SERVICES AWARE INTEROPERABILITY FRAMEWORK

Note to Balloters: This is the first ballot of the canonical Services Aware Interoperability Framework(SAIF) issued by the Architecture review Board(Arb).

Balloters will be expected to respond by document name and line number.

Chair	Charlie Mead	National Cancer Institute (NCI), Center for Biomedical Informatics and Information Technology (CBIIT).”
Co-Chair	Ron Parker	Canada Health Infoway
Editors		
Introduction	Stephen Hufnagel	DoD Military Health System (MHS)
Enterprise Conformance and Compliance Framework (ECCF)	Charlie Mead	National Cancer Institute (NCI), Center for Biomedical Informatics and Information Technology (CBIIT).”
Behavioral Framework (BF)	John Koisch	Guidewire Architecture
	Zoran Milosevic, PhD	nehta National e-Health Transition Authority
Governance Framework (GF)	Jane Curry	Health Information Strategies Inc.
Information Framework (IF)	Cecil Lynch	Ontoreason

The ArB wishes to acknowledge the contributions of the following ArB members:

Andy Bond	nehta National e-Health Transition Authority
Grahame Grieve	Kestrel Computing
Anthony Julian	Mayo Clinic
Patrick Loyd	Gordon point Informatics LTD.
Wendell Ocasio	Agilex Technologies
Abdul-Malki Shakir	Shakir Consulting

Contents

1 Introduction

1.1 Executive Summary

The Service-Aware Interoperability Framework (SAIF) ***goal*** is to assure Working Interoperability[1]; *the biggest impediment to working interoperability is implicit assumptions*.

SAIF's ***technical objective*** is to create and manage easy-to-use, coherent[2] and traceable Interoperability Specifications (**ISs**) regardless of the message, document or service interoperability-paradigm. The SAIF ***focus*** is on managing and specifying artifacts that explicitly express the interoperability characteristics of software components. SAIF's ***approach*** is to organize and manage architectural complexity with a set of constructs, best practices, processes, procedures and categorizations. SAIF's ***scope*** is the interoperability space between business objects, components, capabilities, applications, systems and enterprises. Specifically, SAIF manages the interworking among distributed systems that may involve information exchanges, interactions and state changes. SAIF is not Enterprise Architecture[3]; but instead can be used to augment an EA approach with specific interoperability content and constructs.

TBD 21-26 Mar 2011,

This Concept Map will be done

After seeing four framework sections and maps

Figure 1: SAIF Concept Map

SAIF combines four sub-frameworks for defining and managing comparable interoperability specifications.

- The **Information Framework** (**IF**) defines information and terminology models, metadata, value sets and schemas that specify the static semantics of interactions. This includes patterns for structured and unstructured data, documents, messages and services, quality measures and transformations. The IF *scope* includes the needs of direct clinical care,

---

1 **Working Interoperability** is an instance of two "trading partners" -- human beings, organizations, or systems, successfully exchanging data or information, or coordinating behavior to accomplish a defined task, or both.

2 Coherent implies clear, complete, concise, correct and consistent.

3 An **enterprise architecture (EA)** is a rigorous description of the structure of an enterprise. EA describes the terminology, the composition of subsystems, and their relationships with the external environment, and the guiding principles for the design and evolution of an enterprise. This description is comprehensive, including enterprise goals, business functions, business process, roles, organizational structures, business information, software applications and computer systems.

supportive[4] and information infrastructure areas. The information and terminology models, metadata, vocabularies and value sets specify the static semantics for expressing concepts, relationships (including cardinalities), constraints, rules, and operations needed to specify data, data type bindings, vocabulary and value set bindings.

- The **Behavioral Framework** (**BF**) defines constructs that specify the dynamic semantics of interactions in an interoperability specification. The BF *focus* is the accountability required to achieve working interoperability. Accountability is a description of "*who does what when*." Accountability manifests itself as implicit or explicit contracts at business object, component, application, system and enterprise boundaries. BF accountability is described by the relationships among various stakeholders and system components, applications and their system roles. These relationships involve information exchanges and state changes within use case scenarios.

Jointly, the IF and BF allow the interoperability specification of business objects, components and their services, capabilities, applications, systems and their respective roles, responsibilities and information exchanges.

- The **Governance Framework (GF)** *purpose* is to manage risk by relating decisions and policies, to the IF and BF within the ECCF. The GF scope includes Precepts[5] (e.g., objectives, policies, standards, and guidelines), Entities (e.g., people, organizations and systems), Processes and Metrics. The GF defines expectations, grants power and resources, verifies performance and manages configuration baselines. Governance consists of either a separate process or parts of management or leadership processes; a governing board or council may be established to execute or oversee these processes.
- The **Enterprise Conformance and Compliance Framework** (**ECCF**) *goal* is to ensure Working Interoperability (**WI**) among various healthcare organizations. The ECCF *purpose* is to manage the relationship between architectural artifacts and implementations of those artifacts to insure compatibility[6] among healthcare systems. The *objective* of a fully qualified ECCF is to be a coherent and traceable interoperability specification, which is easy-to-use. The ECCF can be an assessment framework, which supports configuration management baselines, development status, audit compliance and risk assessments throughout a business-capability lifecycle. The ECCF can be used to specify information exchange interoperability and conformance statements for documents, messages and services. An ECCF provides a template, called a Specification Stack (**SS**) that allows you to specify business object, component, capability, application and system interoperability. An ECCF is organized as a matrix of Dimension columns (Enterprise, Information, Computational, Engineering and Technical) and Perspective rows (Conceptual, Logical and Implementable).

The ECCF is the centerpiece of SAIF. It supports both technical interoperability (IF and BF) and the GF management of interoperability. The SAIF ECCF provides external stakeholders with a coherent picture of exactly what is required to interoperate with an organization's software

[4] Support includes research, analysis, workload, workflow, business process, performance, etc.

[5] A **Precept** (from the præcipere, to teach) is a commandment, instruction, or order intended as an authoritative rule of action.

[6] **Compatibility** is a relationship between two or more conformance statements involving two or more specification stack instances. The relationship identifies whether two or more implementations certified to be conformant to the specification stack instances can achieve WI without further transformations. If so, the two SS instances and associated implementations are called *compatible*.

components. A given component's specification is SAIF-compliant if it is compliant with an organization's SAIF implementation guide. The ECCF IG should require "just enough" compatibility to enable the desired level of interoperability[7] for appropriate SS type.

1.2 SAIF Implementation

Any organization choosing to implement SAIF should assemble its own SAIF Implementation Guide (**IG**). An organization's SAIF IG should interpret and localize the canonical constructs defined in this HL7 SAIF book.

SAIF defines the grammars[8] and patterns[9] common to all ECCF Interoperability Specification Stack (**SS**) instances. Each organization should document how to instantiate and guide the population of its interoperability SSs. Note that just as an enterprise may have systems-of-systems, an interoperability SS may reference and be built from component SSs. Additionally, for different SS types[10], an IG may require different SS architectural artifact profiles[11]. This means that an SS for a complete solution may reference SSs for more primitive building blocks, where each interoperability SS type may contain or reference different numbers and different types of artifacts.

Table 1 is a sample template, which shows a super set of common architectural-artifacts within an ECCF SS. As appropriate, within each cell, you might

1) place or reference and discuss appropriate architectural artifacts and specifications,
2) define conformance statements, which are testable-representations of the specifications,
3) assert, as true or false, that one-or-more conformance statements are met
4) manage traceability within columns and consistency across rows.
5) do **Topic Maps** among viewpoints and architectural artifacts to define traceability
6) do **RACI Charts** for each viewpoint to define stakeholder roles and responsibilities
7) identify and mitigate risks.

SS maturity implies that an SS instance is coherent and traceable within-and-across the SS. SS maturity does not require complete coverage of all cells in the SS; rather, coverage should be "fit-for-purpose". Examining relevant SS instances provides a scalable approach to assessing the risk or degree of difficulty and specific amount of effort required to enable trading partners to attain Working Interoperability.

---

[7] **Levels of Interoperability** [Center for Information Technology Leadership]

1. Viewable (e.g., paper based)
2. Machine Transportable (e.g., electronic form, such as PDF)
3. Machine readable structured messages with unstructured content
4. Machine interpretable structured messages with standardized content

[8] **Grammar** is the set of rules that deal with syntax and semantics of interoperability specifications.

[9] **Pattern** is the SS cell placement of architectural artifact types.

[10] **SS types** include business objects, components, capabilities, systems, enterprises

[11] **SS profiles** define the fit-for-purpose architectural artifacts that are distributed across the ECCF matrix of Dimensions (columns) and through the ECCF Perspectives (rows) for different SS types.

ECCF	Enterprise Dimension "Why" - *Policy*	Information Dimension "What" – *Content*	Computational Dimension "How" - *Behavior*	Engineering Dimension "Where" - *Implementation*	Technical Dimension "Where" - *Deployments*
Conceptual Perspective	✓ Inventory of o Use Cases, Contracts o Capabilities-Services o Stakeholders o Non-Functional Rqmts o Methodologies/Processes ✓ Business Vision, Scope ✓ Business Objectives ✓ Policy & Regulations	✓ Inventory of o Domain Entities o Stakeholders, Roles, o Activities, o Associations, o Information Rqmts o Information Models • Conceptual • Domain	✓ Inventories of o Capabilities-Components, o Functions-Services. ✓ Requirements o Accountability, Roles o Functional Rqmts. Profiles, Behaviors, Interactions o Interfaces. Contracts ✓ Conceptual Functional Service Specifications	✓ Inventory of o SW Platforms, Layers o SW Environments o SW Components o SW Services o Technical Rqmts o Enterprise Service Bus ✓ Key Performance Parameter Matrix	✓ Inventory of o HW Platforms o HW Environments o Network Devices o Communication Devices ✓ Technical Requirements
Logical Perspective	✓ Applicable Rules ✓ Use Case Specs ✓ Governance ✓ Implementation Guides ✓ Technology Neutral Standards ✓ Wireframes of o Architectural Layers o Components and o Associations ✓ Contracts	✓ State Variables ✓ Information Models o Localized o Constrained o Project ✓ Vocabularies ✓ Value Sets ✓ Content Specifications o Messages o Documents o Services	✓ State Machines ✓ Specifications o Use Cases, Interactions o Components, Interfaces ✓ Collaboration Participations ✓ Collaboration Types ✓ Function Types ✓ Interface Types ✓ Collaboration Scripts ✓ Service Contracts	✓ Models, Capabilities, Features and Versions for o SW Environments o SW Capabilities o SW Libraries o SW Services o SW Transports	✓ Models, Capabilities, Features and Versions for o HW Platforms o HW Environments o Network Devices o Communication Devices
Implementable Perspective	✓ Business Nodes ✓ Business Rules ✓ Business Procedures ✓ Business Workflows ✓ Technology Specific Standards	✓ Schemas for o Databases o Messages o Documents o Services o Transformations	✓ Automation Units ✓ Technical Interfaces ✓ Technical Operations ✓ Orchestration Scripts	✓ Application Specs ✓ GUI Specs ✓ Component Designs ✓ SW Deployment Topologies	✓ Deployment Specifications, ✓ Deployment Topology ✓ Execution Context ✓ Application Bindings ✓ SW Platform Bindings

Table 1 Notional Super Set of Architectural Artifacts within an ECCF SS

An IG specifies which types of architectural artifacts are required by an organization, program, project, etc. Obviously, an enterprise SS will have different artifacts than a component or business object. In Table 1, a notional super set of architectural artifacts are distributed across the ECCF Dimensions (columns) and through the ECCF Perspectives (rows). "Fit-for-purpose" criteria should be used to determine the appropriate architectural artifacts for a particular SS type. Artifacts are first placed in the most intuitively obvious SS cell and then are organized to facilitate horizontal consistency and vertical traceability. A "mature" or "fully-qualified" interoperability SS need not be densely populated; but, it shall contain a coherent, traceable and easy-to-use set of architectural artifacts. For instance, the Enterprise Dimension is primarily bound to the Conceptual Perspective and the Engineering and Technical Dimensions are primarily bound to the Implementable Perspectives; their other Perspectives may be sparsely populated. Key to understanding SAIF is`the relationship of the IF, BF and GF to the SAIF Dimensions and the relationships among the Dimensions needed to achieve coherency, traceability and ultimately working interoperability. The viewpoints of Table 1 are categorized along the following criteria:

- **Enterprise Dimension (ED)** defines the business and reference context and is concerned with the purpose and behaviors of the subject SS type as it relates to the organization's business objectives and the business processes. This dimension answers the question "why" and refers to policy. Note that the ED is closely related to the overall Conceptual perspective. In particular, you should provide appropriate linkages among the ED Perspective viewpoints and Conceptual Perspective of the Information and Computation Dimensions.
    - **The ED Conceptual Perspective** viewpoint is primarily useful to project sponsors, project managers, program directors, IT directors and requirements analysts.

- **The ED Logical Perspective** viewpoint is primarily useful to project managers and business process experts.
- **The ED Implementable Perspective** viewpoint is primarily useful to implementation managers, compliance staff and auditors.

- **Information Dimension (ID)** is defined by one-or-more domain analysis models and is concerned with the nature of the information handled by systems and constraints on the use and interpretation of that information. This dimension answers the question "what" and refers to information content.
    - **The ID Conceptual Perspective** viewpoint is primarily useful to Clinicians and Clinical Analysts.
    - **The ID Logical Perspective** viewpoint is the ID core. It is primarily useful to clinical Informaticists and Architects.
    - **The ID Implementable Perspective** viewpoint is primarily useful to information modelers, implementers, compliance staff and auditors.
- **Computational Dimension (CD)** is concerned with the functional decomposition of the system into a set of components that exhibit specific behaviors and interact at interfaces. This dimension answers the question "how" and deals with behavior.
    - **The CD Conceptual Perspective** viewpoint is primarily useful to business analysts and functional analysts.
    - **The CD Logical Perspective** viewpoint is the CD core viewpoint and is primarily useful to System Engineers, architects and Business Process Modelers.
    - **The CD Implementable Perspective** viewpoint is primarily useful to system integrators and solution implementers.
- **Engineering Dimension (ED)** is defined by existing platform capabilities and is concerned with the mechanisms and functions required to support the interactions of the computational components. This viewpoint answers the question "where" and refers to the software (**SW**) implementation environments. Note that the engineering viewpoint is closely related to the overall Implementable Perspective. The use of reusable components and services or an Enterprise Service Bus (ESB) may naturally fit into this Dimension.
    - **The ED Conceptual Perspective** viewpoint is primarily useful to Enterprise Architects.
    - **The ED Logical Perspective** viewpoint is primarily useful to Application Architects.
    - **The ED Implementable Perspective** viewpoint contains the core ED content and is primarily useful to Application Developers and Deployment Engineers.
- **The Technology Dimension (TD)** is concerned with the explicit choice of technologies for the implementation of the system, and particularly for the communications among the components. This viewpoint answers the question "where" and refers to the hardware (HW) deployment environments.
    - **The TD Conceptual Perspective** viewpoint is primarily useful to enterprise architects.
    - **The ED Logical Perspective** viewpoint is primarily useful to Solution Architects.
    - **The ED Implementable Perspective** viewpoint is the ED core and is primarily useful to deployment engineers.

2 Enterprise Conformance and Compliance Framework

2.1 Overview: The purpose of the ECCF

a. The Enterprise Conformance and Compliance Framework (ECCF) is formally defined as one of the four "grammars" of the Service-Aware Interoperability Framework (SAIF), a set of grammars that collectively can be used to *explicitly* define the various aspects of a given complex "component" – a term that is intentionally left somewhat vague in terms of its scope, intent, implementation strategies, etc. so as to include systems, sub-systems, software components, and "exchange standards" such as HL7 messages or documents, CDISC interchange structures, etc. – when the component is viewed in the context/from the perspective of interoperability with other "components." In addition – as has been thoroughly discussed by several organizations including the European Union Division of Healthcare Interoperability and the Australia NeHTA (National eHealth Transition Authority) – the concept of "interoperability" itself can be viewed from a number of perspectives including cultural, organizational, informational, technical, etc. (See the general discussion of Working Interoperability (WI) in the SAIF Overview section of this document.) In turn, the term "interoperability" is also intentionally left relatively vague so as to cover several "types" (or degrees-of-difficulty) of interoperability including both syntactic and semantic interoperability as achieved in human-to-human, machine-to-human, human-to-machine, or machine-to-machine interactions. Readers familiar with the challenges of achieving generalized computable semantic interoperability (CSI) – i.e. semantic interoperability between machines without human intervention – will recognize that the other "types" of interoperability mentioned above are less demanding than CSI. As a consequence, the degree to which the potential for explicitness-of-expression that is *possible* through the use of SAIF, may vary considerably according to the interoperability requirements for the component-in-question, i.e. CSI-based WI vs other less rigorous types of interoperability.

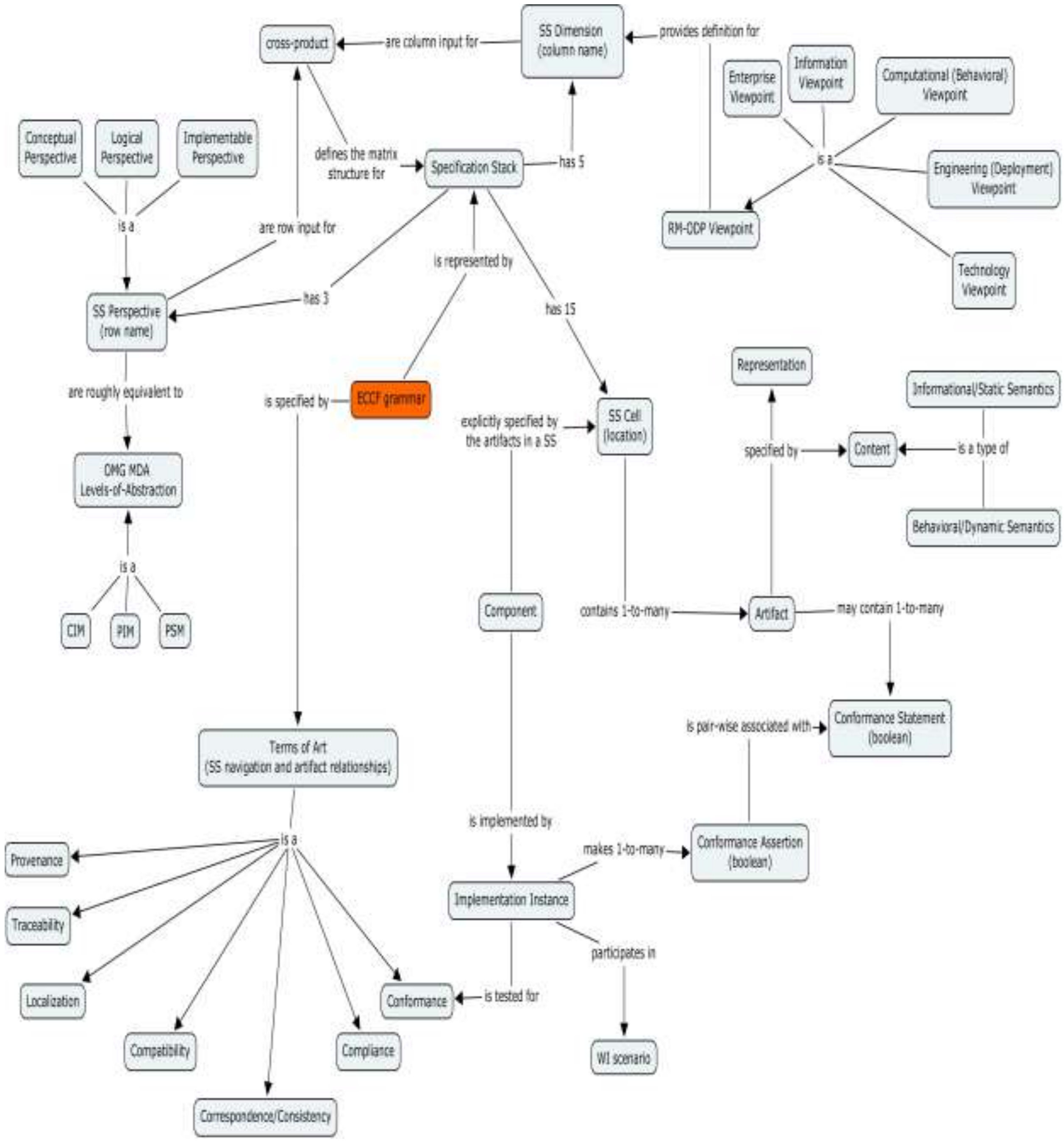
The overarching goals and focus of SAIF are discussed elsewhere in this document. However, it is worth mentioning in this chapter – whose focus is the grammar of the ECCF – how the grammars of the Information, Behavior, and Governance frameworks are related to and manifest in the grammar of the ECCF. The differences and relationships between the other three SAIF grammars and that of the ECCF is best illustrated by examining a one-sentence definition of the ECCF:

- *The ECCF is a collector of artifacts that in combination explicitly – and potentially fully – describe <u>from a number of different perspectives</u> the various informational/static and behavioral/dynamic characteristics of the component that are relevant to the component in a specific instance of Working Interoperability with another component.*

From this single definition, one can draw the following conclusions regarding the relationships between the ECCF grammar and the grammars of the Information, Behavioral, and Governance Frameworks including the following:

- The artifacts collected in a given ECCF artifact (the structure of which will be defined in a later section of this document) contain descriptions of a given component's informational/static and behavioral/dynamic semantics/features/functions.
- Specifications regarding a component's informational/static semantics et al are expressed using the Information Framework grammar.
- Specifications regarding a component's behavioral/dynamic semantics et al are expressed using the Behavioral Framework grammar.
- The overall management of the life cycle of each artifacts – whose content and representation must be defined in the context of a given organization's SAIF Implementation Guide (SAIF IG) – including the correctness and completeness of the artifact as well as the IG-specified RACI relationships for the artifact – are defined by the Governance Framework grammar.

b. The following Concept Map depicts the main concepts and relationships that collectively define and represent the grammar of the ECCF.


SS Dimension (column name)
cross-product
are column input for
provides definition for
Enterprise Viewpoint
Information Viewpoint
Computational (Behavioral) Viewpoint
Engineering (Deployment) Viewpoint
Technology Viewpoint
RM-ODP Viewpoint
is a
Conceptual Perspective
Logical Perspective
Implementable Perspective
defines the matrix structure for
Specification Stack
has 5
are row input for
has 3
is represented by
has 15
SS Perspective (row name)
are roughly equivalent to
OMG MDA Levels-of-Abstraction
CIM
PIM
PSM
ECCF grammar
is specified by
explicitly specified by the artifacts in a SS
SS Cell (location)
Representation
Informational/Static Semantics
specified by
Content
is a type of
Behavioral/Dynamic Semantics
Component
contains 1-to-many
Artifact
may contain 1-to-many
Conformance Statement (boolean)
is pair-wise associated with
Terms of Art (SS navigation and artifact relationships)
is implemented by
makes 1-to-many
Conformance Assertion (boolean)
Implementation Instance
participates in
is tested for
Provenance
Traceability
Localization
Compatibility
Correspondence/Consistency
Compliance
Conformance
WI scenario

Describing complexity: ***Structure of the ECCF Specification Stack***

The underlying “theory” or “motivation” for the ECCF grammar is somewhat in contrast to – or at least distinguishable from – that behind the specification of the Information and Behavioral Framework grammars, and – to a lesser degree – the Governance Framework grammar. In particular, in each of other frameworks, the focus on the grammar is to enable developers or consumers of particular components to more explicitly and expressively define certain aspects of those components, each viewed as from an artifact-specific perspective. Thus, for example, the IF grammar enables a developer to specify (or a consumer to learn about) various aspects of the component’s informational/static semantics. A similar perspective focused on behavioral/dynamic semantics is achieved through use of the BF grammar. Finally, through use of the GF grammar, organizations implementing SAIF can define organization-specific Precepts, People/Roles, Processes, and Metrics which enable the implementation of SAIF to be effectively and efficiently realized and managed.

In contrast to grammars associated with “building” various atomic items in a given SAIF implementation, the ECCF is focused on defining a grammar that enables collections of artifacts defined/specified using the IF, BF, and GF grammars to be *collected* so that – *in combination rather than in isolation* – the artifacts can a developer or consumer of a given component to understand explicitly the nature of the complexity of the component, as well to rationally evaluate and certify the degree to which a given implementation instance of the specification defined by the collection of artifacts is, in fact, realized by the implementation instance.

The notion of a “collection of artifacts” as being both necessary and sufficient to fully and explicitly describe a given component from the perspective of the component’s participation in a Working Interoperability scenario is, in turn, based on the well-established principle that complex systems are best described using a matrix which intersections multiple dimensions with multiple perspectives. In the ECCF, the “grammar” is therefore defined as follows:

- A specification and definition of the dimensions that will be used as– in the case of the ECCF – the columns of a “Specification Stack instance,” i.e. the matrix used to collect the artifacts that together define the WI-relevant characteristics of the component.
- A specification and definition of the perspective that will be used as– in the case of the ECCF – the rows of a “Specification Stack instance,” i.e. the matrix used to collect the artifacts that together define the WI-relevant characteristics of the component.
- An explicit definition of how a given Specification Stack instance – i.e. the collection of artifacts that together define the WI-relevant characteristics of the component-in-question – can be used in the context of certifying or otherwise validating the degree to which a given implementation instance in fact satisfies the specification.
- Explicit definitions – and the importance of explicitness cannot be over-emphasized as “the enemy of Working Interoperability is unspecified, implicit assumptions realized

inconsistently in implementations – of other terms or concepts that define the rules for navigating the cells of a given Specification Stack instance, i.e. that define the relationships between cells as well as between artifacts within a given cell. NOTE: the concepts defined here are not unique to relationships between specific instances of artifacts (those relationships are explicitly defined in a given organization's SAIF IG). Rather, ECCF navigational/relationship terms are concerned with general "meta-relationships" between classes of artifacts as will be explained in the "Terms of ECCF Art" discussion below.

a. The Dimensions (column names) of an ECCF Specification Stack

The Dimensions of the ECCF are taken from the ISO standard Reference Model for Open Distributed Process IRM-ODP, ISO/IEC IS 10746 | ITU-T X.900). In particular, the column names are the RM-ODP Viewpoints. Before providing a definition of each Viewpoint, it is important to emphasize that SAIF is not ODP nor is ODP SAIF. Each has a particular focus, perspective, and set of goals. In particular, RM-ODP provides a comprehensive framework for defining, designing, developing, and deploying large-scale, distributed software architectures. The scope, focus, and goals of SAIF are centered around Working Interoperability. Clearly, the problems and challenges of achieving WI do occur in the context of large-scale, distributed enterprise architectures. As such, SAIF can be viewed as a *complementary adjunct* to ODP (or, for that matter, any enterprise architecture framework, e.g. TOGAF, Zachman2, etc.).

The matrix that is the result of the intersection of Dimensions and Perspectives is called an ECCF Specification Stack (SS). Each SS instance has a particular scope, i.e. component, system, sub-system, specification that is defined via the collection of artifacts lying within the boundary of a single SS instance. The SS's scope is referred to as the Specification Stack Subject. For each cell in a Specification Stack instance, t is important to note that the cell can contain multiple artifacts which may or may not contain artifact-to-artifact links/relationships, and which may be hierarchical in terms of their levels of detail of abstraction.

i. **Enterprise Viewpoint:** This dimension focuses on defining salient aspects of the "organizational context" – in the WI context, more aptly named "the intra- or inter-organizational deployment/interoperability context – in which the specification-in-question is being defined. In particular, the Enterprise Viewpoint dimension should explicitly define – for each of the three Perspectives – aspects of the interoperability context that emerge from an understanding of business objectives and business rules including relevant pre- and post-conditions for interoperability scenarios. Due to the basic nature of the Enterprise Viewpoint dimension, most information at the Logical and Implementable Perspectives will have its source/origin in the Conceptual Perspective, i.e. very little "new" information is added at the Logical and Implementable Perspectives, Perspectives that are most productively contributed to via the Information and Computational Viewpoints

(Dimensions).

ii. **Information Viewpoint:** This dimension focuses on the informational/static semantics that are the responsibility of the component as those semantics relate to an instance of WI. As noted above, these semantics are expressed using various, relevant aspects of the Information Framework grammar and include constructs – discussed in greater detail in the IF chapter of this document – such as information and data models, data types, value sets, etc. It is important to note that the use of the IF is **not** scoped to the Information Viewpoint, i.e. the IF is used to specify any aspect of informational/static semantics that appear in any artifact throughout the ECCF and is not limited to expression of only those artifacts that a given SAIF IG defines as being located in the Information Viewpoint column of the Specification Stack.

iii. **Computational Viewpoint** (also referred to as the Behavioral Viewpoint in the context of SAIF)**:** This dimension focuses on defining the various behavioral/dynamic semantics of a particular component relative to a WI scenario. As noted above, these semantics are expressed using various, relevant aspects of the Behavioral Framework grammar and include constructs – discussed in greater detail in the BF chapter of this document – such as contract and interface specifications, accountability profiles, etc. It is important to note that the use of the BF is **not** scoped to the Computational Viewpoint, i.e. the BF is used to specify any aspect of behavioral/dynamic semantics that appear in any artifact throughout the ECCF and is not limited to expression of only those artifacts that a given SAIF IG defines as being located in the Computational Viewpoint column of the Specification Stack. In addition, it should be noted that the BF (as does the IF and ECCF, although to a lesser degree) makes extensive use of the ODP Enterprise Language, a set of well-defined concepts and constructs that are used to define various topics-of-interest in the ODP Viewpoints/ECCF SS Dimensions.

iv. **Engineering Viewpoint** (also referred to as the Deployment Viewpoint in the context of SAIF)**:** This dimension focuses on defining the possible deployment topologies involved in any of the possible WI scenarios into which the component would be placed. The ODP specification contains considerable detail with respect to a construct referred to as *transparencies.* It is beyond the scope of the SAIF canonical definition of the ECCF to discuss these constructs. However, there are certain SAIF IGs that could benefit substantially from inclusion in certain of the transparency constructs in their organization-specific IGs.

v. **Technology Viewpoint:** This dimension focuses on defining various implementable standards (hardware or software as relevant) which will ultimately support the specification. It may reference other SS cells to appropriately contextualize cell-specific artifacts. Further explanation/discussion of the application of the Technology Viewpoint dimension is more appropriately constrained to SAIF IGs including discussions regarding topics such as technology-specific deployment or configuration guides, technology selection criteria, and maintenance/migration plans. It should also be noted that Conformance Statements are not embedded in the Technology Viewpoint dimension as often as they are in the other dimensions.

b. The Perspectives (row names) of an ECCF Specification Stack

The perspectives of ECCF were chosen to be as general as possible (i.e. coarsely granulated as opposed to – for example – the more finely-granulated Perspectives of Zachman2), as to map in a general manner to the roles of Domain Expert and Analyst, Architect, and Developer. In particular, the artifacts that populate a given row of a SS instance should be developed in the context of a RACI (Responsible for developing, Accountability for development, Communicate the development to, and Interested in being Informed about) chart for each artifact. The specific artifacts developed and the details of the RACI chart that contextualizes them will be different for each organization implementing SAIF as defined in their organization-specific SAIF IT. *(For a further discussion of the relationship between roles and ECCF Perspectives, see the discussion in b.iv.)* As such, the definitions of the three SAIF Perspectives are as follows:

i. **Conceptual Perspective:** The artifacts in the Conceptual Perspective are those that are of interest to – and directly consumable/readable by – Domain/Subject Matter Experts (DEs/SMEs). As such, the artifacts are most commonly focused on the "Problem-Space" rather than the "Solution Space," and contain – distributed across the five columns of the Specification Stack – unambiguous descriptions of the various dimensions of the component/system that is the scope of the Specification Stack. The Conceptual Perspective is normally developed by "outward-facing analysts," i.e. analysts with reasonable domain knowledge who are capable of facilitating dialogues with DEs/SMEs, as well as taking the results of such dialogues and representing the content in structured – but still understandable to DEs/SMEs – artifacts, e.g. clearly-stated business rules, concept maps, simple UML class or activity diagrams, etc. A fully-specified Conceptual Perspective should be simultaneously readable/vettable by DEs/SMEs as well as rigorous enough to serve as input into the development of the Logical Perspective. (NOTE: Previous discussions of the SAIF ECCF have used the MDA-based term "Computationally-Independent Model (CIM). For a variety of reasons – most important of which is that SAIF does not formally use MDA in any way in its grammars – the term CIM is now deprecated.)

ii. **Logical Perspective:** Artifacts in the Logical Perspective represent traceable translations of Conceptual-level artifacts into a form/format usable by and useful to architects and "inward-facing analysts." Note that there is no firm or fixed line that definitively and unambiguously determines where the Conceptual Perspective ends and the Logical Perspective begins. (The same is true for the lack of definitive demarcation boundaries between the Logical and Implementable Perspectives.) Rather, for a given SAIF IG, the most important aspects of defining and locating SAIF artifacts at a given Perspective are the combination of role-based awareness based on artifact creation and consumption, in combination with IG-specific consistent placement of artifacts across multiple SS instances. (NOTE: Previous discussions of the SAIF ECCF have used the MDA-based term “Platform-Independent Model (PIM). For a variety of reasons – most important of which is that SAIF does not formally use MDA in any way in its grammars – the term PIM is now deprecated.)

iii. **Implementable Perspective:** The Implementable Perspective is normally the domain and purvey of developers, often in concert with dialogues with designers and/or architects. Note that the artifacts in this Perspective are not, *per se*¸ actual implementations, but rather *implementable,* i.e. contain all of the necessary technical bindings – e.g. data types, value sets, class libraries, interface specifications, etc. – that will enable one or more instances of the specification to be realized by one or more development teams. . (NOTE: Previous discussions of the SAIF ECCF have used the MDA-based term “Platform-specific Model (PSM). For a variety of reasons – most important of which is that SAIF does not formally use MDA in any way in its grammars – the term PSM is now deprecated.)

iv. **Perspectives and Roles:** *(Following is a more concrete discussion of the somewhat “soft” delineations between the three ECCF SS Perspectives. Specifically, the example uses the Information Viewpoint/Dimension and the Translational Medicine Continuum “from bedside to bench and back” as its reference point. However, specific mention of the “Information Dimension” of the artifacts for a given Perspective should be seen as exemplar and not exclusive to that Dimension. However, although it is theoretically possible to specify artifacts for any of the fifteen cells of an ECCF SS instance, the notion of all-inclusive, trans-Perspective artifact generation across all five SS Dimensions is most often seen more often for the Informational and Computational/Behavioral Dimensions than in any of the other three SS dimensions.) The material is taken from the NCI CBIIT SAIF Implementation Guide. A detailed discussion of the canonical grammar of the Information Framework can be found elsewhere in the SAIF Book.)*

The Information Viewpoint/Dimension is concerned with collecting the various artifacts – represented in various types of models including Concept Maps, UML class and instance diagrams, etc. – the informational/static semantics-of-interest from the perspective of a specific component involved in various WI scenarios. In particular, experience has shown that a broad range of stakeholders in the healthcare, clinical research, and life sciences domains have consider knowledge of and interest in informational/static semantics, and that explicit representation of these semantics are of considerable importance if one is to achieve computable semantic interoperability (CSI) in a loosely-coupled, widely distributed community. It is useful to divide the diverse group of stakeholders based on roles, and, in fact, the SS Perspectives should be viewed as “collectors of roles.”

The Conceptual Perspective (formerly referred to as the CIM row of the SS in previous discussions of the ECCF) collects the *definitional concepts* and relationships of Domain Experts/Subject Matter Experts, e.g. clinicians, trialists, and researchers using the language of those involved. Artifacts in the Conceptual Perspective are developed as the result of interactions between these stakeholder types and “outward-facing” business analysts. The Conceptual Perspective of the Information Dimension therefore contains artifacts such as the static semantic views of a Domain Information Model (scoped to the use of that model by the component that is the Specification Stack Subject), value set domains, etc.

The Logical Perspective (formerly to as the PIM row of the SS) focuses on the *logical representation* of the artifacts defined in the Conceptual Perspective’s Information Dimension for consumption by information architects. (NOTE: Additional concepts and constructs not present in the Conceptual Perspective will also be added by Information Architects as needed in order to fully define a logical information architecture with sufficient rigor to enable its consumption, transformation, and elaboration by the group of stakeholders that provide the specification’s Implementable Perspective.

Artifacts in the Implementable Perspective (formerly referred to as the PSM row of the SS) are focused on transformations of the logical (information-based) artifacts so that they can be bound to specific implementation technologies such as XML, java classes, etc.

c. **Specification Stack Subject:** Each instance of a Specification Stack is scoped to a particular “subject.” The use of the term “component” in Section I of this chapter provided the intentionally non-specific definition of this concept. The SS Subject makes – for a given SS instance – the definition specific for a given component instance, e.g. a service, interoperability specification, sub-system, system, etc. It is important to note that a SS instance can therefore have *any scope that is relevant and that needs to have its complexity more explicitly defined in terms of a collection of artifacts that are themselves sorted/categorized around the cross-product of ECCF-defined Dimensions and Perspectives.*

d. Conformance: Quoting from [ISO/IEC 10746-2*]: "Conformance relates an implementation to a standard. Any proposition that is true of the specification must be true in its implementation."*

The fundamental focus of the ECCF grammar is to provide a way for specification developers and their consumers to explicitly understand the collection of various aspects of a given component that impact the component’s use in a WI scenario. Specifically, it is the goal of the ECCF grammar to provide a grammar that enables an implementation instance of the specification to be evaluated as to its *conformance* to the specification. The notion of providing a framework in which implementation instances can be tested/evaluated as *conformant* to a given specification is defined in the ECCF using the ODP-derived notions of *Conformance Statements* contained within the artifacts within a given SS instance that collectively define a given component, and *pair-wise Conformance Assertions* that are made by an implementation instance against a given specification. (NOTE: Although the notion of Conformance is technically a concept that could properly be discussed in the section “Terms of Art: Navigation and Relationships in an ECCF Specification Stack,” it is included here because Conformance Statements and Conformance Assertions should be viewed as part of the structure of the ECCF that is thus a manifestation of the ECCF grammar.)

i. **Conformance Statements:** Quoting from [ISO/IEC 10746-2*]: "A conformance statement is a statement that identifies conformance points of a specification and states the behavior which must be satisfied at these points. Conformance statements will only occur in standards which are intended to constrain some feature of a real implementation, so that there exists, in principle, the possibility of testing."*
As adapted from ODP/ISO and applied in the SAIF context, Conformance Statements are Boolean statements made in the context of a given specification artifact, i.e. "requirements that the artifact explicitly expresses in a manner that makes them testable/verifiable as a Boolean statement." The conformance of a given implementation instance to a particular specification is thus able verified based on the truth value of a pair-wise Conformance Assertion (see below) made by an implementation instance against a given artifact-resident Conformance Statement within a given specification. <u>It is important to note that the requirement that each Conformance Statement be testable/verifiable, i.e. that each Conformance Statement be a Boolean statement does **not** require that the statement be testable by automated means. In particular, it is often the case that Conformance Statements made from the Conceptual Perspective – and particularly those made in the Enterprise dimension – may only be verifiable as True through human examination of a given implementation instance. Thus, the critical defining feature of a valid ECCF Conformance Statement is its Boolean testability and **not** its particular mode of verification.</u>

ii. **Conformance Assertions:** As indicated in the previous section on Conformance Statements, Conformance Assertions are made by a given implementation instance and are linked pair-wise to a Conformance Statement made in the context of a given artifact as part of a component specification with a given ECCF Specification Subject, i.e. within an artifact collected within a single ECCF Specification Stack instance. The pair-wise association of specification-resident Conformance Statements with implementation-instance-resident Conformance Assertions enables the creation of testing harness/user verification frameworks which thus enable a given implementation instance to be "certified" (aka "tested") as "conformant to a given specification." (see Conformance Testing discussion)

iii. **Conformance Testing:** - Quoting from [ISO/IEC 10746-2*]: "A Reference Point (RP) is a point in the specification which a specifier nominates to be a candidate Conformance Point, i.e. a place where behavior may need to be observed to determine conformance. A specifier may define many RPs in the specification but it may be that only a subset of these can be used for testing in specific scenario - and these are referred to as conformance points. (NOTE:* <u>In the SAIF context, the notion of an RP can be stated as "the statement(s) in a given ECCF SS artifact that that is referred to as an ECCF Conformance Statement.")</u>
ODP defines four broad categories of reference points:

- **Perceptual**: an RP where there is some interaction between the system and the physical world, e.g. human-computer interface.

- **Programmatic**: an RP where a programatic interface can be established to allow access to a function.
- **Interworking** : an RP where there is a physical communication channel through which information exchange can be monitored.
- **Interchange** - an RP where an external physical storage medium can be introduced into the system, e.g. in cases where information can be recorded on one system and then physically transferred, directly or indirectly, to be used on another system)

From the preceding discussion of Conformance Statements and Conformance Assertions, it should be clear that Conformance Testing, i.e. the process whereby a given implementation instance is evaluated to determine which of its various Conformance Assertions are, in fact, valid implementations of a given specification's Conformance Statements, is:

- *a granular construct* ,i.e. is determined at the level of individual Conformance Assertions made by the implementation instance and ***not*** a global characteristic of a given implementation instance (unless, of course, the specification contains only a single global Conformance Statement against which the implementation instance can claim conformance); and
- *exists in a 1-to-many relationship between specifications and implementations,* i.e. there is a 1-to-many relationship between a given ECCF Specification Stack instance – i.e. the collection of artifacts that together explicitly describe a given component and its requirements, expressed in terms of both behavioral/dynamic and informational/static semantics and associated Conformance Statements – and the collection of implementation instances that can claim conformance to the specification.
- NOTE: The term "conformance" can be somewhat confusing as it can be used as a noun – e.g. "an implementation is in conformance with (or, alternatively, "conformant to") a given set of Conformance Statements made by a given specification – a verb – e.g. a given implementation instance's conformance is being evaluated – or as an adjective to describe a particular kind of examination of a given instance – e.g. the implementation is undergoing "conformance testing." The latter concept is also termed "evaluation" or "certification of conformance." The ECCF grammar defines the term *conformance* as either a noun or adjective (the two usages of the term are essentially synonymous as they describe a given implementation relative to a given specification.

e. Defining Specification Artifacts: ***Content, Representation, Specification Stack Location:*** As indicated above, the canonical representation of SAIF does not specify the content, representation, or location of individual artifacts. Artifact specification is, instead, done in the context of a given organization's SAIF IG. (Note that several SAIF IGs have been/are being developed by HL7, the Department of Defense, Canada Health Infoway, Australia NeHTA (National eHealth Transition Authority), and the Center for Biomedical Informatics and Information Technology (CBIIT) of the NCI and are generally available for review and study.) In general, however, it can be said that the most important aspect of artifact specification is its *content,* followed by it *representation.* Its *location* in a given SS instance is really only of major importance with respect to the *consistency* of the location of a given artifact (or, more correctly, artifact type) across multiple SS instances. It should also be noted that a given artifact may occur in more than one SS cell, a reflection of the fact that the Dimensions and Perspectives of the SS matrix are not normalized (as would be the case, for examples, if the SS were instantiated using the Zachman2 matrix of Dimensions x Perspectives). From the perspective of WI, normalization and cell-specific location are not, in fact, as important as explicitness and consistency.

ECCF Terms-of-Art: Navigation and Relationships in the ECCF Specification Stack

There are a number of "terms-of-art" which define specific processes, constructs, and navigational relationships between artifacts contained in an ECCF Specification Stack. Although the operational details of each of these terms are not fully realized in the canonical definition of the ECCF, the specific meanings of the terms are part of the formal definition of the ECCF grammar, of equal importance to the structural definitions discussed in the previous section. The definitions for a number of the ECCF terms-of-art are either taken directly from, or used with appropriate modifications for the SAIF context, the ODP specification, an attribution that is noted when apropos in the following definitional list.

a. Conformance: *(see discussion in ECCF Structure section above).* The most salient aspect of Conformance is that it links a given implementation instance to a given specification instance through a relationship defined by the verified truth of the implementation instance's Conformance Assertions as made against the specification's Conformance Statements. (NOTE: Conformance can be viewed as a specialized instance of the larger concept of Correspondence in the sense that there is a formal relationship between a specification's Conformance Statements and an implementation instance's pair-wise Conformance Assertions. Conformance is a form of Correspondence.

b. Compliance: Quoting from [ISO/IEC 10746-2*]: "Requirements for the necessary consistency of one member of the family of specifications or standards with another are established during the standardization process. Adherence to these requirements is called compliance."*

In the context of SAIF, Compliance is used to refer to logical consistency/correspondence between a source artifact and a target artifact with the target having undergone a transformation (usually a restriction), i.e. given an existing source artifact (e.g. a specification, standard, etc.) and a target artifact that resulted from applying a known transformation to the source, the target is in Compliance with the source if the transformation is considered "legal" by the source artifact's originator. Compliance can therefore be established between artifacts in a single SS cell or, alternatively, across multiple SS cells. When a Compliance relationship crosses cell boundaries, it can do so either horizontally or vertically (diagonal Compliance is also possible although less common then vertical or horizontal Compliance relationships.) Thus, localization is considered a form of Compliance.

NOTE: Unlike Conformance, Compliance is seldom overtly tested since non-compliant transformations producing non-compliant artifacts usually cause other issues which can be discovered in either Correspondence monitoring or Conformance testing.

c. **Certification/Conformance Testing:** *(see discussion in ECCF Structure section above).* It is important that the *process* of Conformance Testing not be confused with the *results* of that testing, i.e. a certification of Conformance (or lack thereof) based on the ability of a given implementation instance to satisfy one or more of the Conformance Assertions made by the implementation instance against the statement's pair-wise Conformance Statement in the specification.

d. **Correspondence/Consistency:** Quoting from [ISO/IEC 10746-2]: *"Viewpoint correspondence is a statement that some terms or other linguistic constructs in a specification from one ODP viewpoint are associated with (e.g. describe the same entities as) terms or constructs in a specification from a second ODP viewpoint. The forms of association that can be expressed will depend on the specification technique used."*
In the SAIF ECCF, Correspondence can be used synonymously with the term *consistency* as both are, in turn, focused on the notion of *logical coherence* of a given Specification Stack instance, i.e. an SS instance that is "unified" in its expression of a given component's various Dimensions and Perspectives. Thus, a "logically coherent" specification demonstrates a high degree of correspondence between its various components, a somewhat hard-to-define but relatively easy (to the trained eye) to perceive "expressive traceability." In summary, the notion of *Correspondence* underscores the fact that the Dimensions of a Specification Stack are **not** orthogonal, but rather express different aspects of a single component, system, sub-system, specification, etc.

It is also worth noting that both *Conformance* and *Compliance* can be viewed as "types" of *Correspondence/Consistency* as each of these ECCF Terms-of-Art refers to a form of logical coherence across a given specification and its collected artifacts (and, in the case of *Conformance*, its implementation instances as well). From the definition of *Compliance* (above), one can also see that *Correspondence* is a particular/specialized form of *Compliance* applied across Specification Stack Dimensions.

e. **Traceability:** In everyday parlance, *traceability* refers to the ability to link an instance with a concept, e.g. a requirement with an implementation-resident functionality. In the context of SAIF, *traceability* has a somewhat more formal meaning: *Traceability defines the relationship that links an attribute or other defining feature of a particular artifact defined in a particular Dimension and at a particular Perspective – including but not limited to semantics or Conformance Statements.* NOTE: Traceability is *vertical* relationship spanning all Perspectives and including any implementation instances associated with a given specification. As such, Traceability includes both Conformance and Compliance relationships.

f. **Provenance:** Documentation that identifies the "reverse traceability" of an existing artifact from its current state to its origination, including whatever attribution and/or context is associated with its various lifecycle changes. As such, *provenance* is, among other things, the source for documenting the various constraints/localizations that a given item undergoes as it moves from (for example) a Conceptual to a Logical to an Implementable SS artifact.

g. **Localization:** A specialization of Compliance whereby some aspect of an artifact's semantics – informational/static or behavioral/dynamic – or other defining attribute is restricted compared to its original occurrence. Localization commonly occurs as a concept passes from the Conceptual Perspective to the Logical Perspective, the Logical Perspective to the Implementable Perspective, and/or the Implementable Perspective to an implementation instance.

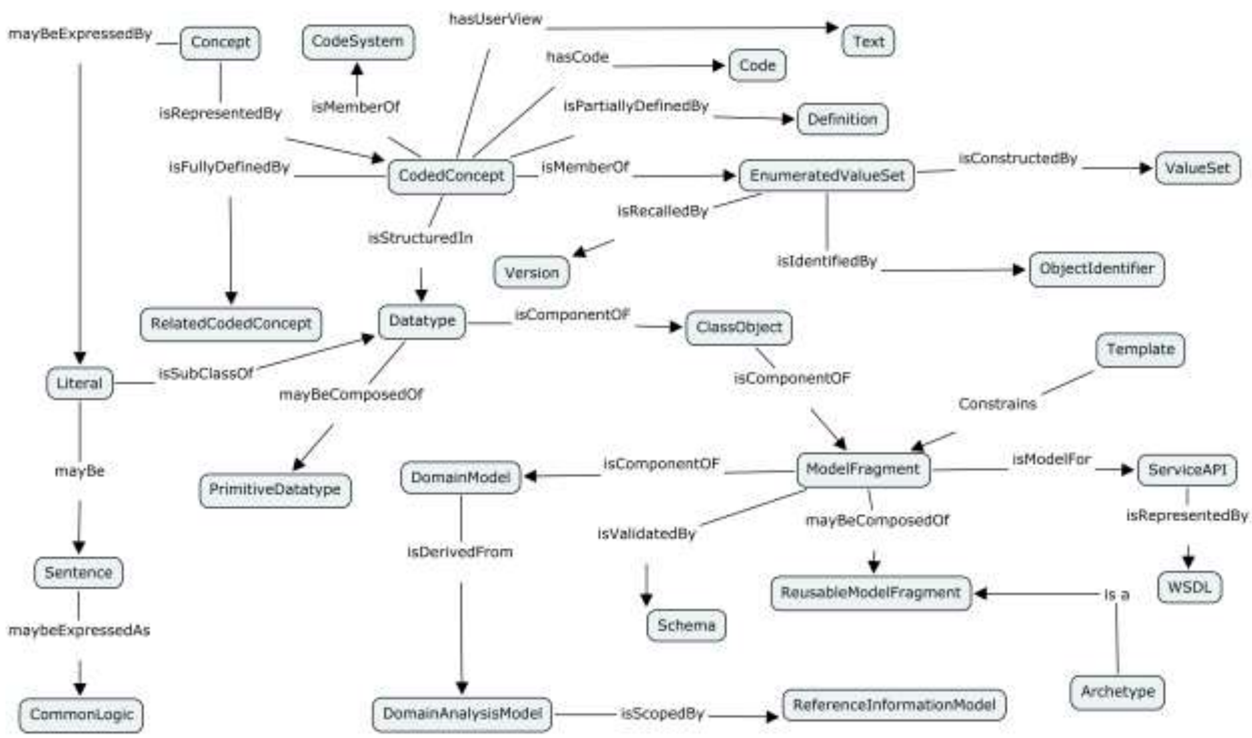
h. **Compatibility:** Given a specification, two implementation instances are said to be *Compatible* if-and-only-if they can successfully engage – *without further modification of their implementation specifics* – in any WI scenario that can be expected to be supported based on the reference specification that is implemented by the involved instances. In other words, two implementation instances are said to be *Compatible* if they do not "localize" by specifying contractor/non-interoperable constraints.

3 Information Framework

3.1 Overview

The information framework chapter defines – at a canonical level – the artifact types and inter-relationships of the Informational Viewpoint from the three SAIF Perspectives.

The concept map below shows the artifact topics that will be discussed in this chapter and the relationships between these artifacts.



3.2 Goals

The goal of the information framework is to describe how the static information of importance to a given domain and the experts within that domain is captured and refined through a traceable process to yield an implemented or implementable information artifact. This implementable information artifact, when developed using the artifacts defined in this framework, delivers the static semantics that contribute to the definition of computable semantic interoperability between systems. The information definitions contained in these artifacts are repeatable, yielding consistency across the range of information modeling tasks encountered within an organization.

Audience and Prerequisites

The audience for this discussion includes the participating domain experts, analysts, architects, developers, quality assurance practitioners, and implementers. All of these roles are typical participants of any software development effort.

Prerequisites for fully understanding the concepts in this document include the basic familiarity with the following:

- SAIF Enterprise Conformance and Compliance Framework (ECCF)
- The four pillars of Computable Semantic Interoperability (CSI)
- The concepts of refinement, constraint and localization,
- System design, Enterprise Architecture, development, and experience with the Unified Modeling Language (UML)
- Familiarity with core principles and applications of Service-Oriented Architecture (SOA)

3.2.1 Information framework essentials

The following descriptions are taken primarily from the NEHTA Information Framework document and are particularly relevant to this information framework chapter.

This section identifies a number of fundamental information principles that form the basis for the IIF. This covers:

- Separation of Information and Knowledge;
- Separation of representation form and interpretation of Information;
- Separation of Information and Data;
- Separation of formal concept representation and Clinical linguistics.
- Traceability from information concepts to organisational/technical concepts and patterns.

The first principle states that information and knowledge are distinct, although related concepts. We use the ISO standard [ODP-RM] as a basis of our definition of information, i.e.

> *Information is any kind of knowledge that is exchangeable amongst users, about things, facts, concepts and so on, in a Universe of Discourse*.

In general, information can be considered to be raw data that has a number of properties:

(1) has been verified to be accurate and timely

(2) is specific and organized for a purpose,

(3) is presented within a context that gives it meaning and relevance, and which

(4) leads to increase in understanding and decrease in uncertainty. The value of information lies solely in its ability to affect a behavior, decision, or outcome.

We take knowledge to mean an 'awareness or familiarity gained by experience, of a person, fact or thing', (Oxford Dictionary). Note that knowledge has an anthropomorphic nature and that its essence is about understanding of real world phenomena, which can be done through experience, e.g. through perception, learning through passing of information by others, or through a mental process. Not all knowledge is exchangeable, for example tacit knowledge.

The second principle states that information has a representation form. This is what makes information communicable. However, it is the interpretation of this representation (meaning) that is relevant in the first place [ODP-RM]. This is because the interpretation can generate some new knowledge. For example, through a medical observation process, a clinician captures key details about a patient, and records them using some representation form, typically written text (either in paper or electronic media). This capture forms information about the patient and the main purpose is to do some interpretation of what was recorded, i.e. patient diagnosis. This can be done by the very clinician who did the observation (based on his existing clinical knowledge) or after passing this information to other specialists for further observation and/or interpretation. It is through this chain of events that new knowledge (about health state of the patient) is generated.

The third principle further refines the second principle above, regarding the Representation Form of information and defining Data [ODP-RM], i.e.:

Data is the representation form of information dealt with by information systems or users thereof.

The definition above implies that there are two perspectives, a human perspective and an information system perspective and it is helpful to think about to sub definitions for data.

Human interpretation: Information in raw or unorganized form (such as alphabets, numbers, or symbols) that refer to, or represent, conditions, ideas, or objects. Data is limitless and present everywhere in the universe.

Machine interpretation: Symbols or signals that are input, stored, and processed by a computer, for output as usable information.

This latter definition is of critical importance and is something that is often forgotten as we think about semantic interoperability. This is because we are used to gathering the context of our data in human terms and fail to realize that the nuance of the context that we provide must be completely reproduced in a form that a machine can understand. In order to reproduce the human context we must be explicit in the data structures that we use and the information models in which we place those data structures so that the reconstruction of that human context (as close as possible) can occur at the other end of the electronic packet and be understood by the machine as well as by a human interpreter.

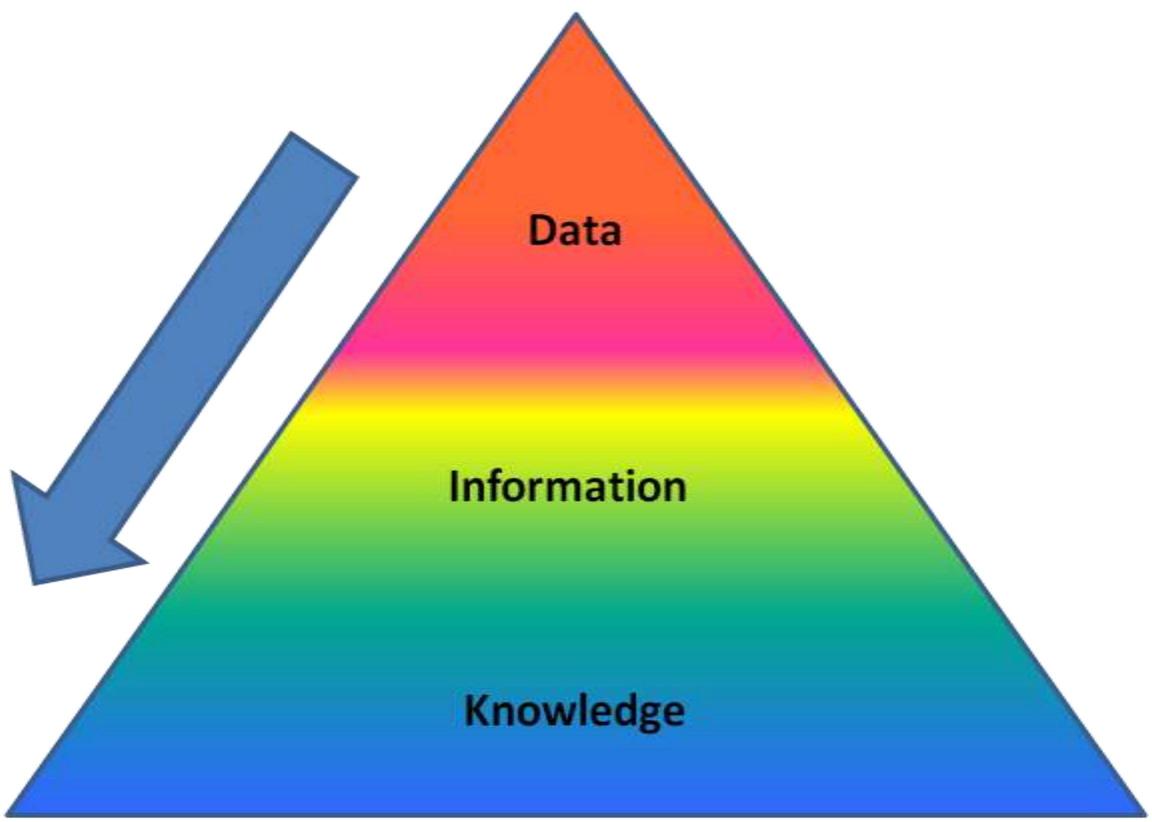
Data is the plural of datum, and datum is the elemental building block of information. The typical medical definition of data is a single observation about patient, such as the result of a temperature measurement, a height or weight, or a blood pressure measurement. If we think about these examples for just a moment we realize that blood pressure is clearly more than one datum because a blood pressure observation is made up of both a systolic and diastolic variable. If we think harder about these data, we realized that each of them is a composite structure made up of at a minimum a numeric value and some unit of measure. This is important because we realize that very little that we do is constrained to datum. In fact, we can argue that given a numeric value and some associated unit of measure that we in fact have information. This combination of the numeric value and its units of measure form an information structure that would be expressed in a complex datatype.

Although in general, information systems can be any system which collects and stores information, in e-health, the aim is to represent data in an electronic form for subsequent electronic processing. An example is terminology inference as in terminology classifications.

The fourth principle is based on [Rector]. It states that, although formal concepts should be informed by clinical linguistics, they should be treated differently, because their users and their purpose are different. Formal concepts systems, such as various terminology systems (using different formalisms), have the purpose of machine-based processing and inferences of formal concepts, while clinical linguistics, has the purpose of expressing or understanding natural language concepts (i.e. words, lexicons, grammars) for the use of clinicians.

The fifth principle states that all information components represent entities from the real world as modelled in the organisational perspective; furthermore some information components will be used by technical components implementing business logic.

The diagram below shows the continuum of data transformation to knowledge that is accomplished by and dependent upon the organization of data in information models and contextual descriptions provided by standardized terminology applied to the data concepts.



The following sections describe the artifacts of interest in the Information Framework. These artifacts are not specific to any particular information modeling paradigm but are required by any modeling paradigm to describe the static semantics of the computable semantic interoperability between two or more systems.

- Domain analysis models
- Reference information models
- Domain information models
- Serializable information models
- Localized information models
- Types - classes, attributes, data types, semantic type
- Vocabulary (including value sets and value set bindings to attributes)

Information models are normally built using a top-down approach or a bottom-up approach depending upon the modeling paradigm used to set context around data elements. There are exceptions to this bottom-up or top-down approach however for information modeling. For example, the ISO-11179 Part 3 metadata specification is a "middle-in" approach where a class forms the data element and attributes of that class are data element concepts. The data element is given context by association with an object class.

The SAIF approach to modeling the static information of a services aware specification stack is a top-down approach. In the SAIF world, a conceptual domain analysis model constrains the association of classes to eventual data element concepts that will be bound at the implementable perspective to valueSets .

3.2.2 Domain analysis model

A domain analysis model is a conceptualization of an area-of-interest expressed in a language that is familiar to the groups who normally work in that domain. The domain analysis model is an abstraction of the information model that captures the business of a particular domain. A domain analysis model may be represented at various levels of granularity and can have multiple layers with refinement occurring from a highly conceptual representation such as a concept map to a more formal representation in UML. We should note that a DAM contains both informational/static and behavioral/dynamic semantics, but that in the context of this chapter, we are only concerned with the informational/static semantics, i.e. only a “piece” of a fully-specified DAM.

A domain analysis model becomes part of the formal information model through its mappings to the semantics of a reference information model.

3.2.3 Reference Information Model

A reference information model is a critical component of any information development process. It represents an model of all possible information in a domain through the representation of abstract classes of information. It is the root of all formal information models and structures developed and allows for the mapping of both formal and informal information models (such as domain analysis model classes and attributes) to a common reference point.

From an information model traceability perspective, it is the root of the model tree. The use of a reference information model allows a model-driven methodology in which a network of inter-related models is developed.

The reference information model provides a static view of the information needs a broad sector of the real world. It includes class and state-machine diagrams and is accompanied by use case models, interaction models, data type models, terminology models, and other types of models to provide a complete view of the information requirements for that sector. The classes, attributes, state-machines, and relationships in a reference information model are used to derive domain-specific information models that are then transformed through a series of constraining refinement processes to eventually yield a static model of the information content of a specific implementable model for data exchange or persistence.

The abstract nature of a reference information model allows for local extension through the refinement of the abstract classes of the reference information model to meet specific information needs of a micro-domain which may or may not be reusable in other domains.

If a reference information model is sufficiently abstract at its root classes and can be extended through vocabulary definitions of class contents then it can be made applicable to any conceivable healthcare system information interchange scenario. In fact, if the reference information model is abstracted to a coarse level of entities and the relationships of those entities through roles to the actions that they somehow participate in then it can be conceptually applicable to any information domain or sector. One can think of a reference information model as an “upper ontology” that describes the static semantics of all possible real world information.

3.2.4 Domain information model

This is the first level of constraint below the reference information model. This model is created by mapping a domain analysis model to the reference information model, data types and terminology concepts to meet the requirements of a particular problem domain. A domain information model may have multiple entry points because it reflects all of the concepts of a particular domain analysis model. As such, a domain information model is not a directly implementable model, and is a fairly general statement of a domain with fairly general vocabulary bindings.

3.2.5 Serializable information model

A serializable information model represents a second level of constraint, based on specific use cases. Serializable information models must have single entry points and navigation paths that allow them to be traversed and unambiguously serialized for a specific implementation target (XML, Java, etc.). A serializable information model is focused on a specific operation or capability rather than an entire subject area or topic. Serializable information models are either derived from a domain information model directly, or from another serializable information model.

Since the serializable information model covers a relatively narrow information type, some of these models will be reusable across multiple information models. An example of this might be the demographics related to a person or an information model that describes the administration of a medication to a patient.

3.2.6 Localized information model

Localized information models are a constraint model that has a single entry point. However localized information models differ from serializable information models in that local information models may be incomplete models for any particular topic area. An incomplete model is one that addresses constraints for only a sub-set of the elements that are contained in the serializable model or domain information model from which it is derived.

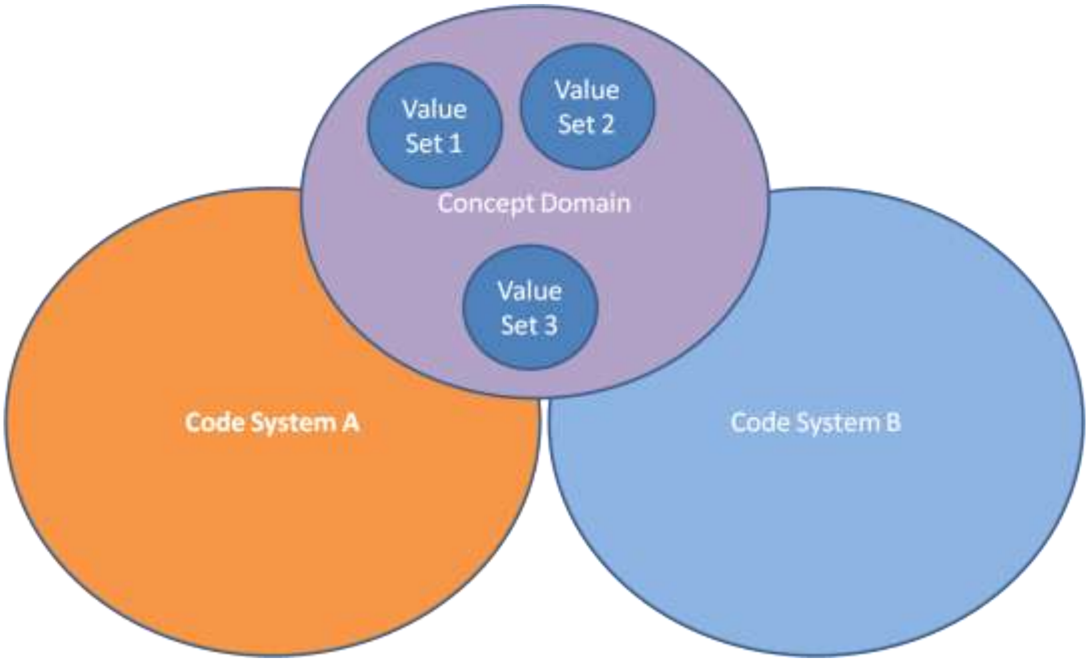
Common examples of localized information models would include constraints on a person entity or organization entity that would satisfy the needs for querying an entity registry. These types of localized information models are common as parameters of service interfaces that perform infrastructure functionality.

3.3 Types - classes, attributes, data types, semantic type –

- Class - A Class is a representation of objects that reflects their structure and behavior within the system. It is a template from which actual running instances are created. A Class can have attributes (data) and in refined models, these attributes are bound to datatypes and may have vocabulary constraints for those attribute types that are coded elements. Classes can inherit characteristics from parent Classes and delegate characteristics to other Classes. Class models describe the logical structure of the system and are the building blocks from which components are built.
- Attributes - Attributes are features of a class that represent the properties of that class. Attributes may be of several different types, defined by the data type to which that attribute is bound. When that data type is of a coded type, the attribute will be bound to a vocabulary element.
- Data type - a data type is a data format specification that describes a specific type or range of values that can be associated with the attribute to which that data type is bound. Data types may be complex or primitive. An example of a complex data type would be the ISO-healthcare data types while a primitive data type example would be the XML data types.
- Semantic type - A Semantic type as defined by a Concept Domain such as orderable labs or pathogenic organisms. A common source of semantic types in healthcare are those defined in the Unified Medical Language System from the National Library of Medicine.

3.4 Vocabulary –

- Concept - A *Concept* is a unitary mental representation of a real or abstract thing – an atomic unit of thought. It should be unique in a given *Code System*. A concept may have synonyms in terms of representation and it may be a single term, or may be constructed of more than one term.
- Code - A *Code* is a concept representation published by the author of a *Code System* as part of the *Code System*, is an entity of that *Code System*, is the preferred unique identifier for that concept in that *Code System*. Codes are sometimes meaningless identifiers, and sometimes they are mnemonics that imply the represented concept to a human reader.
- Code system – A *Code System* is a managed collection of concept identifiers, usually codes, but sometimes more complex sets of rules and references, optionally including additional representations (which may or may not be identifiers of the concepts). *Code Systems* are often described as collections of uniquely identifiable concepts with associated representations, designations, associations, and meanings.
- Concept domain – A *Concept Domain* is a named category of like concepts (a semantic type) that is specified as the vocabulary set allowed for the filler of an attribute in a static model or property in a data type, whose data types are coded or potentially coded. *Concept Domains* exist to constrain the intent of the coded element while deferring the binding of the element to a specific set of codes until later in the model development process when value sets can be constructed based on the specific implementation.
- Value set - A *Value Set* represents a uniquely identifiable set of valid concept identifiers, where any concept identifier in a coded element can be tested to determine whether it is a member of the *Value Set* at a specific point in time. A concept identifier in a *Value Set* may be a single concept code or a post-coordinated expression of a combination of codes.



Vocabulary Relations

3.4.1 Vocabulary Binding

3.4.2 Logical perspective binding

Information models in the Logical perspective are bound to vocabulary by one of three ways.

- The concept domain identified in the Conceptual perspective may be reused without further refinement.
- A sub-domain of the domain information model concept domain may be defined (a refinement) which may be declared a value set.
- A specific code that is fixed for the coded attribute in question.

A concept domain is a named category of like concepts of the same semantic type (all lab result observations, all medications, etc.). A value set represents a uniquely identifiable set of valid concept representations, which may be taken from one or more code systems.

It is only at the Implementable perspective that value set binding can reliably occur since implementation considerations define the terminology required at the interface in almost all cases.

An example to demonstrate the Logical perspective vocabulary binding versus Implementable perspective binding is borrowed from the NCI caCIS project. The value attribute in class that carries the results of a pathology analysis of a specimen may take on a different set of possible values depending on the type of tumor identified in any individual specimen. One could create an exhaustive value set consisting of histopathologic types for all known cancer types. However because at the Logical perspective we would not be able to predict what applications might call the service, the value set would indeed have to be exhaustive to accommodate any user.

An alternative is to create a concept domain at the Logical perspective of “histopathological type’ which can be further refined in the Implementable perspective once an interface binding to a specific cancer type is needed. At that point a value set for BreastCancerHistopathologicType can be created to meet the interface requirement at deployment.

Because we have a terminology concept of Nested Value Sets, i.e. value sets that are sub parts of a larger “base” value set, it is difficult to separate the definition of a “base” value set from a concept domain at all times. Certainly in the example above we could have defined the histopathologic base value set and had as the set of nested value sets, the collection of all cancer type histopathologic type value sets.

3.4.3 Implementable perspective

Vocabulary is constrained in the Implementable perspective by the conversion of vocabulary domains identified in the Logical perspective to specific value sets. An example of this is depicted in the table below.

Logical Vocabulary Domain	Implementable Schema Value Set
HistopathologicType	BreastCancerHistopathologicType
	LungCancerHistopathologicType
	ProstateCancerHistopathologicType

As seen in this example, each cancer type has its own value set because there are many different histopathologic types that are cancer type specific. The individual value sets cannot be determined at the Logical perspective because they are implementation specific. For instance, if an application were deployed in a Gynecologic Oncology practice site that invokes the pathology result service, the cancer types would be restricted to those of the female reproductive system and for a Urologic Oncology practice site, only cancer types of the urogenital tract and prostate would be appropriate.

There is specific metadata for each coded concept that must be supplied in order to understand persisted data over time relative to specific temporal points. In general this can be accomplished through versioning of value sets using a timestamp and by providing identifiers that are permanent for a particular value set. Finally, one must have a value set definition that allows for the resolution of the value set consistently over time from a terminology service.

3.5 Validation forms for information models

3.5.1 Schema

The scheme is a representational information form expressing the metadata about a particular information model in order to do validation of instance data of that particular information model type. The most common form of schema used in healthcare where the payload is communicate via XML is the XML schema definition from the W3C.

3.5.2 Templates

Templates may be used to constrain a particular information model and to provide the necessary rules to consistently interpret the information model. Templates provide a pattern for information models that are intended to be reused.

Templates can take on a number of different usage forms. Examples of templates include archetypes of the openEHR specification and templates applied to the HL7 Clinical Document Architecture structured documents. A more granular form of the template is the detailed clinical model.

3.5.3 Unstructured Information

While the purpose of this Information Framework document is to describe the static semantics of structured information models to allow the participation in computable semantic interoperability, much of the data that exists in healthcare practice is unstructured. It is therefore pertinent to discuss not only the coded elements of information models but also those literals that exist in the form of sentences persisted in the databases and captured in the screens of many healthcare application today.

The word sentence is used here refers to a complex concept that is unencoded and not to the grammatical definition of the sentence.

These sentence based literals are typically expressed as text strings in healthcare information models. There are mechanisms to aid in the understanding of these text strings using natural language processing techniques for tokenizing the sentences which can then be encoded standard terminologies. There are also iso-specifications to provide standard information models for the expression of these sentence literals. The ISO 24707 Common Logic specification provides a grammar to formalize an information model for sentence interpretation and several syntaxes for expressing that information model to allow interoperability between many first-order and partial first order logic languages. The detailed discussion of this specification is out of scope for this document but more information can be found at http://metadata-standards.org/24707/index.html.

4 Behavioral Framework

4.1 Overview: The Purpose of the BF

The purpose of the HL7 Behavioral Framework is to describe the behavioral aspects of systems that participate in Health Information Technology (Health IT). It covers the behavior of (often distributed) system components and the way humans and organizations interact with these components. These behaviors facilitate the creation, exchange, and use of information.

The BF thus provides two sets of grammars – Enterprise and Computational. Each grammar consists of a set of concepts and structuring rules that apply to these concepts. Where concepts are defined, the strict definition is displayed in a normal font, while additional notes or material is displayed in italics.

The concept map below describes the essential elements in the BF and their essential relationships.

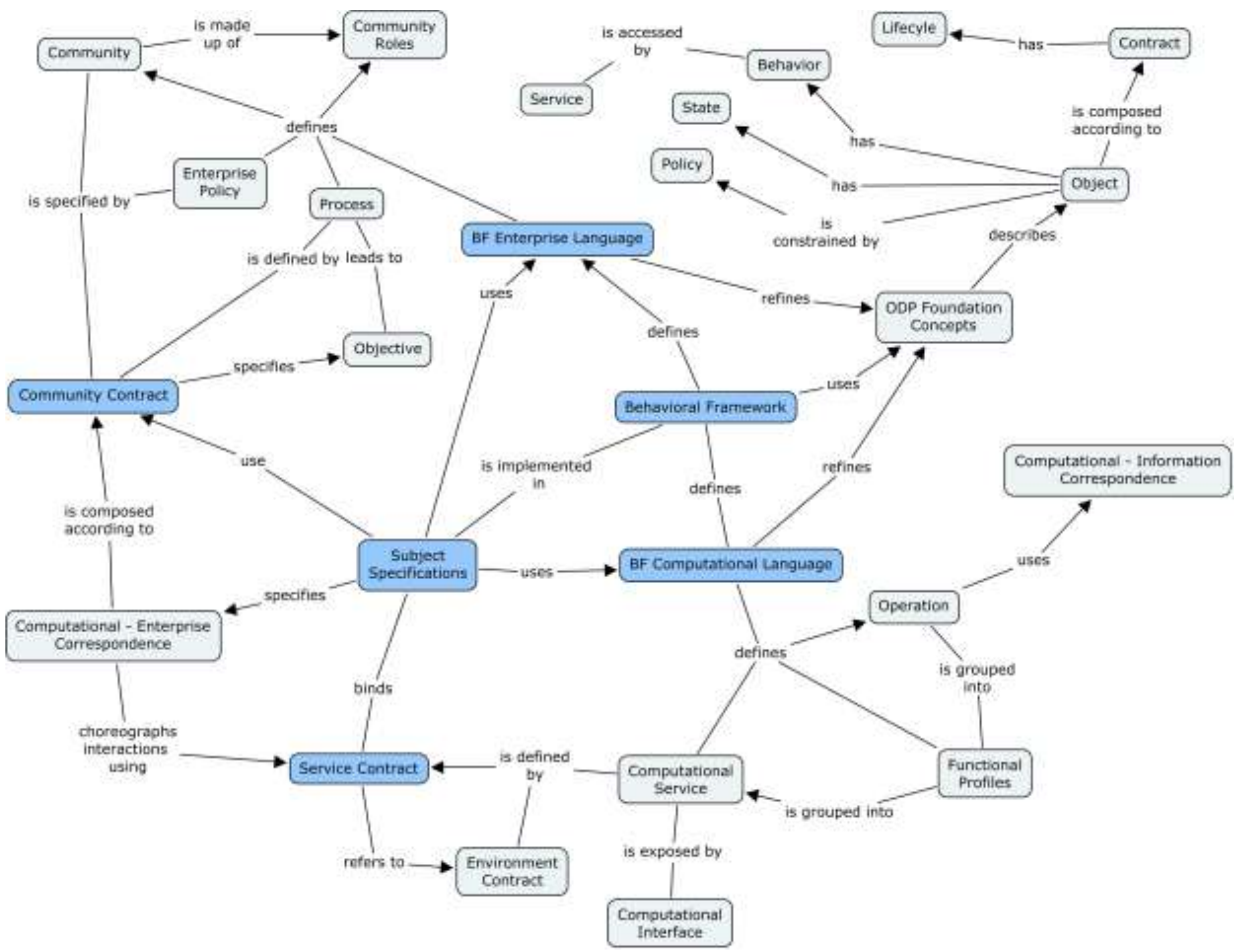


Figure 2: BF Concept Map. Key concepts are in dark blue.

4.2 Key Grammars - Leveraging ISO RM-ODP standards

The Enterprise Grammar focuses on the business context in which Health IT systems are to operate, covering aspects of the collaborative arrangements between parties involved in healthcare while using

Health IT systems. This helps to address concerns of clinical, business, and regulatory stakeholders. Most importantly, from the standpoint of the Behavioral Framework, this grammar sets the stage to couple these stakeholders to the electronic systems that support them.

These concerns are within the scope of the ODP Enterprise Viewpoint. The BF Enterprise Viewpoint uses a relevant subset of the ISO ODP Enterprise Language standard that has been refined to accommodate specific requirements of Health IT.

The ODP Enterprise Language was chosen because of its expressiveness to describe key organizational and policy concepts, in a way close to the human expression of these concepts. It is important to note here that the emphasis is not on supporting the description of social concepts such as acts, roles and entities for the purpose of recording information in the system, as it is in HL7 RIM, but more broadly to describe and interpret these concepts for the purpose of building enterprise systems that are fit for purpose. Nevertheless, the Enterprise Grammar is expressive enough to capture concepts within Health IT such as clinical, administrative, and regulatory practices and policies.

The BF Computational Grammar is based on a subset of the ODP Computational Viewpoint concepts, positioned in the context of certain SOA styles of expression. It is primarily of concern to architects and designers of distributed software and its components. The ODP Computational language was chosen because of its technology independence and precise semantics, allowing support within a solution-focused, conformance-driven framework like the ECCF. The language is also broader in scope than traditional SOA or system architecture approaches, allowing HL7 to support different interaction paradigms and architecture styles such as event-driven architecture, multimedia streams, or for the HL7 interoperability paradigms, i.e., services, messages, or documents.

The Computational Grammar is focused on the technology-neutral description of systems as they interoperate, that is, the services they offer, the way they can be connected to provide more complex capabilities, and the way they align with Health IT policies and practices (from the ODP Enterprise Viewpoint).

In any specification, these two grammars provide the ability to express two separate but related sets of concepts. The BF, thus provides a subset of the grammars to be used within the context of the ECCF, which in turn provides a unified and conformance focused specification for a particular application domain, e.g., referrals, discharge, or care plans. In addition, the concepts from the BF also relate to the concepts from the information framework, and where necessary, to the engineering and technology viewpoints. Within ODP, such cross-grammar alignments are called correspondences.

4.3 Motivation – ODP and Health IT

In general, Health IT systems often span multiple administrative boundaries. Of necessity, they adopt different technology choices, reflecting the specific requirements and build / buy choices of their stakeholders. To respond to this reality, the BF adopted the requirement that there needs to be an approach to facilitate building and standardizing cross-organizational interoperability. This approach reflects generic health standards, policies, and processes, while also accommodating specific organizational and business policies of local healthcare providers.

This is a complex environment and the adoption of a mature international standard, such as RM-ODP (itself developed as a reference model for building interoperable systems), allows the establishment of a common language for delivering working interoperability. This comprises both a common conversation point for people involved in designing and specifying systems, as well as a framework for specifying interoperability contracts between components of the systems involved in exchanging and interpreting healthcare and research information.

The BF is the central place where RM-ODP grammars are adopted within the SAIF. As another approach, the Information Framework adopts the established languages and approaches developed over years in the sphere of health informatics, including the body of knowledge developed in HL7. The Governance Framework in turn is focused on the expression of necessary governance mechanisms needed to serve as an additional assurance so that the processes, policies, and standards for interoperability are implemented and respected – including the adoption and implementation of the concepts and patterns from the BF. This separation into different modeling languages (including correspondence languages) supports a model-driven way of specifying, modeling, and manipulating systems.

The power of the ODP standards is further increased through the use of the recent ISO standard, UML profile for ODP, making it possible to exploit widely used software tools centered around UML - in the specification and implementation of the Health IT systems.

4.4 BF Foundational Concepts

Both the ODP Enterprise and Computational Viewpoints make use of a small set of foundational ODP modeling concepts that are refined for the purpose of providing the Enterprise and Computational Languages. The existence of this set of generic concepts (found below) provides a consistent language across various stakeholders that allows them to express particular additional detail of their concern, while remaining consistent in meaning and representation. For example, the concept of "service," with its refinement and interpretation from the business and system design is given a formal, foundational definition that eliminates much confusion around the use of the word as a "buzzword". This approach is also carried through in the BF.

This section outlines a selected set of the foundation concepts from RM-ODP that are chosen for the purpose of the BF. They are described according with their strict definitions stated in the RM-ODP Foundation standard, along with some explanatory notes (shown in italics). These are a selected subset of ODP foundation concepts that will be further refined in the Enterprise and Behavioral Languages described in the next sections. For a detailed definition and explanation of these concepts refer to RM-ODP, Part 2.

Object – a model of an entity (entity is defined as any concrete or abstract thing of interest). An Object is characterized by Behavior and dually its state. *Note that the concept of object is broader than* *the traditional notion of software objects or business objects used in building object oriented and enterprise system – it is a model of any entity.*

Behavior (of an object) - a collection of Actions with a set of constraints on when they may occur. Constraints may include sequentiality, concurrency, or real-time constraints. An Action is defined as something what happens.

Interaction – a partition of Objects' behavior consisting of a set of actions which takes place with the participation of the environment of the object.

State - at a given instant in time, the condition of an Object that determines the set of all sequences of actions in which the Object can participate.

Interface – an abstraction of the Behavior of an Object that consists of a subset of the Interactions of that Object together with a set of constraints on when they may occur.

Policy - A constraint on a system specification foreseen at design time, but whose detail is determined subsequent to the original design, and capable of being modified from time to time in order to manage the system in changing circumstances. *Policies can be applied in any viewpoint. Examples include an enterprise delegation policy, a computational persistence policy, or an engineering scheduling or quality support policy. In the enterprise viewpoint, Policies may be expressed in terms of obligations, permissions, or prohibitions.*

Service – an Object's Behavior, triggered by an interaction that adds value for the service users by creating, modifying, or consuming information. The effects of invoking the Service become visible in the object's environment. *Note that the provision of a service involves a collaboration between its provider and user. This collaboration may involve a complex series of interactions. The value offered by the invocation of the Service is noted in the corresponding contract. By the same token, Services may invoke additional services/collaborations.*

Contract - An agreement governing part of the collective Behavior of a set of Objects. A Contract specifies Obligations, Permissions, and Prohibitions for the Objects involved. The specification of a contract may include:

- specification of the different roles that objects involved in the contract may assume, and the interfaces associated with the roles;
- Quality of Service constraints
- indications of duration or periods of validity
- indications of behavior which invalidates the contract
- liveness and safety conditions.

Contracts come in three varieties in the BF, reflecting the refinements of the generic, system-theoretic definition above, namely :

- *Community Contracts – specified using the Enterprise language*
- *Service Contracts – specified using the Computational language*
- *Environment Contracts – specifies Quality of Service Constraints for an Endpoint*

Contract life cycle is described using the following concepts define next.

Establishing Behavior - Behavior by which a given contract is put in place, for example, through negotiation between parties to the contract, resulting in a contract, or a publication of a contract offer, of one object to its environment.

Enabled Behavior – Behavior characterizing a set of objects which becomes possible as a result of Establishing Behavior.

Terminating Behavior - Behavior that breaks down the liaison and repudiates the corresponding contractual context and the underlying contract. The figure below depicts different stages in the contract. This is followed by a figure illustrating the description of core contract modeling concepts.

Information - Any kind of knowledge that is exchangeable amongst users , about things, facts, concepts and so on, in a universe of discourse. Although information will necessarily have some forms of representation to make it communicable, it is the interpretation of this representation (the meaning) that is relevant in the first place.

The diagram below shows key BF foundation concepts mentioned in this section. Note that the diagram does not show all relationship between concepts, e.g., further elaboration on state and its link to behavior. This level of detail is beyond the scope of this book.

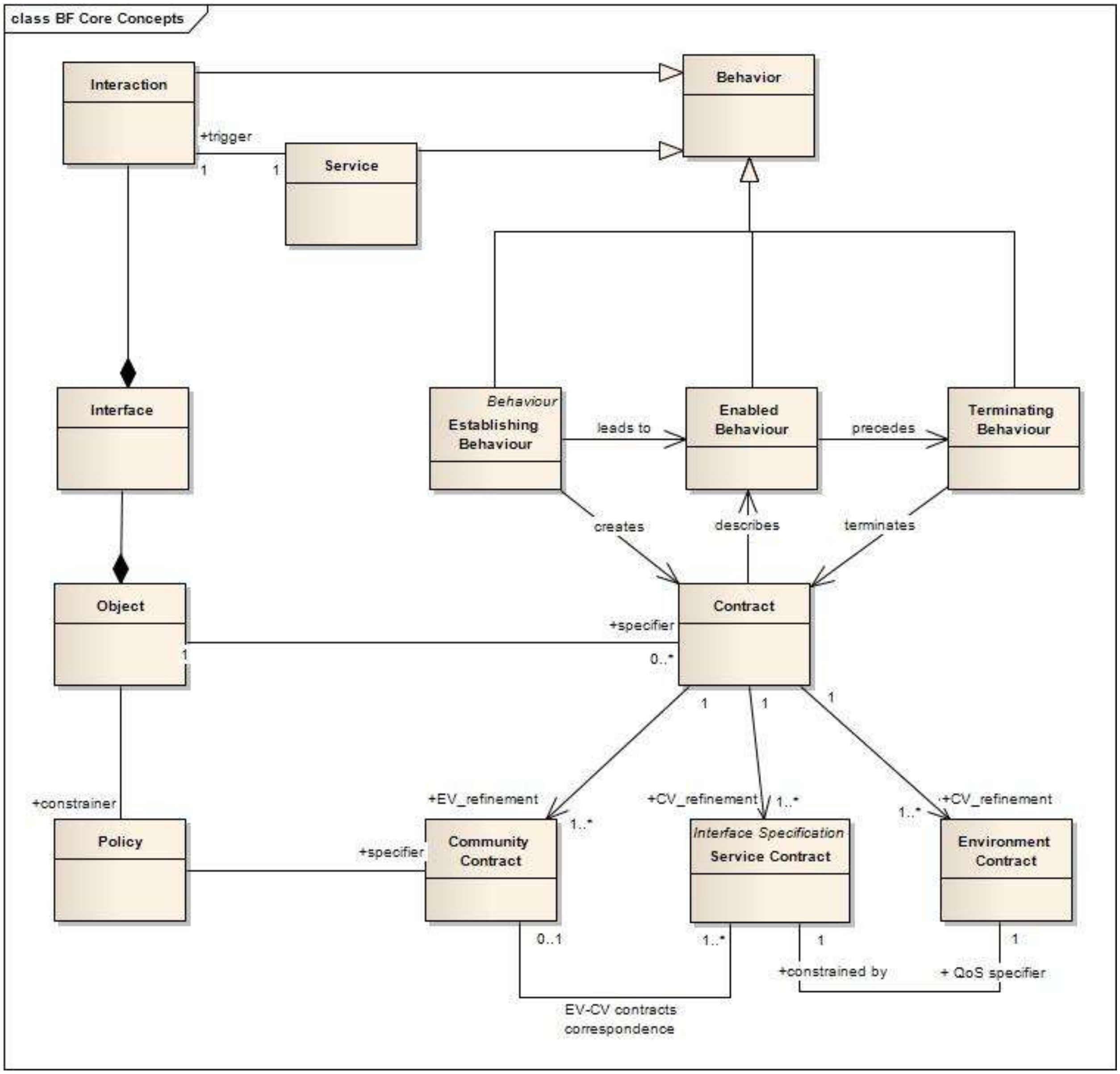


Figure 3: Foundation RM-ODP Concepts that appear throughout the BF

4.5 BF Enterprise Language

The BF Enterprise language concepts are a subset of ODP Enterprise viewpoint concepts that capture ECCF specification elements in the Conceptual Perspective. It expresses the considerations necessary to understand the rules and policies that govern a collection of Enterprise Objects.

Conceptual Perspective

The key Enterprise Language concepts from the Conceptual Perspective are:

Community – A configuration of enterprise objects formed to meet an objective. The **objective** is expressed in a contract, which expresses how this objective can be met by the roles in the community, the interactions required between them, assignment of roles to systems, and the policies governing the collective behavior. *The concept of community is suitable to describe business contexts as it clearly delineates business policies that apply to the roles in the community and their interactions in business processes. The concept of community is suitable to describe business contexts as it clearly delineates business policies that apply to the roles in the community and their interactions in business processes. More information can be found in ODP Part 3. Communities are typically expressed in HL7 as storyboards.*

Enterprise Object - represents a refined view of the generic concept of Object where the focus is on enterprise view of objects. The enterprise objects have life independent from that of a community such that they can model IT systems, people, and organizations.

Community Role – a placeholder for behavior in community that can be filled by an enterprise object that satisfies type specified in the community role. *Community Roles are typically expressed in HL7 using Use Cases, or more broadly as functionality expressed in an EHR Functional Profile.*

Community Contract – specifies community behavior in terms of processes and interactions involving community roles as well as policies that apply to the community roles. *For example, a Community contract can define Obligations of healthcare providers, e-health service organizations, and rights of patients, as well as conditions about efficiency, security, response times and confidentiality to be met when delivering e-health and healthcare services.*

Process - a collection of steps taking place in a prescribed manner and leading to an objective. Step is defined as an abstraction of an action, used in a process, that may leave unspecified objects that participate in that action [ODP-EL]. A process does not have to explicitly nominate the roles involved.

Enterprise Policy – a rule that specifies constraints in the enterprise specification, in particular regarding one the roles in the community. Typical enterprise policies are obligations, permissions and prohibitions.

Enterprise Service – a special kind of Behavior that involves Commissioning and Responsible Roles, to which Enterprise Policies apply. The effects of invoking the Enterprise Service become visible in the Community, and serve to work towards the Community's Objective. They are the realization of the Accountability Pattern, in that they provide the Commissioning and Responsible Actions required in a Community that is partitioned between Roles. *Enterprise Policy is typically expressed in HL7 as Receiver Responsibilities, or more broadly, in terms of policies expressed in an EHR Functional Profile.*

Obligation: A prescription that a particular behavior is required. An obligation is fulfilled by the occurrence of the prescribed behavior.

Permission: A prescription that a particular behavior is allowed to occur. A permission is equivalent to there being no obligation for the behavior not to occur.

Prohibition: A prescription that a particular behavior must not occur. A prohibition is equivalent to there being an obligation for the behavior not to occur.

Party - An enterprise object modeling a natural person or any other entity considered to have some of the rights, powers, and duties of a natural person. *Examples of parties include enterprise objects representing natural persons, legal entities, governments and their parts, and other associations or groups of natural persons. Parties are responsible for their actions and the actions of their agents.*

The diagram below depicts key elements of the BF Enterprise language. Note that the concepts presented fall in the Enterprise/Conceptual cell of the ECCF matrix, though their appearance in specifications may be expressed in models expressing ODP Viewpoint correspondences.

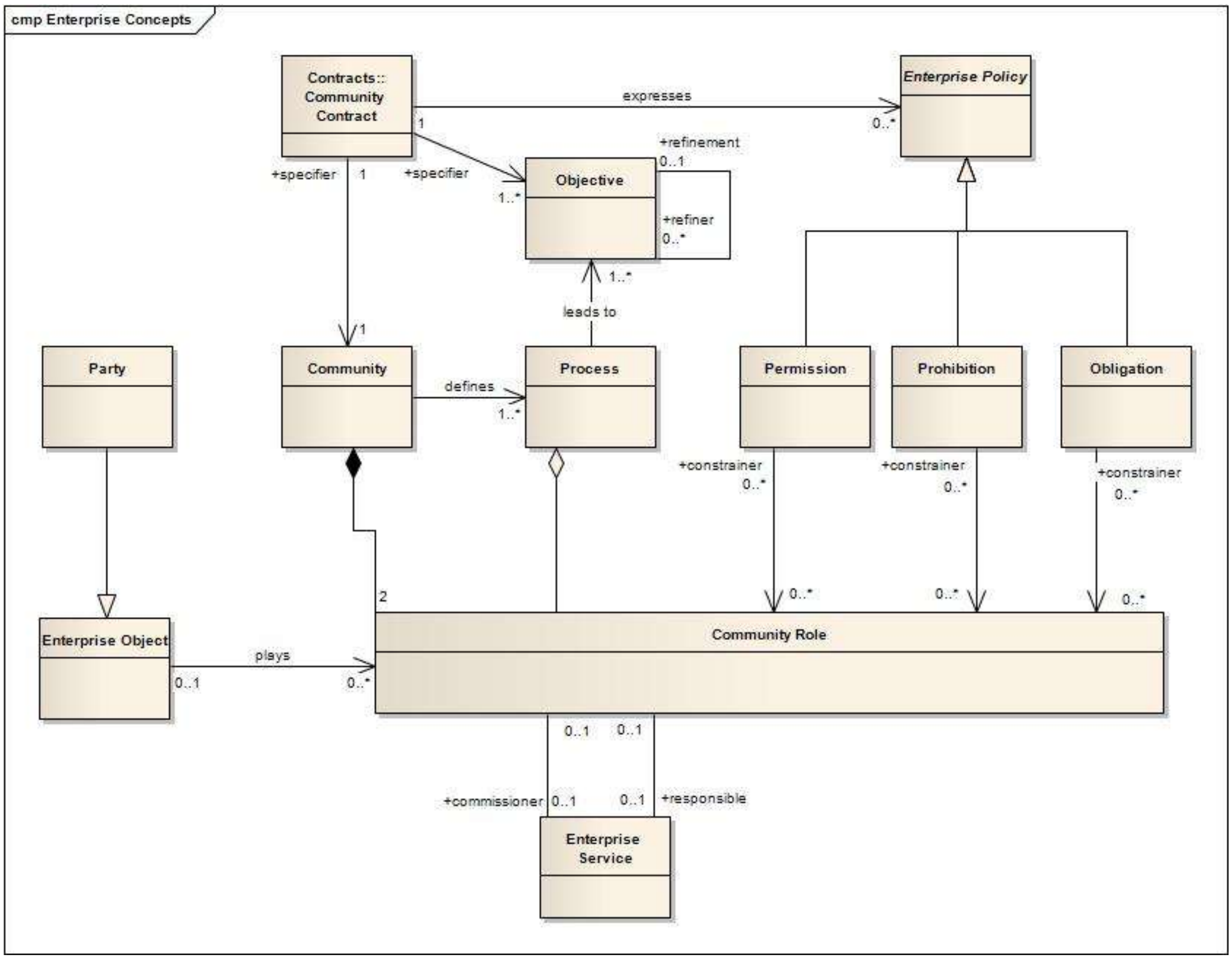


Figure 4: Core concepts in the BF Enterprise Language

4.6 Logical and Implementable Perspectives

There are no Enterprise Language concepts at the Logical or Implementable Perspectives, though the considerations from the Enterprise Viewpoint guides the refinement for a specification within these languages.

4.7 BF Computational language

The BF Computational language concepts are a subset of ODP Computational viewpoint concepts. The BF Computational Grammar is mostly concentrated in the Logical perspective in the ECCF matrix, although several concepts in the Conceptual Perspective are identified to facilitate linkages with the BF Enterprise Language. This provides the ability to define a business context within which a Health IT system may be defined, designed, and built.

The Computational Language at the Conceptual Perspective is designed to align with other artifacts emerging within the HL7 community such as Conceptual Information Models, Domain Information Models, Conceptual State Models, or the EHR Functional Model (and profiles).

4.7.1 Conceptual Perspective

The Computational language concepts from the Conceptual Perspective are:

Computational Service – A service refinement from the ODP Computational Viewpoint, that is, behavior offered by a computational interface, constituting a service contract. In HL7, the Application Role takes on some aspects of the Computational Service.

Functional Profile – an abstraction of behavior that aligns with sets of Role Behaviors. This concept is equivalent to an Interface. In HL7, the Application Role takes on some aspects of the Functional Profile.

Operation – a computational representation of a Role's invokable behavior. In HL7, the Application Role takes on some aspects of the Operation from the Conceptual Perspective.

Conceptual Information Concept – a placeholder for the Conceptual Perspective's information objects.

Exception Condition – exists when an Operation fails to fulfill its Obligation

Pre-Condition – a predicate that a specification requires to be true for an action to occur.

Post-Condition – a predicate that a specification requires to be true immediately after the occurrence of an action.

The diagram below details the Computational Language at the Conceptual Perspective. Note these some of these concepts serve as a form of correspondence between Enterprise-Computational and Computational-Information viewpoints. For example, Functional Profile is an explicit abstraction of behavior that is designed to align with elements from the Enterprise Viewpoint.

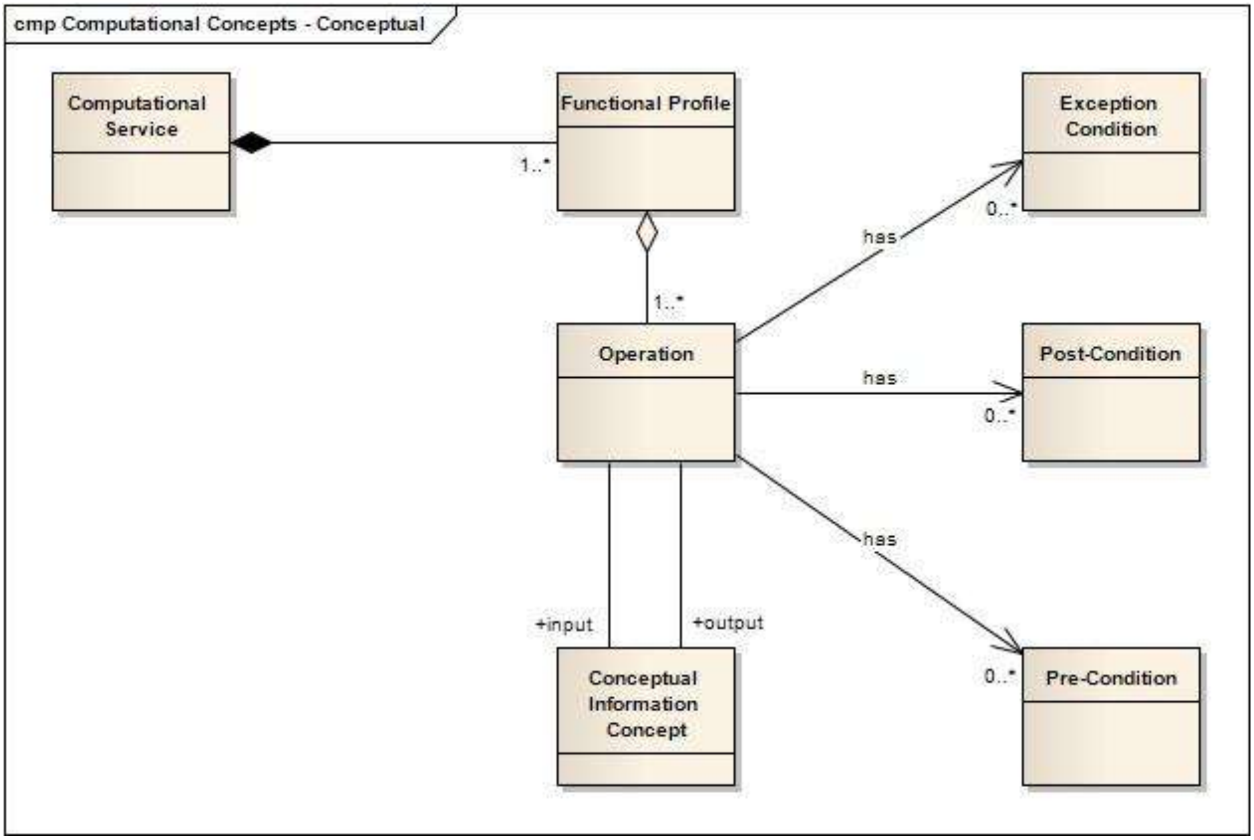


Figure 5: Computational Language from the Conceptual Perspective

4.7.2 Logical Perspective

The Logical Perspective of the BF Computational Grammar capture key concepts needed for the design of components in the system that implement functionality required, such as those identified in the functional profile. It is developed to align with other artifacts emerging within the HL7 community such as Serializable Information Models, Logical State Machines, Abstract Data Types, and Stub Models (CMETs).

The concepts in the Computational Language from the Logical Perspective are:

Computational Object - view of an Object from Computational Viewpoint, giving particular focus on describing units of logical functionality and distribution in the system.

Computational Interface – view of an Interface from Computational Viewpoint, giving particular focus on capturing an externally visible behavior of a computational object. A Computational Object can offer multiple Computational Interfaces. *In HL7, the Application Role takes on some aspects of the Computational Interface.*

Computational Interaction – a view of an interaction from computational viewpoint, focusing on how a system can interact with its environment. Interactions can be of three types: Operations, Streams, or Flows. *Note that the syntactic aspects of interactions are expressed in the signatures of the operations,* *streams, or signals that the interaction supports. All three interaction types are included in this model to support future standards. In HL7, the Interaction takes on some aspects of the Computational Interaction.*

Operation - an Interaction between a client object and a server object that is either an interrogation or an announcement. *One example of interrogation is an RPC calls over the SOAP protocol. One example of an announcement is sending of a message in a messaging system, such as "HL7 Send [Message Payload]—No Acknowledgements". In HL7, the Message End Point or the Application Role takes on some aspects of the Operation.*

Flow - An abstraction of a sequence of Interactions, resulting in conveyance of information from a producer object to a consumer object. *This kind of interaction is typical to sending video and multi-media information or can be used to model continuous flow of periodic sensor readings from certain sensors as in many clinical devices.*

Signal - An atomic shared action resulting in one-way communication from an initiating object to a responding object. *An example of a signal is the initiation of an event notification (for example, ADT) by the sending interface or the receipt of the event notification by the receiving interface.*

Interface Specification – specifies the Interface of the Computational Object, its Behavior, an Environment Contract, and the Interface Signature expressing syntax of the Interactions. *In HL7, the Application Role takes on some aspects of the Interface Specification.*

Interface Signature - The set of Action signatures associated with the interactions of an interface. *These signatures are the syntax for operations, including the representation of the operation, the parameters, and the message exchange pattern in use.*

Environment Contract - A contract between an object and its environment, including Quality of Service constraints, usage and management constraints. *Quality-of-Service constraints include temporal constraints (e.g. deadlines); volume constraints (e.g. throughput); dependability constraints covering aspects of availability, reliability, maintainability, security and safety (e.g. mean time between failures). Usage and management constraints include: – location constraints (i.e. selected locations in space and time); – distribution transparency constraints (i.e. selected distribution transparencies).*

Service Contract – a special kind of Interface Specification modeling externally visible Behavior of Object. It defines obligations of an object to other objects in terms of computational interactions, as stated in the object's interface(s). The computational service contract can be accompanied by the Environment Contract, which states non-functional properties of the computational object. *Service Contract allows the refinement of the Contract concept using Computational Concepts, for example, defining Computational Interfaces. This in turn allows the expression of correspondences of Computational Viewpoint concepts to their Enterprise Viewpoint counterparts, allowing Computational Objects to fulfill Community-defined Roles. For example, this can be used to model traditional notions of SOA Services, that is, as Service Providers realizing the Community Role of Responsible Parties for* *particular types of information. It can also be used to disambiguate traditional HL7 Application Roles like Lab Placer.*

Binding Object – a special kind of Computational Object that encapsulates the functionality required to connect two or more other Computational Objects. Note that the object itself provides a control interface to allow these connecting mechanisms to be configured and managed, which is of interest when one needs to support protocol translation. *The Binding Object is what the BF refers to as a Subject Specification.*

The diagram below depicts the Computational Language concepts from the Logical Perspective.

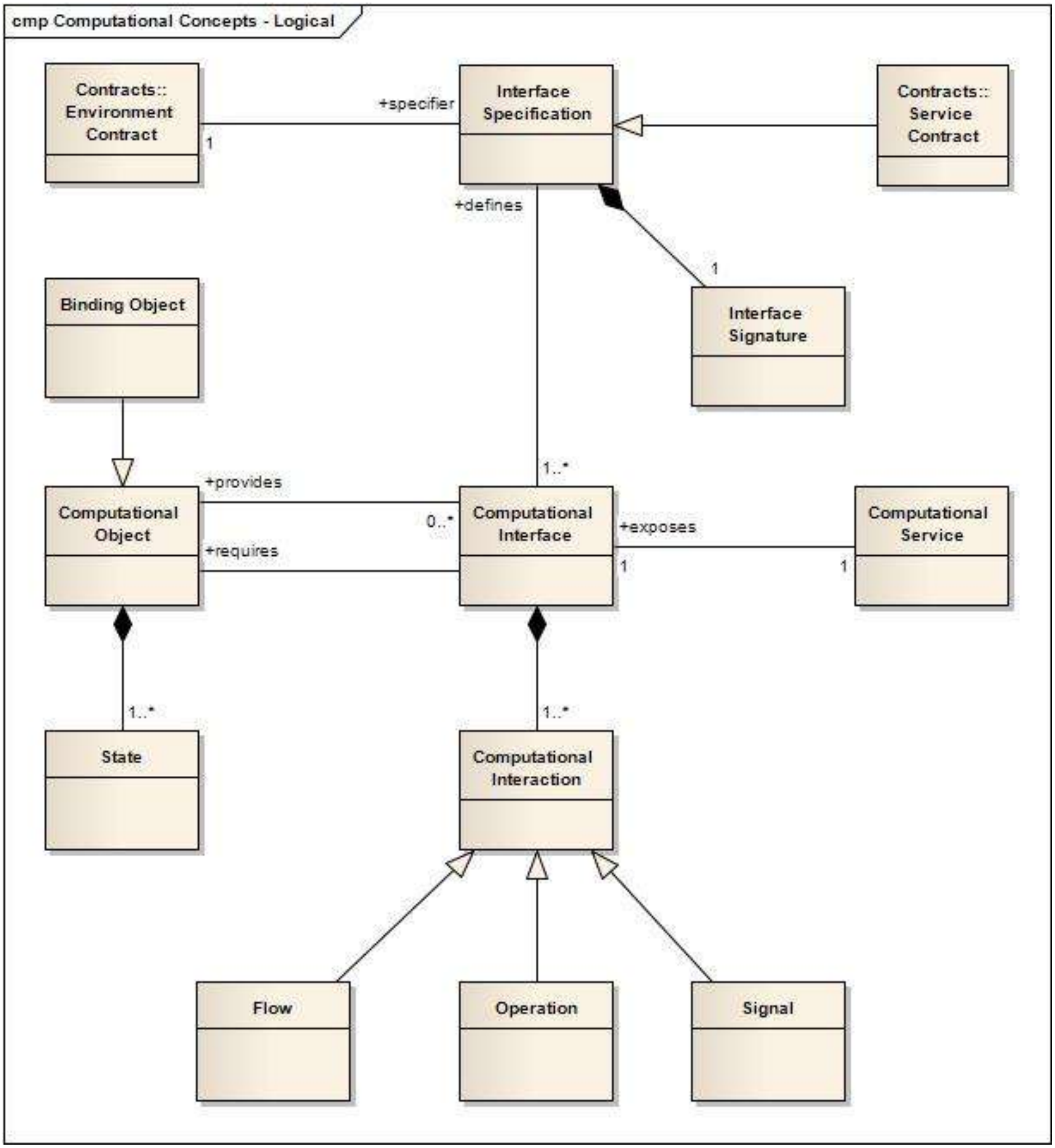


Figure 6: Computational Language from the Logical Perspective

Computational objects are typically identified and defined after the Enterprise and Information specifications have been developed. It is also possible to adopt a bottom-up approach, for example, when making use of the existing library of specifications or Application Roles. In this case, the behavior of existing components can be described using Computational Objects, so they can be used to realize

Behavior of Roles in a community that defines the business purpose for the use of these objects. Roles and Communities do not need to be pre-established for this to be true, although they may be as a matter of governance. Enterprise objects which model IT systems play roles within communities as determined by the interface types exposed by computational objects realizing the enterprise objects.

4.7.3 Implementable Perspective

The BF's Computational Language does not have an Implementable Perspective, though the considerations herein serve to refine specifications that include the Engineering Viewpoint. See End Point and Solution Specifications below.

Contract Lifecycles

As is pointed out in the models above, Contracts represent one of the central concepts in the BF. In fact, they represent the key placeholder for providing continuity and traceability from the design of an HL7 standard to its eventual implementation.

The life cycle of a contract is defined in terms of its establishing behavior, the period in which contract exists, and its terminating behavior.

The *establishing behavior* is defined as the behavior by which a given contract is put in place, for example, through negotiation between parties to the contract, resulting in a contract, or a publication of a contract offer, of one object to its environment.

Once a contract is established, its existence is signified through the *contractual context*, which is the knowledge that a particular contract is in place. A contract being in place signifies that a particular behavior of the set of objects to which the contract applies is required.

A *binding behavior* is a specific type of establishing behavior involving two or more interfaces, and thus their owning objects. *Binding* is a special kind of contractual context, resulting from a given establishing behavior. The purpose of the binding function is to bind together interfaces (signal, operational, and stream) to enable communication between objects.

The establishing behavior also yields a certain relationship between the objects in a set, referred to as *liaison*, effectively stating the fact that objects have a contractual context in common. Examples of liaisons are a distributed transaction, relationship between files and processes which access files, as well as provider and user of a service as defined in a Service Contract.

The *terminating behavior* is the behavior that breaks down the liaison and repudiates the corresponding contractual context and the underlying contract. The figure below depicts different stages in the contract. This is followed by a figure illustrating the description of core contract modeling concepts.

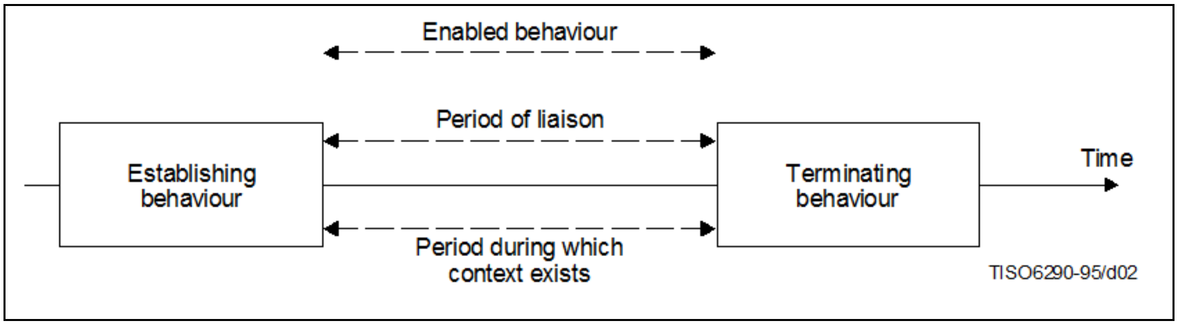


Figure 7: Contract Life Cycle

Contracts can be established in different epochs of the software lifecycle, including, for example, at specification time, governance time, design time, or run time.

4.8 Implementing the BF –Specifications and Correspondences

The BF is implemented by using the metamodels to build specifications that can then be standardized via a governance process, like balloting. Particularly, Subject Specifications are supported using the Enterprise and Computational Languages to define working interoperability. There are two refinements of Subject Specifications: End Point and Solution. They are intended to support an architecture that may mix HL7’s Interoperability paradigms of documents, messages, and services.

Each specification type represents a particular correspondence of the various contracts established in the BF. Below are a number of concepts relevant to Subject Specifications.

End Point Specification – explicit correspondence between Objects allowing Computational Objects to fulfill Community-defined Roles. For example, the End Point models the traditional notion of an SOA Service, that is, a Service Providers realizing the Community Role of Responsible Parties for particular types of information. End Point Specifications utilize a Primitive Binding.

Primitive Binding - signifies a contractual context that allows the objects to connect and to exchange services and information. It is a binding between two Objects.

Primitive Binding Correspondence - for each interaction between roles described by the Enterprise Language, one may identify a set of Computational Binding Object types that are constrained by the enterprise interaction. In this case, two Community Roles are defined: Commissioner and Responsible Party. The Responsibilities are defined in terms of the Specification, and so each Commissioner becomes interchangeable.

Primitive Binding’s Contract Correspondence - for each Enterprise Service specified in a Community Contract there might be one or more Computational Services that realize this Business Service. The Computational Services are specified through Service Contracts offered by Computational Objects.

Solution Specification – explicit correspondence between Objects allowing Computational Objects to fulfill Community-defined Roles. Solution Specifications support multiple party interactions where definition is need for two (2) or more parties involved in the interaction. Solution Specifications specify a Compound Binding.

Compound Binding - Compound binding is a special kind of Computational Object that can define and control Interactions such as message exchange protocol (or choreography), as well as service invocation sequence between multiple interfaces of different objects. Compound Bindings can be composed of Primitive Bindings. *In cases where multi-party interactions must be expressed, the concept of Compound Binding can be used, making it the cornerstone of Solution Specifications.*

Compound Binding Correspondence - for each interaction between roles described by the Enterprise Language, one may identify a set of Computational Binding Object types that are constrained by the enterprise interaction. In this case, multiple Community Roles may be defined. Their interactions, including sequentiality, concurrency, or real-time constraints, are defined as conformance points in the specification.

Compound Binding's Contract Correspondence - for each Enterprise Service specified in a Community Contract there might be one or more Computational Services that realize this Business Service. The Computational Services are specified through Service Contracts offered by Computational Objects. This correspondence is realized in the Solution Specification for 2..* Community Roles.

The model below shows the components of the Subject Specification as defined in the BF. Note that there are Informational and Engineering Correspondences that are out of the scope of the BF.

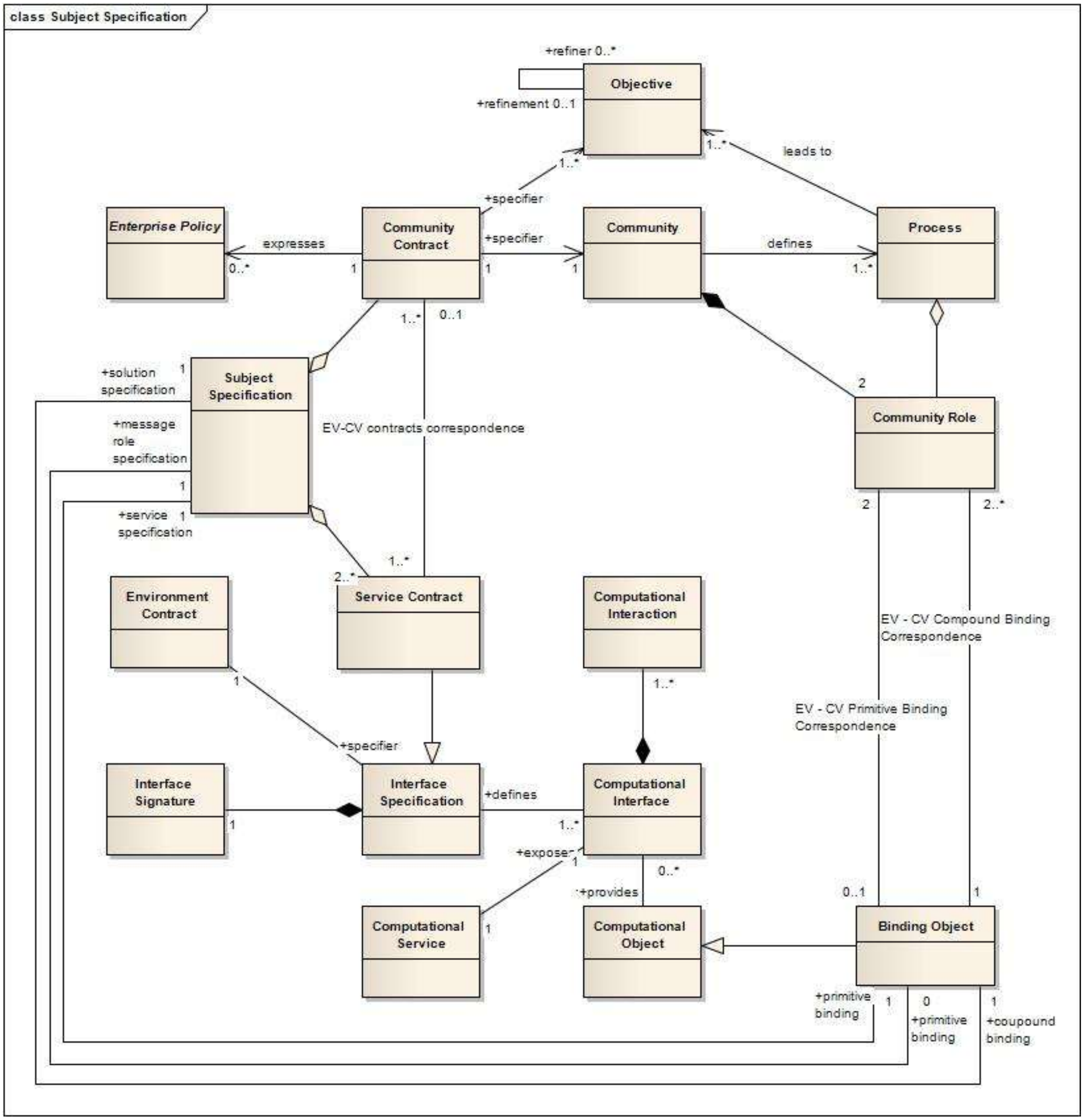


Figure 8: The BF Subject Specification with its relevant correspondences

Messaging represents a design paradigm that partitions responsibility between all parties in a Community, requiring Primitive Bindings to be established during specification to support conformance and to achieve the Community's Objectives. Message End points specified using the BF are therefore incomplete from a conformance standpoint, but may be reused in other specifications by later identifying their counterpart in a Primitive Binding.

SOA Services represent a different paradigm that defines the Responsible Party only, and thus serve to realize the Enterprise Policies around that responsibility. All Commissioning Roles (Enterprise Language) are thus the same, allowing for a Primitive Binding to be established during the design or implementation without undue constraints on the its implementation or deployment. This provides flexibility and supports reuse not only in specification, but in implementation and deployment as well.

Documents represent an Information Object, and are therefore supported in either design paradigm.

End Point Specifications are useful for defining HL7's reusable interfaces, whether they are individual messaging roles or SOA Services. Solution Specifications are used when multiple parties in a Community must be specified together. Documents may be bound to either via the Computational – Information Correspondence (see below).

4.9 Primitive Binding Illustration

In Unified Modeling Language (UML), the Primitive Binding can be illustrated as in the example in the figure below. In the figure, a Responsible agent is expressed as the Provider component and a Commissioning agent is expressed as the Consumer component. The following UML concepts can be used to represent the RM-ODP concepts.

UML components represent ODP computational objects, for example, Consumer and Provider. These components realize behavior of the commissioning and responsible roles.

UML ports represent ODP interface types of the computational objects, for example, Client and Server. These are operations interface types. ODP also supports stream and flow interface types, but these are not discussed in this Implementation Guide.

UML provided and required interfaces represent roles in interaction, that is, a service provider and a service consumer. There is a correspondence between these interfaces and the behavior of responsible and commissioning roles in the community. These interfaces realize behavior specified in community roles. Note the distinction between roles in interaction (as described here) and roles in community specified in the Enterprise Viewpoint (that is, Accountability pattern in the BF specification).

UML interface specifies the signature of the operations that comprise the service.

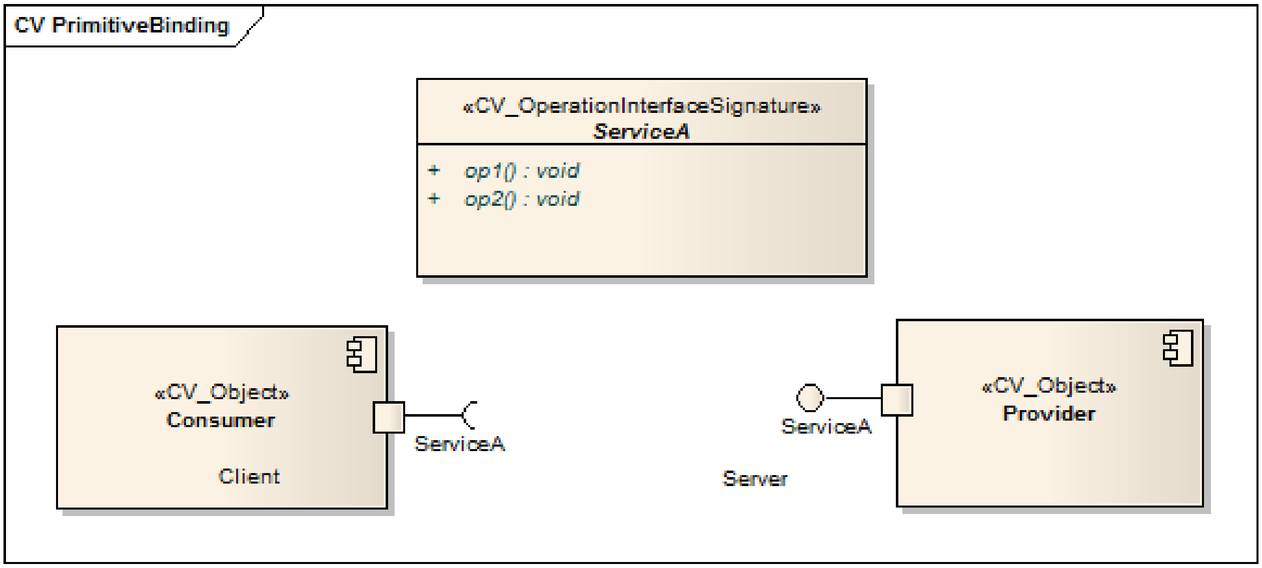


Figure 9: UML Components representing Computational Service

4.9.1 Correspondences

Contracts are expressed in the Enterprise and Computational Languages. While ODP standards do not have an explicit refinement of the basic contract concept in the Engineering or Information Languages, their corresponding concepts appear in specifications or can be derived, as required. This is possible because the above models establish certain correspondences between ODP viewpoints that represent useful compositions of concepts. Specifications, when including the other Viewpoints, are correspondences between Viewpoints. Below, a number of expected correspondences are detailed.

Some examples of correspondence between Enterprise and Information Viewpoints are:

For each Role in each Community in the Enterprise Specification, there may be a list of those Information Object types (if any) that specify information or information processing of an Enterprise Object fulfilling that Community Role.

For each Action in the Enterprise Specification, there may be Information Objects subject to a Dynamic Schema constraining that Action;

For each relationship between enterprise Roles, there may be an Invariant Schema that constrains objects fulfilling Roles in that relationship.

Some examples of correspondence between Enterprise and Computational Viewpoints are:

For each Enterprise Service specified in a Community Contract there may be one or more Computational Services that realize this Business Service. The Computational Services are specified through Service Contracts offered by Computational Objects.

For each Interaction between Roles in the Enterprise Specification, one may identify a set of Computational Binding Object types that are constrained by the Enterprise Behaviors (Interactions).

Solution Specifications, introduced below, include correspondence between Computational and Engineering Viewpoints. Some examples of are:

For each Binding Object that represents a *complex binding* (more than two Computational Objects), the Solution Specification can include UML Communication Diagrams of Sequence Diagrams to convey patterns in communication. This is conveyed using Nodes and Channels from the Engineering Viewpoint.

HL7 ITS represents an Interceptor Object from the ODP Engineering Language, and is included in the description of the Solution.

4.10 References

[ODP-Foundations] ISO/IEC IS 10746-2, Information Technology — Open Distributed Processing — Reference Model: Foundations, 2010. Also published as ITU-T Recommendation X.902.

[ODP-Architecture] ISO/IEC IS 10746-3, Information Technology — Open Distributed Processing — Reference Model: Architecture, 2010. Also published as ITU-T Recommendation X.903.

[ODP-EL] ISO/IEC IS 15414, Information Technology — Open Distributed Processing — Enterprise Language, 2003. Also published as ITU-T Recommendation X.911.

[ODP-UML4ODP] ISO/IEC IS 19793, Information Technology — Open Distributed Processing — Use of UML for ODP System Specifications, 2009. Also published as ITU-T Recommendation X.906.

5 Governance Framework

1. This Chapter describes the motivation for, the structure, content and utilization of the Governance Framework (GF).
2. The GF is the “rules of the game.”

- ***Who gets to make the decisions***
- ***how/when are they made, and***
- ***how/when are they enforced.***

3. ***You gotta have it***
4. ***You gotta document it***