31st Annual INCOSE international symposium
Honolulu, HI, USA
July 17 - 22, 2021

An Elaboration of Service Views within the UAF

Matthew Hause
Systems Solutions Inc (SSI)
3208 Misty Oaks Way, Round Rock, Texas, USA
+1 917 514 7581
mhause@systemxi.com

Lars-Olof Kihlström
Syntell AB
PO Box 10022, SE-10055 Stockholm, Sweden
+46 706661978
lars-olof.kihlstrom@syntell.se

Abstract. Services in the Unified Architecture Framework (UAF) are not just software services. Services can include transport, surveillance, communications, providing healthcare and medical services, etc. The UAF implements DoDAF using the Systems Modeling Language (SysML) as well as the British MODAF and NATO NAF. The DoDAF Service views implement services by duplicating the systems views and labeling the systems elements as services. This causes some confusion with engineers who either implement solution-based service views or ignore them completely. Even when implemented, they can cause confusion in the model as it becomes difficult to tell if a model element describes a service or a system implementing a service. The UAF implementation of the MODAF services views provides a distinct set of views, concepts and traceability. The Service Oriented Views do not specify how the service is to be implemented, but the requirements for the services. The Resources (Systems) Views implement services in various phases and their deployment will modify the configurations of the system at the very highest level. This paper will show how services views trace from capabilities and how that can be used to define system resource requirements.

Introduction

Traditional Service-Oriented Architecture (SOA) is a style of software design where "services are provided to the other components by application specific components, through a communication protocol over a network." (Microsoft, 2006). SOA and services have evolved over the years and have taken on various definitions. The Open Group (2007) defines four properties of SOA:

- Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports).
- Is self-contained.
- May be composed of other services
- Is a "black box" to consumers of the service.

Note that the definition of SOA has changed from "how SOA is implemented in software" to the purpose of SOA as a representation of a business activity and its drivers, influences and structure. (Although not in the list, the service also should be loosely coupled.) The definition continued to evolve. The SOA Manifesto Organization published a manifesto for service-oriented architecture in October, 2009. They came up with the following six core values (formatting is from the original):

“Business value over technical strategy
Strategic goals over project-specific benefits
Intrinsic interoperability over custom integration
Shared services over specific-purpose implementations
Flexibility over optimization
Evolutionary refinement over pursuit of initial perfection

That is, while we value the items on the right, we value the items on the left more.” (SOA Manifesto, 2009) Rather than simply a software implementation pattern, services are linked to business value, process and strategic goals. They define a service at whatever level of the enterprise is appropriate. Business processes are defined, services are specified that further define those processes. The service names should be tied to or derived from business processes, rules, or policies. They can then be implemented by software, systems, people, organizations, or whatever resources are required. It is at this level (requirements) that the service views in the Unified Architecture Framework (UAF) are defined. The purpose of this paper is to help users of the UAF understand how best to use the services views in an architecture.

The Unified Architecture Framework (UAF)

The Systems Modeling Language (SysML) is the most widely used standardized systems modeling language and notation. It is used to model systems in both the abstract and concrete (logical and physical) views that include behavioral, structural, parametric and requirements views. (OMG, 2017) For enterprise modeling, an architecture framework is required to understand systems of systems and how they change over time. DoDAF is the Department of Defense Architecture Framework (DoD, 2012) and MODAF is the Ministry of Defence Architecture Framework (MOD, 2020). NATO created NAF version 3 (NATO Architecture Framework) based on MODAF and has recently adopted NAF version 4 (NATO, 2018). Services are central to NAF version 4 and it has been adopted by many European countries including the UK.

The Unified Architecture Framework (UAF) is built on top of SysML and is used to define the overall goals, strategies, capabilities, interactions, standards, operational and systems architecture, systems patterns and so forth (UAF, 2019). Security and human factors (personnel) views were added to the UAF to improve the coverage of these areas of concern. The UAF was previously called the Unified Profile for DoDAF and MODAF (UPDM) and was ratified by the Object Management Group (OMG). Several papers have been written on the UAF and its support of SoS modeling including (Hause, Dandashi 2015) and (Hause 2014). The full details of SysML and UAF are not included here for space reasons. Please see the above references for more information.

The UAF Services Views

The UAF Service views describe the services needed to directly support the elements described in the Capability and Operational Views. The Service Views do not specify how the service is to be implemented, but the requirements for the services. The implementation of the services is done by the Resources Views. They are deployed in various phases and their deployment will modify the configuration of the system at the very highest level. The UAF service views are derived from MODAF and DoDAF, which have their own definitions for services.

MODAF: The SOVs are a set of views that specify services that are to be used in a service-oriented architecture (SOA). In MODAF terms, services are an implementation-independent specification of a packaged element of functionality. The views describe the specification of these services, how services are orchestrated together for a purpose, the capabilities that services deliver and how services are implemented. Note that the views do not focus on the detailed design of the service, rather on the requirement the service fulfils. (MODAF, 2020)

DoDAF: The Service Views within the Services Viewpoint describe the design for service-based solutions to support operational development processes (JCIDS) and Defense Acquisition System or capability development within the Joint Capability Areas. (DoDAF, 2008)

The relationship between architecture data elements across the Service Viewpoint to the Operational Viewpoint and Capability Viewpoint can be exemplified as services are procured and fielded to support organizations and their operations or a capability. (MODAF, 2020) Services in UAF are intended to allow the operational layer to be developed without impacting on the resource layer provided that the operational layer only makes use of the functionality provided by the service interfaces. In the same sense the resource layer can also develop on its own without impacting on the operational layer provided that the service interfaces are untouched. NAF version 4 places services directly underneath the strategic layer and assumes that services are created directly as a result of strategic decisions. (NATO, 2020)

The Example Model

Problem Statement. A young enterprise architect notices that Valentine’s Day is fast approaching and wants to do something that will make his wife feel happy and appreciated. Being a romantic person, he immediately starts to build a UAF model of the stakeholder wishes, requirements, and detailed views of what he calls the Happiness Enterprise for ordering and delivering flowers. He wants to define a set of services that will provide him the most flexibility for the eventual enterprise. So, he makes sure that he has full traceability in his model. This simple example will help to illustrate how services views fit in with the rest of the architecture and how they can be used to define requirements for their implementing systems. Figure 1 shows the Operational concepts of the Happiness Enterprise.

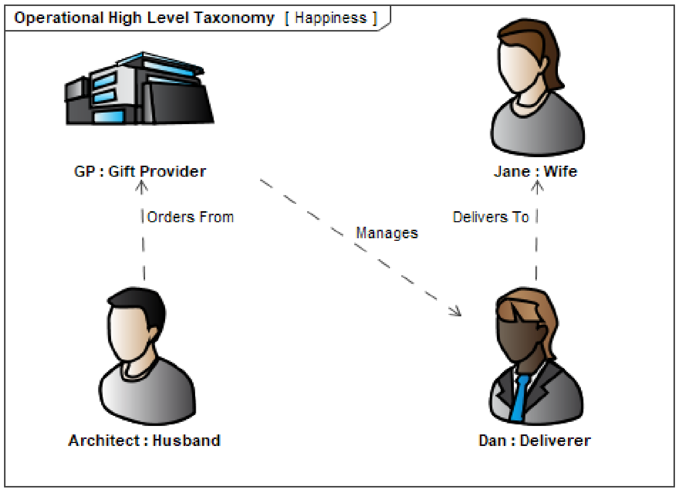
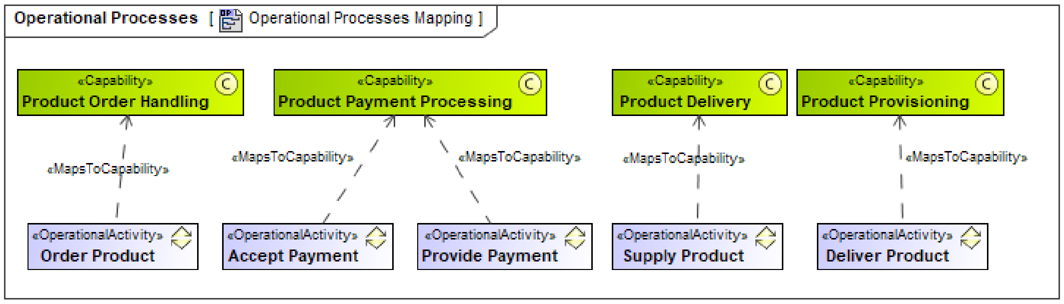


Figure 1: Happiness Enterprise Concept Diagram

Enterprise Capability and Processes

For reasons of brevity, the initial diagrams and architecture have been omitted in order to concentrate on the services view. Figure 2 shows an initial set of capabilities and the operational activities that map to them. These will form the basis of the services to be defined in the architecture.



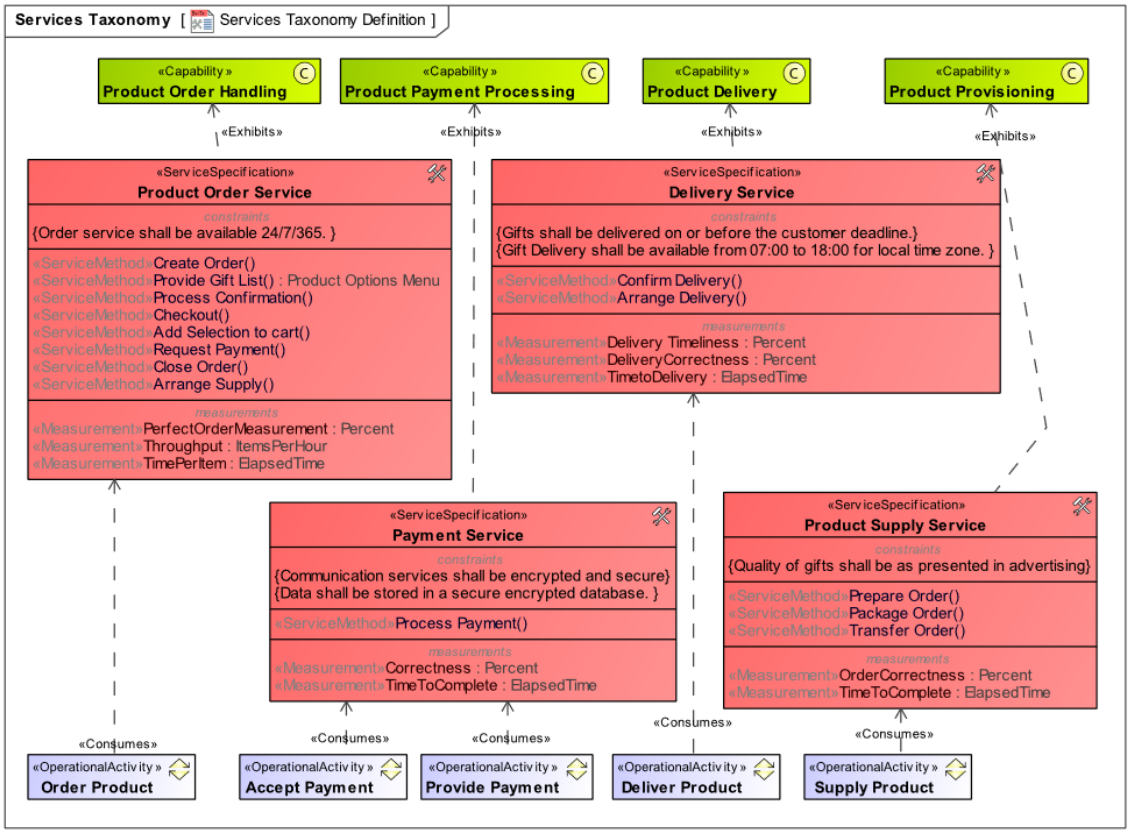
Operational Processes [Operational Processes Mapping]
«Capability» Product Order Handling
«Capability» Product Payment Processing
«Capability» Product Delivery
«Capability» Product Provisioning
«MapsToCapability»
«MapsToCapability»
«MapsToCapability»
«MapsToCapability»
«MapsToCapability»
«OperationalActivity» Order Product
«OperationalActivity» Accept Payment
«OperationalActivity» Provide Payment
«OperationalActivity» Supply Product
«OperationalActivity» Deliver Product

Figure 2: Happiness Enterprise Capabilities and Operational Activities

Figure 2 shows the Product Order, Product Payment, Product Delivery, and Product Provision capabilities. Capabilities define the ability to achieve a desired result. The names of the capabilities are easily understandable. From these capabilities, Operational Activities are created/derived from the capabilities. These map to the business processes that will be further decomposed or expressed as activity diagrams (not shown for reasons of brevity).

Sv-Tx Service Taxonomy

Figure 3 shows the Service Taxonomy which defines service specifications and required metrics.



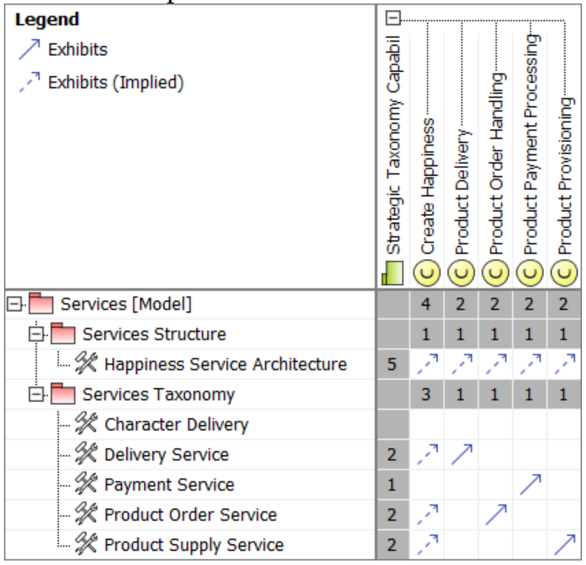
Services Taxonomy [Services Taxonomy Definition]
«Capability» Product Order Handling
«Capability» Product Payment Processing
«Capability» Product Delivery
«Capability» Product Provisioning
«Exhibits»
«Exhibits»
«Exhibits»
«Exhibits»
«ServiceSpecification» Product Order Service
constraints
{Order service shall be available 24/7/365. }
«ServiceMethod»Create Order()
«ServiceMethod»Provide Gift List() : Product Options Menu
«ServiceMethod»Process Confirmation()
«ServiceMethod»Checkout()
«ServiceMethod»Add Selection to cart()
«ServiceMethod»Request Payment()
«ServiceMethod»Close Order()
«ServiceMethod»Arrange Supply()
measurements
«Measurement»PerfectOrderMeasurement : Percent
«Measurement»Throughput : ItemsPerHour
«Measurement»TimePerItem : ElapsedTime
«ServiceSpecification» Delivery Service
constraints
{Gifts shall be delivered on or before the customer deadline.}
{Gift Delivery shall be available from 07:00 to 18:00 for local time zone. }
«ServiceMethod»Confirm Delivery()
«ServiceMethod»Arrange Delivery()
measurements
«Measurement»Delivery Timeliness : Percent
«Measurement»DeliveryCorrectness : Percent
«Measurement»TimetoDelivery : ElapsedTime
«ServiceSpecification» Payment Service
constraints
{Communication services shall be encrypted and secure}
{Data shall be stored in a secure encrypted database. }
«ServiceMethod»Process Payment()
measurements
«Measurement»Correctness : Percent
«Measurement»TimeToComplete : ElapsedTime
«ServiceSpecification» Product Supply Service
constraints
{Quality of gifts shall be as presented in advertising}
«ServiceMethod»Prepare Order()
«ServiceMethod»Package Order()
«ServiceMethod»Transfer Order()
measurements
«Measurement»OrderCorrectness : Percent
«Measurement»TimeToComplete : ElapsedTime
«Consumes»
«Consumes»
«Consumes»
«Consumes»
«Consumes»
«OperationalActivity» Order Product
«OperationalActivity» Accept Payment
«OperationalActivity» Provide Payment
«OperationalActivity» Deliver Product
«OperationalActivity» Supply Product

Figure 3: Services Taxonomy Diagram

Figure 3 shows the taxonomy of service specifications and the expectations of how well those services are provided or required. The Sv-Tx view specifies the hierarchy of services as well as the relationships between them. Services are not limited to internal system functions and can include Human Computer Interface (HCI) and Graphical User Interface (GUI) functions or functions that consume or produce service data to or from service functions. The external service data providers and consumers can be used to represent the human that interacts with the service. Figure 3 shows the services within the Happiness enterprise. Figure 3 shows the previously defined capabilities and operational activities. From these we have defined four service specifications. These are the Product Order Service, the Payment Service, The Product Supply Service and the Delivery Service. These are the basis of the service architecture that will define how the Happiness Enterprise will enable capabilities and business processes. Service methods owned by the services are also shown. The Operational Activities consume/map to services and the services exhibit capabilities.

Sv-Tr Capability to Service Mapping

UAF has several built-in reports defining the traceability between and within views. The Sv-Tr shows the traceability between capabilities and the service specifications that support them. These reports can also depict the mapping of service specifications to operational activities and how service specifications contribute to the achievement of a capability. Figure 4 shows which services contribute to the achievement of the capabilities.


Legend

↗	Exhibits
⇢	Exhibits (Implied)

	Strategic Taxonomy Capabil	Create Happiness	Product Delivery	Product Order Handling	Product Payment Processing	Product Provisioning
Services [Model]		4	2	2	2	2
Services Structure		1	1	1	1	1
Happiness Service Architecture	5	⇢	⇢	⇢	⇢	⇢
Services Taxonomy		3	1	1	1	1
Character Delivery						
Delivery Service	2	⇢	↗			
Payment Service	1				↗	
Product Order Service	2	⇢		↗		
Product Supply Service	2	⇢				↗

Figure 4 – Sv-Tr

Sv-Tx Service Interface Specifications

A service interface is a published interface used to invoke a service. An interface can be implemented using any number of technologies. Service interface definitions implement a service-oriented architecture to achieve interoperability among applications across a varied base of underlying and changing technologies. Many UAF resource elements can provide and consume services. Specifying the interface for the service provides a means of determining compatibility between service consumers and providers. These service interfaces will be mapped to resource interfaces in the same way that resource elements such as systems and capability configurations will be mapped to service specifications. Figure 5 defines the interfaces that will provide access to the services and those required by services.

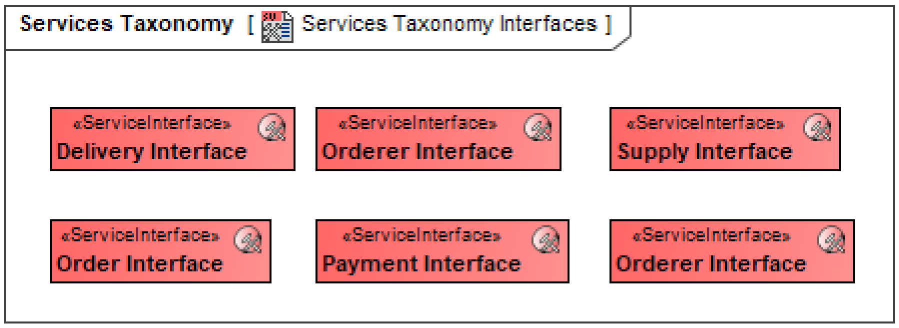

Services Taxonomy [Services Taxonomy Interfaces]
«ServiceInterface» Delivery Interface
«ServiceInterface» Orderer Interface
«ServiceInterface» Supply Interface
«ServiceInterface» Order Interface
«ServiceInterface» Payment Interface
«ServiceInterface» Orderer Interface

Figure 5: Services Interfaces

The Delivery Interface and others describe how services interact with one another as well as interfaces that implementing resources must provide.

Sv-Sr Services Internal Connectivity

The Service Internal Connectivity view demonstrate a combination of services required to exhibit one or more capabilities. This shows the composition of services and how services are combined into a higher-level service required to exhibit a capability or support an operational activity. This also shows how the services are accessed and cooperate to support the capabilities. Figure 6 shows the interactions between the services.

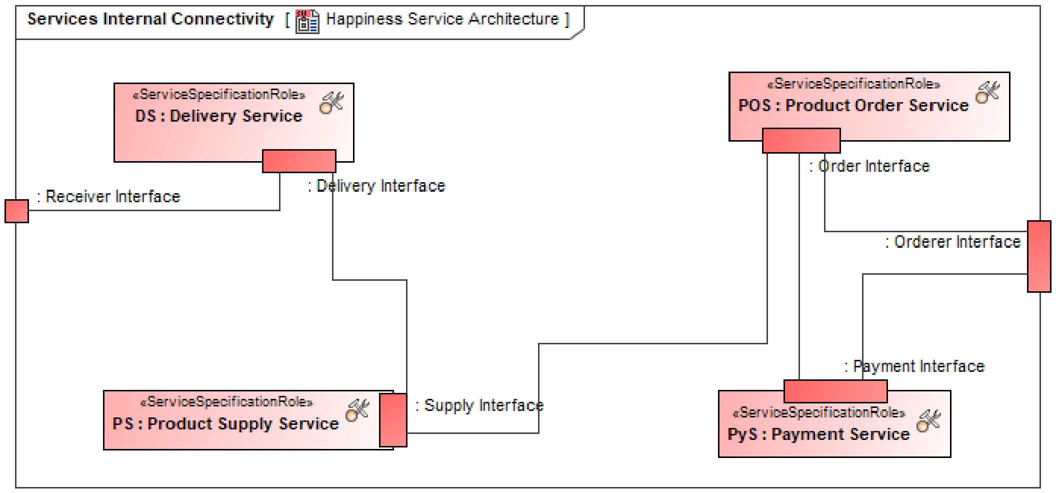

Services Internal Connectivity [Happiness Service Architecture]
«ServiceSpecificationRole» DS : Delivery Service
«ServiceSpecificationRole» POS : Product Order Service
: Receiver Interface
: Delivery Interface
: Order Interface
: Orderer Interface
: Payment Interface
: Supply Interface
«ServiceSpecificationRole» PS : Product Supply Service
«ServiceSpecificationRole» PyS : Payment Service

Figure 6 Happiness Services Internal Connectivity

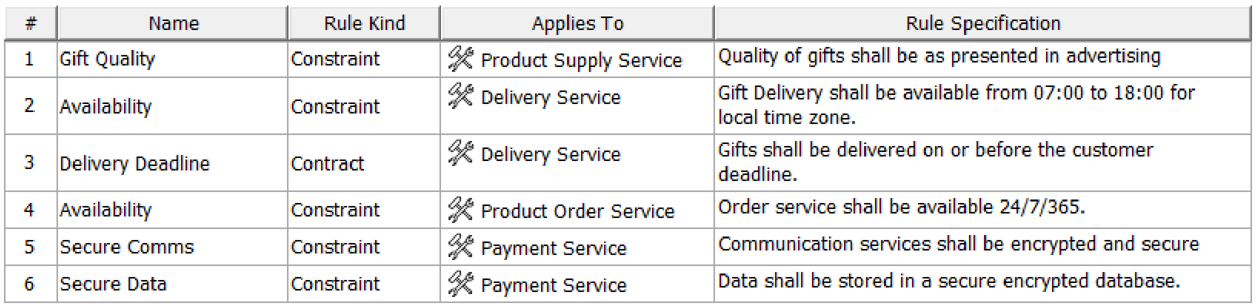
An architecture such as this should be handled cautiously. Since the services are consumed by the operational activities, they are activated from within the activities. This would imply that since they are totally unaware of one another, they are just consumed. In UAF 1.1 the concept of a service architecture does not exist, however this will be changed in the upcoming version 1.2. The handling there would assume that a service can be built up of smaller services, i.e. if we want to create a happiness service with its interfaces, we could build it up from the smaller services. It would still need an interface towards the consumer, however. There may be several consumers of the service and some form of consumption could be initiated by the service itself. (As this will be realized by a commercial organization, one hopes that they will have many customers.) The service would be invoked by the person that wants to surprise his wife with flowers and the interface could give him options to do this. Payment Services can also be invoked by the service.

When showing a set of services interacting, one needs to remember that services are supposed to be loosely coupled and that as far as the individual service is concerned the only external entity is an unknown consumer, i.e. it has no internal awareness of the outside world. The orchestration performed when defining how a set of services interact is owned by the service that is built up from these services. Since these services are specifications, the orchestration is also just a way of specifying the more complex service based on services that already exist thus avoiding the need to specify internal parts of an already defined service specification. This means that an implementation of the complete service could occur in other ways. There is no requirement that the implemented total service include the implementations of the service specifications that were used to specify the total service. The implementation is up to the service provider.

Sv-Ct Service Policies

The Service Constraints view defines service policies that apply to implementations of service specifications. It specifies traditional textual service policies that are constraints on the way that service specifications are implemented within resources. The addition of SysML parametrics provides a computational means of defining service policies across the enterprise or within a specific service specification. The Sv-Ct defines constraints that must be adhered to by Consumers and Providers of the Services via Service Policies. This also provides a means of performing trade-off analysis of the possible service providers. As a minimum it defines a set of criteria to determine whether the service provider meets the provision requirements defined by the constraints. Table 1 shows a sample of the services and their associated service policies.

Table 1 – Sv-Ct Service Policies



#	Name	Rule Kind	Applies To	Rule Specification
1	Gift Quality	Constraint	Product Supply Service	Quality of gifts shall be as presented in advertising
2	Availability	Constraint	Delivery Service	Gift Delivery shall be available from 07:00 to 18:00 for local time zone.
3	Delivery Deadline	Contract	Delivery Service	Gifts shall be delivered on or before the customer deadline.
4	Availability	Constraint	Product Order Service	Order service shall be available 24/7/365.
5	Secure Comms	Constraint	Payment Service	Communication services shall be encrypted and secure
6	Secure Data	Constraint	Payment Service	Data shall be stored in a secure encrypted database.

Sv-Is Service Interactions

The Service Interactions View defines the behavior of a service specification or set of specifications in terms of expected time-ordered examination of the interactions between service roles (exemplified in Figure 6). It also specifies how service roles interact with each other, service providers and consumers, and the sequence and dependencies of those interactions. It also shows the invocation of the Service Methods to realize the services. Figure 7 shows a subset of the interactions between the different services from the order request from the Husband to the delivery of flowers to the Wife. The interactions between the services are events and signals. The message to self is the invocation of a service method. Service Methods are behaviors owned by the service to implement the service and are types of operations. Typically, architects will use service functions (activities) or service methods (operations) to define and implement the behaviors as modeling both could be redundant. Service functions provide a simple activity diagram format to define the order of the activities and the required inputs and outputs. Service methods also define required and provided functionality with required inputs and outputs. Showing the order of invocation of the service methods on an interaction diagram is problematic if only a single service specification is required. (There would only be a single lifeline on the diagram, which looks silly.)

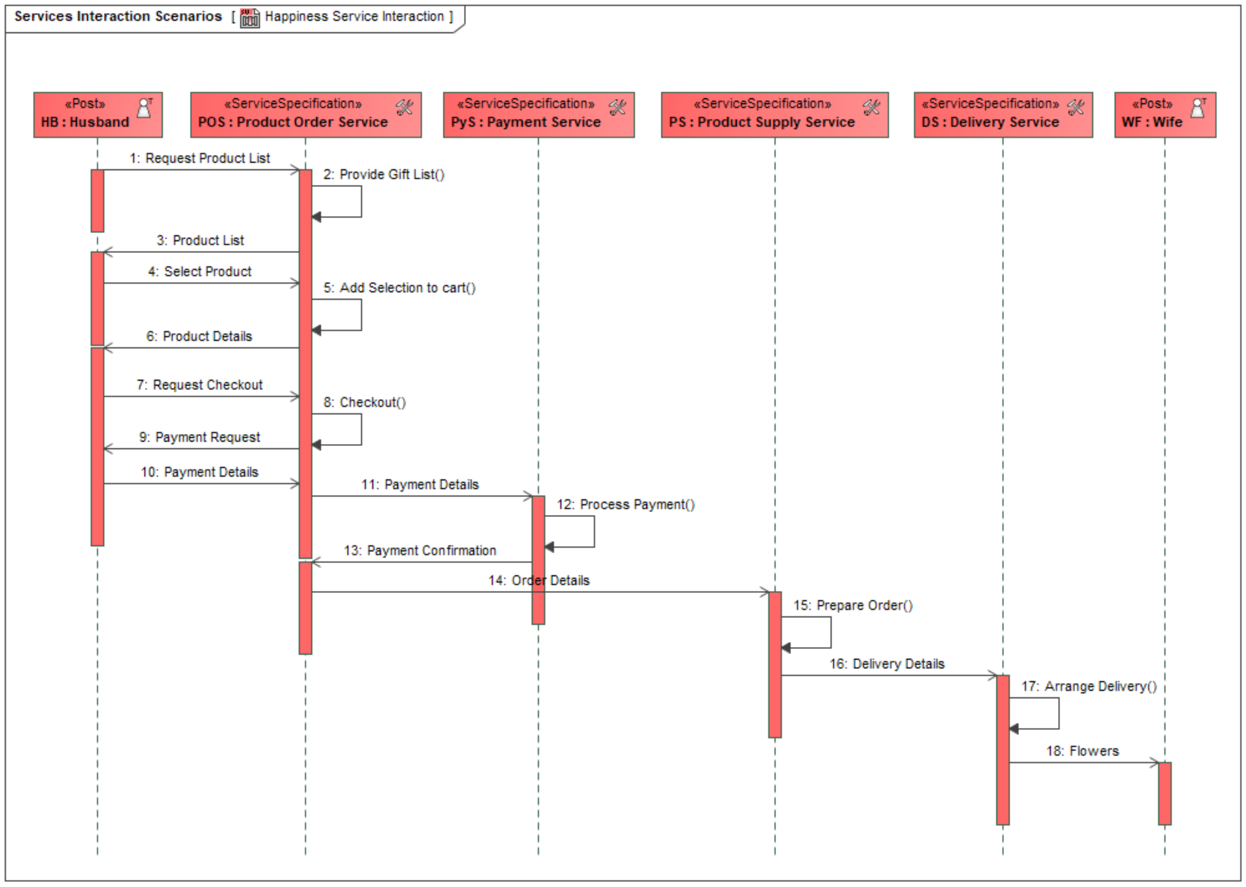


Figure 7 – Sv-Is Service Interaction Diagram

Both service functions and service methods can be show on state diagrams. The example in this paper uses both to demonstrate the different ways of modeling behavior.

Sv-St Service States

The Service States View defines the behavior of a service specification in terms of state-based behavior. Figure 8 shows the state diagram describing the state-based behavior of the Product Order Service. The Sv-St defines behavioral constraints that must be adhered to by Consumers and Providers of the Services. Specifically, it defines the state-based behavior of the service defining the states, transitions between those states, the events that cause those transitions to take place and behaviors within those states. Figure 8 shows the states of the Product Order Service from order selection and creation through delivery. From the point of view of the customer, the order service will track the order until complete delivered and notify the customer.

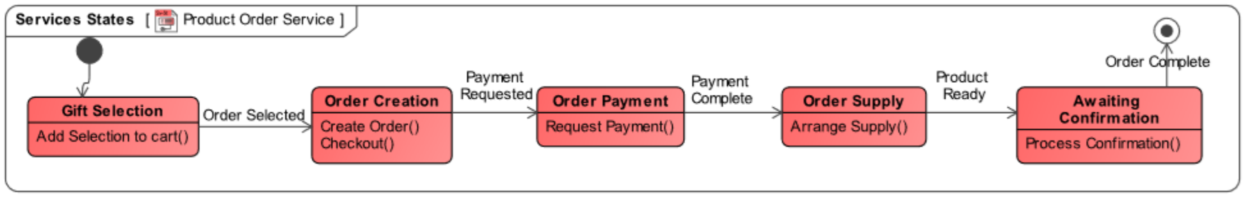


Figure 8 – Sv-St Service State Diagram

Other behaviors could be added such as a follow up questionnaire or satisfaction survey as is typical when ordering products. Note that this service is aware of the state of the order as it progresses, but that it is the other services that perform the functions. Also note that it is the Product Supply Service that arranges the delivery. Responsibility could have been assigned to the Product Order Service, but the decision was made to outsource this to a local supplier. This is important when determining the resource that will implement the service and whether it is done correctly.

Sv-Pr Service Processes

The Service Processes View defines the behavior of a service in terms of the service functions it is expected to support and the resource functions that implement them. They provide detailed information regarding the allocation of service functions to service specifications, and data flows between service functions. Two diagrams are used to define Service Functions – the Service Processes Definition Diagram (a SysML Block Definition Diagram) and a Services Process Flow Diagram (A SysML Activity Diagram). Figure 9 defines the Service Functions that the service implementation performs and how they map to resource functions. In this example, the services are responsible for product delivery, payment, etc., which are shown as service functions. Service functions can be difficult to map properly to other parts such as the interactions defined by the service itself. This is due to their granularity as well as the interactive nature of services.

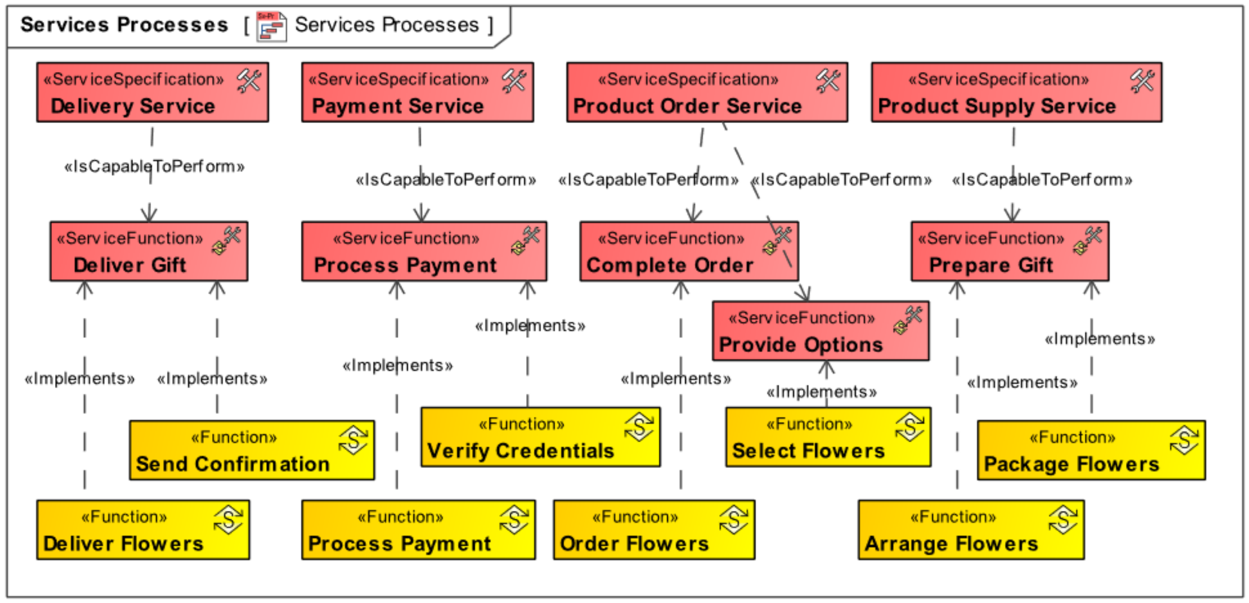

Services Processes [Services Processes]
«ServiceSpecification» Delivery Service
«ServiceSpecification» Payment Service
«ServiceSpecification» Product Order Service
«ServiceSpecification» Product Supply Service
«IsCapableToPerform»
«ServiceFunction» Deliver Gift
«ServiceFunction» Process Payment
«ServiceFunction» Complete Order
«ServiceFunction» Prepare Gift
«ServiceFunction» Provide Options
«Implements»
«Function» Send Confirmation
«Function» Verify Credentials
«Function» Select Flowers
«Function» Package Flowers
«Function» Deliver Flowers
«Function» Process Payment
«Function» Order Flowers
«Function» Arrange Flowers

Figure 9 Service Functions Mapped to Resource Functions and Service Specifications

A few of the service functions and resource functions are shown as examples. These service functions would be further decomposed to sub-functions and expressed in activity diagrams. The functions are implemented by the resource functions as shown in Figure 9. Figure 10 shows a simplified activity diagram. This would be further refined to show order cancel, payment request interactions, as well as additional interactions with the person making the order. At this stage it simply specifies the required service functions and required data.

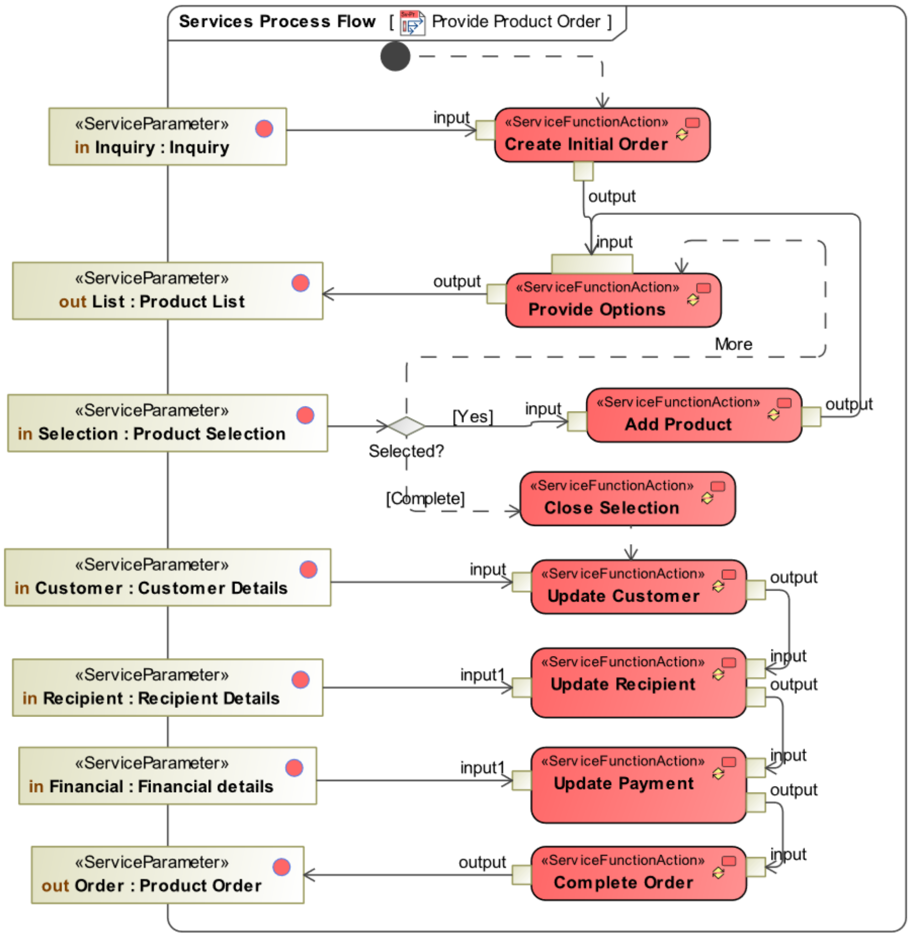

Services Process Flow [Provide Product Order]
«ServiceParameter» in Inquiry : Inquiry
«ServiceParameter» out List : Product List
«ServiceParameter» in Selection : Product Selection
«ServiceParameter» in Customer : Customer Details
«ServiceParameter» in Recipient : Recipient Details
«ServiceParameter» in Financial : Financial details
«ServiceParameter» out Order : Product Order
«ServiceFunctionAction» Create Initial Order
«ServiceFunctionAction» Provide Options
«ServiceFunctionAction» Add Product
«ServiceFunctionAction» Close Selection
«ServiceFunctionAction» Update Customer
«ServiceFunctionAction» Update Recipient
«ServiceFunctionAction» Update Payment
«ServiceFunctionAction» Complete Order
input
output
input
output
More
[Yes]
input
output
Selected?
[Complete]
input
output
input1
input
output
input1
input
output
output
input

Figure 10 Provide Product Order Service Process

Services Roadmap

The Services Roadmap provides an overview of how a service specification changes over time. It shows the combination of several service specifications mapped against a timeline. Figure 10 shows the Services for implementing Bob’s Flowers.

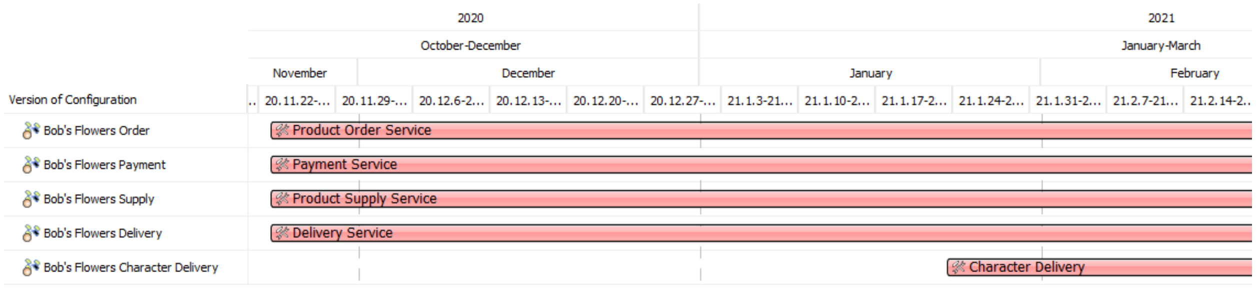
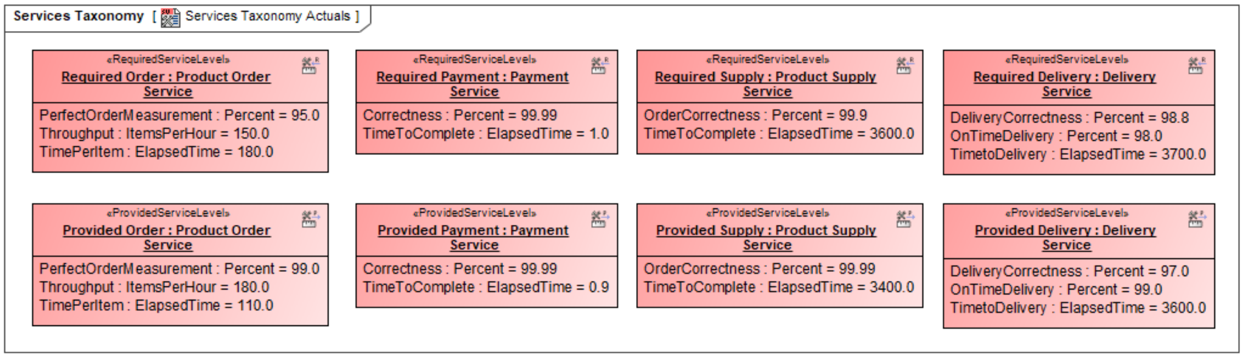


Figure 10 – Services Roadmap for Bob’s Flowers

Bob’s Flowers is a simple flower shop with all the services provided by the employees. For a large enterprise this would show multiple service deployment, with the addition of new services being deployed over time. In this case, Bob’s Flowers has decided to roll out what is known as a character delivery service. This is a common service provided with popular costumed characters in addition to other products for parties or special occasions. This type of upsell would include helium balloon bouquets, cards photo prints, etc.

Service Constraints

This view identifies measurable properties that can be used to support analysis such as KPIs, MoEs, TPIs etc. It shows the measurable properties of something in the physical world, expressed in amounts of a unit of measure that can be associated with any element in the architecture. Figure 11 shows a set of Required Service Levels and Provided Service Levels.


Services Taxonomy [Services Taxonomy Actuals]
«RequiredServiceLevel»
Required Order : Product Order Service
«RequiredServiceLevel»
Required Payment : Payment Service
«RequiredServiceLevel»
Required Supply : Product Supply Service
«RequiredServiceLevel»
Required Delivery : Delivery Service
«ProvidedServiceLevel»
Provided Order : Product Order Service
«ProvidedServiceLevel»
Provided Payment : Payment Service
«ProvidedServiceLevel»
Provided Supply : Product Supply Service
«ProvidedServiceLevel»
Provided Delivery : Delivery Service

Required Order : Product Order Service
PerfectOrderMeasurement : Percent = 95.0
Throughput : ItemsPerHour = 150.0
TimePerItem : ElapsedTime = 180.0

Required Payment : Payment Service
Correctness : Percent = 99.99
TimeToComplete : ElapsedTime = 1.0

Required Supply : Product Supply Service
OrderCorrectness : Percent = 99.9
TimeToComplete : ElapsedTime = 3600.0

Required Delivery : Delivery Service
DeliveryCorrectness : Percent = 98.8
OnTimeDelivery : Percent = 98.0
TimetoDelivery : ElapsedTime = 3700.0

Provided Order : Product Order Service
PerfectOrderMeasurement : Percent = 99.0
Throughput : ItemsPerHour = 180.0
TimePerItem : ElapsedTime = 110.0

Provided Payment : Payment Service
Correctness : Percent = 99.99
TimeToComplete : ElapsedTime = 0.9

Provided Supply : Product Supply Service
OrderCorrectness : Percent = 99.99
TimeToComplete : ElapsedTime = 3400.0

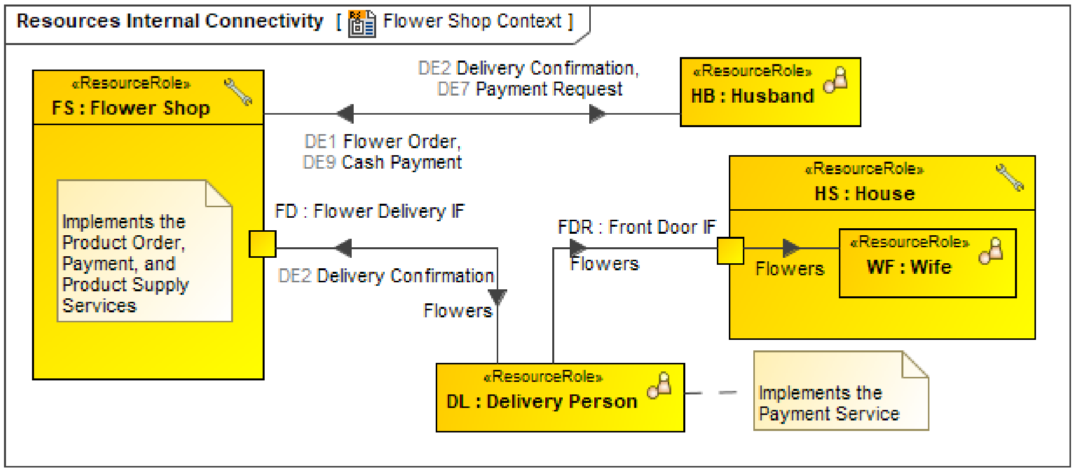
Provided Delivery : Delivery Service
DeliveryCorrectness : Percent = 97.0
OnTimeDelivery : Percent = 99.0
TimetoDelivery : ElapsedTime = 3600.0

Figure 11 – Required and Provided Services

A Required Service Level is a set of parameters from the client to a service provider describing their service expectations. A service provider prepares a service level agreement based on the requirements from the customer. For example: A customer may require a system be operational for 99.95% of the year excluding maintenance. For trade-off analysis, these provide a means of determining which implementation meets the customer's expectations and recording provided values.

Service Implementation in Resources

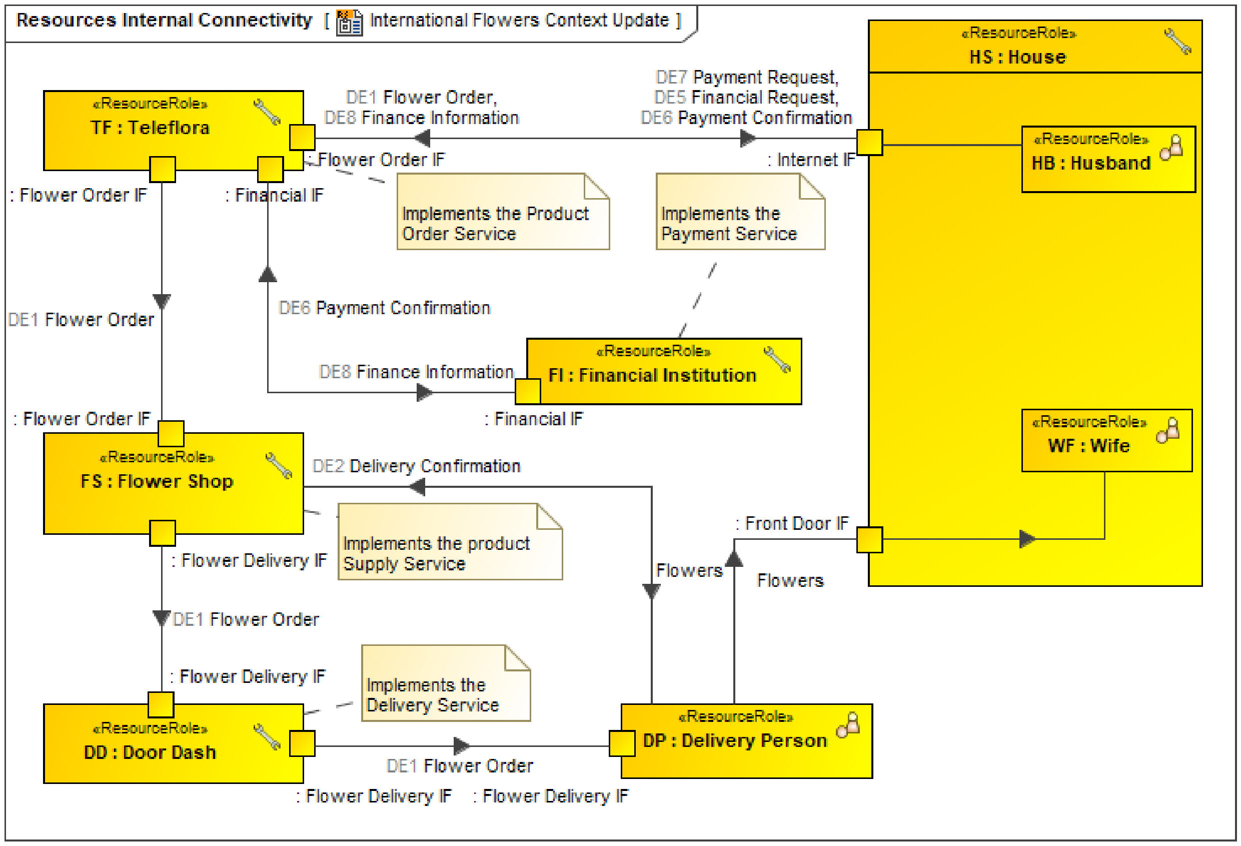
The Services Domain View describes services and their interconnections that provide or support capabilities. The Service Models associate service resources to the operational and capability requirements. These resources support the operational activities and facilitate the exchange of information. The relationship between architectural data elements across the Services Viewpoint to the Operational Viewpoint and Capability Viewpoint can be exemplified as services are procured and fielded to support the operations and capabilities of organizations. The structural and behavioral models in the operational and service views allow architects and stakeholders to quickly ascertain which functions are carried out by humans and/or systems for each alternative specification. This allows engineers to carry out tradeoff analysis based on risk, cost, reliability, etc. Services can be implemented in a variety of different ways using systems, software, people, organizations, natural resources, etc. Figure 12 below shows a simple implementation of the various services defined so far. In this example, the simple flower store implements the Product Order, Payment, and Product Supply services. The Delivery Service is implemented by the Delivery person. This represents the most basic form of service provisions making use of people and processes. This of course assumes that the implementation of the services complies with the service policies defined in the constraints views.



Resources Internal Connectivity [Flower Shop Context]
«ResourceRole» FS : Flower Shop
DE2 Delivery Confirmation, DE7 Payment Request
«ResourceRole» HB : Husband
DE1 Flower Order, DE9 Cash Payment
«ResourceRole» HS : House
Implements the Product Order, Payment, and Product Supply Services
FD : Flower Delivery IF
FDR : Front Door IF
«ResourceRole» WF : Wife
Flowers
Flowers
DE2 Delivery Confirmation
Flowers
«ResourceRole» DL : Delivery Person
Implements the Payment Service

Figure 12: Flower Shop Context

In this case, the Husband physically goes to the Flower Shop. The flowers are selected by the Husband and the owner or employee prepares the flowers. The Husband receives the flowers and hands over a cash payment. The owner gives the flowers to the delivery person who delivers them to the Wife. The owner informs the Husband that the flowers have been delivered and all is well. Figure 13 defines a more complex implementation of the services architecture.



Resources Internal Connectivity [International Flowers Context Update]
«ResourceRole» HS : House
«ResourceRole» TF : Teleflora
DE1 Flower Order, DE8 Finance Information
DE7 Payment Request, DE5 Financial Request, DE6 Payment Confirmation
«ResourceRole» HB : Husband
: Flower Order IF
: Internet IF
: Flower Order IF
: Financial IF
Implements the Product Order Service
Implements the Payment Service
DE6 Payment Confirmation
DE1 Flower Order
DE8 Finance Information
«ResourceRole» FI : Financial Institution
: Flower Order IF
: Financial IF
«ResourceRole» WF : Wife
«ResourceRole» FS : Flower Shop
DE2 Delivery Confirmation
Implements the product Supply Service
: Front Door IF
: Flower Delivery IF
Flowers
Flowers
DE1 Flower Order
: Flower Delivery IF
Implements the Delivery Service
«ResourceRole» DD : Door Dash
«ResourceRole» DP : Delivery Person
DE1 Flower Order
: Flower Delivery IF
: Flower Delivery IF

Figure 13: International Flower Company Context

In this architecture the services are each implemented by complex systems such as international corporations and involves the outsourcing of services. The Product Order service is provided by Teleflora. Aspects of the service could be outsourced such as the web presence, call services, and internationalization. The Financial Service provides the Payment service. The Product Supply service is provided by Door Dash which further outsources to a Delivery Person as a Gig Worker, and the Flower Store supplies the Product Supply Service.

The UAF Services Views and MOSA

Modular Open Systems Approach (MOSA) is an integrated business and technical strategy for assessment and implementation of open systems in the DoD. An open system is a system that employs modular design tenets, uses widely supported and consensus-based standards for its key Interfaces, and is subject to validation and verification, including test and evaluation, to ensure the openness of its key interfaces. The Office of the Deputy Assistant Secretary of Defense (ODASD) describes the DoD vision of MOSA and its benefits. "The Department of Defense's (DoD) MOSA is to design systems with highly cohesive, loosely coupled, and severable modules that can be competed separately and acquired from independent vendors. This approach allows the Department to acquire warfighting capabilities, including systems, subsystems, software components, and services, with more flexibility and competition. MOSA implies a structure in which system interfaces share common, widely accepted standards, with which conformance can be verified. The DoD is actively pursuing MOSA in the life-cycle activities of its major defense acquisition programs (MDAP) and major automated information systems (MAIS), in large part due to the rapid evolution in technology and threats that require much faster cycle time for fielding and modifying warfighting capabilities." (ODASD, 2004) This is part of a comprehensive systems engineering strategy. MOSA can accelerate and simplify the incremental delivery of new capabilities into systems, enhance competition, facilitate technology refresh, incorporate innovation, enable cost savings/cost avoidance and improve interoperability. Let's look at MOSA systems characteristics and principles and how the UAF services views addresses them

- Modular design that is cohesive, encapsulated, self-contained, and highly binned
  The object-oriented mechanisms underpinning SysML and the UAF are based on the principles of modular design, separation of concerns, encapsulation, and well-defined, reusable components. Systems, services, architectures, software, etc. can be defined and reused throughout the architecture. Making use of services views provides a means of sharing between architectures, across projects and the systems.

- Key interface definition
  UAF architectures define the required behavior/functionality interfaces leading to system-to-system interfaces, dependencies and interactions with other systems and within the subsystems, define interaction types: data, physical, logical, electrical, etc., define logical interface requirements, define interaction performance characteristics, allocation of logical to physical interfaces, service definitions, policies, standards and constraints.

- Design requirements (e.g., mandated open standards and protocols)
  Requirements views provide a means of integrating requirements into the model to demonstrate traceability and compliance. The Standards views define standards of all sorts

including open standards, protocols, operating procedures, processes, etc. System interfaces include protocols, and protocol stacks. The UAF itself is an open mandated standard.

- Architectural attributes (e.g., need for an adaptable, upgradeable and reconfigurable system architecture)
  UAF service views define implementation independent specifications for system capabilities. Multiple system solutions can be evaluated to determine which meets the requirements in the best way. Operational system components can be replaced based on well defined interfaces, functional specifications and performance requirements.

- Conformance certification
  Standards compliance can be traced to all elements in the architecture and reports generated to demonstrate compliance. Requirements traceability reports demonstrate that system requirements have been satisfied and verified. Integrated systems test and analysis tools certify performance and functional requirements compliance.

Further Work on the UAF Services Views

UAF is currently undergoing a revision that will result in the creation of UAF version 1.2. This version will still be based on the underlying use of SysML version 1.7. Since the connection between UAF and SysML will be maintained, UAF version 2 will presumably be needed once SysML version 2 has been completed. In the revision work for 1.2 the handling of services is being given an overhaul based on comments received from several parties. Below are several issues that are being dealt with in UAF version 1.2:

- The concept of a service contract will be introduced that will act as a constraint on a selected set of operational connectors in between operational elements, making it possible to designate specific parts of the operational exchanges as a specific contract.
- In order to clarify the distinction between a specification and a realization of a service, the concept of Resource Service will be introduced as well as a resource service interface.
- As part of the creation of a service architecture element, the concept of service exchanges will be introduced.
- There are other more detailed changes that are being considered.

Conclusions

The use of services and SOA has fulfilled its original purpose of making it possible to act as an intermediary between the operational layer and the resources layer within an architecture. As long as the interfaces that the operational layer makes use of are not touched, the operational logic can be modified, and the operational approach be changed. In case these changes require the creation of additional services, such services can be specified and implemented by the resource layer. In the same fashion, the implementation of a service can also change as technology develops provided that the interfaces are not changed. This enables the operations to continue without being affected by the realization modification.

The service concept still causes a lot of confusion and it is hoped that the discussions in this paper will help to alleviate this. It is hoped that the additions made in UAF version 1.2 will make it even easier to define a proper use of the services concept as a part of an enterprise architecture.

References

DoDAF DoD CIO, 2012, DoD Architecture Framework Version 2.02, DoD Deputy Chief Information Officer, Available online at http://dodcio.defense.gov/dodaf20/dodaf20_pes.aspx, accessed June, 2014.

Microsoft, 2006, Chapter 1: Service Oriented Architecture, (SOA) https://msdn.microsoft.com/en-us/library/bb833022.aspx

MOD Architectural Framework, Version 1.2, 2020, Office of Public Sector Information, https://www.gov.uk/guidance/mod-architecture-framework/

NATO Architecture Framework Version 4, January 2018, Architecture Capability Team Consultation, Command & Control Board

ODASD, 2004, Modular Open Systems Approach (MOSA) available at https://www.acq.osd.mil/log/MPP/ats_opensystems.html

Object Management Group (OMG). 2013. OMG2013-08-04:2013. Unified Profile for DoDAF/MODAF (UPDM) V2.1, http://www.omg.org/spec/UPDM/2.1/PDF
2012. OMG2012-06-01.OMG Systems Modeling Language (OMG SysML™), V1.3, http://www.omg.org/spec/SysML/1.3/PDF/.

Object Management Group (OMG), 2019, The Unified Architecture Framework, (UAF) Available from https://www.omg.org/spec/UAF

The SOA Manifesto, 2009, SOA Manifesto, Available at: http://www.soa-manifesto.org/

The Open Group, 2007, "Service-Oriented Architecture Standards - The Open Group". Available from: www.opengroup.org/standards/soa

Biography

Lars-Olof Kihlström. Lars-Olof Kihlström is a principal consultant at Syntell AB where he has worked since 2013, primarily in the area of MBSE. He has been a core member of the UAF group within the OMG since its start as the UPDM group. He was involved in the development of NAF as well as MODAF. He has worked with modelling in a variety of domains such as telecommunications, automotive, defence as well as financial systems. He is specifically interested in models that can be used to analyze the behavior of system of systems.

Matthew Hause. Matthew Hause is a principal consultant at SSI, a member of the UAF group, and a member of the OMG SysML specification team. He has been developing multi-national complex systems for almost 40 years as a systems and software engineer. He worked in the power systems industry, command and control systems, process control, SCADA, military systems, and many other areas. His role at SSI includes consulting, mentoring, standards development, specification of the UAF profile and training.