--.

Health Level Seven, Inc.

S2 –The SAIF Behavior Framework

Table of Contents

1 Behavioral Framework Overview

The Behavioral Framework (BF) provides a set of constructs for defining the behavioral semantics of specifications, which enable Working Interoperability. As a result, the focus of the BF is *accountability* – a description of "who does what when." Accountability describes the perspective of the various technology components that are involved in a particular instance or scenario designed to achieve Working Interoperability. The BF is technology-neutral and, therefore, can be used within model-driven specification stacks, such as the Services-Aware Interoperability Framework (SAIF) Enterprise Conformance and Compliance Framework (ECCF).

This discussion assumes that the BF is one of the sub-frameworks of SAIF. As explained in the SAIF Introduction, each sub-framework is a grammar/set of meta-models, which enables one to describe particular aspects of a specification that is associated with the specified component's involvement in an instance of Working Interoperability.

In particular, the BF specifies the grammar that is used to construct the essential artifacts necessary to comprehensively specify the various aspects of the Computational (and, to a lesser degree, the Information) viewpoints of the ECCF's specification stack instances for a given organization's implementation of SAIF.

Following the in this section, Section 2 provides a detailed description of the fundamental concepts and constructs of the BF. Following that, Section 3 presents the various models that collectively define the BF at each of the three levels of the ECCF:

- The Computationally Independent Model (CIM)
- The Platform-Independent Model (PIM)
- The Platform-Specific Model (PSM)

Sections 4 and 5 discuss overall usage guidelines and BF patterns that are essentially implementation-neutral. Following are Appendices presenting the mapping of the BF to the HL7 Legacy Dynamic Model and a more detailed discussion of the differences between Reference Information Model (RIM)-based

HL7 Version 3 messaging and services as examples of two different interoperability paradigms. In reading the discussion of the BF, note that the BF is informed and to a large (but not exclusive) extent scoped by the Reference Model for Open Distributed Processing (RM-ODP) Computational viewpoint and draws on the terminology of RM-ODP whenever possible.

The BF is used to describe both *the functional decomposition of systems* and *the means by which they interact with their environment* and with other systems. The BF also provides the associated static semantics, which are bound to various specified behaviors. Thus, the BF focuses on the specifics of the actual run-time behavior of software running at a computational node in a deployed architecture, e.g. a software component's interface. More specifically, a "focus on behavior of a node" means the quantitative, unambiguous specification and documentation of the details of "conversations or interactions between nodes, which collectively create business value." Examples of such "conversations or interactions" include everything from simple *push* messaging, to publish-subscribe distributions, and to longer-running, multi-party transactions.

The BF combines notions of a *loosely coupled event-driven architecture* – and is thus compatible with a traditional message-based environment, such as HL7 V2 or V3 – with *inter-component procedural activities* to achieve three overarching goals and capabilities:

- Documentation of human-mediated interoperability patterns, such as those present in healthcare IT solutions.
- Documentation and encapsulation of automated interoperability patterns.
- Documentation of the definitional characteristics of the technological structures (for example, components and interfaces) that assume roles within a deployed architecture *in a manner that enables the definition and validation of accountability at a per-component granularity.*

Because the BF is intended to be used in the context of the ECCF, it facilitates the development of testable and certifiable *conformance statements,* which denote *conformance points* at which a given implementation can make pairs of *conformance assertions.*

Note: The ECCF document defines and discusses the concepts of conformance statement, conformance assertion, and certification.

The BF should not be confused with a given architecture specification's formalisms, which are used to express conformance statements within a given ECCF specification stack instance, but rather should be seen as a grammar for expressing these statements.

This document is primarily concerned with defining the syntax and semantics of the BF rather than providing an explanation of how it is applied. Each organization adopting SAIF will develop the specifics of using the various SAIF grammars in an organization-specific *SAIF Implementation Guide*. As such, this discussion provides relatively few concrete examples of BF applications, specific artifacts, and so on. However, when it is helpful in defining a specific syntactic or semantic point in the definition of the BF, a brief example is included.

Note: By necessity, certain formalisms are required to express the BF's core concepts. When a particular formalism constitutes a normative choice, the text will note that choice. Similarly, certain components, concepts, and constructs of the framework persist through specifications irrespective of their content or context, and the discussion will note those situations.

The formal models associated with the HL7 Behavioral Framework, which are included in Section 3 of this document, are published at:
http://www.ncientarch.info/hl7_bf/hl7_bf/

1.1 Goals

In the larger context of SAIF, this document has the following goals:

- Define all relevant concepts and relationships that collectively define the BF.
- Include a concept-by-concept mapping between HL7's Legacy Dynamic Model and the BF, including a mapping between V3 message interaction examples and the BF.
- Demonstrate how to use the BF in definition component interactions in both service- and message-based component specifications.
- Discuss the impact of the BF on interoperability paradigm decisions.
- Discuss the impact of the BF on both conformance and governance when applied at either the intra- or inter-enterprise level. (Other aspects of this

discussion can be found in the ECCF and Governance Framework sections of the SAIF document, respectively.)

As mentioned above, detailed examples and stylistic issues concerning the use of the BF are out-of-scope for this document. You can find these examples at the model publication site (http://www.ncientarch.info/hl7_bf/hl7_bf/) and in the context of specific organizational *SAIF Implementation Guides.*

1.2 Audience and Prerequisites

The audience for this discussion includes architects (both system- and enterprise-level), standards developers, tool developers, and system designers. In particular, anyone who is interested in using the BF to create specification-specific conformance statements, to understand the relationships of the computational expression of those statements, and to dive more deeply into conformance statements, will benefit from this document. The BF will also be of interest to developers and engineers from the perspective of how the BF enables traceability from specification to implementation (for example, through contract templates) and consequent durability of specifications, i.e. the ability to build specifications and deploy implementations based on these specifications that are resilient to changes in business context.

Prerequisites for this document include at least some familiarity with the following topics:

- SAIF Enterprise Conformance and Compliance Framework (ECCF)
- SAIF Governance Framework (GF)
- The four pillars of Computable Semantic Interoperability (CSI)[1]
- HL7 Core Principles, Reference Information Model, Data Type Specification, and Vocabulary Best Practices (**Note:** These specifications will be combined into a single SAIF Information Framework document.)
- System design, Enterprise Architecture, development, and experience with Unified Modeling Language (UML)
- Familiarity with core principles and applications of Service-Oriented Architecture (SOA)

[1] Journal of Health Information Management, published by HIMSS, January 2005.

- Component Based Development and Integration Service Architecture and Engineering$_{TM}$ meta-model for SOA Version 2.0 (CBDI-SAE$_{TM}$)[2] "Meta-model for SOA, version 2.0)
- Reference Model for Open Distributed Process[3]

1.3 Background and History

The HL7 Architectural Board (ArB) – acting at the behest of the HL7 Chief Technology Officer – commissioned the development of the Behavioral Framework (BF) as a project that was initially executed in parallel with the development of the ArB's SAIF activities. Services, with their foundational emphasis on behavior, provided an essential paradigm to partition and disambiguate the semantics associated with Working Interoperability (what is being standardized and why) from the historical interoperability paradigms traditionally discussed within HL7 (messages and documents). The working assumption behind the BF is that, at some level, the behavioral semantics of a given interaction could be loosely coupled to the semantics of the static information that is associated with the interaction.

Consequently, the BF should ultimately provide a means to specify interactions between trading partners using any of the HL7 (or other agreed-upon) interoperability paradigms – messages, documents, or services. That is, the essential requirement for the BF developed the need to provide a framework that would allow specification developers to formalize behavior, which would ensure the ability to achieve Working Interoperability in a predictable and tractable fashion for a specific interoperability paradigm (messages, services, or documents). The notions of specifying both interaction semantics and information (static) semantics were merged when the SAIF effort surfaced the need for a formal means of specifying behavior semantics in the context of Working Interoperability.

---

[2] CBDI-SAE$_{TM}$ s a comprehensive, defined approach for Service-Oriented Architecture (SOA) including taxonomy, classification and policies together with repeatable service engineering processes that guide the delivery of the agile enterprise, implemented in a knowledgebase with integrity between the architecture concepts, processes, tasks, techniques, and deliverables.

[3] ISO/IEC 10746 RM-ODP.

In general, the SAIF effort identified the following service-based notions as its primary organizing principles and requirements:

- Specifications should have direct traceability to business needs.
- Specifications should be technology-neutral.
- Conformance should be measurable at a component's interface in a Working Interoperability (WI) context.
- Conformance should be specified via conformance statements in a specification and pairs of conformance assertions made by a given technology binding and component implementation.
- Each WI context must be specified for the exchange of both business-based behaviors and associated information.
- WI contexts should be formally specified using the central notion of contracts.

ISO Reference Model for Open Distributed Processing (RM-ODP) is the overarching meta-framework that defines the core concepts, relationships, and constructs of BF. In particular, RM-ODP provides several key structural elements that the BF uses, as well as a rigorously defined notion of *conformance*. RM-ODP is also extremely useful in collecting the multidimensional, multilayered, interrelated aspects of static and behavioral semantics within a single framework through using the viewpoints construct.

Of particular importance to the BF are the following viewpoints:

- The *Enterprise* viewpoint where critical aspects of policy, obligation, and community are captured

- The *Computation* viewpoint, where many of the foundational concepts of the BF are specified

- Additional viewpoints that aid in the synthesis of a unified behavioral model based on *roles* and their obligations scoped to Working Interoperability. One can specify and verify the completeness and

correctness of these viewpoints as *contract accountabilities* (as the following sections discuss).[4]

In summary, the BF provides a set of layered concepts and relationships for defining how collections of artifacts within a component specification collectively define contract templates that are component-specific, which intend to:

- Surface the complexity of interoperability rather than hiding it.
- Formalize accountability in a layered, measured way.
- Provide uniformity across specifications.
- Create a foundation for scalable implementations, including development.
- Provide key guidance for understanding how to implement a given specification.
- Decrease the overall effort involved in producing a given specification.

2 Behavioral Framework Essentials

The primary goal of the Behavioral Framework (BF) is to give standards developers the tools to distribute accountability between participants and to embody it in a Behavioral Model (an implementation of the Behavioral Framework). This section will include an overview of the essential Foundational Concepts of the Behavioral Framework:

- Roles
- Behaviors
- Interactions
- Accountability
- Interactions: Accountability and Behavior
- Contracts
- Solution Specifications

[4] The semantics of the BF using a number of grammars, including Web Services Choreography Description Language (WS-CDL), Business Process Modeling Notation (BPMN2), and (less exactly) Service-Oriented Architecture Modeling Language (SOA-ML), and Unified Modeling Language (UML). For more information, see the OMG Web site (http://www.uml.org/) and the OASIS Web site (http://www.oasis-open.org/home/index.php).

- Collaborations within communities
- HL7 interoperability paradigms

2.1 Roles

A *role* is identification or specification of a party (for example, person, system, or component) associated with a particular capability, capacity, or competency. A given instance of a party may play more than one role, and multiple instances of a party may assume the same role. A role instance usually asserts itself and is verified by another role instance. This role instance, in turn, contextualizes the assertion role, a relationship that HL7 refers to as the "player" vs. the "scoper" of the role respectively.

Examples of roles (Figure 1) include:

- Person is a citizen of a given country
- A system is an order-management system
- A component is an adverse-event management service
- Healthcare Information System
- Order Manager
- Specimen Manager

It is important to note that roles may be systems, organizations, or persons. One system, organization, or person may play more than one role, and a role may be played by more than one instance of a system, organization, or person. Instances of roles are usually time-sensitive. They exist for a defined time after which the instance assuming the role is no longer valid in the role, or may need to be reasserted and re-verified. Finally, roles are usually associated with specific behaviors, permissions, obligations, accountabilities, and prohibitions.

Comment [(1]: Does this clause refer to the ROLE or the INSTANCE?

Note: The RM-ODP Computational and Enterprise viewpoints precisely define these terms.

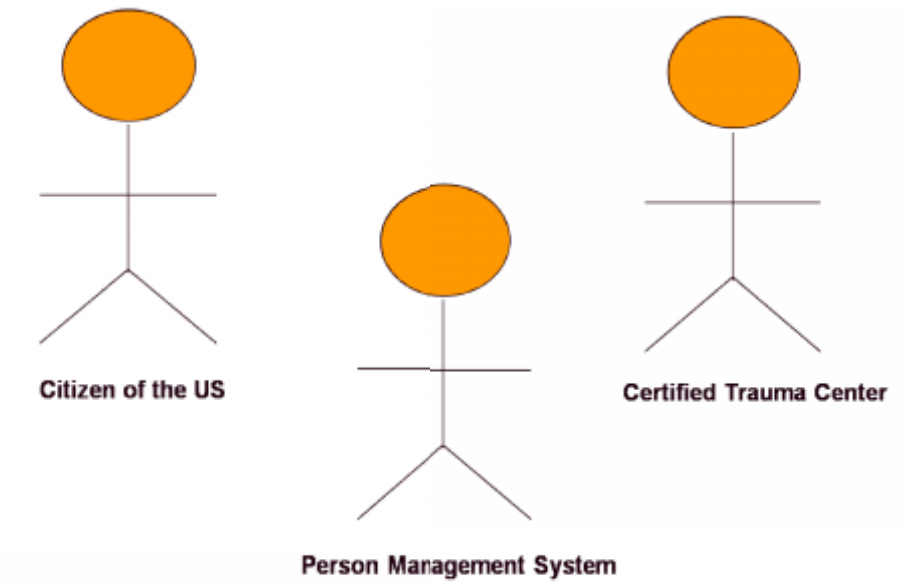


Figure 1: A role is a capability, capacity, or competency, asserted by a given party (person, organization, or system), and normally validated by an independent party. For example, JoeD is a citizen of the US; hospital A is a certified trauma center; and application B is a Person Management System supporting all Reference Information Model (RIM) Person class attributes. Also, note that roles usually are claimed and validated for an identified period, after which they may require restatement and reconfirmation.

2.2 Behaviors

Behaviors are *collections of actions* associated with instances of roles (see Figure 2). The actions are associated with a *set of constraints* on when they can occur. The BF is specifically concerned with the expressions that allow behavior to be abstracted so that systems can perform specific tasks repeatedly and unambiguously.

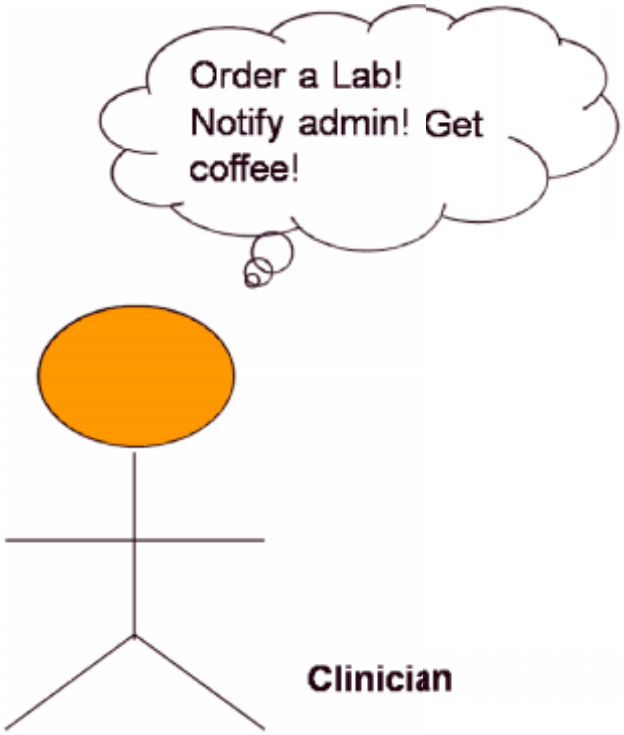


Figure 2: Example behaviors of a person in the role of a clinician.

2.3 Interactions

In general, roles – or, more correctly, instances of a given role -- have defined behaviors that are realized through the execution of *internal or external actions*, the latter, which is more specifically characterized as *interactions* with other role instances. Note that interactions normally involve *information exchanges between roles.* In addition, interactions occur between instances of systems, organizations, or persons that assume well-defined roles, and include the "environment/context" of the role, that is, the specification of other systems playing other roles. Finally, note that "communities" may be assembled because of interactions. That is, the various roles participating in given set of interactions can productively be viewed (and analyzed) from the perspective of a community as "a collection of parties with shared interests, goals, processes, and governance agreements."

Figure 3 shows an interaction between two roles.

Important: Because the BF is only concerned with defining the behavioral aspects of Working Interoperability, it is scoped to specifying interactions; thus, the BF is *not* concerned with the "internal" actions associated with roles.

Figure 3: An example of an interaction between two roles and the resulting restricted set of actions that are in scope. This figure shows an example of an interaction between two human role instances involving a lab order.

Additional examples of interactions executed by software components include:

- Request to a Person service to Create a Person in a registry
- Notification by a Specimen Management service of a Specimen Collection (an act having been completed)
- Publication of an Admission Discharge Transfer (ADT) message

2.4 Accountability

As noted above, accountability can be concisely defined as "Who does what?" Martin Fowler's *Accountability pattern* (Figure 4) shows a more formal definition of accountability. The Accountability pattern is a Unified Modeling Language (UML) expression of the relationship between two parties – one assuming the role of Responsible Party and one assuming the role of Commissioning Party – who have come together in the context of a defined set of responsibilities and goals. The most important feature of the pattern is the *explicit separation of behaviors between the Responsible and Commissioning Parties.*

Note: As will be discussed later in this document, the ability to build durable and agile communities of trading partners bound together by formal notions of

accountability is due mostly to the ability of specifications to define explicitly interactions in a manner in which all accountability in a given interaction rests on the Responsible Party. Therefore, the Commissioning Party can be virtualized.

Note: Patterns are empirically proven approaches of conceptualizing and solving problems. The Accountability pattern may be applied irrespective of technology or implementation, thereby fulfilling one of the basic requirements of the BF's expressive constructs.

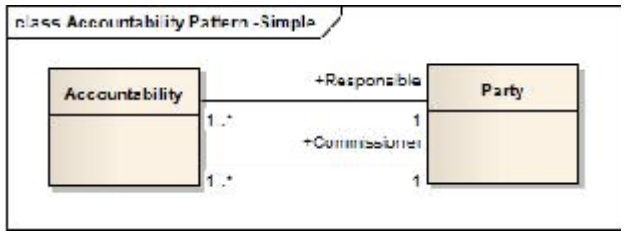


Figure 4: Basic Accountability pattern (Martin Fowler -- http://martinfowler.com/ap2/index.html)

2.5 Interactions: Accountability and Behavior

As can be seen in Figure 4, each instance of accountability involves two party types – the *Commissioning Party* and the *Responsible Party*. This binary structure provides an organic and scalable means of measuring behavior that that is applicable in multiple analysis contexts. In particular, in architectural contexts that are strongly bound to a business process, for example, in a SOA context, the Accountability pattern facilitates the decomposition of a business process and the resulting interactions into atomic instances of accountability.

Thus, one can define behaviors as:

> *A collection of **interactions** with a set of constraints on when they can occur in a given Working Interoperability/business process context.*

Interactions are expressed in the following terms:

- The overall *scope* of each role's obligations and expectations.

- A *series of logically conjoined interactions that* realize accountability on the part of both commissioning and responsible parties. (For example, a lab order is placed that must be unambiguous if it is to be correctly placed.)

- The *goal* of the actions – the result that the commissioning party expects the responsible party to produce.

In the Behavioral Framework, the essential analytical step in creating specifications is to identify the atomic elements of accountability, i.e. the specific milestones that occur over the course of the interactions that define the total overarching accountability required for successful completion of the business process has, in fact, been achieved. Once defined, accountability elements are assigned to interactions. Finally, multiple interactions can be compositionally linked in a single collaboration. Each interaction is associated with one or more of the set of accountability elements that are required to achieve the overarching accountability of the WI instance (Figure 5).

Comment [KGS2]: I'm not sure how you were using the phrase "in total" in this context.

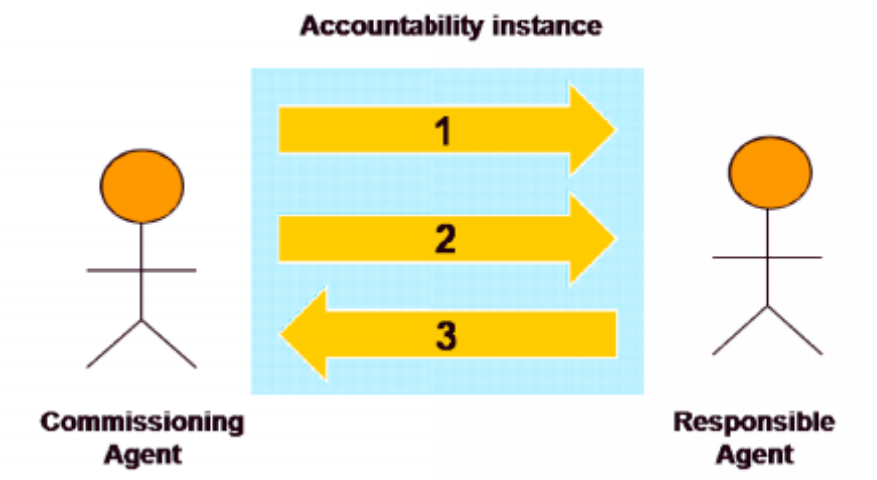


Figure 5: A single interaction between a commissioning agent (CA) and responsible agent (RA). In a given interaction, the roles of CA and RA can only be assumed by one of the two participating roles. The roles cannot not be "switched" during the interaction. (They can, however, be switched during multiple interactions involved in a complex collaboration.) A given interaction allows the exchange of information in either direction between the two roles

(indicated by directional arrows in the figure) to achieve an accountability element. Not shown in the figure is that multiple interactions may be necessary to accomplish all of the accountability elements associated with the WI instance that are needed to fully achieve the overarching accountability. Finally, note that the terms CA and RA are semantically identical to Fowler's Commissioning Party (CP) and Responsible Party (RP) terms. This change was made to align with the software development term "agent" as being more specific than the general business term "party."

Accountability is defined in an interaction by one or more exchanges of information. Accountability also can be tied to a transaction. A *transaction* is a set of interactions happening in a defined sequence.

The following example of two file clerks filing medical records includes two interactions and three instances of accountabilities:

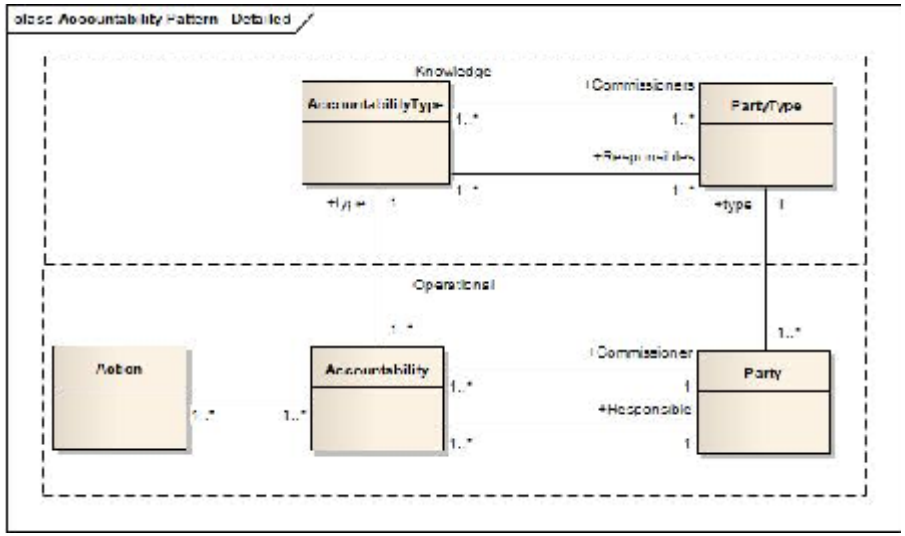
1. File clerk A asks File clerk B to file a personnel record, and File clerk B agrees to do the job. (This represents the first interaction and the first instance of accountability.)

2. File clerk B asks File clerk A to file a lab test record. File clerk A asks questions about the test results, which File clerk B answers. (This represents the second interaction and second instance of accountability.)

3. Both clerks file the records. (The filing of the two sets of records represents the *business process* that needs accountability attached to it. Therefore, this action is the third set of accountability that comprises the other two interactions whose success is required.)

2.6 Contracts

Contracts aggregate *accountability*, typecast *parties*, and define *actions* to support *Accountability Types*, which are contracts that bind design-time specifications to run-time components. Fowler captures this by defining two levels in his Accountability pattern (Figure 6 and Table 1). For example, at design-time (Knowledge level), travel agents issue tickets for a traveler through the Accountability Type of Travel Agency. At run-time (Operational level), Expedia (Responsible party) issues Joe (Commissioning party) a ticket to Kyoto. The run-time Accountability is the set of activities that collectively define Expedia acting as a Travel Agent for Joe. Examples include the following: Create Account, Show Ticket Options, Purchase Ticket, and Deliver Ticket.



414

415 **Figure 6: Fowler's Accountability pattern refined to separate design-time (Knowledge level)**
416 **from run-time (Operational level) constructs.**

Element	Description	Notes	From
Action	Represents something that happens.	In this case, *action* is tied to accountability to stress that within a community, activities have meaning.	RM-ODP
Accountability Type	Represents valid types of accountability.		Martin Fowler
Accountability	Represents a complex graph of typed relationships between parties.	Who does what when?	Martin Fowler
Party Type	Represents meta-class for a party.		Martin Fowler
Party	Represents people and organizations.		Martin Fowler
Knowledge Level	Represents a group of objects that describe how another group of objects behaves.	Represents the meta level.	Martin Fowler
Operational Level	Represents a group of objects whose behavior is described by a knowledge level.		Martin Fowler

417 **Table 1: Core concepts defined by Fowler's Accountability pattern.**

418 In summary, contracts describe an agreement, which defines the interactions
419 between and among instances and collections of roles. Contracts specify rules
420 about content, platforms, and localizations. The contract defines the
421 requirements for commissioning and responsible agents, and interaction
422 patterns, accountabilities, permissions, and restrictions that collectively define
423 the requirements for meeting the specified Accountability pattern. In addition,
424 however, contracts may contain Quality of Service Agreements that pertain only
425 in a particular environment or deployment, and not part of the specification
426 itself.

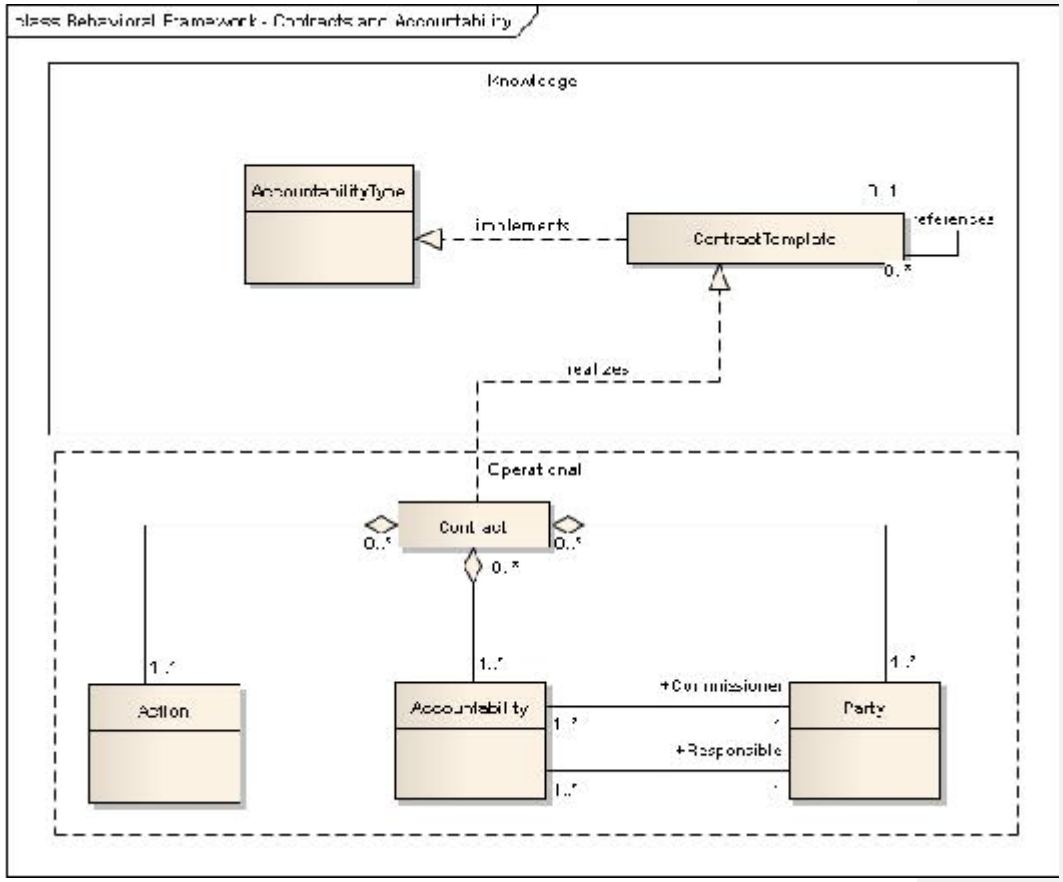
427

2.6.1 Contract Templates

It is worth asking why contracts -- constructs whose primary implications are realized at run-time via specific, deployed technology structures – should be of interest at the specification (design-time) level. To answer this question, recall that the overarching motivation of the ECCF (as noted in the ECCF document) explicitly states *the relevant assumptions that collectively result in achieving Working Interoperability (WI) between trading partners.* To accomplish this goal, a layered specification process that exposes the salient aspects of a given run-time interaction *before* that interaction is of considerable benefit in achieving tractable, scalable, and reproducible Working Interoperability.

As such, the BF specification framework allows for the identification of *contract templates* that enable the specification of constructs, such as Accountability, Party Types, Interactions, and Interaction Patterns. Contract templates (Figure 7) are instantiated at run-time and provide analysis, design, and run-time value to achieving Working Interoperability between trading partners.

Note: This Quality of Service Agreement is an example of a technology *localization,* as discussed in the ECCF document.



445

446 **Figure 7: An extension of the Accountability pattern depicting the concept of contracts capturing**
447 **actions as a critical link between Accountability types and Accountabilities.**

448 ### 2.6.2 Contracts, Specifications, Conformance, and SAIF

449 Both contracts and contract templates are formalisms for expressing the
450 accountability involved in any given interaction in a manner that is *explicit* and
451 not – as is often the case – allowing these details to reside exclusively (and, for
452 non-developers) in the software code, configuration specifications, or other
453 deeply technical documentation. As such, contracts and contract templates can
454 be contextualized in the SAIF Stairway to Heaven, as shown in Figure 8.

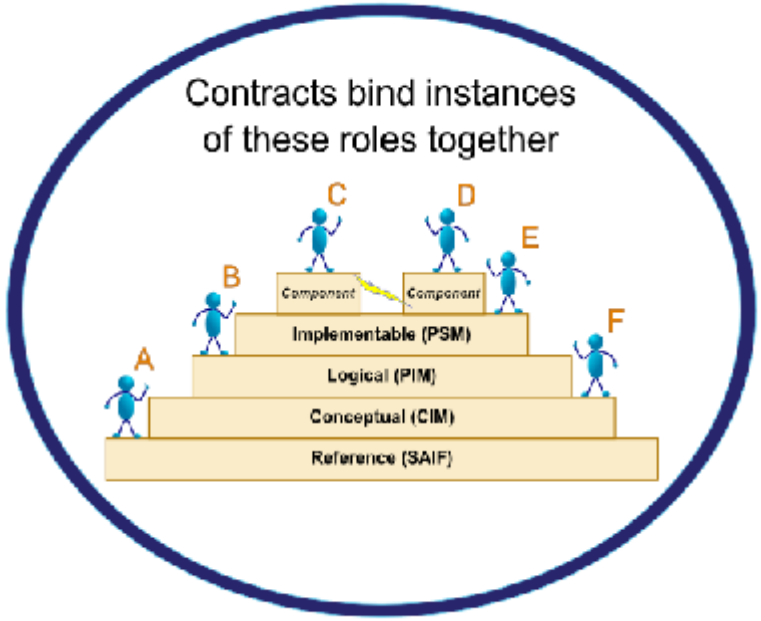


Figure 8: The ECCF specifications provide the keys to achieving Working Interoperability in a tractable, scalable, and predictable manner. The BF provides the means of specifying the interactions between those parties (also called trading partners) who want to achieve Working Interoperability. Thus, the notion of *contracts* contextualizes the SAIF Stairway to Heaven. Although not shown in the figure, a trading partner working at the Reference level can still interact with the other trading partners via contract-driven interoperability. However, the terms of the contract would be neither precise nor accurate.

Figure 8 legend:

- The lightning bolt represents implementation.
- Implementable (PSM) = Platform-Specific Model
- Logical (PIM) = Platform-Independent Model
- Conceptual (CIM) = Computationally Independent Model

The BF, where appropriate, allows the making of explicit conformance statements regarding accountability between roles such that the terms of accountability may be met. Accountability, as noted earlier, is defined by

contract templates at design-time, and manifest at run-time by instances of contracts. Thus, in summary, the validation and certification of conformance is based on evaluating interactions between roles, based on their contractual obligations.

2.7 Solution Specifications

The Solutions package contains informational and behavioral elements relating to the way that instances of specified roles are assembled to provide an Accountability Community (AC) whose focus is on achieving a particular, overarching business goal, i.e. a given Business Capability characterized by one or more verifiable accountabilities. A given Solution provides behaviors associated with two or more specifications – each of which is specified via an individual ECCF specification stack instance. As discussed above, the Solutions package is dependent on elements in the CIM, PIM, and PSM packages. Consequently, a given Solution expresses the collection of conformance statements, which are expressible at any level-of-abstraction, as required by the Solution package's overarching deployment context.

Solution Specifications reflect a set of choices by the specification developer regarding how conformance will ultimately be measured and certified in any instance of the Working Interoperability in which the Solution's particular specification is implemented. In particular, these choices reflect the following:

- The interoperability paradigm chosen for implementation and deployment (messages, documents, services – see a more detailed discussion below). This choice is, in turn, dependent on a number of factors including:
    - The characteristics of the AC (for example, loosely coupled, intra- vs. inter-enterprise, presence or absence of trust fabric, and complexity of interactions).
    - The maturity and extent of existing governance in the deployment space.
- A number of design- and PSM-level choices including:
    - *Interaction granularity:* Level-of-detail involved in the various interactions that occur between the commissioning and responsible agents over the course of the Solution.

- *Functional Profiles:* The granularity of operations at the interfaces.
- *Semantic Profile:* The corresponding syntactic structure and semantic complexity of the information exchanged during the various interactions.

2.8 Collaborations within Communities

Collaborations are ways to compose accountability into a business process between multiple parties (for example, Order Fulfillment, Portions of Adverse Event Management, and Treatment Plan Management). In enterprises, collaborations are often considered the result of an integration project, but through the Behavioral Framework, they represent a collection of predefined resources that can be defined and arranged to meet certain desirable goals and obligations. These resources take the form of other contracts; that is, services and V3 messages.

The Behavioral Framework enables the development of specifications that constrain sets of actions that, in turn, **collectively distribute the *Accountability*** necessary to satisfy a given business capability across one or more Accountability Communities (ACs) involved in particular collaborations. In other words, at deployment and run-time, the Accountabilities that define the successful delivery of a given business capability identified by, and specified in, one or more ECCF specification stack instances, must be satisfied by the collective behavior of *all* the participating parties involved in a particular AC. (See Figure 9.)

Comment [(3]: The Word grammar check said that this sentence was too long, so I eliminated the final clause. I figured that the final clause wasn't necessary because the sentence was talking about all of the parties involved in the AC. Thus, it probably doesn't matter when they join the AC. <KGS>

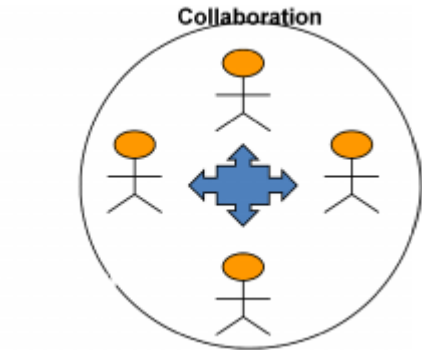


Figure 9: An Accountability Community (AC) consisting of a number of well-defined obligations, goals, and Accountabilities. Communities are defined by referring to the underlying roles and defined contracts between roles. Conformance levels for collaborations are specified in terms of the underlying roles, their associated behaviors, and the resulting possible contracts.

The complexity of the overarching business capability is therefore a major factor driving the choice of the interoperability paradigm that most effectively fulfills the defined Accountability.

The following example of a customer ordering a book online illustrates a collaboration community:

- Collaboration Accountability: The customer receives a book.
- Community: Customer, seller, warehouse, payment site
- Transaction: The customer pays for the book.
- Interactions (each with their own accountability):
    1. The customer orders the book.
    2. The customer payment information is validated and the transaction is charged to the credit card.
    3. The warehouse is notified of the order and provides a shipping number.
    4. UPS is notified of the delivery.

This example includes five accountabilities, four interactions, and one transaction. The fifth accountability and the business goal is the customer getting the book. The seller, warehouse staff, and payment site collaborate to ensure that the customer gets the book.

2.8.1 Services

Services are abstractions of role behavior that describe Accountability in a durable, reusable manner, which formalizes the separation of concerns inherent in the underlying Business Capability. The semantics of the BF are expressed best via services as the appropriate interoperability paradigm in situations where:

- The integration semantics expressed in the PSM level of *mature* specification stack instances are *clearly traceable* to those expressed at the CIM and PIM levels (for example, via Business Architecture and/or Domain Analysis Models). Comprehensive, complete traceability means that the accountability inherent in the Business Capability is unambiguously expressed (i.e. expressed in both a human-readable and computational representation, such as WSDL[5]) in contract specifications available for discovery in an appropriate run-time environment (for example, in a contract/metadata registry).

Furthermore, when services are the chosen interoperability paradigm, the specified Business Capability:

- Represents a known, unambiguously describable, and constant set of responsibilities defined community for the involved trading partners who are, in turn, part of the larger Accountability Community.
- Has been leveled so that the Commissioning agent has no accountabilities in the interaction or business process. The accountability rests solely on the responsible agent, a fact that is most often directly expressed via the involved contracts that collectively define the accountability.
- Has been appropriately contextualized within the participating organization-centric enterprise architecture. (Examples include design patterns, usage, or composability constraints, and localizations.)

In combination, these factors allow all involved commissioning agents to be virtualized; that is, to be interchangeable (Figure 10). The virtualization of commissioning agents thus results in a durable community with durable goals and obligations that may be exposed simply and consistently because systems may be characterized by the services that they expose and the Accountabilities inherent in the various compositions of those services.

[5] WSDL = Web Services Definition Language

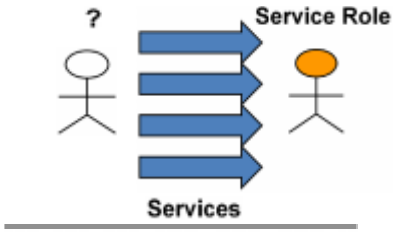


Figure 10: The choice of *services* as the interoperability paradigm allows commissioning agents to be virtualized because commissioning agents have no Accountabilities, i.e. all Accountabilities are associated with responsible agents (labeled as "Service Role" in the graphic above). The use of services as the interoperability paradigm therefore allows for the development of a durable community consisting of any number of commissioning agents with access to the required responsible agents able to fulfill the Service Roles required to meet the community's various business needs. For example, a Service Role might provide the Patient Registrar. All patient registrations could be handled by the Service Role *Register Patient* in where all of the obligations and responsibilities associated with patient registration are handled by and accountable to the Patient Registrar via its Register Patient service. The service could therefore be used by any commissioning agent with access to the service.

2.8.2 Messages

The choice of *messages* as the interoperability paradigm carries with it a fundamentally different – and considerably more limited – ability to define Accountability Communities. In particular, when applied in a loosely coupled environment with unknown run-time context, such as that of the traditional deployment topology for HL7 messages, the messaging interoperability paradigm provides *fine-grained* Accountability between commissioning and responsible agents who *share Accountabilities during their interaction.* This is in distinct contrast to the situation found with the services interoperability paradigm, where Accountabilities are possessed *solely* by the responsible agent in a collaboration.

The immediate consequence of the sharing of Accountability is that – from the perspective of specification itself, the AC is not durable. Instead, the AC is defined and invoked at run-time, usually in response to a single business "event" that invokes one or more messages specified as responses to that event. In particular, the obligations, goals and (ultimately) the Accountabilities associated with the business event are invoked by the message "contract" inherent in the

sending of the message. As a result, the details of the "contract" – that is, the definition and distribution of the Accountabilities – must be contained in the body of the message itself and thereby parsed at run-time by message-specific machinery. The result of these "messaging facts of life" is that the AC is limited to the two trading partners involved in the specific exchange defined by the event trigger and message-body specifics (Figure 11).

In addition, note that if the obligations, goals, and Accountabilities contained in the message body are not violated, the community may have more than two members by virtue of having multiple end-points. However, in this situation each end-point must have the ability to parse the message body to determine the portions of the shared Accountability that is distributed to it. Historically, HL7 Version 3 messages have attempted to address these complexities through the concepts and constructs associated with Application Roles and Receiver Responsibilities. The BF formally defines and extends the concepts and constructs of Application Roles and Receiver Responsibilities in the service interoperability paradigm.

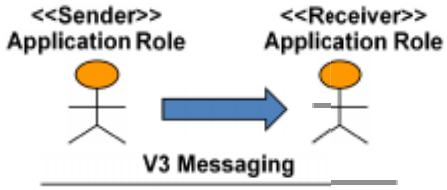


Figure 11: In contrast to the service interoperability paradigm, messaging defines an Accountability Community (AC) of two. This AC is based on fine-grained interactions in which the goals, obligations, and Accountabilities of a given collaboration are shared between the two Application Roles, which limits the size of the AC and requires that each end-point have the ability to parse the body of the message to determine the specifics of Accountabilities distributed to it.

In summary, HL7 Version 3 *messages* are interaction specifications that support a fine-grained accountability between a commissioning and responsible agent. It makes demands on both parties to accomplish the desired goal, and presumes little in terms of existing infrastructure. HL7 V3 specifications are ideal for "drive-by interoperability."

Note: "Drive-by interoperability" is defined with minimal or absent run-time context.

One may view *documents* as a form of messaging that focuses on static content and carries a set of additional constraints and expectations involving persistence, wholeness, human readability, and so on.

Services are abstractions of role behavior that describe accountability such that every commissioning agent is conceptually identical from the perspective of its role in a given specification. Services are deployed as interfaces (durable structures) that are reusable in multiple situations and may be adapted to multiple infrastructures.

2.9 More on Interoperability Paradigms

Different approaches to achieving Working Interoperability have been used and are, in general, representative of three approaches to implementing behavioral interoperability between systems. Historically, Working Interoperability is referred to within HL7 as interoperability paradigms (IP). The three interoperability paradigms are messages, documents, and services.

HL7 Version 3 *messages* are Interaction Specifications that support a fine-grained accountability between a commissioning and responsible agent. It makes demands on both parties to accomplish the desired goal, and presumes little in terms of existing infrastructure. HL7 V3 specifications are ideal for "drive-by interoperability."

Note: "Drive-by interoperability" is defined with minimal or absent run-time context.

One may view *documents* as a form of messaging that focuses on static content and carries a set of additional constraints and expectations involving persistence, wholeness, human readability, and so on.

Services are abstractions of role behavior that describe accountability such that every commissioning agent is conceptually identical from the perspective of its role in a given specification. Services are deployed as interfaces (durable

structures) that are reusable in multiple situations and may be adapted to multiple infrastructures.

2.9.1 Business Process and Interoperability Paradigms

In the context of the BF, the term *business process* is used to refer to one or more *defined* interactions between two trading partners who desire to accomplish a common goal. A business process may be as simple as a one-way exchange of information following a business "trigger" event that occurs in the operational context of one of the partners – that is, the setting in which much of HL7's traditional messaging constructs have been defined – or as complex as a multi-operation, bidirectional coordination of both behavior and information exchange. Regardless of the details, a business process carries with it the notion of *human* oversight in terms of accountability, conformance, and standards for quality and execution. When trading partners use software systems to perform all or part of a given business process, notions of *interoperability* in general and *computable semantic interoperability* in particular come into consideration. In particular, the concept of the *interoperability paradigm (IP)* that is used may have a significant impact on the ability of the participating software systems to correctly and/or support completely the execution of the business process.

In particular, as the complexity of the business process that links two trading partners increases, the importance of the chosen IP to provide *traceability* from the overarching, human-defined business process to the supporting *technical implementation* increases. When one considers the three IPs most commonly used – messages, documents, and services – one finds considerable differences between the three approaches, particularly regarding *Accountability* and *Conformance* (Table 2).

In general, services provide the cleanest and most complete traceable link between a given business process between two trading partners and the underlying implementation. Services have the ability to separate concerns and to bind specific operations (*functional profile*) to specific semantics (*semantic profile*) in a specific context and in a testable, verifiable fashion (*conformance profile*)

Messages offer the least support as the complexity of business processes increases, primarily because of their fine granularity (business processes tend to

be more coarsely granulated), and context-free nature. Documents have the advantage of being "human understandable" and, therefore, if a given business process can be traced to a single document, serve as an effective IP. However, as the complexity of the supported business processes increases, the realities usually involve multiple documents and, hence, the traceability from a *single document* is less than desired. Thus, services most effectively bundle the combination of operations, information definition, accountability, and conformance required to provide full traceability from the IP-to-business process.

Paradigm	Accountability	Conformance	Notes
Services	Interface bundles and expresses the specific operations that handle service obligations in the context of a business process.	Conformance is bound to implementable interface via a conformance profile.	Accountability is formalized through contracts and separation of concerns (functional profile and semantic profile).
V3 Messages	Receiver and sender responsibilities are defined at a granular level and linked to a single initiating trigger.	Conformance is linked to static structures but implied only in dynamic structures.	Accountability is only partially specified, or underspecified.
Documents	Document models – for example, Clinical Document Architecture – provide traceability and accountability only if a single document can define a given business process.	Conformance is tied to structure, encapsulation, persistence, and human readability of a single document.	Accountability for exchange is deferred for the receiver role, and is explicit for the sender role. Accountability is not easily bundled across business processes requiring multiple documents.

Table 2: Comparison of interoperability paradigms and business context regarding accountability and conformance.

2.10 Summary

The BF captures the contractual nature of integration by dealing explicitly with accountability partition across multiple artifacts (described in Section 3), including:

- Reusable structures
- Computable representations of business processes
- Rules important to an implementation

Because the BF provides the ability to specify units of Accountability that are based on the RM-ODP Enterprise and Computational viewpoints, the BF harmonizes semantics among the various interoperability paradigms by allowing paradigm-specific differences to be exposed before run-time. (For example, static and event semantics are well expressed in V3, while services provide durable, role-bound interfaces.)

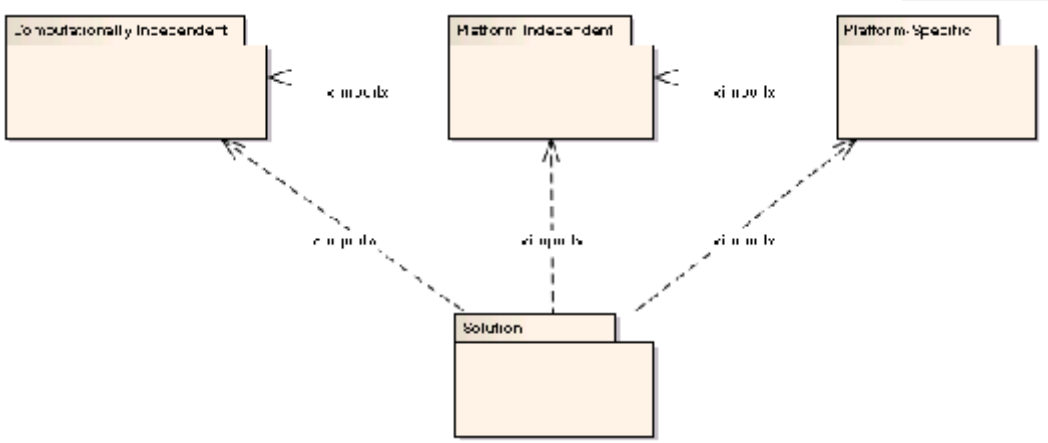
3 Behavioral Framework Foundational Concepts and Models

This section describes the Behavioral Framework (BF) foundational concepts and models. The packages for each model type include:

- Computationally Independent (CIM)
- Platform-Independent (PIM)
- Platform-Specific (PSM)
- Solution

3.1 Package Organization

The Behavioral Framework is represented as a collection of related Unified Modeling Language (UML) packages (Figure 12 and Table 3). This organization separates three packages in which roles bind to accountability from the shared specifications of behaviors which define a given solution. Readers familiar with the ECCF will note the correspondence between first three folders and the rows of the Enterprise Conformance and Compliance Framework (ECCF) specification stack (SS). The three BF packages associated with the rows of the ECCF SS are, in fact, models defining the semantics of the grammar used to develop artifacts that populate cells within the like-named row, primarily in the Computational viewpoint of the ECCF SS. The Solution package contains the models that define the BF-relevant concepts and associated grammar that you can use to develop a particular technology binding and solution.



749
750 **Figure 12: The package topology of the BF. Roles are bound to Accountabilities in the CIM, PIM,**
751 **and PSM packages, each of which contains artifacts that are specific to levels of abstraction. The**
752 **package topology is intentionally structured to correspond to the three levels of the OMG's**
753 **(Object Management Group) MDA (Model-Driven Architecture) framework and is mirrored in**
754 **the rows of the ECCF. An example for the "Import" relationship is, "If Package B (e.g. Solution)**
755 **imports Package A (CIM), Package B can use Package A's types, but not vice versa."**

756

Element	Description	Notes	From
Computationally Independent Package	The Computationally Independent (CIM) package contains informational and behavioral elements of a subject specification. These elements are characterized by the classes in the package, and often exhibit business-aligned capabilities.	For example, a conceptual specification covering travel agency would allow for ticketing without detailing the design and architecture, which would allow the ticket to be delivered to the customer.	HL7 ArB
Platform-Independent Package	The Platform-Independent package contains informational and behavioral elements of a subject specification, related to the CIM package	For example, a conceptual specification covering travel agency would allow for ticketing without detailing the design and architecture	HL7 ArB

Element	Description	Notes	From
	elements. The classes in the package characterize these Platform-Independent elements. These elements often exhibit logical refinements of the business-aligned capabilities expressed in the CIM package.	that would allow the ticket to be delivered to the customer. The Platform-Independent package may make it clear that all travel tickets are handled the same way, and that bus, train, and plane tickets will share a universal numbering scheme.	
Platform-Specific Package	The Platform-Specific package contains informational and behavioral elements of a subject specification, related to the CIM and Platform Independent package elements. These Platform-Specific elements are characterized by the classes in the package, and often exhibit platform constraints on top of the logical refinements of the business-aligned capabilities expressed in the CIM package.	For example, a conceptual specification covering travel agency would allow for ticketing without detailing the design and architecture that would allow the ticket to be delivered to the customer. The Platform-Independent package may make it clear that all travel tickets are handled the same way, and that bus, train, and plane tickets will share a universal numbering scheme. The Platform-Specific package would detail how a set of .NET web services would use the MS GUID generator to provide	HL7 ArB

Element	Description	Notes	From
		that common numbering scheme.	
Solutions Package	The Solutions package contains informational and behavioral elements relating to the way that instances of the other three packages are assembled to provide a community of accountability to achieve some overarching business goal. It may provide behaviors associated with subject specifications assembled at various levels of conformance as desired.	Thus, a Solutions package may be defined that requires multiple airlines to be conceptually conformant, and the e-Commerce system to be of a particular platform.	HL7 ArB

Table 3: Global definitions for BF including source of name and definition (see Section 2 for further discussion).

The following BF Foundational Concepts – discussed in Section 2 – are common across all four packages:

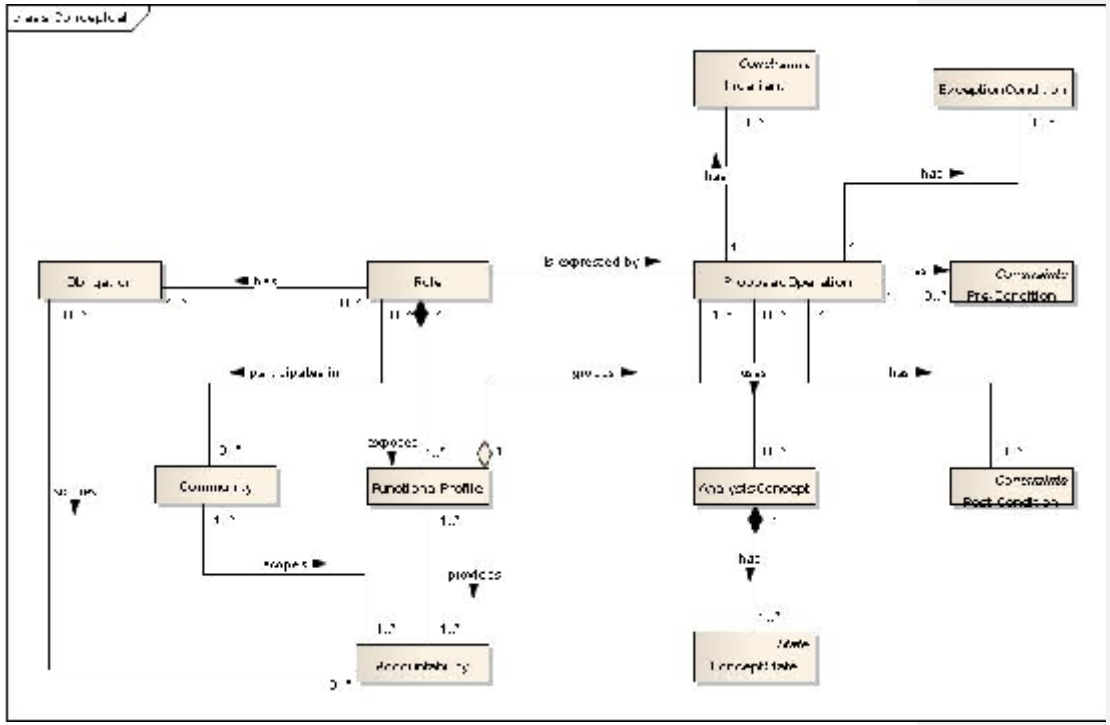
- ***Role*** - A cohesive set of capabilities, capacities, or competencies abstracted as behaviors, which can be invoked at run-time.
- ***Behavior*** – Sets of actions and constraints on when they can occur.
- ***Interaction*** – Something that happens between a role's interfaces and other roles in its environment.
- ***Contract*** (also called Contract Template) - An agreement covering part of the collective behavior of any number of role instances.
- ***Interface*** - An interface is 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 that define when the identified interactions can occur.

3.2 CIM Package

Figure 13 shows the model of the elements contained within the Computationally Independent Model (CIM) package, while Table 4 contains the documentation within the model itself. Referring to Table 4 for definitions, the following concepts are of particular importance in the CIM package.

A given role is associated with a functional profile that defines the set of actions that the role can be held accountable to perform, as a result of "commissioning" the functional profile. Thus, functional profiles aggregate behaviors (such as run-time operations abstracted to the CIM level-of-abstraction).

From a behavioral perspective, operations are tied to familiar analysis concepts, such as pre- and post-conditions, invariants, and exception conditions. From an "information" (static semantics) perspective, operations at the CIM layer are linked to analysis-level concepts, such as those articulated in business rules, policies, and so on, and described in detail in Domain Analysis Models.



787

788 **Figure 13: Elements of the CIM package of the BF.**

789

Element	Description	Notes	From
Accountability	Accountability is defined in terms of "Who does what when" within a community designed to achieve some set of business goals. It is expressed in terms of a responsibility or a need.	Functional profiles provide accountability to fulfill a role's obligation within a community by grouping behaviors (Proposed Operations).	HL7 ArB
Analysis Concept	An Analysis Concept is tied to		HL7 ArB

Element	Description	Notes	From
	the static model that represents information at the CIM level of the ECCF, usually expressed in terms of the static classes that make up the information components of a specific DAM.		
Concept State	Analysis Concepts always have at least one state that may or may not be expressed through Proposed Operations and / or an expressed state machine.		HL7 ArB
Community	A Community is an aggregation of responsibility and need that are expressed in terms of Accountability. Communities have some objective, although this may not be expressed.	Communities may be expressed simply in terms of a responsible agent (a service), or both a commissioning and responsible agent (a messaging solution).	RM-ODP, profiled by HL7 ArB
Exception Condition	In RM-ODP, an Exception Condition is known as a fault. A fault is something that could lead to an error.	Exceptions can be active or dormant. Active exceptions can only be detected when they produce errors. Errors appear at the Platform-Independent level.	RM-ODP, profiled by HL7 ArB
Functional Profile	A Functional Profile is a collection of		HL7 ArB

Element	Description	Notes	From
	Proposed Operations that align with some intended usage patterns. Often, these are characterized by quality considerations, such as security or performance, though they may not be so.		
Obligation	Roles have behaviors associated with accountabilities that are perceived as obligations within the community. Behaviors are collections of actions with constraints on when they occur.	No specific UML meta-class is extended to express this concept. If required, the fact that some behavior places or fulfills an obligation may be stated in a comment on that behavior. In the Behavioral Framework, behavior is not a modeled concept in its own right, but is abstracted into Functional Profiles and Proposed Operations.	RM-ODP, profiled by HL7 ArB
Proposed Operation	The actions a role may take within a community.		Component Based Development and Integration (CBDI), profiled by HL7 ArB
Role	A set of obligations and responsibilities within a given community. A cohesive set of		HL7 ArB

Element	Description	Notes	From
	invokable capabilities, capacities, or competencies realized through behaviors, which are realized through Proposed Operations.		

Table 4: The element names and definitions from the BF CIM package.

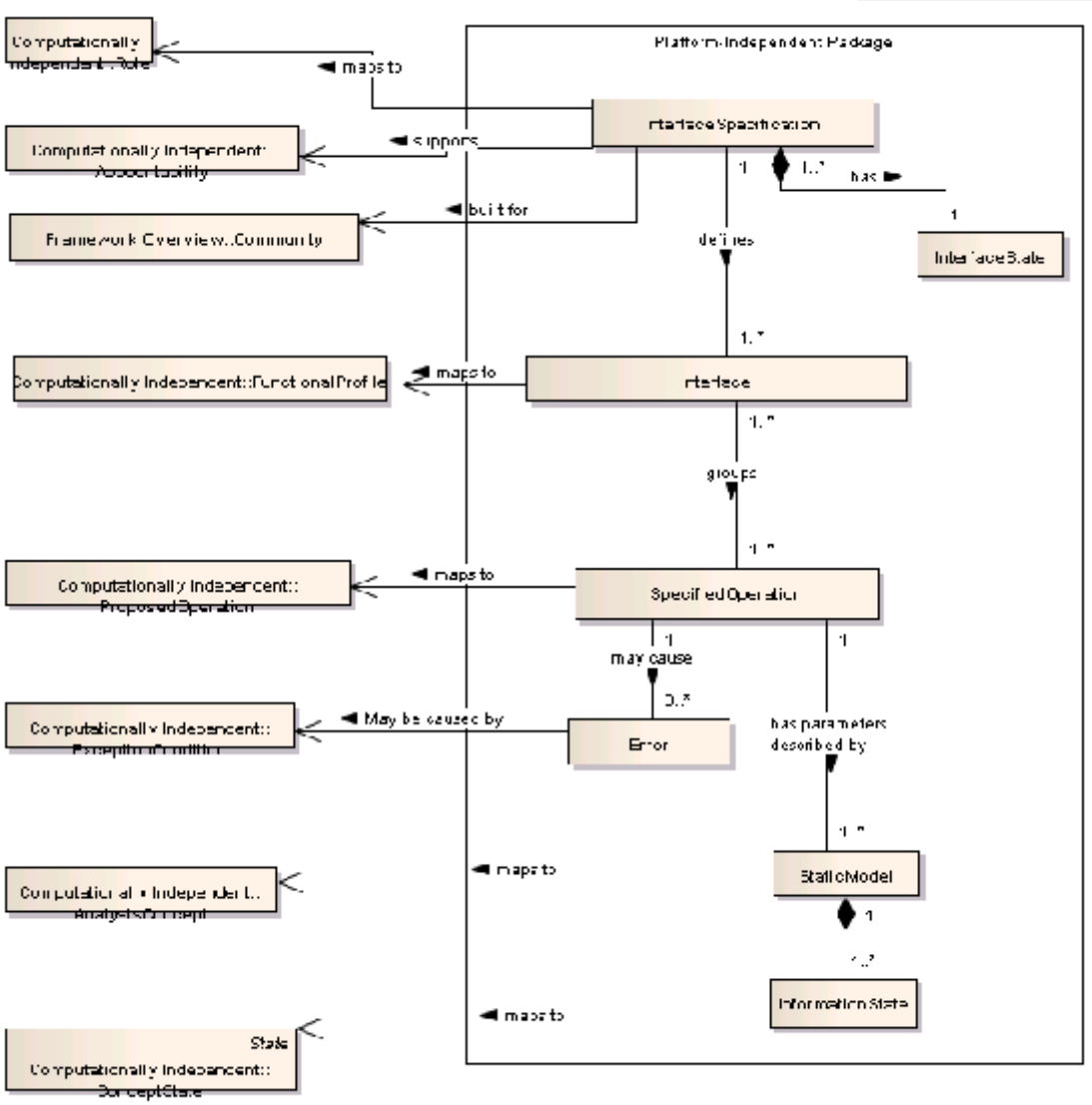
3.3 PIM Package

Figure 14 contains the model of the elements contained within the PIM (Platform-Independent Model) package. Table 5 contains the documentation within the model itself. Referring to Table 5 for definitions, the following points are of particular interest and importance at the PIM layer of the BF.

The PIM package contains elements that are more refined than those at the CIM level are. The diagram below shows these elements and their relationships to the CIM-level elements. The basic relationship of **Role -> Functional Profile -> Operation -> Information** is preserved, but takes on additional information, and begins to be shaped by an understanding of implementation details.

For example, a single Proposed Operation (CIM level) may be decomposed into additional Specified Operations. Additionally, these Specified Operations would use logically specified information elements. An example would be elements contained an HL7 Refined Message Information Model (RMIM). As part of PIM-level refinements, Exception Conditions can specify actual errors that are manifest at the interface. If this is important in the Specification Stack Subject, then these errors may need to be dealt with as exchanges of information that portray the error to trading partners. The Specified Operations are expected to adhere to the Analysis Concepts that apply to the Proposed Operations (CIM level). The pre- and post-conditions of a Proposed Operation should be preserved in the Specified Operation (or collection of operations), that is, the semantics conveyed by CIM-level analysis should not be lost.

Note: the ECCF defines CIM-to-PIM traceability as a form of compliance.



814

815 **Figure 14: Elements of the PIM package of the BF.**

816

Element	Description	Notes	From
Error	An Error is a state in an object's state machine. It may lead to a failure.	For interoperability specifications, there may or may not be a tie between the state of an information object and an error, and the error may or may not be messaged to other members of the community.	RM-ODP, profiled by HL7 ArB
Information State	Static Models always have at least one state that may or may not be expressed through Specified Operations and / or an expressed state machine.	Information State is used for conformance testing.	HL7 ArB
Interface	An Interface is 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 for when they can occur.	Object is used in a general way. An Object might mean a system.	RM-ODP
Interface Specification	A specification of the interface that expresses the traceability to conceptual concepts, including the ties to community, accountability, and role.	The Interface Specification should express traceability to one or more functional profiles.	CBDI, profiled by HL7 ArB
Interface State	Interfaces always have at least one state that may or may not be expressed formally.	In practice, the Interface State class has rarely been used, but it is clear that certain types of	RM-ODP

Element	Description	Notes	From
		interfaces may need this (control interfaces, for example).	
Specified Operation	Operations expressed logically that support accountability. While a tie to the business-specific behavior (the proposed operation) exists, the specified operation refines this behavior and allows patterns of activities to be grouped together to support the role.	For example, a business operation of "Order Ticket" may rely on a conversation through the specified interface consisting of numerous messages.	CBDI and RM-ODP, profiled by HL7 ArB
Static Model	A Static Model represents information at the Platform-Independent level of the ECCF, usually expressed in terms of the static classes.	The Static Model may use formalisms and patterns that are more refined than those used for DAM are. For example, the static model should be tied to a formal data type specification and would likely be tied to formal value set representations.	HL7 ArB

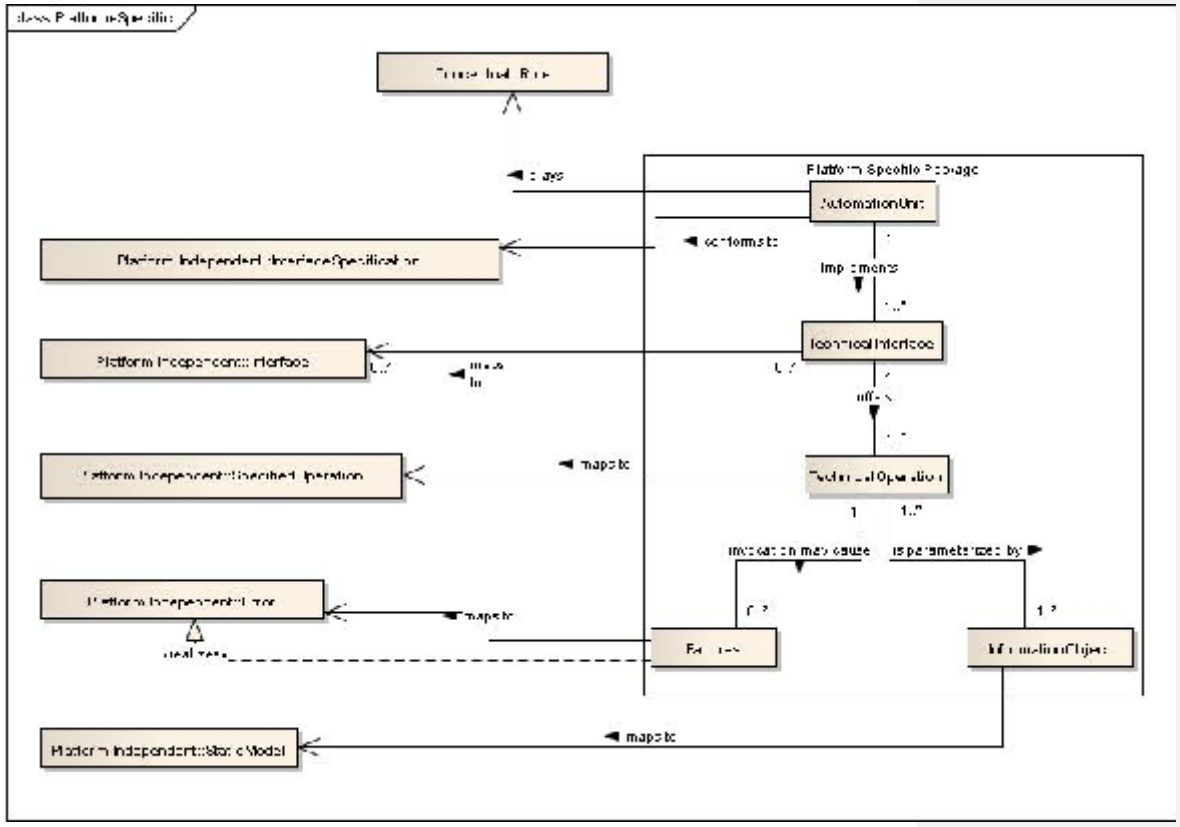
817 **Table 5: The element names and definitions from the BF PIM package.**

818 ## 3.4 PSM Package

819 Figure 15 contains the model of the elements contained within the PSM
820 (Platform-Specific Model) package. Table 6 contains the documentation within
821 the model itself. Referring to Table 6 for definitions, the following points are of
822 particular interest and importance at the PSM layer of the BF.

823 The PSM package further refines the PIM elements. The following diagram
824 shows how actual deployed artifacts are portrayed as taking on roles and
825 conforming to Interface Specifications by implementing technical interfaces. This
826 implementation must conform to the platform of choice; that is, the way a Java

827 object implements an interface in a Java 2 Platform, Enterprise Edition (J2EE)
828 environment may be different from the way that a Java Web Service is
829 implemented. One key note is that at the PSM level, the technology may
830 experience failures, which should be mappable to errors and to exception
831 conditions. In an interoperability scenario, these failures may need to be
832 communicated as contextualized within the business. Database failures, for
833 example, may be characterized as, "The Lab Order was not placed." Finally,
834 information objects that comply with the PIM's Static Model support the
835 technical operations via compliant transforms from the PIM row to the PSM row.



836

837 **Figure 15: Elements of the PSM package of the BF.**

Element	Description	Notes	From
Automation Unit	An Automation Unit describes the implementation of a single service, several services, or an application. It is itself a specialized form of versioned specification. It consists of a collection of deployable artifacts.	An Automation Unit can be decomposed into several distributed Automation Units. Each Automation Unit is hosted on a separate node of a computing network. An Automation Unit might also represent a part of the implementation of a service, or several services or an application.	CBDI, profiled by HL7 ArB
Failure	Failures are realizations of errors that are expressed to the community. Failures signal the inability to fulfill completely the obligation of the role to the community.	Not all errors lead to failures.	RM-ODP, profiled by HL7 ArB
Information Type	A type that defines the data accessible to a service via its Service interfaces. It is bound to the platform expression of that object.	For example, an XML schema would express the Information Type. Technical operations use Information Types as parameters.	CBDI, profiled by HL7 ArB
Technical Interface	An interface provided by an Automation Unit, which is technology specific.	For example, a specific Technical Interface is defined in Java.	CBDI
Technical Operation	A specific function provided by a particular Technical Interface, which is technology specific.	For example, a Technical Operation is a Java method.	CBDI

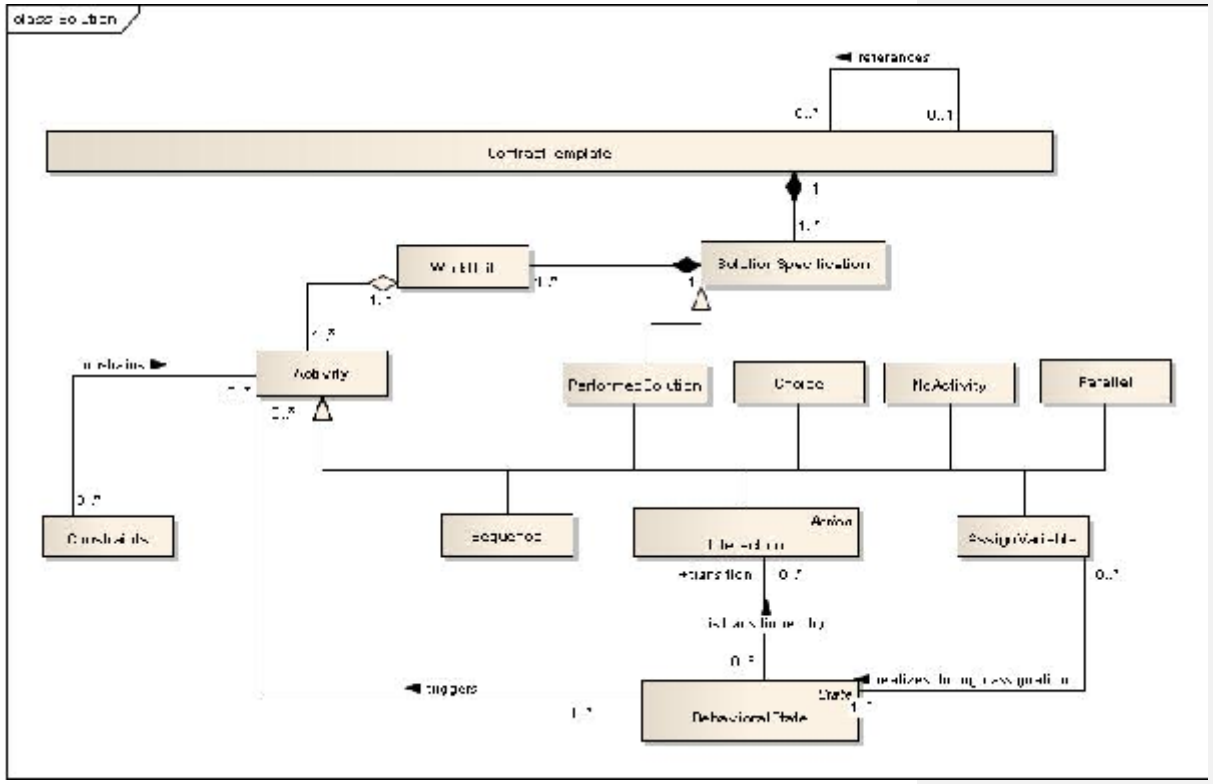
838 **Table 6: The element names and definitions from the BF PSM package.**

3.5 Solution Package

Figure 16 contains the model of the elements contained within the Solutions Package. Table 7 contains the documentation. Referring to Table 7 for definitions, the notion of the Contract Template is of particular interest. It is composed of a Solution Specification that:

- Relates and groups interactions into units of work.
- Relates units of work to accountability.
- Relates units of work to each other.
- Enumerates and describes states that are appropriate to the overall solution, and relates them to the units of work.
- Relates accountability described by a contract template to other elements critical to a full specification of a Working-Interoperability-capable solution.

Important: Although the concepts and relationships for the Solution package are contained in a separate package, these elements cannot exist independently of structural specifications.



857

858 **Figure 16: Elements of the Solutions package of the BF.**

859

Element	Description	Notes	From
Activity	Something that happens.		RM-ODP
Assign Variable	A pattern of behavior for describing system-to-system behavior.	A type of activity.	CDL
Behavioral State	Solution Specifications always have at least one state that must be expressed through sequences of activities. These states should be		HL7 ArB

Element	Description	Notes	From
	expressed with a state machine or other formalism. State transitions should be associated with Work Units, and may be associated with global variables that are shared by the community.		
Choice	A pattern of behavior for describing system-to-system behavior.	A type of activity.	CDL
Constraints	Constraints are expressed in the Behavioral Framework as pre-conditions, post-conditions, exception conditions, and invariants.		HL7 ArB
Contract Template	An agreement covering part of the collective behavior of ***n*** roles.	Other attributes of a contract template have not yet been identified, such as provenance and jurisdiction. This ties conformance levels to service level agreements found in the Solution Specification. Contracts may reference other contracts.	RM-ODP, profiled by the HL7 ArB
Interaction	An interaction takes place with the environment of an object. Interactions are the smallest unit of communication that has business value.		RM-ODP, profiled by HL7 ArB

Element	Description	Notes	From
	An interaction realizes accountability.		
No Activity	A pattern of behavior for describing system-to-system behavior.	A type of activity.	CDL
Parallel	A pattern of behavior for describing system-to-system behavior.	A type of activity.	CDL
Performed Solution	Solution Specifications can reference other Solution Specifications that, once performed, may be aggregated into a work unit.		CDL, as profiled by HL7 ArB
Sequence	A pattern of behavior for describing system-to-system behavior.	A type of activity.	CDL
Solution Specification	A thorough description of what a community does and aims for, which avoids defining how it is deployed. This description includes operation behavior and service quality levels.		CBDI, profiled by HL7 ArB
Work Unit	Collections of activities that may be composed to form solutions for a community.		CDL, profiled by HL7 ArB

860 **Table 7: The element names and definitions from the BF PSM package.**

861

4 Using the Behavioral Framework Packages

The BF provides a language for expressing the Computational viewpoint in the ECCF ("*How* things happen") in the larger context of a specification focused on enabling Working Interoperability. The BF achieves this goal by integrating – and, in some cases, synthesizing concepts drawn from the Enterprise (Why), Information (What), and Engineering (Where) viewpoints. In particular, one notes that semantic aspects of each of these viewpoints appear in the three "Role/Accountability" packages defined in Section 3.

In contrast, the use-case and business-capability scenarios primarily drive the Solutions package. It provides guidance on implementing and deploying the structures that are collected in the Role/Accountability packages. The Solutions package focuses on the contract template (design-time) and contract (run-time), and enables component developers to "assemble" a fully specified, non-ambiguous behavioral specification with a flexible approach to defining how the accountability is realized in a verifiable manner.

4.1 Contracts and Context

One of the most important aspects of the package topology presented in Section 3 is that an implementation that contains only CIM-level artifacts specification elements may be used as part of a separate, contextually broader solution that contractually binds implementations that include elements that are more specific. For example, these elements are defined at the PIM or PSM levels of the separate specification. This re-contextualization-based reuse becomes possible because ***the unit of accountability does not change from level to level, though the means by which it is achieved can be quite different.***

This flexibility is essential and highlights the question of what a standards- or specification-development organization, working in a vertical space like healthcare, can and should be focusing on (as opposed to ignoring or restricting to an internal-only focus.) For example, within HL7, V3 messages and structured documents focus on standardizing static semantics without any interference or attempts to define conformance with respect to the business process. Their structures focused mainly on static models of information content that are instantiated and transmitted at well-defined points in a business process. These structures presume little infrastructure, almost no existing architecture, and little

or no behavioral semantic sophistication on the part of the parties involved. However, each of these pieces clearly defines aspects of a client architectural context. Those aspects of the context that affect the achieving of Working Interoperability can be formalized into a BF-compliant contract that, in turn, can be formally expressed as a BF-derived contract template.

One can characterize – without judgmental denigration -- HL7's traditional static-centric, message-focused approach as "drive-by interoperability," that is, interoperability that is defined with minimal or absent run-time context. Thus, specifications need to be applicable anywhere and everywhere in an almost *ad-hoc* fashion. The interoperability context itself is independent of business processes outside of the semantics of the transaction that involve the static semantics that are specified in the transaction. In contrast, SAIF, with its emphasis on Enterprise Architecture and Working Interoperability, acknowledges that in some cases run-time context *is* important to the specification and standardization process.

In particular, large organizations or mega-enterprises (for example, Kaiser Permanente, the VA, the Military Health System, the NCI, Canada Health Infoway, and others) define their business processes, create technology to mirror that, and then expect the infrastructure to adapt to achieve Working Interoperability. Implementing specifications in that context is, therefore, a significantly different effort than working in a point-to-point environment where no substantive trust or trust facility exists.

In this large- or cross-enterprise context, the BF is the set of concepts, relationships, and associated tools that allow specification and component developers to formalize, build, and deploy components in a context in which accountability can be structurally defined or, if necessary, deferred contextually.

4.2 Solution Packages and Contract Templates

Recalling the discussion in Section 3, contract templates are patterns for defining and instantiating accountability in the context of implementations. They facilitate *exchanges of information related to shared state* and provide provable accountability along lines of role-based responsibilities. As discussed in the explanation of the Solution package models, contracts instantiated from contract templates may then be recursively represented by executable structures at the run-time

component level, thereby providing a mechanism of binding design-time requirements and constraints (the Knowledge level in the Accountability pattern) with run-time components (Operation Level).

More specifically, contract templates describe an "interoperability lifecycle" made up of three times characterized by the behaviors: establishing, enabled, and terminating (see Figure 17). Each behavior may be individually specified and referenced through the Solution Specification.

Note: No scale exists for the Interoperability Lifecycle. The establishing behavior may be implicit, legal, or syn/ack, or even explicit in another contract. This behavior allows contract templates to be created from the perspective of potential reuse and/or referenced by other contract templates.

Comment [KGS4]: Would the audience be familiar with "syn/ack", which is a TC P/IP term?

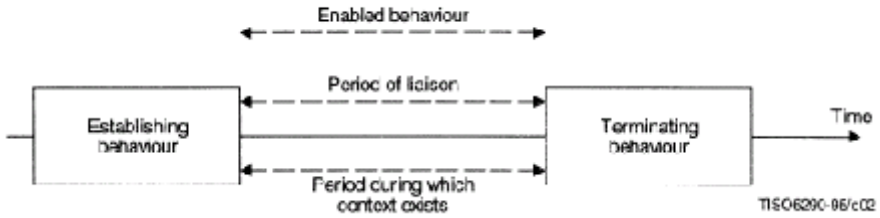


Figure 17: The "Interoperability Lifecycle" consisting of three types of behavior: establishing, enabled, and termination[6] (from ISO/IEC 10746 (RM-ODP)). The WI "context" exists for the total duration of the Interoperability Lifecycle.

Implicit in Figure 17 are two addition concepts:

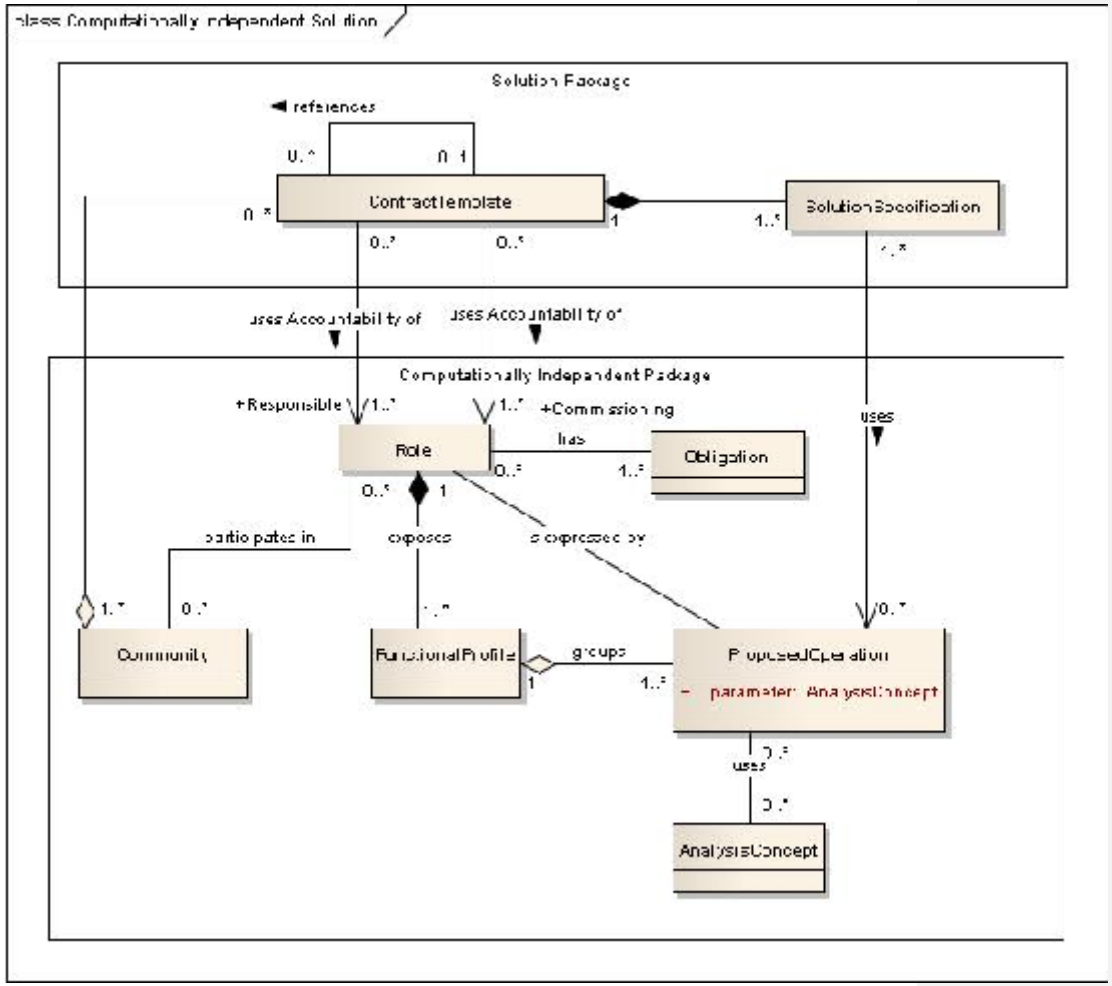
- *Contractual context*: The knowledge by a context manager that a specific contract exists between two parties at a moment in time.
- *Liaison*: The relationship between a set of objects which results from the performance of some *establishing behavior*, *that is,* ***the state of having a contractual context in common.***

[6] This diagram came from ISO/IEC 10746 (RM-ODP).

Figure 18-Figure 20 show the layered binding of the elements of the Roles/Accountability packages to the elements of the Solutions Package. *The net effect of this multilayered binding is that a solution can itself be specified as a Model-Driven Architecture (MDA)-compliant, layered construct.*

Figure 18 shows the flexibility inherent in a contract-based model for interoperability. In particular, it emphasizes the binding between contract templates and CIM-level constructs, in particular roles. Contract templates thus become the design-time "glue" that associates roles and their associated capabilities, capacities, or competencies with the specific structural aspects of a contract (through a contract template) via the semantics of commissioning and responsible agent. In particular, at the CIM level, contract templates bind roles together around Accountabilities expressed as obligations. Obligations, in turn, are manifest as being able to be fulfilled via the functional profiles exposed by a role contextualized within a community. In other words, at the CIM level, the Solution Specification that is associated with a contract template collects interactions that are within the scope of the roles involved, as defined by the roles' compositional functional profiles, interfaces, and proposed operations.

This separation allows behaviors to be contextualized by accountability, but described by specific structures.



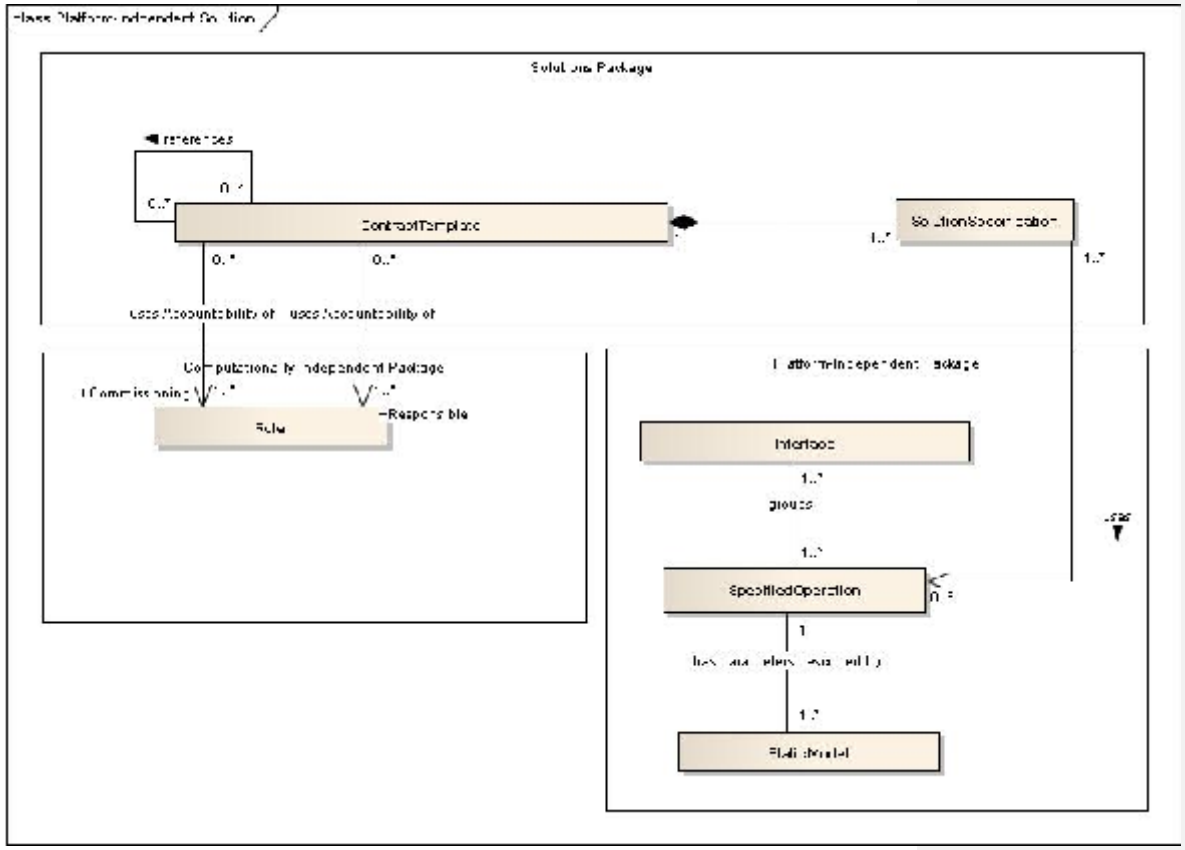
969

970 **Figure 18: CIM-level Solution model. The flexibility of the BF is most directly manifest in its**
971 **ability to describe roles which are able – but not required – to become contract participants**
972 **independent of their actual participation in contract templates and contracts.**

973 The PIM Solution model (Figure 19) continues to bind to the roles from the CIM
974 Solution model. The traceability from CIM role to PIM-level elements is implicit
975 in the contract template, but made explicit in the Platform Independent Interface
976 Specification. As with the CIM Solution model, the Solution Specification uses

977 the operations (now expressed at the PIM level) to facilitate the interactions
978 defined therein.

979



980

981 **Figure 19: PIM-level Solution model. Additional PIM-level specificity, which is, by definition,**
982 **traceable to the CIM level specification, can be bound to contract templates.**

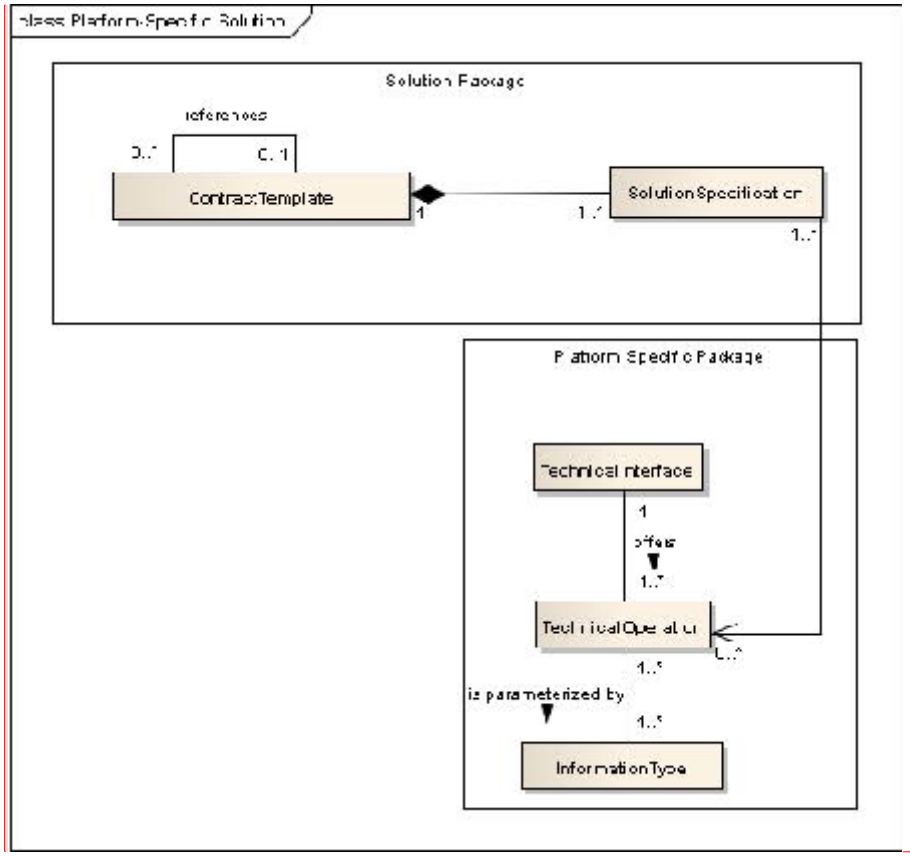


Figure 20: PSM-level Solution model. When appropriate, PSM-level specificity, which is, by definition, traceable to the CIM and PIM level specifications, can be bound to contract templates.

In summary, binding the accountability required for a contract template to a role defined at the CIM level allows the Solution Specification to be bound in a traceable manner to PIM and PSM Solution-specific structural specifications. Conformance may therefore be tested at different levels of structural refinement against a given contract template in the context of focus on a given use case.

Comment [KGS5]: Fig 20 needs a cross-reference or lead-in sentence.

5 Behavioral Patterns

Formalizing accountability in the context of contracts surfaces certain patterns in behavioral models. These behavioral patterns can be helpful when creating and implementing specifications. To date, two types of patterns have emerged from initial experience with the BF:

- Functional Patterns (see Table 8)
- A Taxonomy of Service-Oriented Encapsulations of Accountability (see Table 9)

Type	*Examples*	*BF Element*
Publishing a state transition	Admission Discharge Transfer (ADT) messages, clinical report stream	BehavioralSpecification.Event is bound to state transitions of a single focal class.
Managing a State Machine	Registries, Entity Identify Service (EIS)	Interface is bound to the Information States.
Request / Fulfilment	Orders, Referrals	Behavioral Specification, Interface
Query	Retrieval, Location, and Update Service (RLUS)	Interface is bound to StaticModel.
Publish Business Process	Initiate, Suspend, Resume, Cancel	Interface is bound to combined shared state (for example, across multiple focal classes).

Comment [(6]: Is "StaticModel" spelled correctly?

Table 8: Functional Patterns expressible at interfaces.

1003

Name	Description
Process Services	Represent virtualized business processes with reusable patterns of behavior. Often, these processes represent realized sets of business rules upon which an organization has agreed. They are generally not concerned with the *states of domain entities* other than how they affect the *state of the process*. They tend to be coarsely granulated, limiting the number of external calls made to enhance performance and to allow for the specific business process to be appropriately scoped. By definition, they are usually "stateful" services (which may be implemented in several ways).
Capability Services	Represent a unified, contiguous set of functions that expose a set of cohesive business functionality explicitly and unambiguously. In general, they are concerned with business focal classes (domain classes) and their state transitions. The core business logic around these focal classes is virtualized behind a Capability Service's interface.
Core Services	Expose sets of information. The functional profiles of the service are generally not focused on the state of the underlying information or in the trigger events that modify the state of that information. They often are focused vertically along the line of business - typically along the lines of an information profile (for example, a RIM-based patient class, a Clinical Document Architecture (CDA)-based Continuity of Care (CCD) profile).
Utility Services	Provide supporting services that are still along the lines of business (as opposed to technology focused), but are not necessarily focused on particular information profiles or business classes or processes. Examples include areas such as Eligibility, Referral, Terminology, Template Management, and Anonymization.

Infrastructure Services	Provide collections of functionality that is technology focused. In general, Infrastructure Services should not encompass business or process logic, or virtualize key domain concepts, but *should* expose reusable technical functionality (an e-mail service, for example).

Table 9: Taxonomy of Service-Oriented encapsulations based on Accountability types, which may be encapsulated behind an interface. The taxonomy suggests certain types of supporting infrastructure including security models, trust patterns, and so on.

Figure 21 shows a different representation of the same semantics as Table 8 does. Figure 21 gives a sample deployment topology, which effectively overlays the behavior patterns that are expressible as interfaces (Table 8) with the taxonomy described in Table 9. As a result, the graphic suggests some conclusions regarding the mapping between business process and the details of a particular solution. For example, a V3 message could be defined to realize the requirements of a particular business process and would be classified in the taxonomy as a Capability Service. Constructs defined to satisfy particular business rules for certain types of trading partners would be considered Process Services, on the other hand.

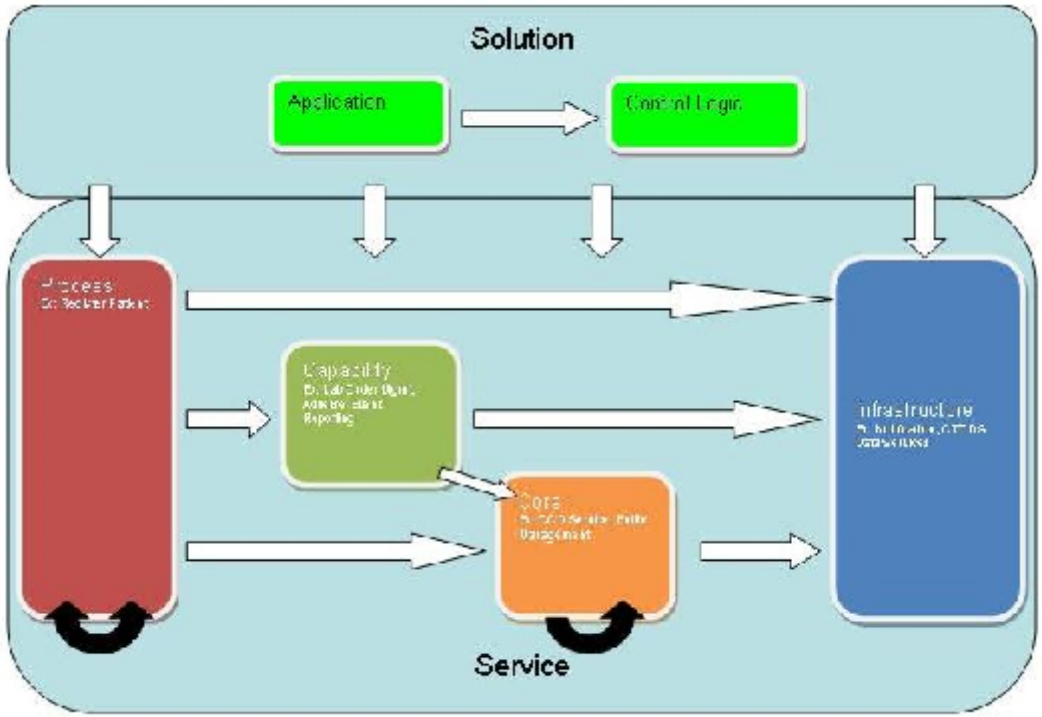


Figure 21: Behavior Pattern of a standard Service Taxonomy as manifest in a sample deployment. Note that the taxonomy is often presented in a layered "vertical" form that is semantically identical to the above graphic.

6 Appendix A: The BF and the HL7 Legacy Dynamic Model

Comment [KGS7]: This term isn't used consistently. Should it be HL7 Legacy Dynamic Model, or legacy HL7 Dynamic Model, or HL7 V3 Dynamic Model? It's ok to use a shortened version of the term, such as Dynamic Model after the first introduction, though.

From the onset of work on the BF, the BF was required to subsume the HL7 Legacy Dynamic Model. The Dynamic Model was to be used as a minimal set of requirements for the BF. Early analysis revealed that the Dynamic Model defined a *context-free* notion of behavior in which interoperability is specified with loose coupling to underlying business process. As is evident from the previous discussions, the BF adds considerable context to behavior semantics. In fact, the BF formally subsumes the Dynamic Model. Figure 22 shows a model of the essential concepts and relationships of the legacy Dynamic Model. The concepts

that are generated from the model documentation are formally defined in Table 10, which presents the concept-by-concept mapping of the Dynamic Model concepts to the SAIF BF.

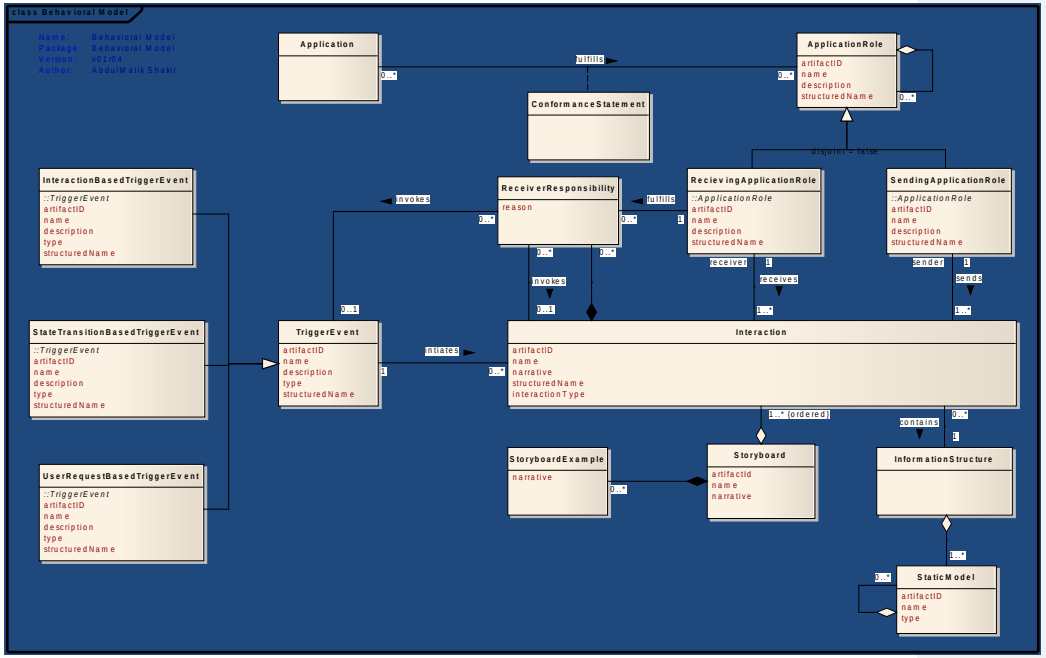


Figure 22: HL7 Legacy Dynamic Model.

HL7 Legacy Dynamic Model maps to…	SAIF Behavior Framework
Interaction	Interactions, exchanges, and choreographies in a Solution Specification
Application Role	Role bound to interface to realize role's behavior
Receiver Responsibility	Solution Specification, Shared State, Accountability, Obligation, Interface

Trigger Event	Events in Solution Specification, Behavioral State
Information Structure	Static Model
Storyboard	Solution Specification, contract
Application	Components playing a role by implementing an Interface

Table 10: Concept-by-concept mapping of HL7 Legacy Dynamic Model to SAIF BF.

As Figure 23 and Figure 24 show, the core elements of the HL7 Legacy Dynamic Model can be separated into two packages and placed within the BF. Figure 23 shows the elements, which can be represented within the BF Solution package. Figure 24 shows those elements in a Structure package, which contains elements predominately from the PIM Structure package of the BF.

Note: For clarity of presentation, a single graphic is shown. Also, note that the Dynamic Model elements are not fully normative within HL7.

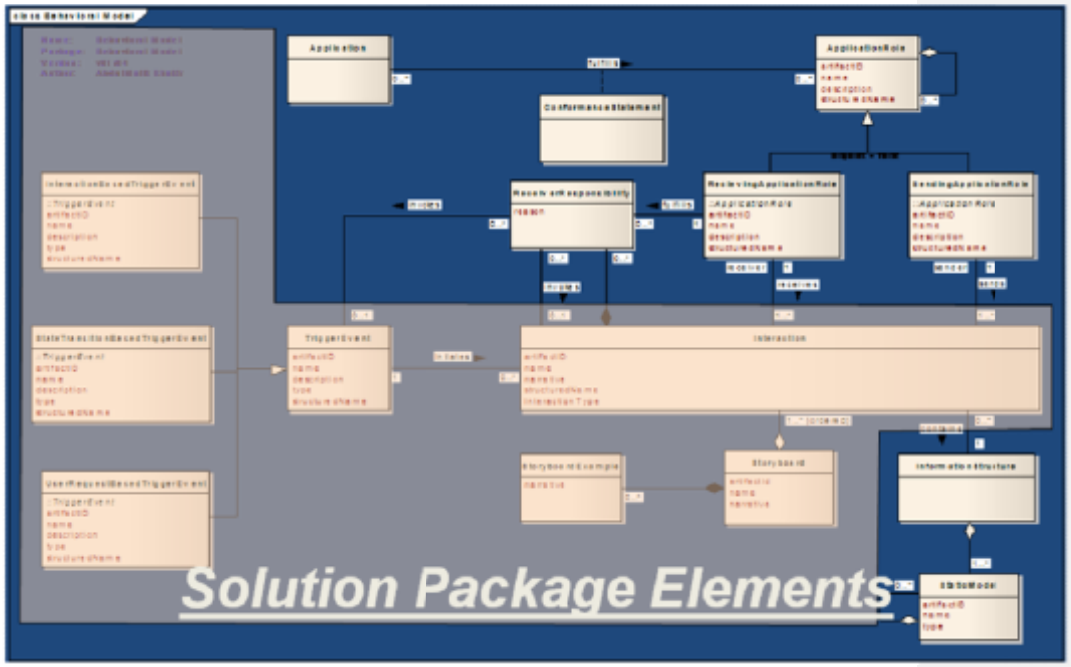
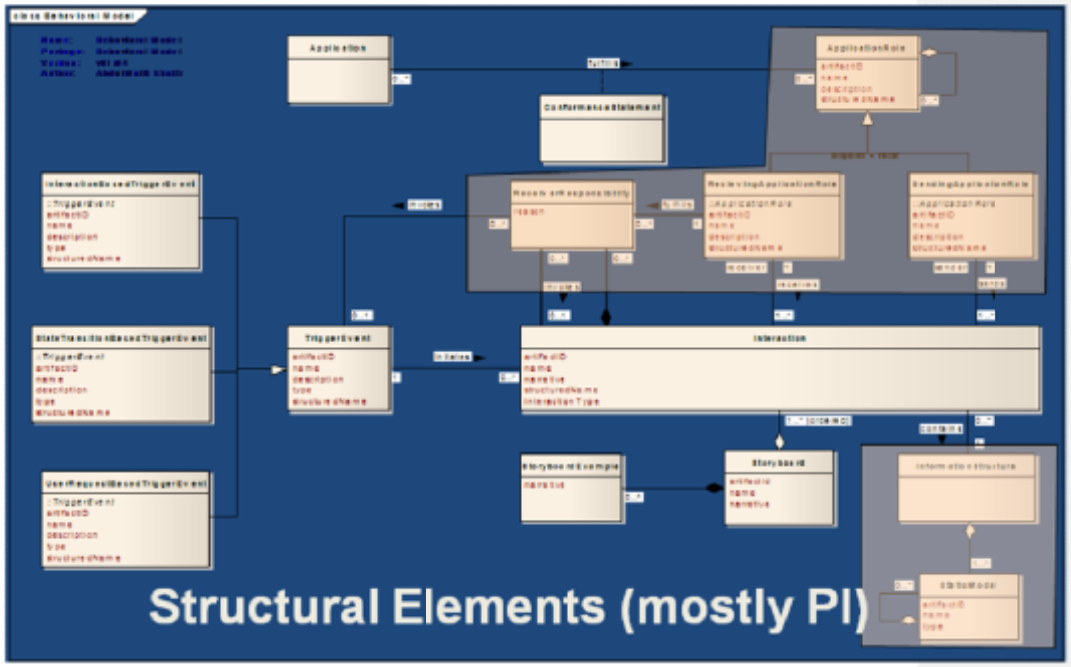


Figure 23: HL7 Dynamic Model, which is subsumed by the BF Solution Package model.



1046

1047 **Figure 24: Remaining elements of Dynamic Model, which are subsumed by a composite**
1048 **Structure package.**

1049

7 Appendix B: RIM-Based Services and V3 (RIM-Based) Messages

The essence of much of the discrepancy between the RIM-based service and V3 message worlds centers on that the V3 message universe is essentially context-free and an event-driven. As was stated in the discussion of the HL7 Legacy Dynamic Model, an overarching requirement of the BF was that it support traditional HL7 message-based, context-free ("drive-by") interoperability. This interoperability environment is characterized by little or no formal trust fabric and minimal true coordination of functionality across systems. However, HL7 message-based interoperability is formally triggered by business-process-level state changes; a somewhat difficult construct to manage given the context-free nature of the interoperability specifications themselves (for example, messages). In addition, as the complexity of component-to-component interactions increases, and additional behavioral semantics emerge as critical to being able to achieve Working Interoperability between two trading partners in a particular business context, the management becomes increasingly difficult in a messaging paradigm.

In contrast, the management remains relatively tractable in a service paradigm where one can assume the presence of a trust fabrics, shared information and technical infrastructures, and coordinated business processes. The BF aims to support both paradigms. In such an environment, documented specifications, which make explicit at design-time the myriad of assumptions that must ultimately be manifest at run-time, provide the key mechanism to identify in a predictable, scalable, tractable manner where the intersection between "vertical standard and horizontal deployed architecture" exists, i.e. provide the path of most predictability and minimal cost to achieving Working Interoperability.

Looking more closely at some of the substantive constructs and assumptions of both the RIM-based services and V3 RIM-based message approaches to Working Interoperability, the following general, but informative observations can be made:

- In the V3 messaging paradigm, arbitrary activities occur and result in the fact that computable structures are required to change state, an event that results in one or more messages being sent or received.

- In the RIM-based services paradigm, events are typically thought of as being more deterministic, procedural, and sequential in nature (for example, Request / Response).

Conceptually, therefore, event-driven and deterministic, procedural, and sequential views of the interoperability universe can be viewed as complementary rather than competing, inconsistent, or otherwise incompatible interoperability contexts.

Figure 25 (RIM-based service paradigm) and Figure 26 (V3 message paradigm) highlight the differences between the two paradigms from the perspective of the cardinalities of the relationships between the BF Solutions and Structures elements. Table 11 presents the differences between the two paradigms with color codes that refer to the figures:

- Green = Fully-specified construct
- Orange = Partially or underspecified construct (from a SAIF perspective)

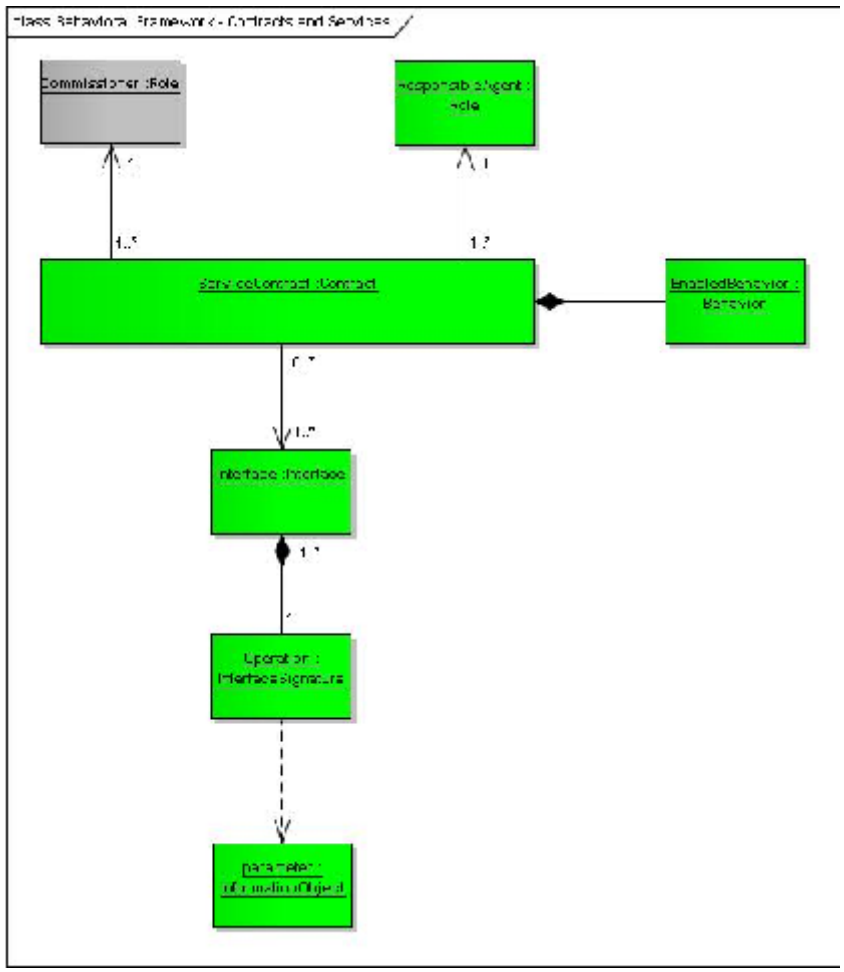


Figure 25: Cardinalities of the relationships between Solutions and Structures in a RIM-based Services paradigm.

Services are, in other words, a common but special case of contract templates where:

- Services define the responsible agent as a durable, reusable structure.
- Each commissioning agent is the same, so they can be virtualized.
- The enabled behavior is based on the interface specification.

Some of the consequences of these special circumstances are detailed in behavioral patterns (Section 5), as services allow ways for these patterns to be realized in technological artifacts.

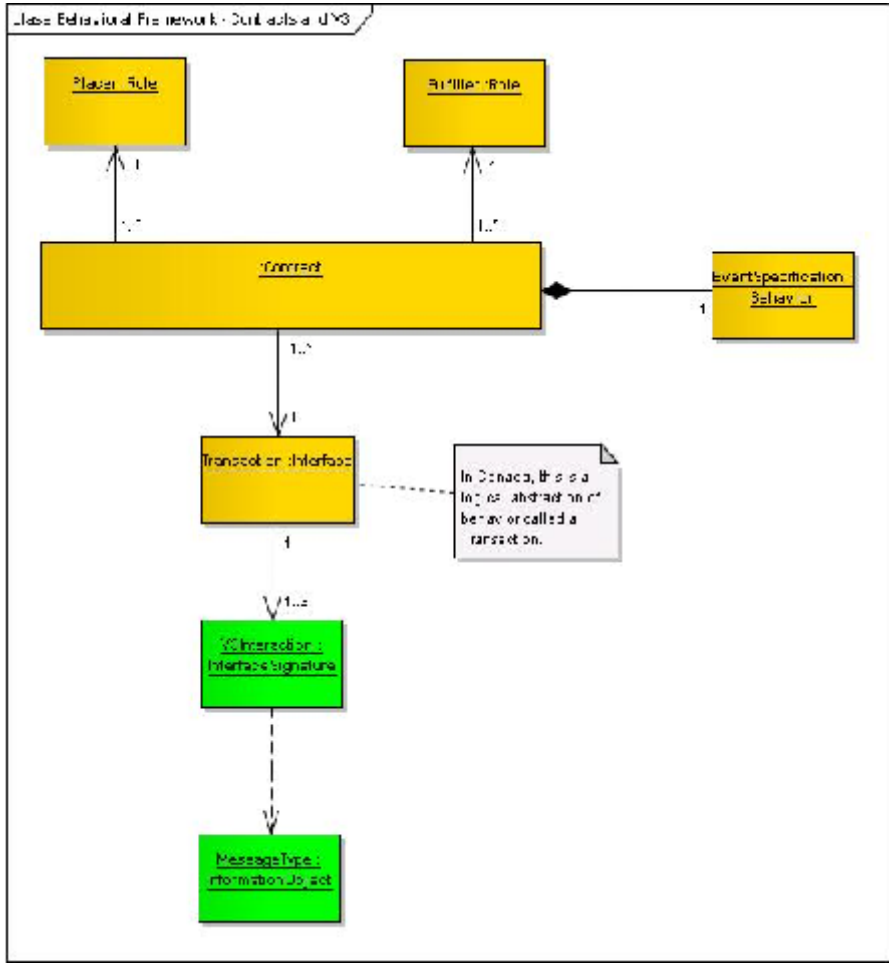


Figure 26: Cardinalities of the relationships between solutions and structures in a V3 messaging paradigm. Application roles are partially specified, in that their obligations are scoped to a particular interaction. No explicit contract template or abstraction of behavior exists (although Canada Health Infoway has attempted to address this issue by defining a V3 transaction).

Element	RIM-based Services	V3 (RIM-based) Messaging
Roles	Defined in terms of obligations that are apparent at the interface.	Loosely coupled to responsibilities.
Interfaces	Computable abstraction closely tied to role.	Assembled to support known processes.
Signatures	Operations that support accountability.	Interactions that support accountability.
Behavior Specifications	Virtualize subset of behaviors that characterize the role.	Event driven, tied to triggers.
Interactions	Services consistently play the Responsible agents.	V3 message senders are Commissioning agents.
Information Objects	Well defined, bound to behavior.	Well defined, bound to responsibility.
Contracts	Holds context, may compose accountability.	Context-free, notions of responsibilities.

Table 11: Comparison of perspectives and approaches of two interoperability paradigms -- RIM-based services and V3 messages

Green classes indicate that the interoperability paradigm operates in a "pure/native, non-SAIF manner" and fully specifies the concept to a degree sufficient for use in the larger, multi-paradigm SAIF context.

Comment [KGS8]: I made your requested changes to this table; however this table uses a different style than all the other tables in this document. In DITA, the style sheet controls the style of elements such as tables, and by default, all tables have plain white cells with a light tan header. If it's critically important that this table have the green and orange (or yellow) highlighting, I could set up a special property to do that. Let me know. <CNM> Color not important – just contrasts . Please do what is easy and standard in other tables </CNM>

<KGS> In the DITA document, I will use different font styles (italic, bold, monospace) to represent the gray, green, and orange cells. </KGS>

Orange classes are, in their native context, underspecified from a SAIF perspective.

Grey classes are either virtualized or ignored by the native paradigm. Concepts shown in grey are either virtualized or simply ignored in a services paradigm, something which is NOT possible in a messaging paradigm where all concepts must be determined in a run-time context because of the loosely-coupled context and the mechanisms of specification (so-called "drive-by interoperability" in this discussion).

A concept-by-concept comparison is as follows:

Roles are well defined in the service paradigm by identifying roles via business analysis. In contrast, in the messaging paradigm, roles are defined only at a system level, e.g. Application Roles, and are not directly traceable to a business context.

Interfaces are fully specified in a services environment, but underspecified in the V3 paradigm (if they are specified at all) in the sense of an interface being defined as "an abstraction of expected behavior."

Signatures are fully specified in both paradigms.

Behavior Specifications are fully specified – by definition and necessity – in a services paradigm, because a service is a set of behavioral actions that are bound together at an interface that supports a cohesive and coherent set of actions. Behavior is underspecified in the messaging paradigm because behavior can only be specified on an interaction-by-interaction basis, i.e. a collective set of "responsibilities" that are associated with a single role is difficult and of extremely fine grain size when compared to business processes.

Interactions are well specified in both native paradigms.

Information objects are well specified in both native paradigms.

Contracts are the core of the single biggest difference between the service and

messaging paradigms. In the services paradigm, service interface collect operations as functional profiles, which express specific business goals and patterns of usage and are correspondingly bound to semantic profiles. The formalism of the specification allows a number of constructs – for example, commissioning and responsible agents – to always be assumed to be the same in a service invocation.

In contrast, the messaging paradigm – by being context-free – is forced to finely granulate behavior and thereby force individual systems to form one-off, run-time-specific collections of behaviors to fulfill larger business goals or patterns.

In summary, RIM-based services are coarsely granulated, grounded in contracts, and deeply context-dependent. In contrast, V3 RIM-based messages are finely granulated, based on partial events, and largely context-free. These core differences result in difference approaches to the representation and specification of key concepts as noted in the table above.

SAIF and its BF are designed to support both interoperability paradigms. Clearly, however, the choice of interoperability paradigm has implications with respect to a number of factors including existence of trust fabric, complexity of interactions, and other critical considerations, which collectively determine how an enterprise models, organizes, and deploys its resources.

8 Appendix C: References

The following links point to two different Behavioral Framework documents.

The HL7 Behavioral Framework is published at:
http://www.ncientarch.info/hl7_bf/hl7_bf/

The generalized Behavioral Framework, including mappings to the RM-ODP is published at: http://www.ncientarch.info/hl7_bf/general_bf/

Comment [KGS9]: Do these two links point to TWO different Behavioral Framework documents?
<CNM>
yes, I believe so.
</CNM>

Comment [CT10]: Ended review at about 2:15 PM ET on 12/21/2009 – did not see anything I felt the need to change (nor any question that I could quickly answer). Just a general observation: I think that this chapter in the future can be rewritten in a much more concise manner. Since this is an overview document, a lot of the material could be pushed down into a more technical document. Maybe this will happen during/after the Alpha testing. Overall, one again, a much improved document when compared to some of the earlier work
<CNM>
There are probably a couple of documents living in this document, e.g. sections 1 and 2 could be separated from 3 et al
</CNM> KGS – Yes, Sections 1 and 2 are high level while Sections 3-5 and the appendixes are very technical. Would implementers of the BF need to know about the Legacy Dynamic model and RIM-based messages and services?