Integration Framework

Business Services (Contracts) User Guidelines

GB942U

TM Forum Approved Version 3.0

May, 2011

Notice

This material, including documents, code and models, has been through review cycles; however, due to the inherent complexity in the design and implementation of software and systems, no liability is accepted for any errors or omissions or for consequences of any use made of this material.

Under no circumstances will the TM Forum be liable for direct or indirect damages or any costs or losses resulting from the use of this specification. The risk of designing and implementing products in accordance with this specification is borne solely by the user of this specification.

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

Table of Contents

List of Figures

Executive Summary

This guidebook produced by the Architecture Harmonization (AH) team, focuses on guidance for using Business Services (aka Contracts), such as granularity guidelines, implementing Business Services, and Tooling for Business Services and other TM Forum Frameworx. The intended audience for this guide book is IT managers, architects and developers. It is recognized that due to their respective needs some sections of this book are more technical than others. This guide book is also intended to be used in conjunction with the open-source tooling software being developed.

Business Services extend Service Oriented Architecture (SOA) to recognize that there is a relationship between specific client and server systems in the application environment. This relationship may be short-lived or long-lived based on the specification for a given Business Services. The relationship may be implemented via point-to-point interactions or via an integration framework, such as hub-and-spoke or an Enterprise Server Bus (ESB). These systems are defined by the Application Framework (aka TAM). SOA is also extended with granularity guidelines that derive from Business Framework (aka eTOM) and finally the information model that is implemented in Business Services is compliant with the Information Framework (aka SID).

We expect that TM Forum will manage real world, pragmatic Business Services based on the definition presented here using tooling so that the standards can be conformance-managed and change-controlled and so that the standards can be applied directly to integrations. The current Business Service meta-model is implemented in the Eclipse tooling as a standard Ecore meta-model.

1. Business Services Granularity Guidelines

It is a given that a Business Service implements zero (utility centric) or more business entities, one or more business use cases. With only this guidance a Business Services could be quite large or quite small; a Business Services could act on a large and possibly diverse set of business entities. This chapter provides additional guidelines to assist those involved in Business Services development to right-size their Business Services.

These guidelines attempt to ensure that some degree of consistency of Business Services scope exists across the many groups and individuals who will be defining and developing Business Services. The guidelines included here are just that and should not be construed as rules or dictates.

1.1. Granularity Guidelines

With the relationship established between Business Services, use cases, and Business Process Framework processes, this relationship can be used to define Business Services granularity guidelines in terms of use case granularity guidelines and Business Process Framework process granularity guidelines. This is in addition to guidelines specific to Business Services granularity presented in this section.

One way to define the granularity of a Business Services is to reflect the granularity of the Tasks it specifies so that the granularity reflects the business purpose of the Business Services. So a fine-grained Business Services will have use-cases, or Tasks, that are themselves fine-grained, implying that the purpose deals with detailed strategic or operational tasks or detailed data entities. For example, it is natural to think of a Business Services that controls the configuration of a device to be fine-grained, whereas a Business Services to manage an end-to-end customer service would be considered coarse-grained. However, both these Business Services are fulfilling a perfectly legitimate and necessary business function. So it is more useful to use the term 'right-grained' for a well formed Business Services, where the Tasks have the correct granularity for the business purpose being conducted, irrespective of whether the purpose is detailed, or highly aggregated.

Of course, another way of describing the granularity of a Business Services is the number of use cases it implements. A fine-grained Business Services would implement one use case; a larger grained Business Services would implement two or more use cases. In this situation, the fine-grained Business Services simply provides additional features to the use-case it implements, which is closely related to the concept of a SOA Business Services, which embellishes an SOA Service. However, the large grained Business Services supports more lifecycle states of an entity, because the various Tasks it implements may be triggered at different times over the life of the entity, as and when the Tasks trigger preconditions are realised. This leaves two questions.

To facilitate interoperability and re-use, the Business Services scope, the use cases implemented or services offered by a Business Services should exhibit certain properties. One of these properties is the granularity of a Business Services. Business Services should be of the "just about right" granularity. The guidelines provided here help ensure that Business Services exhibit the right level of granularity.

The following bulleted items provide a synopsis of the guidelines. The synopsis is followed by a more complete description including examples.

A Business Services

- Should not contain business process management logic
- Should not represent a use case that involves multiple actors
- Should exhibit preferred degrees of cohesion and coupling

Should not include processes/use cases which create instances of entities that cross Business Process Framework boundaries

The business purpose is not a high level aggregation, as found in L2 and above, but is detailed enough to be specific to each Business Services. The guideline here is that the business purpose of a task centric Business Services should have a granularity equivalent to Level 3+ Business Process Framework processes. This is not a direct equivalence, as a Business Services is not a process element, but a standards artifact that specifies interface behavior. Nonetheless, there is equivalence between the granularity of an L3+ Business Process Framework process and the interface specifications that are used to implement the process and this should be sufficient to set a unique business purpose for each Business Services.

1.1.1. Business Process Management Logic in a Business Services

A Business Services should not have business process management embedded within the Business Services. If this is the case then the Business Services is too large and should be decomposed into smaller Business Services. In fact the business process is correctly implemented in the client system, where its behavior and the processes that it supports are fully specified in one place, supported by system documentation that describes how the business processes are supported. This enables the ISV to differentiate their product offering against competitors. If processes were embedded in Business Services, Contacts could become arbitrarily complex, which would complicate the overall solution, prevent vendor innovation and undermine the purpose of the Business Services: to be a clean hand-off point between systems.

For example, suppose there is a candidate Business Services called 'provision DSL service'. Provision DSL service consists of qualifying the DSL service, possibly installing equipment or allowing the customer to install the equipment, designing the service, and activating the service. If the provider offering the service does not "own" the local loop to the customer, then a request must be sent to the provider that does own the local loop to see if the distance requirements to a DSLAM are met. If not, then the service cannot be provisioned. If the customer would like the "self-service" package, then the DSL modem is sent to the customer. Otherwise, a technician must be dispatched to install the modem. This example will be used throughout this section.

Figure 0-1 - Provision DSL Service Flow depicts this Business Services and the logic embedded within it.

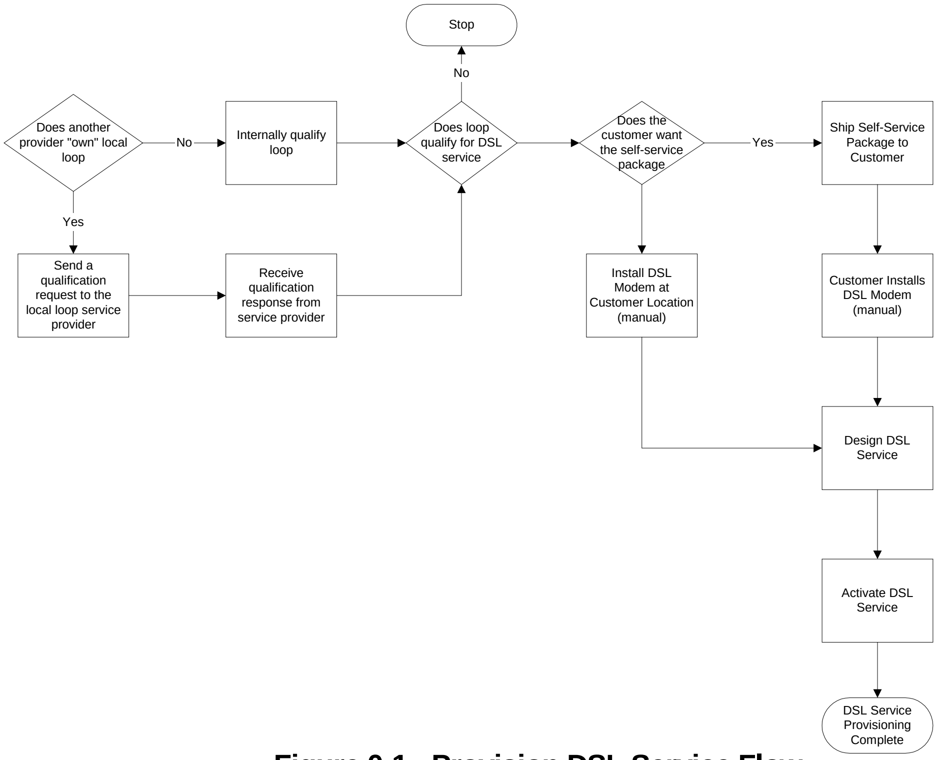

Stop
No
Does another provider "own" local loop
No
Internally qualify loop
Does loop qualify for DSL service
Does the customer want the self-service package
Yes
Ship Self-Service Package to Customer
Yes
Send a qualification request to the local loop service provider
Receive qualification response from service provider
Install DSL Modem at Customer Location (manual)
Customer Installs DSL Modem (manual)
Design DSL Service
Activate DSL Service
DSL Service Provisioning Complete

Figure 0-1 - Provision DSL Service Flow

Figure 0-1 - Provision DSL Service Flow definitely exposes the fact that this Business Services contains business process logic and also manual processes embedded within it. The business process logic is contained in the diamond-shaped decision points. The manual activities are annotated as such in the figure. Based on this, the Business Services should be decomposed into a number of subordinate Business Services. Each box in the figure is a Business Services candidate.

One other note regarding this Business Services is that during its execution it can terminate without completing all of its actions. This occurs when the DSL service requested does not qualify for service. This termination is illustrative of not following cohesion and coupling guidelines provided in a later section of this document. However, it is possible for a Business Services to terminate having failed to provide the desired business purpose. This failed state would still be a legal post condition.

1.1.2. Use Cases with Multiple Actors

Use Cases are effective requirements gathering tools. However, they can depict business scenarios that are quite large. Therefore, it is not sufficient to assume that every Use Case makes a good Business Services candidate. With this in mind, a simple guideline that helps ensure that Business Services are of the correct scope is to draw a Use Case Diagram for the Use Case. If multiple actors are involved in the Use Case, the resultant Business Services based on the Use Case may be too large.

Figure 0-2 - Provision DSL Service Use Case Diagram depicts the Provision DSL Service Business Services candidate shown in Figure 0-1 - Provision DSL Service Flow in use case format.

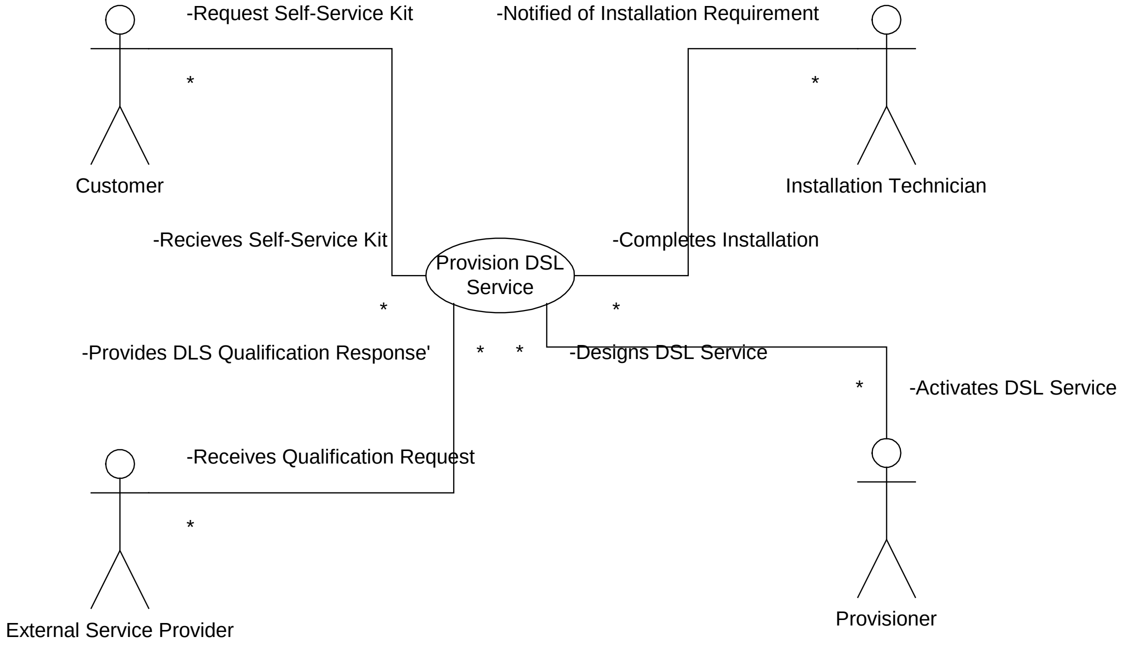

-Request Self-Service Kit
-Notified of Installation Requirement
*
*
Customer
Installation Technician
-Recieves Self-Service Kit
Provision DSL Service
-Completes Installation
*
*
-Provides DLS Qualification Response'
*
*
-Designs DSL Service
*
-Activates DSL Service
-Receives Qualification Request
*
External Service Provider
Provisioner

Figure 0-2 - Provision DSL Service Use Case Diagram

This rather large Use Case can be represented by a number of subordinate Use Cases, each involving a single actor. One of the diagrams representing a subordinate Use Case is shown in Figure 0-3 - Self-Service DSL Installation Use Case Diagram. Self-Service DSL Installation represents a more preferred granularity for a Business Services.

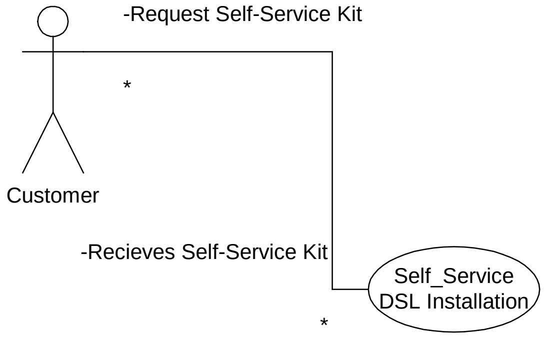

-Request Self-Service Kit
*
Customer
-Recieves Self-Service Kit
Self_Service DSL Installation
*

Figure 0-3 - Self-Service DSL Installation Use Case Diagram

The Use Case Diagram below depicts the request/response Business Services represented by the two business activities; Send a Qualification Request to the Local Loop Service Provider and Receive Qualification Response from Service Provider, mentioned earlier in this section. This is an example of a single use case supporting two Business Process Framework low level (L4 and below) processes.

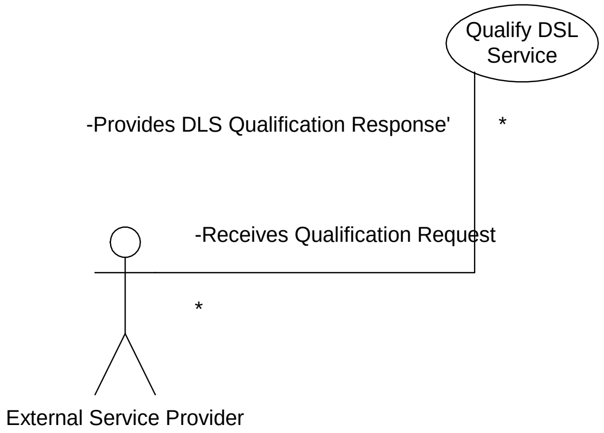

Qualify DSL Service
-Provides DLS Qualification Response'
*
-Receives Qualification Request
*
External Service Provider

Figure 0-4 - Qualify DSL Service Use Case Diagram

By developing Business Services based on Use Cases that involve only one actor, an appropriate level of scope and granularity can be achieved.

1.1.3. Degrees of Cohesion and Coupling

The concepts of cohesion and coupling can also be employed to decide whether a candidate Business Services and/or use cases are of the “just about right” size. This is a key concept from a Business Process Framework decomposition perspective, a use case decomposition perspective, and a SOA service perspective.

These concepts have been applied for a number of years to aid in the decomposition of business activities and also as an aid in modular software design. A quick review of the concepts is contained in the following paragraphs. Both cohesion and coupling concepts can be applied to Business Services and/or use cases. Business Services is the term used throughout the remainder of this section.

In computer programming, cohesion is a measure of how strongly-related and focused the various responsibilities of a software module are. Cohesion is an ordinal type of measurement and is usually expressed as "high cohesion" or "low cohesion" when being discussed. Modules with high cohesion tend to be preferable because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability whereas low cohesion is associated with undesirable traits such as being difficult to maintain, difficult to test, difficult to reuse, and even difficult to understand.

Cohesion is often contrasted with coupling, a different concept. Nonetheless high cohesion often correlates with loose coupling, and vice versa. The software quality metrics of coupling and cohesion were defined by Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine from source code analysis they conducted on several programming projects while at IBM, all in an effort to identify the characteristics of "good" programming practices.

Cohesion is a measure of how strongly-related and focused the responsibilities of a single class are. In object-oriented programming, if the methods that serve the given class tend to be similar in many aspects the class is said to have high cohesion. In a highly-cohesive system, code readability and the likelihood of reuse is increased, while complexity is kept manageable.

Cohesion is decreased if:

- The responsibilities (methods) of a class have little in common.
- Methods carry out many varied activities, often using coarsely-grained or unrelated sets of data.

Disadvantages of low cohesion (or "weak cohesion") are:

- Increased difficulty in understanding modules.
- Increased difficulty in maintaining a system, because logical changes in the domain affect multiple modules, and because changes in one module require changes in related modules.
- Increased difficulty in reusing a module because most applications won't need the random set of operations provided by a module.

In computer science, coupling or dependency is the degree to which each program module relies on each one of the other modules.

Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa. The software quality metrics of coupling and cohesion were invented by Larry Constantine, original developer of Structured Design (see also Structured Systems Analysis and Design Method **-** SSADM).

Coupling can be "low" (also "loose" and "weak") or "high" (also "tight" and "strong"). Low coupling refers to a relationship in which one module interacts with another module through a stable interface and does not need to be concerned with the other module's internal implementation (Information Hiding)[1]. With low coupling, a change in one module will not require a change in the implementation of another module. Low coupling is often a sign of a well-structured computer system, and when combined with high cohesion, supports the general goals of high readability and maintainability.

Systems that do not exhibit low coupling might experience the following developmental difficulties:

- Change in one module forces a ripple of changes in other modules.
- Modules are difficult to understand in isolation.
- Modules are difficult to reuse or test because dependent modules must be included.

The concept of coupling is usually related to the concept of cohesion: low coupling facilitates high cohesion, and vice versa. For example, one approach to increasing cohesion is functional design, which seeks to limit the responsibilities of modules

[1] If code module X doesn't really need to know something about how code module Y works, don't make it known. Then, if part of Y changes, you don't have to go back and change X. Many examples can be found on the internet.
e.g. http://www.stevemcconnell.com/ieeesoftware/bp02.htm : **Object-oriented Software Construction 2nd Edition –** Bertrand Meyer, has a very thorough coverage of this and other O-O design concepts plus design by Contract using pre, post conditions and invariants

along functionally-related boundaries. Modules with single responsibilities tend to communicate less with other modules, which typically causes the side-effect of reduced coupling. This can also be seen in object-oriented programming, where coherence is said to increase when classes are re-factored to contain more closely-related code. This tends to cause the connections between the classes to become less dependent on their internal implementations, which results in reduced coupling.

Low coupling may also reduce performance, and a highly-coupled system is sometimes desirable to achieve maximum efficiency. Regardless, in many modern computing systems, the cost of reduced performance is often seen as a worthy trade for the benefits to the software development process that result from low coupling.

From a Business Services perspective, cohesion is the degree of interaction among the parts that make up a Business Services. It is the glue that holds the Business Services together. The degree of cohesion that a Business Services possesses should be maximized.

The types of cohesion from highest (best) degree to the lowest (worst) degree are

- Functional Cohesion – a Business Services performs *exactly* one action or achieves a single goal
- Informational Cohesion – a Business Service performs a number of actions, each action with its own entry point, each action independent from the others, all performed on the same data structure
- Communicational Cohesion – a Business Service performs a series of actions related by the sequence of steps to be followed, *and also* all actions are performed on the same data
- Procedural Cohesion – a Business Service performs a series of actions related by the sequence of the steps to be followed by the Business Services
- Temporal Cohesion – a Business Service performs a series of actions related by time
- Logical Cohesion – a Business Service performs a series of related actions, one of which is selected by the calling Business Services or business process
- Coincidental Cohesion – a Business Service performs multiple unrelated actions, or actions cannot be defined

The first bullet implies the Business Services implements a single use-case, Task, SOA service, or operation. The second bullet also indicates a good level of cohesion for Business Services and allows the Business Services to provide value to the solution specification by implementing a number of Tasks required to deliver the business purpose of the Business Services. An example might be a ‘Create and Resolve Customer Problem’ Business Service with the CRM as the client and Service Management as the provider. This will specify the set of Tasks required to ensure the customer problem is managed correctly and will have a clear subject (the Customer Problem) and a clear and delimited business purpose. However, real-world experience and expertise will help confirm the granularity guidelines presented here.

From a Business Service perspective, coupling is the degree of interaction among Business Services or business process logic invoking the Business Services. It is the wiring that connects the Business Services with other Business Services and/or business process logic. The degree of coupling, if any, should be minimized.

The types of coupling from lowest (best) degree to the highest (worst) degree are

- Data Coupling – all inputs/outputs are homogenous data types
- Stamp Coupling – inputs/outputs include complex data structures, but a Business Service operates only on some of the pieces during one execution of the Business Services
- Control Coupling – a control element is passed to a Business Service, that is, one Business Service or business process explicitly controls the logic of another
- Common Coupling – multiple Business Services have access to the same global data
- Content Coupling – one Business Service directly references the contents of another

Note that the systems related by Business Services themselves have access to all forms of coupling in order to execute their operations; the Business Service itself should be limited to the lowest degree of coupling. However, lower levels of cohesion result in higher levels of coupling. For example, a Business Service that implements multiple use cases may not exhibit data coupling, but stamp coupling instead.

The Provision DSL Service candidate Business Service example used in this section does not exhibit the best degrees of either cohesion or coupling.

From a cohesion perspective, the Business Service does not exhibit functional, informational, or communicational cohesion. It does appear to exhibit procedural cohesion, not a very desirable degree of cohesion. It does not exhibit functional cohesion because it performs a series of actions. It does not exhibit informational cohesion because its parts act on different sets of data. For example, the qualify loop set of actions operates on the customers address, while the self-service installation operates on a DSL modem. It does not exhibit communicational cohesion because while the actions represent a series of steps, the actions do not operate on the same data. Having separate Business Services for Qualify DSL Service and Self-Service DSL Installation results in functional cohesion and data coupling, the best forms of both.

From a coupling perspective, the Business Service does not exhibit data coupling, but it may exhibit stamp coupling and control coupling. It does not exhibit data coupling because the parameters are not homogeneous. The parameters represent a mix of somewhat related data. It may exhibit stamp coupling because the Business Services' inputs/outputs do represent a complex data structure, ranging from information about the customer, such as address and so forth, to activation parameters about the service to be provisioned. The Business Service actions operate on different sets of this data. Control coupling is evident in the fact that the business process logic controls the logic executed within the Business Service. For example, during one execution of the Provision DSL Service Business Service, the business logic must specify that loop qualification should take place.

Since the Business Service does not represent the better degrees of cohesion and coupling, it is a candidate for further decomposition.

While each Business Service should strive to achieve functional cohesion and data coupling (if necessary), a lower degree is acceptable. Functional cohesion may lead to Business Services that are too granular which means that informational cohesion, such as that associated with an information services Business Service that retrieves data such as different groups of customer data, and data coupling are adequate for a Business Service.

1.1.4. Creating Instances of Entities that Cross Business Process Framework boundaries

This guideline moves away from looking at the Business Service itself and focuses on the use cases/processes and instances of data created during their execution. The Business Process Framework domain framework boundaries are used in this guideline as an example. However, Business Services can be organized within any type of framework, such as the Applications Framework (Application Framework). These guidelines apply to any framework that represents Business Services organization.

From a basic standpoint, the Business Process Framework Level 1 can be viewed as one that contains, but is not limited to, first level Business Services organized by customer, service, and resource. There are further subdivisions of the Business Process Framework, but this first layer will suffice to show that the Provision DSL Service candidate Business Service violates this guideline.

The Provision DSL Service candidate Business Service creates instances of data across all first level framework members, customer, service, and resource. The Business Service creates an instance of a request to another service provider for loop qualification. This data resides in the Supplier/Partner domain within the framework. The Business Service creates instances of the service's design, which resides in the Service domain of the framework. The Business Service also creates instances activation parameters that also reside in the Service domain of the framework. This appears to be a major violation of this guideline, since the Business Service creates instances of data that cross the first level of the framework. Therefore, the Business Service should be further decomposed.

Creation of data across framework boundaries can proceed from the first level of the framework to subsequent levels. Any Business Service that violates the guideline at the upper levels of the framework is a definite candidate for further decomposition.

2. Implementing Business Services

So far this document has described the Business Services from a business and system view and provides the TM Forum members with the ability to create TM Forum standard conformant Business Services from these two perspectives.

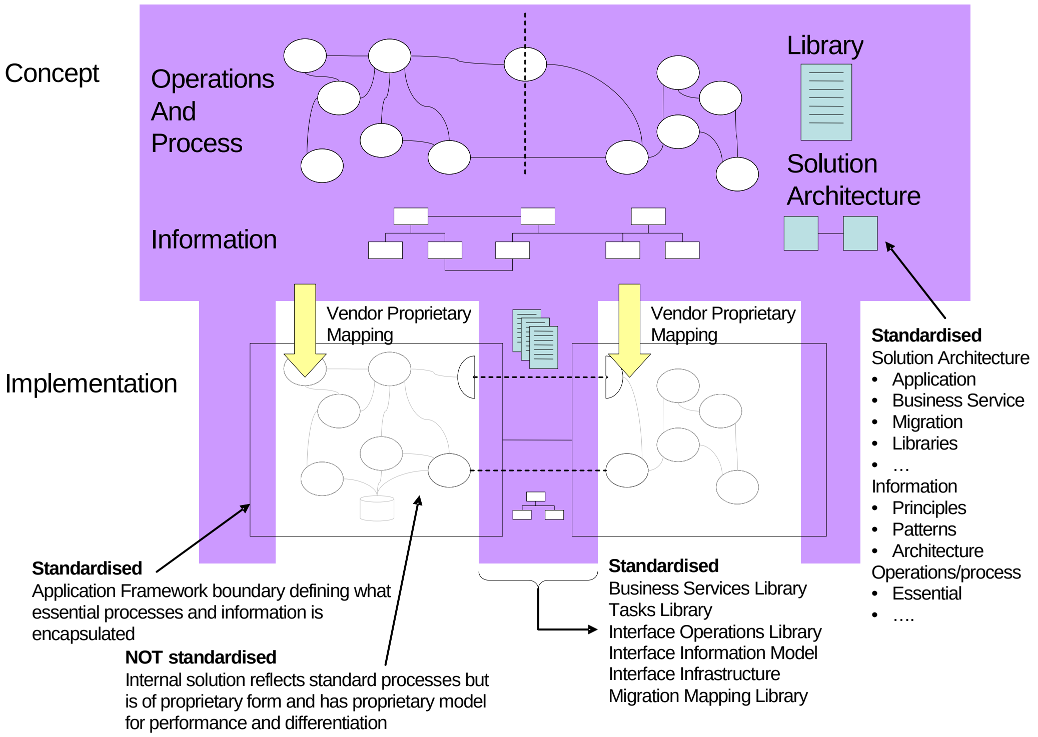
IMPORTANT NOTE: This section focuses on the standard implementation of TM Forum Business Services. To be compliant with the TM Forum Business Services specifications from an implementation perspective, the TM Forum Interface Program work **must** be used. At this stage the implementation forms (i.e. OSS/J, MTOSI, IPDR, and MTNM) do not provide full support for TM Forum initiation, SLA features etc and as a consequence it is not possible to claim compliance to TM Forum Business Services specifications from an implementation perspective. It is however possible to support key task interactions using the current Interface Program interfaces with minor vendor extensions to carry the Business Services and task information and an example of this can be found via the Harmony catalyst.

A number of activities are in place to accelerate the TM Forum Interface Program work to enable compliance. Member companies who wish to provide standard implementation forms of Business Services should work through the Interface Program to ensure consistency with other Interface Program work.

2.1. Standardization of implementation

It is important to focus the consideration of standardization of implementation carefully. It is clear that many aspects of the implementation provide vendor differentiation and are not appropriate for standardization from an implementation perspective. For example the specific implementation of an algorithm may yield a performance advantage for a particular vendor, the choice of a software technology or balance of technologies may also yield a performance advantage. Likewise particular use of tooling may yield advantages in terms of rate of implementation or testing etc that enable differentiation in the market place. None of these considerations are appropriate for standardization by TM Forum with respect to the Business Services consideration.

2.1.1. Positioning Standardization Work



Standardised
Solution Architecture
- Application
- Business Service
- Migration
- Libraries
- …

Information
- Principles
- Patterns
- Architecture

Operations/process
- Essential
- ….

Standardised
Application Framework boundary defining what essential processes and information is encapsulated

NOT standardised
Internal solution reflects standard processes but is of proprietary form and has proprietary model for performance and differentiation

Standardised
Business Services Library
Tasks Library
Interface Operations Library
Interface Information Model
Interface Infrastructure
Migration Mapping Library

Figure 0-5 -Positioning Standardization Work

The figure above depicts the problem space in terms of standardization of Concept and standardization of Implementation relevant to TM Forum. The area marked in purple can be considered for standardization as highlighted below.

The conceptual space is open to treatment via TM Forum standardization. The depth may vary depending upon the maturity of the space being standardized. Maturity of a particular problem-solution area tends to lead to both refinement of understanding and reduction of opportunity for innovation both of which increase the likelihood of agreement. The figure shows Information Model (i.e. Information Framework/TIP) and Operations Process (i.e. Business Process Framework/TIP) in the conceptual view.

The implementation space is only open to TM Forum standardization at (or near to) the interfaces that exist at the boundaries between rational system components. The system components are identified by Application Framework and this is depicted in the implementation layer of the figure. An exchange between Application Framework applications is defined in the conceptual layer by the Business Services.

It is important to note that the core of the applications, although governed by and directed by the definitions in Business Process Framework and Information Framework, is not exposed (otherwise it would be at an interface) and as a consequence does not need to actually follow any aspect of the standard so long as it appears to have done so when viewed through an interface.

Another way of looking at the same set of relationships is depicted below.

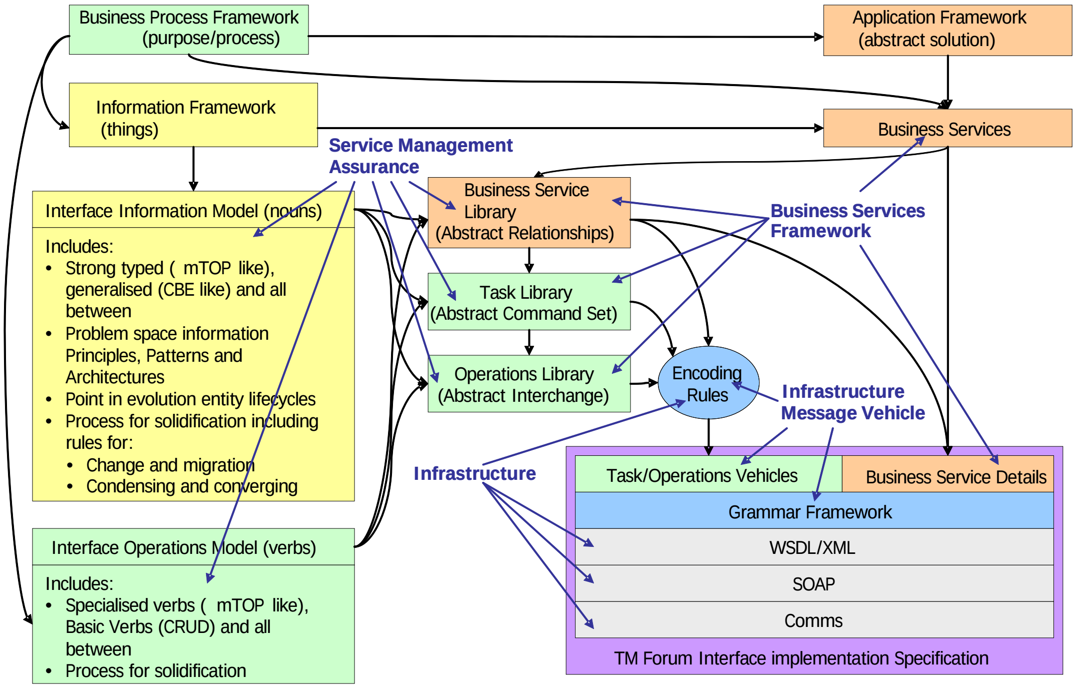

Business Process Framework
(purpose/process)
Application Framework
(abstract solution)
Information Framework
(things)
Business Services
Service Management
Assurance
Interface Information Model (nouns)
Includes:
• Strong typed (mTOP like), generalised (CBE like) and all between
• Problem space information Principles, Patterns and Architectures
• Point in evolution entity lifecycles
• Process for solidification including rules for:
• Change and migration
• Condensing and converging
Business Service Library
(Abstract Relationships)
Task Library
(Abstract Command Set)
Operations Library
(Abstract Interchange)
Business Services Framework
Encoding Rules
Infrastructure Message Vehicle
Infrastructure
Task/Operations Vehicles
Business Service Details
Grammar Framework
WSDL/XML
SOAP
Comms
TM Forum Interface implementation Specification
Interface Operations Model (verbs)
Includes:
• Specialised verbs (mTOP like), Basic Verbs (CRUD) and all between
• Process for solidification

Figure 0-6 - Conceptual work feeding to implementation

This can be considered in a simplified layered view as in the following diagram.

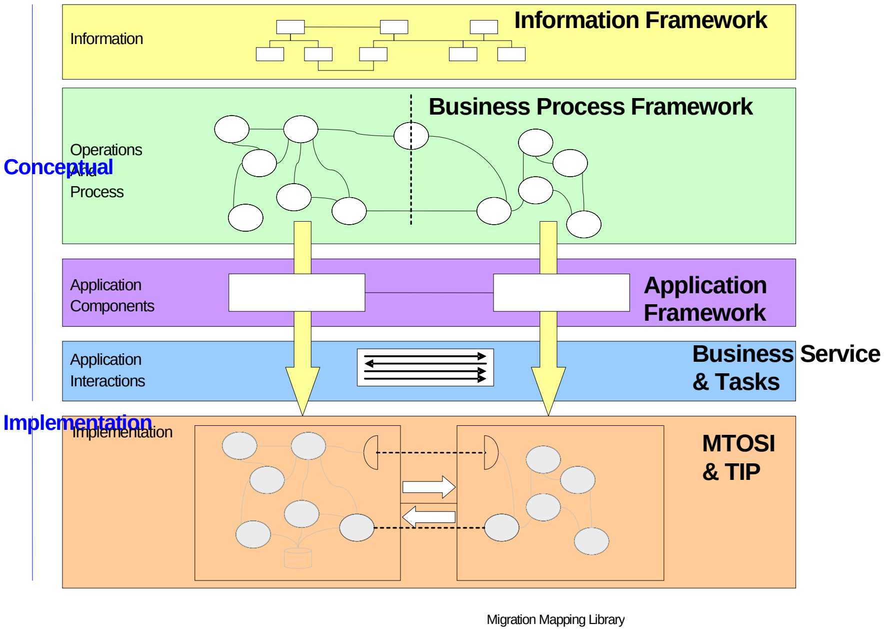


Figure 0-7 – Simplified “layered” view

2.1.2. External Interfaces, Internal Interfaces, and Meditation

Considering the External Interfaces further, as noted above the conceptual interaction between the two applications is specified by the Business Services, however, it is vital for interoperability and integration cost reduction that the TM Forum standardization work also accurately specifies all aspects of the interface from the communications choices through the binding, encoding, MEPs (Message Exchange Patterns) etc. This is the area where implementation standardization has high value in enabling interoperability and is essentially mandatory. This is the primary focus of the TM Forum Interface Program (and prior to the formation of that team of the TM Forum Interface Program team (TIP), IPDR team and OSS/J team), and the products of this program such as the MTOSI interfaces can be considered as an early implementation form of Business Services. The software solution that provides the external Interface implementation façade could be of any form and detail so long as it provides the correct, expected, standardized externally visible behavior via the specified mechanisms using the specified "External" technology.

Considering this software implementation of the interface further, it is clearly beneficial to consider normalizing the implementation forms that provide the External Interface. This will benefit the implementation organizations and accelerate deployments. Mechanisms for sharing the benefits of common software are offered by the software industry via Open Source etc.[2]. The OSS/J External Interfaces can be considered as an implementation of Business Services. The form and structure of the APIs and supporting software is beyond the scope of Business Services standardization.

One potential challenge in this use of common software is the apparent imposition that the API makes on the model/interaction choice for the internal application. If the external interface is MTOSI version 1 (model and interaction) so must the internal model/interactions be. Clearly a lock step relationship of this form is not ideal[3]. The External form and internal form can be decoupled by considering a mediation function between the API and the External Interface as shown below.

[2] Another consideration is the standardization/normalization of the implementation environment and provision of an Open Source OSS infrastructure. This is orthogonal to the consideration of implementation of Business Services and Frameworx Interfaces and hence outside the scope of this document but nevertheless is a relevant focus for potential standardization/normalization.

[3] Consider versioning challenges and MEP (Message Exchange Pattern) variety in different External environments.

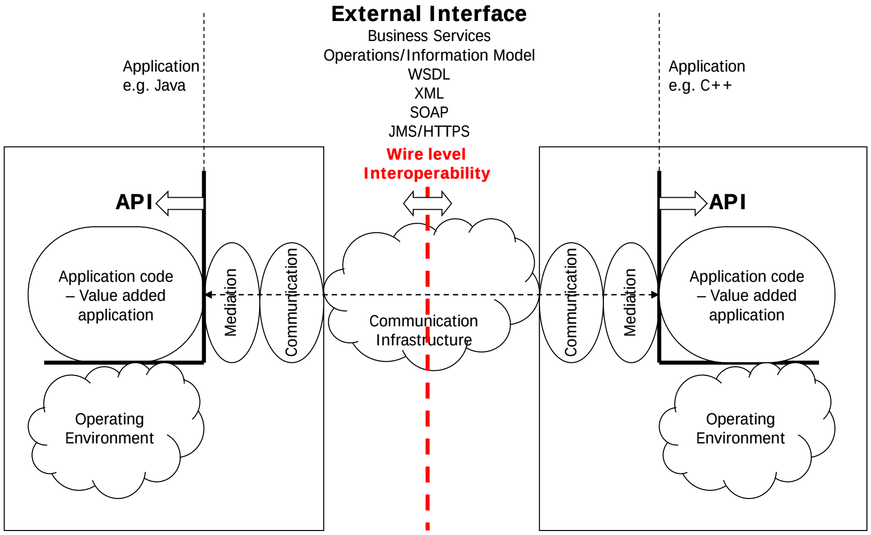
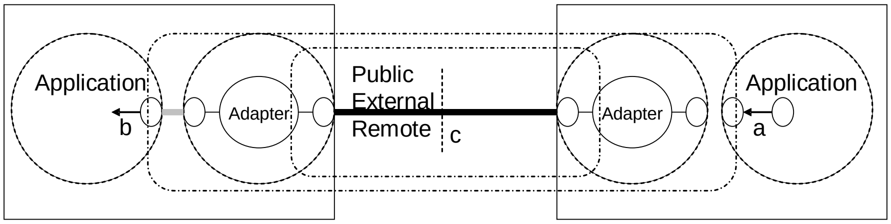


Figure 0-8 – External Interfaces mediated to Internal APIs

It should be noted that although a mediation function is shown on both the client and provider platforms in some situations there may only be relevant mediation at one end and further there may be a mediation function provided by the middleware in the communications environment dealing with aspects of interoperability. These cases are not considered further here.

The implications of the mediation are described in the following two figures. Figure 0-9 considers a single direction of interaction between two entities. This could be the Request of a Request-Response MEP (Message Exchange Pattern) or a Response of the Request-Response MEP or a Notification, etc.

The second Figure 0-10 considers the implications of partial incompatibility between the two systems (client and provider) and between the systems and the External Interface definition. Although mediation can perform the necessary transformations to overcome some differences, clearly it is not possible to use mediation to bring about a capability that the provider system does not support or to convey information that is not supported by the External Interface.

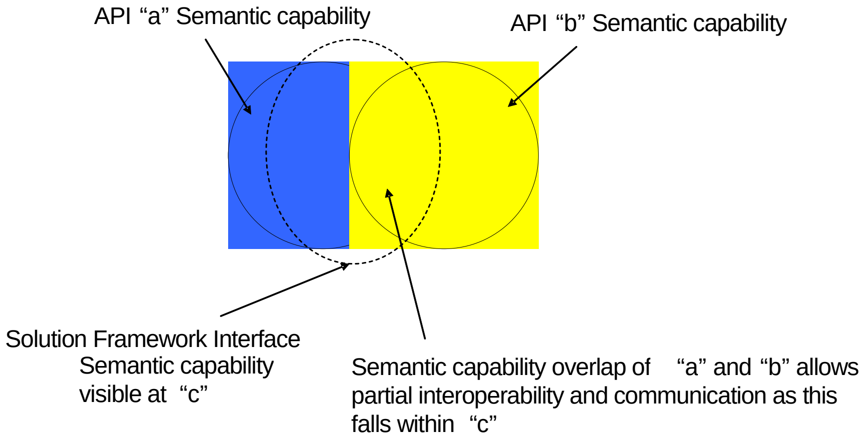


Although the essence of the exchange seen at “c” that intersects the Public External Remote interface is the same as that at “a”/”b”, the form and detailed expression can be quite different as it is serialised/encoded to pass across the communications system and is adapted at each end. The adaptation allows a very different form to be conveyed over the communications system to that perceived by the Applications at “a” and “b”.

The Adapters may be asymmetric, i.e. a and b may not be the same, and indeed the adapter at one end may be null, e.g. “a” may be the same as “c”.

Figure 0-9 - Adaptation “within” the Interface between Applications

Although the expression of API “a” and “b” may have quite different representations and indeed semantic granularity the two can interoperate via mediation as there is an overlap.


API “a” Semantic capability
API “b” Semantic capability
Solution Framework Interface Semantic capability visible at “c”

Semantic capability overlap of “a” and “b” allows partial interoperability and communication as this falls within “c”

Figure 0-10 - Semantic Overlap

The relationship between Business Services and TM Forum Interfaces can be seen in Figure 0-11. In this scenario four systems are interacting. The systems are decomposed using the Application framework Application Framework. The interactions between these applications are specified using Business Services. The tasks which are embodied with in a Business Service are an abstraction of the Business Process Framework processes for that underlining system. The message flows between the systems are implemented by interface messaging.

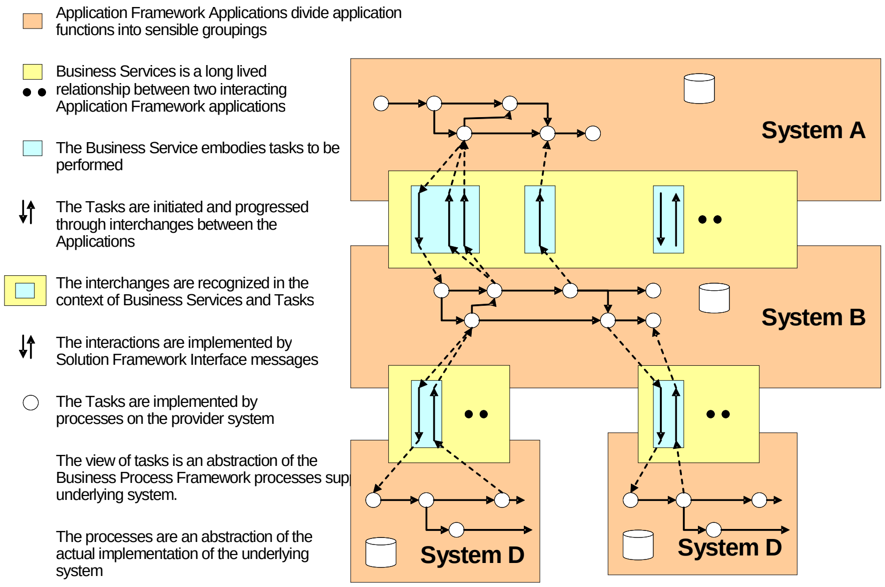

Application Framework Applications divide application functions into sensible groupings
Business Services is a long lived relationship between two interacting Application Framework applications
The Business Service embodies tasks to be performed
The Tasks are initiated and progressed through interchanges between the Applications
The interchanges are recognized in the context of Business Services and Tasks
The interactions are implemented by Solution Framework Interface messages
The Tasks are implemented by processes on the provider system
The view of tasks is an abstraction of the Business Process Framework processes sup underlying system.
The processes are an abstraction of the actual implementation of the underlying system
System A
System B
System D
System D

Figure 0-11 – Architect's View of Business Services and TM Forum Interface

2.2. Business Services from Concept to Implementation

This section provides an initial view of the breakdown of the process of moving from the Conceptual form of Business Services to the Implementation form. The section considers this from two major perspectives: the standardization development process and the selection process of the purchaser of applications.

From a standardization perspective there appears to be two further aspects, the development of the Essential model and the application of this model for specific cases.

The development of the Essential model is captured in the figure below that shows the progression from the Business Services Meta-Model (read anti-clockwise from "Develop Business Services Meta-Model"). The figure shows three distinct implementation artifacts: MEP, Message Structure and Message Encoding. These can be considered as forming the Infrastructure of the TM Forum Interface. The figure represents the process of continual refinement.

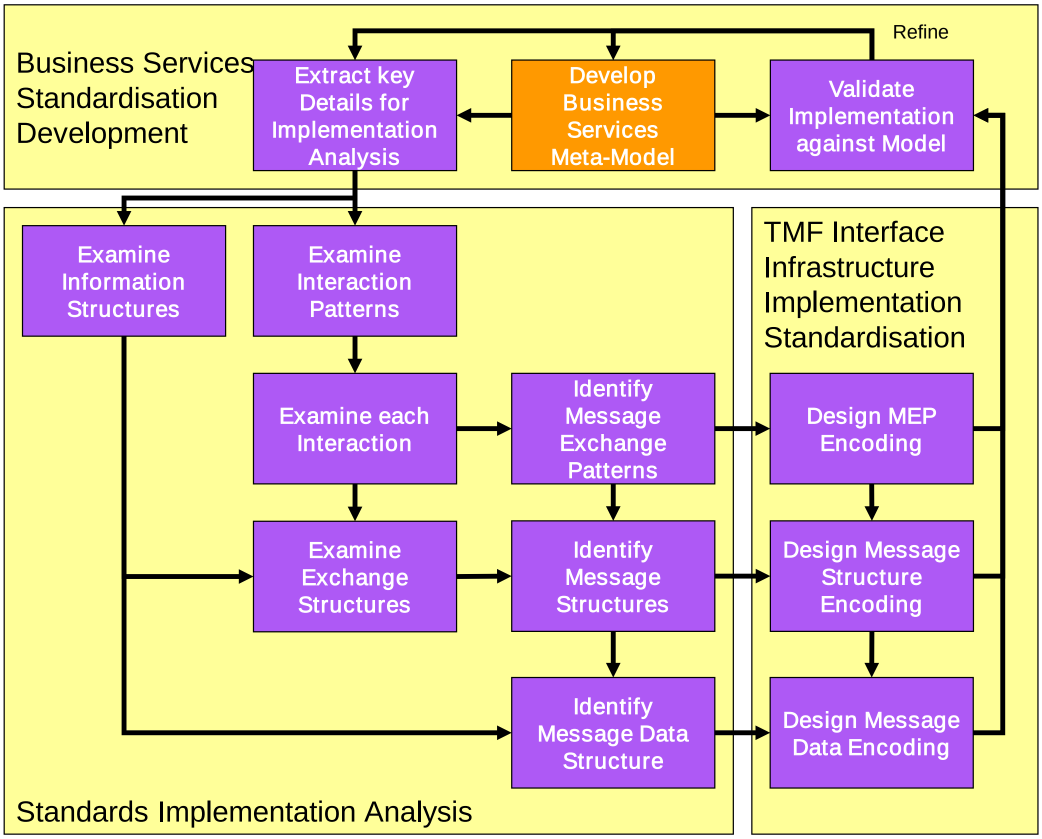

Refine
Business Services Standardisation Development
Extract key Details for Implementation Analysis
Develop Business Services Meta-Model
Validate Implementation against Model
Examine Information Structures
Examine Interaction Patterns
TMF Interface Infrastructure Implementation Standardisation
Examine each Interaction
Identify Message Exchange Patterns
Design MEP Encoding
Examine Exchange Structures
Identify Message Structures
Design Message Structure Encoding
Identify Message Data Structure
Design Message Data Encoding
Standards Implementation Analysis

Figure 0-12 - Standardizing the Business Services Integration Framework

The current TM Forum Interfaces (MTOSI, MTNM, OSS/J and IPDR) can be considered as a stage in the refinement illustrated above. The Business Services Meta-Model is maturing and the TM Forum Interface Program is validating current implementation forms against the model. This process will continue as the Interface Program develops its work.

The Interface Implementation patterns can be used in conjunction with the Business Services Meta-Model to solve particular application problems from a standards perspective to build a library of solution components. This is shown in the figure below.

The figure (in this case read from top to bottom) illustrates that after an initial Business Analysis phase Business Services that already exist may be identified. These Business Services can start to build a model of Business Services interactions for the particular case but often these will need to be augmented by new Business Services. These Business Services may give rise to changes in the Interface implementation not only in terms of the specific problem but also on occasions in terms of the Interface Infrastructure that may need extension or adjustment. This in turn may on rare occasions lead to a requirement to change the Business Services Meta-Model.

This process will provide additions to the Libraries of Business Services / Task artifacts and of TM Forum Interface artifacts.

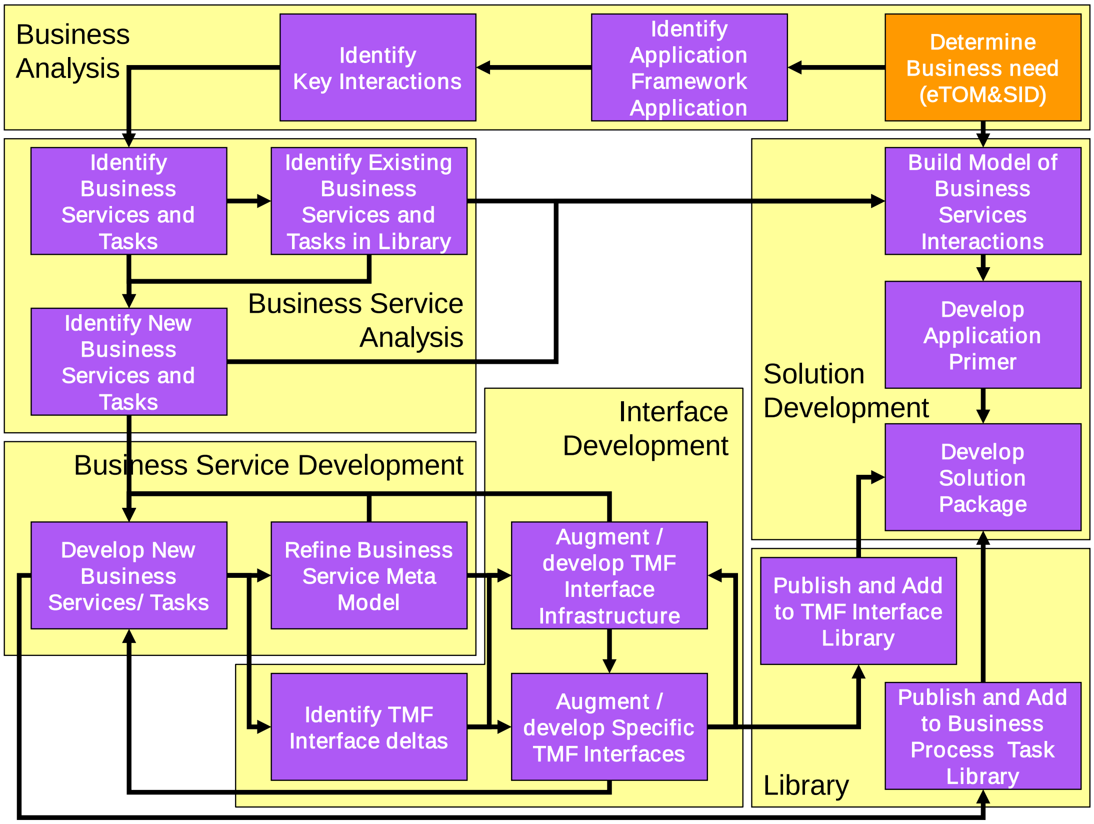

Business Analysis
Identify Key Interactions
Identify Application Framework Application
Determine Business need (eTOM&SID)
Identify Business Services and Tasks
Identify Existing Business Services and Tasks in Library
Business Service Analysis
Identify New Business Services and Tasks
Build Model of Business Services Interactions
Develop Application Primer
Solution Development
Develop Solution Package
Interface Development
Business Service Development
Develop New Business Services/ Tasks
Refine Business Service Meta Model
Augment / develop TMF Interface Infrastructure
Publish and Add to TMF Interface Library
Identify TMF Interface deltas
Augment / develop Specific TMF Interfaces
Publish and Add to Business Process Task Library
Library

Figure 0-13 - Adding an Implementation Business Services to the Library

The final consideration covered here is a process for deployment of a system that has been specified in terms of Application Framework, Business Services and TM Forum Interfaces.

This process recognizes that there may not be solutions already available for the specific case to be covered and this may lead to some combination of application development, Business Services addition or extension and Interface addition or extension.

The Business Services and TM Forum Interfaces are architected such that they may be extended by proprietary addition. This extension can be brought back to the Standards community via a process similar to that described above. The standardized output resulting from a refinement of a business use-case may be added to the standard library. This aspect of the process has not been explored in detail here and will need further analysis and extension as part of the establishment of the Business Services Management Board.

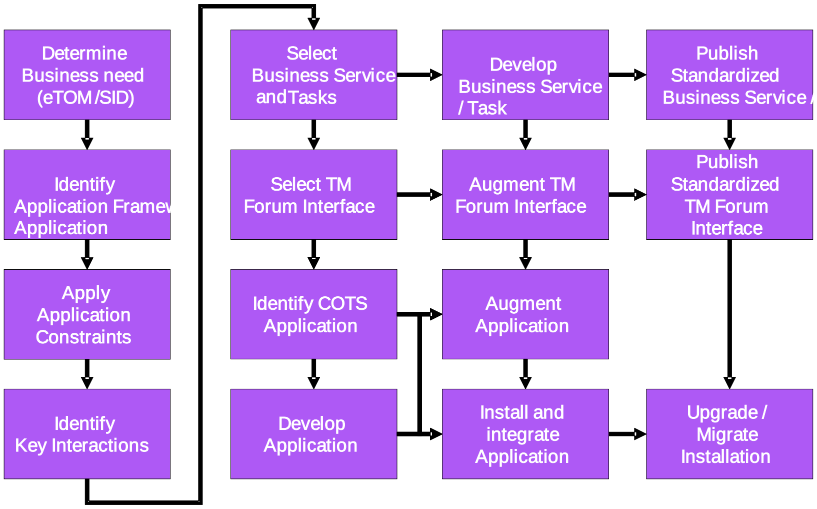

Determine Business need (eTOM/SID)
Identify Application Framev Application
Apply Application Constraints
Identify Key Interactions
Select Business Service andTasks
Select TM Forum Interface
Identify COTS Application
Develop Application
Develop Business Service / Task
Augment TM Forum Interface
Augment Application
Install and integrate Application
Publish Standardized Business Service /
Publish Standardized TM Forum Interface
Upgrade / Migrate Installation

Figure 0-14 - Finding the right Business Services

The following figure illustrates the process of standardization showing that variety, resulting from innovation during the introduction of a feature, will reduce over time as the feature matures and that this variety reduction will result in increasing opportunity to standardize. This standardization will result in additions to the library of Business Services/Tasks and TM Forum Interfaces.

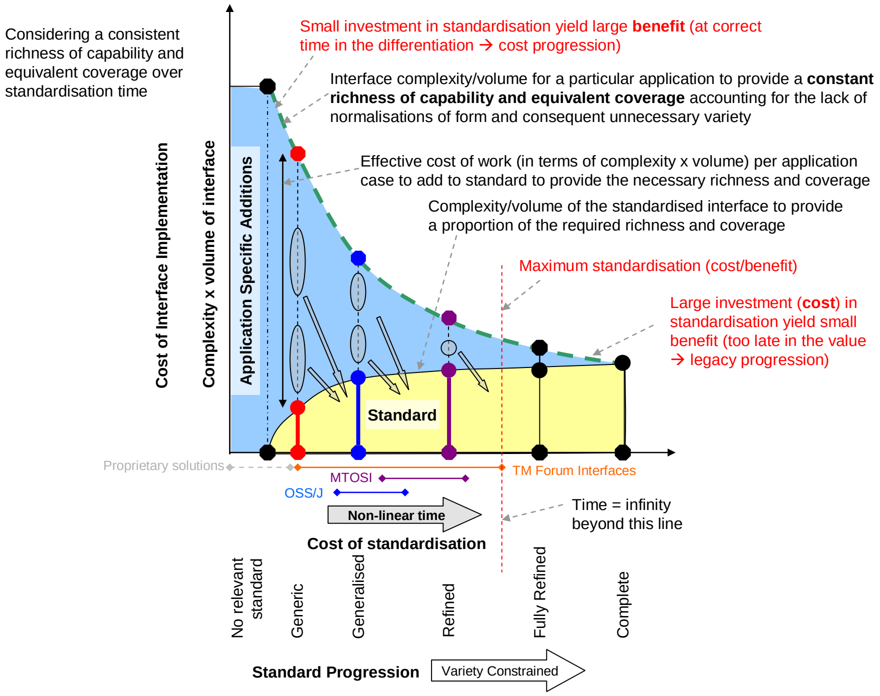

Considering a consistent richness of capability and equivalent coverage over standardisation time
Small investment in standardisation yield large **benefit** (at correct time in the differentiation → cost progression)
Interface complexity/volume for a particular application to provide a **constant richness of capability and equivalent coverage** accounting for the lack of normalisations of form and consequent unnecessary variety
Effective cost of work (in terms of complexity x volume) per application case to add to standard to provide the necessary richness and coverage
Complexity/volume of the standardised interface to provide a proportion of the required richness and coverage
Maximum standardisation (cost/benefit)
Large investment (**cost**) in standardisation yield small benefit (too late in the value → legacy progression)
Cost of Interface Implementation
Complexity x volume of interface
Application Specific Additions
Standard
Proprietary solutions
MTOSI
OSS/J
TM Forum Interfaces
Non-linear time
Time = infinity beyond this line
Cost of standardisation
No relevant standard
Generic
Generalised
Refined
Fully Refined
Complete
Standard Progression
Variety Constrained

Figure 0-15 - Standards development progression

2.3. The Task Body – Advancing the TM Forum Interfaces

Consider the analogy of the traditional telephone bill. The service provider and the consumer have a contractual relationship and within that Business Services the service provider has the task of sending the bill to the consumer. This bill is sent in an envelope and for this analogy the envelope can be considered as the task body in physical implementation interface[4]. The bill payment is mandatory (much to the annoyance of the consumer) and the bill is marked with SLAs on payment (last payment date) that indicates clearly that it is mandatory. Often the service provider uses the envelope to send other material too (that tends to go in the recycling bin). This material includes "special offers" and details of other services. Although the service provider would clearly like the consumer to take up the offers, they are in no way mandatory in nature. So through this analogy, it is clear that in that particular Business Services environment the task body can contain several separate items some of which are mandatory and some of which are best effort.

The External Interface form of the OSS/J Order API method createAndStartOrderByValue would also convey a number of requests in one message (and hence effectively in one task body) but all of them have to be mandatory (as the "try…" version of the method deals with best effort, but all of the items have to be best effort). Clearly the OSS/J form requires some enhancement if it was to cover the general case. That is assuming that the general case is applicable to the telecoms management environment.

[4] Clearly the envelope is more than just the task body as it is actually the entire message but for this analogy the difference in nesting of structure in the physical world and the logical world can be ignored.

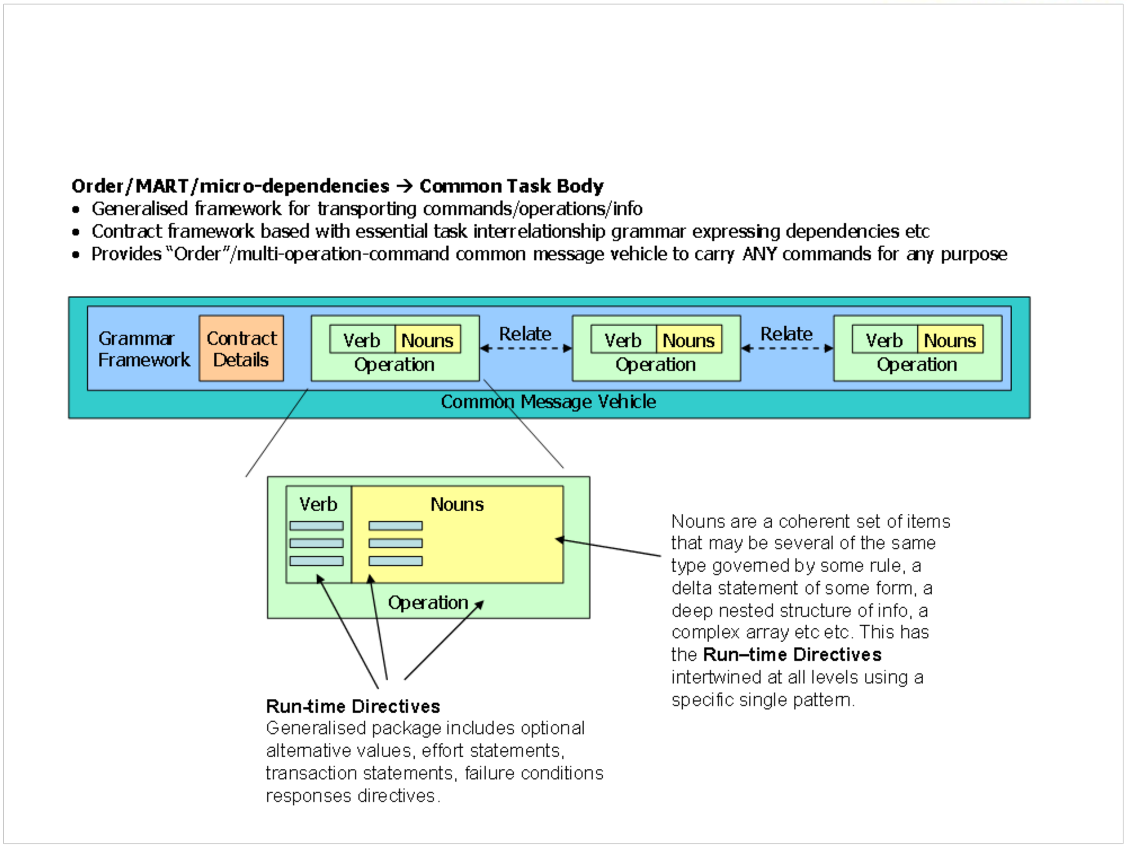
MTOSI MART (Multi-Action and Request Transactions) provides an additional capability over and above the Order API in that there is a capability to send several similar requests as deltas on a base request in one message. Cases for this can be readily seen in the area of bulk provisioning.

Considering the MTNM[5] interface (especially in the CreateAndActivateSNC operation) it is clear that there are specific examples of effort statements (such as protection effort) where there are opportunities to mark particular attribute values as mandatory or optional. However, each of these cases is “hand crafted” in the interface specification and there is no generalized mechanism. It is clear that there is a need for the statement of “effort” and also a need for a generalized approach to this.

It is clear that some blend of the various capabilities discussed above will be needed to satisfy the general Business Services requirement and that construction and deployment of this evolved blended form will require careful consideration of evolution and migration. The evolution/migration approach is for further study outside the scope of this document but it should be noted that the mediation/adaptation architecture discussed earlier along with the publication of transforms appears to form the basis for a migration approach.

A sketch of the form of interface that may result from the considerations above is depicted below. The structures shown have intentionally been related to familiar terms from grammar to emphasize the relationship with human language and the need for a structure that provides the combinatorial fluidity of a human language and the clear decoupling of the form from the content. Work on this structure is at an early stage.

[5] Multi-Technology Network Management



Order/MART/micro-dependencies → Common Task Body

- Generalised framework for transporting commands/operations/info
- Contract framework based with essential task interrelationship grammar expressing dependencies etc
- Provides "Order"/multi-operation-command common message vehicle to carry ANY commands for any purpose

Nouns are a coherent set of items that may be several of the same type governed by some rule, a delta statement of some form, a deep nested structure of info, a complex array etc etc. This has the **Run–time Directives** intertwined at all levels using a specific single pattern.

Run-time Directives
Generalised package includes optional alternative values, effort statements, transaction statements, failure conditions responses directives.

Figure 0-16 - The Task Body

3. Tooling for Solutions Frameworks

TM Forum Frameworx spans through multiple domains – Business Process modeling and Application Components definition (via Business Process Framework and Application Framework), Information and Data modeling, Business Services modeling and Interface definition. It establishes specifications and defines lifecycle of its numerous components throughout a system of many views – Business, System, Implementation and Deployment.

In order to support the complex modeling methodologies developed in various areas of it is necessary to have proper tooling. Moreover, such tooling helps to move beyond development of paper-only specifications to the system of machine-readable rigorously defined artefacts that can be distributed among the members.

The tool set needs to explicitly demonstrate support for the specification of TM Forum Architecture and Designs based on Business Process Framework/Information Framework/Application Framework and Integration Framework. The tool set needs to provide open standards based mechanisms for the export and import of design information between the TM Forum tool set components and individual members' tools.

Consolidation of work on the tools across entire ensures the consistency and practicality of specifications developed by TM Forum.

3.1. Creation of Artifacts

The complex end-to-end process of creation of Artifacts can be broken into several activities (Fig. 5.1):

- **Business Process modeling:** describes the elements of Service Provider business processes via Business Process Framework. In order to support continuity and rigorousness of the Architecture, Business Process Framework process elements as well end-to-end processes need to be formally modeled.
- **Application Components modeling:** uses Application Framework to describe the functional breakdown of Service Provider OSS architecture.
- **Information and Data modeling:** uses the Information Framework (Information Framework) as well as a set of Data models (TIP models, etc.)
- **Model harmonization:** Model harmonization is necessary to simplify and thus to reduce the cost of integration between application components using different information and data models, which are in numerous existences throughout telecom and enterprise management industry.
- **Business Services:** Business Services enable putting together an OSS comprised of multiple Application components in order to execute target Business Processes.
- **Interface definition:** Interfaces between various Application components need to be formally specified in order for components to work together.

- **Code generation:** Architecture covers various aspects of OSS lifecycle, including implementation, which implies code generation and deployment.
- **Change and Migration support:** As Architecture elements evolve over time, it is necessary to support this evolution and adjust other dependent components to accommodate the change.

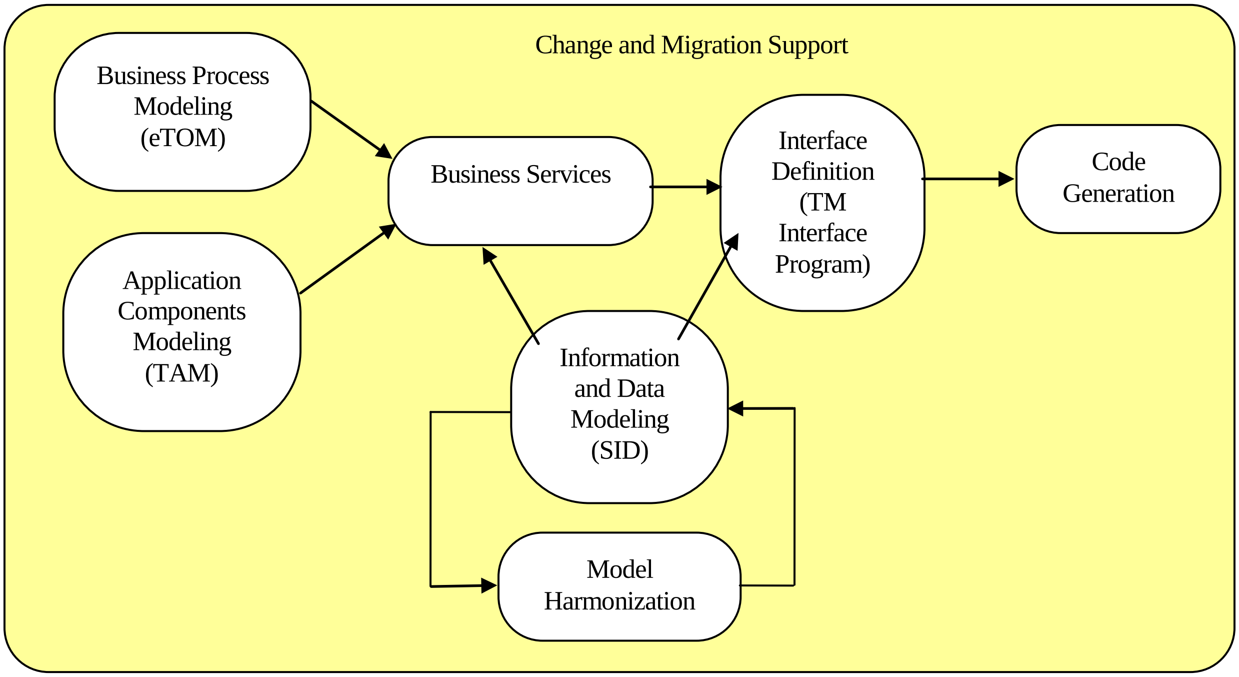

Change and Migration Support
Business Process Modeling (eTOM)
Application Components Modeling (TAM)
Business Services
Interface Definition (TM Interface Program)
Code Generation
Information and Data Modeling (SID)
Model Harmonization

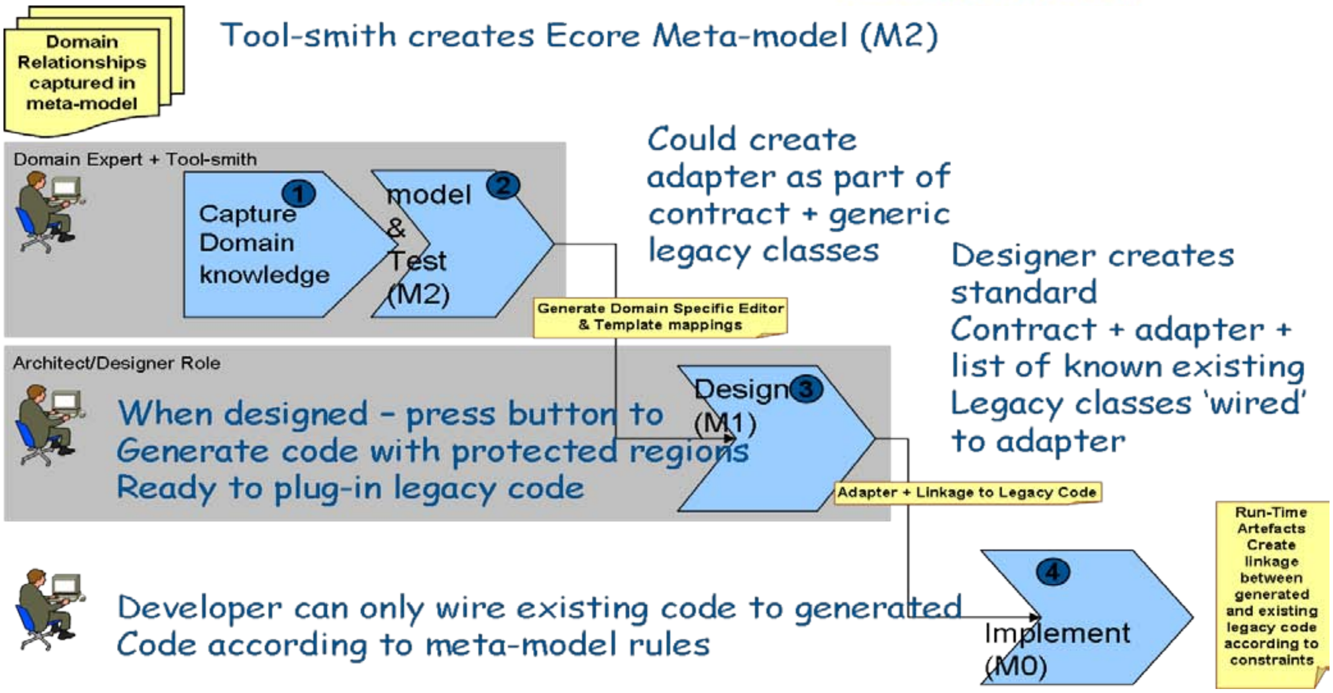
Figure 0-17 - Artifacts Creation Process

All these activities are "chained" together, which creates the requirements on the compatibility between the format of the artifacts generated by one process components and formats consumed by other process components.

The compatibility should exist not only at the level of data formats of the artifacts (XML Schema, IDL, etc) but also at the level of metamodels used by different process components.

3.1.1. Business Service Tooling in Eclipse

As is outline in the GB942 CP the normative specification of Business Service is based on metamodeling approach. Within Eclipse and analogous MOF layering, ECORE (M3) can be used to define a new Domain Specific Language meta-model (M2). This new meta-model constrains what is semantically permissible in the DSL model (M1). In effect it allows a tool builder to create a new modeling language tailor-made for a specific problem domain (e.g. Business Service definition). By using Eclipse tools such as GMF (GMF is a tool for automatically generating rich visual M1 editors from ECORE), new graphical editors can be created for modeling any desired problem spaces. The following illustration demonstrates the process for building ECORE-GMF based tools.


Domain Relationships captured in meta-model
Tool-smith creates Ecore Meta-model (M2)
Domain Expert + Tool-smith
1
Capture Domain knowledge
2
model & Test (M2)
Could create adapter as part of contract + generic legacy classes
Generate Domain Specific Editor & Template mappings
Designer creates standard Contract + adapter + list of known existing Legacy classes 'wired' to adapter
Architect/Designer Role
When designed - press button to Generate code with protected regions Ready to plug-in legacy code
3
Design (M1)
Adapter + Linkage to Legacy Code
Developer can only wire existing code to generated Code according to meta-model rules
4
Implement (M0)
Run-Time Artefacts Create linkage between generated and existing legacy code according to constraints

In addition the tool capability can be enhanced by the following means;

- Additional constraints may be provided by OCL.
- Using a code generator language on the created diagram (Xpand, Jet, and Velocity) allows us to generate running code (M0) from the DSL. Each user created diagram corresponds to a specific Business Services hence many Business Services can be quickly defined and runtimes created.
- Languages such as QVT (Query View Transform) allow us to do bi-directional transforms between M1 models, e.g. DSL to/from UML.

The ability to abstract in models is highly desirable; the use of a meta-model effectively allows us to abstract the abstraction!

Sections below describe the requirements, specific to each individual activity within Architecture modeling process.

Note: the term meta-model is synonymous with the older term 'abstract syntax' which may be seen elsewhere.

3.1.2. Modeling of Application Framework components and Business Process Framework business processes

TM Forum describes the Digital Media, Communications and Information industries business processes via Business Process Framework process elements. Application

Framework is used to describe the functional breakdown of industries systems architecture. In order to support continuity and rigorousness of the Architecture, Business Process Framework process elements, end-to-end processes and Application Framework Components need to be formally modeled.

Requirements:

- The tool set shall allow creation and reuse of domain specific metamodels to model Application Framework entities and Business Process Framework process components;
- The tool set shall allow tying of modeled Application Framework entities and Business Process Framework business processes to the implementation-independent Business Services definition.

3.1.3. Business Services Modeling

Business Services enable putting together an OSS comprised of multiple Application components in order to execute target Business Processes.

Requirements:

- The tool set shall allow creation of metamodels for implementation-independent definition of Business Services;
- The tool set shall allow creation of implementation-independent models of Business Services using specialized metamodels;
- The tool set shall allow tying of implementation-independent models of Business Process Framework Business Processes and Application Framework Components to implementation-independent definition of Business Services;
- The tool set shall allow tying of various Information and Data models to implementation-independent definition of Business Services;
- The tool set shall allow tying of implementation-specific data models (WSDL/IDL) to implementation-independent models of Business Services.

It should be possible to use a standard language (for example, OMG Object Constraint Language (OCL)) to specify constraints for Business Services Pre and Post-conditions.

3.1.4. Management Interface Definition and Code Generation

In order for Application Components to work together, management interfaces between them need to be formally specified. Such interface specifications are closely tied to the implementation view of the Business Services, Data Models and are using Interface Definition Information models.

Requirements:

- The tool set shall allow creation of metamodels for implementation-independent definition of Interface Information Models;
- The tool set shall allow creation of Interface Definition models using specialized metamodels;

- The tool set shall allow tying of various Data models to the definition of Management Interfaces;
- The tool set shall allow tying of implementation-specific data models (WSDL/IDL) to implementation-independent models of Management Interfaces;
- The tool set shall allow generation of implementation stubs and test code from implementation-specific Interface Definition

It should be possible to use a standard language (for example, OMG Object Constraint Language (OCL)) to specify constraints for Interface Operations and Operation Parameters definitions.

3.1.5. Information and Data Modeling - Model Harmonization

3.2. Information Framework Technical Requirements by Functional Area

3.2.1. Common requirements

- Elements of the tooling chain shall support UML 2.x;
- Elements of the tooling chain shall Support consistent modeling “taxonomy” across multiple tools to allow chaining;
- Elements of the tooling chain shall support for OMG standardized XMI for metadata exchange;
- Elements of the tooling chain shall have the ability to import and/or export UML and generate ECORE models from xsd.
- Elements of the tooling chain shall support model artifacts/interface specification/implementation code evolution and migration;
- Elements of the tooling chain shall have an ability to export the supported artifacts in a viewable format (e.g. HTML).
- Elements of the tooling chain shall support multi-user development environment;
- Elements of the tooling chain shall allow easy integration with Eclipse environment (support Eclipse as IDE);
- Elements of the tooling chain shall support integration with version control system (Subversion as a choice of TM Forum);
- Elements of the tooling chain shall have an ability to generate documentation.

3.2.2. Modeling Environment

- Modeling environment shall provide full OCL support including validation and mapping into code generation;

- Modeling environment shall support QVT or other model to model transform language;
- Modeling environment shall provide the Logical and Diagrammatic manipulation APIs to the programmer;
- Modeling environment shall support Domain Specific Language.

3.2.3. Code Generation Environment

- Code generation environment shall provide an ability to generate interface specification: WSDL/XML, CORBA IDL;
- Code generation environment shall provide an ability to generate variety of interface implementation code: Java, C++, C#, Ruby, Cobol/Algol, etc. for legacy systems;
- Code generation environment shall support for test case specification and/or generation;
- Code generation environment shall provide the process “emulation” capability;
- Code generation environment shall provide the ability to perform change Impact analysis.

3.2.4. Model and Interface Mapping Environment

- Model and interface mapping environment shall provide the capability to perform mapping between fragments of different information and data models;
- Model and interface mapping environment shall provide the capability to perform mapping between semantically equivalent operations of the management interfaces;
- Model and interface mapping environment shall support test case specification and/or test code generation;
- Model and interface mapping environment shall provide the process “emulation” capability
- Model and interface mapping environment shall provide the ability to perform change Impact analysis.

3.2.5. TM Forum Tool

BT have contributed, free of charge and without IPR, Eclipse-based tooling which can be downloaded by AH team members from the Integration Framework Collaboration Workspace. This tooling utilizes Eclipse EMF, GMF, Xpand and BIRT to meet the previous requirements. Comments, feedback and change requests can be made through the Integration Framework exploder until the tool is fully managed under TM Forum Integration Framework community post release 1.0

Future work is ongoing in relation to Tooling with TM Forum through the Tooling task force. The Tooling Task Force will identify and address the most outstanding issues with TM Forum Tooling, targeting the process of creation, sharing and maintenance of artifacts throughout their lifecycle. The effort will leverage existing work on tooling conducted prior under various TM Forum teams (mTOP Methodology, Architecture Harmonization, MTOSS/J).

It is also desired that the group leverage the work of other major standards organization so that the resulting TM Forum recommendations are as consistent with the industry at large as possible

4. Administrative Appendix

4.1. Glossary

Item	Description
Eclipse	An Open-Source tool environment/platform
Ecore	Eclipse based tool language for creating meta-models, MOF reference implementation.
GMF	Graphical Modeling Framework – Eclipse tool for generating visual editors from Ecore
MOF	OMG’s definition of a meta-meta model.
MTNM	Multi-Technology Network Management
MTOSI	Multi-Technology Operations System Interface
	New Generation Operations Systems and Software
SNC	Sub Network Connection
TigerStripe	Open-Source Eclipse tool used by TIP
TIP	TM Forum Interface Program
UML	Unified Modeling Language

4.2. About this document

This is a TM Forum Guidebook. The guidebook format is used when:

The document lays out a ‘core’ part of TM Forum’s approach to automating business processes. Such guidebooks would include the Telecom Operations Map and the Technology Integration Map, but not the detailed specifications that are developed in support of the approach.

Information about TM Forum policy, or goals or programs is provided, such as the Strategic Plan or Operating Plan.

Information about the marketplace is provided, as in the report on the size of the OSS market.

4.3. Document History

4.3.1. Version History

<This section records the changes between this and the previous document version as it is edited by the team concerned. Note: this is an incremental number which does not have to match the release number>

Version Number	Date Modified	Modified by:	Description of changes
0.1	11 Sept 2007	John Reilly	First version that contains Business Service granularity guidelines
0.5	10 Dec 2007	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis	Updates based on team reviews/discussions…intermediate versions included similar changes.
0.7	24 Jan 2008	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis, Alex Zhdankin	Updates based on team reviews/discussions…intermediate versions included similar changes.
0.9	3 Mar 2008	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis, Alex Zhdankin,Steve Orobec	Further Updates based on team reviews/discussions…intermediate versions included similar changes
0.9.1	16th June 2008	Ian Best	Minor formatting changes
1.0	14th July – Aug 18th 2008	Steve Orobec	Final version review comment edits
1.1	20 Aug 2008	Richard Mishra, Steve Orobec	Edits based on Approval Committee and Technical Program Council review comments
1.2	9 Sept 2008	Tina O'Sullivan	Final corrections prior to web posting
2.0	30 Oct 2008	John Reilly	Split into Concepts and Principles and User Guidelines.
2.01	01 August 2009	David Cleary, Richard Mishra, Nigel Davis,	Clean up after split documents
2.02	18 august 2009	David Cleary	Update on review comment
2.03	08 October 2009	Ken Dilbeck ,David Cleary	Formatting and proof reading
2.04	29 October 2009	Alicja Kawecki	Minor cosmetic corrections for web posting and ME
2.05	27Jan/2010	Ken Dilbeck	Solution Framework to Frameworx per Marketing
2.06	10 March 2010	Tina O'Sullivan	Minor updates to release numbering.
2.07	08 June 2010	Alicja Kawecki	Updated to reflect TM Forum Approved status
3.0	19 May 2011	Alicja Kawecki	Upversioned to align with R2.5 for Frameworx 11.0 – content remains unchanged

4.3.2. Release History

Release Number	Date Modified	Modified by:	Description of changes

1.0	September 2008	Initial release	First release to member for comments
2.01	November 2008	John Reilly	Aligned with other document sets and Integration Framework concepts
2.02	December 2008	John Reilly	Updated based on reviews
2.03	October 2009	Ken Dilbeck	Updates
2.1	January 2010	Pascale Pecha	Pre-launch of Frameworx

4.4. Company Contact Details

Company	**Team Member Representative**
Amdocs	Richard Mishra richard.mishra@amdocs.com +44 1225 327236
BT	Steve Orobec Steve.orobec@bt.com +44 7710 046883
Harris	Alex Zhdankin Alex.zhdankin@hstx.com +1-321-674-4755
Nortel	Nigel Davis nigeld@nortel.com +44-1279-405978
TM Forum	John Reilly jreilly@tmforum.org +1 469 363 4546
TM Forum	Mike Kelly mkelly@tmforum.org +44 1727 863764

4.5. Acknowledgments

[Editor's note: If we have missed anyone please contact one of the editors to have the list corrected and please accept our apologies]

This document was prepared by the members of the TM Forum Architecture Harmonization team:

Steve Orobec, BT Team Co-Lead
Richard Mishra, Amdocs, Editor
Alex Zhdankin, Harris-Stratex Network, Editor
Nigel Davis, Nortel, Editor
John Reilly, TM Forum, Editor
Mike Kelly, TM Forum, Editor
Steve Fratini
Francesco Caruso
Martin Huddleston

Dave Raymer
Dave Milham
John Petrie
Eric Dillon
Richard Craddock