integration. The CBIIT has extended the MDA paradigm to include registered metadata and controlled terminologies. The use of metadata was especially important to eliminate ambiguity in the definitions assigned to particular data classes, attributes, and values. CBIIT has adopted ISO 11179 standard to define a framework and protocols for how such metadata can be specified, consistently maintained, and shared across diverse domains. This approach has been serving well within the context of static semantics.

With the adoption of grid technology, the scope of caBIG® interoperability has been tailored towards grid-centric development and deployment paradigm. The technology that the grid infrastructure is based upon has reached a point where it is out of date with the latest industry standards.

The current caGrid Security Infrastructure based on X.509 credentials, poses interoperability challenges with other security federations. In addition, the current caGrid services use transport layer security (TLS) for secure point-to-point communication.  It lacks message-level security to enable other modes of communication to participate in workflow conditions or in enterprise service bus (ESB) architectures.

With the wider participation in caBIG® community, there has been significant need to provide federated, agile and services based solutions and tools supporting latest standards and technologies to achieve enterprise goals and meet the growing demand in caBIG® and BIG Health consortium.

# Introduction

The mission of caBIG® is to develop a truly collaborative information network that accelerates the discovery of new approaches for the detection, diagnosis, treatment, and prevention of cancer, ultimately improving patient outcomes.

The “BIG 3”priorities set at the enterprise-level are:

* Enterprise-level Interoperability

Our applications, data, and analytical services need the capacity to connect and meaningfully exchange information and coordinate behavior.

* Enterprise Security

Develop and implement a clear, integrated enterprise strategy and operational tactics that unify and integrate our approach to security.

* Deployment Support

Comprehensively support the requirements of the 21st century healthcare community including a) on-going stakeholder identification and management b) development of technology-independent specifications and reference implementations of software that provides clear benefit to the community.

The caCORE mission is to provide necessary tools, services, APIs and specifications to enable enterprise-level interoperability within caBIG® community. Enterprise-level interoperability is the collection of structures, processes, and components that support Computable Semantic Interoperability (CSI) between two parties (“trading partners”) who are interacting (for example, exchanging information, coordinating behavior) to achieve one or more business goals. Interoperability, in this context, is further defined to be the deterministic exchange of data or information in a manner that preserves shared meaning. Shared meaning means that the details are worked out in advance so that both the sender and receiver understand the meaning in common.

The requirements for the caCORE components that enable enterprise-level interoperability include the need to facilitate:

* Semantics and structure of the data and information to be exchanged
* Inputs, outputs, and transformations (such as functions) that enable the exchange
* Traceable mappings of functions to real-world events and business processes
* Reference terminologies or language for sorting and discussing the above processes
* Engineering and deployment topologies
* Technology bindings to achieve enterprise-level interoperability

NCI CBIIT has adapted SAIF (Services-Aware Interoperability Framework) as the architectural framework to build or improve its components to achieve enterprise-level interoperability within health care systems.

As discussed in the *HL7 SAIF Book²*, the overarching goal of SAIF is to enable the development of software components that can participate in instances of Working Interoperability (WI), i.e., the unambiguous, predictable, system-mediated exchange of data and/or the coordination of inter-component behaviors. At the core of the requirement that the interaction instance between two components be “unambiguous, predictable, and system-mediated,” is the overarching requirement that the semantics – i.e., the meaning – of all of the various data involved in the interaction be explicitly defined in terms that are amenable to processing by a machine. Hence, the term “WI” is shorthand for “computable semantic interoperability” (CSI) with the notion of semantics applied to not only the static data passed between machines, but also to the behavioral/functional operations exposed for coordination of behaviors during the interaction, e.g., the data expressed via a set of component application programming interfaces (APIs).

# caCORE Enterprise Infrastructure

The caCORE, at the heart of the caBIG® enterprise-level interoperability initiative, has been providing rich set of specifications, tools, APIs and services to the caBIG® community. The components of caCORE support the semantic consistency, clarity, and comparability of biomedical research data and information. Following diagram depicts big picture of caCORE in the realm of enabling Computable Semantic Infrastructure through the applications generated based on caCORE tools, services, processes and specifications.

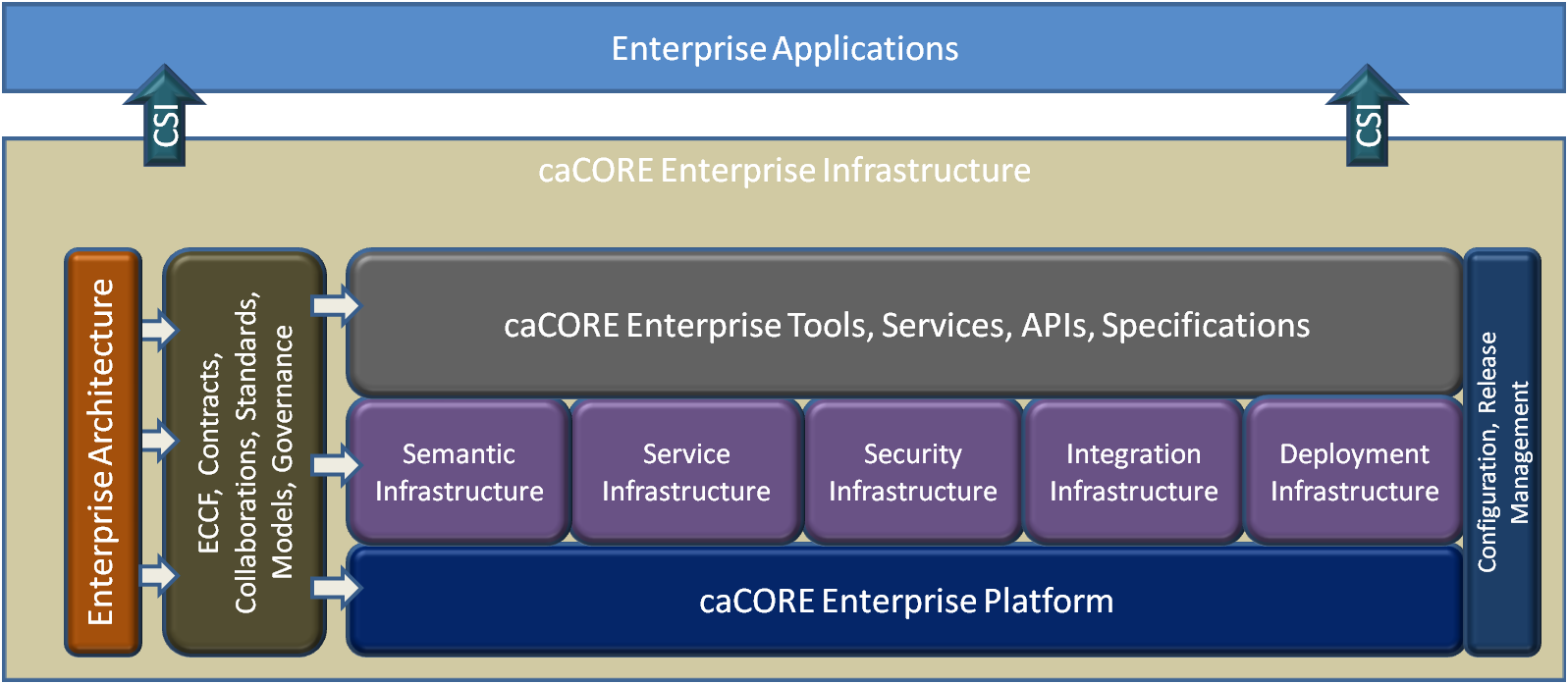


Figure 1 caCORE Infrastructure

NCI Enterprise Architecture Specification (EAS) ³, an instance of BIG HEAS (Big Health Enterprise Architecture Specification based on SAIF) describes enterprise architecture components that are required to achieve enterprise-level interoperability. In line with NCI EAS and interoperability requirements, caCORE Enterprise Platform is an enabling open source platform providing necessary components built up on policies, standards and compliances to build, deploy and run enterprise applications in a federated collaborative environment. As shown in the above diagram, caCORE Enterprise Platform mission is to provide necessary tools, APIs and specifications to build semantically aware services built up on controlled vocabularies, terminologies and data structures those can be easily advertized, discovered, integrated, secured and deployed in a collaborative environment.

caCORE Enterprise Platform can be viewed as a collection of following components:

1. Semantic Infrastructure
2. Service Infrastructure
3. Security Infrastructure
4. Integration Infrastructure
5. Deployment Infrastructure

Each of these components should be able to provide services independently, but work collectively to provide a unified platform to enable semantically service oriented applications. Following sections explains each of these components in achieving “BIG 3” priorities.

## Enterprise Architecture

NCI CBIIT adopts/adapts SAIF for its own use by defining enterprise-specific artifacts, processes, localizations, etc. that describe the organization’s particular focus on and context for intra- and inter-enterprise interoperability. The Services-Aware Interoperability Framework (SAIF) consists of four core sub-frameworks:

* Information Framework (IF)
* Behavioral Framework (BF)
* Enterprise Conformance and Compliance Framework (ECCF)
* Governance Framework (GF)

Each of the SAIF sub-frameworks is a “grammar” that is used for describing a particular topic-of-interest. While the Information and Behavioral Framework specifications are being worked out at HL7, the current focus at CBIIT is on Enterprise Conformance and Compliance Framework (ECCF).

The goal of the ECCF is to promote interoperability. The keyword is interoperability and not integration: integration is when two parties communicate with each other after they have developed a common understanding of the information that is being exchanged. Interoperability is about providing enough detail (via a rich set of semantics) that the parties can identify the gaps without extensive interactions with the other party. This is analogous to providing user-friendly documentation that allows users to use a system, application, or API without having to interact with the developer who developed the application or service. ECCF does not guarantee interoperability but helps the two parties that want to interoperate identify gaps and develop the appropriate transforms. These transforms or adapters help the parties interoperate; the specifications are meant to provide enough detail and precision that extensive interaction with the developer of the service is not required.

As NCI CBIIT is moving towards semantically service-oriented architecture, CBIIT’s goal is to develop a portfolio of reusable enterprise services. The scope of ECCF is restricted to only those services that satisfy an interoperability requirement.

## Semantic Infrastructure

The basis of semantic interoperability6 is to provide common vocabulary and terminology between two interoperable systems. A well-defined set of allowed terms appropriate to a particular domain of concepts is critical for common understanding, and thus imperative for semantic interoperability. Semantic interoperability is the ability of computer systems to communicate information and have that information properly interpreted by the receiving system in the same sense as intended by the transmitting system. "Proper interpretation" means that the transmitted information will be used appropriately by a receiving computer system because the logical implications derivable from transmitted information will be the same as those that the sending system would derive. Semantic Interoperability requires that any two systems will derive the same inferences from the same information.

All parties that intend to achieve working interoperability need to agree on the set of semantics that are associated with achieving their mutual objective. SAIF makes those semantics explicit, rather than requiring implementers to interpret implicit semantics and then work out what they have to implement explicitly. The more implicit, the less the interoperability solution will scale.

Meaningfully exchange data is the key concept in achieving enterprise-level interoperability. To enable this, any data should be structured based on standard and coded vocabularies allowing exchange of documents that are both machine and human-readable enabling electronic processing. HL7 V3 defines three types of interoperability paradigms, namely 1) Messaging 2) Documents 3) Services. HL7 v3 CDA7 defines standard document framework for the incremental growth in the amount and precision of structured, vocabulary-bound clinical information exchange. Use of standardized reference implementation models (RIM) and derived domain information models (DIM) provide level of abstraction to enable template based models to localize, constrain and generate semantically annotated models necessary for meaningful data exchange.

A common ontology allows all interoperating systems to specify meanings of terms with precision, by linking terms used in specific contexts to the ontology elements that describe the meanings of those terms in logical format. A key aspect of the approach to working interoperability is the notion of fully specified business interactions. The context and content of those interactions must be managed by a robust repository such that they can be versioned, published, discovered, and verified. Thus the semantic infrastructure must provide management and curation capabilities for all aspects and artifacts output from the enterprise framework in a federated manner.

The notion of a loosely coupled federation requires infrastructure support for the mapping, harmonization, controlled federation (in light of such issues as licensing), and community-based production and curation of terminology. It should include concepts and their representation, lexical properties of the terms used, semantic relationships, etc.

Following link provides an overview of NCI CBIIT Semantic Infrastructure.

<https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/SI_Conop_Overview>

## Service Infrastructure

Service-Oriented Architecture (SOA) and Enterprise Architecture are mutually reinforcing disciplines that address how IT groups can address system design and team organization to best create more re-usable and survivable systems which break down the traditional application silos. Interoperability is the most important principle of SOA. Enterprise Services enable loosely coupled, platform, operating system and programming language independent solution to generate interoperable applications. Enterprise Services generally have the following characteristics:

* Services may be individually useful, or they can be integrated—composed—to provide higher-level services. Among other benefits, this promotes re-use of existing functionality.
* Services communicate with their clients by exchanging messages: they are defined by the messages they can accept and the responses they can give.
* Services can participate in a workflow, where the order in which messages are sent and received affects the outcome of the operations performed by a service. This notion is defined as “service choreography.”
* Services may be completely self-contained, or they may depend on the availability of other services, or on the existence of a resource such as a database. In the simplest case, a service might perform a calculation such as computing the cube root of a supplied number without needing to refer to any external resource, or it may have pre-loaded all the data that it needs for its lifetime.
* Services advertise details such as their capabilities, interfaces, policies, and supported communications protocols. Implementation details such as programming language and hosting platform are of no concern to clients, and are not revealed.

Enterprise services must be agnostic regarding the choice of operating system, object model, and programming language to succeed in the heterogeneity of the Web. CBIIT currently has a published approach for classifying services and is currently working on governance process/framework that will help guide teams in making these decisions.

While SOA enables loosely coupled enterprise services with above mentioned characteristics, the need for semantic capabilities requires Semantically-Aware Service Oriented Architecture (sSOA) that brings together an approach to achieve enterprise-level interoperability.

Enterprise services can be realized through the use of web services, as one of the key benefits of web services is interoperability, which allows different distributed web services to run on a variety of software platforms and hardware architectures.

The mainstream XML standards for interoperation of web services specify only syntactic interoperability, not the semantic meaning of messages. For example, Web Services Description Language (WSDL) can specify the operations available through a web service and the structure of data sent and received but cannot specify semantic meaning of the data or semantic constraints on the data. Computable Semantic Interoperability can be achieved through Web services built up on semantically annotated models and standard data structures.

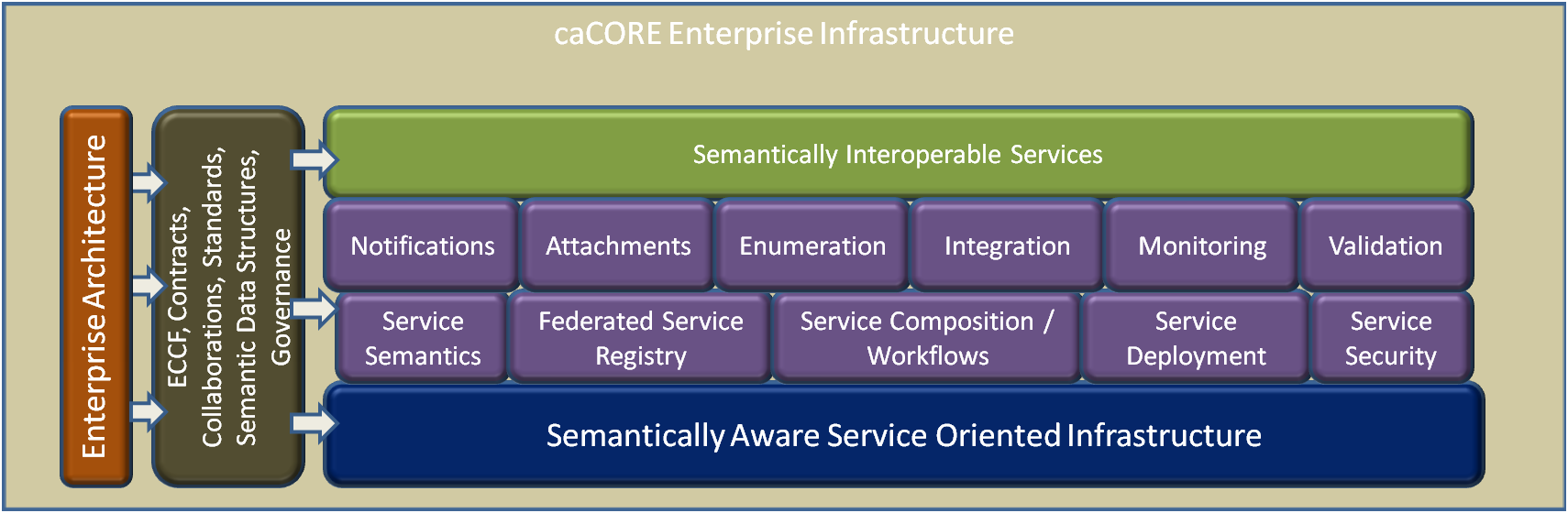


Figure 2 caCORE Service Infrastructure

As shown in the diagram above, the caCORE Semantically Aware Service Oriented Infrastructure should be able to support automated service generation, discovery, invocation and composition based on semantically annotated models. Efficient and usable (IDE extensions, UI driven, easy to use) tools should be created to remove the barrier to entry imposed on completing the fully-specified behavioral semantics of services and potentially reusing specification defined by others. Creating a service specification containing semantically annotated behavioral descriptions mandates that a common terminology be identified or created for use in annotating these specifications. This support for the runtime use of behavioral semantics must build on top of the current support in caBIG® for static semantics or data model and information semantics that is currently used to describe the data that flows throughout the architecture.

The infrastructure should also support following important features to enable next generation Enterprise services framework:

* Multi-protocol service invocations: Ability to specify a different protocol using the same API, for example by changing the URL
* Federated Service Registry: Ability to register, synchronize, discover and invocate services in a federated service registry to reduce the need for centralized approach.
* Transactional: Ability to support transactional capabilities in a workflow
* Multiple Message exchange patterns: Ability to support synchronous and asynchronous service interactions
* Behavioral semantics containing semantically annotated behavioral descriptions
* Testable conformance framework to assert with specifications across different viewpoints

Following links provides information on NCI CBIIT service principles and classifications.

<https://wiki.nci.nih.gov/display/EAWiki/Service+Principles>

<https://wiki.nci.nih.gov/display/EAWiki/SOA+Service+Taxonomy>

## Security Infrastructure

In a federated, collaborative environment, information security plays a critical role in safeguarding privileges while maintaining confidentiality, integrity, and availability of information. In a service oriented system, each resource is made available to a collaborative environment as a service. A service wraps the functionality of the resources in a set of well-defined interfaces. These interfaces (and the associated client side application programming interfaces) are used by client applications to interact with the resource. The inherent benefit of Service Oriented Architecture (SOA) is the loose coupling between the producer and the consumer, which eases the construction of component-based solutions and promotes abstraction. To preserve loose coupling, security must also be implemented as a service - to avoid tightly bound security and thereby tight binding of the services themselves. Tightly bound security scheme however will not work for SOA where services consumers and producers are distributed outside a single security domain and loosely coupled. Key to this loose coupling is to implement security-as-a-service with federated identity management based on standardized mechanisms and formats for the communication of identity information between the domains.

It is critical that the infrastructure provide a mechanism for various members of the enterprise to carry out numerous different business interactions in a variety of security contexts, using each participating entity's own policies and mechanisms. In other words, it cannot be expected that all members of the enterprise will employ the same mechanisms of security foundations (authentication, authorization, etc), or that a central authority will dictate policy. The infrastructure must allow for a diversity of policy and a diversity of complex trust arrangements between its participants (which may vary from one interaction to the next).

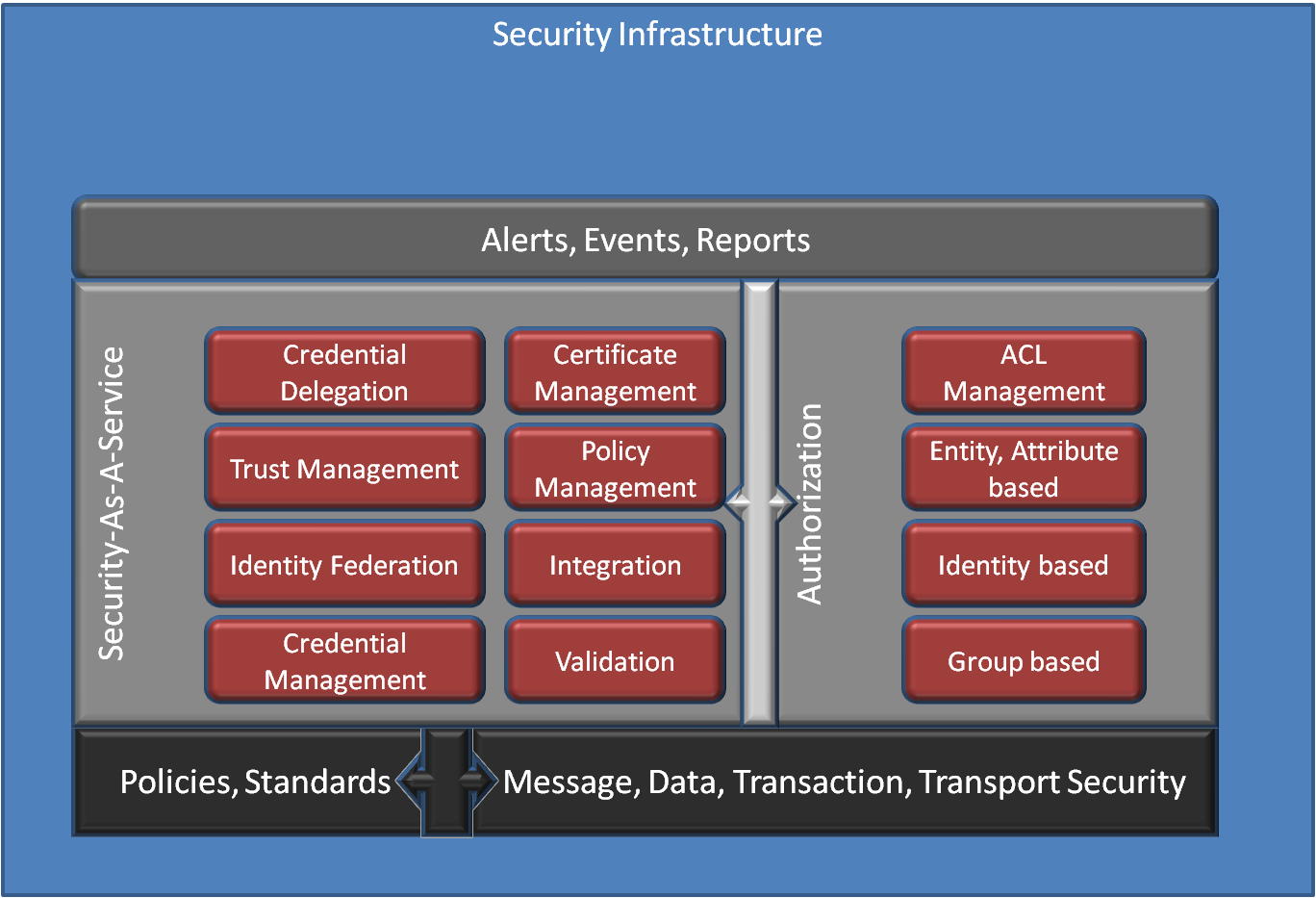


Figure 3 caCORE Security Infrastructure

Above diagram depicts majors components involved in achieving data security in a federated collaborative environment. All these components together should provide consistent and complete application and enterprise-level security capabilities. As it is explained above, security itself needs to be federated to achieve loose coupling between services, providers and users. As explained in section 2, the security infrastructure should address the limitations with X.509 credentials and point-to-point communication.

## Integration Infrastructure

An integration platform is vital in a federated collaborative environment providing secure, interoperable services and applications to bring together different participants. It facilitates using common industry standards, patterns, tools and specifications for easier integration and collaboration.  As the health care, life sciences, and clinical research domains continue to evolve and expand with different standards and technologies, an Integration methodology should not only focus on caCORE infrastructure components, but also focus on external domains, trading partners or government agencies working towards personalized healthcare networks or systems.

Interoperability-based approaches focus on the exchange of meaningful, context-driven data between autonomous systems. Integration approaches, in contrast, typically attempt to build a monolithic view of the enterprise. They integrate processes and applications at the event and message levels so multiple systems become one logical unit. The two approaches can be complementary. Care must be taken to make sure the integration approach should not gear towards monolithic solutions.

***Security***

As the Enterprise Security continue to evolve to support complex requirements of different domains, it is important to define and standardize integration methodology and infrastructure to achieve working interoperability between the participants using different security infrastructures. One example is, technologies like OpenID and Windows Live™ ID are some of growing identity providers supported by many social media healthcare communities. Another example is, CONNECT is an open source software solution that supports health information exchange – both locally and at the national level. CONNECT uses Nationwide Health Information Network (NHIN) standards and governance to make sure that health information exchanges are compatible with other exchanges being set up throughout the country. Security integration should also address bridging together different versions of application and security services.

***Technology***

ECCF helps standardizing specifications at conceptual, platform independent and platform dependant levels to promote loose coupling between models and technologies. It promotes standardized way of representing a domain model using different technologies.

Enterprise Services should be designed to be language and platform neutral promoting portability and interoperability of applications. Enterprise services must be agnostic regarding the choice of operating system, object model, and programming language to succeed in the heterogeneity of the Web. Enterprise services developed and deployed with different technologies or versions may pose complexity in achieving interoperability. Integration Infrastructure could help in providing general agnosticism to operating-systems and programming-languages; for example, it should enable interoperability between Java and .NET applications. It should not only support simple point-to-point messaging but also orchestration and workflows to solve complex healthcare challenges. It could also provide adapters for supporting integration with legacy systems.

***Terminology / Semantics***

As the healthcare terminology continues to evolve, there is a necessity to provide integration infrastructure to enable mapping and transformation services using an assortment of messaging standards or formats such as HL7 V2 and V3 and object and data models. It should also facilitate transformation of data formats and values, including transformation services (often via XSLT or XQuery) between the formats of the sending application and the receiving application.

With the possibility of different semantic frameworks, adapters connecting them may be necessary to enable mapping and transformation between the models.

## Deployment Infrastructure

One of the key benefits of enterprise services is to enable loosely coupled, platform, operating system and programming language independent solution to generate interoperable applications. A standardized deployment infrastructure is necessary to help caCORE based application developers/providers to deploy applications efficiently to achieve interoperability. Automatic builds, continuous integration, local, remote deployments, IDE integration, customizable configurations, secure or non-secure environments, flexible and elastic deployments are some of the key aspects in deploying caCORE based systems. With the Cloud Computing evolving rapidly, deployment policies and procedures to deploy caCORE based services or applications in an automated way on to a cloud environment would greatly benefit the community. It should also explore possibilities of providing CBIIT software-as-a-service or infrastructure-as-a-service.

# Achieving Enterprise-level Interoperability

caCORE infrastructure provides tools, frameworks, services and specifications to build semantically interoperable applications in healthcare, life sciences and clinical research domains. Computable Semantic Interoperability has been the ongoing mission for caCORE Enterprise Infrastructure that defines caCORE Enterprise Platform based on following principles.

* Standards
* Open Source
* Semantics and Vocabulary
* Interoperability

The caCORE infrastructure, as a whole with the adoption of SAIF defines the foundation path to achieve working interoperability (WI). Following are the four pillars of WI as described in HL7 SAIF book2:

1. Common model across all domains-of-interest
2. Robust data type specifications mapped to common models
3. Methodology for binding terms from concept-based terminologies to build Structures
4. A formally-defined process for specifying the static and behavioral semantics of the interoperability

As shown in the diagram below, at the high level, Domain Information Model (DIM) derived from domain specifications, industry standards, Reference Implementation Models (RIM) or Analysis Models is the basis of MDA driven automated system generation. This Platform Independent Model (PIM) can be mapped, annotated and transformed into desired format and stored in a common metadata repository. This model then can be transformed to Platform Specific Model (PSM) to generate desired enterprise services using different languages or platforms. This process would also support any non-annotated models to generate desired enterprise services. Each of the four pillars of WI is represented in the diagram.

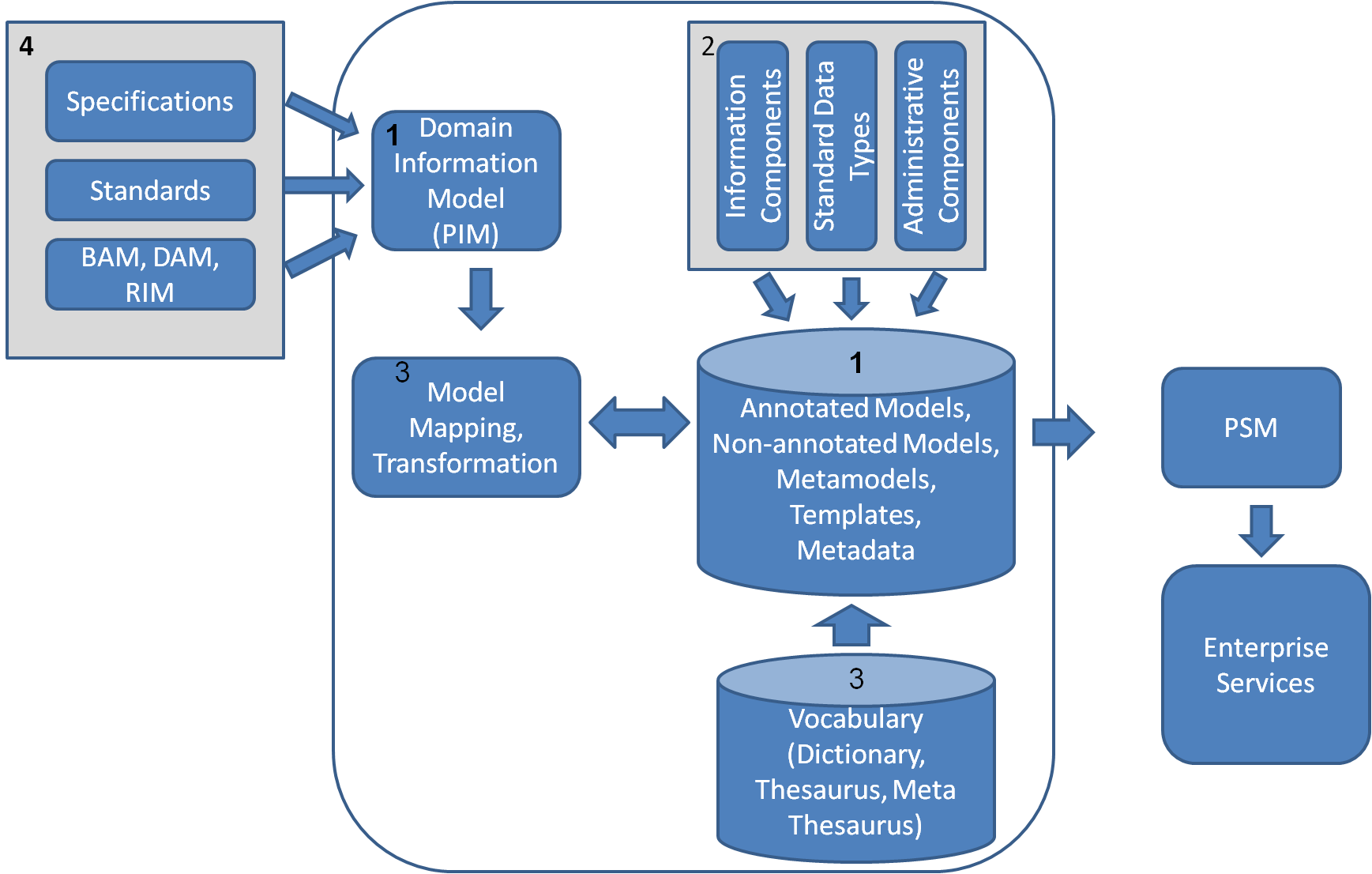


Figure 4 Process for an Interoperable System

The current build process of an interoperable service or application can be viewed at:

<https://wiki.nci.nih.gov/display/caCORE/caCORE+Build+Process+Current+Diagram>

As shown in the diagram below, the caCORE Enterprise Platform defines different independent and interacting components resulting multiple tools, APIs and Services facilitating rapid development and deployment of semantically service oriented applications.

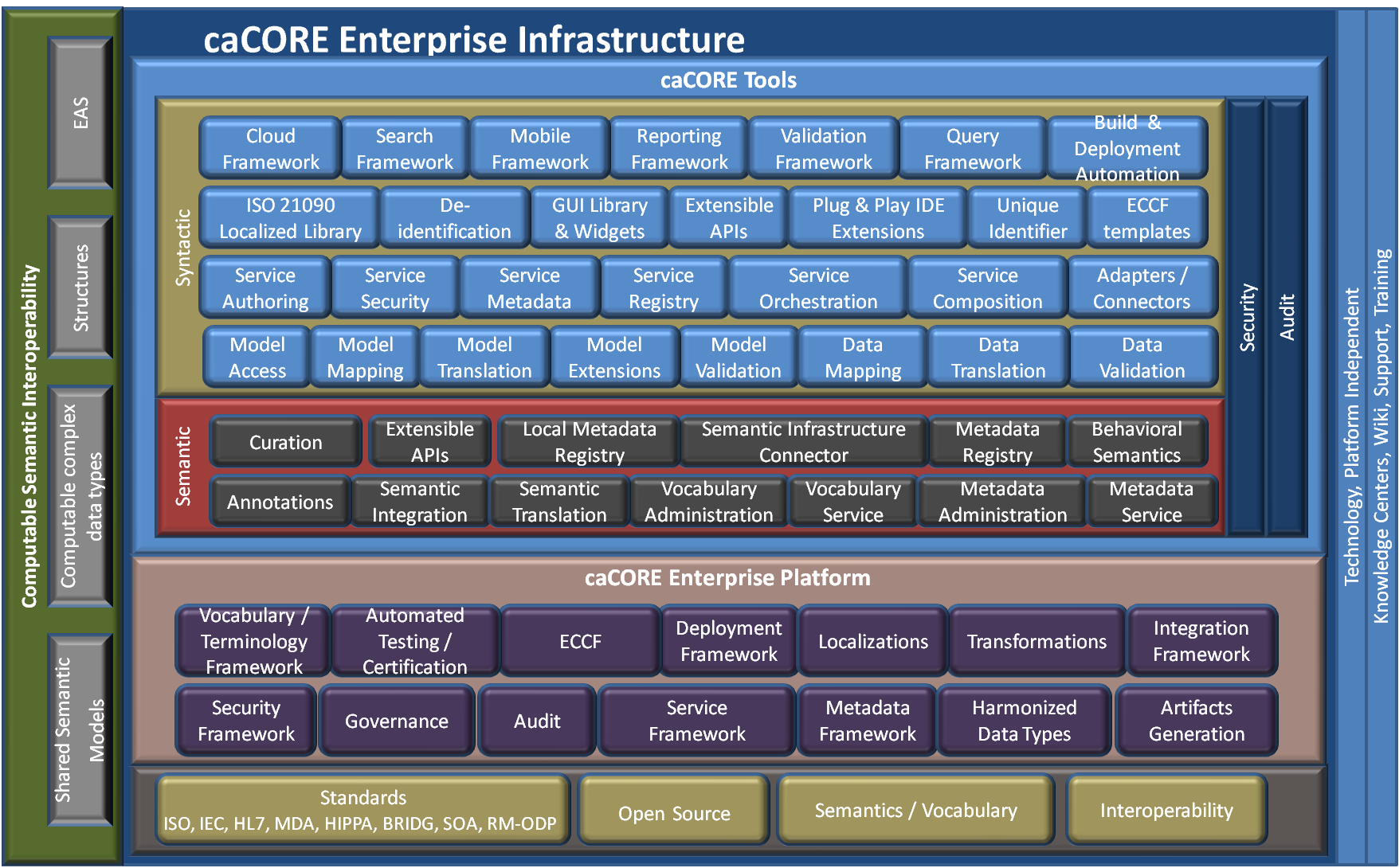


Figure 5 caCORE Infrastructure Tools

## ECCF4

The primary reason that WI is so difficult to achieve in a predictable, scalable manner is that most specifications do not make their myriad of semantic “facts” explicit until the actual implementation. Many of these explicit statements are, unfortunately, incorrect expressions of implicit assumptions that should have been made explicit prior to their expression in the code base. Thus, when “working interoperability doesn’t work,” the involved parties are forced to dig for the reasons at the code level, having no alternative to search for “what’s going on.” With this description of the problem in mind, the ECCF can be described as “a grammar for navigating and quantitatively evaluating a layered collection of artifacts that explicitly express the totality of the static and behavioral semantics necessary to achieve WI in a loosely coupled, technology-diverse, distributed deployment environment.”

The ECCF – through the construct of the Specification Stack, a construct that is fully documented in the HL7 SAIF Book provides CBIIT with the following capabilities:

* A well-defined and structured approach to automated (or semi-automated) conformance certification;
* A well-defined framework to evaluate (and formally certify if necessary) compliance
* A well-defined framework for integrating externally developed specifications or standards.

## Artifacts Generation

Artifacts generation is an important component of caCORE platform. It facilitates syntactic interoperability by providing uniformed way of generating artifacts driven by semantically annotated or non-annotated models.

Syntactic interoperability enables a consumer (e.g., a client program) to programmatically access a resource (e.g., a service). The major obstacles to syntactic interoperability are the heterogeneity of the programming, messaging and service interface syntax and data structures that encapsulate the same type of data across different systems. That is, a software system cannot access the components of another system unless 1) there are programmatic interfaces to those components, 2) these interfaces and how they can be invoked are well-defined, and 3) the two systems have agreed on the structure of data types exchanged between the components of the two systems. As it is explained in section 2.2, sSOA provides ways to define and share static and behavioral semantics along with well defined interfaces using structured data types.

Following the principles of sSOA, caCORE artifact generation capabilities should include:

* Parse different formats of annotated UML models
* Perform validations on UML models
* Support using structured complex healthcare data types
* Support data query capabilities independent to query languages
* Provide extendable APIs to be able to customize its capabilities
* Generate readily deployable and usable services or applications
* Generate data sources capable of working with different relational databases
* Generate Service metadata to dynamically register, advertize and discover
* Support dynamic service composition to generate workflows
* Generate client APIs to access services locally and remotely through different transport protocols
* Provide ability to easily integrate with internal or external Security systems
* Provide pluggable capability to validate, marshal and unmarshal data based on complex structures
* Provide UI widgets / IDE extensions to facilitate easy development and integration
* Provide automated Deployment in varied run time environments
* Generate highly scalable, configurable and customizable systems
* Provide extensions to integrate with externally developed specifications or standards
* Generate documentation or documentation templates required for ECCF
* Generate automated test case or test scripts

## Harmonized Data Types

One of the pillars of Computable Semantic Interoperability is to use shared structured data types where the participants with in an interoperable paradigm would be able to seamlessly exchange data in a predictable and useful manner. The 21090 International Data type Standard (ISO 21090)17 provides the kinds of complex data types necessary for the efficient and effective exchange and analysis of healthcare data. According to the published international standard, ISO 21090:

* Provides data type definitions for representing and exchanging basic concepts that are commonly encountered in healthcare environments in support of information exchange in the healthcare environment, [and]
* Specifies a collection of healthcare-related data types suitable for use in a number of health related information environments.

This standard is a culmination of a large-scale joint effort among standards bodies such as HL7 and ISO, and has been reviewed by experts in the field. The caCORE infrastructure supports building services or systems based on localized ISO 21090 data types.

## Audit

Audit trail is one of the requirements of FDA 21 CFR Part 11 that mandates use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records and shall be available for review and copying.

Auditing capability should provide necessary tools, APIS to log, view, analyze, de-identify, import, export audit in a secure manner. In general, it should provide following capabilities:

* Log message context: Log messages with context to enable grouping messages by transactions, workflows, users, organizations etc.
* Audit Persistent Objects: Logging and tracking persistent object changes
* Alerts and Notifications: Generating alerts and notifications based on persistent object changes.
* Configurable audit levels.

## Localizations

Localization represents the real world of specification application. A given party such as an organization or development team often applies one or more localizations to a given specification to make it specifically and exactly contextually relevant to its stakeholder base. Localizations are not the one-off enemy of WI; rather, they are necessary and manageable by virtue of the ECCF’s framework, which facilitates the expression of a given localization as an explicit assumption being made by a given specification. As such, all localization must be in compliance with and traceable to the artifacts that are being localized.

Localizations can be applied to any aspect of any artifact throughout a specification stack instance, as required. In addition, localizations can be applied using a particular implementation/technology binding.

caCORE Platform supports NCI CBIIT localization of ISO 21090 data types18.

## Transformations

Semantic interoperability indicates the meaning of data can be comprehended unambiguously by both humans and computer programs, and that information can be processed in a meaningful way. Semantic integration is the means to achieve semantic interoperability and can be considered as a subset of information integration, which includes data access, aggregation, correlation, and transformation. The less reliance on manual intervention and the greater degree of automated transformation to meet the different areas of concern, the more flexible and durable the entire environment becomes. Following are the main concerns in supporting transformations:

* **Data Transformations**: Transforming data from one format to another format to support interoperable standards. For example, transforming a HL7 v2 message or CSV format into other HL7 v3 document format automatically is important for collaboration between different systems using multiple standards.
* **Metadata Transformations:** Different formats of metadata require a process and framework to transform them from one format to another.

## Vocabulary/Terminology Framework

As explained in section 3.2, the basis of semantic interoperability is to provide common vocabulary and terminology between two interoperable systems that will enable deriving the same inferences from the same information. caCORE Infrastructure should provide tools and services to define, curate, harmonize, publish, version and discover vocabularies and terminologies through federation.

Details on NCI CBIIT provided vocabulary services, tools can be found at:

<https://cabig.nci.nih.gov/concepts/EVS/>

## Behavioral Semantics

Creating a service specification containing semantically annotated behavioral descriptions mandates that a common terminology be identified or created for use in annotating these specifications. This new support for the runtime use of behavioral semantics must build on top of the current support in caBIG® for static semantics or data model and information semantics that is currently used to describe the data that flows throughout the architecture.

## Automated Testing / Certification

A well defined testing methodology ensures quality of a process or product to meet the requirements. Automated testing based on testing methodologies is necessary for faster, consistent and predictable results. The caCORE infrastructure should provide tools to generate, extend and execute automated test procedures and cases to ensure software quality.

A methodology should be used or developed to ensure that any artifacts defined during the development lifecycle can be validated against other artifacts in preceding and subsequent. This ensures that the final delivered system is guaranteed to meet the original business requirements. Through runtime governance, it is also possible to ensure that the running system continues to meet those requirements.

As a system evolves, through subsequent enhancements to the business requirements, having such a precise understanding of the system, from requirements, through design, implementation and deployment, enables more sophisticated change management techniques to be considered.

SAVARA8 defines a methodology called "Testable Integration Architecture" to address some of the concerns mentioned above.

## Metadata Framework

Sharing static and dynamic (behavioral) semantics is important in achieving working interoperability. One of the core principles of services-awareness is that the *behavioral* constructs of *interface*, *composition*, and *contract* collectively define the *interaction* *semantics* between systems that are engaged in WI scenarios. A common terminology should be used to identify and create metadata (data about static and dynamic semantics) identifying annotated behavioral and static descriptions. Standardized way of accessing metadata dynamically would facilitate different participants working towards computable semantic interoperability. Metadata can be classified into following categories:

* Service metadata: Details on service interfaces, point of contacts, hosting environment and the underlying semantics of the data models used by the service's operations. This would also allow
* Data Type metadata: Schema definitions of data types used within annotated models.
* Model metadata: Semantic annotations and structures of registered models
* Security metadata: Allow to dynamically determine the security requirements for a service such that they may configure themselves to appropriately connect to the service.
* Presentation metadata: Metadata to generate or transform presentation formats.

Metadata framework should provide common tools, services to annotate, store, retrieve, compare and version platform independent and platform dependant models.

## Security Framework

As it is mentioned in section 4.4, security should be implemented in a federated and service oriented manner to enable loose coupling between the security and the consumer. Security as a service would fit the requirement to integrate security well into enterprise Services, thereby enabling enterprise applications to utilize SaaS in the role of service clients. The security framework would need to provide security specifications and reference security services implementations with different tools to develop and assist integrating these security services with enterprise services or applications. With different service security standards available, the security framework should follow enterprise security policies and industry standards to develop a common federated solution to address the needs of the community.

## Service Framework

Enterprise Services enable loosely coupled, platform, operating system and programming language independent solution to generate interoperable applications.

A common service framework should provide necessary tools to model, generate, integrate, deploy, register and locate enterprise services in order to achieve decentralized interoperability. It should not focus on particular technologies used to fulfill the functions but rather divide the problem space into sub-problems with specified relationships. A common security framework should follow the requirements from Service Infrastructure component of caCORE infrastructure and provide necessary tools to enable development and deployment of enterprise services easily. It should provide tools for service authoring, service metadata, service security, service registry, service composition and service orchestration while supporting transactions, reliability, performance and scalability. Tools should also provide ways to migrate existing enterprise services to next generation service framework.

## Integration Framework

As it is mentioned in section 4.5, Integration infrastructure is vital in bringing together disparate data, message, security and technology standards together to provide a common and unified solution. Integration framework should provide conceptual, platform independent specifications and reference implementation using platform specific specifications. Integration framework can looked as a common platform to develop different integration tools required to support different versions of caCORE tools, APIs, services and integration with external participants.

## caCORE Tools

The caCORE Platform focuses on providing open-source software to support the building of software components that enable enterprise-level interoperability. The platform should provide tooling that supports the specification of information exchange standards and all the functions necessary to deploy standards-based automated components, whether the interoperability channel chosen is messaging, documents, or services.

Following sections explain different components/tools of caCORE platform required to achieve enterprise-level interoperability.

### **Semantic Infrastructure**

As it is mentioned in section 2.2, semantic infrastructure provides the basis of semantic interoperability in a collaborative environment. Following capabilities are necessary to achieve the expected semantic interoperability:

* Capabilities for expressing, querying and federating enterprise models and information
* Information aggregation across diverse enterprise data sources
* Support for workflows, alerts, business reports, compliance audits and other analytics
* Capture and manage enterprise information models and metadata while following required compliances and standards

CBIIT is adopting HL7 V3 standard that is based on formal methodologies and object oriented principles to achieve above mentioned capabilities. Following are the major components of HL7 V3 standard:

***RIM - ISO/HL7 21731***

The Reference Information Model (RIM) is the cornerstone of the HL7 Version 3 development process and an essential part of the HL7 V3 development methodology. RIM expresses the data content needed in a specific clinical or administrative context and provides an explicit representation of the semantic and lexical connections that exist between the information carried in the fields of HL7 messages.

***V3 Messaging***

The HL7 version 3 messaging standard defines a series of electronic messages (called *interactions*) to support any and all healthcare workflows. HL7 v3 messages are based on an XML encoding syntax.

***V3 Clinical Document Architecture - ISO 10781***

The HL7 version 3 Clinical Document Architecture (CDA) is an XML-based markup standard intended to specify the encoding, structure and semantics of clinical documents for exchange.

Following sections describe different components required to present the capabilities mentioned above:

**Semantic Integration:** semantic integration will facilitate or potentially automate the communication between computer systems using metadata publishing. Metadata publishing potentially offers the ability to automatically link ontologies. One approach to (semi-)automated ontology mapping requires the definition of a semantic distance or its inverse, semantic similarity and appropriate rules. Other approaches include so-called *lexical methods,* as well as methodologies that rely on exploiting the structures of the ontologies.

**Semantic Translation:** Semantic translation is the process of using semantic information to aid in the translation of data in one representation or data model to another representation or data model. Semantic translation takes should take advantage of semantics that associate meaning with individual data elements in one dictionary to create an equivalent meaning in a second system. It is also possible that different modeling languages can be used while referring same data dictionary. A common metamodel need to be defined to enable different translations supporting different modeling languages.

An example of semantic translation is the conversion of XML data from one data model to a second data model using formal ontologies for each system such as the Web Ontology Language (owl). Another example is the Semantic Web that cuts across the boundaries of applications, enterprises and industries. The Semantic Web links and relates elements of the data model to a common ontology. It uses the Resource Description Framework and the Web Ontology Language to allow data to be shared and reused on the Web.

**Annotation:** Semantic Annotation helps to bridge the ambiguity of the natural language when expressing notions and their computational representation in a formal language. Annotation tool should provide capabilities to add metadata to a UML model and map concepts, data constructs or value domains to UML elements. It should also support adding semantic Annotations to WSDL and XML Schema to represent input and output message structures, interfaces and operations.

**Curation**: Curation is the process of creating/editing content in a metadata repository. It involves characterizing the data that one intends to collect or has collected as a structured data element. Curation tool should provide capabilities to access metadata repository, search metadata elements and create or edit those elements as necessary.

**Metadata Registry:** A metadata registry provides access to registered models and their annotations. caBIG® adopts a model-driven architecture best practice and requires that all data types used on the grid are formally described, curated, and semantically harmonized. These efforts result in the identification of structured data constructs, controlled vocabularies, and object-based abstractions for all cancer research domains. caCORE metadata infrastructure should provide ability to administer and provide access those items.

Metadata Registry is not limited to administering annotated data models. It could also support metamodels and other types of models mentioned in section 5.10 above.

**Local Metadata registry:** A local metadata registry provides federation capability to the concept of maintaining a centralized metadata registry. Federation would provide autonomy to collaborating parties to work with their local registry and synchronize with the central registry or with another registry later to share or promote a solution. This can be accommodated by using Infrastructure metadata connector to connect, compare and synchronize different registries.

**Metadata Service:** Metadata service provides access capabilities to the registered components in the metadata registry. This access capability should not be limited to a particular technology or implementation as the registry can be accessed by different diversified parties. ECCF guidelines should be followed in implementing this service.

**Metadata Administration:** This tool should provide capabilities to perform administrative and curational tasks including Conceptual Domains, Classification Schemes and Protocols.

### **Syntactic Infrastructure**

Syntactic infrastructure facilitates building interoperable systems based on Model Driven Architecture (MDA) principles. Combined with controlled vocabularies and registered metadata, the resulting system is capable of semantically interoperable in a collaborative environment.

Model driven architecture (MDA) has been proposed as an approach to deal with complex software systems by splitting the development process into three separate model layers and automatically transforming models from one layer into the other: 1) The Platform Independent Model (PIM) layer holds a high level representation of the entire system without committing to any specific operating system, middleware or programming language. The PIM provides a formal definition of an application’s functionality without burdening the user with too much detail. 2) The Platform Specific Model (PSM) layer holds a representation of the software specific to a certain target platform such as J2EE, .NET etc. 3) The Code Layer consists of the actual source code and supporting files which can be compiled into a working piece of software. In this layer, every part of the system is completely specified.

Following sections describes each of the functional areas highlighting the necessity of next generation tools required to support caBIG® community.

**Model Access:** As it is mentioned above, MDA principles provide a layered and consistent way to generate software systems from given models. MDA separates business and application logic from underlying platform technology. Modeling provides higher level of abstraction in designing software systems with standards like UML, MOF, XMI, executable UML etc. facilitating to build PIM, PSM. OMG’s Unified Modeling Language (UML) forms a foundation for MDA. A UML model can be either platform independent or platform specific, as it is necessary, and MDA development process uses both of these forms. Modeling access tool should be capable of processing different types modeling formats to support wider audience.

**Model Translation:** The main advantages of MDA are the ability to transform one PIM into several PSMs, one for each platform or technology in which the final system will be deployed, and the automatic code generation that implements the system for those platforms from the corresponding PSMs. The caCORE syntactic infrastructure should be able to provide appropriate tools and processes to transform PIM into PSM and ultimately into services based software systems with minimum human intervention.

**Model Mapping:** Model mapping is the process of linking the Logical Model of the system with the Data Model. This step is required to generate a data service from a UML model. The specific mapping between the Logical Model and the Data Model is done by adding additional information to the UML Model using tag values. A model mapping tool should help automating the process and add the proper tag values to the proper model elements. Manual annotation has the potential to introduce human errors that can cripple operation of the other caCORE tools and cause long delays in the entire workflow.

**Model Extensions:** UML Profiles provide a generic extension mechanism for building UML models in particular domains. They are based on additional Stereotypes and Tagged values that are applied to Elements, Attributes, Methods, Links, Link Ends and more. A profile is a collection of such extensions that together describe some particular modeling problem and facilitate modeling constructs in that domain. The caCORE could provide such extensions to enable modeling domain specific requirements. One example is to provide a business process modeling extension to model workflow conditions.

**Model Validation:** A common modeling framework should be defined and developed to validate UML models to be used for automated artifact generation. This validation should include parsing different formats of modeling, model mappings, model annotations, model elements and associations.

**Data Mapping:** With different industry standards in the healthcare domain, an automated or semi-automated process/tool should be developed to map different formats of data sets, messages and models to enable automated transformation.

**Data Translation:** A common framework / tooling should be able to transform different data sets, messages and models in an automated fashion.

**Data Validation:** A common framework / tooling should be able to validate data sets, messages with vocabulary and HL7 structural attributes.

**Service Authoring**: Service authoring tooling is required to generate enterprise service artifacts automatically from various inputs. These inputs can be metamodels, models, XSDs, WSDL, UDDI, BPEL4WS or POJOS. A service authorizing tool should support following capabilities to generate deployable enterprise service artifacts automatically.

* A user-friendly, customizable, pluggable GUI that can integrate well with any major IDEs.
* Read and parse different UML modeling formats
* Understand complex structured data types
* Generate client and service interfaces
* Generate test interfaces
* Access or generate service metadata and register
* Support pluggable business logic
* Support modifying existing services with plugged in business logic intact
* Support Configuring service security
* Enable service workflow conditions
* Work with different data marshaling implementations
* Generate ECCF service specifications

The service authoring tool should follow provide technology independent specifications and a reference implementation.

**Service Security:** Authentication, data integrity and confidentiality as the key factors in providing secured enterprise services. Service security tools should provide an end-to-end solution including message, transport security from the time a message was generated till it is received. The service security tool should follow enterprise security policies and standards to support following capabilities:

* Message level security assuring integrity and confidentiality
* Support different signature formats, encryptions algorithms and multiple trust domains
* Work with Identity providers, trust levels, certificate authorities
* Work with pluggable authentication mechanisms
* Provide service and operation level security
* Provide easily implementable security in a service workflow conditions
* Address performance overheads

**Service Metadata:** Enterprise servicesuse metadata to describe what other endpoints need to know to interact with them. Service metadata includes description about operations, concrete network protocols, end point addresses, requirements, capabilities, schema describing structure and contents of messages. caBIG adopts a model-driven architecture best practice and requires that all data types used on the grid are formally described, curated, and semantically harmonized. These efforts result in the identification of common data structures, controlled vocabularies, and object-based abstractions for all cancer research domains. A common service metadata standard should be defined to address the needs mentioned and provide necessary tools to facilitate automated service metadata generation, registration and access through a common federated repository.

**Service Registry:** Service Registry is white and yellow pages for enterprise services. It provides ability to search registry entries based on a criteria, drill down queries to retrieve further details and to add qualifiers to the queries.

Federation of registries can provide several advantages over individual registries by facilitating autonomy, heterogeneity, scalability and distribution. Recent versions of UDDI have made changes to accommodate interactions between distributed registries.

A service registry would deliver the foundation for enabling service lifecycle governance as part of the service oriented architecture (SOA) governance. It would enable:

* Publishing and locating services
* Service design time to run time lifecycle governance
* Integrate and federate with other standard registries and repositories
* Optimized access to service metadata
* Collaborative environment
* Manage role-based access to services, changes, versioning, and service retirement
* Enforce service contracts, security, policies, versions and classifications
* Monitor, Report and Auditing

**Service Orchestration / Composition:** Services orchestration is about providing an open, standards-based approach for connecting services together to create higher-level business processes. Orchestration describes how services can interact with each other at the message level, including the business logic and execution order of the interactions. These interactions may span applications and/or organizations, and result in a long lived, transactional, multi-step process model. Choreography tracks the sequence of messages that may involve multiple parties and multiple sources, including customers, suppliers, and partners. Choreography is typically associated with the public message exchanges that occur between multiple services, rather than a specific business process that is executed by a single party.

There is an important distinction between web services orchestration and choreography. Orchestration refers to an executable business process that may interact with both internal and external web services. For orchestration, the process is always controlled from the perspective of one of the business parties. Choreography is more collaborative in nature, in which each party involved in the process describes the part they play in the interaction.

Orchestration includes the management of the transactions between the individual services, including any necessary error handling, as well as describing the overall process. Orchestration describes a business process from each interaction between two services to complete cases that links together these individual service interactions.

**Adapters/Connectors:** Adapters provides integration point between different technologies, terminologies or different formats of datasets. caCORE Enterprise Platform would need to identify and provide adapter services to bring together applications built up on different versions, technologies, platforms and terminologies as necessary.

**ISO 21090 Localized Library:** NCI CBIIT ISO 21090 Localization Common Library is Specialized/constrained version of the ISO 21090 Healthcare Data Types. Localization is the list of approved ISO 21090 data types that will be maintained by CBIIT, along with common implementation artifacts. Localization provides indirect conformance to ISO 21090 Healthcare data types by:

* Providing mappings between internal data types and the healthcare data types
* Specifying for which of the data types an inward mapping is provided, for which an outward mapping is provided, and for which no mapping is provided
* Specifying whether the XML representation described herein is used when the data types are represented in XML, or optionally to provide an alternative namespace for the XML representation

**De-Identification:** De-identification is a process of protecting patient data (PHI or PII) in compliance with HIPPA act. The caCORE infrastructure should provide necessary tools and process to achieve de-identification across multiple domains using different underlying platforms. A provision to de-identify data by rules, permissions, data elements and users should be considered.

More details on de-identification can be referred at:

<http://www.ucdmc.ucdavis.edu/compliance/guidance/privacy/deident.html>

**GUI Library & Widgets:** Applications interact with users through user interface would greatly benefit with uniform, innovative, shared and standards compliance GUI library for rapid application development.

**Plug & Play Implementation:** Plug and Play implementation promotes easier integration of caCORE tools with developer friendly IDEs for faster, easier and seamless environment for development activities.

**Extensible APIs:** Extensibility provides capability to extend, customize an API or framework to better suit enterprise needs. caCORE infrastructure tools and APIs should be extendable to help the community with new innovations or tailored solutions.

**ECCF Templates:** ECCF defines the grammar required to specify static and behavioral semantic explicitly. With number of possible artifacts required for enterprise compliance and conformance, an automated process would help the developer community to generate required artifacts or templates based on service definitions.

**Unique Identifier:** A unique identifier is any identifier which is guaranteed to be unique among all identifiers used for those objects and for a specific purpose. OIDs are the preferred scheme for unique identifiers in HL7. OIDs can be allocated by any organization using a unique OID root. A single message can use OIDs from various sources and a single scheme can be identified by more than one OID (e.g. by an OID from more than one organization). Once issued an OID is never withdrawn and always identifies the same scheme or object.

**Could Framework:** Cloud computing has become a reality where necessary resources or services can be utilized or accessed on-demand. Cloud computing provides a rapid entry point to large data storages, variety of platforms and resources. A framework should be developed based on enterprise-wide methodologies, policies and principles to enable deploying both internally and externally developed enterprise applications based on caCORE infrastructure. This framework should provide capabilities to manage, secure, monitor and meter on-demand resources as needed.

**Mobile Framework:** As the mobile computing is gaining traction with faster wireless networks, caCORE infrastructure should adopt existing standards to build next generation framework that enable building mobile based applications rapidly.

**Reporting Framework:** Reporting provides capabilities for reasoning and decision support. A common framework should be defined to enable reporting on different services or components and provide a unified view to the results.

**Validation Framework:** With different complex structures, processes, services and APIs, it is important to have a common validation framework that can perform customizable tests or validations automatically, consistently and rapidly. The framework should support validation of final delivered system to match with original business requirements in an automated way.

The validation framework should be able to support following categories:

*Model Validation:* Validate model or metamodel constructs

*Mapping Validation:* Validate model mapping with data types, constructs and other models.

*Transformation validation:* Validate model transformation.

*Semantic Integration validation:* Validate model integration with semantic annotations.

*XML/Schema Validation:* Validation data representations, formats and structures.

*Business Process Validation:* Validate service compositions, workflows and business processes.

*Service validation:* Validate service life cycle including service requirements, design, implementation, deployment and runtime.

**Query Framework:** With the possibility of multiple data services in a collaborative environment using different query mechanisms, a unified, query language neutral framework would provide a mechanism to perform distributed aggregations and joins of queries over multiple data services. This would enable composing a data service capable of querying multiple data services and provide unfied and aggregated view.

**Search Framework:** A basic and popular task that most applications would need to support is to provide data search capability. Search tool/framework should be able to provide common interface to perform query activities on data, metadata, models, metamodels, concepts and data elements. Search framework can intern take advantage of Query Framework mentioned above to query distributed data sources.

**Deployment Framework:** As it is described in section 4.6, a deployment framework should provide unified and flexible ways to build, resolve dependencies, package and deploy in supported platforms in physical and virtual environments. It should also support continuous integration, reporting, versioning and alert mechanisms for better visibility.

# Conclusion

Achieving Working Interoperability has been the main focus in bringing together the Cancer community to address personalized healthcare in finding cure for Cancer. The caCORE infrastructure has been one of the key enabling platforms to achieve the goal. It should continue finding ways while recognizing its gaps to better serve the community with tools using latest technologies and industry standards. It should adapt more open source principles to involve wider developer community across different technologies and platforms. This document has identified different functional areas required tooling support. A detailed analysis is necessary to identify the gaps between existing and proposed tools and define a strategy to design, develop and support the community with next generation caCORE infrastructure tools. APPENDIX A below provides a summary of existing and proposed tools with comments.

Even though this top-down analysis is geared more towards SDK, other infrastructure areas mentioned in the document are also equally important in achieving caBIG® goals. SDK can be instrumental in generating or parsing platform independent models, converting them to platform specific models and generate a deployable system using developer friendly tools and techniques.

An analysis addressing bottom-up approach proposing next generation service development framework has been done separately16. Both of these documents should be able to provide necessary next generation tooling information to perform data exchange meaningfully in a coordinated manner.

# References

1. <https://cabig.nci.nih.gov/overview/>
2. <http://gforge.hl7.org/gf/download/docmanfileversion/5569/7139/saif_introduction_for_peer_review_20100409.pdf>
3. <http://ec2-174-129-196-76.compute-1.amazonaws.com/mediawiki/index.php/Main_Page>
4. <https://wiki.nci.nih.gov/display/VCDE/ECCF+Implementation+Guide+%28NCI+CBIIT%29>
5. <https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/SI_Main_Page>
6. <http://en.wikipedia.org/wiki/Semantic_interoperability>
7. <http://www.hl7.org.uk/repository/uploads/565/1/A%20basic%20view%20of%20CDA%20v3.doc>
8. <http://www.jboss.org/savara>
9. <https://wiki.nci.nih.gov/display/seminfra/Init1pm14+-+Transformation+Services>
10. <http://en.wikipedia.org/wiki/Unique_identifier>
11. <https://cabig.nci.nih.gov/tools/concepts/caCORE_overview>
12. <http://www.jboss.org/savara>
13. <http://me.jtpollock.us/pubs/2001.08-BigIssue_eAIJournal.pdf>
14. <http://healthinfo.med.dal.ca/HL7Intro/>
15. <http://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf>
16. Provide link to Bediako’s document
17. <https://wiki.nci.nih.gov/display/ISO21090/ISO+21090+Published+Standard>
18. <https://wiki.nci.nih.gov/display/ISO21090/ISO+21090+Wiki>

# APPENDIX A

|  |  |  |
| --- | --- | --- |
| **Proposed Component** | **Existing Component** | **Comments** |
| Model Access | XMI Handler API | Supports limited features of UML with limited write capability. |
| Model Mapping | caAdapter MMS |  |
| Model Translation |  |  |
| Model Extensions |  |  |
| Model Validation | SDK, caAdapter MMS, SIW | Functionality scattered over multiple tools. Adds complexity in service development workflow. |
| Data Mapping | caAdapter |  |
| Data Translation | caAdapter |  |
| Data Validation | caAdapter |  |
| Service Authoring | SDK, Workbench, Introduce | Multiple tools and complex steps to follow to create a service. |
| Service Security | GAARDS, CSM |  |
| Service Metadata | GME, caDSR |  |
| Service Registry | Index Service, Discovery Service |  |
| Service Orchestration |  |  |
| Service Composition | Taverna, BPEL |  |
| Adapters/Connectors |  |  |
| ISO 21090 Localized Library | CBIIT ISO localized library |  |
| De-Identification |  |  |
| GUI Library & Widgets | NCI CBIIT UI standards library |  |
| Plug & Play IDE extensions |  |  |
| Unique Identifier | LexBIG, BIGID, Grid Identifiers, HL7 OID |  |
| ECCF Templates |  |  |
| Cloud Framework |  |  |
| Search Framework |  |  |
| Mobile Framework |  |  |
| Reporting Framework |  |  |
| Validation Framework |  |  |
| Query Framework | CQL, SDK |  |
| Deployment Framework | BDA | Focused on CBIIT specific deployment |
| Security | GAARDS, CSM |  |
| Audit | CLM | Limited functionality |
|  |  |  |
| Curation | Curation tool |  |
| Annotation | SIW |  |
| Semantic Integration | UML loader |  |
| Semantic Translation |  |  |
| Vocabulary Administration | Protégé |  |
| Vocabulary Service | LexEVS |  |
| Metadata Administration | Admin tool |  |
| Metadata Service | caDSR, LexEVS, MMS, GME |  |
| Metadata Registry | caDSR, cgMDR |  |
| Local Metadata Registry |  |  |
| Semantic Infrastructure Connector |  |  |
| Behavioral Semantics |  |  |
|  |  |  |