Project title:

Networked **E**nterprise trans**F**ormation and resource management in **F**uture internet enabled **I**nnovation **C**loud**S**

Acronym: NEFFICS Project No: 258076

THEME: ICT-2009.1.3
Internet of Things and Enterprise Environments

VDML Manufacturing Use Case

OMG document
bmi/2012-11-10

Based on NEFFICS Work Package 3 – Deliverable D3.3

www.neffics.eu

Date: 2012.10.30

Version: 1.0

Leading partner:	CORDYS
Editor(s):	Henk de Man
Author(s):	Henk de Man, Arne J. Berre
Dissemination level:	Public

Executive Summary

This report describes a method approcah for possible use of VDML, and shows it on a use case for VDML (Value Delivery Modeling Language) in manufacturing. VDML is a modeling language developed in response to an RFP by the Object Management Group , as described in the document: Value Delivery Modeling Language 1.0, OMG document bmi/2012-11-07.

The purpose of this use case example is to describe a method for value modeling, and to show the usage of this as an example us of the the VDML metamodel and notation, and to provide an example of how an analyst might use VDML to develop a model for evaluation of a proposed business change.

This use case is focusing on the domain of manufacturing, and is one of the use cases studied in the NEFFICS project, www.neffics.eu

This use case demonstrates the use of VDML for modeling and analysis of a proposed change to an as-is business system. It also demonstrates how the scope of analysis can be limited to consideration of a specific area of concern.

The document starts with first introducing a structured modeling approach for “value delivery models”, based on the Value Delivery Modeling Language (VDML). The report then applies this to a manufacturing use case.

Table of contents

EXECUTIVE SUMMARY **2**

TABLE OF CONTENTS **3**

1 INTRODUCTION **4**

1.1 OBJECTIVE OF THIS DOCUMENT 4
1.2 STRUCTURE OF THIS DOCUMENT 4

2 MOTIVATION AND REQUIREMENTS FOR VALUE DELIVERY MODELING **5**

2.1 MAIN MOTIVATION 5
2.2 SPECIFIC REQUIREMENTS FOR VALUE DELIVERY MODELING 8
2.2.1 Business values motivate value delivery models 8
2.2.2 Value delivery model supports business model 8
2.2.3 Value delivery model discovers process and service models 10
2.3 LEVERAGING ESSENTIALS OF EXISTING APPROACHES 13
2.3.1 Value network Analysis (VNA) 13
2.3.2 SCOR- and VRM-based approaches 14
2.3.3 Lean Value Streams 15
2.3.4 Resource-Event-Agent (REA) 15
2.3.5 e3Value 16
2.3.6 Capability analysis 17
2.3.7 Summary of limitations 18
2.3.8 Summary of strengths, taken as further requirements 18
2.4 PROVIDING ADVANCED INNOVATION SUPPORT 19

3 VALUE DELIVERY MODELING LANGUAGE (VDML) **22**

3.1 VDML EXPLAINED 22
3.1.1 Approach to explain VDML 22
3.1.2 Business network and collaboration 24
3.1.3 Organizations and capabilities 30
3.1.4 Activities 39
3.1.5 Value 71
3.1.6 Model data organization and re-use 99
3.2 VDML ADVANCED ELEMENTS 101

4 HOW VDML ADDRESSES REQUIREMENTS **104**

4.1 HOW VDML ADDRESSES SPECIFIC REQUIREMENTS 104
4.2 HOW VDML MEETS MOTIVATION 109

5 CONCLUSION **111**

5.1 SUMMARY OF RESULTS 111
5.2 FUTURE WORK 111

6 REFERENCES **112**

7 GLOSSARY **115**

7.1 JUSTIFICATION OF TERMS AND DEFINITIONS 115
7.2 TERMS AND DEFINITIONS 115

1 Introduction

1.1 Objective of this document

This document is based on deliverable "Value Delivery Model and Methods" (D3.3), as part of work package "Networked Business Value Analysis Models" (WP3) in the NEFFICS project, www.neffics.eu

Its purpose is twofold:

- Introducing a structured modeling approach for "value delivery models", based on the Value Delivery Modeling Language (VDML).
- Providing a detailed application of VDML to a use case. The document explains VDML by means of the use case application. This explanation is concerned with both aspects of the modeling language and aspects of methodology for using it.

The document means to provide understanding to users of VDML, for how VDML, as a structured modeling language, can support them in applying business value analysis and in analyzing and designing the business system to support business models, in the context of business innovation and in discovering and implementing services and processes.

It also means to guide implementers of VDML modeling support designers of VDML-based modeling methodology and to guide business analysts in creating value delivery models and conducting analysis and to create their business designs.

1.2 Structure of this document

This document serves as an example of use of the VDML specification, and should be read together with this.

In chapter 2 we will motivate value delivery modeling and analyze requirements that need to be addressed in order to meet the motivation. Chapter 3 focuses on explaining VDML based on its application to a use case. This use case is a simplified version of a part of the business of NEFFICS partner Vlastuin. In chapter 4 we consider how, given its detailed explanation in chapter 3, VDML addresses requirements as analyzed in chapter 2. The document is concluded in Chapter 5, in which chapter we also outline remaining work, both during NEFFICS, and beyond (such as application for VDML for automated business simulation).

2 Motivation and requirements for value delivery modeling

Section 2.1 provides the motivation behind value delivery modeling. Specific requirements for value delivery modeling are analyzed in 2.2. In section 2.3 some common and existing approaches, relevant in the domain that overlaps with value delivery modeling, are analyzed and some of their "strengths" are considered further requirements for value delivery modeling. This is also for the reason that value delivery modeling should not cause a point of discontinuity for practitioners that adopt such approaches, but should rather broaden their opportunities for business design and analysis, and enable them to be aligned with a more integral, as well as standard and potentially technology-supported modeling approach. Section 2.4 investigates requirements for advanced use of value delivery models, such as to serve as basis for business simulation. It also clarifies the role of value delivery models in the broader context of business innovation.

2.1 Main motivation

Three streams of research in NEFFICS, are all conducted in the context of and focus on innovation (see Figure 1):

- Business value (WP3)
- Business model, and business model innovation (WP4)
- Process and service modeling and management (WP5).

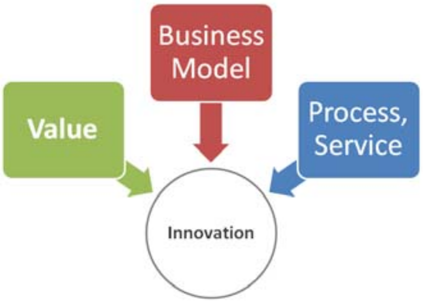


Figure 1. Streams of innovation-related research

Innovation is, amongst others, focused on discovery of new values, and development or improvement of ways how values are created, exchanged and consumed.

This overlaps with what the business model community understands as business model innovation, whereby business models serve as context for and also as subject of innovation.

This is at least suggested by the following definition of "business model", as proposed by Osterwalder and Pigneur (2010): "*A business model describes the rationale of how an organization creates, delivers, and captures value*".

A business model is a description of the rationale of creating value, or, as a part of it, making money. But this description does normally not take the form of a structured model. Osterwalder and Pigneur (2010), Osterwalder, Johnson et al. (2010), and others, introduced frameworks to describe business models, which are merely mental models to think about business models. The business modeling framework, as developed in NEFFICS, as part of WP4, can be found in NEFFICS D4.1 (2011) and NEFFICS D4.3 (2012).

Next to the business modeling community, there is the community of people and practitioners that are involved with process and service innovation. This community has evolved, at least in part, from movements such as business process re-engineering, lean six sigma and business process management (BPM).

Process and service innovation communities are used to describe processes and services, and how they relate or interact, by structured models (we talk about "structured model" here, to avoid the ambiguity of the word "model"). These models are, in fact "engineering representations" or "architectural representations" of how the processes and services work and interact.

Structured models lend themselves well for unambiguous interpretation by both humans and systems, and support, amongst others:

- Automatic validation of model correctness, completeness and consistency.
- Consistent notation, graphical and otherwise.
- Model interoperability and exchange.
- Modularity and re-use of model elements
- Expression- and query-based navigation over models, including model-based calculations
- Change impact analysis, also based on where-used queries.
- Model comparisons
- Model simulation
- Transformation of models into models that represent different viewpoints, e.g. for purpose of model-based automation.
- Life cycle management, including versioning and access control of model elements.

Everything that happens comes from a process, regardless of whether it has been formally defined. Processes are typically involved as services. It is reasonable to assume that the "how" of creation, delivery and capturing of value is substantially about processes and services. Process and service innovation is a significant part of business model innovation therefore.

However, process and service innovation communities have not been able to model how processes and services contribute to "the business", e.g. how processes contribute to value, how value is exchanged and consumed, etc. The engineering representation reaches as far as is required to automate orchestration and choreography (interaction) of processes and services, and has so-far had a technical focus.

This leads to the following observations:

- The "rationale of how an organization creates, delivers, and captures value" cannot yet be analyzed and designed with the same rigor according to which processes and services are analyzed and designed. Frameworks are not yet supported by an "engineering representation" or "architectural representation", based on structured models.
- Analysis and design of process and service models, though based on structured models, does not exploit its actual and ultimate potential, as these models do not clarify how processes and services are part of the "rationale of how an organization creates, delivers, and captures value". This is really a missed opportunity !

In NEFFICS, we intend to better align the areas of value analysis, business model analysis and process and service analysis, and based on this, facilitate a more rigorous business design and business architectural take on business model innovation. We intend to do this by developing a structured modeling approach, based on so-called "value delivery models".

The modeling and analysis area that "value delivery modeling" intends to cover has not been defined so-far. There is no common term available that covers this area, and therefore we pick the term "value delivery modeling". There are existing approaches, such as value chain analysis, value stream analysis, value network analysis, etc., but, as will be analyzed in a later section, none of these is providing sufficient coverage in the area of concern.

Value delivery modeling is about creating value delivery models. We define a value delivery model as "*a model that supports business analysis and design based on evaluation of performance and stakeholder satisfaction achieved through the activities and interactions of people and organizations using business capabilities to apply resources and deliver stakeholder values*". The meaning of the various terms that are used in this definition will become clear during discussions throughout this document.

Figure 2 indicates how value delivery models can serve to bridge the gap between structured representations of processes in process models, and high-level abstractions of business models in business model frameworks.

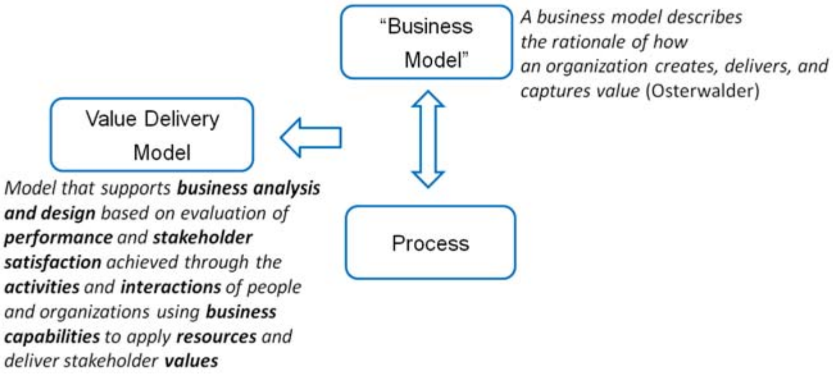

"Business Model"
A business model describes the rationale of how an organization creates, delivers, and captures value (Osterwalder)
Value Delivery Model
Model that supports **business analysis and design** based on evaluation of **performance** and **stakeholder satisfaction** achieved through the **activities** and **interactions** of people and organizations using **business capabilities** to apply **resources** and deliver stakeholder **values**
Process

Figure 2. Value delivery modeling bridging between "business model" and process

A more detailed positioning of value delivery models is presented in Figure 3.

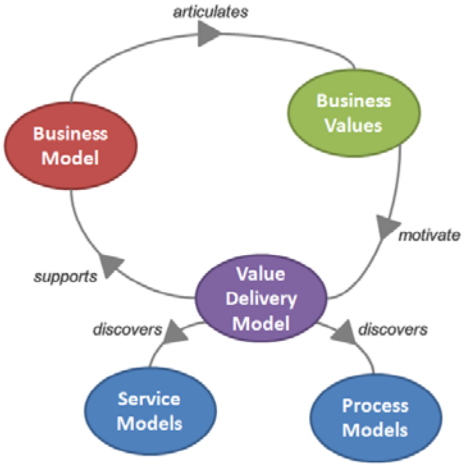


Figure 3. Position and role of a value delivery model

As Figure 3 suggests, value delivery models will:

- Be motivated by business values
- Support business models
- Be basis for discovery of process and service models

The fact that business models need to articulate business values is not subject of analysis in this document, but will need to be addressed in next WP4 deliverables in NEFFICS.

Integrated support for value delivery models will enable to:

- Support value and business modeling frameworks with the "engine", to unambiguously analyze the as-is business (system) and analyze and design to-be business (system) scenarios, entirely or in parts, and to measure, calculate and simulate aspects of the "*rationale of how an organization creates, delivers, and captures value*".
- Align value and business modeling frameworks with structured process and service models, to guide implementation, automation, management and measurement of business model "rationale" in the "real world", as well as to provide performance and value measurement feedback to structured-model-based representations of the business (and business models).

This will enable integral and closed-loop business model innovation. We will revisit this later in the document.

In particular, value delivery models will enable the discovery and (re) design of processes and services, from a value contribution and value delivery perspective. Value delivery models can serve as process and service discovery models.

2.2 Specific requirements for value delivery modeling

In the next sub-sections we will analyze specific requirements for value delivery models, whereby we follow the structure as presented in Figure 3.

In subsequent sections we will discuss how value delivery model can leverage essentials of existing, though partial, approaches, and how value delivery models can also provide more advanced innovation support, such as support for analysis of impact of business designs on value objectives, as well as support for simulation.

2.2.1 Business values motivate value delivery models

The following requirements for value delivery models are taken from NEFFICS D3.2 (2011), its section 6 on "ICT support".

Value identification

- Identify any form of value, financial and non-financial, tangible and non-tangible.
- Definable types of value

Value flow, intra and inter-enterprise

- Value creation, distribution (or "delivery") and consumption
- By and between an unlimited number of roles, attached to an unlimited number of organizations
- Link value flow within the business to value flow with customers and business partners (extended enterprise, enterprise network, business ecosystem, extended value flow network).

Activities and activity networks

- Value is created, distributed and consumed by activities.
- A value may depend on a single activity or on a combination (or network) of activities.

Value measurement

- Value and aspects of value creation, distribution and consumption, should be measurable, based on definable types of measures.

2.2.2 Value delivery model supports business model

The business modeling framework of Osterwalder and Pigneur (2010) is constituted from the building blocks as represented in their "business model canvas" as shown in Figure 4.

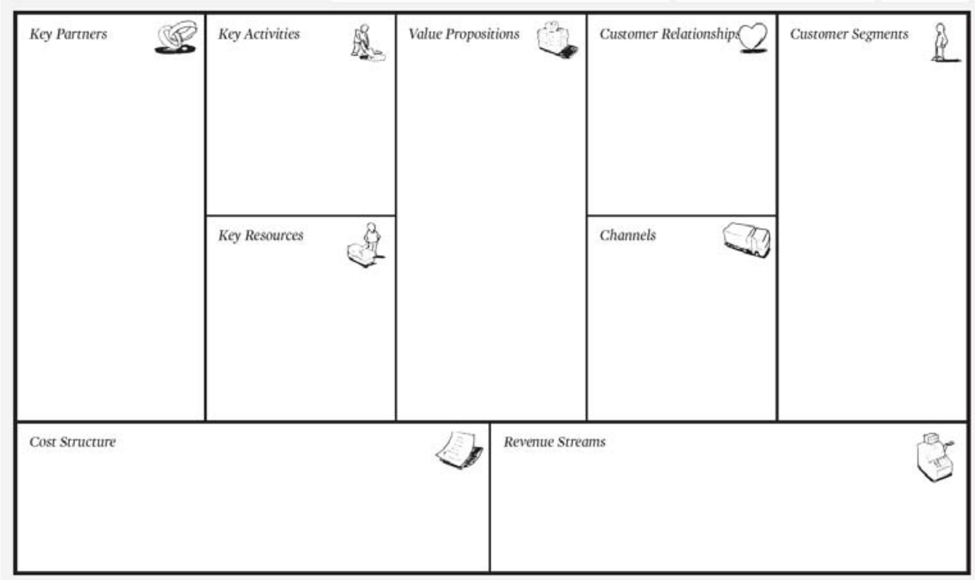


Figure 4. Business model Canvas (source: Osterwalder and Pigneur (2010))

Other business modeling frameworks, such as the fore mentioned frameworks of Johnson et al. (2010), and the NEFFICS business modeling framework, as documented in NEFFICS D4.1 (2011) and NEFFICS D4.3 (2012), have a rather similar structure, as is shown in Table 1.

OSTERWALDER AND PIGNEUR	JOHNSON ET AL.	NEFFICS
Customer segments	Customer value proposition	Target users (non-invoiced stakeholders), customers and market segments
Customer relationships		
Channels		
Value propositions		Value propositions
Key activities	Key processes	Internal value chain, using the functions that are applied to create value
		Competences, representing resources and activities
Key resources	Key resources	
Key partners		Network and network partners
Revenue streams	Profit formula	Profit formula, or more generally, value formulas
Cost structure		

Table 1. Harmonization of business modeling frameworks

Value delivery models, when used to provide structured modeling support for such business modeling frameworks, will need to capture the following:

Customer, market segments and other stakeholders

- Identify customers, market segments and stakeholders, other than customers, and business relationships with them, to which parties the business provides value.

Value proposition

- Define value propositions, related to the products and services that are offered to these parties, and which articulate the values that are delivered to them, in terms that relate to how their needs are satisfied.

Resources and activities

- Identify the resources and the activities that they perform or are used by, which are required to apply the functions that are needed to create the value that is required.

Network partners

- Identify network partners, such as suppliers and others, and business relationships with them, via which these parties contribute value, that is part of the overall value that the business provides.

Profit and value calculations

- Define the structures and related computation mechanics that can determine all relevant measurements of cost, revenue, as well as, more generally, value provided, value received, and, though maybe subjectively, "value margin", related to the operations of the business, in the context of the business model.

The same terminology is used here as is typically used to describe building blocks in these frameworks.

Table 2 shows how both sets of requirements, from 2.2.1 and 2.2.2, relate to each other.

VALUE DELIVERY MODEL	
"Motivated by Business Values"	**"Supports Business Model"**
Value identification	Value proposition
Value flow, intra and inter-enterprise	Customer, market segments and other stakeholders
	Network partners
Activities and activity networks	Resources and activities
Value measurement	Profit and value calculations

Table 2. Value delivery models versus business values and business models

When looking again to the detailed requirements in these various groups of requirements, it is apparent that the requirements from 2.2.1, which focus on business values, are merely complementary to requirements from 2.2.2, which focus on business models.

According to the current state of affairs, business modeling frameworks are not particularly explicit in dealing with values.

Support for value delivery modeling, can smoothly bring these areas together into a single modeling and analysis environment.

2.2.3 Value delivery model discovers process and service models

Discovery of process models based on value delivery models implies support for process improvement or discovery based on identification of opportunities to improve customer value.

It is worth integrating value delivery models with "automation models", such as process (automation) models and related service models. The two "worlds", of value delivery modeling (and analysis) and automation, when integrated, are empowering each other in both directions.

Process and service design will become more productive, when it will be possible to, eventually, automatically derive essential parts of process and service models, from the business know-how as captured in value delivery models.

Value delivery models contextualize elements of automation models, whereas automation models empower analysis by providing rigor in terms of measurements of performance and risk, and identification of issues.

When automation models are executed and monitored, real-world performance measurements can be fed back to support next cycles of analysis. And when innovation cycles start again, no effort is required to again discover value delivery models, as exploration of the as-is business system can be conducted right-away, as these models will be "live" and integrated artifacts in a "business operations platform".

The context that analysis models provide to automation, is concerned with defining the purpose and setting the right priorities for capability investments in business operations, and thereby, for innovation of processes, etc., and in particular for automation of them.

In order to facilitate all this, it is essential that concepts underlying value delivery models are sufficiently aligned with process and service modeling concepts.

According to BPMN (2011), a process is "*a sequence or flow of activities in an organization with the objective of carrying out work*". The concept of activities (and sequences or networks of them) is covered already by Table 2.

More elaboration is required on service modeling concepts however. Essential to modeling services are modeling of capabilities, interfaces to capabilities, and, collaborations (of roles) to engage capabilities, through interfaces. We will take a closer look into these concepts, based on a number of authoritative specifications of service oriented architecture (SOA) and modeling services.

Capabilities and interfaces of capabilities

- According to SOA-RM (2006), a service is "*a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description*".

- A capability, as defined by SOA-RM (2006), is a "*real-world effect that a service provider is able to provide to a service consumer*".

- SOA-RA (2011) defines a capability, less abstractly, as an "*ability that an organization, person, or system possesses to deliver a product or service*".

- SoaML (2012) defines a capability as the "*ability to act and produce an outcome that achieves a result*". According to SoaML (2012), capabilities are used to identify needed services, and to organize them into catalogs in order to communicate the needs and capabilities of a service area. Participants that provide a service must have a capability to provide it, but different providers may have different capabilities to provide the same service. The capability behind the service will provide the service. The service capability is frequently integral to the provider's business process. Capabilities can be seen from two perspectives, capabilities that a participant has that can be exploited to provide services, and capabilities that an enterprise needs that can be used to identify candidate services.

Collaboration to engage capabilities

- SOA-RM (2006) also defines interaction as the "*activity involved in making use of a capability offered*".

- SoaML (2012) defines collaboration as a "*description of a pattern of interaction between roles*", whereby entities or participants (e.g. persons, organizations, or systems) "play a role" in a collaboration.

- SoaML (2012) states that the concept of collaboration is equivalent to what SOA-RM (2006) calls interaction.

SoaML (2012) also addresses organizational aspects of services and capabilities.

Organizational alignment of capabilities, activities and resources

- Capabilities and services are possessed and provided by organizations
- Capabilities require a combination of organization, people, processes, and technology (SOA-RA (2011)).
- A process view, as complementary to a service view focuses on what activities parties perform to provide and use services.

Loose coupling of activity networks through stores

- From above it is clear that processes, or activity networks support capabilities, that are provided as services, and that capabilities are involved through activities as well. This implies many dependencies in the business system. These dependencies cannot all be activity-based however, as this would lead to a representation of a business as one vast activity network. This is not realistic. A business comprises many activity networks, which are loosely coupled, via decoupling buffers, in this document further denotes as stores.

Monitoring-based scenarios and measurements

- Processes and services, as discovered, and implemented, will be executed, and it is required to be able to feed measurements from monitoring of implemented processes and services back to the value delivery model, as input for next cycles of innovation.
- As-monitored measurements, based on as-implemented parts of a value delivery model should be distinguishable from e.g. to-be scenarios, with estimated, planned or simulated measurements. It is essential that scenario-based analysis can be applied, based on the same value delivery model.

Table 3 provides a harmonization of streams of requirements for value delivery models, as discussed so-far.

VALUE DELIVERY MODEL		
"Motivated by Business Values"	**"Supports Business Model"**	**"Discovers Process and Service Models"**
Value identification	Value proposition	
Value flow, intra and inter-enterprise	Customer, market segments and other stakeholders	Collaboration to engage capabilities
	Network partners	
Activities and activity networks	Resources and activities	Capabilities and interfaces of capabilities
		Organizational alignment of capabilities, activities and resources
		Loose coupling of activity networks through stores
Value measurement	Profit and value calculations	Monitoring-based scenarios and measurements

Table 3. Harmonization of three streams of requirements for value delivery models

Note that SoaML (2012) also relates to concept of service and capability to delivery and exchange of value:

- A service is "*value delivered to another through a well-defined interface and available to a community"*.
- Service is defined as "*the delivery of value to another party, enabled by one or more capabilities"*.

- The service may be connected to a business motivational element to indicate its intended value proposition.
- The exchange of value is the enactment of the service.

SoaML (2012) does not provide a precise and normative specification of the concepts of value, value proposition and value exchange however. Integration of service concepts in value delivery models will provide a unique opportunity therefore.

Some of the requirements in this section will also enable transformation of corresponding parts of value delivery models to process models and related service models. Transformation to process models and service models goes beyond the scope of this document however, and might be dealt with in subsequent deliverables of work packages WP5 and/or WP7.

2.3 Leveraging essentials of existing approaches

We will discuss the following approaches, some of which have been recognized as relevant to the area of analysis of business values by NEFFICS D3.2 (2012) as well (see Figure 5):

- Value network analysis (VNA)
- SCOR- and VRM-based approaches
- Lean value stream mapping
- Resource, Event, Agent modeling (REA)
- e3Value
- Capability analysis.

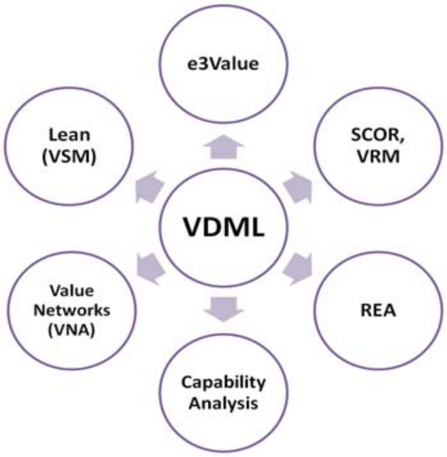


Figure 5. Existing approaches related to the value delivery modeling domain

We will analyze their limitations, that give room to value delivery models to provide more integral coverage, as well as their essentials that we want value delivery models to provide support for as well.

2.3.1 Value network Analysis (VNA)

Description

Value Network Analysis (VNA) uses a modelling technique that defines the specific roles in a collaboration and their value interactions that create value through the exchange of deliverables. Roles and deliverables are made visible through visual graphs (see Figure 6).

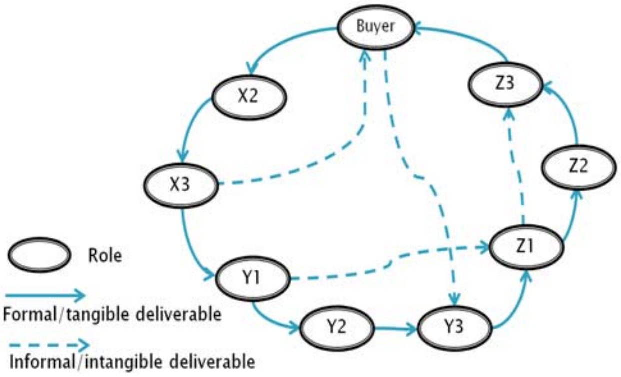


Figure 6. Value network with tangibles and intangibles

The goal of the method is to increase and/or optimize value outputs, to leverage financial and non-financial resources (including intangible assets) for improving financial and organizational performance, to find new value opportunities and improve operational performance and flows of value. See Allee (2003), Allee (2008) and Allee (2011) for a detailed description of VNA.

Limitations

Every detail, including activity, resource and value detail that goes beyond the definition of roles and deliverable flows (exchanges between roles) is defined as formatted text.

Strengths

The concept of role collaboration and exchange of deliverables that convey tangible and intangible value, is very useful. It is a natural pattern of how people and/or organizations interact and cooperate, and by doing so, exchanging value and/or together creating value. It also provides excellent basis for seeing the big picture (e.g. business networks or business eco-systems), as well as for drilling down into more detailed collaborations. It is also a far better basis for providing a complete picture, including flows of tangibles and intangibles, than an activity network (or “process”) can be. An activity network view would quickly become unreadable if it would contain the same level of detail.

This strength is basically covering all VNA provides, as the approach is not going further than just this feature. Therefore: What VNA provides is necessary, but not sufficient, to support value delivery modeling as intended.

2.3.2 SCOR- and VRM-based approaches

Description

Supply Chain Operations Reference (SCOR) model and Value Reference Model (VRM) are similar approaches. SCOR, as is e.g. presented by Bolstorff and Rosenbaum (2003), is often used for design and redesign of supply chains. VRM, as more recently presented by Mercer et al. (2010, 1), Mercer et al. (2010, 2) and Mercer et al. (2010, 3), is typically used for discovery of business processes that are eventually automated.

Both approaches focus on mapping chains of value chain activities, based on standardized reference model elements: capabilities (sometimes confusingly addressed as “processes”), together with associated measures, practices, and resources that maybe used and deliverables produced. They combine, in a sense, the value chain approaches of Porter (1985) and the Balanced Score Card approach of Kaplan and Norton (1996): embedding measurements, based on standardized measures, in activities in the value chain, comparing value chain designs based on performance measurements.

The concept of applying standardized library elements for mapping value chains or processes, and as basis for performance measurement, is also common in other frameworks (see Harmon (2007)).

Limitations

These approaches lack explicit focus on value proposition and value. Focus is on performance and performance benchmarks (see e.g. Francis (2007)). Focus is on activity networks. Support for modeling business networks, organization alignment, resource use, etc. is limited. Levels of abstractions can be defined by isolated models, which are not aligned by the model itself. There is no explicit support for creation of integrated models with a scope that is broader than a single process. There are no explicitly defined decoupling points (stores).

These approaches can be used to model a value chain or a process. But not to model a business model or a structured model that represents the operation of a business model. Focus is more on administrative and control flows. There is no support for analyzing parts of the same model in multiple contexts or scenarios.

Strengths

Particular strength of these approaches are:

- Integrated measurement approach
- Libraries for standardized reference model elements
- Activity network modeling
- Explicit modeling of resources and deliverables.

2.3.3 Lean Value Streams

Description

Value stream mapping, as explained by Rother and Shook (1998), is a lean manufacturing technique used to analyze and design the flow of materials and information required to bring a product or service to a consumer. Customer value is the leading motivation, and focus is on improving value, by reducing waste. It combines material flow (product produced) and information flow (e.g. sales orders or forecasts that trigger production). Broader systems can be modeled via decoupling buffers or stores (called “supermarkets”). The focus is on improving operational performance via detection and elimination of non-value added (i.e. wasted) time.

Limitations

Focus is on customer value, but value is never made explicit. Measurement relates to just cycle time (lead time), with distinction between lead time that is wasted (e.g. due to delays) and lead time that is productive (related to actually producing things that the customer wants). The approach fits better to manufacturing (shop floor work) than to administrative (knowledge) work, mostly due to the fact that it actually depicts a linear material flow of just the product of concern. Focus is on free-hand drafting of sketches. There is not much coming with the standard approach that supports detailed analysis or simulation.

Strengths

The following is strong and useful:

- Measurement and aggregation of lead time (performance), with distinction between added value and waste.
- The concept of stores, as buffers that decouple or link value streams.

2.3.4 Resource-Event-Agent (REA)

Description

Hruby et al. (2006) provide a detailed explanation and application of the Resource-Event-Agent (REA) pattern. This pattern provides an accounting perspective on the creation, transformation and exchange of economic resources by economic agents. In REA, economic agents perform economic events that convert or exchange economic resources. REA does not deal with more subjective forms of value, but focuses on products, services and resources that have defined costs and prices.

Limitations

Focus is on economic value, actually just price versus cost. The broader concept of value, as explored in NEFFICS D3.2 (2012), and intended to be supported by value delivery models, is not supported. The pattern that REA provides is rather abstract, and would have to be significantly extended, by implementers, to make it adequate of modeling complete operations of a business model. Levels of abstraction aren't supported explicitly. It would all depend on what an implementer uses it for. The intention of REA, from its inception on, has never been to serve as business analysis and design language to the extent value delivery models are meant for. It has rather been meant to drive model-driven logic to process accounting transactions (as is also clearly demonstrated by Hruby et al. (2006)).

Strengths

Strong features that are useful to consider for value delivery models:

- Explicit modeling of resources, resource stores and how resources are consumed and produced or received.
- Distinction between the role resource ("agent") and other resources that are actually transformed or exchanged by the role resource.
- The concept of reciprocity (something received compensating for something provided).

The basics of value exchange between "agents", is in a way more detailed than value network analysis provides, but on the other hand more limited, as it is just economic value.

2.3.5 e3Value

Description

e3Value, as presented by Gordijn and Akkermans (2003), is a modeling language for evaluation of the viability of e-commerce business models or value constellations, representing a group of economically independent entities, including market segments, that exchange transactions with economic value for mutual benefit. This seems straightforward, but in e-commerce the number of entities, their different interests and multiple exchanges can obscure the net value realized by the different participants. Each of the participants must have a sustainable business model for the overall exchange to be viable.

Limitations

Like REA, also e3Value focuses on just cost and revenue (so, economic value only), and has no broader concept of value. Cost of production of deliverables is just stated as a single measurement per deliverable. The capabilities applied by each party aren't defined explicitly. The e3Value approach adopts a rather specific way of how inputs received are transformed into outputs created. It is merely just a computation structure that would be equivalent to a set of expressions that are combined (OR, AND, XOR, etc.) to transform the cost of inputs into cost of outputs. Actually the entire effort of a party maybe considered a single activity with possibly complex computation logic associated, without explicit modeling of more specific activities, resources involved and stores of resources. This makes e3Value a good approach for quick assessment of exchange in n-ary business relationships, given cost prices of deliverables and given demand forecasts, but it is not a good approach to facilitate modeling and analysis of operations of business models in broader fashion. Alignment with processes and services would be difficult and fuzzy, based on e3Value.

Strengths

Some features of e3Value are relevant to value delivery models too, though value delivery models will need to address these in a more generic fashion.

These features are:

- Support for quantitative what-if analysis or simulation of profit, based on cost and revenue in complex exchanges (n-ary business relationships), also based on demand forecasts of the parties involved.

- Explicit definition of the deliverables that are produced and exchanged, which are the carriers of cost and sales price.
- Explicit definition of scenarios, to enable what-if simulation of different scenarios, based on the same model.
- Support for rather complex aggregations of measurements, albeit only cost-related.

2.3.6 Capability analysis

Description

A capability map, as used in capability analysis, defines a hierarchy of capabilities required for the enterprise to deliver the desired results along with assessment of the importance and performance of these capabilities. The capability map is analyzed to identify those capabilities that require improvement—often called a capability "heat" map, an example of a part of which is shown in Figure 7.

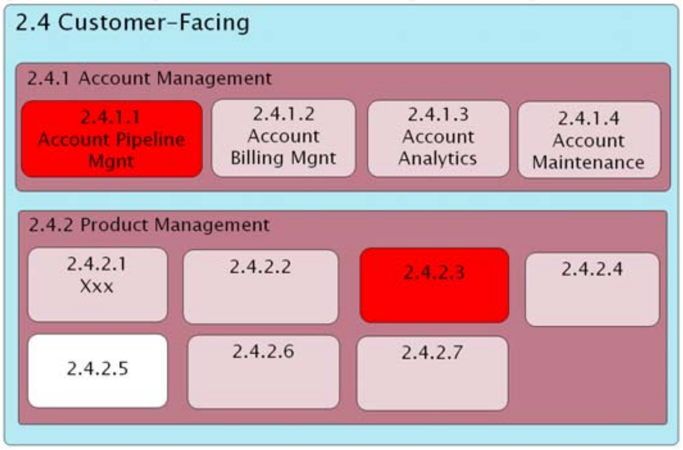


Figure 7. Capability "heat map"

According to Krohn (2011), the capability map is the framework for defining scope and analyzing impact. A capability is "what" the business does. By focusing on the what, the map becomes very stable. "How" something is done changes frequently; with every system implementation or process improvement, it is altered. However, what is done remains relatively the same, year after year. The map organizes these capabilities into a hierarchy, with each capability level providing progressively more detail. The hierarchy enables to start with a broad discussion and then dive into more detail where needed. Creating a capability map, containing commonly used or usable definitions of capabilities, with their associated detail, establishes a common vocabulary across the business. This will enforce productivity in design or re-design of business models, and will facilitate discovery of opportunities to consolidate or outsource (or purposefully not doing so) capabilities.

Some approaches, such as presented by Ulrich and McWhorter (2011), also associate capabilities with value chains or value streams, as the means for performing activities, and a way to define how capabilities, as provided by the business, depend on each other, and, together, deliver the value to the customer. Rather than getting lost in the details of IT implementations, organizations should instead focus on the capabilities required to maximize value stream benefits. A value stream may involve dozens of capabilities. The same capabilities are often deployed in many places. Value stream mapping provides planning input to help determine which capabilities may be required to align, consolidate, and automate various business processes across product lines and business units. When the same capability is shown in multiple value streams, this implies that it is reused across these value streams. If two or more apparently identical capabilities are being used in different value streams, then there may be redundancies in the organization that require alignment from a business and/or IT perspective. This approach is similar to what is proposed by Cummins and De Man (2011).

Limitations

Actually all what is intended by these approaches is valuable as key aspect of potential value delivery models. Identification of capabilities alone and in isolation, and e.g. just creating "heat maps" of capabilities is not sufficient however. Applying capabilities in value chains or value streams is essential, but will still only provide a partial coverage of what a business model represents.

Strengths

Define capabilities, and apply them in value streams of value chains, to enforce a common vocabulary in the business, more robust business design, better comparability of the possibly multiple ways in which a business does something, and to enable separation of the “what” business does, from the “how” it is done, to facilitate analysis and design at different levels of abstraction and detail.

2.3.7 Summary of limitations

None of these approaches are sufficient to serve as basis for value delivery models as envisaged. Creating modeling support for value delivery models is not a matter of just combining these approaches, as:

- Their ways of expressing concepts is too heterogeneous, as are their fundamental purposes and ambition levels.
- There are overlaps and conflicts of concepts that would not be resolvable, and there would still be significant gaps that are not addressed by these methods.

It is, instead much better to introduce a new modeling language, called Value Delivery Modeling Language (VDML), in such a way that essential and strong concepts of the various approaches are still supported, and that model representations of existing approaches might be mapped to (or transformed from) parts of value delivery models.

2.3.8 Summary of strengths, taken as further requirements

Value flow through role collaboration

- The concept of role collaboration and exchange of deliverables that convey tangible and intangible value.

- The concept of reciprocity in collaborations or value exchanges (something received compensating for something provided).

Capability and value stream / chain analysis

- Define capabilities and apply them in value chains or value streams.

- Libraries for standardized reference model elements, such as capabilities and related measures, resources or deliverables and practices).

Explicit modeling of resources, resource stores, resource use and deliverables

- Explicit modeling of resources and deliverables.

- Explicit modeling of resource stores and how resources are consumed and produced or received.

- Distinction between the role resource (“agent”) and other resources that are actually transformed or exchanged by the role resource.

- The concept of stores, as buffers that decouple or link value streams.

Emphasis on measurement of performance and value, also applied in scenario-based analysis

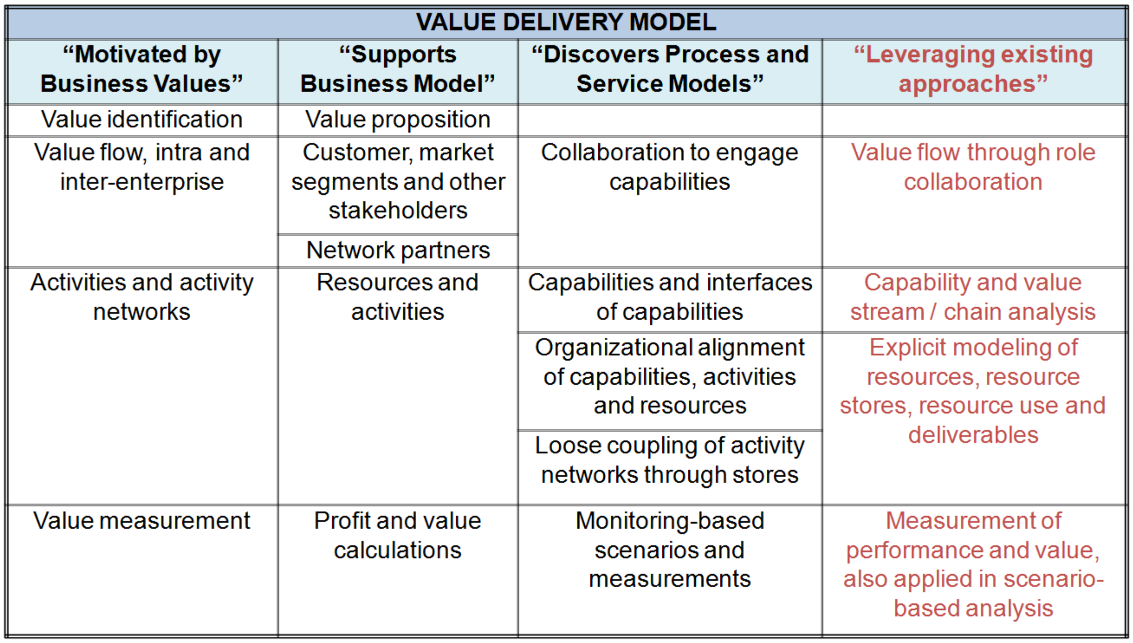
- Aggregation of performance measurement, with distinction between added value and waste (as lean value stream maps do with lead time).

- Support for rather complex aggregations of measurements (cost-related, and more).

- Support for quantitative what-if analysis or simulation of profit, based on cost and revenue in complex exchanges (n-ary business relationships), also based on demand forecasts of the parties involved.

- The explicit definition of scenarios, to enable what-if simulation of different scenarios, based on the same model.

Table 4 provides a harmonization of the various streams of requirements for value delivery models, as discussed so-far.



VALUE DELIVERY MODEL			
“Motivated by Business Values”	**“Supports Business Model”**	**“Discovers Process and Service Models”**	**“Leveraging existing approaches”**
Value identification	Value proposition		
Value flow, intra and inter-enterprise	Customer, market segments and other stakeholders	Collaboration to engage capabilities	Value flow through role collaboration
	Network partners		
Activities and activity networks	Resources and activities	Capabilities and interfaces of capabilities	Capability and value stream / chain analysis
		Organizational alignment of capabilities, activities and resources	Explicit modeling of resources, resource stores, resource use and deliverables
		Loose coupling of activity networks through stores	
Value measurement	Profit and value calculations	Monitoring-based scenarios and measurements	Measurement of performance and value, also applied in scenario-based analysis

Table 4. Harmonization of four streams of requirements for value delivery models

In the next section we discuss some requirements that relate to more advanced support by value delivery models.

2.4 Providing advanced innovation support

As has been stated earlier, value delivery models will help enabling integral and closed-loop business model innovation. They will particularly:

- Support designing, and analyzing the as-is business (system).
- Serve as context for innovation ideas, as well as support designing prototypes of what is envisaged.
- Support discovery and design of and/or alignment with processes and services. This includes implementation, automation, and management of processes in the “real world”, as well as measurement of performance and value contribution of these processes and feedback of measurement results for purpose of further analysis. This might lead to discovery of issues that need to be resolved by further improvement of the business system.

Implementing process and service improvements or new processes and services is where the business innovation model rubber meets the road, as this might have major impact in the organization, might be disruptive and may introduce significant risk.

It is essential that value delivery model scenarios, supporting to-be strategies to further innovate the business, are thoroughly defined and analyzed. Value delivery model-based impact analysis is essential, to see to which extent they help achieving value objectives, targeted by innovation. “Winning” strategies are typically broke down in tactics, i.e. particular change requirements to transform the as-is business system, and which are typically conducted as initiatives or projects. Such tactics will involve changes in value delivery models and related process models, service models, etc., as well as changes in the "real world" business system itself, such as training people, putting them on new positions, getting new partnerships in place, etc. Figure 8 indicates the role that value delivery models can play in this context.

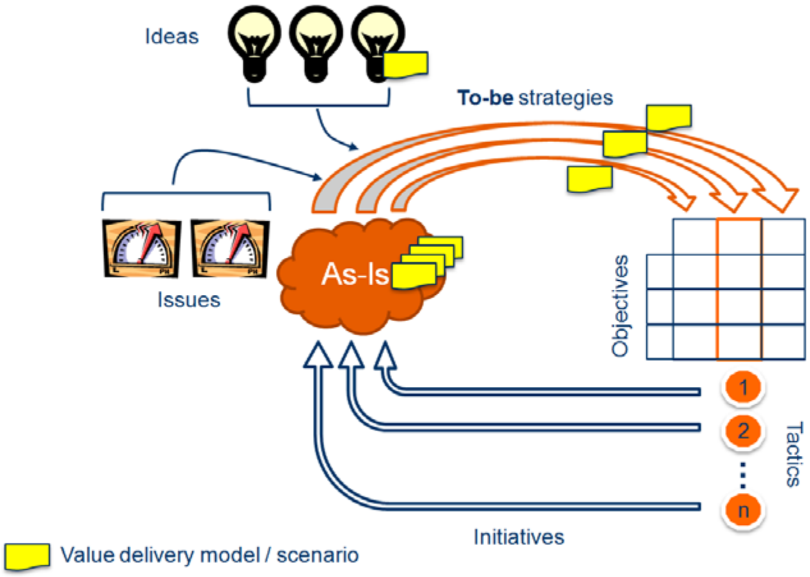


Figure 8. The role of value delivery models in business innovation

Impact estimation requires analysis of multiple scenarios, possibly based on the same model, and a measurement system that is scenario based. Scenario-based measurements might be determined manually (based on estimates), through monitoring of as-implemented processes and services, or based on simulation.

Simulations might be diverse, such as interactive what-if scenario analysis, similar to scenario-based analysis in e3Value, or based on average measurements of performance and value.

Simulations may often require more power, in order to understand robustness of innovations, or to detect potential risks that would otherwise be overlooked. Common types of simulation, that value delivery models eventually may have to support, are:

- Monte Carlo simulation. A Monte Carlo simulation consists of many (typically hundreds or thousands) of trials, each trial sampling from the distribution of a set of input elements, and then aggregate the composite answer (output); thus the return would be characterized not by a single number, but by a stochastic distribution of possible outcomes
- Discrete Event simulation. Discrete event simulation involves modeling the system as a set of objects evolving over time according to the availability of resources and the triggering of events.
- System Dynamics simulation. System dynamics simulation involves identifying the key "state" variables that define the behavior of the system, and then relating those variables to one another through coupled, differential equations.

See Dooley (2002) for a short discussion of these simulation approaches. See also Hubbard (2010), who provides a quite readable discussion of how Monte Carlo simulation can help assess risks.

Both Monte Carlo simulation and Discrete Event simulation require that measurements can be determined stochastically.

Discrete Event simulation, as well as System Dynamics simulation require that resource stores can be used, as part of activity networks. Such stores are normally called "stocks" in System Dynamics.

In summary: advanced analysis and simulation, as discussed in this section, require the following from value delivery models:

- Scenario-based measurement and analysis.
- Modeling activity networks that contain resource stores.
- Measurements enabled for stochastic determination.

These partly confirm what is already captured in Table 4, and partly add to it (such as the enablement of measurements for stochastic determination).

Other more specific and detailed requirements for simulation will be discussed later.

3 Value Delivery Modeling Language (VDML)

VDML, as modeling language to create structured value delivery models, is meant to provide integral and integrated modeling support that addresses the requirements as discussed in chapter 2.

In this chapter we will first explain VDML. In the next chapter we will asses how VDML addresses the requirements as have been analyzed in chapter 0.

3.1 VDML explained

3.1.1 Approach to explain VDML

As Figure 9 suggests, VDML can be thought of as constructed from the following parts:

- Libraries, such as measure libraries, capability libraries, etc.. Libraries contain elements that are standardized in a business community or company, and that foster re-use of business design elements, enforce common vocabulary and normalization of business designs, and makes business modeling more productive.
- "Business design elements", which are used to construct the actual design of a business system, and which may refer to library elements.
- Scenarios, supporting analysis of a business design, or a part of it, in different circumstances of use.
- "Mechanics", representing all model elements that are not directly exposed to business analysts, that create or use value delivery models, but that serve as technical modeling foundations to other model elements. These are e.g. concerned with to how fundamentally model roles and collaboration, how to support decomposition, etc.
- Integration with a measurement framework, as specified by the Structured Metrics Metamodel, published as SMM (2012). The NEFFICS project partners SINTEF and Cordys are currently participating in, and co-managing a revision task force in the OMG, to create a revision of SMM that is best supports VDML concepts and integration.

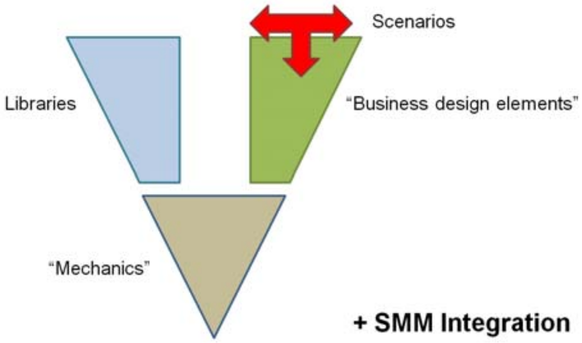


Figure 9 VDML constituents

The formal and detailed VDML 1.0 specification, traverses the VDML language part by part, more or less in this sequence: SMM integration, "mechanics", "business design elements", scenarios, libraries.

In this document, we opt to discuss VDML concepts from a user, or use case perspective, and to discuss underlying details less formally. Per each aspect of the business use case, we will discuss VDML and SMM elements that support modeling and analyzing it, regardless of its nature (i.e. regardless which part in Figure 9 it belongs to). We will not be exhaustive in explaining metamodel details. The reader should refer to the VDML 1.0 specification document to get the complete specification, covering all details.

The small use case example that we will use to explain VDML originates from NEFFICS partner Vlastuin, but is generic enough to be commonly encountered in any product-manufacturing business. Vlastuin produces, amongst others, trailers, and hence this use case example is about the business of a product family of trailers, denoted as XTtrailers. According to this use case, transporters (XTrailer customers) place orders, based on which the manufacturer (here Vlastuin) produces and delivers the trailers. Transporters also provide feedback, in the form of ideas for further innovation of the trailers (in the XTrailer product family). The manufacturer uses these ideas to apply new innovations and to make these available to the transporters market.

Discussion will be supported by four different types of diagrams, as indicated in Figure 10 :

- Mockups of graphical model notation. These mockups suggest how model data may be exposed to users. A VDML model user interface will be more extensive than the set of mockups that are discussed in this document. A model user interface will consist of both graphical diagrams and data-forms, but in this document we will only be concerned with a selective subset of mockups of graphical diagrams. These mockups are not included in the normative part of the VDML specification, but are vendor-specific ideas of how VDML model data can be exposed to the user. Some of these ideas might be included as non-normative parts of the VDML specification.
- Object diagrams. Objects are instances of classes, and denote the actual model data, or model elements, a subset of which is exposed via graphical notation. In this document, we make use of UML object diagrams.
- Metamodel diagrams. Meta-model diagrams contain the meta-model classes, that are instantiated to create model objects. These diagrams belong to the normative part of the VDML specification, as submitted to the OMG. As indicated above, we will use slight revisions of some of them. Meta-models are an abstraction of underlying concepts, and express their related semantics. It is common practice in the OMG to adopt Unified Modeling Language (UML) -based class modeling notation, to express meta-model diagrams. For explanation of UML in general, and UML class diagrams and object diagrams in particular, the user might refer to Fowler (2004) for a user-oriented introduction to UML, or to UML (2011) for a formal specification.
- Informal diagrams, to explain some underlying VDML concepts.

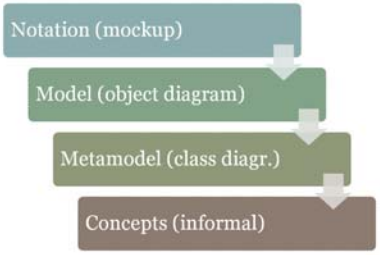


Figure 10. Diagrams used in explaining VDML

We will follow the following route in presenting the use case and discussing how VDML supports modeling and analyzing aspects of it, with reference to the VDML ontology in Figure 11, which is a high level abstraction of core VDML meta-model concepts:

- Business networks and collaboration
- Organizations and capabilities
- Activities
- Value

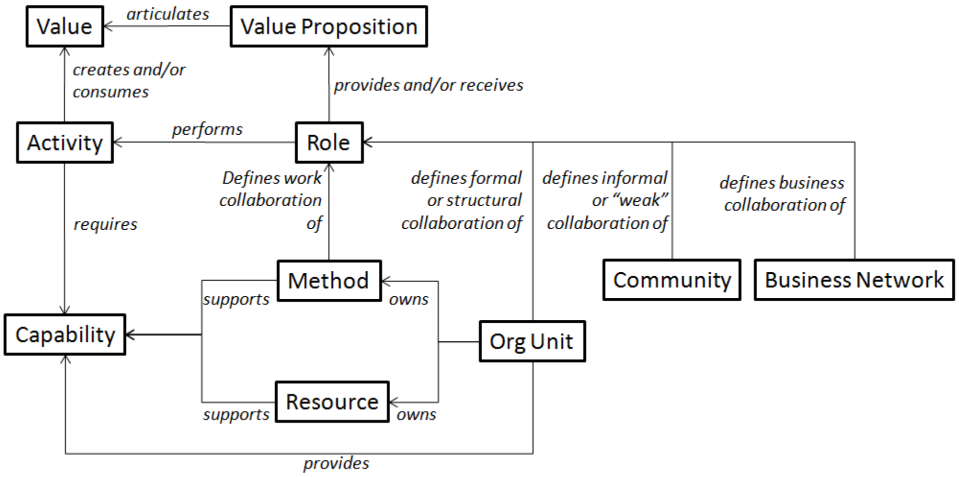


Figure 11. VDML high-level ontology

Definitions of metamodel terms, and related concepts, as far as relevant in the context of the use case, will be provided throughout the course of the discussion. When no reference is provided in the text itself, the definition is a VDML definition. Most definitions are provided in the Glossary in chapter 7, which glossary also provides references to other sources that make comparable use of similar terms. For some more technical VDML-specific definitions, not contained in the Glossary, the user can refer for more details to the VDML 1.0 specification document.

3.1.2 Business network and collaboration

According to VDML modelling collaboration is fundamental to modelling business systems. The essence of what is going on in business is "collaboration". Value is created through collaboration. We will discuss about "value" later on. VDML defines a collaboration as a "*collection of participants working together for a shared purpose*", whereby a participant is defined as "*anyone or anything that can fill a role in a collaboration*". Participants can be actors (human or automatons) or collaborations by themselves, or roles of actors or collaborations. An actor is "*an individual (indivisible) participant, which might be human (a person) or non human (e.g. a software agent or machine)".* Basically a business system is constructed as a collaboration of collaborations. An entire enterprise or business network is a collaboration, and so is any team, project, process, taskforce, etc. VDML also provides the means to align collaborations with each other and to construct collaborations from other collaborations and individuals (actors). We will discuss these throughout the course of this chapter. Participants participate in a collaboration, and as such, collaborate, through roles. Each collaboration defines and contains a set of roles, a role being "*an expected behavior pattern or capability profile associated with participation in a collaboration*". Roles define what is expected from participants, and what they have to perform, in the specific context of a collaboration. The definition of role introduces the term "capability". We will discuss about capabilities later. Capability expectations of roles come from capability requirements of activities that roles have to perform in collaborations. We will discuss about activities and capability requirements later as well. Normally roles are also associated with rights, such as access rights to applications or information, but such concerns go beyond the scope of VDML.

VDML specializes collaboration into a few common sub-types. Figure 12 highlights two of them: "community" and "business network".

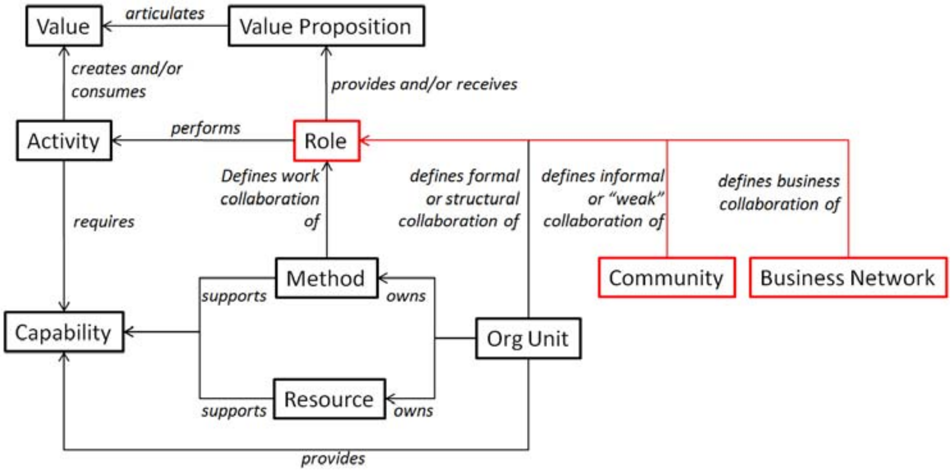


Figure 12. VDML high-level ontology: Collaboration highlighted

A business network is "*a collaboration between independent business (or economic) entities, potentially companies, agencies, individuals or anonymous members of communities of independent business entities, participating in an economic exchange*". A business network defines a business collaboration of roles.

A community is "*a collaboration of participants with similar interests that work together for some shared purpose such as sharing knowledge*". A community defines an informal or "weak" collaboration of roles. The collaboration that a community represents is said to be "informal" and "weak", as opposed to collaboration in the formal and structured part of the organization, as defined by organization units. Organization units (or org units) will be discussed later. We will revisit communities later, and will first focus discussion on business networks.

Figure 13 provides a model view mockup of the simple business network in the XTrailer use case example. This notation is compliant with the value network notation as introduced by Allee (2008).

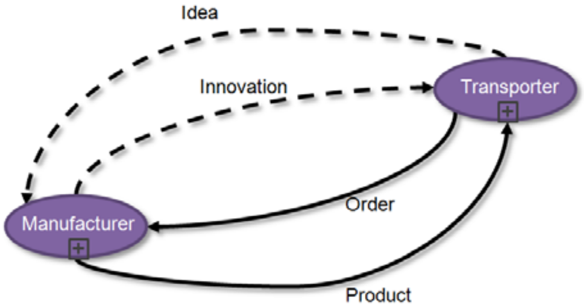


Figure 13. Business network collaboration mockup

It represents collaboration between two parties, indicated by the ovals: the manufacturer and the transporter. Parties are specialized roles. Roles in business network are "parties". Collaboration between Manufacturer and Transporter is represented by their exchange of deliverables, via deliverable flows, depicted by the connectors. Solid connectors denote flows of tangibles. Dashed connectors denote flows of intangibles.

A deliverable flow is defined as "*a transfer of a deliverable from a provider (or producer) to a recipient (or consumer)*". A tangible is "*a deliverable that represents something that is contracted, mandated or expected by the recipient and which may generate revenue*". In the example, Order and Product are tangibles. In the context of the XTrailer use case, Order can be taken as trailer order and Product as a trailer. An intangible is "*a deliverable that represents something that is unpaid or non-contractual that makes things work smoothly or efficiently*". In the example, Idea and Innovation are intangibles. Transporters provide feedback to the manufacturer, in the form of ideas. The manufacturer works ideas into innovations of the product, provided to the transporters. These go beyond what is expected or contracted.

The model view mockup in Figure 13 is a user's view of the model objects in the object diagram in Figure 14. Note again that this is the technical view "behind" the user's view. Note that object diagrams generally show more objects than are graphically exposed in view mockups.

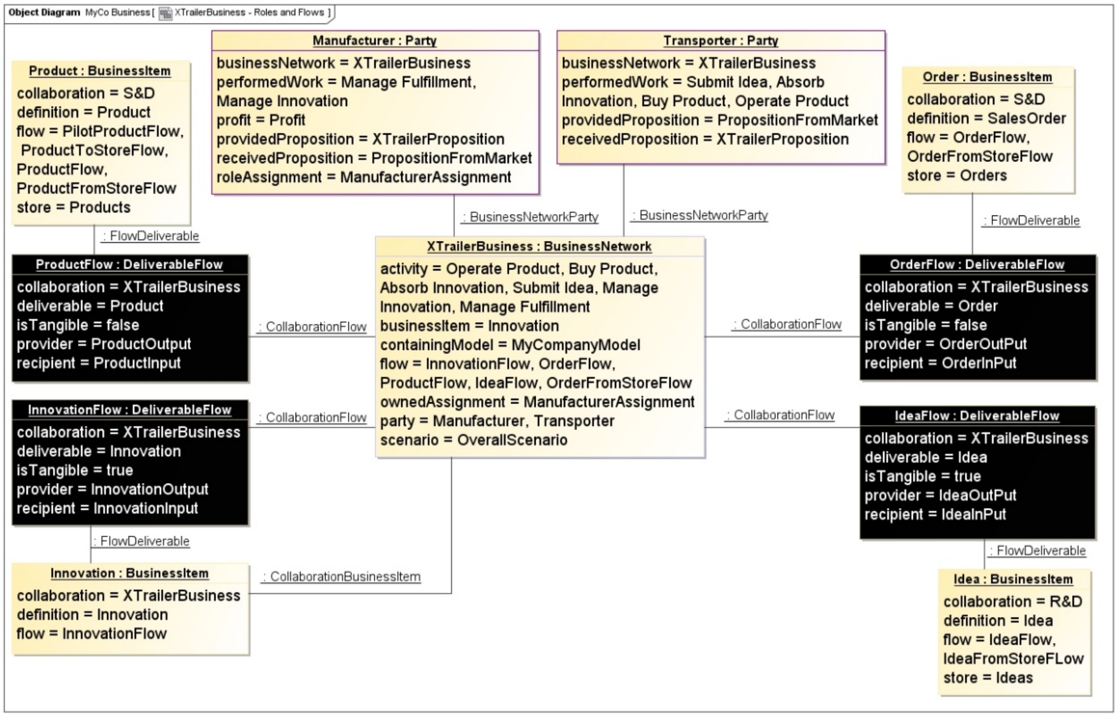

Object Diagram MyCo Business [XTrailerBusiness - Roles and Flows]
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
Transporter : Party
businessNetwork = XTrailerBusiness
performedWork = Submit Idea, Absorb Innovation, Buy Product, Operate Product
providedProposition = PropositionFromMarket
receivedProposition = XTrailerProposition
Product : BusinessItem
collaboration = S&D
definition = Product
flow = PilotProductFlow, ProductToStoreFlow, ProductFlow, ProductFromStoreFlow
store = Products
Order : BusinessItem
collaboration = S&D
definition = SalesOrder
flow = OrderFlow, OrderFromStoreFlow
store = Orders
: BusinessNetworkParty
: BusinessNetworkParty
: FlowDeliverable
: FlowDeliverable
XTrailerBusiness : BusinessNetwork
activity = Operate Product, Buy Product, Absorb Innovation, Submit Idea, Manage Innovation, Manage Fulfillment
businessItem = Innovation
containingModel = MyCompanyModel
flow = InnovationFlow, OrderFlow, ProductFlow, IdeaFlow, OrderFromStoreFlow
ownedAssignment = ManufacturerAssignment
party = Manufacturer, Transporter
scenario = OverallScenario
ProductFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Product
isTangible = false
provider = ProductOutput
recipient = ProductInput
OrderFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Order
isTangible = false
provider = OrderOutPut
recipient = OrderInPut
: CollaborationFlow
: CollaborationFlow
: CollaborationFlow
: CollaborationFlow
InnovationFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Innovation
isTangible = true
provider = InnovationOutput
recipient = InnovationInput
IdeaFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Idea
isTangible = true
provider = IdeaOutPut
recipient = IdeaInPut
: FlowDeliverable
: FlowDeliverable
Innovation : BusinessItem
collaboration = XTrailerBusiness
definition = Innovation
flow = InnovationFlow
: CollaborationBusinessItem
Idea : BusinessItem
collaboration = R&D
definition = Idea
flow = IdeaFlow, IdeaFromStoreFLow
store = Ideas

Figure 14. Business network collaboration objects

The two top-most objects represent the parties (roles). Of the four deliverable flow objects, two are marked as tangible (isTangible = true), and two as intangible (isTangible = false). Parties and flows are contained in the business network. The business network object itself is not visible in the model view mockup in Figure 13.

Deliverables, shown as names along the connectors in Figure 13, are defined as business items, associated with the deliverable flows. A business item is "*anything that can be acquired or created, that conveys information, obligation or other forms of value and that can be conveyed from a provider to a recipient. For example, it includes parts, products, units of fluids, orders, emails, notices, contracts, currency, assignments, devices, property and other resources*".

Only one business item, Innovation, is contained in the business network. The other business items aren't specific to the business network, but are used across multiple collaborations.

Objects in the object diagram in Figure 14. are instances of a subset of the classes in the metamodel diagrams in Figure 15, Figure 16 and Figure 17. Several elements in these diagrams will be discussed later. Figure 15 shows that a business network is a collaboration, and that its parties are roles.

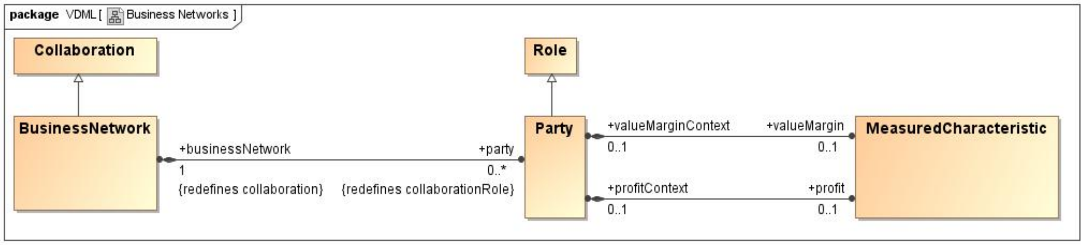


Figure 15. Business Networks metamodel

Figure 16 shows how a collaboration contains roles, deliverable flows and business items. Later on we will revisit this diagram, to discuss about assignments of participants to roles.

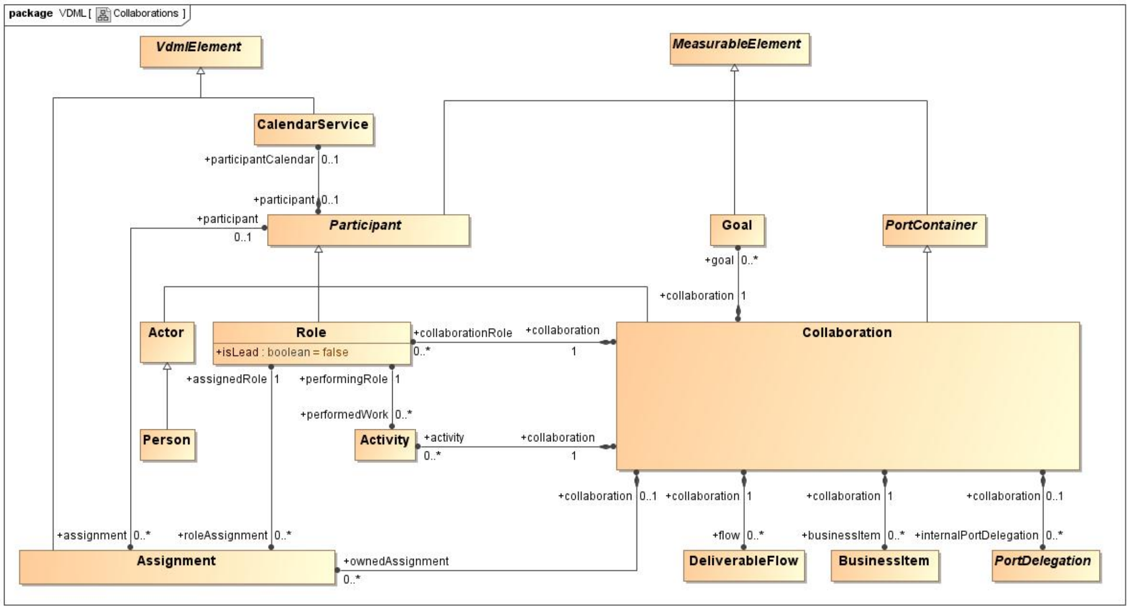


Figure 16. Collaborations metamodel

Figure 17 shows how a business item is associated to a deliverable flow, and that a deliverable flow can be marked as tangible or intangible (via a boolean property).

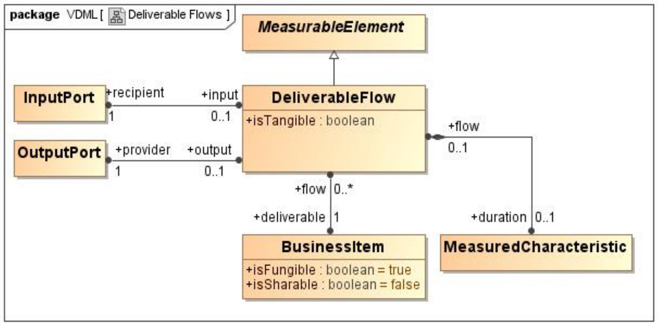


Figure 17. Deliverable Flows metamodel

Note that deliverable flows connect to recipient and provider via “ports” (input port, output port). This detail, which is abstracted away from the role collaboration view mockup in Figure 13, will be discussed in 3.1.4.

Figure 18 shows how a business item can be associated with a business item definition.

It can be defined for a business item, whether its instances are interchangeable (i.e. “fungible”) or whether the same instance can be used simultaneously in different locations (i.e. it is “sharable”). These properties, that are essential in discrete event simulation based on the model, have not been applied in the XTrailer use case example.

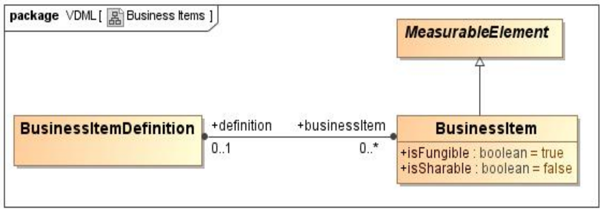

package VDML [Business Items]
MeasurableElement
BusinessItemDefinition
+definition
0..1
+businessItem
0..*
BusinessItem
+isFungible : boolean = true
+isSharable : boolean = false

Figure 18. Business Items metamodel

Business item definitions are contained in a business item library. A business item library contains a taxonomy of business items, consisting of business item definitions and categories of them, and is applied to enforce consistency in the definition of business items. Multiple business items that are associated with the same business item definition, are considered similar from the perspective of the library.

Figure 19 shows an object diagram that contains the business item definitions in the business item library as part of the XTrailer use case example.

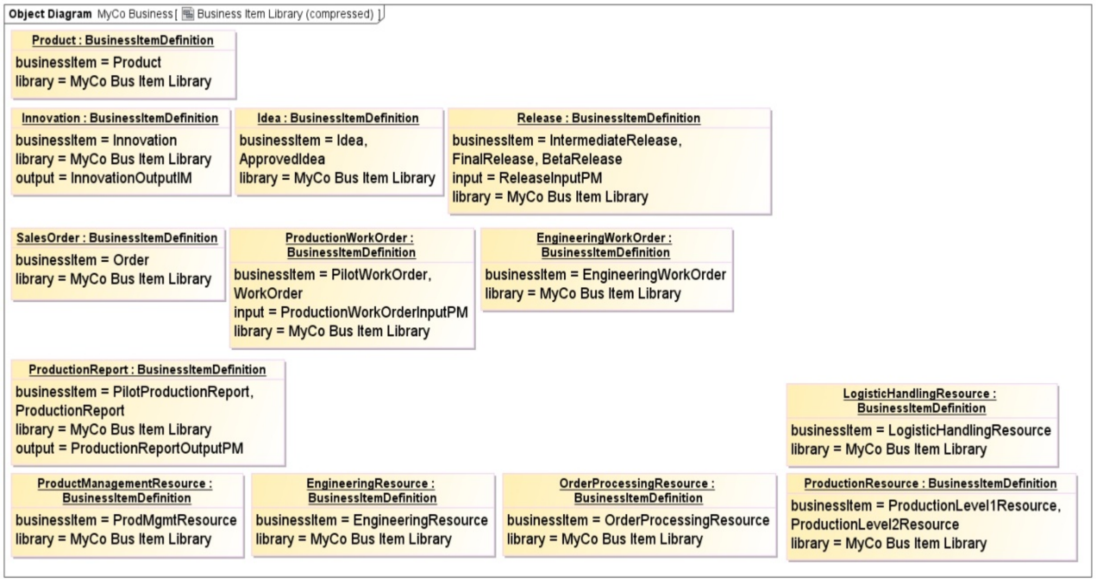

Object Diagram MyCo Business [Business Item Library (compressed)]
Product : BusinessItemDefinition
businessItem = Product
library = MyCo Bus Item Library
Innovation : BusinessItemDefinition
businessItem = Innovation
library = MyCo Bus Item Library
output = InnovationOutputIM
Idea : BusinessItemDefinition
businessItem = Idea, ApprovedIdea
library = MyCo Bus Item Library
Release : BusinessItemDefinition
businessItem = IntermediateRelease, FinalRelease, BetaRelease
input = ReleaseInputPM
library = MyCo Bus Item Library
SalesOrder : BusinessItemDefinition
businessItem = Order
library = MyCo Bus Item Library
ProductionWorkOrder : BusinessItemDefinition
businessItem = PilotWorkOrder, WorkOrder
input = ProductionWorkOrderInputPM
library = MyCo Bus Item Library
EngineeringWorkOrder : BusinessItemDefinition
businessItem = EngineeringWorkOrder
library = MyCo Bus Item Library
ProductionReport : BusinessItemDefinition
businessItem = PilotProductionReport, ProductionReport
library = MyCo Bus Item Library
output = ProductionReportOutputPM
LogisticHandlingResource : BusinessItemDefinition
businessItem = LogisticHandlingResource
library = MyCo Bus Item Library
ProductManagementResource : BusinessItemDefinition
businessItem = ProdMgmtResource
library = MyCo Bus Item Library
EngineeringResource : BusinessItemDefinition
businessItem = EngineeringResource
library = MyCo Bus Item Library
OrderProcessingResource : BusinessItemDefinition
businessItem = OrderProcessingResource
library = MyCo Bus Item Library
ProductionResource : BusinessItemDefinition
businessItem = ProductionLevel1Resource, ProductionLevel2Resource
library = MyCo Bus Item Library

Figure 19. Business Item Library objects

Some business item definitions in this library, namely Idea, Release, Production Work Order, Production Report and Production Resource are referenced from multiple business items (see property “business item” in the objects in Figure 19).

The objects in the object diagram in Figure 19 are instances of classes in the metamodel diagram in Figure 20. Business item categories are not applied in the XTrailer use case example. Characteristics will be discussed later.

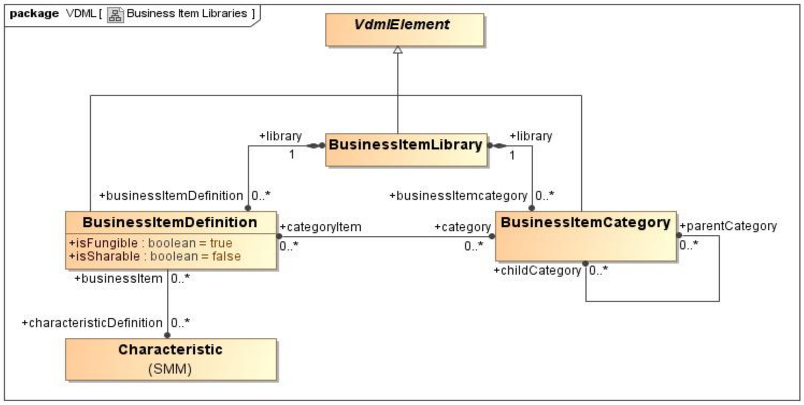


Figure 20. Business Item Libraries metamodel

A value delivery modeling tool can also use a business item library to guide the modeling user in productively discovering deliverable flows in a role collaboration view (like the simple one in Figure 13).

Figure 21 provides a model view mockup that represents the structure of the business network: participants assigned to party roles.

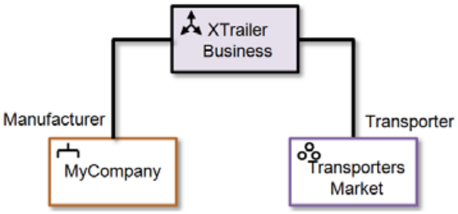


Figure 21. Business network structure mockup

The Manufacturer party role is assigned to the company MyCompany, and the Transporter party role is assigned to the market segment Transporter Market. A market segment can be modeled as a community, as has been introduced earlier. A community is a specialization of collaboration, as indicated in the metamodel diagram in Figure 22.

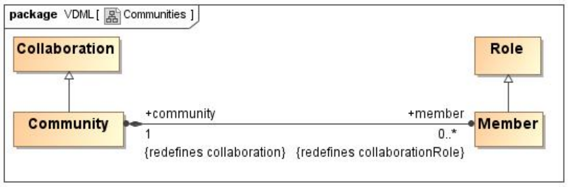

package VDML [Communities]
Collaboration
Role
Community
+community
1
{redefines collaboration}
+member
0..*
{redefines collaborationRole}
Member

Figure 22. Communities metamodel

The object diagram in Figure 23 shows the objects behind the model view mockup in Figure 22. Note that, to keep the size of the XTrailer use case example minimal, the assignment of the Transporter party role to the Transporter Market has not been implemented.

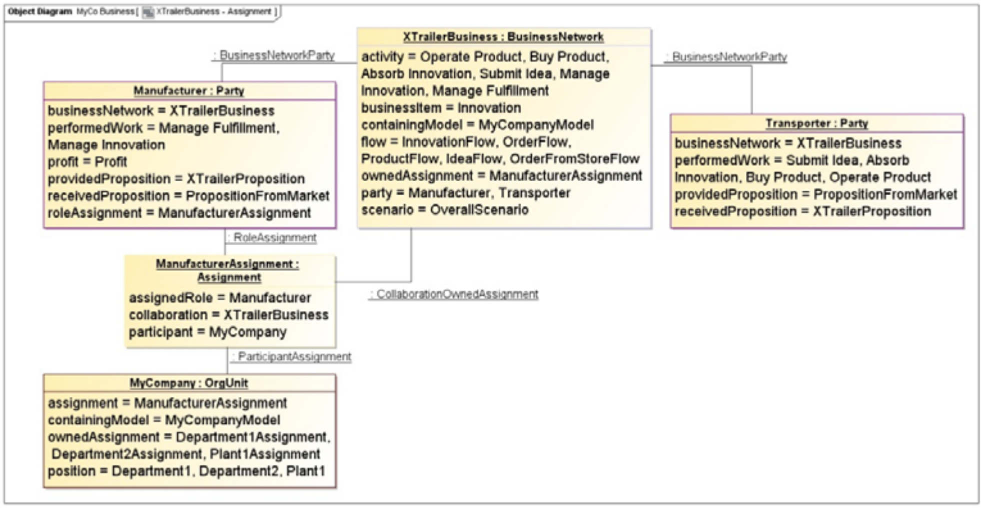

Object Diagram MyCo Business[XTrailerBusiness - Assignment]
XTrailerBusiness : BusinessNetwork
activity = Operate Product, Buy Product, Absorb Innovation, Submit Idea, Manage Innovation, Manage Fulfillment
businessItem = Innovation
containingModel = MyCompanyModel
flow = InnovationFlow, OrderFlow, ProductFlow, IdeaFlow, OrderFromStoreFlow
ownedAssignment = ManufacturerAssignment
party = Manufacturer, Transporter
scenario = OverallScenario
: BusinessNetworkParty
: BusinessNetworkParty
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
Transporter : Party
businessNetwork = XTrailerBusiness
performedWork = Submit Idea, Absorb Innovation, Buy Product, Operate Product
providedProposition = PropositionFromMarket
receivedProposition = XTrailerProposition
: RoleAssignment
ManufacturerAssignment : Assignment
assignedRole = Manufacturer
collaboration = XTrailerBusiness
participant = MyCompany
: CollaborationOwnedAssignment
: ParticipantAssignment
MyCompany : OrgUnit
assignment = ManufacturerAssignment
containingModel = MyCompanyModel
ownedAssignment = Department1Assignment, Department2Assignment, Plant1Assignment
position = Department1, Department2, Plant1

Figure 23. Business network structure objects

MyCompany is modeled as organization unit. Organization units are discussed in 3.1.3. The assignment of the Manufacturer party role to MyCompany is an instance of the Assignment class in the earlier discussed metamodel diagram of Figure 16. As an organization unit is a itself a collaboration, this assignment construct is one of the ways to construct collaborations from other collaborations.

3.1.3 Organizations and capabilities

Organization units are the building blocks to define the more formal and structured parts of organizations. An organization can be defined as "*an administrative or functional structure normally interpreted as a network of organization units at a higher level in an organizational hierarchy*". An organization unit is "*an administrative or functional organizational collaboration, with responsibility for defined resources, including a collaboration that occurs in the typical organization hierarchy, such as business units and departments (and also the company itself), as well as less formal organizational collaboration such as a committee, project, or task force*".

Organization units provide capabilities. A capability is "*the ability to perform a particular kind of work and deliver desired value*". We will discuss about value later, but analyze capabilities in more detail first. Organization units that provide capabilities do need capability methods and/or resources to support these capabilities. The same or different organization units own these. A resource is "*anything that is used or consumed in the production of a deliverable*". A capability method is "*a reusable template for a collaboration configured for participants to perform activities to deliver a capability and to contribute value in a particular situation*". We will discuss about activities later. Both organization units and capability methods are collaborations of roles. As will be discussed later, capability methods, especially the more structured ones, are a perfect basis for process discovery. In particular, process definitions that comply with the BPMN standard (see BPMN (2011)), can be generated from capability methods (model transformation).

These concepts, as highlighted, in the VDML ontology diagram in Figure 24, will be discussed and analyzed in more detailed and applied to the XTrailer use case example below.

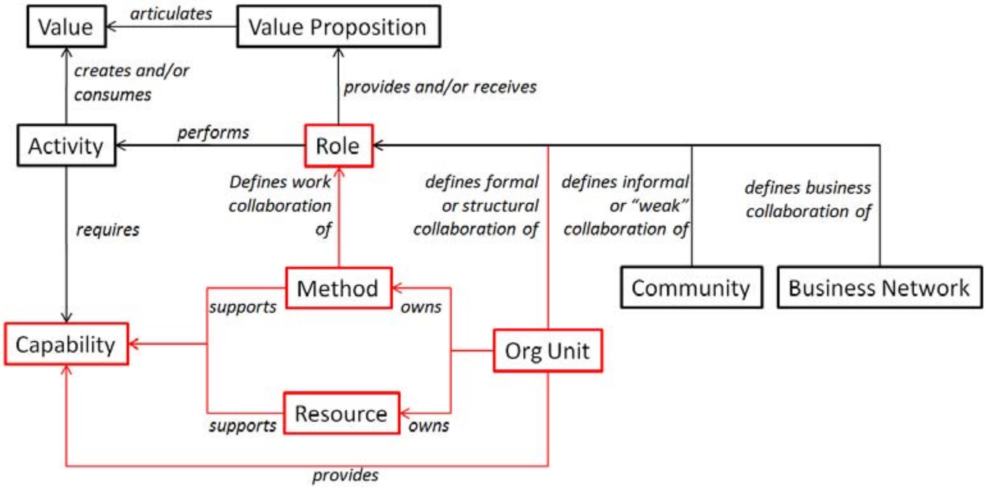


Figure 24. VDML high-level ontology: Organization Unit and Capability highlighted

Figure 25 shows the model view mockup of the structure of the MyCompany organization. As an organization unit by itself, it consists of three other organization units: R&D, Trailer Plant and S&D, being assigned the position roles Department1, Plant1 and Department2 in MyCompany respectively. R&D is responsible for research and development, Trailer Plant for the actual production of the trailers, and S&D for sales and delivery.

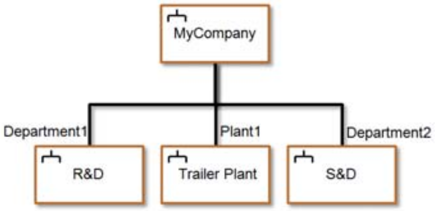


Figure 25. Organization structure mockup

As the metamodel of organization units in Figure 26 indicates, an organization unit (org unit) is a collaboration, and its positions are roles.

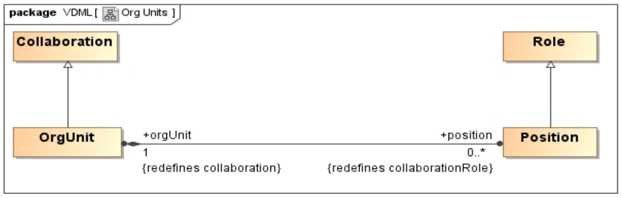

package VDML [Org Units]
Collaboration
Role
OrgUnit
+orgUnit
1
{redefines collaboration}
+position
0..*
{redefines collaborationRole}
Position

Figure 26. Organization Units metamodel

Consequently, and in accordance to the earlier discussed Collaborations metamodel diagram (see Figure 16), an organization unit, as collaboration, can contain assignments of positions (as roles) to other organization units (as collaborations, and hence participants). The object diagram in Figure 27 shows the objects that are behind the model view mockup in Figure 25. These instantiate classes in both the metamodel diagrams.

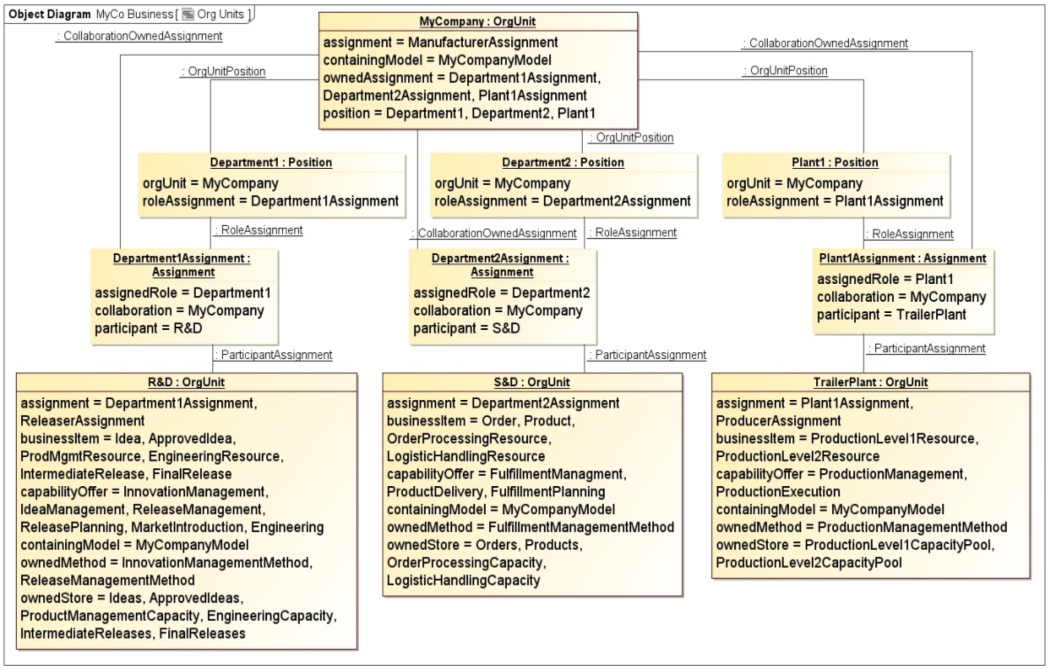

Object Diagram MyCo Business [Org Units]
MyCompany : OrgUnit
assignment = ManufacturerAssignment
containingModel = MyCompanyModel
ownedAssignment = Department1Assignment, Department2Assignment, Plant1Assignment
position = Department1, Department2, Plant1
: CollaborationOwnedAssignment
: OrgUnitPosition
Department1 : Position
orgUnit = MyCompany
roleAssignment = Department1Assignment
Department2 : Position
orgUnit = MyCompany
roleAssignment = Department2Assignment
Plant1 : Position
orgUnit = MyCompany
roleAssignment = Plant1Assignment
: RoleAssignment
Department1Assignment : Assignment
assignedRole = Department1
collaboration = MyCompany
participant = R&D
Department2Assignment : Assignment
assignedRole = Department2
collaboration = MyCompany
participant = S&D
Plant1Assignment : Assignment
assignedRole = Plant1
collaboration = MyCompany
participant = TrailerPlant
: ParticipantAssignment
R&D : OrgUnit
assignment = Department1Assignment, ReleaserAssignment
businessItem = Idea, ApprovedIdea, ProdMgmtResource, EngineeringResource, IntermediateRelease, FinalRelease
capabilityOffer = InnovationManagement, IdeaManagement, ReleaseManagement, ReleasePlanning, MarketIntroduction, Engineering
containingModel = MyCompanyModel
ownedMethod = InnovationManagementMethod, ReleaseManagementMethod
ownedStore = Ideas, ApprovedIdeas, ProductManagementCapacity, EngineeringCapacity, IntermediateReleases, FinalReleases
S&D : OrgUnit
assignment = Department2Assignment
businessItem = Order, Product, OrderProcessingResource, LogisticHandlingResource
capabilityOffer = FulfillmentManagment, ProductDelivery, FulfillmentPlanning
containingModel = MyCompanyModel
ownedMethod = FulfillmentManagementMethod
ownedStore = Orders, Products, OrderProcessingCapacity, LogisticHandlingCapacity
TrailerPlant : OrgUnit
assignment = Plant1Assignment, ProducerAssignment
businessItem = ProductionLevel1Resource, ProductionLevel2Resource
capabilityOffer = ProductionManagement, ProductionExecution
containingModel = MyCompanyModel
ownedMethod = ProductionManagementMethod
ownedStore = ProductionLevel1CapacityPool, ProductionLevel2CapacityPool

Figure 27. Organization structure objects

As indicated in the conceptual diagram in Figure 28, distinction is made between capabilities and capability offers. An organization unit that provides a capability is actually having a capability offer for that capability. The name "capability" is reserved for types of capabilities that are maintained in a capability library. A capability library contains a taxonomy of capabilities, consisting of capability definitions and categories of them, and is applied to enforce consistency in the definition of capabilities. Multiple capability offers that are associated with the same capability, are considered similar from the perspective of the library. Such libraries might be industry-standard, company-standard, but might also be specific to a particular modeling effort. A capability offer represents "*the ability to perform a particular kind of work and deliver desired value, by applying resources that are managed together, possibly based on formalized methods*".

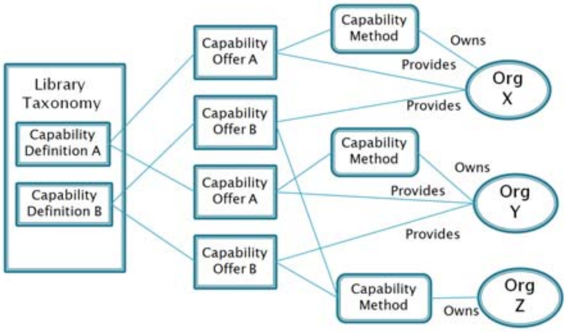


Figure 28. Capability management

As Figure 28 also indicates, a capability method that supports a capability offer might be owned by the same organization unit that also provides the capability offer, but might also be owned by a different organization unit, e.g. in case a certain method is enforced on multiple organization units, all providing their capability offer, for the same capability. In the XTrailer use case example we will only consider the situation in which an organization unit that uses a capability method to support its capability offer, also owns that method. The model view mockup in Figure 29 shows the capability library that is used in the XTrailer use case example.

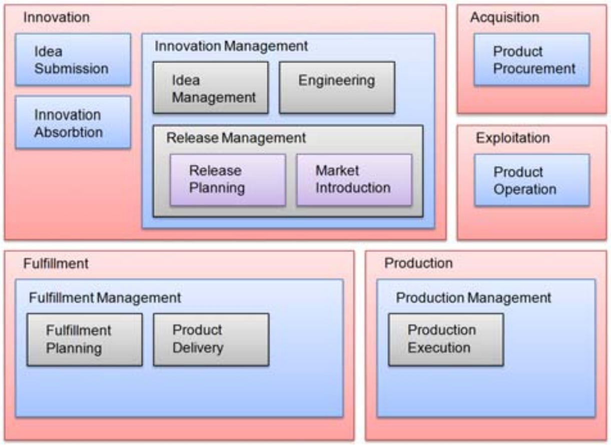


Figure 29. Capability library mockup

This is a view on the model objects in the object diagram in Figure 30.

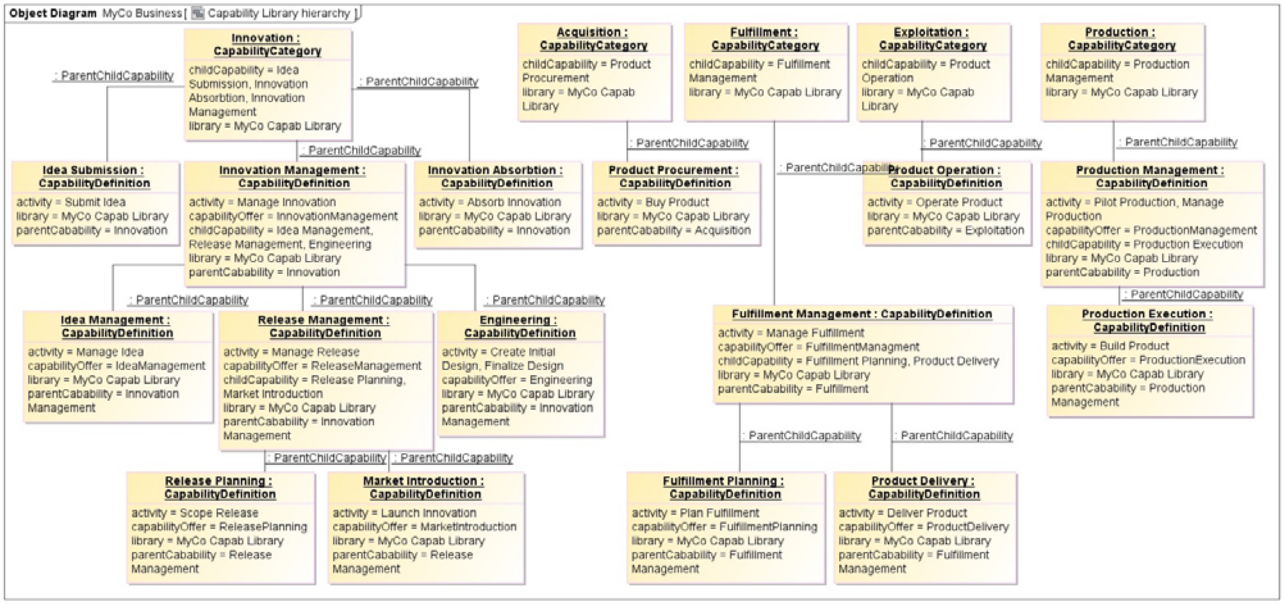

Object Diagram MyCo Business [Capability Library hierarchy]
Innovation : CapabilityCategory
childCapability = Idea Submission, Innovation Absorbtion, Innovation Management
library = MyCo Capab Library
Acquisition : CapabilityCategory
childCapability = Product Procurement
library = MyCo Capab Library
Fulfillment : CapabilityCategory
childCapability = Fulfillment Management
library = MyCo Capab Library
Exploitation : CapabilityCategory
childCapability = Product Operation
library = MyCo Capab Library
Production : CapabilityCategory
childCapability = Production Management
library = MyCo Capab Library
: ParentChildCapability
Idea Submission : CapabilityDefinition
activity = Submit Idea
library = MyCo Capab Library
parentCabability = Innovation
Innovation Management : CapabilityDefinition
activity = Manage Innovation
capabilityOffer = InnovationManagement
childCapability = Idea Management, Release Management, Engineering
library = MyCo Capab Library
parentCabability = Innovation
Innovation Absorbtion : CapabilityDefinition
activity = Absorb Innovation
library = MyCo Capab Library
parentCabability = Innovation
Product Procurement : CapabilityDefinition
activity = Buy Product
library = MyCo Capab Library
parentCabability = Acquisition
Product Operation : CapabilityDefinition
activity = Operate Product
library = MyCo Capab Library
parentCabability = Exploitation
Production Management : CapabilityDefinition
activity = Pilot Production, Manage Production
capabilityOffer = ProductionManagement
childCapability = Production Execution
library = MyCo Capab Library
parentCabability = Production
Idea Management : CapabilityDefinition
activity = Manage Idea
capabilityOffer = IdeaManagement
library = MyCo Capab Library
parentCabability = Innovation Management
Release Management : CapabilityDefinition
activity = Manage Release
capabilityOffer = ReleaseManagement
childCapability = Release Planning, Market Introduction
library = MyCo Capab Library
parentCabability = Innovation Management
Engineering : CapabilityDefinition
activity = Create Initial Design, Finalize Design
capabilityOffer = Engineering
library = MyCo Capab Library
parentCabability = Innovation Management
Fulfillment Management : CapabilityDefinition
activity = Manage Fulfillment
capabilityOffer = FulfillmentManagment
childCapability = Fulfillment Planning, Product Delivery
library = MyCo Capab Library
parentCabability = Fulfillment
Production Execution : CapabilityDefinition
activity = Build Product
capabilityOffer = ProductionExecution
library = MyCo Capab Library
parentCabability = Production Management
Release Planning : CapabilityDefinition
activity = Scope Release
capabilityOffer = ReleasePlanning
library = MyCo Capab Library
parentCabability = Release Management
Market Introduction : CapabilityDefinition
activity = Launch Innovation
capabilityOffer = MarketIntroduction
library = MyCo Capab Library
parentCabability = Release Management
Fulfillment Planning : CapabilityDefinition
activity = Plan Fulfillment
capabilityOffer = FulfillmentPlanning
library = MyCo Capab Library
parentCabability = Fulfillment Management
Product Delivery : CapabilityDefinition
activity = Deliver Product
capabilityOffer = ProductDelivery
library = MyCo Capab Library
parentCabability = Fulfillment Management

Figure 30. Capability library objects

These objects instantiate the classes in the metamodel diagram in Figure 31 .

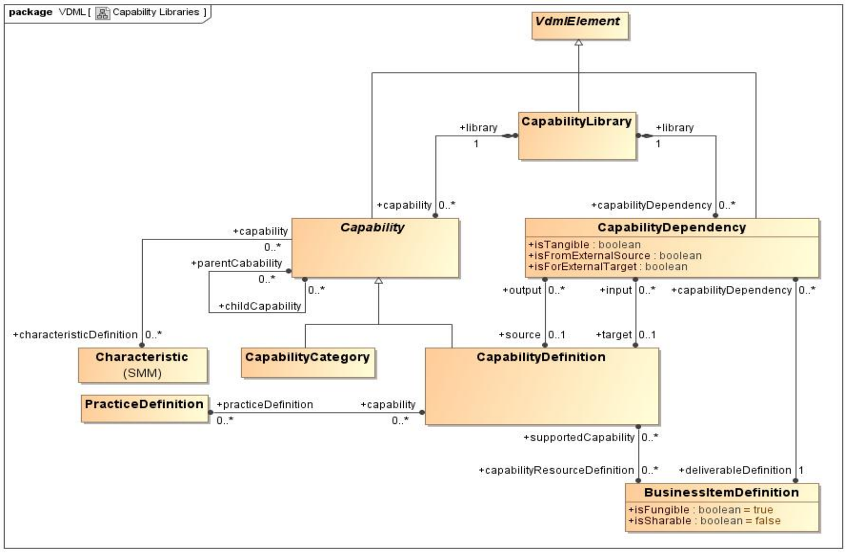


Figure 31. Capability Libraries metamodel

Note that some classes in Figure 31 have not been implemented in the XTrailer use case example.

A value delivery modeling tool can also use a capability library to guide the modeling user in productively discovering:

- Activities; as will be discussed later, an activity defines its capability requirement by referencing a capability in the library.
- Capability offers; multiple capability offers might refer to the same capability in the library.
- Capability resources; resources are business items, which can reference business items definitions, associated with capability definitions in the library.
- Inputs and outputs that capability methods need to handle, given the capability that they support; we will discuss details of this later.

The linkage between capability offer and capability will also provide the opportunity to rationalize capability offers, by e.g. comparing them, e.g. based on the capability methods and resources that support these, or based on performance measurements (see 3.1.5), and based on these comparisons, deciding to e.g. combine capability offers where appropriate, e.g. to improve economy-of-scale.

As is indicated in Figure 31, capability definitions can be associated with practice definitions. Practice definitions are maintained in practice libraries, the meta-model diagram of which is provided in Figure 32.

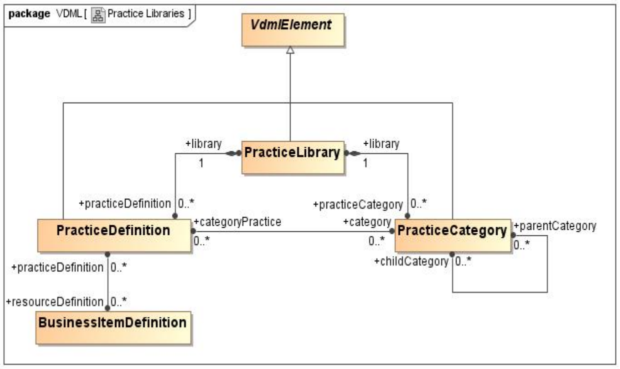


Figure 32. Practice Libraries metamodel

A practice is "*a proven way to handle specific types of work and that have been successfully used by multiple organizations*". Practices have not been applied in the XTrailer use case example and are not further discussed in this document.

Figure 33 provides a model view mockup of the R&D organization unit, together with its capability offers that it provides (the hexagons on the R&D boundary), the capability methods that support these capability offers (the rectangles with activity-flow marker, inside the R&D boundary) and the resource stores and pools that support the capability offers (the triangles without marker and the ones with re-use marker respectively). Dotted lines indicate how a capability method that supports one capability offer can depend on other capability offers.

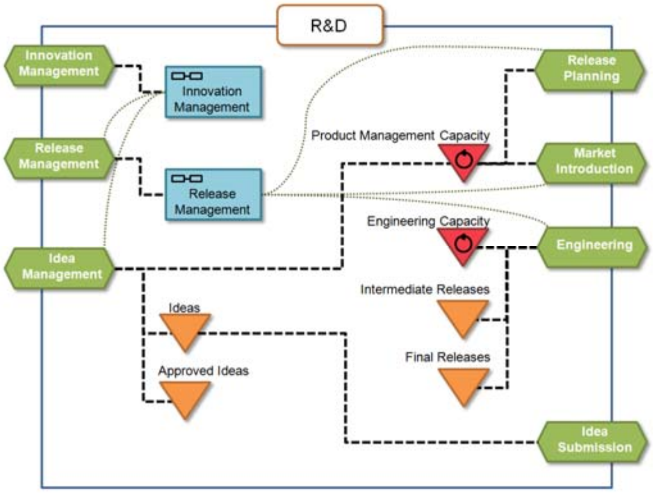


Figure 33. R&D capability management mockup

The capability methods are themselves collaborations. Their details will be discussed in 3.1.4.

A store represents "*a container of resource*". The resource that is received or provided is identified by a business item. A pool is a specialized store. A pool is "*a store that contains re-usable resource, i.e. resource that is returned to the pool after having been used, so that it is again available for use*".

As will be discussed in 3.1.4, stores are fundamental to decouple, or loosely couple collaborations.

The objects that are exposed via the model view mockup in Figure 33, are represented in the object diagram in Figure 34.

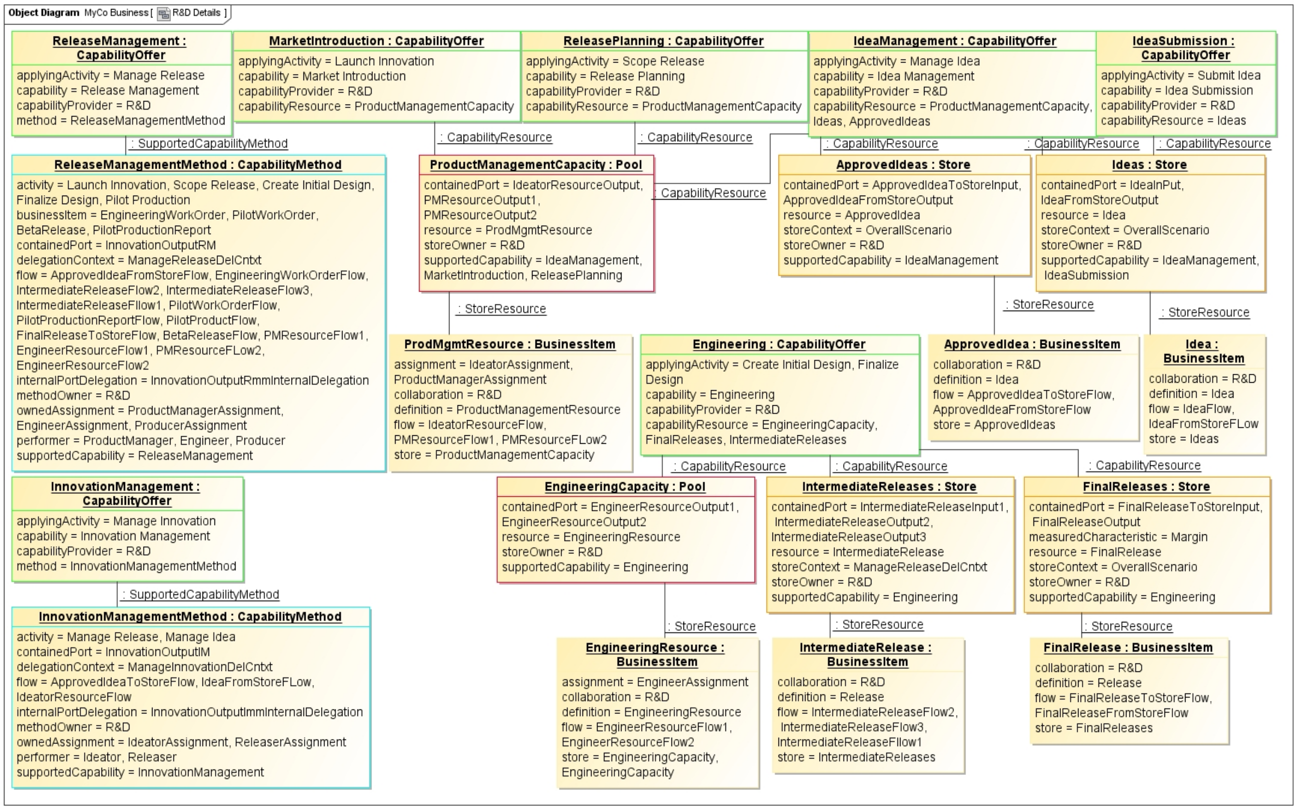

Object Diagram MyCo Business [R&D Details]
ReleaseManagement : CapabilityOffer
applyingActivity = Manage Release
capability = Release Management
capabilityProvider = R&D
method = ReleaseManagementMethod
MarketIntroduction : CapabilityOffer
applyingActivity = Launch Innovation
capability = Market Introduction
capabilityProvider = R&D
capabilityResource = ProductManagementCapacity
ReleasePlanning : CapabilityOffer
applyingActivity = Scope Release
capability = Release Planning
capabilityProvider = R&D
capabilityResource = ProductManagementCapacity
IdeaManagement : CapabilityOffer
applyingActivity = Manage Idea
capability = Idea Management
capabilityProvider = R&D
capabilityResource = ProductManagementCapacity, Ideas, ApprovedIdeas
IdeaSubmission : CapabilityOffer
applyingActivity = Submit Idea
capability = Idea Submission
capabilityProvider = R&D
capabilityResource = Ideas
: SupportedCapabilityMethod
: CapabilityResource
: CapabilityResource
: CapabilityResource
: CapabilityResource
: CapabilityResource
ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement
ProductManagementCapacity : Pool
containedPort = IdeatorResourceOutput, PMResourceOutput1, PMResourceOutput2
resource = ProdMgmtResource
storeOwner = R&D
supportedCapability = IdeaManagement, MarketIntroduction, ReleasePlanning
: CapabilityResource
ApprovedIdeas : Store
containedPort = ApprovedIdeaToStoreInput, ApprovedIdeaFromStoreOutput
resource = ApprovedIdea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement
Ideas : Store
containedPort = IdeaInPut, IdeaFromStoreOutput
resource = Idea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement, IdeaSubmission
: StoreResource
: StoreResource
: StoreResource
ProdMgmtResource : BusinessItem
assignment = IdeatorAssignment, ProductManagerAssignment
collaboration = R&D
definition = ProductManagementResource
flow = IdeatorResourceFlow, PMResourceFlow1, PMResourceFLow2
store = ProductManagementCapacity
Engineering : CapabilityOffer
applyingActivity = Create Initial Design, Finalize Design
capability = Engineering
capabilityProvider = R&D
capabilityResource = EngineeringCapacity, FinalReleases, IntermediateReleases
ApprovedIdea : BusinessItem
collaboration = R&D
definition = Idea
flow = ApprovedIdeaToStoreFlow, ApprovedIdeaFromStoreFlow
store = ApprovedIdeas
Idea : BusinessItem
collaboration = R&D
definition = Idea
flow = IdeaFlow, IdeaFromStoreFLow
store = Ideas
: CapabilityResource
: CapabilityResource
: CapabilityResource
InnovationManagement : CapabilityOffer
applyingActivity = Manage Innovation
capability = Innovation Management
capabilityProvider = R&D
method = InnovationManagementMethod
EngineeringCapacity : Pool
containedPort = EngineerResourceOutput1, EngineerResourceOutput2
resource = EngineeringResource
storeOwner = R&D
supportedCapability = Engineering
IntermediateReleases : Store
containedPort = IntermediateReleaseInput1, IntermediateReleaseOutput2, IntermediateReleaseOutput3
resource = IntermediateRelease
storeContext = ManageReleaseDelCntxt
storeOwner = R&D
supportedCapability = Engineering
FinalReleases : Store
containedPort = FinalReleaseToStoreInput, FinalReleaseOutput
measuredCharacteristic = Margin
resource = FinalRelease
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = Engineering
: SupportedCapabilityMethod
: StoreResource
: StoreResource
: StoreResource
InnovationManagementMethod : CapabilityMethod
activity = Manage Release, Manage Idea
containedPort = InnovationOutputIM
delegationContext = ManageInnovationDelCntxt
flow = ApprovedIdeaToStoreFlow, IdeaFromStoreFLow, IdeatorResourceFlow
internalPortDelegation = InnovationOutputImmInternalDelegation
methodOwner = R&D
ownedAssignment = IdeatorAssignment, ReleaserAssignment
performer = Ideator, Releaser
supportedCapability = InnovationManagement
EngineeringResource : BusinessItem
assignment = EngineerAssignment
collaboration = R&D
definition = EngineeringResource
flow = EngineerResourceFlow1, EngineerResourceFlow2
store = EngineeringCapacity, EngineeringCapacity
IntermediateRelease : BusinessItem
collaboration = R&D
definition = Release
flow = IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1
store = IntermediateReleases
FinalRelease : BusinessItem
collaboration = R&D
definition = Release
flow = FinalReleaseToStoreFlow, FinalReleaseFromStoreFlow
store = FinalReleases

Figure 34. R&D capability management objects

Objects in the object diagram in Figure 34 are instances of classes in the Organization Units and Capabilities metamodel diagram Figure 35, and the Stores metamodel diagram in Figure 36.

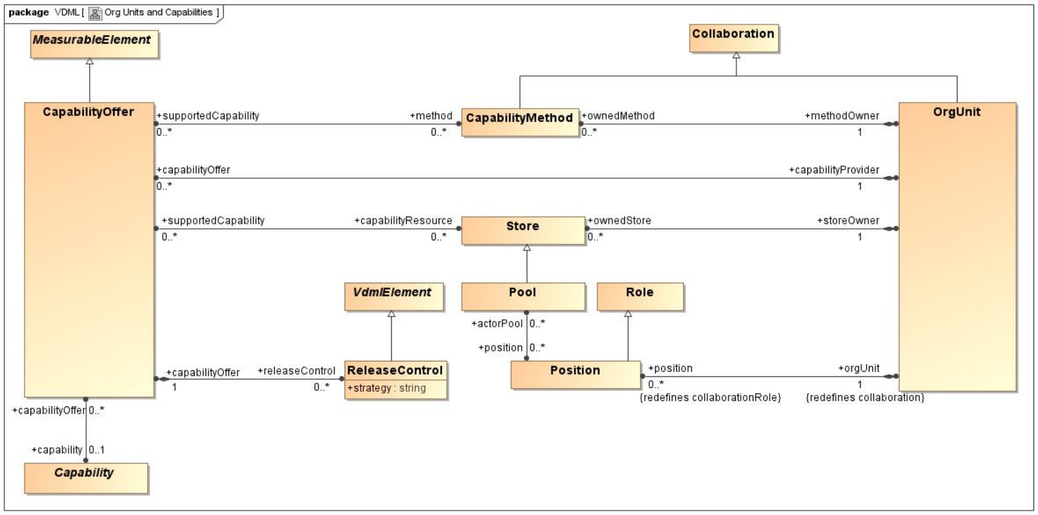


Figure 35. Organizations and Capabilities metamodel

It is important to note the association between Pool and Position in the metamodel diagram in Figure 35. A position defines a role in an organization unit. Resources that an organization unit has to perform work, can be put in positions. Positions can be assigned to other roles, or to actors. As discussed, actors might be human or non-human. Actors are individuals. In larger scale organizations, it might not be wanted to have all positions (and maybe actors in positions) defined individually, before any meaningful analysis and simulation can be done, based on value delivery models. It is important to enable analysis and simulation, based on modeling aggregated resource capacity via pools. VDML enables to model just pools, and to omit modeling positions of actors. This is also the approach that we follow in the XTrailer use case example. It might also be possible to just model a selective few positions. By associating positions with pools, it is defined which capability offer(s) they support, as well as which portion of aggregated pool capacity is "covered" by explicitly modeled positions.

Some elements, essential to support simulation, such as release control in Figure 35 and calendar service in Figure 36, have not been applied in the XTrailer use case example. Measured characteristics will be discussed later. The measurement framework that is used in VDML, based on integration with SMM will e.g. allow defining unit of measure of inventory levels of stores (and pools) and pool sizes of pools. For instance, measuring the size of the Engineering Capacity pool in "pieces" (numbers of engineers), or in "hours" (available capacity hours). Pool size expresses the total capacity associated with a pool, whereas inventory level expresses the amount of resource actually available in the store (or pool). These characteristics are essential to simulation. "Manual" analysis will normally be based on just averages. In "manual" analysis, the impact of a pool, due to capacity availability, might be defined as simple as an additional delay that a pool incurs on the duration of activities. As will be discussed later, many elements in VDML, including stores (and so pools), are "measurable", allowing a modeler to define custom measured characteristics on these elements. The "additional delay" characteristic is a possible example.

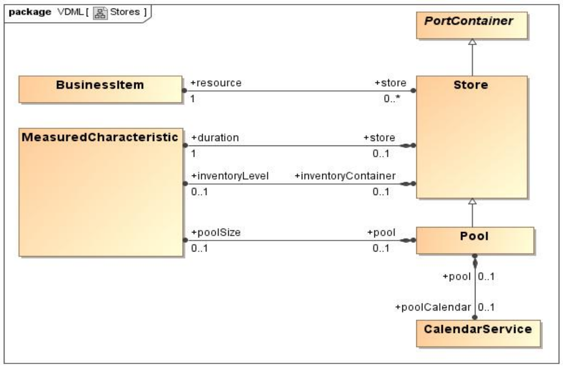


Figure 36. Stores metamodel

Figure 37 provides a similar model view mockup of the S&D organization unit, its capability offers and its methods and resources.

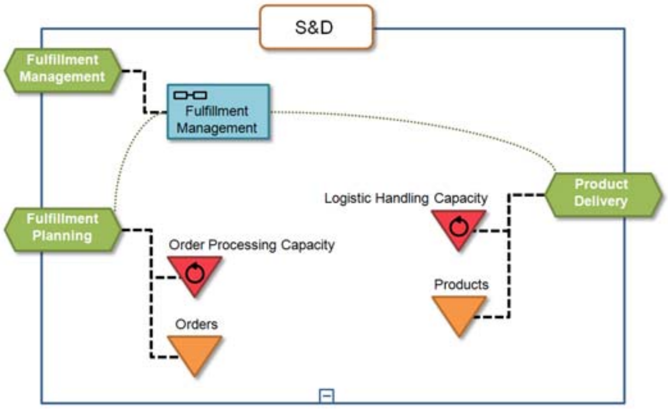


Figure 37. S&D capability management mockup

The corresponding object diagram is presented in Figure 38.

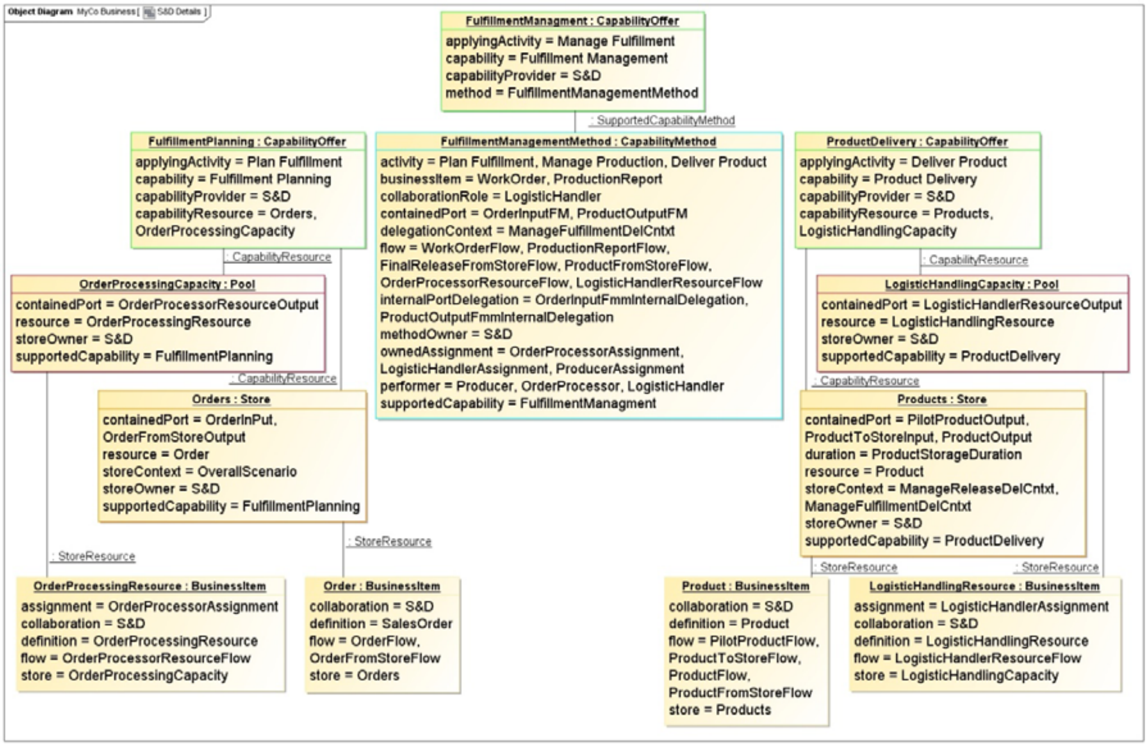

Object Diagram MyCo Business [S&D Details]
FulfillmentManagment : CapabilityOffer
applyingActivity = Manage Fulfillment
capability = Fulfillment Management
capabilityProvider = S&D
method = FulfillmentManagementMethod
: SupportedCapabilityMethod
FulfillmentPlanning : CapabilityOffer
applyingActivity = Plan Fulfillment
capability = Fulfillment Planning
capabilityProvider = S&D
capabilityResource = Orders, OrderProcessingCapacity
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
ProductDelivery : CapabilityOffer
applyingActivity = Deliver Product
capability = Product Delivery
capabilityProvider = S&D
capabilityResource = Products, LogisticHandlingCapacity
: CapabilityResource
OrderProcessingCapacity : Pool
containedPort = OrderProcessorResourceOutput
resource = OrderProcessingResource
storeOwner = S&D
supportedCapability = FulfillmentPlanning
: CapabilityResource
LogisticHandlingCapacity : Pool
containedPort = LogisticHandlerResourceOutput
resource = LogisticHandlingResource
storeOwner = S&D
supportedCapability = ProductDelivery
: CapabilityResource
: CapabilityResource
Orders : Store
containedPort = OrderInPut, OrderFromStoreOutput
resource = Order
storeContext = OverallScenario
storeOwner = S&D
supportedCapability = FulfillmentPlanning
Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery
: StoreResource
: StoreResource
: StoreResource
: StoreResource
OrderProcessingResource : BusinessItem
assignment = OrderProcessorAssignment
collaboration = S&D
definition = OrderProcessingResource
flow = OrderProcessorResourceFlow
store = OrderProcessingCapacity
Order : BusinessItem
collaboration = S&D
definition = SalesOrder
flow = OrderFlow, OrderFromStoreFlow
store = Orders
Product : BusinessItem
collaboration = S&D
definition = Product
flow = PilotProductFlow, ProductToStoreFlow, ProductFlow, ProductFromStoreFlow
store = Products
LogisticHandlingResource : BusinessItem
assignment = LogisticHandlerAssignment
collaboration = S&D
definition = LogisticHandlingResource
flow = LogisticHandlerResourceFlow
store = LogisticHandlingCapacity

Figure 38. S&D capability management objects

The model view mockup for the Trailer Plant's capability offers and related methods and resources is provided by Figure 39. As it will appear from the discussions in 3.1.4, the two pools that support Production Execution are considered alternative capacities. Production operators might be available as part of Production Level 2 Capacity. These are the certified operators, that can be involved in both pilot production (in the context of developing new trailer releases) and commercial production (producing trailers on customer orders, based on current releases). Production Level 1 Capacity represents the non-certified operators, which are only involved in pilot production of trailers. Production Level 2 Capacity can be deployed to do both commercial production work and pilot production work. The latter only if Production Level 1 Capacity, being expectedly cheaper, is not available. Production Level 1 Capacity is only available for pilot production work.

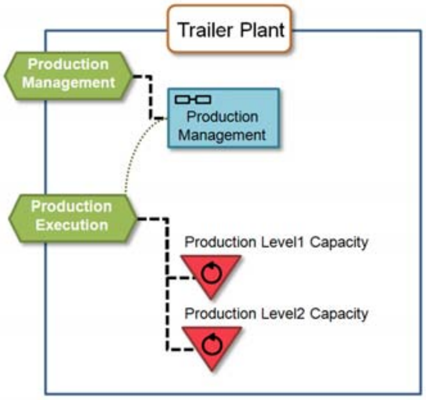


Figure 39. Trailer Plant capability management mockup

The object diagram, representing the objects that are exposed in the model view mockup in Figure 39 is presented in Figure 40.

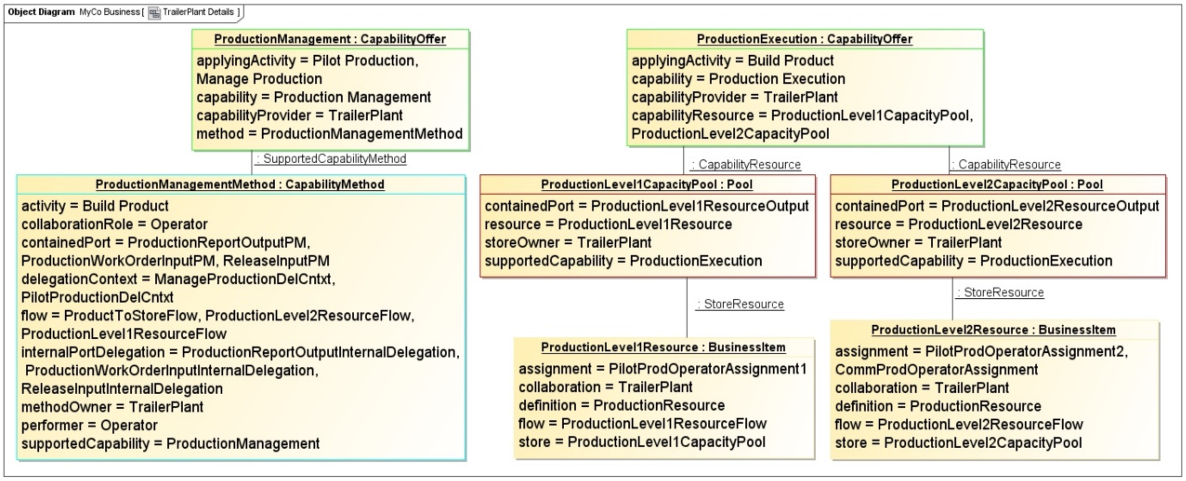

Object Diagram MyCo Business [TrailerPlant Details]
ProductionManagement : CapabilityOffer
applyingActivity = Pilot Production, Manage Production
capability = Production Management
capabilityProvider = TrailerPlant
method = ProductionManagementMethod
: SupportedCapabilityMethod
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
ProductionExecution : CapabilityOffer
applyingActivity = Build Product
capability = Production Execution
capabilityProvider = TrailerPlant
capabilityResource = ProductionLevel1CapacityPool, ProductionLevel2CapacityPool
: CapabilityResource
: CapabilityResource
ProductionLevel1CapacityPool : Pool
containedPort = ProductionLevel1ResourceOutput
resource = ProductionLevel1Resource
storeOwner = TrailerPlant
supportedCapability = ProductionExecution
ProductionLevel2CapacityPool : Pool
containedPort = ProductionLevel2ResourceOutput
resource = ProductionLevel2Resource
storeOwner = TrailerPlant
supportedCapability = ProductionExecution
: StoreResource
: StoreResource
ProductionLevel1Resource : BusinessItem
assignment = PilotProdOperatorAssignment1
collaboration = TrailerPlant
definition = ProductionResource
flow = ProductionLevel1ResourceFlow
store = ProductionLevel1CapacityPool
ProductionLevel2Resource : BusinessItem
assignment = PilotProdOperatorAssignment2, CommProdOperatorAssignment
collaboration = TrailerPlant
definition = ProductionResource
flow = ProductionLevel2ResourceFlow
store = ProductionLevel2CapacityPool

Figure 40. Trailer Plant capability management objects

So-far we have dealt with the structural parts of the XTrailer business system: its collaborations, capabilities and resources, and how these are related to and aligned with each other. Note however, that everywhere where a deliverable is exchanged, via a deliverable flow, there is a store or activity at the source, and a store or activity at the target of that flow. VDML defines a deliverable as a "*product or service produced by an activity or delivered from a store that can be conveyed to another activity or store*". Though in the meta-model provisioning or producing and receiving or consuming are handled via activities or stores, graphical diagrams can abstract away activity and store details, and just depict a deliverable flow as occurring between providing and receiving roles. That is why we could defer analysis of activities so-far. The 3.1.4 will deal with the behavioral parts of the business system, based on activities.

3.1.4 Activities

An activity represents "*work contributed to a collaboration by a participant in a role of the collaboration*". A role may contribute to multiple activities in the same collaboration. As the VDML ontology diagram in Figure 41 highlights, roles perform activities, and activities require capabilities. These aspects will be analyzed in detail in this section, in the context of the XTrailer use case example. Activities also create and/or consume value. This latter aspect will be discussed in 3.1.5.

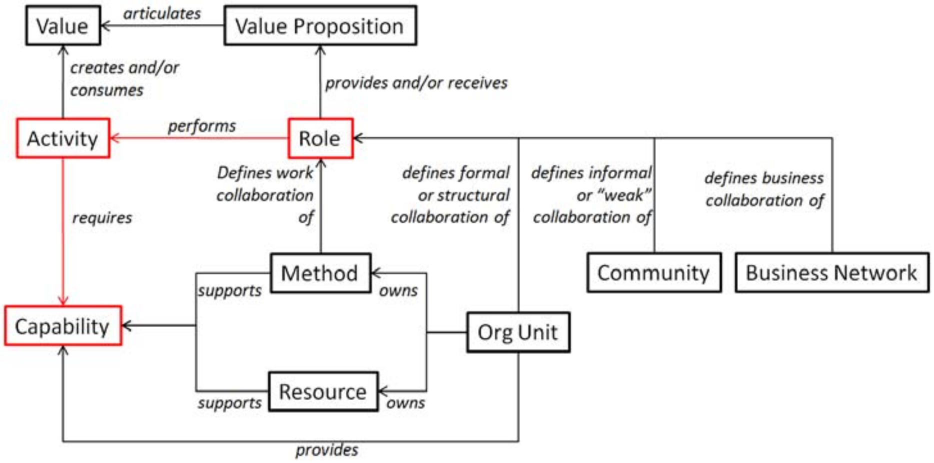


Figure 41. VDML high-level ontology: Activity highlighted

In a value delivery model, activities may be designed by following these steps:

- Define which capability is required in order to perform the work. This capability requirement can be defined based on a reference to a capability in a capability library.

- Find organization units that provide capability offers for the capability that is required, and select one capability offer. This selection might be supported by assessment of measurements of performance and value contribution. Performance and value measurement will be discussed in 3.1.5.
- When the capability offer is supported by a capability method, which it typically the case with broader capabilities, delegate the work of the activity to that method, and assign the activity performing role to the organization that provides the capability offer of choice.
- When the capability is more atomic, no method might be defined, but the capability offer might be supported directly by resources, including people. These resources can then be considered for use by the activity. A subset of them may also be assigned the activity performing role.
- It is also possible to define assignment of method roles specifically in the context of the delegation to that method.

In this section, these steps will all be demonstrated in the context of the XTrailer use case example.

Note that more variations are possible, such as:

- When an organization unit provides just one capability offer, and all it does is performed in the context of just that capability, there might be no need to organize activities in a separately defined capability method. Rather the organization unit, as collaboration, can contain activities directly, and activities can delegate their work to the organization unit directly. It is even possible to delegate work to communities and business networks, though matching of required capability versus offered capability cannot be performed explicitly then.
- When an organization unit provides a capability offer that matches the capability that is required by an activity, and the activity delegates its work to that method accordingly, the activity performing role maybe be assigned to the method itself, instead of to the organization unit that provides the capability offer.

These and other possible variations will not be demonstrated in the XTrailer use case.

We now revisit the XTrailer business network collaboration. Its role collaboration view, or “value network” view was already represented by the model view mockup in Figure 13. Figure 42 provides a model view mockup of its activity network.

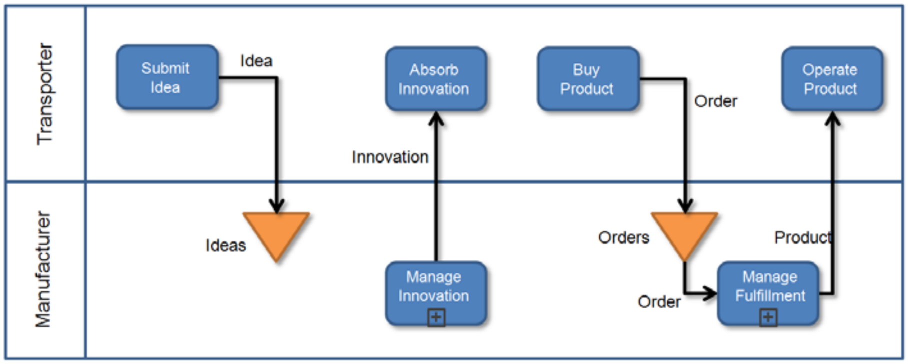


Figure 42. Business network’s activity network mockup

Note that both the role collaboration view, and the activity network view are views on the same underlying model. Figure 43 combines both the view in one picture, for convenience and ease of recognition.

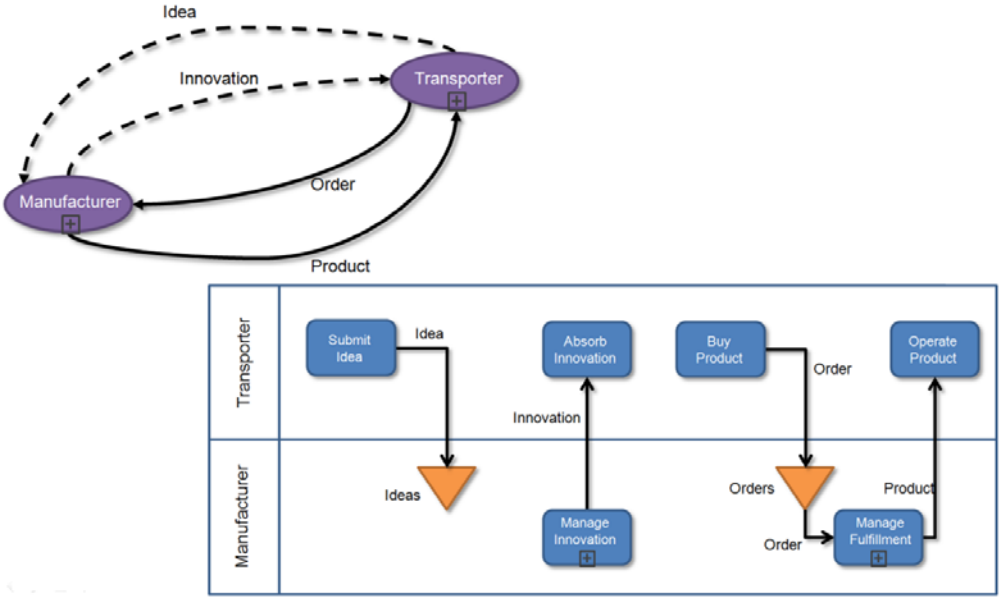


Figure 43. Business network: role collaboration versus activity network

Roles, in one view represented as ovals, in the other as swim lanes, are the same. Four out of five deliverable flows exposed in both views. The activity network view is providing detail that is abstracted away in the role collaboration view. These details are:

- Activities of the business network. Note that business networks, as any collaboration, can contain activities, according to the Collaborations metamodel diagram that was already provided in Figure 16.
- Stores that are owned by the organization that fills the Manufacturer party role. Though the Idea flow into the Ideas store, and the Order flow into the Orders store are shown in the role collaboration view, the stores themselves aren't. These store inputs are "handled" under the responsibility of the Manufacturer role in the business network collaboration (see also the object diagram in Figure 44. The Ideas store is owned by R&D and the Orders store by S&D, both being organization units that are part of MyCompany.
- The deliverable flow of Order, from the Orders store to the Manage Fulfillment activity.

Two activities, "Manage Innovation" and "Manage fulfillment", are represented by a rectangle shape that contains an "expansion button" (a small box with "+" sign in it), in Figure 42. These activities "delegate" their work to another collaboration, as will be discussed below. Basically these activities represent the use of sub-collaborations, in the context of the business network collaboration.

Figure 44 shows the object diagram that contains the objects behind the activity network view in Figure 42.

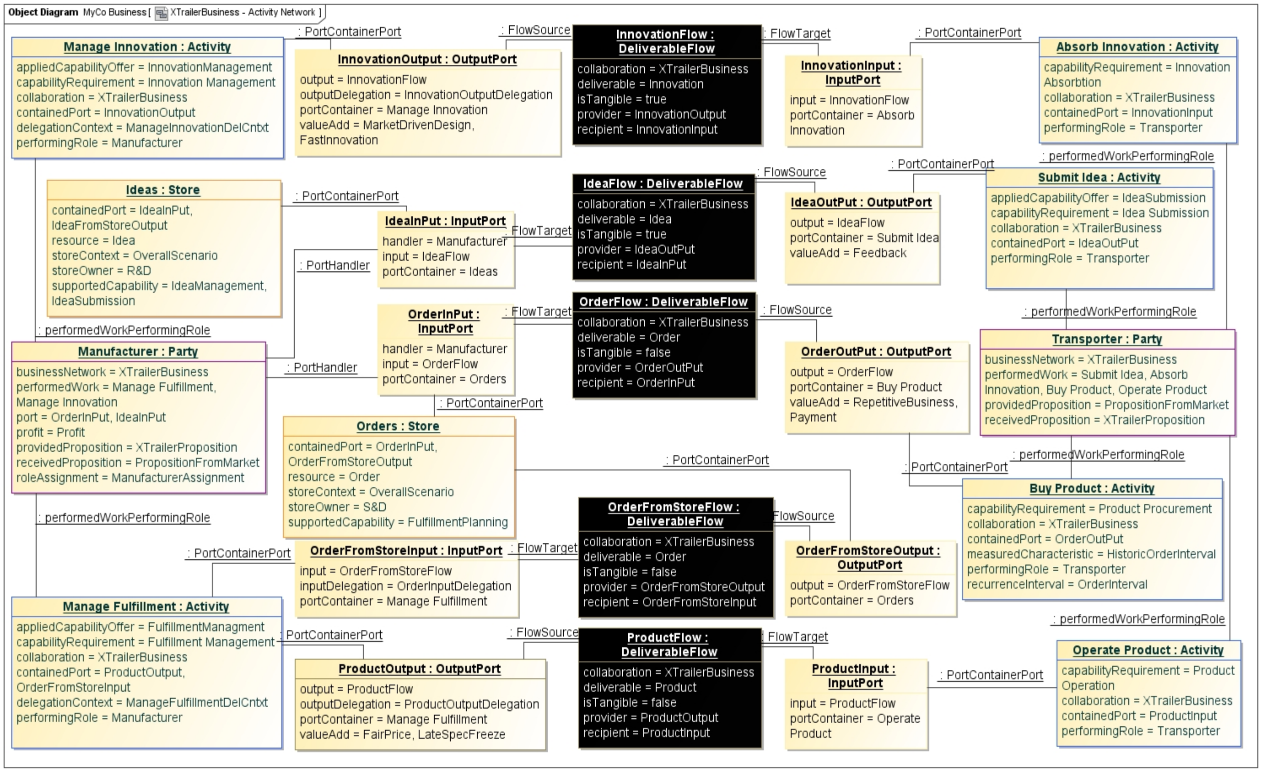

Object Diagram MyCo Business [XTrailerBusiness - Activity Network]
Manage Innovation : Activity
appliedCapabilityOffer = InnovationManagement
capabilityRequirement = Innovation Management
collaboration = XTrailerBusiness
containedPort = InnovationOutput
delegationContext = ManageInnovationDelCntxt
performingRole = Manufacturer
: PortContainerPort
InnovationOutput : OutputPort
output = InnovationFlow
outputDelegation = InnovationOutputDelegation
portContainer = Manage Innovation
valueAdd = MarketDrivenDesign, FastInnovation
: FlowSource
InnovationFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Innovation
isTangible = true
provider = InnovationOutput
recipient = InnovationInput
: FlowTarget
InnovationInput : InputPort
input = InnovationFlow
portContainer = Absorb Innovation
: PortContainerPort
Absorb Innovation : Activity
capabilityRequirement = Innovation Absorbtion
collaboration = XTrailerBusiness
containedPort = InnovationInput
performingRole = Transporter
: performedWorkPerformingRole
Ideas : Store
containedPort = IdeaInPut, IdeaFromStoreOutput
resource = Idea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement, IdeaSubmission
: PortContainerPort
IdeaInPut : InputPort
handler = Manufacturer
input = IdeaFlow
portContainer = Ideas
: FlowTarget
: PortHandler
IdeaFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Idea
isTangible = true
provider = IdeaOutPut
recipient = IdeaInPut
: FlowSource
IdeaOutPut : OutputPort
output = IdeaFlow
portContainer = Submit Idea
valueAdd = Feedback
: PortContainerPort
Submit Idea : Activity
appliedCapabilityOffer = IdeaSubmission
capabilityRequirement = Idea Submission
collaboration = XTrailerBusiness
containedPort = IdeaOutPut
performingRole = Transporter
: performedWorkPerformingRole
: performedWorkPerformingRole
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
port = OrderInPut, IdeaInPut
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
: PortHandler
OrderInPut : InputPort
handler = Manufacturer
input = OrderFlow
portContainer = Orders
: FlowTarget
OrderFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Order
isTangible = false
provider = OrderOutPut
recipient = OrderInPut
: FlowSource
OrderOutPut : OutputPort
output = OrderFlow
portContainer = Buy Product
valueAdd = RepetitiveBusiness, Payment
: performedWorkPerformingRole
Transporter : Party
businessNetwork = XTrailerBusiness
performedWork = Submit Idea, Absorb Innovation, Buy Product, Operate Product
providedProposition = PropositionFromMarket
receivedProposition = XTrailerProposition
: PortContainerPort
Orders : Store
containedPort = OrderInPut, OrderFromStoreOutput
resource = Order
storeContext = OverallScenario
storeOwner = S&D
supportedCapability = FulfillmentPlanning
: PortContainerPort
: PortContainerPort
: performedWorkPerformingRole
Buy Product : Activity
capabilityRequirement = Product Procurement
collaboration = XTrailerBusiness
containedPort = OrderOutPut
measuredCharacteristic = HistoricOrderInterval
performingRole = Transporter
recurrenceInterval = OrderInterval
: performedWorkPerformingRole
: PortContainerPort
OrderFromStoreInput : InputPort
input = OrderFromStoreFlow
inputDelegation = OrderInputDelegation
portContainer = Manage Fulfillment
: FlowTarget
OrderFromStoreFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Order
isTangible = false
provider = OrderFromStoreOutput
recipient = OrderFromStoreInput
: FlowSource
OrderFromStoreOutput : OutputPort
output = OrderFromStoreFlow
portContainer = Orders
: performedWorkPerformingRole
Manage Fulfillment : Activity
appliedCapabilityOffer = FulfillmentManagment
capabilityRequirement = Fulfillment Management
collaboration = XTrailerBusiness
containedPort = ProductOutput, OrderFromStoreInput
delegationContext = ManageFulfillmentDelCntxt
performingRole = Manufacturer
: PortContainerPort
ProductOutput : OutputPort
output = ProductFlow
outputDelegation = ProductOutputDelegation
portContainer = Manage Fulfillment
valueAdd = FairPrice, LateSpecFreeze
: FlowSource
ProductFlow : DeliverableFlow
collaboration = XTrailerBusiness
deliverable = Product
isTangible = false
provider = ProductOutput
recipient = ProductInput
: FlowTarget
ProductInput : InputPort
input = ProductFlow
portContainer = Operate Product
: PortContainerPort
Operate Product : Activity
capabilityRequirement = Product Operation
collaboration = XTrailerBusiness
containedPort = ProductInput
performingRole = Transporter

Figure 44. Business network's activity network objects

Party roles, activities that they perform, stores and deliverable flows connecting activities and stores can be recognized easily. The object diagram also contains "ports", being input ports and output ports of activities and stores.

The objects in the object diagram in Figure 44 above are instances of classes in the metamodel diagrams of Port Containers (Figure 45), Activities (Figure 46), Stores (Figure 36) and Deliverable Flows (Figure 17). The notion of "port container" needs further explanation.

As was indicated earlier, in relation to the Deliverable Flows metamodel diagram in Figure 17, deliverable flows connect to other elements via input ports and output ports. They do not connect to activities and stores directly. They connect to "port containers". A port container is a container of ports. Port and port containers are fundamental parts of VDML metamodel "mechanics". A port is a connection point to a port container, used to handle inputs and outputs (e.g. consume inputs, produce outputs or delegate inputs or outputs to ports of other port containers). As will appear later, ports are also instrumental in carrying information, in particular value contribution measurements.

The Port Containers metamodel diagram is provided by Figure 45.

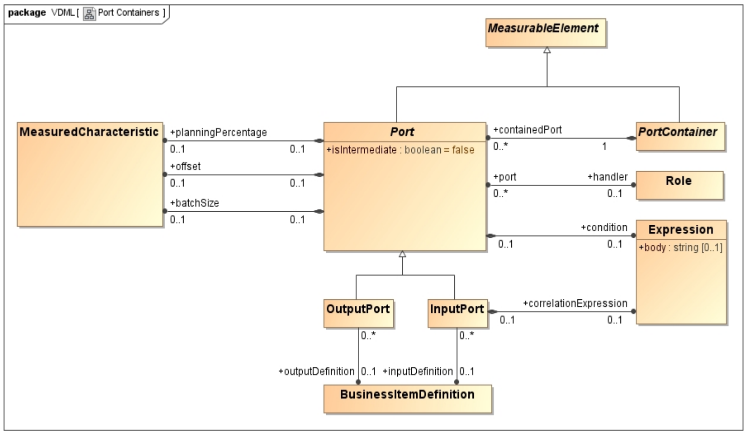


Figure 45. Port Containers metamodel

Both activities and stores are defined as port containers. The Stores metamodel, as provided earlier, in Figure 36, indicates how a store is defined as port container. Figure 46 below, containing a subset of the Activities metamodel, defines activity as a port container.

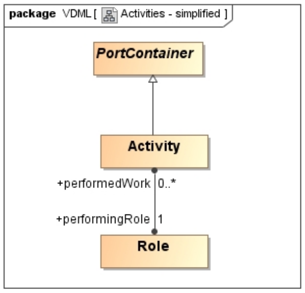


Figure 46. Activities metamodel (partial)

From the Collaborations metamodel diagram, provided earlier in Figure 16, it appears that a collaboration is a port container as well. The usefulness of that will appear from discussions later in the document.

The informal schema in Figure 47 provides an overview of what will be discussed in detail, based on formal models, in the remainder of this section. It represents all activities in the XTrailer use case example. Note that this use case is an over-simplification of reality, for sake of keeping the example small.

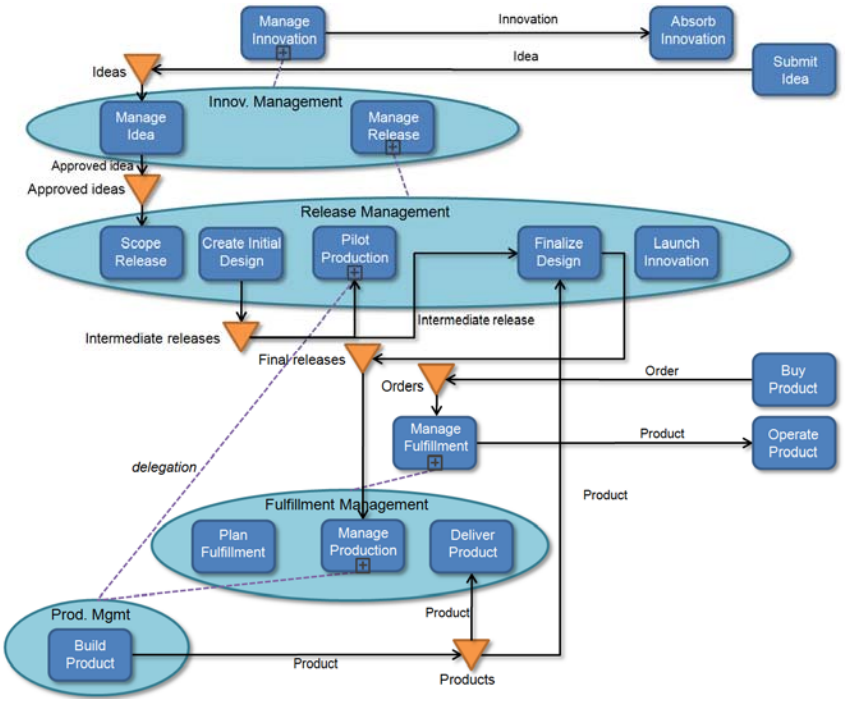


Figure 47. Use case overview: collaborations, stores and activities

The four ovals, containing activities, represent the capability methods that have been introduced in 3.1.3. The six activities, not contained in any of these, are the ones contained in the business network itself, and are the same as the activities in Figure 42 and Figure 43. Figure 47 provides an impression of how activities delegate their work to other collaborations (here capability methods), and how stores can serve as decoupling points between collaborations. Delegations are indicated by the dashed connectors in Figure 47.

When an activity delegates its work to another collaboration, inputs and outputs of the (parent) activity should be mapped to inputs and outputs of the (sub) collaboration. This is handled by "delegating" input and output ports from the activity to input and output ports of the collaboration. It is a common situation also, that collaborations, that are involved by activities, through delegation, independent of each other, do still depend on an input or output of each other. Example: the Manage Fulfilment activity in Figure 47 delegates its work to the Fulfillment Management method. Independent of this, the Release Management method is involved through delegation from the Manage Release activity in the Innovation Management method, which is involved through delegation from the Manage Innovation activity. Consequently, the two capability methods, Release Management and Fulfillment Management, run independent from each other. But the Manage Production activity does require a final release as input, which is an output of the Finalize Design activity in the Release Management method. The Finalize Design activity can pass on its output, a final release, as "side effect" deliverable, to the Final releases store, as owned by R&D (see Figure 39 above), from which the Manage Production activity can use it as an input resource. This way both collaborations can be loosely coupled (or actually decoupled), via the store. Consequently, communication of inputs to and outputs from collaborations can be achieved by a hybrid communication of (1) port delegation, in the context of an activity delegating its work to the collaboration, and (2) delivering outputs to a store or using inputs from a store.

As Figure 47 indicates, five activities delegate their work, each to one particular capability method. But two of these activities, Pilot production in the Release Management method, and Manage production in the Fulfillment Management method, delegate their work to the same method, Production Management. As will be discussed in detail below, each of these delegations comes with its own mapping of inputs and outputs (between delegating activity and the capability method to which the work is delegated), requires specific assignments of resources to performer roles inside the capability method, and results in its own set of measurements of performance and value contribution aspects. Per delegation a "delegation context" is set, to facilitate this. From Figure 47 it can be inferred that such delegation contexts form a “context tree” (a tree of analysis contexts). Measurements will be discussed in 3.1.5.

A delegation context is “*a specialized Analysis Context, set by an activity and in which the activity delegates its work to a collaboration; a delegation context also defines the delegations of activity inputs and/or outputs to/from collaboration inputs and/or outputs, and may define assignments of roles within the collaboration*”.

Analysis of a consistent business use case does require a broader “analysis context” than just one delegation context. Though most flow details are not presented in Figure 47 above, as they will be discussed based on detailed and formal models below, consider, for instance, the flow of ideas from the market, to innovations that flow back to the market. Consider for instance such measurements as the duration and cost of processing ideas into innovations, or analysis of how the ratio of successful ideas, that make it to launched innovations, along the flow. This requires an “analysis context” that does not only involve the Innovation Management and Release Management methods (both through delegation), but also the Ideas and Approved ideas stores, and probably the Intermediate releases store as well. Or consider, for instance, the build-up of cost of a product, consisting from an allocation of a portion of the cost of creating a new release, and the cost of producing and delivering the product. The “analysis context” would probably involve all methods and stores in Figure 47.

Consequently, in addition to “delegation contexts”, one or more sub-trees of delegation contexts come together in a “root context”, which VDML calls “scenario”. A scenario defines “*a consistent business use case of a VDML model by specifying a, possibly recursive, analysis context for elements in scope of that use case; the nesting of contexts allows a collaboration to be used as a sub-collaboration by more than one activity, each of which sets its particular context and measurements*”.

Though all methods, as indicated in Figure 47, are involved from and to support the XTrailer business network, and the use case example in this document does only focus on this single business network, the MyCompany organization is most likely involved in multiple such business networks, related to different types of trailers and/or different market segments or customers. All these business networks, having their own activities, do involve the same capability methods and reach out to the same stores, in the same organization units, etc. This will lead to a scenario tree, consisting of even more branches.

Both scenarios and delegation contexts are analysis contexts, an analysis context being defined as “*a set of measurements of a particular use of one or more collaborations or a store when used as a decoupling point between collaborations; when a collaboration is used by an activity, a context also defines the delegations of activity inputs and/or outputs to/from collaboration inputs and/or outputs, and may define assignments of roles within the collaboration*”.

Figure 48 provides an informal and conceptual representation of a context tree.

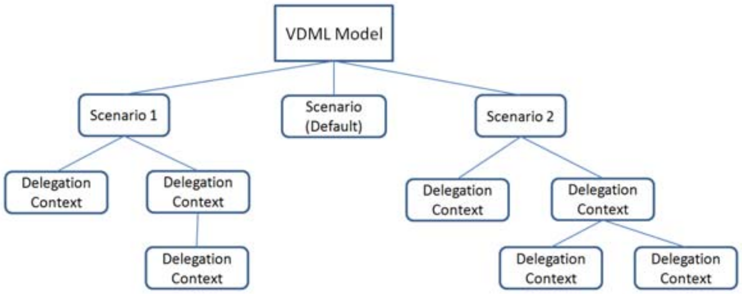


Figure 48. Context tree

Figure 48 suggests that the same VDML Model, or value delivery model, may contain multiple scenarios, one of which serves as default scenario. A default scenario can be used when various measurements, throughout multiple collaborations and stores, are indifferent towards variation of scenarios. They will only be provided once, via the default scenario. The use of multiple scenarios, based on the same model, can be particularly useful in the following situations:

- A scenario does only involve a subset of elements of a value delivery model, e.g. just all about innovation and release management, but no fulfilment.

- Alternative scenarios, analyzing the same model, or a subset of it, in different circumstances, whereby e.g. various context based aspects, such as measurements and role assignments are varied. One scenario might reflect the “as-is” situation, whereas other scenarios might reflect alternative “to be” situations. Measurements of “as-is” scenarios might be based on actual performance as observed in the real-world business situation that the model is supposed to represent. Measurements of “to-be” scenarios might be estimates as manually entered by a business analyst, or they might result from model-based simulations. Different simulations might result into different scenarios.

Figure 49 provides a model view mockup of the scenario tree in the XTrailer use case example.

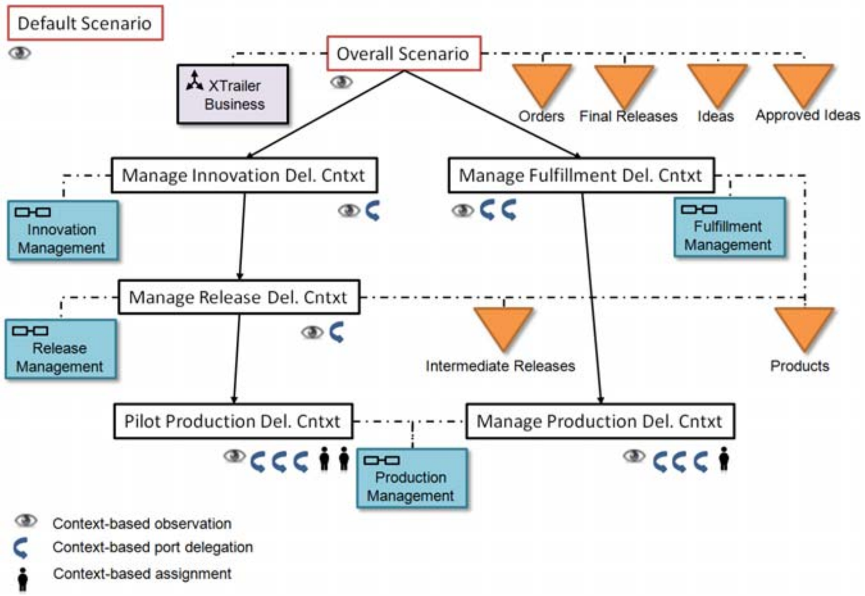


Figure 49. Context tree mockup

It shows that the Overall Scenario relates to the business network itself, as well as four of the stores. It contains an observation. As will be discussed in 3.1.5, an observation is the element that contains measurements, and in this case measurements of measured characteristics of measurable elements of the business network and the stores. We will revisit all about measurements in 3.1.5.

As the capability methods Innovation Management and Fulfillment Management are involved through delegation of the Manage innovation and Manage Fulfillment activities respectively, it can be understood why the Overall Scenario contains the delegation contexts “Manage Innovation Delegation Context” and “Manage Fulfillment Delegation Context”, that relate to the Manage Innovation and Manage Fulfillment methods respectively.

As already discussed earlier, the Production Management method is involved through delegation from both the Pilot Production activity in the Release Management method and the Manage Production activity in the Fulfillment Management method. This explains why the Production Management method is reached in two ways from the scenario context tree in Figure 49.

As will be discussed later, Products store-related measurements are required specifically in the context of the Manage Release activity, delegating to the Release Management method, as well as ion the context of the Manage Fulfillment activity, delegating to the Fulfillment Management method. This explains why also the Products store is reached in two ways from the scenario tree in Figure 49.

The model view mockup in Figure 49 also indicates, that, although a scenario only contains delegation contexts, and observations (i.e. sets of measurements), delegation contexts may also contain context-based port delegations and context-based assignments. We will discuss these in detail below, based on the XTrailer use case example.

Note that the scenario context tree in Figure 49 does not show pools and organization units. This is because we assume that, in the XTrailer use case example, any measurement that relates to these elements is contained in the observation of the default scenario, which is not expanded in Figure 49.

The object diagram, containing the objects behind the model view mockup in Figure 49, is shown in Figure 50.

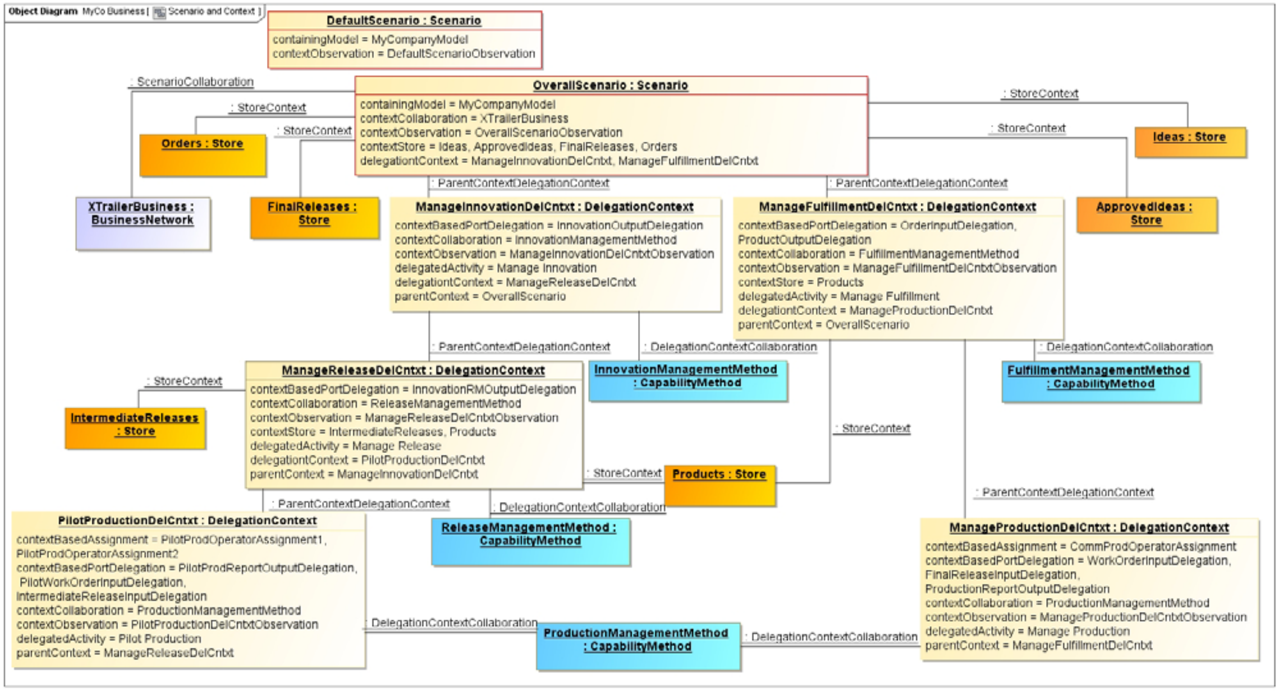

Object Diagram MyCo Business [Scenario and Context]
DefaultScenario : Scenario
containingModel = MyCompanyModel
contextObservation = DefaultScenarioObservation
: ScenarioCollaboration
OverallScenario : Scenario
containingModel = MyCompanyModel
contextCollaboration = XTrailerBusiness
contextObservation = OverallScenarioObservation
contextStore = Ideas, ApprovedIdeas, FinalReleases, Orders
delegationtContext = ManageInnovationDelCntxt, ManageFulfillmentDelCntxt
: StoreContext
: StoreContext
: StoreContext
: StoreContext
Orders : Store
Ideas : Store
XTrailerBusiness : BusinessNetwork
FinalReleases : Store
: ParentContextDelegationContext
: ParentContextDelegationContext
ManageInnovationDelCntxt : DelegationContext
contextBasedPortDelegation = InnovationOutputDelegation
contextCollaboration = InnovationManagementMethod
contextObservation = ManageInnovationDelCntxtObservation
delegatedActivity = Manage Innovation
delegationtContext = ManageReleaseDelCntxt
parentContext = OverallScenario
ManageFulfillmentDelCntxt : DelegationContext
contextBasedPortDelegation = OrderInputDelegation, ProductOutputDelegation
contextCollaboration = FulfillmentManagementMethod
contextObservation = ManageFulfillmentDelCntxtObservation
contextStore = Products
delegatedActivity = Manage Fulfillment
delegationtContext = ManageProductionDelCntxt
parentContext = OverallScenario
ApprovedIdeas : Store
: ParentContextDelegationContext
: DelegationContextCollaboration
: DelegationContextCollaboration
: StoreContext
ManageReleaseDelCntxt : DelegationContext
contextBasedPortDelegation = InnovationRMOutputDelegation
contextCollaboration = ReleaseManagementMethod
contextObservation = ManageReleaseDelCntxtObservation
contextStore = IntermediateReleases, Products
delegatedActivity = Manage Release
delegationtContext = PilotProductionDelCntxt
parentContext = ManageInnovationDelCntxt
InnovationManagementMethod : CapabilityMethod
FulfillmentManagementMethod : CapabilityMethod
IntermediateReleases : Store
: StoreContext
: StoreContext
Products : Store
: ParentContextDelegationContext
: ParentContextDelegationContext
: DelegationContextCollaboration
PilotProductionDelCntxt : DelegationContext
contextBasedAssignment = PilotProdOperatorAssignment1, PilotProdOperatorAssignment2
contextBasedPortDelegation = PilotProdReportOutputDelegation, PilotWorkOrderInputDelegation, IntermediateReleaseInputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = PilotProductionDelCntxtObservation
delegatedActivity = Pilot Production
parentContext = ManageReleaseDelCntxt
ReleaseManagementMethod : CapabilityMethod
ManageProductionDelCntxt : DelegationContext
contextBasedAssignment = CommProdOperatorAssignment
contextBasedPortDelegation = WorkOrderInputDelegation, FinalReleaseInputDelegation, ProductionReportOutputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = ManageProductionDelCntxtObservation
delegatedActivity = Manage Production
parentContext = ManageFulfillmentDelCntxt
: DelegationContextCollaboration
ProductionManagementMethod : CapabilityMethod
: DelegationContextCollaboration

Figure 50. Context tree objects

These objects are instances of classes in the Scenario and Analysis Context meta-model that is shown in Figure 51.

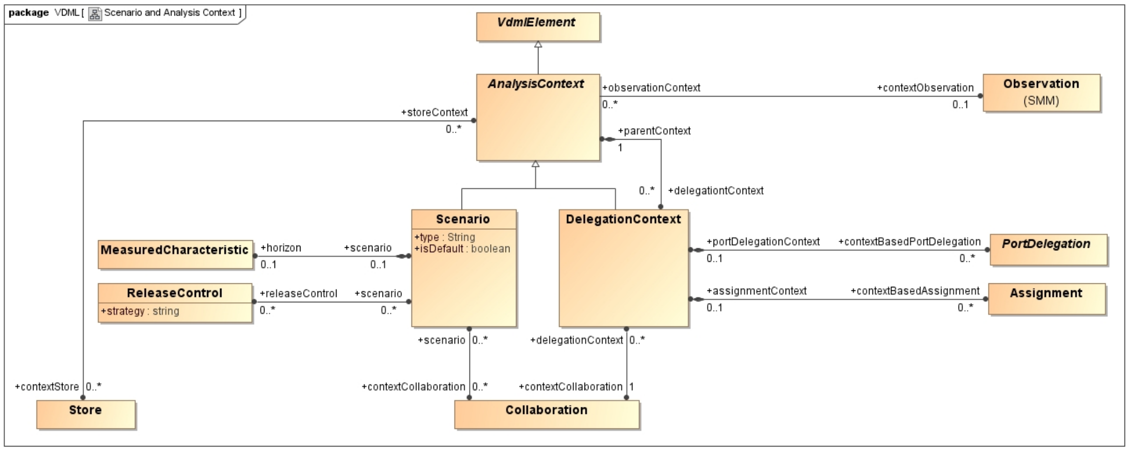
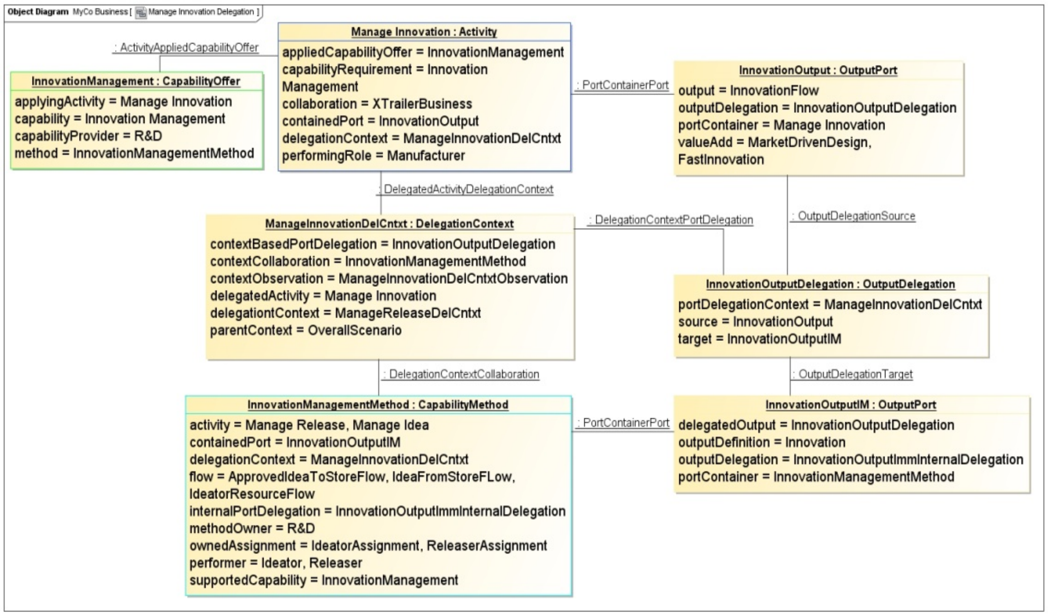


Figure 51. Scenario and Analysis Context metamodel

As it appears from Figure 51, a scenario can also drive the variation of a release control, which defines a strategy that determines, for a capability offer, the priority of work to be performed, relative to other work. This is required for simulation, and will not be discussed further in the context of the XTrailer use case example. Per scenario a horizon can be defined, which may serve as the horizon for simulation. This will neither be discussed further in this document.

In the remainder of this section, discussion will focus on activity networks, activity delegation and related details. All of these are applied in the context of the XTrailer use case example.

We first consider the Manage Innovation activity, as performed by the Manufacturer party role in the business network. The object diagram that defines this activity is shown in Figure 52.


Object Diagram MyCo Business [Manage Innovation Delegation]
: ActivityAppliedCapabilityOffer
: PortContainerPort
: DelegatedActivityDelegationContext
: DelegationContextPortDelegation
: OutputDelegationSource
: DelegationContextCollaboration
: OutputDelegationTarget
: PortContainerPort

Manage Innovation : Activity
appliedCapabilityOffer = InnovationManagement
capabilityRequirement = Innovation Management
collaboration = XTrailerBusiness
containedPort = InnovationOutput
delegationContext = ManageInnovationDelCntxt
performingRole = Manufacturer

InnovationManagement : CapabilityOffer
applyingActivity = Manage Innovation
capability = Innovation Management
capabilityProvider = R&D
method = InnovationManagementMethod

InnovationOutput : OutputPort
output = InnovationFlow
outputDelegation = InnovationOutputDelegation
portContainer = Manage Innovation
valueAdd = MarketDrivenDesign, FastInnovation

ManageInnovationDelCntxt : DelegationContext
contextBasedPortDelegation = InnovationOutputDelegation
contextCollaboration = InnovationManagementMethod
contextObservation = ManageInnovationDelCntxtObservation
delegatedActivity = Manage Innovation
delegationtContext = ManageReleaseDelCntxt
parentContext = OverallScenario

InnovationOutputDelegation : OutputDelegation
portDelegationContext = ManageInnovationDelCntxt
source = InnovationOutput
target = InnovationOutputIM

InnovationManagementMethod : CapabilityMethod
activity = Manage Release, Manage Idea
containedPort = InnovationOutputIM
delegationContext = ManageInnovationDelCntxt
flow = ApprovedIdeaToStoreFlow, IdeaFromStoreFLow, IdeatorResourceFlow
internalPortDelegation = InnovationOutputImmInternalDelegation
methodOwner = R&D
ownedAssignment = IdeatorAssignment, ReleaserAssignment
performer = Ideator, Releaser
supportedCapability = InnovationManagement

InnovationOutputIM : OutputPort
delegatedOutput = InnovationOutputDelegation
outputDefinition = Innovation
outputDelegation = InnovationOutputImmInternalDelegation
portContainer = InnovationManagementMethod

Figure 52. Delegation to Innovation Management method (objects)

The activity object in Figure 52 shows, that the Manage Innovation activity has "Innovation Management" as capability requirement. In accordance to that requirement, the Innovation Management capability offer of R&D as the capability providing organization unit, is selected as the applied capability offer (the top-left object in Figure 52). The activity delegates its work to the Innovation Management method (indicated by the large vertical object shape at the bottom of the diagram), through the "Manage Innovation Delegation Context" delegation context. That delegation context contains the delegation of the "Innovation Output" output port of the activity to the corresponding output port of the capability method. As was already indicated earlier, a capability method, as any collaboration, is a "port container", which can contain ports. This is exemplified here, by the Innovation Management method, containing an output port. This can be considered a port on the "boundary" of the method. It allows "black boxing" of the method to its using contexts. A (parent) activity just delegates to that method, thereby delegating its ports to corresponding method ports, without having to know the details of the activity configuration inside the method. As will be discussed later, this will also make value delivery models suitable as basis to drive definition of more IT or automation-related artifacts such as web-service interfaces or user interfaces.

The delegation context object in the object diagram in Figure 52 is an instance of the Delegation Context class in the Scenario and Analysis Context diagram in Figure 51. The contained output port delegation object is an instance of the Output Delegation class in the Port Delegations meta-model diagram in Figure 53.

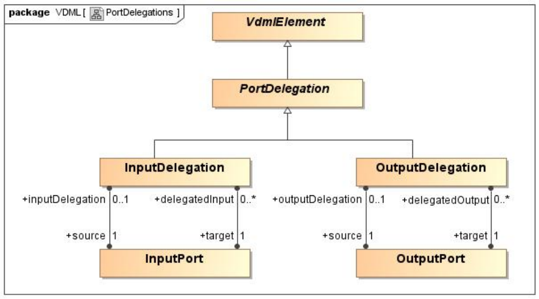


Figure 53. Port Delegations metamodel

The Activity object, including its various links, is an instance of the Activity class in the Activities metamodel diagram in Figure 54.

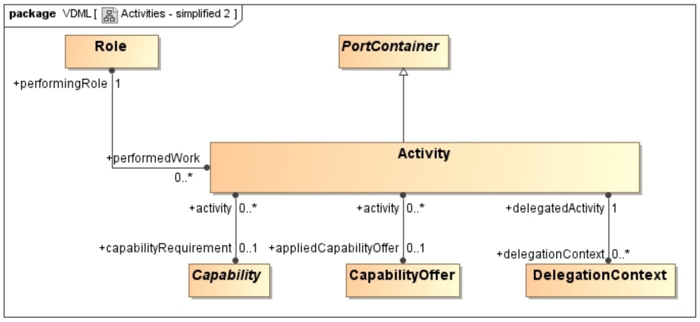


Figure 54. Activities metamodel (partial)

The basis for the capability method object, as contained in the object diagram in Figure 52, to connect via a port, is the Capability Methods metamodel, as presented in the metamodel diagram in Figure 55. It defines a Capability Method as a Collaboration, and, as a Collaboration is a Port Container, it can contain ports. Note also that a Capability Method contains performers, which are specialized roles.

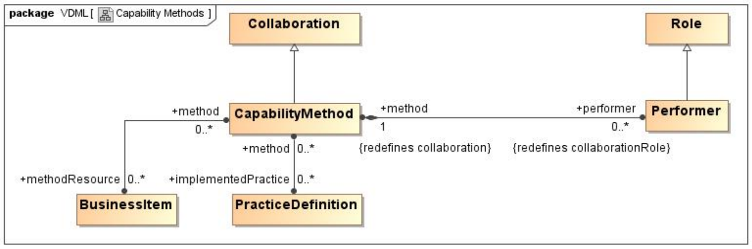


Figure 55. Capability Methods metamodel

Note also that the meta-model diagram in Figure 55 shows some more Capability Method detail, that is not applied in the XTrailer use case example:

- Association of business items, directly with capability methods. This is useful for resources for which storage in stores does not apply, and for which consideration of use or consumption by individual activities is not meaningful, but for which it is still relevant to define or assert that capability methods are supported by such resources. Examples of such resources are patents, or technologies such as Enterprise Resource Planning (ERP) suites or Business Process Management (BPM) technology, such as a “process execution engine”.
- Association to practice definitions in a practice library, indicating which practices are meant to be implemented by a particular capability method.

Note that, so-far a model view mockup of delegation, including its port delegations, has not been suggested. This is because, most likely, the definition of delegation does not involve graphical diagrams, but merely a tabular and table-oriented user interface.

Figure 56 provides a model view mockup of the activity network of the Innovation Management method.

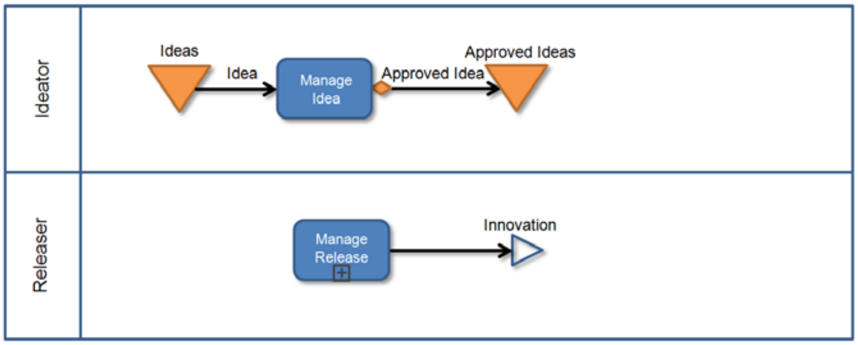


Figure 56. Innovation Management method's activity network mockup

The corresponding object diagram is provided in Figure 57.

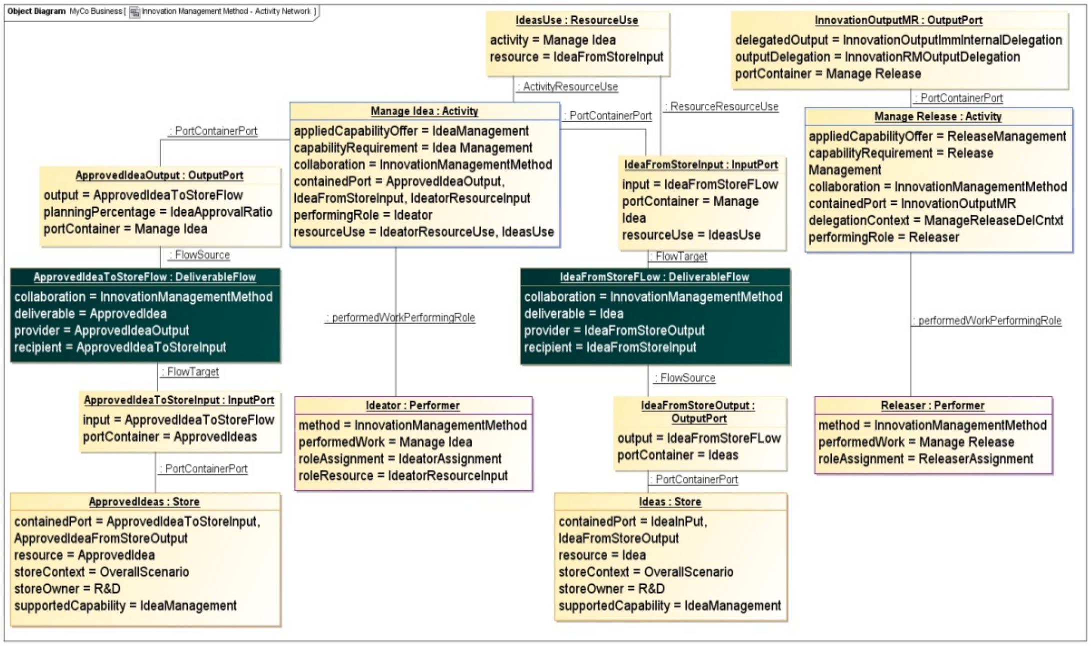

Object Diagram MyCo Business [Innovation Management Method - Activity Network]
IdeasUse : ResourceUse
activity = Manage Idea
resource = IdeaFromStoreInput
InnovationOutputMR : OutputPort
delegatedOutput = InnovationOutputImmInternalDelegation
outputDelegation = InnovationRMOutputDelegation
portContainer = Manage Release
: ActivityResourceUse
: PortContainerPort
: ResourceResourceUse
: PortContainerPort
Manage Idea : Activity
appliedCapabilityOffer = IdeaManagement
capabilityRequirement = Idea Management
collaboration = InnovationManagementMethod
containedPort = ApprovedIdeaOutput, IdeaFromStoreInput, IdeatorResourceInput
performingRole = Ideator
resourceUse = IdeatorResourceUse, IdeasUse
: PortContainerPort
Manage Release : Activity
appliedCapabilityOffer = ReleaseManagement
capabilityRequirement = Release Management
collaboration = InnovationManagementMethod
containedPort = InnovationOutputMR
delegationContext = ManageReleaseDelCntxt
performingRole = Releaser
: PortContainerPort
ApprovedIdeaOutput : OutputPort
output = ApprovedIdeaToStoreFlow
planningPercentage = IdeaApprovalRatio
portContainer = Manage Idea
IdeaFromStoreInput : InputPort
input = IdeaFromStoreFLow
portContainer = Manage Idea
resourceUse = IdeasUse
: FlowSource
: FlowTarget
ApprovedIdeaToStoreFlow : DeliverableFlow
collaboration = InnovationManagementMethod
deliverable = ApprovedIdea
provider = ApprovedIdeaOutput
recipient = ApprovedIdeaToStoreInput
: performedWorkPerformingRole
IdeaFromStoreFLow : DeliverableFlow
collaboration = InnovationManagementMethod
deliverable = Idea
provider = IdeaFromStoreOutput
recipient = IdeaFromStoreInput
: performedWorkPerformingRole
: FlowTarget
: FlowSource
ApprovedIdeaToStoreInput : InputPort
input = ApprovedIdeaToStoreFlow
portContainer = ApprovedIdeas
Ideator : Performer
method = InnovationManagementMethod
performedWork = Manage Idea
roleAssignment = IdeatorAssignment
roleResource = IdeatorResourceInput
IdeaFromStoreOutput : OutputPort
output = IdeaFromStoreFLow
portContainer = Ideas
Releaser : Performer
method = InnovationManagementMethod
performedWork = Manage Release
roleAssignment = ReleaserAssignment
: PortContainerPort
: PortContainerPort
ApprovedIdeas : Store
containedPort = ApprovedIdeaToStoreInput, ApprovedIdeaFromStoreOutput
resource = ApprovedIdea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement
Ideas : Store
containedPort = IdeaInPut, IdeaFromStoreOutput
resource = Idea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement

Figure 57. Innovation Management method's activity network objects

Note the Resource Use object, defined in relation to the "idea from store input" of the Manage idea activity. A resource use might specify how much resource (input) is required, possibly in relation to an output (e.g. "four wheels go into a car"). The "ideas use" object could be used here to e.g. define the average number of ideas that the Manage idea activity takes from the Ideas store. Details of this have not been applied to the use case here. Resource use will be discussed in more detail later, in relation to using resources from pools.

Note also that, though the model view mockup in Figure 56 shows three connectors, there are only two deliverable flow objects instantiated according to Figure 57. The output port "Innovation Output MR" of the Manage release activity is not related to a deliverable flow object. The reason for that is clear from the objects diagram in Figure 58, which defines the further delegation of the output port of the Innovation Management method (its "boundary port") to a port inside the method, namely the "Innovation Output MR" output port of the Manage Release activity.

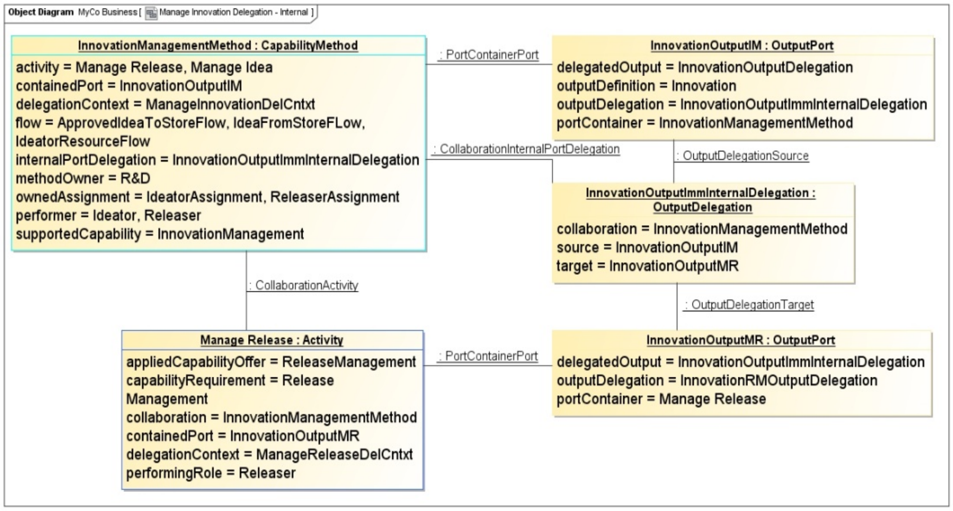

Object Diagram MyCo Business [Manage Innovation Delegation - Internal]
InnovationManagementMethod : CapabilityMethod
activity = Manage Release, Manage Idea
containedPort = InnovationOutputIM
delegationContext = ManageInnovationDelCntxt
flow = ApprovedIdeaToStoreFlow, IdeaFromStoreFLow, IdeatorResourceFlow
internalPortDelegation = InnovationOutputImmInternalDelegation
methodOwner = R&D
ownedAssignment = IdeatorAssignment, ReleaserAssignment
performer = Ideator, Releaser
supportedCapability = InnovationManagement
: PortContainerPort
InnovationOutputIM : OutputPort
delegatedOutput = InnovationOutputDelegation
outputDefinition = Innovation
outputDelegation = InnovationOutputImmInternalDelegation
portContainer = InnovationManagementMethod
: CollaborationInternalPortDelegation
: OutputDelegationSource
InnovationOutputImmInternalDelegation : OutputDelegation
collaboration = InnovationManagementMethod
source = InnovationOutputIM
target = InnovationOutputMR
: CollaborationActivity
: OutputDelegationTarget
Manage Release : Activity
appliedCapabilityOffer = ReleaseManagement
capabilityRequirement = Release Management
collaboration = InnovationManagementMethod
containedPort = InnovationOutputMR
delegationContext = ManageReleaseDelCntxt
performingRole = Releaser
: PortContainerPort
InnovationOutputMR : OutputPort
delegatedOutput = InnovationOutputImmInternalDelegation
outputDelegation = InnovationRMOutputDelegation
portContainer = Manage Release

Figure 58. Innovation Management method’s internal delegation objects

The output delegation object, as associated with the capability method object in Figure 58, is an instance of Port Delegation class, as associated to the Collaboration class, and serving as its “internal port delegation”, in the Collaborations meta-model diagram of Figure 16 above.

In the mockup of the activity network diagram, in Figure 56, “boundary” port shows as a small “bottom-left pyramid” shape. The port delegation, from the that port to the activity port is visualized as a connector, similar to the other flow connectors, though its meaning is different: the “actual” flow is in the parent collaboration, being the business network collaboration, the network of which was given in Figure 43. Actually the port shape in Figure 56 serves as a “placeholder” for the actual deliverable flow in Figure 43.

Note that, when a capability method would be delegation target of an activity, and in the context of that delegation, there are no port delegations to one or more capability method ports (“boundary” ports), the connectors that represent port delegations from these “boundary” ports to ports of activities inside the capability method, would not be shown in the context of that delegation.

The Approved Idea flow in the activity network view mockup in Figure 56 shows as “conditioned”, the corresponding output port being explicitly visible and showing as a small diamond. This might express either one or both of the following situations:

- A condition on a port, here an output port, defined as expression, associated with the port. Port Containers. The Port Containers metamodel diagram, as presented earlier in Figure 45, defines how a port can be conditioned. The Expressions metamodel diagram is presented in Figure 59. Expressions are essential to simulation. They are also useful when capability methods are used as basis for generation of process definitions. Expressions are not applied in the XTrailer use case example and will not be discussed further.
- A planning percentage on a port, which is especially useful in “manual” analysis based on value delivery models, where average measurements are applied. The objects diagram in Figure 60 shows partial application of this in the context of the use case. The Port Containers metamodel diagram in Figure 45 defines how a port can have a planning percentage.

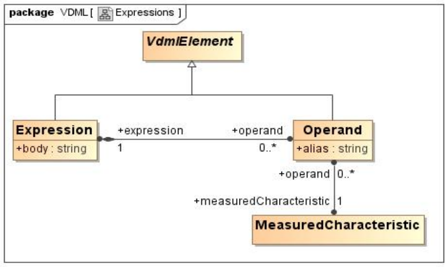
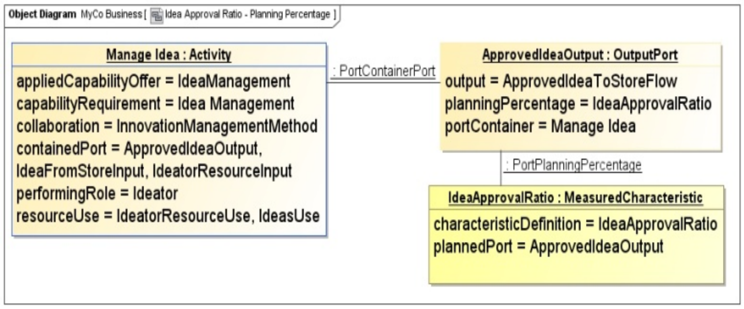


Figure 59. Expressions metamodel

Only activity ports can be conditioned or have a planning percentage. This is not obvious from the meta-model, but is expressed as constrained over the meta-model, in the VDML 1.0 specification. According to the object diagram in Figure 60, the “Approved Idea Output” output port of the Manage idea activity has a planning percentage. This is visualized in the activity network diagram mockup in Figure 56 via the small diamond on the corresponding deliverable flow connector.


Object Diagram MyCo Business [Idea Approval Ratio - Planning Percentage]
: PortContainerPort
: PortPlanningPercentage

Manage Idea : Activity

appliedCapabilityOffer = IdeaManagement
capabilityRequirement = Idea Management
collaboration = InnovationManagementMethod
containedPort = ApprovedIdeaOutput, IdeaFromStoreInput, IdeatorResourceInput
performingRole = Ideator
resourceUse = IdeatorResourceUse, IdeasUse

ApprovedIdeaOutput : OutputPort

output = ApprovedIdeaToStoreFlow
planningPercentage = IdeaApprovalRatio
portContainer = Manage Idea

IdeaApprovalRatio : MeasuredCharacteristic

characteristicDefinition = IdeaApprovalRatio
plannedPort = ApprovedIdeaOutput

Figure 60. Planning percentage objects

This planning percentage has a functional meaning. It is a measured characteristic that defines the idea approval ratio of the Manage Idea activity. The activity might, for example, approve 80 % of ideas on average. This is the first example of a measured characteristic that we encounter in the discussion of the use case so-far.

As the VDML elements metamodel diagram in Figure 61, being the most “core” and “technical” part of the VDML metamodel, shows, a measured characteristic can be defined by association with a characteristic. Characteristics, as well as associated measures, are defined in measure libraries, based on the SMM metamodel (see SMM (2012)). A characteristic is defined “*a distinguishing feature or quality that can be qualified or quantified by applying a measure*”.

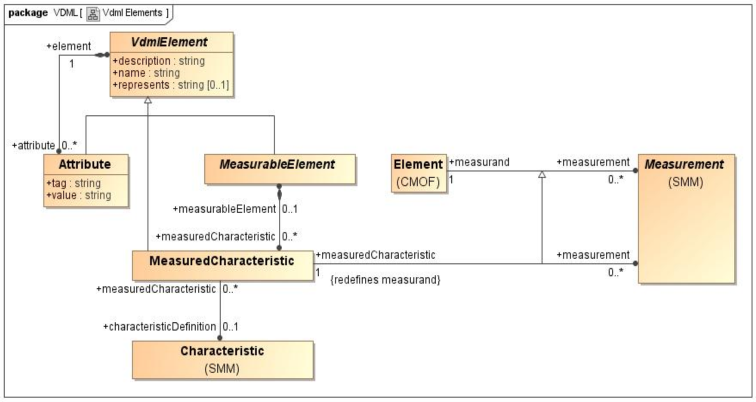


Figure 61. VDML Elements metamodel

The measured characteristic object in the objects diagram in Figure 60, is an instance of the Measured Characteristic class in the metamodel diagram in Figure 61. Its characteristic definition “Idea Approval Ratio” is an instance of the associated Characteristic class. This characteristic is stored in a measure library. The reader might refer to SMM (2012) for the measure library metamodel. The object diagram of the measure library that we use in the XTrailer use case example, as far as modeling characteristics is concerned, is provided in Figure 62. We will discuss the use of several more of these characteristics, together with its associated measures, in 3.1.5.

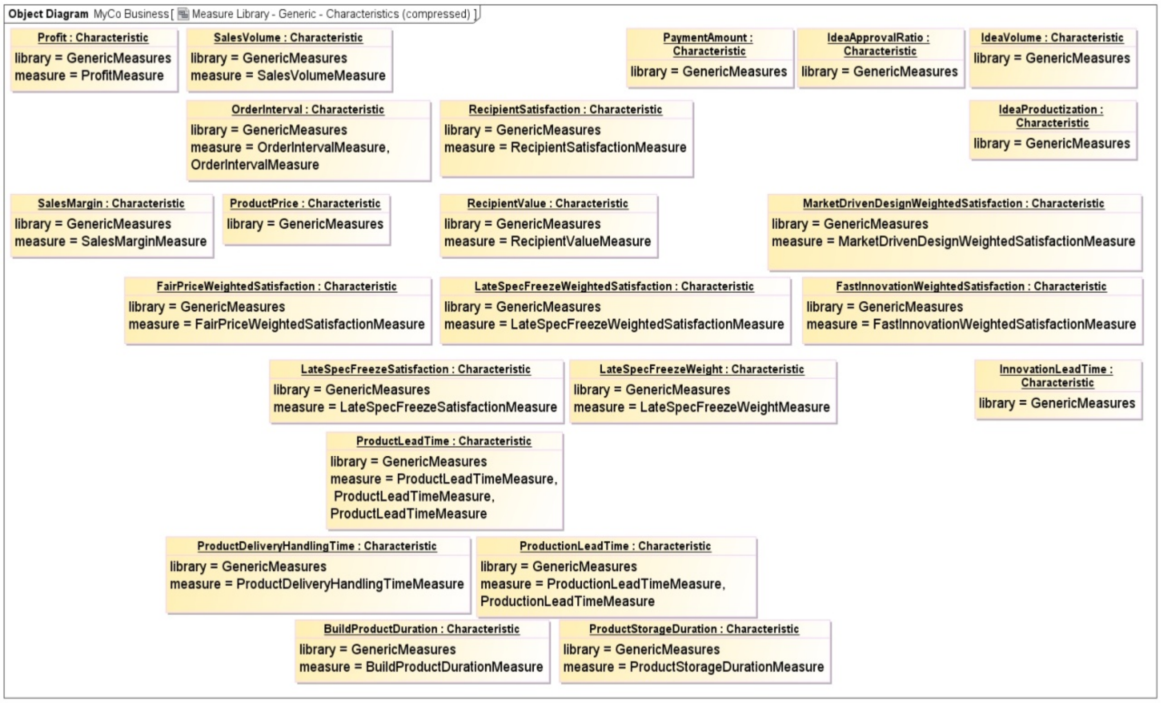

Object Diagram MyCo Business[Measure Library - Generic - Characteristics (compressed)]
Profit : Characteristic
library = GenericMeasures
measure = ProfitMeasure
SalesVolume : Characteristic
library = GenericMeasures
measure = SalesVolumeMeasure
PaymentAmount : Characteristic
library = GenericMeasures
IdeaApprovalRatio : Characteristic
library = GenericMeasures
IdeaVolume : Characteristic
library = GenericMeasures
OrderInterval : Characteristic
library = GenericMeasures
measure = OrderIntervalMeasure, OrderIntervalMeasure
RecipientSatisfaction : Characteristic
library = GenericMeasures
measure = RecipientSatisfactionMeasure
IdeaProductization : Characteristic
library = GenericMeasures
SalesMargin : Characteristic
library = GenericMeasures
measure = SalesMarginMeasure
ProductPrice : Characteristic
library = GenericMeasures
RecipientValue : Characteristic
library = GenericMeasures
measure = RecipientValueMeasure
MarketDrivenDesignWeightedSatisfaction : Characteristic
library = GenericMeasures
measure = MarketDrivenDesignWeightedSatisfactionMeasure
FairPriceWeightedSatisfaction : Characteristic
library = GenericMeasures
measure = FairPriceWeightedSatisfactionMeasure
LateSpecFreezeWeightedSatisfaction : Characteristic
library = GenericMeasures
measure = LateSpecFreezeWeightedSatisfactionMeasure
FastInnovationWeightedSatisfaction : Characteristic
library = GenericMeasures
measure = FastInnovationWeightedSatisfactionMeasure
LateSpecFreezeSatisfaction : Characteristic
library = GenericMeasures
measure = LateSpecFreezeSatisfactionMeasure
LateSpecFreezeWeight : Characteristic
library = GenericMeasures
measure = LateSpecFreezeWeightMeasure
InnovationLeadTime : Characteristic
library = GenericMeasures
ProductLeadTime : Characteristic
library = GenericMeasures
measure = ProductLeadTimeMeasure, ProductLeadTimeMeasure, ProductLeadTimeMeasure
ProductDeliveryHandlingTime : Characteristic
library = GenericMeasures
measure = ProductDeliveryHandlingTimeMeasure
ProductionLeadTime : Characteristic
library = GenericMeasures
measure = ProductionLeadTimeMeasure, ProductionLeadTimeMeasure
BuildProductDuration : Characteristic
library = GenericMeasures
measure = BuildProductDurationMeasure
ProductStorageDuration : Characteristic
library = GenericMeasures
measure = ProductStorageDurationMeasure

Figure 62. Measure Library objects (Characteristics)

As the VDML Elements metamodel in Figure 61 shows, a measured characteristic can be associated zero or more measurements, Measurement being an SMM metamodel class as well. VDML enforces, per metamodel constraint, that when N measurements are associated with the same measured characteristic, these measurements will be contained in N different observations, each observation containing just one measurement for the measured characteristic. This is important to support analysis context as discussed earlier: An analysis context can have a context-specific measurement for a measured characteristic (of e.g. a store or collaboration or part of it), as contained in its observation. Observation is an SMM meta-model class as well. Analysis Context is a VDML meta-model class. The metamodel diagrams of VDML Elements (in Figure 61) and Analysis Context and Scenario (in Figure 51) define the integration between VDML and SMM.

The application of characteristics, measures, measurements and observations, to the XTrailer use case example, will be discussed in detail in 3.1.5.

As is clear from the Port Containers meta-model diagram in Figure 45, planning percentage is a metamodel-defined association to the Measured Characteristic class. VDML defines several more associations to Measured Characteristic, each with its particular semantics. Some of them, in the area of value and value proposition, will be discussed in 3.1.5.

As the VDML Elements metamodel diagram in Figure 61 shows, any measurable element can have zero or more measured characteristics associated. These are the measured characteristics that a business analyst can define, as custom model elements. As the Port Containers metamodel diagram in Figure 45 shows, both ports and port containers are measurable elements, and so can have custom defined measured characteristics. In 3.1.5 various examples of custom measured characteristics will be discussed, in the context of the XTrailer use case example.

As the mockup of the activity network diagram of the Innovation Management method in Figure 56 indicates, the Manage Release activity, shown via a rectangle shape containing an “expansion button” (small box with “+” sign), is delegating its work to another collaboration. The object diagram in Figure 63 shows the objects that define this delegation. The Manage Release activity sets a delegation context, to delegate its work to the Release Management method that supports the Release Management capability offer of R&D. In the context of that delegation, the Innovation output is delegated further “down”, from the Manage Release activity to the Release Management method (the output port on its “boundary”).

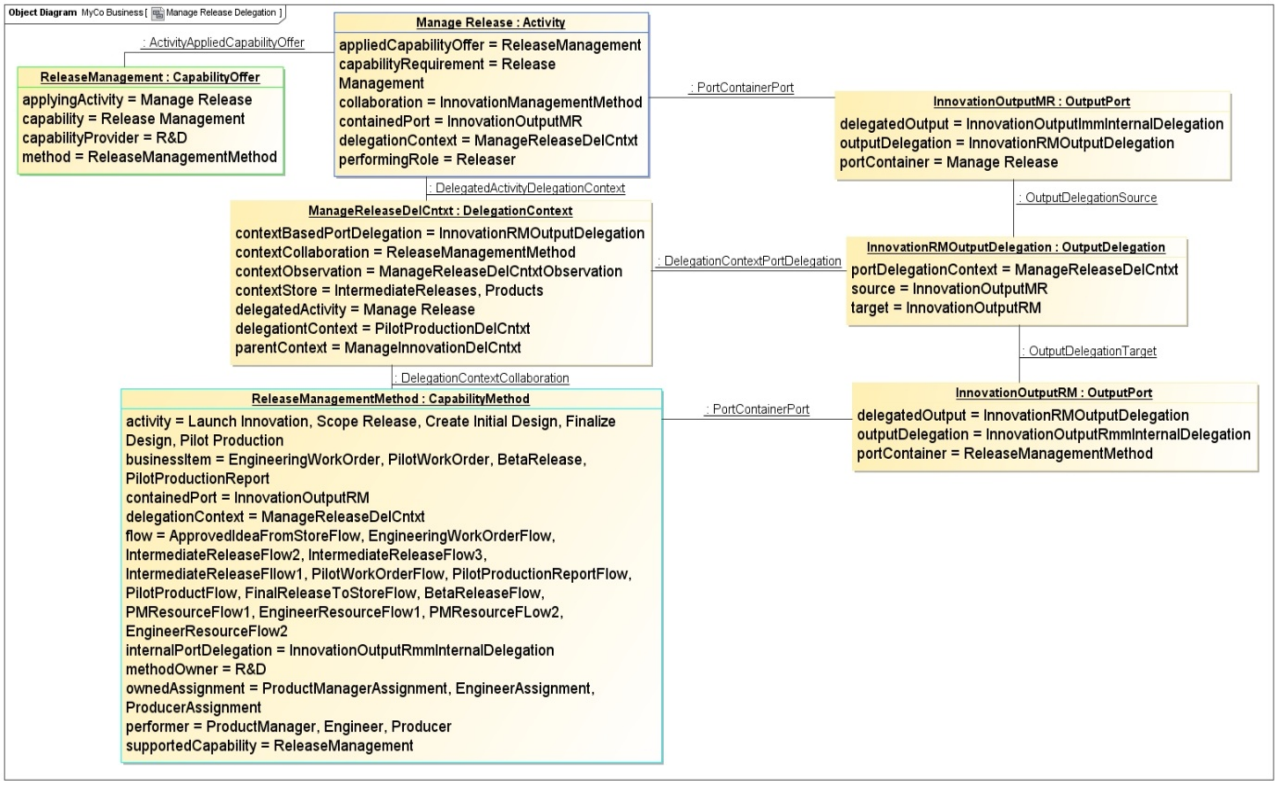

Object Diagram MyCo Business [Manage Release Delegation]
Manage Release : Activity
appliedCapabilityOffer = ReleaseManagement
capabilityRequirement = Release Management
collaboration = InnovationManagementMethod
containedPort = InnovationOutputMR
delegationContext = ManageReleaseDelCntxt
performingRole = Releaser
: ActivityAppliedCapabilityOffer
ReleaseManagement : CapabilityOffer
applyingActivity = Manage Release
capability = Release Management
capabilityProvider = R&D
method = ReleaseManagementMethod
: PortContainerPort
InnovationOutputMR : OutputPort
delegatedOutput = InnovationOutputImmInternalDelegation
outputDelegation = InnovationRMOutputDelegation
portContainer = Manage Release
: DelegatedActivityDelegationContext
: OutputDelegationSource
ManageReleaseDelCntxt : DelegationContext
contextBasedPortDelegation = InnovationRMOutputDelegation
contextCollaboration = ReleaseManagementMethod
contextObservation = ManageReleaseDelCntxtObservation
contextStore = IntermediateReleases, Products
delegatedActivity = Manage Release
delegationtContext = PilotProductionDelCntxt
parentContext = ManageInnovationDelCntxt
: DelegationContextPortDelegation
InnovationRMOutputDelegation : OutputDelegation
portDelegationContext = ManageReleaseDelCntxt
source = InnovationOutputMR
target = InnovationOutputRM
: OutputDelegationTarget
: DelegationContextCollaboration
ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement
: PortContainerPort
InnovationOutputRM : OutputPort
delegatedOutput = InnovationRMOutputDelegation
outputDelegation = InnovationOutputRmmInternalDelegation
portContainer = ReleaseManagementMethod

Figure 63. Delegation to Release Management method (objects)

According to the activity network diagram mockup in Figure 56, showing the activity network of the Innovation Management method, and according to its corresponding object diagram in Figure 57, the Manage Release activity is performed by the Releaser, being a performer role. As the Manage Release activity delegates its work to the Release Management method of R&D, it is logical to assign the Releaser role to the R&D organization unit. This assignment is defined in the object diagram in Figure 64.

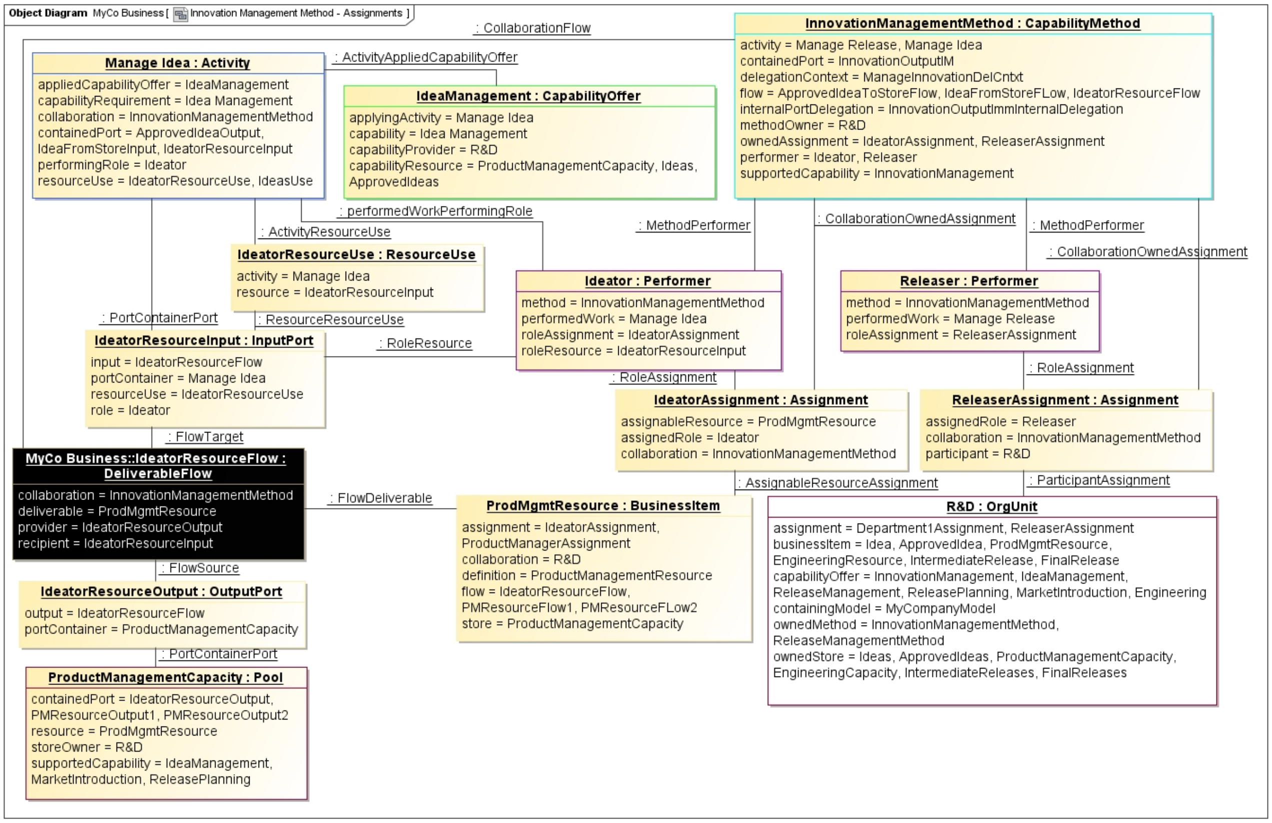

Object Diagram MyCo Business [Innovation Management Method - Assignments]
: CollaborationFlow
InnovationManagementMethod : CapabilityMethod
activity = Manage Release, Manage Idea
containedPort = InnovationOutputIM
delegationContext = ManageInnovationDelCntxt
flow = ApprovedIdeaToStoreFlow, IdeaFromStoreFLow, IdeatorResourceFlow
internalPortDelegation = InnovationOutputImmInternalDelegation
methodOwner = R&D
ownedAssignment = IdeatorAssignment, ReleaserAssignment
performer = Ideator, Releaser
supportedCapability = InnovationManagement
Manage Idea : Activity
appliedCapabilityOffer = IdeaManagement
capabilityRequirement = Idea Management
collaboration = InnovationManagementMethod
containedPort = ApprovedIdeaOutput,
IdeaFromStoreInput, IdeatorResourceInput
performingRole = Ideator
resourceUse = IdeatorResourceUse, IdeasUse
: ActivityAppliedCapabilityOffer
IdeaManagement : CapabilityOffer
applyingActivity = Manage Idea
capability = Idea Management
capabilityProvider = R&D
capabilityResource = ProductManagementCapacity, Ideas,
ApprovedIdeas
: performedWorkPerformingRole
: ActivityResourceUse
: MethodPerformer
: CollaborationOwnedAssignment
: MethodPerformer
: CollaborationOwnedAssignment
IdeatorResourceUse : ResourceUse
activity = Manage Idea
resource = IdeatorResourceInput
Ideator : Performer
method = InnovationManagementMethod
performedWork = Manage Idea
roleAssignment = IdeatorAssignment
roleResource = IdeatorResourceInput
Releaser : Performer
method = InnovationManagementMethod
performedWork = Manage Release
roleAssignment = ReleaserAssignment
: PortContainerPort
: ResourceResourceUse
IdeatorResourceInput : InputPort
input = IdeatorResourceFlow
portContainer = Manage Idea
resourceUse = IdeatorResourceUse
role = Ideator
: RoleResource
: RoleAssignment
: RoleAssignment
IdeatorAssignment : Assignment
assignableResource = ProdMgmtResource
assignedRole = Ideator
collaboration = InnovationManagementMethod
ReleaserAssignment : Assignment
assignedRole = Releaser
collaboration = InnovationManagementMethod
participant = R&D
: FlowTarget
MyCo Business::IdeatorResourceFlow : DeliverableFlow
collaboration = InnovationManagementMethod
deliverable = ProdMgmtResource
provider = IdeatorResourceOutput
recipient = IdeatorResourceInput
: FlowDeliverable
: AssignableResourceAssignment
: ParticipantAssignment
ProdMgmtResource : BusinessItem
assignment = IdeatorAssignment,
ProductManagerAssignment
collaboration = R&D
definition = ProductManagementResource
flow = IdeatorResourceFlow,
PMResourceFlow1, PMResourceFLow2
store = ProductManagementCapacity
R&D : OrgUnit
assignment = Department1Assignment, ReleaserAssignment
businessItem = Idea, ApprovedIdea, ProdMgmtResource,
EngineeringResource, IntermediateRelease, FinalRelease
capabilityOffer = InnovationManagement, IdeaManagement,
ReleaseManagement, ReleasePlanning, MarketIntroduction, Engineering
containingModel = MyCompanyModel
ownedMethod = InnovationManagementMethod,
ReleaseManagementMethod
ownedStore = Ideas, ApprovedIdeas, ProductManagementCapacity,
EngineeringCapacity, IntermediateReleases, FinalReleases
: FlowSource
IdeatorResourceOutput : OutputPort
output = IdeatorResourceFlow
portContainer = ProductManagementCapacity
: PortContainerPort
ProductManagementCapacity : Pool
containedPort = IdeatorResourceOutput,
PMResourceOutput1, PMResourceOutput2
resource = ProdMgmtResource
storeOwner = R&D
supportedCapability = IdeaManagement,
MarketIntroduction, ReleasePlanning

Figure 64. Innovation Management method's role assignment objects

The Ideator role performs the Manage Idea activity. This activity requires an Idea Management capability, and based on that, the Idea Management capability offer of R&D is applied. No capability method is defined to support it. As shown in the capability management diagram mockup of R&D in Figure 33 above, the Idea Management capability offer is supported by the Product Management Capacity pool. The object diagram in Figure 64 shows how capacity from that pool is assigned to the Ideator role:

- Product management resource, defined as business item, flows from the product management capacity pool, via the Ideator resource flow, to the Manage Idea activity.
- The corresponding input port of the activity is associated with the Ideator performer role, indicator that the resource is meant as role resource.
- The activity also defines a resource use (Ideator Resource Use) in relation to the input port. This resource use can be used to, for instance, specify the resource quantity required (which is not specified in the use case example).
- The Innovation Management method contains an assignment (Ideator Assignment) to assign the Ideator role to the product management resource.

Note that the delivery flow of role resource from the pool to the Manage Idea activity is not visualized in the activity network mockup in Figure 56 As it is expected that such flows will clutter activity network diagrams too much, we assume that resource flows from and to pools are, by default, not shown in these diagrams. It would be reasonable that a user option would be supported to show them where and when wanted.

Activity and activity-related objects in the object diagram in Figure 64 are instances of several classes in the Activities metamodel diagram in Figure 65. Note that not all properties have been instantiated in the use case example.

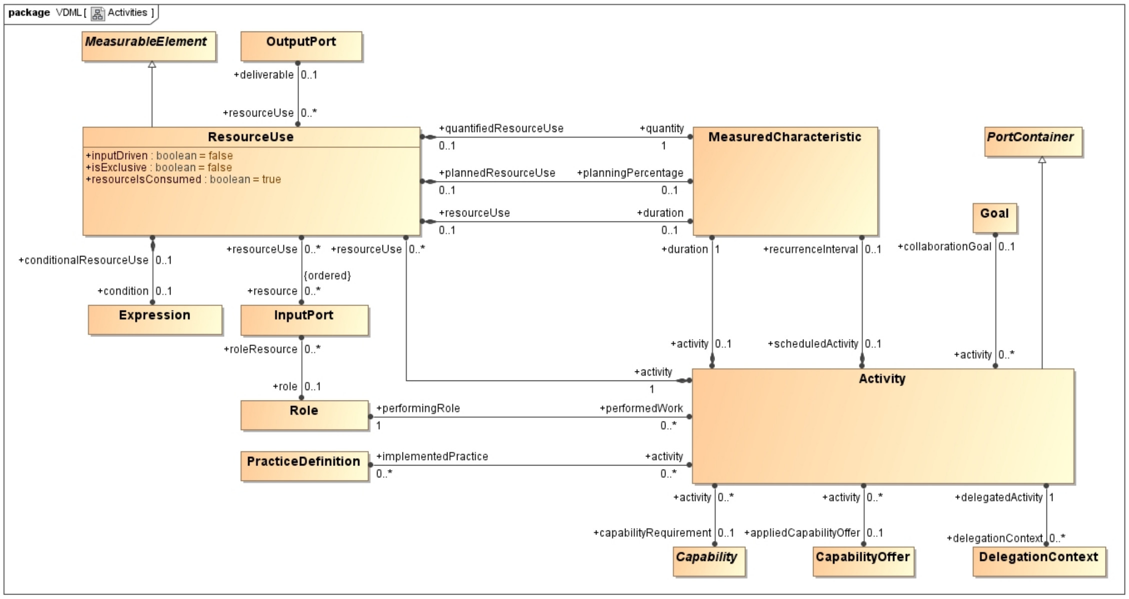


Figure 65. Activities metamodel

Assignment and assignment-related objects in the object diagram in Figure 64 are instances of some of the classes in the Assignments metamodel diagram in Figure 66.

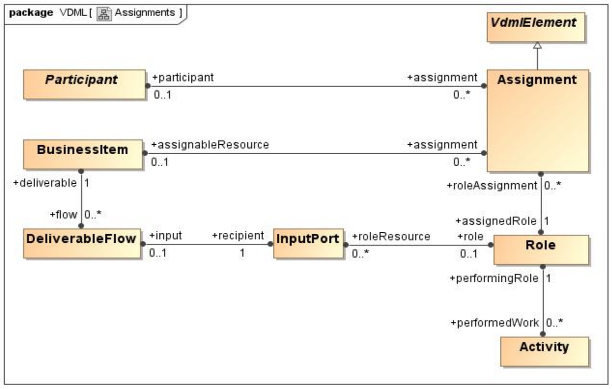


Figure 66. Assignments metamodel

Figure 67 contains the activity network diagram mockup of the Release Management method.

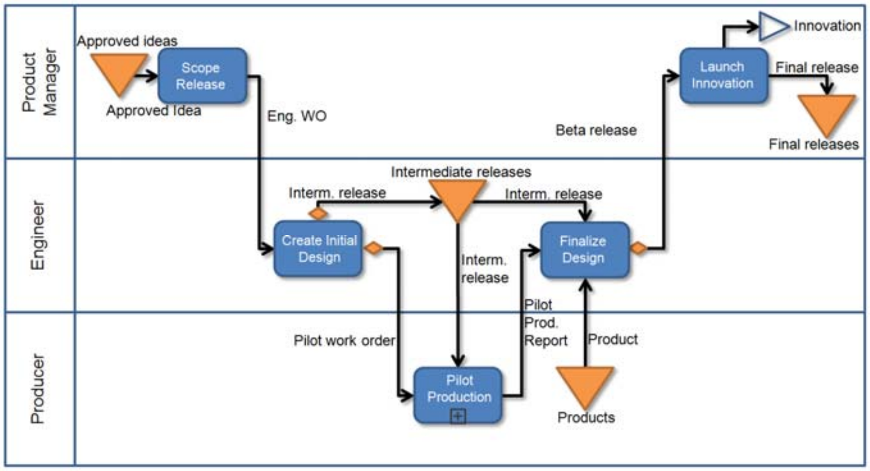


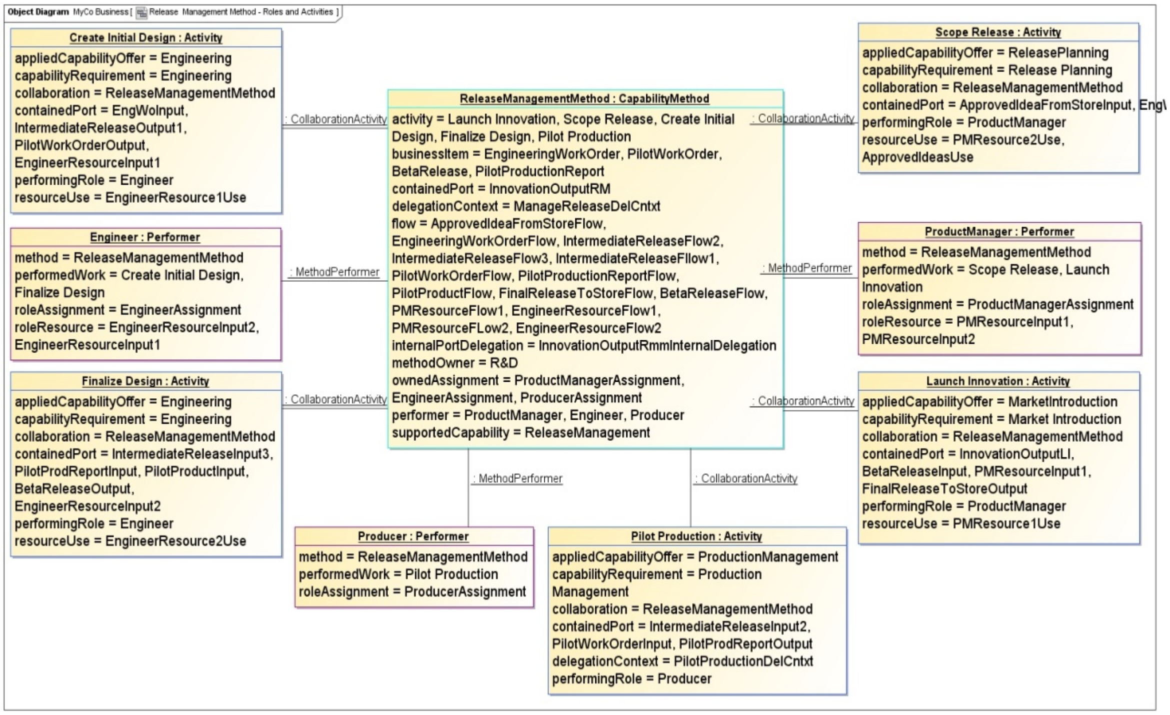
Figure 67. Release Management method's activity network mockup

Note that this capability method, as well as other methods that we consider in the use case, will contain much more detail in a real-world situation. For sake of simplicity and keeping the use case content minimal, we only model over-simplified parts of the business system. For example, in relation to the activity network in Figure 67, we assume the following:

- Any approved idea goes into a next release; modeling alternatives is omitted (e.g. to model removing of aged ideas).
- When a design fails, it fails permanently and completely; modeling of rework and re-use of successful parts of a design is omitted

Such assumptions do not lead to real-world-complete models, but we do not want to extend the content of the use case beyond what is needed to explain the modeling constructs of VDML, their semantics, and ways to expose them to business users.

The object diagram in Figure 68 shows performer roles and activities of the Release Management method. Delivery flow objects detail, as exposed in the mockup in Figure 67, will be shown on subsequent diagrams, one per each performer role, as the number of objects that represent the method's activity network are too many to show in a single diagram.



Create Initial Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = EngWoInput, IntermediateReleaseOutput1, PilotWorkOrderOutput, EngineerResourceInput1
performingRole = Engineer
resourceUse = EngineerResource1Use

Engineer : Performer
method = ReleaseManagementMethod
performedWork = Create Initial Design, Finalize Design
roleAssignment = EngineerAssignment
roleResource = EngineerResourceInput2, EngineerResourceInput1

Finalize Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput3, PilotProdReportInput, PilotProductInput, BetaReleaseOutput, EngineerResourceInput2
performingRole = Engineer
resourceUse = EngineerResource2Use

ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement

Producer : Performer
method = ReleaseManagementMethod
performedWork = Pilot Production
roleAssignment = ProducerAssignment

Pilot Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput2, PilotWorkOrderInput, PilotProdReportOutput
delegationContext = PilotProductionDelCntxt
performingRole = Producer

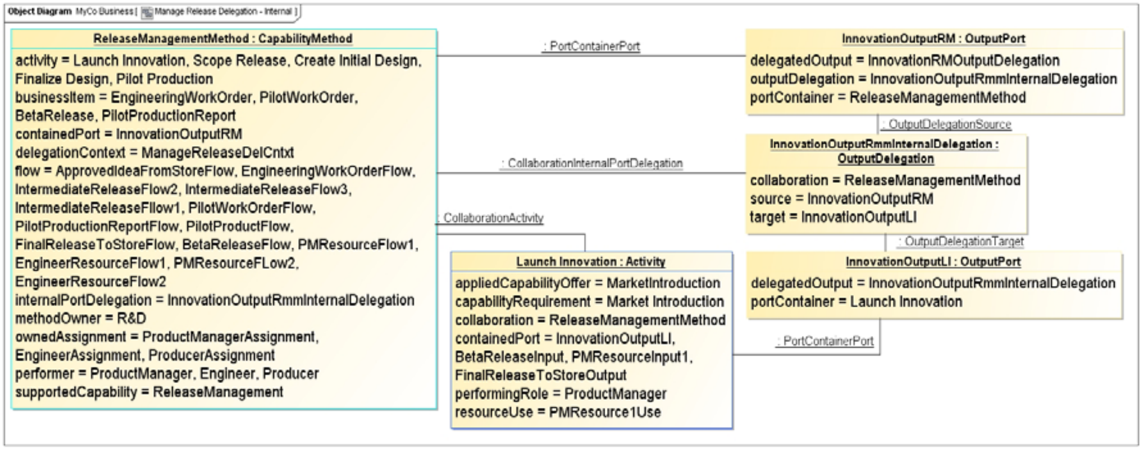
Scope Release : Activity
appliedCapabilityOffer = ReleasePlanning
capabilityRequirement = Release Planning
collaboration = ReleaseManagementMethod
containedPort = ApprovedIdeaFromStoreInput, Eng\
performingRole = ProductManager
resourceUse = PMResource2Use, ApprovedIdeasUse

ProductManager : Performer
method = ReleaseManagementMethod
performedWork = Scope Release, Launch Innovation
roleAssignment = ProductManagerAssignment
roleResource = PMResourceInput1, PMResourceInput2

Launch Innovation : Activity
appliedCapabilityOffer = MarketIntroduction
capabilityRequirement = Market Introduction
collaboration = ReleaseManagementMethod
containedPort = InnovationOutputLI, BetaReleaseInput, PMResourceInput1, FinalReleaseToStoreOutput
performingRole = ProductManager
resourceUse = PMResource1Use

Figure 68. Release Management method object's (partial)

The object diagram in Figure 69 contains the objects that define how the Innovation output on the "boundary" of the Release Management method is delegated to an activity output port inside the method. This is similar to what has been discussed earlier in relation to the Innovation Management method.



ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement

InnovationOutputRM : OutputPort
delegatedOutput = InnovationRMOutputDelegation
outputDelegation = InnovationOutputRmmInternalDelegation
portContainer = ReleaseManagementMethod

InnovationOutputRmmInternalDelegation : OutputDelegation
collaboration = ReleaseManagementMethod
source = InnovationOutputRM
target = InnovationOutputLI

Launch Innovation : Activity
appliedCapabilityOffer = MarketIntroduction
capabilityRequirement = Market Introduction
collaboration = ReleaseManagementMethod
containedPort = InnovationOutputLI, BetaReleaseInput, PMResourceInput1, FinalReleaseToStoreOutput
performingRole = ProductManager
resourceUse = PMResource1Use

InnovationOutputLI : OutputPort
delegatedOutput = InnovationOutputRmmInternalDelegation
portContainer = Launch Innovation

Figure 69. Release Management method's internal delegation objects

Objects behind the activity network diagram of the Release Management method in Figure 67, are contained in the object diagrams of Figure 70 (for the Engineer performer role), Figure 71 (for the Producer performer role) and Figure 72 (for the Product Manager performer role). These diagrams show examples of modeling constructs that have been discussed earlier.

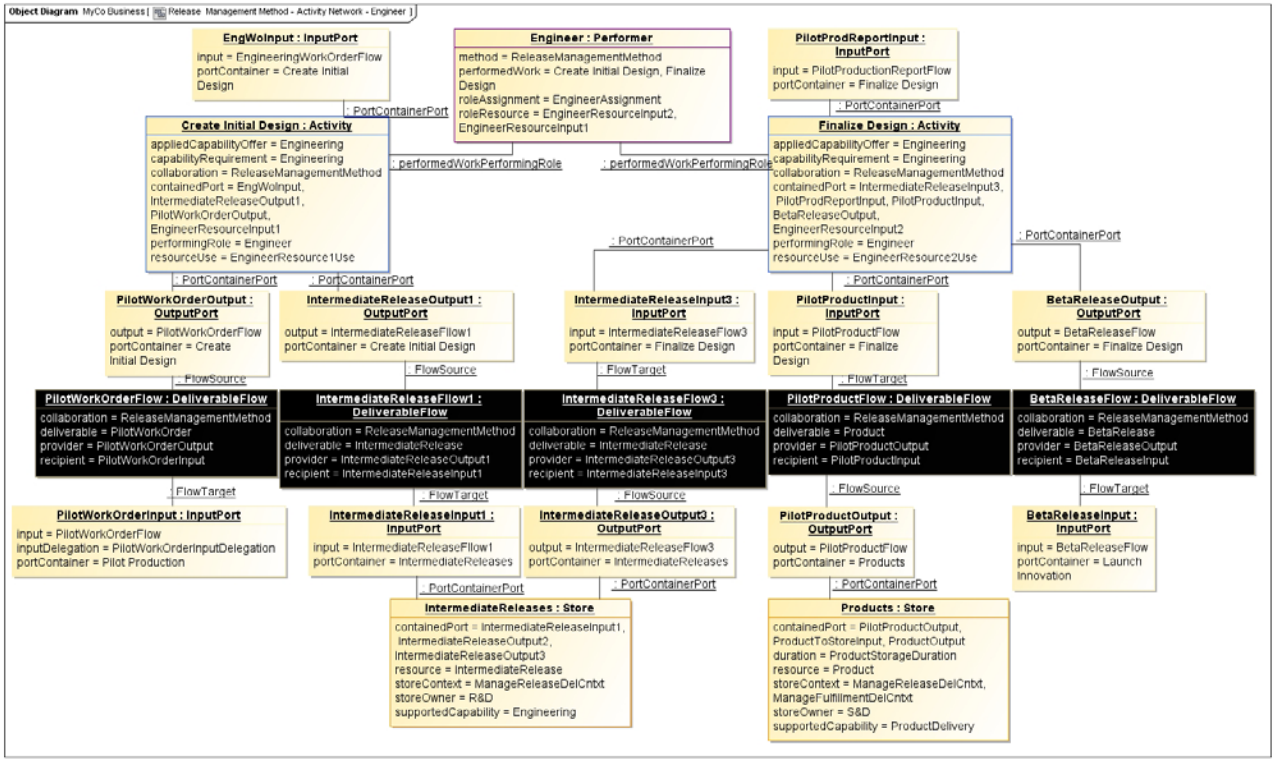

Object Diagram MyCo Business [Release Management Method - Activity Network - Engineer]
EngWoInput : InputPort
input = EngineeringWorkOrderFlow
portContainer = Create Initial Design
Engineer : Performer
method = ReleaseManagementMethod
performedWork = Create Initial Design, Finalize Design
roleAssignment = EngineerAssignment
roleResource = EngineerResourceInput2, EngineerResourceInput1
PilotProdReportInput : InputPort
input = PilotProductionReportFlow
portContainer = Finalize Design
: PortContainerPort
Create Initial Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = EngWoInput, IntermediateReleaseOutput1, PilotWorkOrderOutput, EngineerResourceInput1
performingRole = Engineer
resourceUse = EngineerResource1Use
: performedWorkPerformingRole
: performedWorkPerformingRole
: PortContainerPort
Finalize Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput3, PilotProdReportInput, PilotProductInput, BetaReleaseOutput, EngineerResourceInput2
performingRole = Engineer
resourceUse = EngineerResource2Use
: PortContainerPort
: PortContainerPort
: PortContainerPort
: PortContainerPort
: PortContainerPort
PilotWorkOrderOutput : OutputPort
output = PilotWorkOrderFlow
portContainer = Create Initial Design
IntermediateReleaseOutput1 : OutputPort
output = IntermediateReleaseFlow1
portContainer = Create Initial Design
IntermediateReleaseInput3 : InputPort
input = IntermediateReleaseFlow3
portContainer = Finalize Design
PilotProductInput : InputPort
input = PilotProductFlow
portContainer = Finalize Design
BetaReleaseOutput : OutputPort
output = BetaReleaseFlow
portContainer = Finalize Design
: FlowSource
: FlowSource
: FlowTarget
: FlowTarget
: FlowSource
PilotWorkOrderFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = PilotWorkOrder
provider = PilotWorkOrderOutput
recipient = PilotWorkOrderInput
IntermediateReleaseFllow1 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = IntermediateRelease
provider = IntermediateReleaseOutput1
recipient = IntermediateReleaseInput1
IntermediateReleaseFlow3 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = IntermediateRelease
provider = IntermediateReleaseOutput3
recipient = IntermediateReleaseInput3
PilotProductFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = Product
provider = PilotProductOutput
recipient = PilotProductInput
BetaReleaseFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = BetaRelease
provider = BetaReleaseOutput
recipient = BetaReleaseInput
: FlowTarget
: FlowTarget
: FlowSource
: FlowSource
: FlowTarget
PilotWorkOrderInput : InputPort
input = PilotWorkOrderFlow
inputDelegation = PilotWorkOrderInputDelegation
portContainer = Pilot Production
IntermediateReleaseInput1 : InputPort
input = IntermediateReleaseFllow1
portContainer = IntermediateReleases
IntermediateReleaseOutput3 : OutputPort
output = IntermediateReleaseFlow3
portContainer = IntermediateReleases
PilotProductOutput : OutputPort
output = PilotProductFlow
portContainer = Products
BetaReleaseInput : InputPort
input = BetaReleaseFlow
portContainer = Launch Innovation
: PortContainerPort
: PortContainerPort
: PortContainerPort
IntermediateReleases : Store
containedPort = IntermediateReleaseInput1, IntermediateReleaseOutput2, IntermediateReleaseOutput3
resource = IntermediateRelease
storeContext = ManageReleaseDelCntxt
storeOwner = R&D
supportedCapability = Engineering
Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery

Figure 70. Release Management method’s activity network objects (for Engineer performer role)

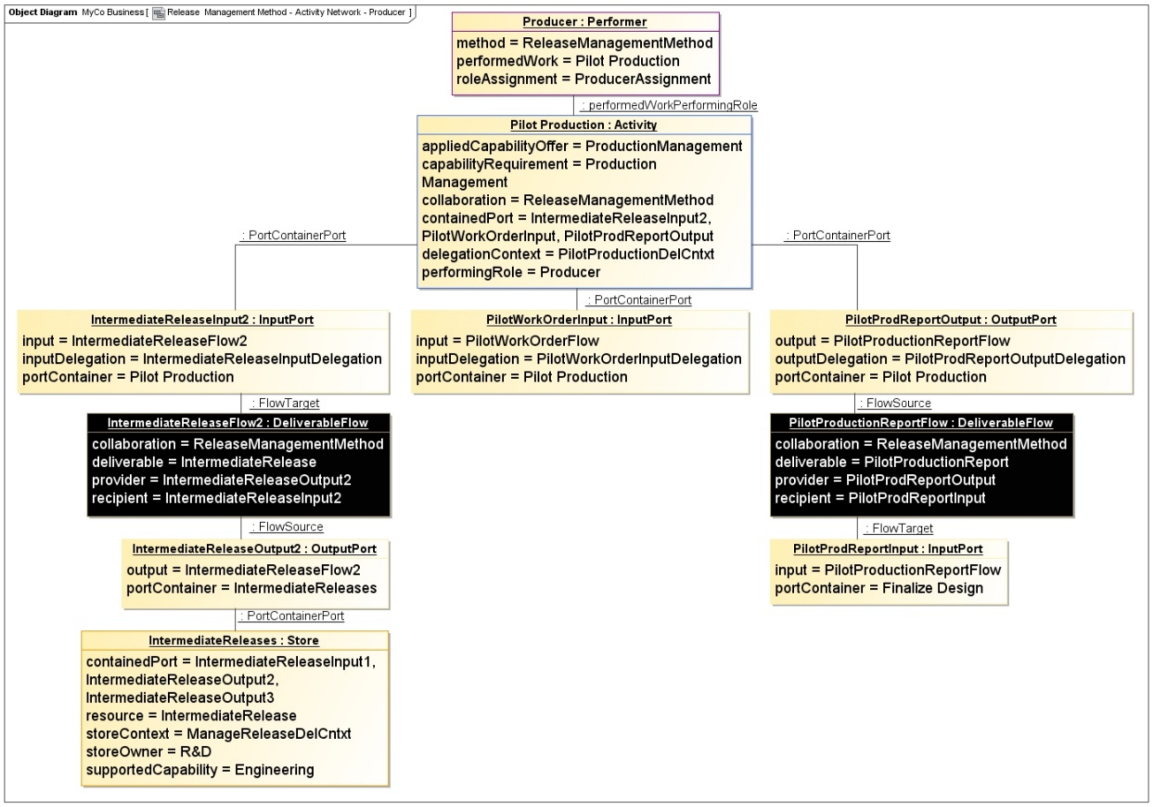

Object Diagram MyCo Business [Release Management Method - Activity Network - Producer]
Producer : Performer
method = ReleaseManagementMethod
performedWork = Pilot Production
roleAssignment = ProducerAssignment
: performedWorkPerformingRole
Pilot Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput2, PilotWorkOrderInput, PilotProdReportOutput
delegationContext = PilotProductionDelCntxt
performingRole = Producer
: PortContainerPort
: PortContainerPort
: PortContainerPort
IntermediateReleaseInput2 : InputPort
input = IntermediateReleaseFlow2
inputDelegation = IntermediateReleaseInputDelegation
portContainer = Pilot Production
PilotWorkOrderInput : InputPort
input = PilotWorkOrderFlow
inputDelegation = PilotWorkOrderInputDelegation
portContainer = Pilot Production
PilotProdReportOutput : OutputPort
output = PilotProductionReportFlow
outputDelegation = PilotProdReportOutputDelegation
portContainer = Pilot Production
: FlowTarget
: FlowSource
IntermediateReleaseFlow2 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = IntermediateRelease
provider = IntermediateReleaseOutput2
recipient = IntermediateReleaseInput2
PilotProductionReportFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = PilotProductionReport
provider = PilotProdReportOutput
recipient = PilotProdReportInput
: FlowSource
: FlowTarget
IntermediateReleaseOutput2 : OutputPort
output = IntermediateReleaseFlow2
portContainer = IntermediateReleases
PilotProdReportInput : InputPort
input = PilotProductionReportFlow
portContainer = Finalize Design
: PortContainerPort
IntermediateReleases : Store
containedPort = IntermediateReleaseInput1, IntermediateReleaseOutput2, IntermediateReleaseOutput3
resource = IntermediateRelease
storeContext = ManageReleaseDelCntxt
storeOwner = R&D
supportedCapability = Engineering

Figure 71. Release Management method’s activity network objects (for Producer performer role)

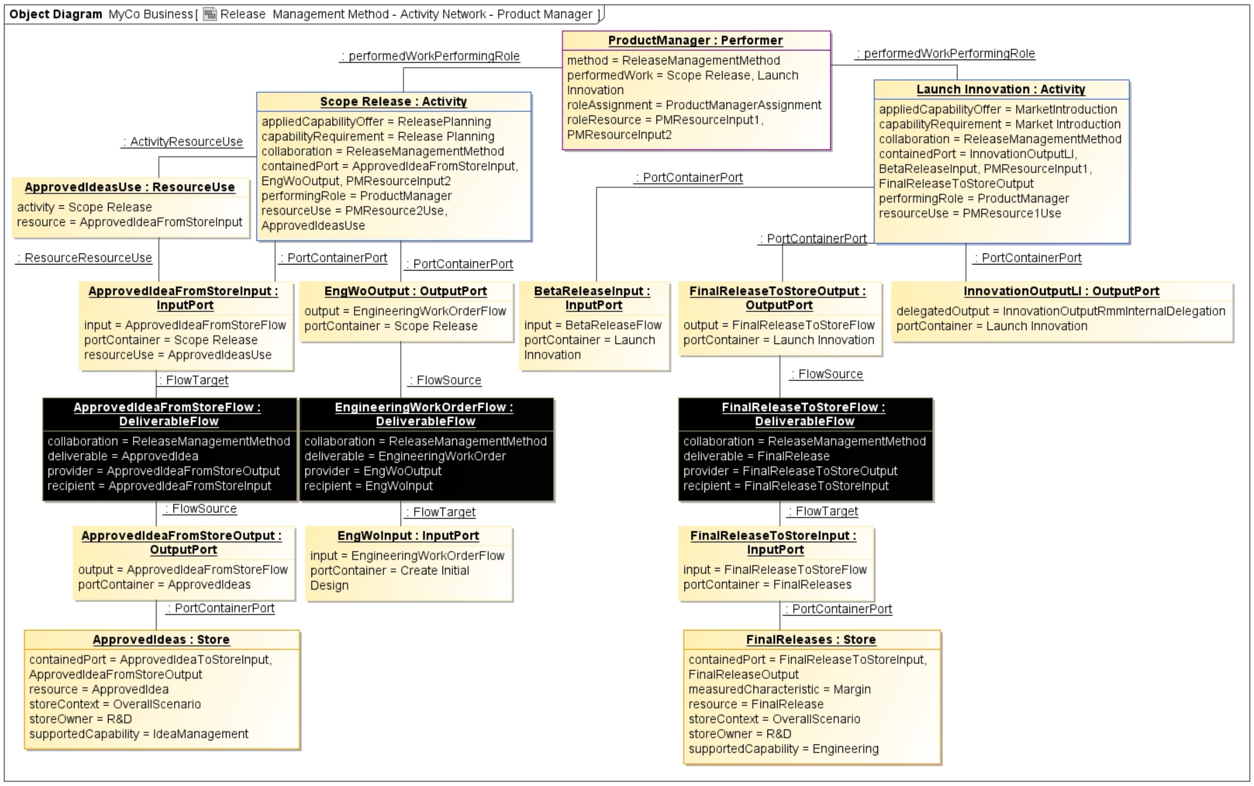

Object Diagram MyCo Business[Release Management Method - Activity Network - Product Manager]
ProductManager : Performer
method = ReleaseManagementMethod
performedWork = Scope Release, Launch Innovation
roleAssignment = ProductManagerAssignment
roleResource = PMResourceInput1, PMResourceInput2
: performedWorkPerformingRole
: performedWorkPerformingRole
Scope Release : Activity
appliedCapabilityOffer = ReleasePlanning
capabilityRequirement = Release Planning
collaboration = ReleaseManagementMethod
containedPort = ApprovedIdeaFromStoreInput, EngWoOutput, PMResourceInput2
performingRole = ProductManager
resourceUse = PMResource2Use, ApprovedIdeasUse
Launch Innovation : Activity
appliedCapabilityOffer = MarketIntroduction
capabilityRequirement = Market Introduction
collaboration = ReleaseManagementMethod
containedPort = InnovationOutputLI, BetaReleaseInput, PMResourceInput1, FinalReleaseToStoreOutput
performingRole = ProductManager
resourceUse = PMResource1Use
: ActivityResourceUse
ApprovedIdeasUse : ResourceUse
activity = Scope Release
resource = ApprovedIdeaFromStoreInput
: ResourceResourceUse
: PortContainerPort
: PortContainerPort
: PortContainerPort
: PortContainerPort
: PortContainerPort
ApprovedIdeaFromStoreInput : InputPort
input = ApprovedIdeaFromStoreFlow
portContainer = Scope Release
resourceUse = ApprovedIdeasUse
EngWoOutput : OutputPort
output = EngineeringWorkOrderFlow
portContainer = Scope Release
BetaReleaseInput : InputPort
input = BetaReleaseFlow
portContainer = Launch Innovation
FinalReleaseToStoreOutput : OutputPort
output = FinalReleaseToStoreFlow
portContainer = Launch Innovation
InnovationOutputLI : OutputPort
delegatedOutput = InnovationOutputRmmInternalDelegation
portContainer = Launch Innovation
: FlowTarget
: FlowSource
: FlowSource
ApprovedIdeaFromStoreFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = ApprovedIdea
provider = ApprovedIdeaFromStoreOutput
recipient = ApprovedIdeaFromStoreInput
EngineeringWorkOrderFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = EngineeringWorkOrder
provider = EngWoOutput
recipient = EngWoInput
FinalReleaseToStoreFlow : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = FinalRelease
provider = FinalReleaseToStoreOutput
recipient = FinalReleaseToStoreInput
: FlowSource
: FlowTarget
: FlowTarget
ApprovedIdeaFromStoreOutput : OutputPort
output = ApprovedIdeaFromStoreFlow
portContainer = ApprovedIdeas
EngWoInput : InputPort
input = EngineeringWorkOrderFlow
portContainer = Create Initial Design
FinalReleaseToStoreInput : InputPort
input = FinalReleaseToStoreFlow
portContainer = FinalReleases
: PortContainerPort
: PortContainerPort
ApprovedIdeas : Store
containedPort = ApprovedIdeaToStoreInput, ApprovedIdeaFromStoreOutput
resource = ApprovedIdea
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = IdeaManagement
FinalReleases : Store
containedPort = FinalReleaseToStoreInput, FinalReleaseOutput
measuredCharacteristic = Margin
resource = FinalRelease
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = Engineering

Figure 72. Release Management method's activity network objects (for Product Manager performer role)

Note in Figure 72 that no deliverable flow has been modeled to convey the Innovation output of the Launch Innovation activity. This corresponds with the small "bottom-left pyramid" shape, representing a "boundary" port, in the activity network mockup in Figure 67. And this is because the corresponding output port is delegated from the output port on the "boundary" of the Release Management method, which is itself delegated from the output port of the Manage Release activity in the Innovation Management method. The Innovation output in the business network itself is delegated, in multiple steps, all the way down to the Innovation output of the Launch Innovation activity in the Release Management method. The business item that denotes the Innovation going to the market, as well as the deliverable flow conveying it to the market, are only defined in the business network. From there it is just delegated to ports that do not connect to a deliverable flow themselves.

As the mockup of the activity network of the Release Management method in Figure 67 suggests, the Pilot Production activity, performed by the Producer, delegates its work to another collaboration. The object diagram in Figure 73 shows that it delegates to the Production Management capability method. As discussed earlier (see e.g. Figure 39 and Figure 40) this method supports the Production Management capability offer of the Trailer Plant. This capability offer is applied, given the capability requirement of the Pilot Production activity (it requires a Product Management capability). Figure 73 also shows delegation of two input ports and one output port to the corresponding ports on the "boundary" of the Production Management method.

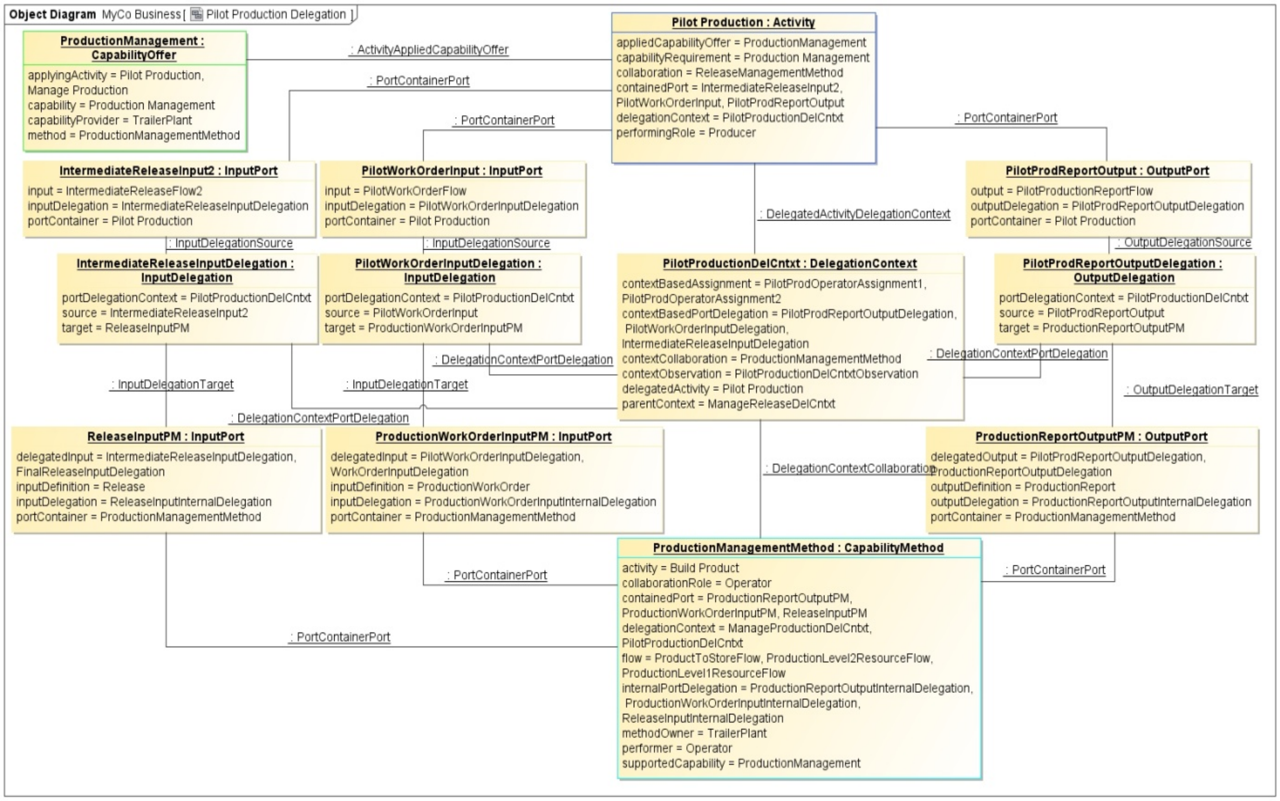

Object Diagram MyCo Business [Pilot Production Delegation]
ProductionManagement : CapabilityOffer
applyingActivity = Pilot Production, Manage Production
capability = Production Management
capabilityProvider = TrailerPlant
method = ProductionManagementMethod
: ActivityAppliedCapabilityOffer
: PortContainerPort
Pilot Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput2, PilotWorkOrderInput, PilotProdReportOutput
delegationContext = PilotProductionDelCntxt
performingRole = Producer
IntermediateReleaseInput2 : InputPort
input = IntermediateReleaseFlow2
inputDelegation = IntermediateReleaseInputDelegation
portContainer = Pilot Production
PilotWorkOrderInput : InputPort
input = PilotWorkOrderFlow
inputDelegation = PilotWorkOrderInputDelegation
portContainer = Pilot Production
PilotProdReportOutput : OutputPort
output = PilotProductionReportFlow
outputDelegation = PilotProdReportOutputDelegation
portContainer = Pilot Production
: DelegatedActivityDelegationContext
: InputDelegationSource
: OutputDelegationSource
IntermediateReleaseInputDelegation : InputDelegation
portDelegationContext = PilotProductionDelCntxt
source = IntermediateReleaseInput2
target = ReleaseInputPM
PilotWorkOrderInputDelegation : InputDelegation
portDelegationContext = PilotProductionDelCntxt
source = PilotWorkOrderInput
target = ProductionWorkOrderInputPM
PilotProductionDelCntxt : DelegationContext
contextBasedAssignment = PilotProdOperatorAssignment1, PilotProdOperatorAssignment2
contextBasedPortDelegation = PilotProdReportOutputDelegation, PilotWorkOrderInputDelegation, IntermediateReleaseInputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = PilotProductionDelCntxtObservation
delegatedActivity = Pilot Production
parentContext = ManageReleaseDelCntxt
PilotProdReportOutputDelegation : OutputDelegation
portDelegationContext = PilotProductionDelCntxt
source = PilotProdReportOutput
target = ProductionReportOutputPM
: DelegationContextPortDelegation
: InputDelegationTarget
: OutputDelegationTarget
ReleaseInputPM : InputPort
delegatedInput = IntermediateReleaseInputDelegation, FinalReleaseInputDelegation
inputDefinition = Release
inputDelegation = ReleaseInputInternalDelegation
portContainer = ProductionManagementMethod
ProductionWorkOrderInputPM : InputPort
delegatedInput = PilotWorkOrderInputDelegation, WorkOrderInputDelegation
inputDefinition = ProductionWorkOrder
inputDelegation = ProductionWorkOrderInputInternalDelegation
portContainer = ProductionManagementMethod
: DelegationContextCollaboration
ProductionReportOutputPM : OutputPort
delegatedOutput = PilotProdReportOutputDelegation, ProductionReportOutputDelegation
outputDefinition = ProductionReport
outputDelegation = ProductionReportOutputInternalDelegation
portContainer = ProductionManagementMethod
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement

Figure 73. Delegation to Production Management method, from Pilot Production (objects)

Given that the Producer role in the Release Management method is performing the Pilot Production activity, and given that this activity delegates its work to the Production Management method of the Trailer Plant, it is obvious that the Trailer Plant is assigned the Producer role. The objects that defined this assignment are contained in the object diagram in Figure 74.

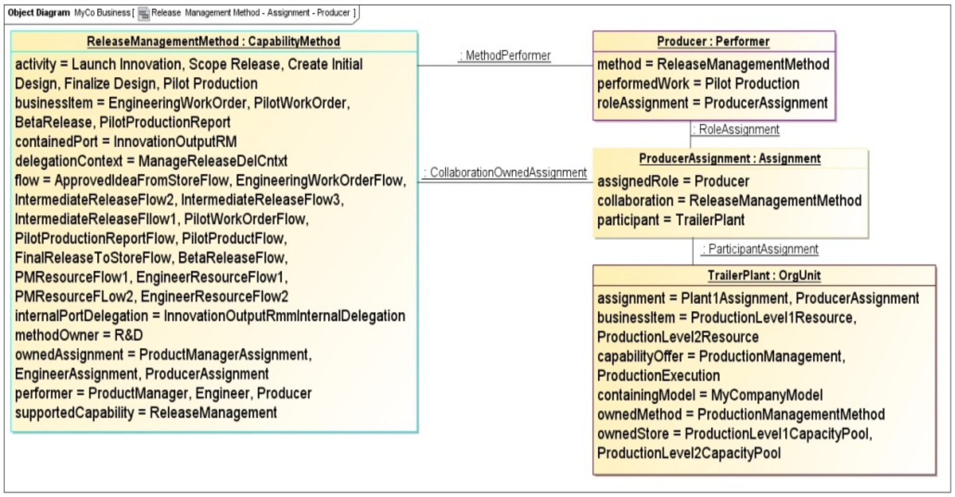

Object Diagram MyCo Business [Release Management Method - Assignment - Producer]
ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement
: MethodPerformer
: CollaborationOwnedAssignment
Producer : Performer
method = ReleaseManagementMethod
performedWork = Pilot Production
roleAssignment = ProducerAssignment
: RoleAssignment
ProducerAssignment : Assignment
assignedRole = Producer
collaboration = ReleaseManagementMethod
participant = TrailerPlant
: ParticipantAssignment
TrailerPlant : OrgUnit
assignment = Plant1Assignment, ProducerAssignment
businessItem = ProductionLevel1Resource, ProductionLevel2Resource
capabilityOffer = ProductionManagement, ProductionExecution
containingModel = MyCompanyModel
ownedMethod = ProductionManagementMethod
ownedStore = ProductionLevel1CapacityPool, ProductionLevel2CapacityPool

Figure 74. Release Management method's Producer role assignment objects

The objects in the object diagram in Figure 75 define assignment of Engineering resource, from the Engineering Capacity pool, to the Engineer performer role in the Release Management method. The metamodel concepts to support this have been discussed earlier, when we discussed about assignment of product management resource to the Ideator role in the Innovation Management method.

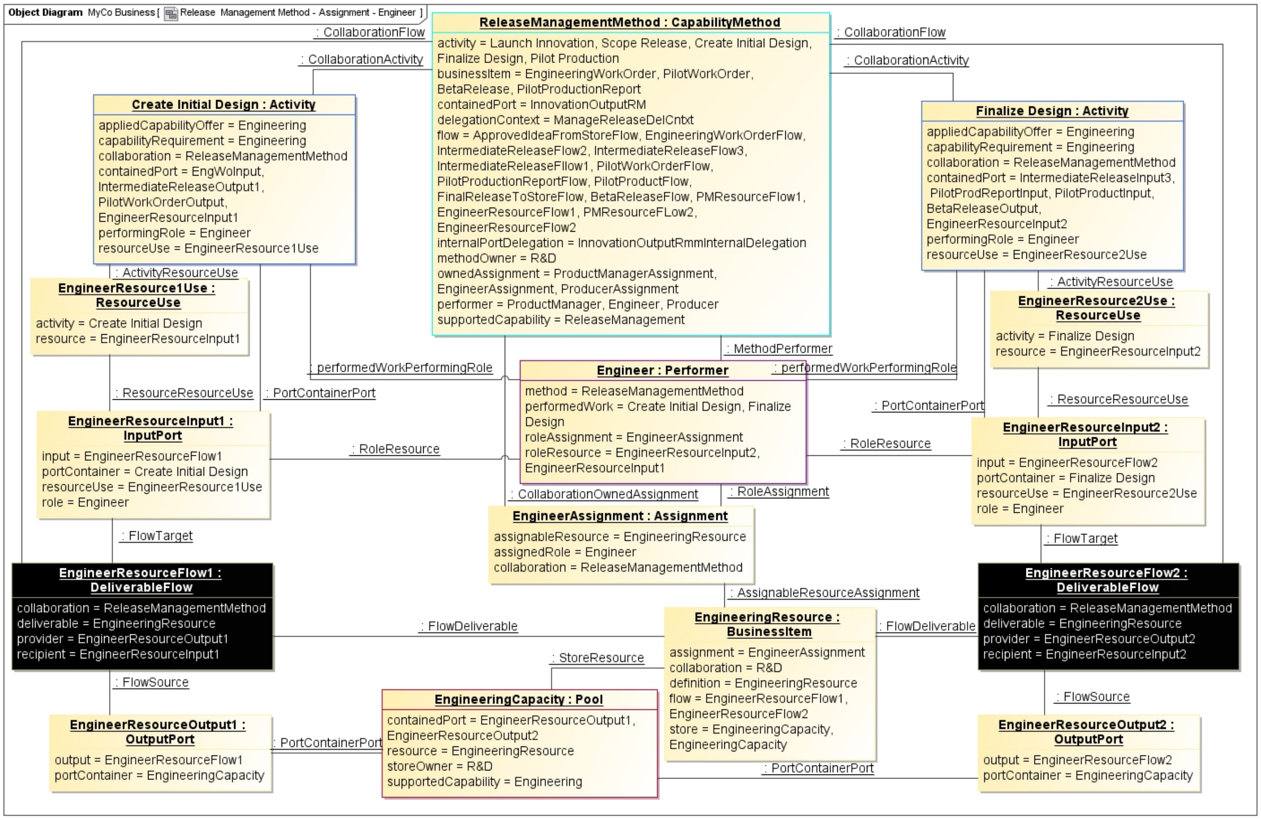

Object Diagram MyCo Business [Release Management Method - Assignment - Engineer]
ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement
: CollaborationFlow
: CollaborationActivity
: CollaborationFlow
: CollaborationActivity
Create Initial Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = EngWoInput, IntermediateReleaseOutput1, PilotWorkOrderOutput, EngineerResourceInput1
performingRole = Engineer
resourceUse = EngineerResource1Use
Finalize Design : Activity
appliedCapabilityOffer = Engineering
capabilityRequirement = Engineering
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput3, PilotProdReportInput, PilotProductInput, BetaReleaseOutput, EngineerResourceInput2
performingRole = Engineer
resourceUse = EngineerResource2Use
: ActivityResourceUse
EngineerResource1Use : ResourceUse
activity = Create Initial Design
resource = EngineerResourceInput1
: ActivityResourceUse
EngineerResource2Use : ResourceUse
activity = Finalize Design
resource = EngineerResourceInput2
: MethodPerformer
: performedWorkPerformingRole
: performedWorkPerformingRole
Engineer : Performer
method = ReleaseManagementMethod
performedWork = Create Initial Design, Finalize Design
roleAssignment = EngineerAssignment
roleResource = EngineerResourceInput2, EngineerResourceInput1
: ResourceResourceUse
: PortContainerPort
: PortContainerPort
: ResourceResourceUse
EngineerResourceInput1 : InputPort
input = EngineerResourceFlow1
portContainer = Create Initial Design
resourceUse = EngineerResource1Use
role = Engineer
: RoleResource
: RoleResource
EngineerResourceInput2 : InputPort
input = EngineerResourceFlow2
portContainer = Finalize Design
resourceUse = EngineerResource2Use
role = Engineer
: CollaborationOwnedAssignment
: RoleAssignment
EngineerAssignment : Assignment
assignableResource = EngineeringResource
assignedRole = Engineer
collaboration = ReleaseManagementMethod
: FlowTarget
: FlowTarget
EngineerResourceFlow1 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = EngineeringResource
provider = EngineerResourceOutput1
recipient = EngineerResourceInput1
: AssignableResourceAssignment
EngineerResourceFlow2 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = EngineeringResource
provider = EngineerResourceOutput2
recipient = EngineerResourceInput2
: FlowDeliverable
: FlowDeliverable
EngineeringResource : BusinessItem
assignment = EngineerAssignment
collaboration = R&D
definition = EngineeringResource
flow = EngineerResourceFlow1, EngineerResourceFlow2
store = EngineeringCapacity, EngineeringCapacity
: StoreResource
: FlowSource
: FlowSource
EngineeringCapacity : Pool
containedPort = EngineerResourceOutput1, EngineerResourceOutput2
resource = EngineeringResource
storeOwner = R&D
supportedCapability = Engineering
EngineerResourceOutput1 : OutputPort
output = EngineerResourceFlow1
portContainer = EngineeringCapacity
: PortContainerPort
: PortContainerPort
EngineerResourceOutput2 : OutputPort
output = EngineerResourceFlow2
portContainer = EngineeringCapacity

Figure 75. Release Management method's Engineer role assignment objects

In a similar way, product management resource, from the Product Management Capacity pool of R&D is assigned to the Product Manager role in the Release Management method. This is defined by the objects in the object diagram in Figure 76.

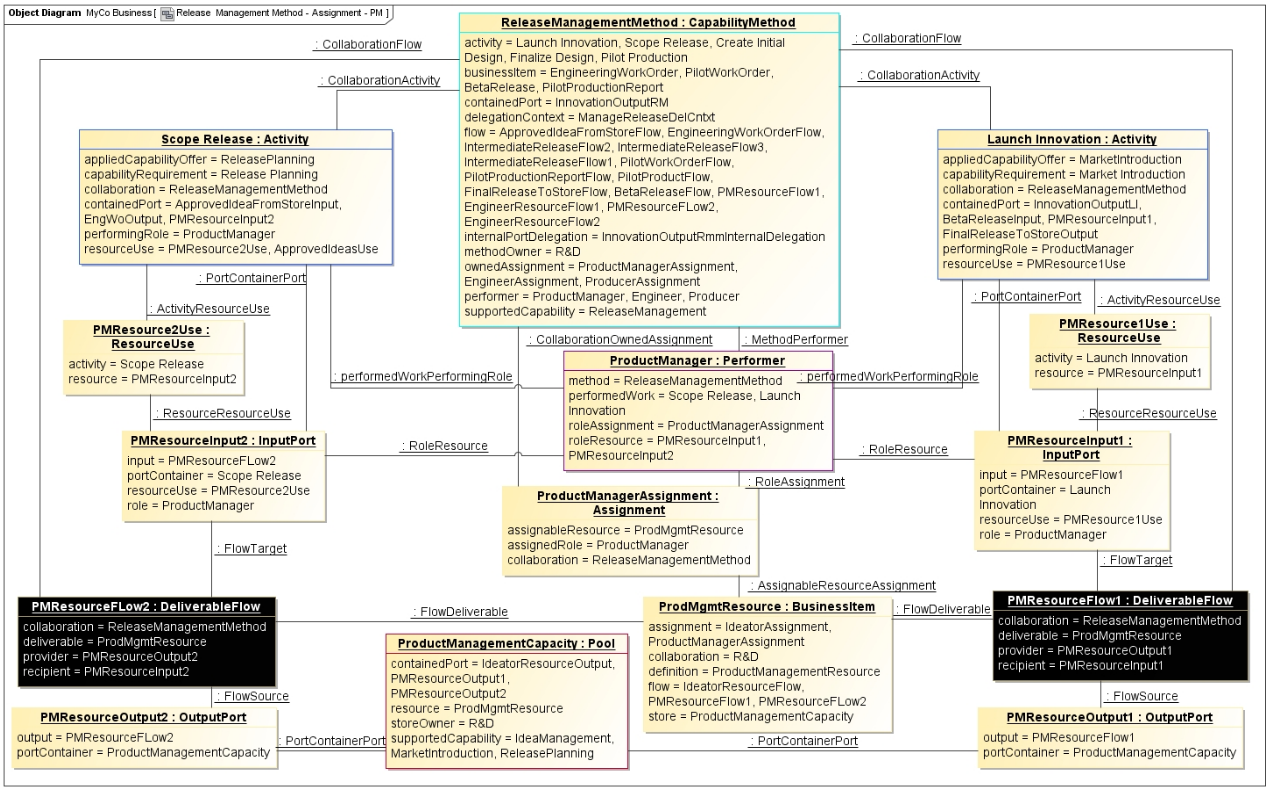

Object Diagram MyCo Business [Release Management Method - Assignment - PM]
ReleaseManagementMethod : CapabilityMethod
activity = Launch Innovation, Scope Release, Create Initial Design, Finalize Design, Pilot Production
businessItem = EngineeringWorkOrder, PilotWorkOrder, BetaRelease, PilotProductionReport
containedPort = InnovationOutputRM
delegationContext = ManageReleaseDelCntxt
flow = ApprovedIdeaFromStoreFlow, EngineeringWorkOrderFlow, IntermediateReleaseFlow2, IntermediateReleaseFlow3, IntermediateReleaseFllow1, PilotWorkOrderFlow, PilotProductionReportFlow, PilotProductFlow, FinalReleaseToStoreFlow, BetaReleaseFlow, PMResourceFlow1, EngineerResourceFlow1, PMResourceFLow2, EngineerResourceFlow2
internalPortDelegation = InnovationOutputRmmInternalDelegation
methodOwner = R&D
ownedAssignment = ProductManagerAssignment, EngineerAssignment, ProducerAssignment
performer = ProductManager, Engineer, Producer
supportedCapability = ReleaseManagement
: CollaborationFlow
: CollaborationActivity
: CollaborationFlow
: CollaborationActivity
Scope Release : Activity
appliedCapabilityOffer = ReleasePlanning
capabilityRequirement = Release Planning
collaboration = ReleaseManagementMethod
containedPort = ApprovedIdeaFromStoreInput, EngWoOutput, PMResourceInput2
performingRole = ProductManager
resourceUse = PMResource2Use, ApprovedIdeasUse
Launch Innovation : Activity
appliedCapabilityOffer = MarketIntroduction
capabilityRequirement = Market Introduction
collaboration = ReleaseManagementMethod
containedPort = InnovationOutputLI, BetaReleaseInput, PMResourceInput1, FinalReleaseToStoreOutput
performingRole = ProductManager
resourceUse = PMResource1Use
: PortContainerPort
: ActivityResourceUse
: PortContainerPort
: ActivityResourceUse
PMResource2Use : ResourceUse
activity = Scope Release
resource = PMResourceInput2
PMResource1Use : ResourceUse
activity = Launch Innovation
resource = PMResourceInput1
: CollaborationOwnedAssignment
: MethodPerformer
ProductManager : Performer
method = ReleaseManagementMethod
performedWork = Scope Release, Launch Innovation
roleAssignment = ProductManagerAssignment
roleResource = PMResourceInput1, PMResourceInput2
: performedWorkPerformingRole
: performedWorkPerformingRole
: ResourceResourceUse
: ResourceResourceUse
PMResourceInput2 : InputPort
input = PMResourceFLow2
portContainer = Scope Release
resourceUse = PMResource2Use
role = ProductManager
: RoleResource
: RoleResource
PMResourceInput1 : InputPort
input = PMResourceFlow1
portContainer = Launch Innovation
resourceUse = PMResource1Use
role = ProductManager
: RoleAssignment
ProductManagerAssignment : Assignment
assignableResource = ProdMgmtResource
assignedRole = ProductManager
collaboration = ReleaseManagementMethod
: FlowTarget
: FlowTarget
: AssignableResourceAssignment
PMResourceFLow2 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = ProdMgmtResource
provider = PMResourceOutput2
recipient = PMResourceInput2
: FlowDeliverable
ProdMgmtResource : BusinessItem
assignment = IdeatorAssignment, ProductManagerAssignment
collaboration = R&D
definition = ProductManagementResource
flow = IdeatorResourceFlow, PMResourceFlow1, PMResourceFLow2
store = ProductManagementCapacity
: FlowDeliverable
PMResourceFlow1 : DeliverableFlow
collaboration = ReleaseManagementMethod
deliverable = ProdMgmtResource
provider = PMResourceOutput1
recipient = PMResourceInput1
ProductManagementCapacity : Pool
containedPort = IdeatorResourceOutput, PMResourceOutput1, PMResourceOutput2
resource = ProdMgmtResource
storeOwner = R&D
supportedCapability = IdeaManagement, MarketIntroduction, ReleasePlanning
: FlowSource
: FlowSource
PMResourceOutput2 : OutputPort
output = PMResourceFLow2
portContainer = ProductManagementCapacity
: PortContainerPort
: PortContainerPort
PMResourceOutput1 : OutputPort
output = PMResourceFlow1
portContainer = ProductManagementCapacity

Figure 76. Release Management method's Product Manager role assignment objects

The XTrailer business network is concerned with both innovation and fulfillment. Structural and behavioral parts that relate to innovation, as well as to the involvement of production in innovation, have been discussed in detail so-far. In the remainder of this section the discussion will focus on what is behind the fulfillment part. The sub-scope of the use case that relates to this is highlighted in the use case overview diagram in Figure 77.

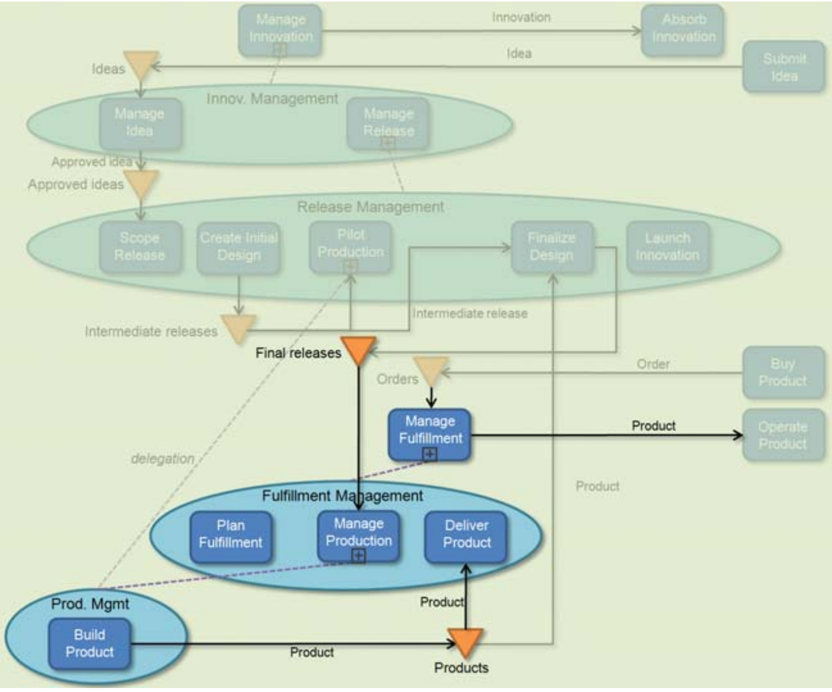
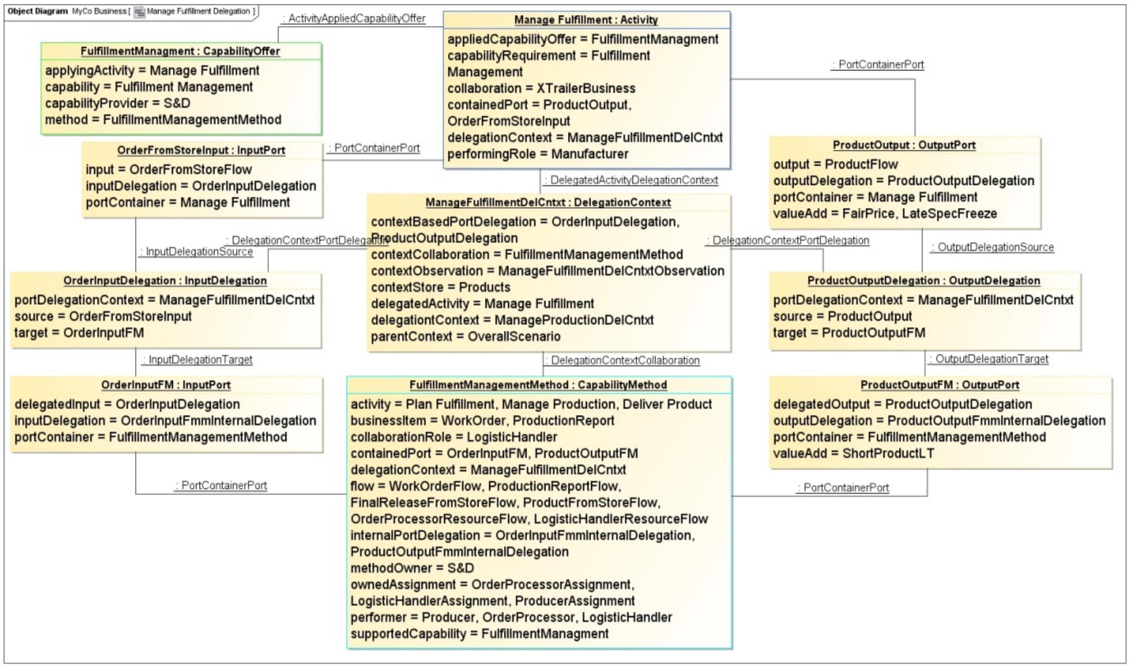


Figure 77. Use case overview: Fulfillment part highlighted

We will in particular discuss the Manage Production activity of the Fulfillment Management method imposes a second delegation context on the Product Management method, as it is involved for both pilot production in release management and “commercial” production in fulfillment management.

Figure 78 shows the objects that are involved in delegating the work of the Manage Fulfillment activity in the business network. For convenience, the reader might again lookup that activity in the activity network mockup of the business network in Figure 42. The activity delegates to the Fulfillment Management method that supports the Fulfillment Management capability offer of S&D. All metamodel constructs that support this have been discussed before.


Object Diagram MyCo Business [Manage Fulfillment Delegation]
: ActivityAppliedCapabilityOffer
: PortContainerPort
: PortContainerPort
: DelegatedActivityDelegationContext
: DelegationContextPortDelegation
: InputDelegationSource
: DelegationContextPortDelegation
: OutputDelegationSource
: InputDelegationTarget
: DelegationContextCollaboration
: OutputDelegationTarget
: PortContainerPort
: PortContainerPort

FulfillmentManagment : CapabilityOffer
applyingActivity = Manage Fulfillment
capability = Fulfillment Management
capabilityProvider = S&D
method = FulfillmentManagementMethod

Manage Fulfillment : Activity
appliedCapabilityOffer = FulfillmentManagment
capabilityRequirement = Fulfillment Management
collaboration = XTrailerBusiness
containedPort = ProductOutput, OrderFromStoreInput
delegationContext = ManageFulfillmentDelCntxt
performingRole = Manufacturer

OrderFromStoreInput : InputPort
input = OrderFromStoreFlow
inputDelegation = OrderInputDelegation
portContainer = Manage Fulfillment

ProductOutput : OutputPort
output = ProductFlow
outputDelegation = ProductOutputDelegation
portContainer = Manage Fulfillment
valueAdd = FairPrice, LateSpecFreeze

ManageFulfillmentDelCntxt : DelegationContext
contextBasedPortDelegation = OrderInputDelegation, ProductOutputDelegation
contextCollaboration = FulfillmentManagementMethod
contextObservation = ManageFulfillmentDelCntxtObservation
contextStore = Products
delegatedActivity = Manage Fulfillment
delegationtContext = ManageProductionDelCntxt
parentContext = OverallScenario

OrderInputDelegation : InputDelegation
portDelegationContext = ManageFulfillmentDelCntxt
source = OrderFromStoreInput
target = OrderInputFM

ProductOutputDelegation : OutputDelegation
portDelegationContext = ManageFulfillmentDelCntxt
source = ProductOutput
target = ProductOutputFM

OrderInputFM : InputPort
delegatedInput = OrderInputDelegation
inputDelegation = OrderInputFmmInternalDelegation
portContainer = FulfillmentManagementMethod

FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment

ProductOutputFM : OutputPort
delegatedOutput = ProductOutputDelegation
outputDelegation = ProductOutputFmmInternalDelegation
portContainer = FulfillmentManagementMethod
valueAdd = ShortProductLT

Figure 78. Delegation to Fulfillment Management method (objects)

Figure 79 shows the activity network mockup of the Fulfillment Management method.

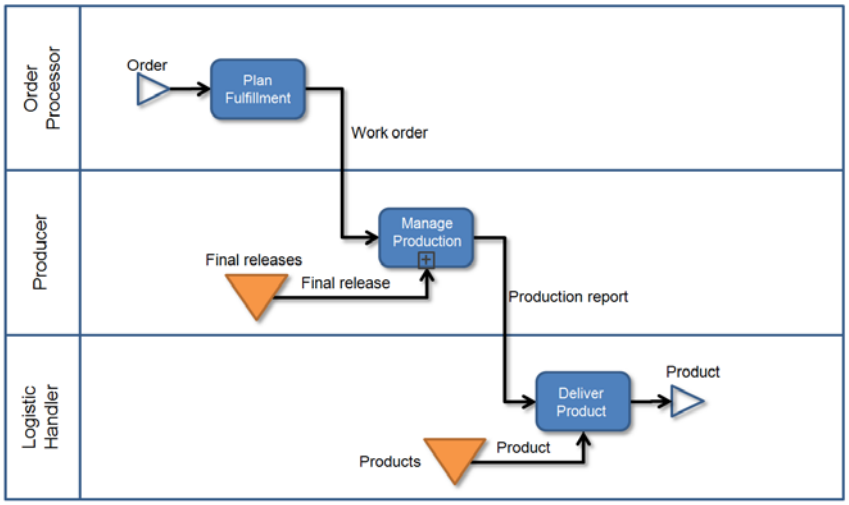
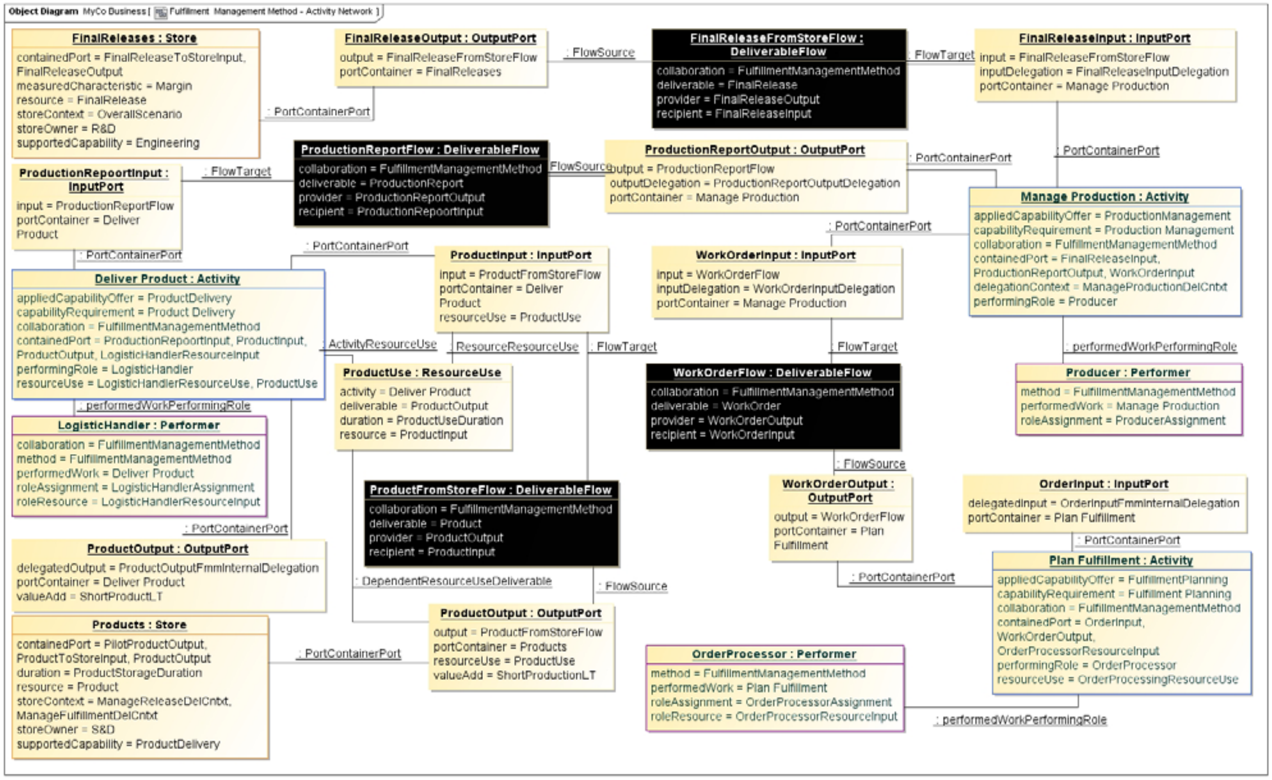


Figure 79. Fulfillment Management method's activity network mockup

The corresponding objects are contained in the object diagram of Figure 80. As the two connectors in the mockup in Figure 79 map to port delegations, rather than to deliverable flows directly, it is clear why the diagram in Figure 80 contains four deliverable flow objects, rather than six. Similar situations have been discussed earlier, in relation to other activity networks of other capability methods.


Object Diagram MyCo Business [Fulfillment Management Method - Activity Network]
: FlowSource
: FlowTarget
: PortContainerPort
: ActivityResourceUse
: ResourceResourceUse
: performedWorkPerformingRole
: DependentResourceUseDeliverable

FinalReleases : Store
containedPort = FinalReleaseToStoreInput, FinalReleaseOutput
measuredCharacteristic = Margin
resource = FinalRelease
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = Engineering

FinalReleaseOutput : OutputPort
output = FinalReleaseFromStoreFlow
portContainer = FinalReleases

FinalReleaseFromStoreFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = FinalRelease
provider = FinalReleaseOutput
recipient = FinalReleaseInput

FinalReleaseInput : InputPort
input = FinalReleaseFromStoreFlow
inputDelegation = FinalReleaseInputDelegation
portContainer = Manage Production

ProductionRepoortInput : InputPort
input = ProductionReportFlow
portContainer = Deliver Product

ProductionReportFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = ProductionReport
provider = ProductionReportOutput
recipient = ProductionRepoortInput

ProductionReportOutput : OutputPort
output = ProductionReportFlow
outputDelegation = ProductionReportOutputDelegation
portContainer = Manage Production

Manage Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = FulfillmentManagementMethod
containedPort = FinalReleaseInput, ProductionReportOutput, WorkOrderInput
delegationContext = ManageProductionDelCntxt
performingRole = Producer

Deliver Product : Activity
appliedCapabilityOffer = ProductDelivery
capabilityRequirement = Product Delivery
collaboration = FulfillmentManagementMethod
containedPort = ProductionRepoortInput, ProductInput, ProductOutput, LogisticHandlerResourceInput
performingRole = LogisticHandler
resourceUse = LogisticHandlerResourceUse, ProductUse

ProductInput : InputPort
input = ProductFromStoreFlow
portContainer = Deliver Product
resourceUse = ProductUse

WorkOrderInput : InputPort
input = WorkOrderFlow
inputDelegation = WorkOrderInputDelegation
portContainer = Manage Production

ProductUse : ResourceUse
activity = Deliver Product
deliverable = ProductOutput
duration = ProductUseDuration
resource = ProductInput

WorkOrderFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = WorkOrder
provider = WorkOrderOutput
recipient = WorkOrderInput

Producer : Performer
method = FulfillmentManagementMethod
performedWork = Manage Production
roleAssignment = ProducerAssignment

LogisticHandler : Performer
collaboration = FulfillmentManagementMethod
method = FulfillmentManagementMethod
performedWork = Deliver Product
roleAssignment = LogisticHandlerAssignment
roleResource = LogisticHandlerResourceInput

ProductFromStoreFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = Product
provider = ProductOutput
recipient = ProductInput

WorkOrderOutput : OutputPort
output = WorkOrderFlow
portContainer = Plan Fulfillment

OrderInput : InputPort
delegatedInput = OrderInputFmmInternalDelegation
portContainer = Plan Fulfillment

ProductOutput : OutputPort
delegatedOutput = ProductOutputFmmInternalDelegation
portContainer = Deliver Product
valueAdd = ShortProductLT

Plan Fulfillment : Activity
appliedCapabilityOffer = FulfillmentPlanning
capabilityRequirement = Fulfillment Planning
collaboration = FulfillmentManagementMethod
containedPort = OrderInput, WorkOrderOutput, OrderProcessorResourceInput
performingRole = OrderProcessor
resourceUse = OrderProcessingResourceUse

Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery

ProductOutput : OutputPort
output = ProductFromStoreFlow
portContainer = Products
resourceUse = ProductUse
valueAdd = ShortProductionLT

OrderProcessor : Performer
method = FulfillmentManagementMethod
performedWork = Plan Fulfillment
roleAssignment = OrderProcessorAssignment
roleResource = OrderProcessorResourceInput

Figure 80. Fulfillment Management method's activity network objects

The object diagram in Figure 81 contains the objects that define how the Order input and Product output on the "boundary" of the Fulfillment Management method are delegated to corresponding ports of activities inside the method. Similar constructs have been discussed in relation to the Innovation Management and Release Management methods earlier.

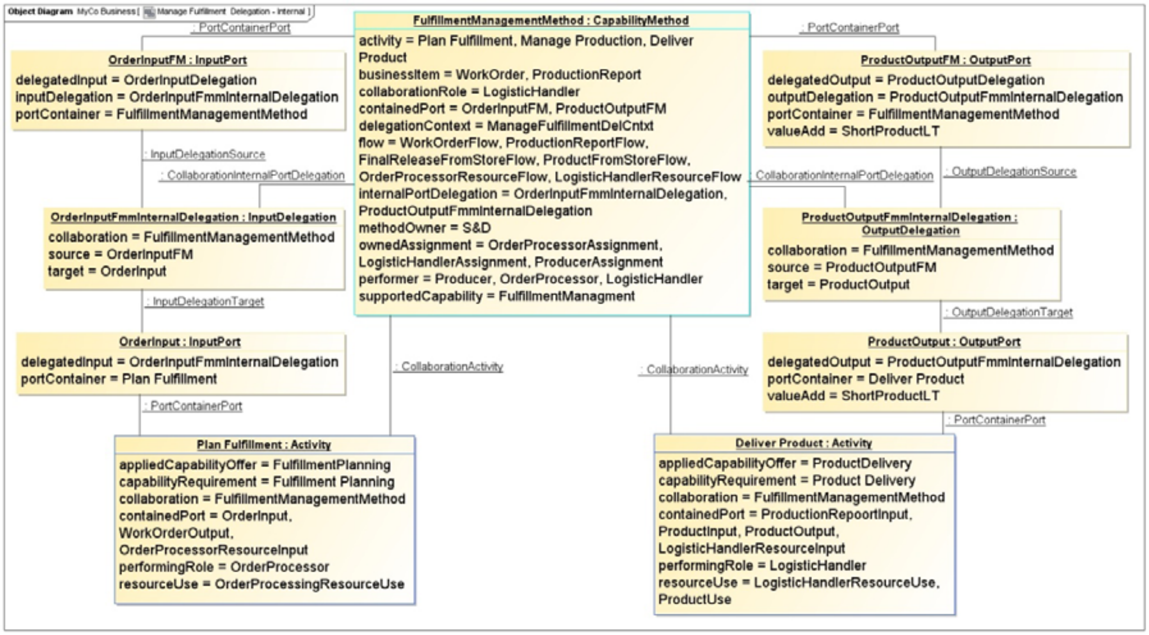

Object Diagram MyCo Business [Manage Fulfillment Delegation - Internal]
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
: PortContainerPort
OrderInputFM : InputPort
delegatedInput = OrderInputDelegation
inputDelegation = OrderInputFmmInternalDelegation
portContainer = FulfillmentManagementMethod
: InputDelegationSource
: CollaborationInternalPortDelegation
OrderInputFmmInternalDelegation : InputDelegation
collaboration = FulfillmentManagementMethod
source = OrderInputFM
target = OrderInput
: InputDelegationTarget
OrderInput : InputPort
delegatedInput = OrderInputFmmInternalDelegation
portContainer = Plan Fulfillment
: PortContainerPort
: CollaborationActivity
Plan Fulfillment : Activity
appliedCapabilityOffer = FulfillmentPlanning
capabilityRequirement = Fulfillment Planning
collaboration = FulfillmentManagementMethod
containedPort = OrderInput, WorkOrderOutput, OrderProcessorResourceInput
performingRole = OrderProcessor
resourceUse = OrderProcessingResourceUse
: PortContainerPort
ProductOutputFM : OutputPort
delegatedOutput = ProductOutputDelegation
outputDelegation = ProductOutputFmmInternalDelegation
portContainer = FulfillmentManagementMethod
valueAdd = ShortProductLT
: CollaborationInternalPortDelegation
: OutputDelegationSource
ProductOutputFmmInternalDelegation : OutputDelegation
collaboration = FulfillmentManagementMethod
source = ProductOutputFM
target = ProductOutput
: OutputDelegationTarget
ProductOutput : OutputPort
delegatedOutput = ProductOutputFmmInternalDelegation
portContainer = Deliver Product
valueAdd = ShortProductLT
: CollaborationActivity
: PortContainerPort
Deliver Product : Activity
appliedCapabilityOffer = ProductDelivery
capabilityRequirement = Product Delivery
collaboration = FulfillmentManagementMethod
containedPort = ProductionRepoortInput, ProductInput, ProductOutput, LogisticHandlerResourceInput
performingRole = LogisticHandler
resourceUse = LogisticHandlerResourceUse, ProductUse

Figure 81. Fulfillment Management method's internal delegation objects

The objects in the object diagram in Figure 82 define assignment of Order Processing resource, from the Order Processing Capacity pool, to the Order Processor performer role in the Fufillment Management method. Underlying metamodel concepts have been discussed earlier, in relation to similar constructs in the Innovation Management and Release Management methods.

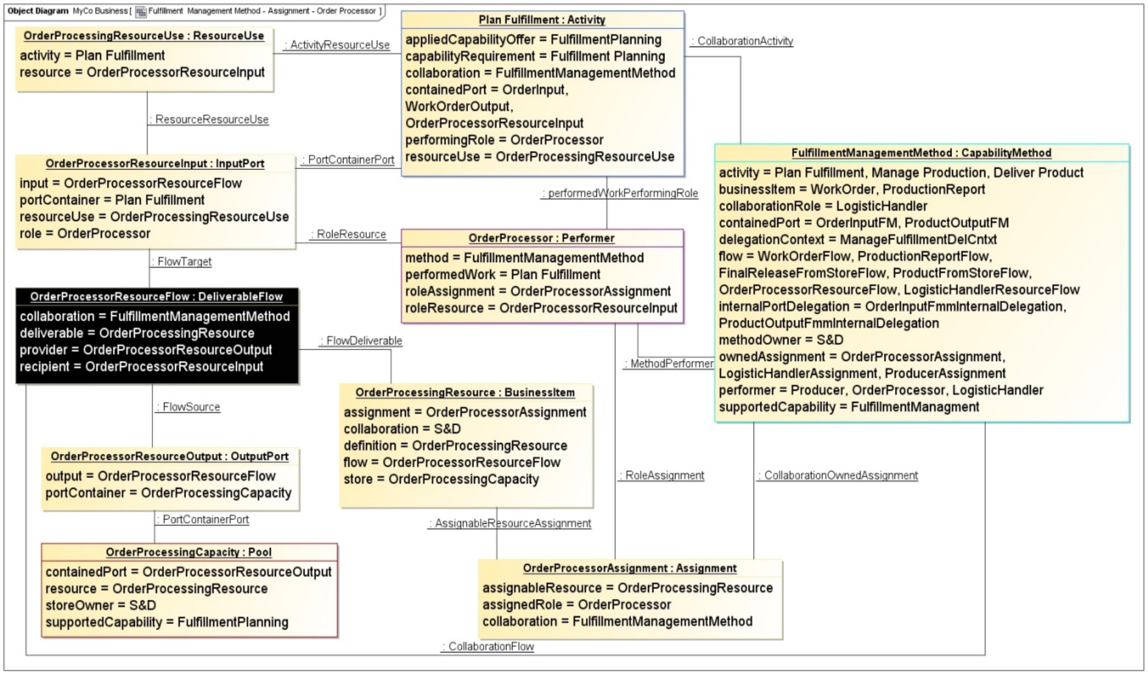

Object Diagram MyCo Business [Fulfillment Management Method - Assignment - Order Processor]
OrderProcessingResourceUse : ResourceUse
activity = Plan Fulfillment
resource = OrderProcessorResourceInput
: ActivityResourceUse
: ResourceResourceUse
Plan Fulfillment : Activity
appliedCapabilityOffer = FulfillmentPlanning
capabilityRequirement = Fulfillment Planning
collaboration = FulfillmentManagementMethod
containedPort = OrderInput, WorkOrderOutput, OrderProcessorResourceInput
performingRole = OrderProcessor
resourceUse = OrderProcessingResourceUse
: CollaborationActivity
OrderProcessorResourceInput : InputPort
input = OrderProcessorResourceFlow
portContainer = Plan Fulfillment
resourceUse = OrderProcessingResourceUse
role = OrderProcessor
: PortContainerPort
: performedWorkPerformingRole
: RoleResource
OrderProcessor : Performer
method = FulfillmentManagementMethod
performedWork = Plan Fulfillment
roleAssignment = OrderProcessorAssignment
roleResource = OrderProcessorResourceInput
: FlowTarget
OrderProcessorResourceFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = OrderProcessingResource
provider = OrderProcessorResourceOutput
recipient = OrderProcessorResourceInput
: FlowDeliverable
: MethodPerformer
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
: FlowSource
OrderProcessingResource : BusinessItem
assignment = OrderProcessorAssignment
collaboration = S&D
definition = OrderProcessingResource
flow = OrderProcessorResourceFlow
store = OrderProcessingCapacity
OrderProcessorResourceOutput : OutputPort
output = OrderProcessorResourceFlow
portContainer = OrderProcessingCapacity
: RoleAssignment
: CollaborationOwnedAssignment
: PortContainerPort
: AssignableResourceAssignment
OrderProcessingCapacity : Pool
containedPort = OrderProcessorResourceOutput
resource = OrderProcessingResource
storeOwner = S&D
supportedCapability = FulfillmentPlanning
OrderProcessorAssignment : Assignment
assignableResource = OrderProcessingResource
assignedRole = OrderProcessor
collaboration = FulfillmentManagementMethod
: CollaborationFlow

Figure 82. Fulfillment Management method's Order Processor role assignment objects

Similarly, the object diagram in Figure 83 contains the objects that model assignment of role resource to the Logistic Handler performer role.

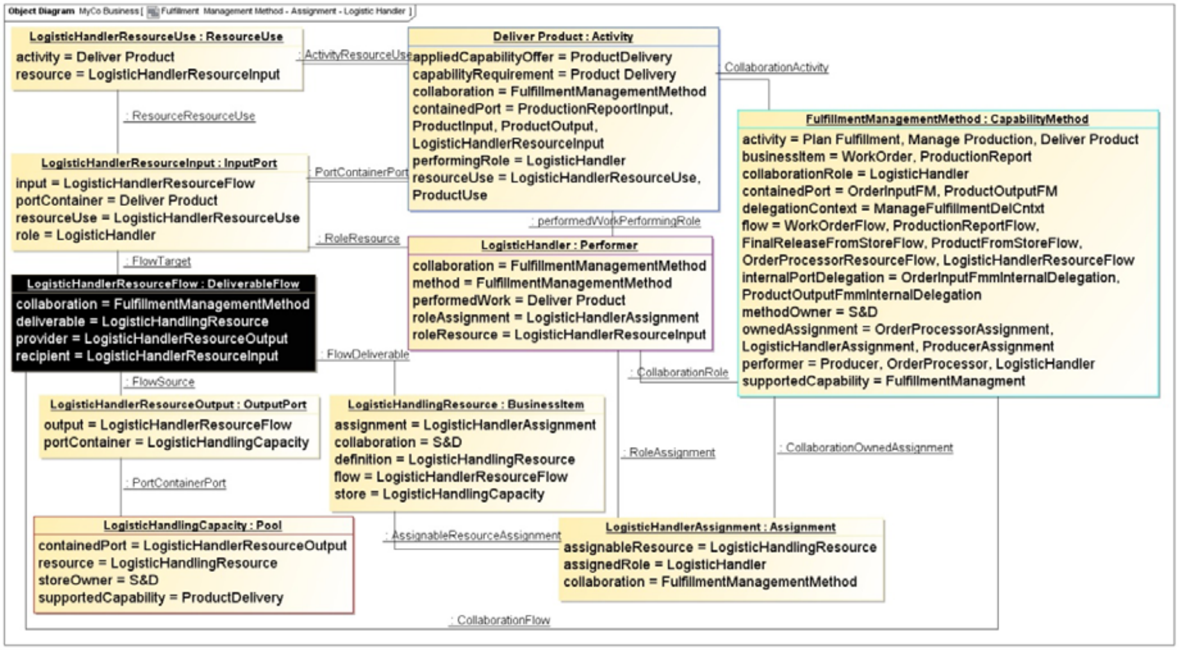

Object Diagram MyCo Business [Fulfillment Management Method - Assignment - Logistic Handler]
LogisticHandlerResourceUse : ResourceUse
activity = Deliver Product
resource = LogisticHandlerResourceInput
: ActivityResourceUse
Deliver Product : Activity
appliedCapabilityOffer = ProductDelivery
capabilityRequirement = Product Delivery
collaboration = FulfillmentManagementMethod
containedPort = ProductionRepoortInput, ProductInput, ProductOutput, LogisticHandlerResourceInput
performingRole = LogisticHandler
resourceUse = LogisticHandlerResourceUse, ProductUse
CollaborationActivity
: ResourceResourceUse
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
LogisticHandlerResourceInput : InputPort
input = LogisticHandlerResourceFlow
portContainer = Deliver Product
resourceUse = LogisticHandlerResourceUse
role = LogisticHandler
: PortContainerPort
: performedWorkPerformingRole
: RoleResource
LogisticHandler : Performer
collaboration = FulfillmentManagementMethod
method = FulfillmentManagementMethod
performedWork = Deliver Product
roleAssignment = LogisticHandlerAssignment
roleResource = LogisticHandlerResourceInput
: FlowTarget
LogisticHandlerResourceFlow : DeliverableFlow
collaboration = FulfillmentManagementMethod
deliverable = LogisticHandlingResource
provider = LogisticHandlerResourceOutput
recipient = LogisticHandlerResourceInput
: FlowDeliverable
: CollaborationRole
: FlowSource
LogisticHandlerResourceOutput : OutputPort
output = LogisticHandlerResourceFlow
portContainer = LogisticHandlingCapacity
LogisticHandlingResource : BusinessItem
assignment = LogisticHandlerAssignment
collaboration = S&D
definition = LogisticHandlingResource
flow = LogisticHandlerResourceFlow
store = LogisticHandlingCapacity
: RoleAssignment
: CollaborationOwnedAssignment
: PortContainerPort
LogisticHandlingCapacity : Pool
containedPort = LogisticHandlerResourceOutput
resource = LogisticHandlingResource
storeOwner = S&D
supportedCapability = ProductDelivery
: AssignableResourceAssignment
LogisticHandlerAssignment : Assignment
assignableResource = LogisticHandlingResource
assignedRole = LogisticHandler
collaboration = FulfillmentManagementMethod
: CollaborationFlow

Figure 83. Fulfillment Management method’s Logistic Handler role assignment objects

As the object diagram in Figure 84 suggests, the Producer role in the Fufillment Management method is assigned to the Trailer Plant.

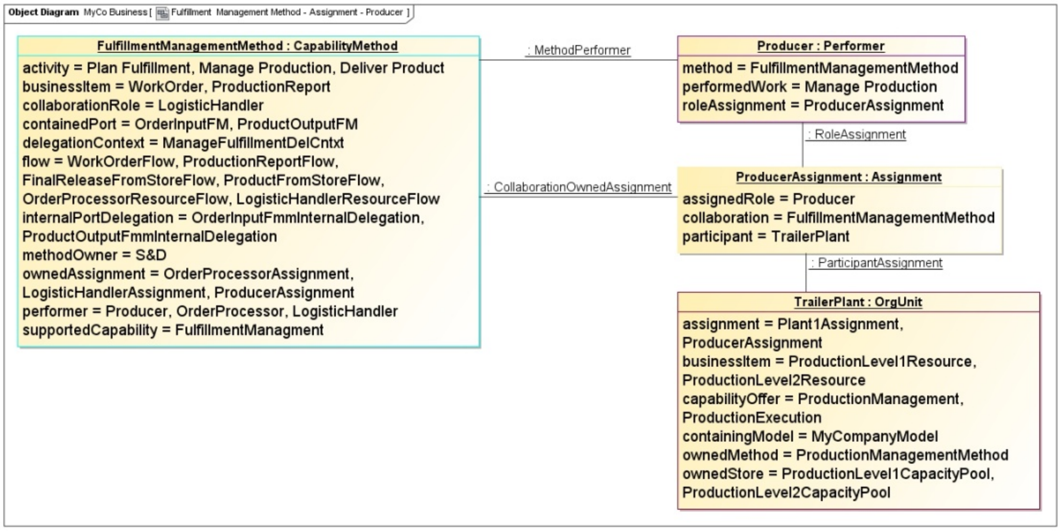

Object Diagram MyCo Business [Fulfillment Management Method - Assignment - Producer]
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
: MethodPerformer
Producer : Performer
method = FulfillmentManagementMethod
performedWork = Manage Production
roleAssignment = ProducerAssignment
: RoleAssignment
: CollaborationOwnedAssignment
ProducerAssignment : Assignment
assignedRole = Producer
collaboration = FulfillmentManagementMethod
participant = TrailerPlant
: ParticipantAssignment
TrailerPlant : OrgUnit
assignment = Plant1Assignment, ProducerAssignment
businessItem = ProductionLevel1Resource, ProductionLevel2Resource
capabilityOffer = ProductionManagement, ProductionExecution
containingModel = MyCompanyModel
ownedMethod = ProductionManagementMethod
ownedStore = ProductionLevel1CapacityPool, ProductionLevel2CapacityPool

Figure 84. Fulfillment Management method’s Producer role assignment objects

And this is based on the fact that the Trailer Plant organization unit provides a capability offer for the Product Management capability (see again the mockup in Figure 39), that is required by the Manage Production activity, which is performed by the Producer role in the Fulfillment Management method. The Manage Production activity delegates its work to the Production Management method of the Trailer Plant, as is defined by the objects in the object diagram in Figure 85. Note that the delegation involves three port delegations, from ports of the activity to ports on the “boundary” of the Product Management method. Two of these delegated ports handle inputs: Final Release and Work Order. One of them handles an output: Production Report.

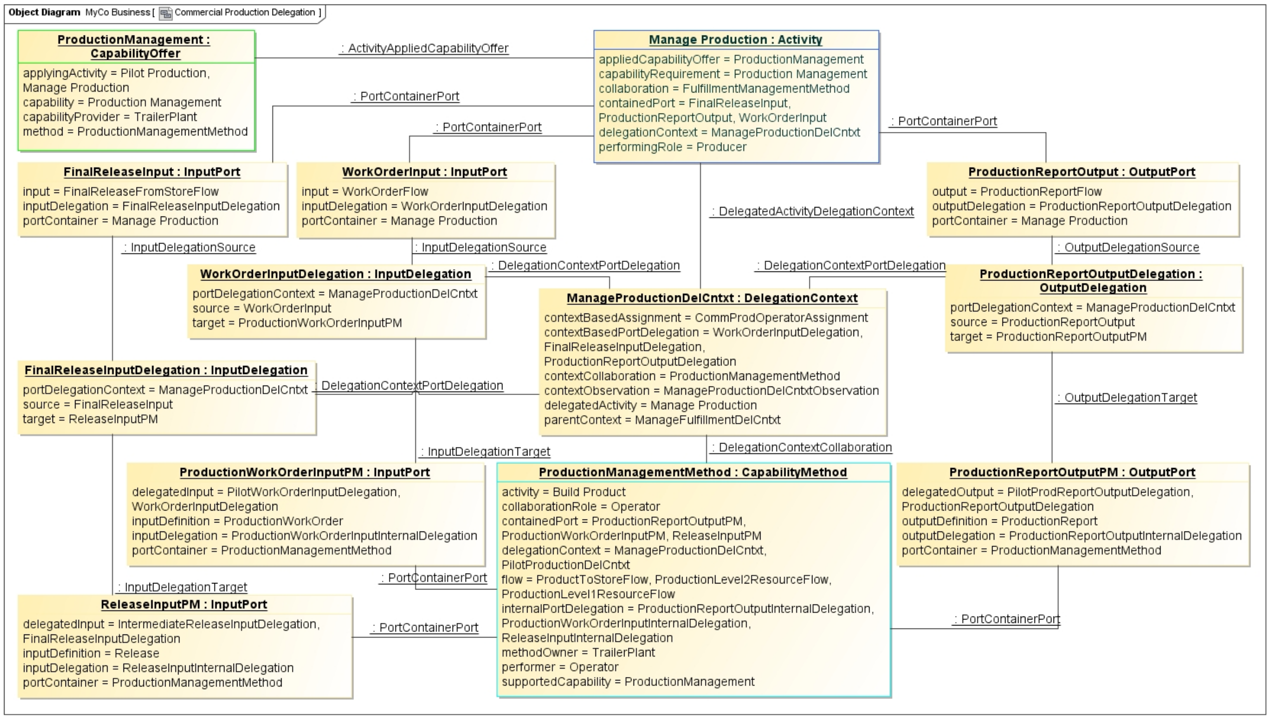

Object Diagram MyCo Business [Commercial Production Delegation]
ProductionManagement : CapabilityOffer
applyingActivity = Pilot Production, Manage Production
capability = Production Management
capabilityProvider = TrailerPlant
method = ProductionManagementMethod
: ActivityAppliedCapabilityOffer
Manage Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = FulfillmentManagementMethod
containedPort = FinalReleaseInput, ProductionReportOutput, WorkOrderInput
delegationContext = ManageProductionDelCntxt
performingRole = Producer
: PortContainerPort
: PortContainerPort
: PortContainerPort
FinalReleaseInput : InputPort
input = FinalReleaseFromStoreFlow
inputDelegation = FinalReleaseInputDelegation
portContainer = Manage Production
WorkOrderInput : InputPort
input = WorkOrderFlow
inputDelegation = WorkOrderInputDelegation
portContainer = Manage Production
: DelegatedActivityDelegationContext
ProductionReportOutput : OutputPort
output = ProductionReportFlow
outputDelegation = ProductionReportOutputDelegation
portContainer = Manage Production
: InputDelegationSource
: InputDelegationSource
: OutputDelegationSource
: DelegationContextPortDelegation
: DelegationContextPortDelegation
WorkOrderInputDelegation : InputDelegation
portDelegationContext = ManageProductionDelCntxt
source = WorkOrderInput
target = ProductionWorkOrderInputPM
ManageProductionDelCntxt : DelegationContext
contextBasedAssignment = CommProdOperatorAssignment
contextBasedPortDelegation = WorkOrderInputDelegation, FinalReleaseInputDelegation, ProductionReportOutputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = ManageProductionDelCntxtObservation
delegatedActivity = Manage Production
parentContext = ManageFulfillmentDelCntxt
ProductionReportOutputDelegation : OutputDelegation
portDelegationContext = ManageProductionDelCntxt
source = ProductionReportOutput
target = ProductionReportOutputPM
FinalReleaseInputDelegation : InputDelegation
portDelegationContext = ManageProductionDelCntxt
source = FinalReleaseInput
target = ReleaseInputPM
: DelegationContextPortDelegation
: OutputDelegationTarget
: InputDelegationTarget
: DelegationContextCollaboration
ProductionWorkOrderInputPM : InputPort
delegatedInput = PilotWorkOrderInputDelegation, WorkOrderInputDelegation
inputDefinition = ProductionWorkOrder
inputDelegation = ProductionWorkOrderInputInternalDelegation
portContainer = ProductionManagementMethod
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
ProductionReportOutputPM : OutputPort
delegatedOutput = PilotProdReportOutputDelegation, ProductionReportOutputDelegation
outputDefinition = ProductionReport
outputDelegation = ProductionReportOutputInternalDelegation
portContainer = ProductionManagementMethod
: PortContainerPort
: InputDelegationTarget
ReleaseInputPM : InputPort
delegatedInput = IntermediateReleaseInputDelegation, FinalReleaseInputDelegation
inputDefinition = Release
inputDelegation = ReleaseInputInternalDelegation
portContainer = ProductionManagementMethod
: PortContainerPort
: PortContainerPort

Figure 85. Delegation to Production Management method, from Manage Production (objects)

Figure 86 shows the activity network mockup of the Production Management method. The small "bottom-left pyramid" shapes denote ports, two output ports and one input ports on the "boundary" of the method. Their related connectors denote port delegations from these "boundary" ports to ports of the Build Product activity.

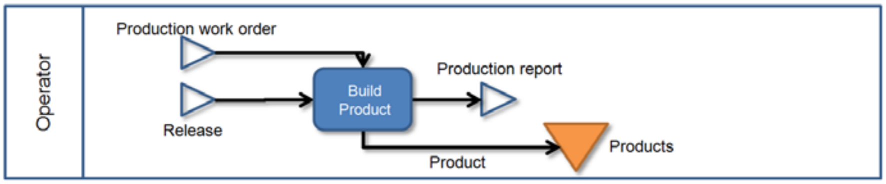


Figure 86. Production Management method's activity network mockup

The objects that define these method-internal delegations are contained in the object diagram of Figure 87.

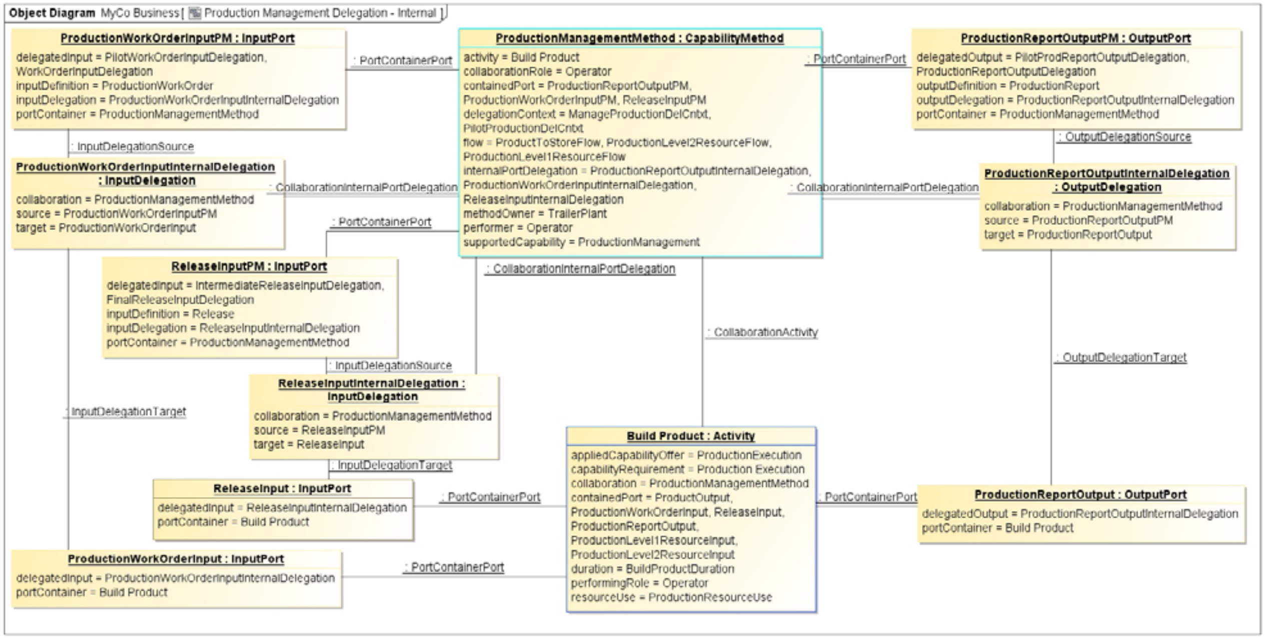

Object Diagram MyCo Business [Production Management Delegation - Internal]
ProductionWorkOrderInputPM : InputPort
delegatedInput = PilotWorkOrderInputDelegation, WorkOrderInputDelegation
inputDefinition = ProductionWorkOrder
inputDelegation = ProductionWorkOrderInputInternalDelegation
portContainer = ProductionManagementMethod
: PortContainerPort
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
: PortContainerPort
ProductionReportOutputPM : OutputPort
delegatedOutput = PilotProdReportOutputDelegation, ProductionReportOutputDelegation
outputDefinition = ProductionReport
outputDelegation = ProductionReportOutputInternalDelegation
portContainer = ProductionManagementMethod
: InputDelegationSource
ProductionWorkOrderInputInternalDelegation : InputDelegation
collaboration = ProductionManagementMethod
source = ProductionWorkOrderInputPM
target = ProductionWorkOrderInput
: CollaborationInternalPortDelegation
: OutputDelegationSource
: CollaborationInternalPortDelegation
ProductionReportOutputInternalDelegation : OutputDelegation
collaboration = ProductionManagementMethod
source = ProductionReportOutputPM
target = ProductionReportOutput
: PortContainerPort
ReleaseInputPM : InputPort
delegatedInput = IntermediateReleaseInputDelegation, FinalReleaseInputDelegation
inputDefinition = Release
inputDelegation = ReleaseInputInternalDelegation
portContainer = ProductionManagementMethod
: CollaborationInternalPortDelegation
: CollaborationActivity
: OutputDelegationTarget
: InputDelegationSource
ReleaseInputInternalDelegation : InputDelegation
collaboration = ProductionManagementMethod
source = ReleaseInputPM
target = ReleaseInput
: InputDelegationTarget
: InputDelegationTarget
Build Product : Activity
appliedCapabilityOffer = ProductionExecution
capabilityRequirement = Production Execution
collaboration = ProductionManagementMethod
containedPort = ProductOutput, ProductionWorkOrderInput, ReleaseInput, ProductionReportOutput, ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
duration = BuildProductDuration
performingRole = Operator
resourceUse = ProductionResourceUse
ReleaseInput : InputPort
delegatedInput = ReleaseInputInternalDelegation
portContainer = Build Product
: PortContainerPort
: PortContainerPort
ProductionReportOutput : OutputPort
delegatedOutput = ProductionReportOutputInternalDelegation
portContainer = Build Product
ProductionWorkOrderInput : InputPort
delegatedInput = ProductionWorkOrderInputInternalDelegation
portContainer = Build Product
: PortContainerPort

Figure 87. Production Management method’s internal delegation objects

The objects that define the activity network of the Production Management method in the mockup in Figure 86 are contained in the object diagram in Figure 88.

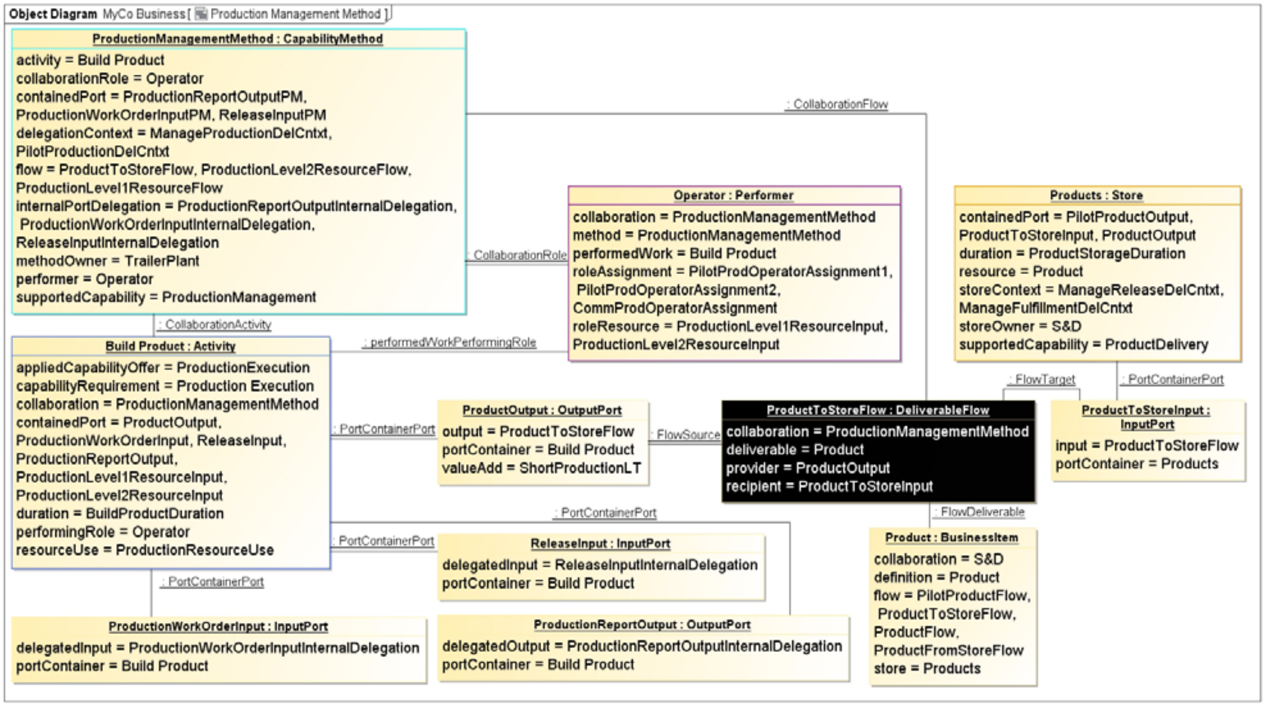

Object Diagram MyCo Business [Production Management Method]
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
: CollaborationFlow
: CollaborationRole
Operator : Performer
collaboration = ProductionManagementMethod
method = ProductionManagementMethod
performedWork = Build Product
roleAssignment = PilotProdOperatorAssignment1, PilotProdOperatorAssignment2, CommProdOperatorAssignment
roleResource = ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery
: CollaborationActivity
: performedWorkPerformingRole
Build Product : Activity
appliedCapabilityOffer = ProductionExecution
capabilityRequirement = Production Execution
collaboration = ProductionManagementMethod
containedPort = ProductOutput, ProductionWorkOrderInput, ReleaseInput, ProductionReportOutput, ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
duration = BuildProductDuration
performingRole = Operator
resourceUse = ProductionResourceUse
: FlowTarget
: PortContainerPort
: PortContainerPort
ProductOutput : OutputPort
output = ProductToStoreFlow
portContainer = Build Product
valueAdd = ShortProductionLT
: FlowSource
ProductToStoreFlow : DeliverableFlow
collaboration = ProductionManagementMethod
deliverable = Product
provider = ProductOutput
recipient = ProductToStoreInput
ProductToStoreInput : InputPort
input = ProductToStoreFlow
portContainer = Products
: PortContainerPort
: FlowDeliverable
: PortContainerPort
ReleaseInput : InputPort
delegatedInput = ReleaseInputInternalDelegation
portContainer = Build Product
Product : BusinessItem
collaboration = S&D
definition = Product
flow = PilotProductFlow, ProductToStoreFlow, ProductFlow, ProductFromStoreFlow
store = Products
: PortContainerPort
ProductionWorkOrderInput : InputPort
delegatedInput = ProductionWorkOrderInputInternalDelegation
portContainer = Build Product
ProductionReportOutput : OutputPort
delegatedOutput = ProductionReportOutputInternalDelegation
portContainer = Build Product

Figure 88. Production Management method’s activity network objects

Note again, that, as three of the four activity ports are delegated from the method’s “boundary” ports, this object diagram contains just one delivery flow object. This is the flow that delivers the product to the Products store.

The next two object diagrams, in Figure 89, and Figure 90, demonstrate how assignment of roles in a collaboration (here assignment of performer roles in a capability method) can be made dependent on and specific to the delegation context in which the collaboration (here the capability method) is used. As the earlier discussed metamodel diagram of Scenario and Analysis Context in Figure 51 suggests, a delegation context can specify:

- Context based port delegations; these have been applied several times in the use case so-far.
- Context based assignments; these have not been applied to the use case so-far, but will be discussed here, in relation to the Operator role in the Production Management method.
- As analysis context: a context observation. Observations will be discussed in 3.1.5.

As mentioned earlier, "commercial" production work requires higher operator skills than pilot production work (i.e. production work in the context of developing new releases). The production plant's capability offer is supported by two resource pools (see again the mockup of the Trailer Plant's capability offers in Figure 39): Production Level 1 Capacity (non-certified resources), and Production Level 2 Capacity (certified resources).

When pilot production work has to be conducted, in the context of delegation of the work of the Pilot Production activity of the Release Management method to the Production Management method, resource from both pools can be used, but preferably from Production Level 1 Capacity, as these resources cannot be deployed for "commercial" production work, and they are cheaper also. The objects in the object diagram in Figure 89 define assignment of the Operator role, to role resource from both the pools. As the object diagram indicates, these assignments depend on the delegation context for pilot production. The assignments serve as alternative to each other. Note that there is a single resource use object that relates to both resources (via the corresponding input ports). Though not applied here, the resource use could specify for instance the quantity of resource required. Preferences for both resources, essential to simulation, can be expressed based on the ordering that is defined for the association between Resource Use and Input Port, as follows from the Activities metamodel diagram in Figure 65 above.

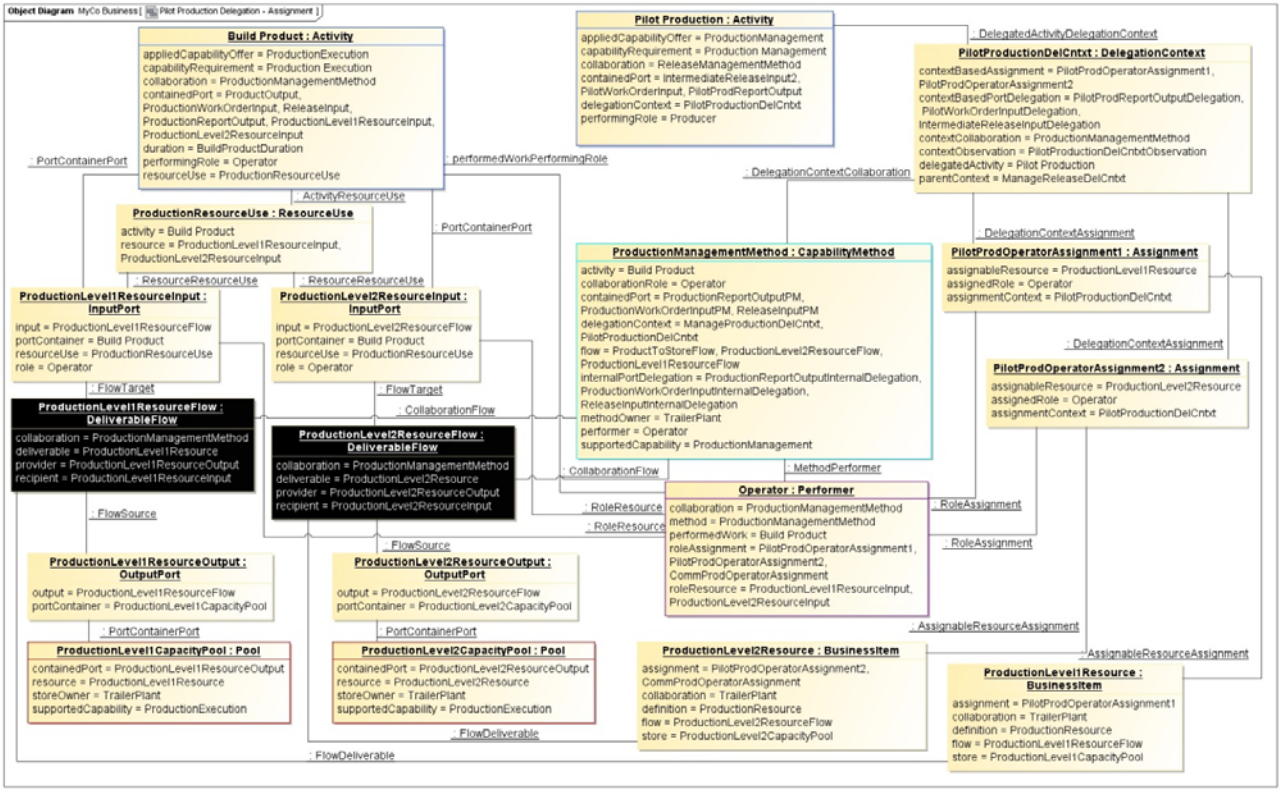

Object Diagram MyCo Business [Pilot Production Delegation - Assignment]
Build Product : Activity
appliedCapabilityOffer = ProductionExecution
capabilityRequirement = Production Execution
collaboration = ProductionManagementMethod
containedPort = ProductOutput,
ProductionWorkOrderInput, ReleaseInput,
ProductionReportOutput, ProductionLevel1ResourceInput,
ProductionLevel2ResourceInput
duration = BuildProductDuration
performingRole = Operator
resourceUse = ProductionResourceUse
Pilot Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = ReleaseManagementMethod
containedPort = IntermediateReleaseInput2,
PilotWorkOrderInput, PilotProdReportOutput
delegationContext = PilotProductionDelCntxt
performingRole = Producer
: DelegatedActivityDelegationContext
PilotProductionDelCntxt : DelegationContext
contextBasedAssignment = PilotProdOperatorAssignment1,
PilotProdOperatorAssignment2
contextBasedPortDelegation = PilotProdReportOutputDelegation,
PilotWorkOrderInputDelegation,
IntermediateReleaseInputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = PilotProductionDelCntxtObservation
delegatedActivity = Pilot Production
parentContext = ManageReleaseDelCntxt
: PortContainerPort
: performedWorkPerformingRole
: DelegationContextCollaboration
: ActivityResourceUse
ProductionResourceUse : ResourceUse
activity = Build Product
resource = ProductionLevel1ResourceInput,
ProductionLevel2ResourceInput
: PortContainerPort
: DelegationContextAssignment
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM,
ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt,
PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow,
ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation,
ProductionWorkOrderInputInternalDelegation,
ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
PilotProdOperatorAssignment1 : Assignment
assignableResource = ProductionLevel1Resource
assignedRole = Operator
assignmentContext = PilotProductionDelCntxt
: ResourceResourceUse
: ResourceResourceUse
ProductionLevel1ResourceInput : InputPort
input = ProductionLevel1ResourceFlow
portContainer = Build Product
resourceUse = ProductionResourceUse
role = Operator
ProductionLevel2ResourceInput : InputPort
input = ProductionLevel2ResourceFlow
portContainer = Build Product
resourceUse = ProductionResourceUse
role = Operator
: DelegationContextAssignment
PilotProdOperatorAssignment2 : Assignment
assignableResource = ProductionLevel2Resource
assignedRole = Operator
assignmentContext = PilotProductionDelCntxt
: FlowTarget
: FlowTarget
: CollaborationFlow
ProductionLevel1ResourceFlow : DeliverableFlow
collaboration = ProductionManagementMethod
deliverable = ProductionLevel1Resource
provider = ProductionLevel1ResourceOutput
recipient = ProductionLevel1ResourceInput
ProductionLevel2ResourceFlow : DeliverableFlow
collaboration = ProductionManagementMethod
deliverable = ProductionLevel2Resource
provider = ProductionLevel2ResourceOutput
recipient = ProductionLevel2ResourceInput
: CollaborationFlow
: MethodPerformer
Operator : Performer
collaboration = ProductionManagementMethod
method = ProductionManagementMethod
performedWork = Build Product
roleAssignment = PilotProdOperatorAssignment1,
PilotProdOperatorAssignment2,
CommProdOperatorAssignment
roleResource = ProductionLevel1ResourceInput,
ProductionLevel2ResourceInput
: RoleResource
: RoleResource
: RoleAssignment
: RoleAssignment
: FlowSource
: FlowSource
ProductionLevel1ResourceOutput : OutputPort
output = ProductionLevel1ResourceFlow
portContainer = ProductionLevel1CapacityPool
ProductionLevel2ResourceOutput : OutputPort
output = ProductionLevel2ResourceFlow
portContainer = ProductionLevel2CapacityPool
: AssignableResourceAssignment
: PortContainerPort
: PortContainerPort
ProductionLevel1CapacityPool : Pool
containedPort = ProductionLevel1ResourceOutput
resource = ProductionLevel1Resource
storeOwner = TrailerPlant
supportedCapability = ProductionExecution
ProductionLevel2CapacityPool : Pool
containedPort = ProductionLevel2ResourceOutput
resource = ProductionLevel2Resource
storeOwner = TrailerPlant
supportedCapability = ProductionExecution
ProductionLevel2Resource : BusinessItem
assignment = PilotProdOperatorAssignment2,
CommProdOperatorAssignment
collaboration = TrailerPlant
definition = ProductionResource
flow = ProductionLevel2ResourceFlow
store = ProductionLevel2CapacityPool
: AssignableResourceAssignment
ProductionLevel1Resource : BusinessItem
assignment = PilotProdOperatorAssignment1
collaboration = TrailerPlant
definition = ProductionResource
flow = ProductionLevel1ResourceFlow
store = ProductionLevel1CapacityPool
: FlowDeliverable
: FlowDeliverable

Figure 89. Production Management method's Operator role assignment objects (context: pilot production)

Note that such aspects as costs of resources, and calculations based on these, can be modeled via measured characteristics of e.g. pools, from where these costs can be further aggregated into measurements of other measured characteristics. The measurement part of VDML will be demonstrated in 3.1.5, though only applied selectively, to keep the use case small.

The object diagram of Figure 90 shows how the Operator role is just assigned to role resource from the Production Level 2 Capacity pool, as only certified resources are supposed to produce a product that will be exchanged with customers.

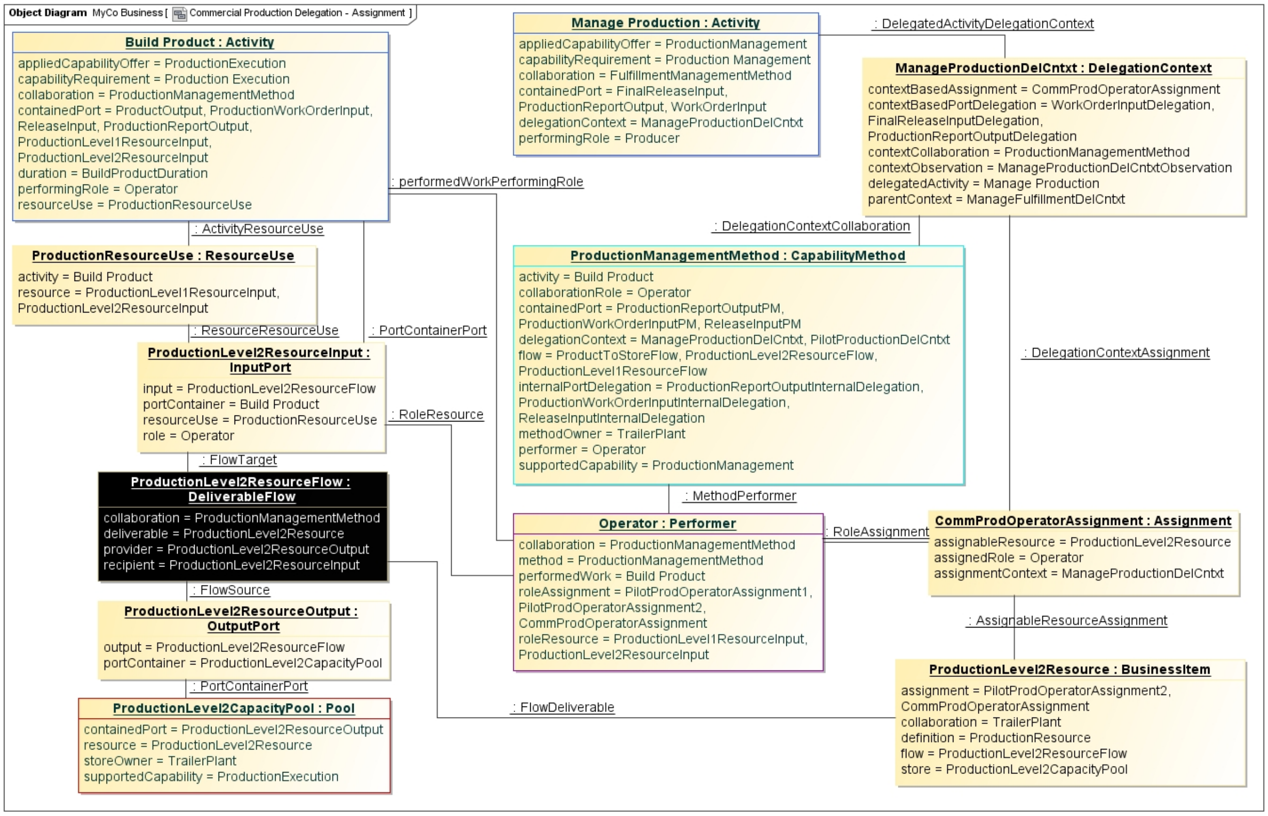

Object Diagram MyCo Business [Commercial Production Delegation - Assignment]
Build Product : Activity
appliedCapabilityOffer = ProductionExecution
capabilityRequirement = Production Execution
collaboration = ProductionManagementMethod
containedPort = ProductOutput, ProductionWorkOrderInput, ReleaseInput, ProductionReportOutput, ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
duration = BuildProductDuration
performingRole = Operator
resourceUse = ProductionResourceUse
Manage Production : Activity
appliedCapabilityOffer = ProductionManagement
capabilityRequirement = Production Management
collaboration = FulfillmentManagementMethod
containedPort = FinalReleaseInput, ProductionReportOutput, WorkOrderInput
delegationContext = ManageProductionDelCntxt
performingRole = Producer
: DelegatedActivityDelegationContext
ManageProductionDelCntxt : DelegationContext
contextBasedAssignment = CommProdOperatorAssignment
contextBasedPortDelegation = WorkOrderInputDelegation, FinalReleaseInputDelegation, ProductionReportOutputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = ManageProductionDelCntxtObservation
delegatedActivity = Manage Production
parentContext = ManageFulfillmentDelCntxt
: performedWorkPerformingRole
: ActivityResourceUse
: DelegationContextCollaboration
ProductionResourceUse : ResourceUse
activity = Build Product
resource = ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
ProductionManagementMethod : CapabilityMethod
activity = Build Product
collaborationRole = Operator
containedPort = ProductionReportOutputPM, ProductionWorkOrderInputPM, ReleaseInputPM
delegationContext = ManageProductionDelCntxt, PilotProductionDelCntxt
flow = ProductToStoreFlow, ProductionLevel2ResourceFlow, ProductionLevel1ResourceFlow
internalPortDelegation = ProductionReportOutputInternalDelegation, ProductionWorkOrderInputInternalDelegation, ReleaseInputInternalDelegation
methodOwner = TrailerPlant
performer = Operator
supportedCapability = ProductionManagement
: ResourceResourceUse
: PortContainerPort
: DelegationContextAssignment
ProductionLevel2ResourceInput : InputPort
input = ProductionLevel2ResourceFlow
portContainer = Build Product
resourceUse = ProductionResourceUse
role = Operator
: RoleResource
: FlowTarget
ProductionLevel2ResourceFlow : DeliverableFlow
collaboration = ProductionManagementMethod
deliverable = ProductionLevel2Resource
provider = ProductionLevel2ResourceOutput
recipient = ProductionLevel2ResourceInput
: MethodPerformer
Operator : Performer
collaboration = ProductionManagementMethod
method = ProductionManagementMethod
performedWork = Build Product
roleAssignment = PilotProdOperatorAssignment1, PilotProdOperatorAssignment2, CommProdOperatorAssignment
roleResource = ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
: RoleAssignment
CommProdOperatorAssignment : Assignment
assignableResource = ProductionLevel2Resource
assignedRole = Operator
assignmentContext = ManageProductionDelCntxt
: FlowSource
ProductionLevel2ResourceOutput : OutputPort
output = ProductionLevel2ResourceFlow
portContainer = ProductionLevel2CapacityPool
: AssignableResourceAssignment
: PortContainerPort
ProductionLevel2CapacityPool : Pool
containedPort = ProductionLevel2ResourceOutput
resource = ProductionLevel2Resource
storeOwner = TrailerPlant
supportedCapability = ProductionExecution
: FlowDeliverable
ProductionLevel2Resource : BusinessItem
assignment = PilotProdOperatorAssignment2, CommProdOperatorAssignment
collaboration = TrailerPlant
definition = ProductionResource
flow = ProductionLevel2ResourceFlow
store = ProductionLevel2CapacityPool

Figure 90. Production Management method's Operator role assignment objects (context: "commercial" production)

So-far we have discussed the structural and behavioral parts of the business system in the XTrailer use case example. It is now time to analyze how this business system can create value for the stakeholders, in this case the transporters market. This will require additional elements, related to the concept of "value", as well as the integration with a measurement framework. In the next section we will analyze these things, based on snippets of examples in the context of the XTrailer use case.

3.1.5 Value

As the VDML ontology diagram in Figure 91 highlights, roles, based on their collaboration with each other, provide and/or receive value propositions. Value propositions articulate values. A value proposition is "*an expression of the values offered to a recipient in terms of the recipient's level of satisfaction*". In VDML "value" is defined as "*a measurable benefit delivered to a recipient in association with a business item*". Any value should be identifiable and measurable, either objectively or subjectively. A value may have an objective measure, whereas the measure of the recipient's level of satisfaction is subjective. Though values need not be intrinsic in a deliverable, but may often relate to a transaction, a relationship, a corporate "image" or reputation, etc., values are not actually received by recipients when there is no deliverable (or business item) exchanged whatsoever. Delivery of business items is the basis of exchanging values. Note again that deliverables need not be tangible. They are often intangible. Many values maybe associated with intangibles. As discussed in 3.1.4, roles perform activity. Activities create and consume value. These concepts will be discussed and analyzed in detail, based on application to the XTrailer use case example.

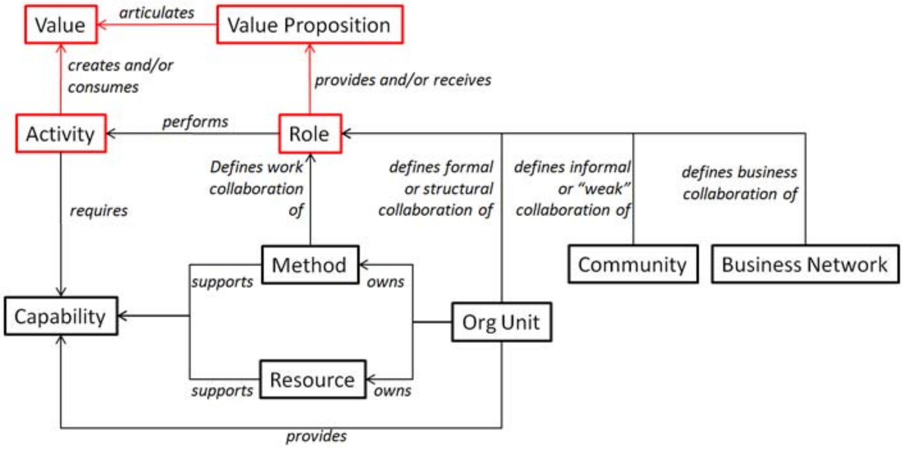


Figure 91. VDML high-level ontology: Value highlighted

Figure 92 shows mockups of the XTrailer business network collaboration. The expanded view, being the actual role collaboration view, has been considered earlier. It shows how party roles exchange deliverables.

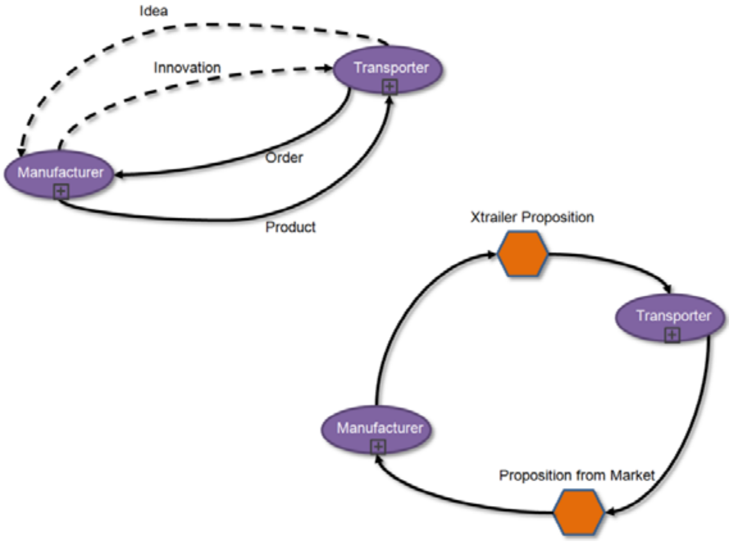


Figure 92. Business network: role collaboration view and proposition exchange view mockups

In collapsed view it just shows how party roles exchange value propositions. Of the two value propositions in Figure 92, we will analyze the XTrailer Proposition, provided by the manufacturer and received by the transporter, in detail. Later on we will briefly consider the value proposition that the transporter provides to the manufacturer in return.

As will be discussed in more detail below, a value proposition articulates values, defined as "value add" elements that are related to the output ports to which the deliverable flows connect. This will enable "toggling" between the two views in Figure 92.

The objects in the proposition exchange view mockup in Figure 92, that define the exchange of the XTrailer Proposition, are contained in the object diagram in Figure 93: The value proposition, as well as the two party roles in the business network, one of which is the provider of the value proposition, and the other being the recipient. As the metamodel diagram of values and value propositions in Figure 97 indicates, the providing role is also the owner of the value proposition.

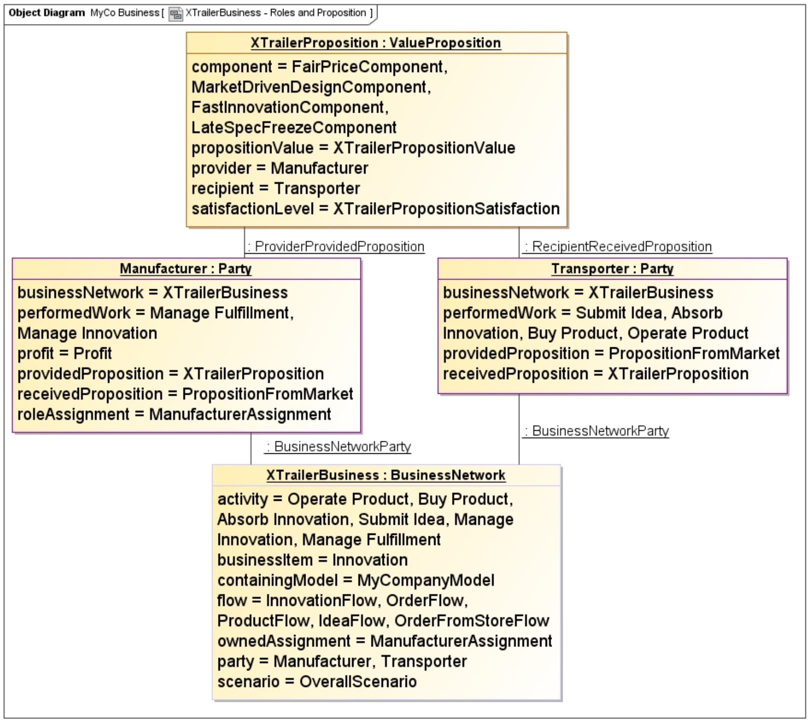
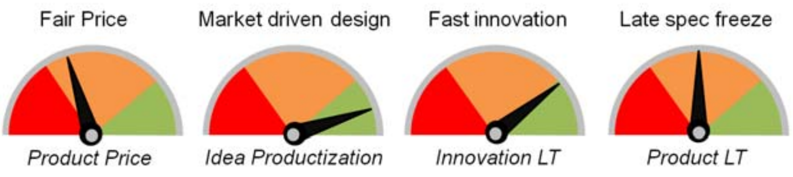

Object Diagram MyCo Business [XTrailerBusiness - Roles and Proposition]
XTrailerProposition : ValueProposition
component = FairPriceComponent, MarketDrivenDesignComponent, FastInnovationComponent, LateSpecFreezeComponent
propositionValue = XTrailerPropositionValue
provider = Manufacturer
recipient = Transporter
satisfactionLevel = XTrailerPropositionSatisfaction
: ProviderProvidedProposition
: RecipientReceivedProposition
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
Transporter : Party
businessNetwork = XTrailerBusiness
performedWork = Submit Idea, Absorb Innovation, Buy Product, Operate Product
providedProposition = PropositionFromMarket
receivedProposition = XTrailerProposition
: BusinessNetworkParty
: BusinessNetworkParty
XTrailerBusiness : BusinessNetwork
activity = Operate Product, Buy Product, Absorb Innovation, Submit Idea, Manage Innovation, Manage Fulfillment
businessItem = Innovation
containingModel = MyCompanyModel
flow = InnovationFlow, OrderFlow, ProductFlow, IdeaFlow, OrderFromStoreFlow
ownedAssignment = ManufacturerAssignment
party = Manufacturer, Transporter
scenario = OverallScenario

Figure 93. Value proposition provided and received (Objects)

In the XTrailer use case we assume that the Transporter cares about the following four values:

- Fair Price. This relates to the price that transporters has to pay for a trailer. This value is measured based on the measure of product price.
- Market driven design. This is about the extent to which innovation (design and development of trailers) is driven by feedback from the transporters. This value is measured based on the measure of idea productization, i.e. the ratio of ideas that make it to new products, relative to all ideas that have been received.
- Fast innovation. This is about the time it takes to translate transporter's feedback into new trailers that are available on the market. This value is measured based on the measure of innovation lead time, which is the time ideas reside in the ideas store(s), plus the duration of Manage Idea activity, plus the duration of the Release Management method.
- Late spec freeze. This is about how flexible a transporter is in making changes to the specification of trailers that are ordered. The shorter time it takes to actually produce a trailer, the more flexible the transporter is in making changes to the order, as the specification of the order is "frozen" at the moment the production starts. Hence this value is measured based on the measure of product lead time, which is the duration of the Build Product activity, plus the time a trailer resides in the Products store, plus the duration of the Deliver Product activity.

As indicated in the model view mockup in Figure 94, the value proposition "XTrailer Proposition" consists of four components, one for each of these four values.

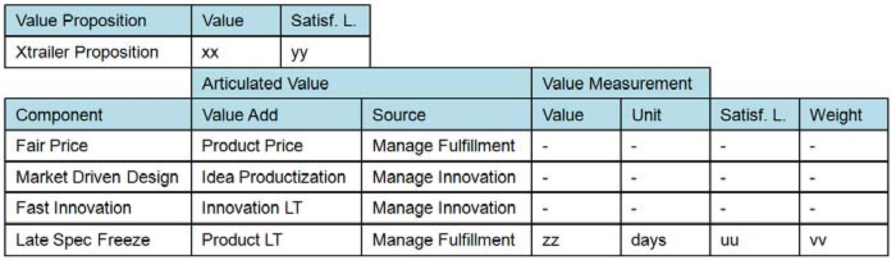


Satisfaction level: Projection on “pie” sectors
Value measurement: “Needle” position

Figure 94. Value proposition components mockup

Note that, though this mockup is more indicative than “actual”, it suggests that an appropriate view on a value proposition may also have the quality of “dashboard”, with drill down to underlying details. Measurement of the “underlying” value, as well as of the level of satisfaction of the recipient with that value, will be analyzed in detail below.

The mockup diagram in Figure 95 provides a more complete representation of the value proposition in tabular form. We will refer to this diagram several times below.



Value Proposition	Value	Satisf. L.
Xtrailer Proposition	xx	yy

	Articulated Value		Value Measurement			
Component	Value Add	Source	Value	Unit	Satisf. L.	Weight
Fair Price	Product Price	Manage Fulfillment	-	-	-	-
Market Driven Design	Idea Productization	Manage Innovation	-	-	-	-
Fast Innovation	Innovation LT	Manage Innovation	-	-	-	-
Late Spec Freeze	Product LT	Manage Fulfillment	zz	days	uu	vv

Figure 95. Value proposition structure mockup

The objects in the object diagram in Figure 96 define the structure of the value proposition in the mockup in Figure 95: the value proposition consisting of the four components, each of which relates to a particular value (Value Add object).

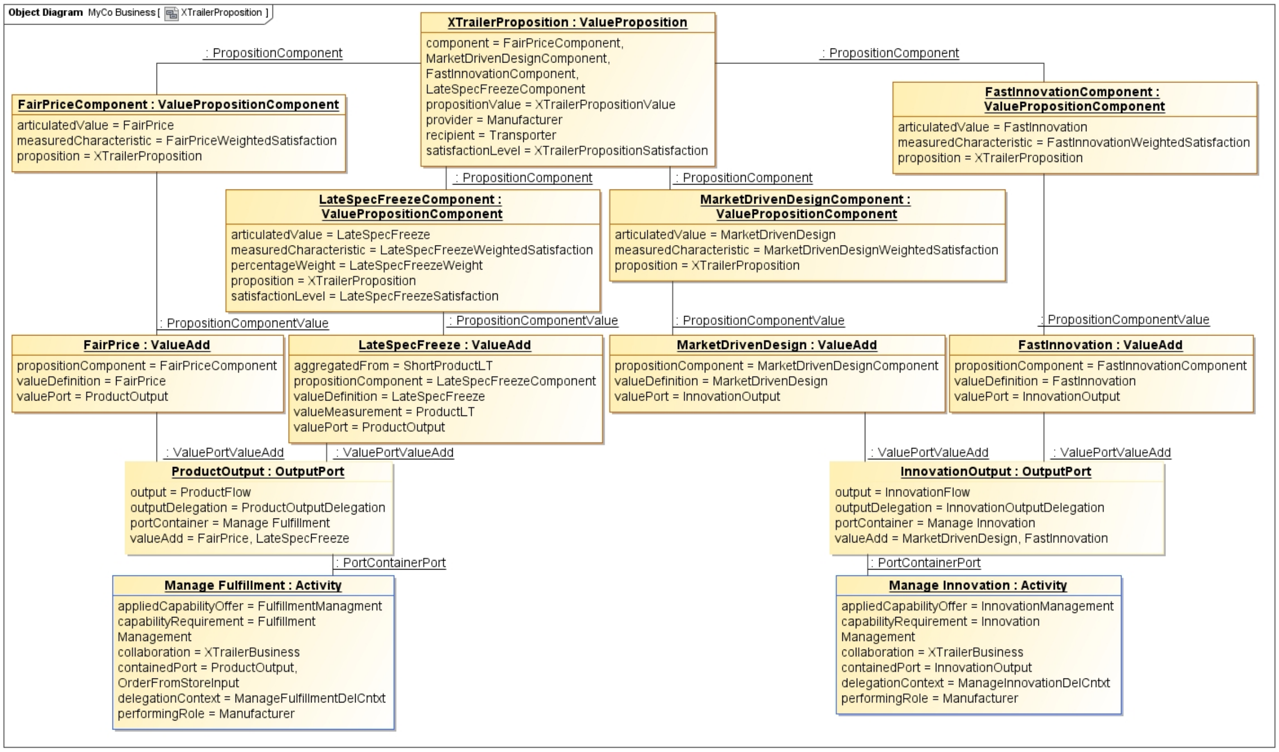

Object Diagram MyCo Business [XTrailerProposition]
XTrailerProposition : ValueProposition
component = FairPriceComponent, MarketDrivenDesignComponent, FastInnovationComponent, LateSpecFreezeComponent
propositionValue = XTrailerPropositionValue
provider = Manufacturer
recipient = Transporter
satisfactionLevel = XTrailerPropositionSatisfaction
: PropositionComponent
FairPriceComponent : ValuePropositionComponent
articulatedValue = FairPrice
measuredCharacteristic = FairPriceWeightedSatisfaction
proposition = XTrailerProposition
FastInnovationComponent : ValuePropositionComponent
articulatedValue = FastInnovation
measuredCharacteristic = FastInnovationWeightedSatisfaction
proposition = XTrailerProposition
LateSpecFreezeComponent : ValuePropositionComponent
articulatedValue = LateSpecFreeze
measuredCharacteristic = LateSpecFreezeWeightedSatisfaction
percentageWeight = LateSpecFreezeWeight
proposition = XTrailerProposition
satisfactionLevel = LateSpecFreezeSatisfaction
MarketDrivenDesignComponent : ValuePropositionComponent
articulatedValue = MarketDrivenDesign
measuredCharacteristic = MarketDrivenDesignWeightedSatisfaction
proposition = XTrailerProposition
: PropositionComponentValue
FairPrice : ValueAdd
propositionComponent = FairPriceComponent
valueDefinition = FairPrice
valuePort = ProductOutput
LateSpecFreeze : ValueAdd
aggregatedFrom = ShortProductLT
propositionComponent = LateSpecFreezeComponent
valueDefinition = LateSpecFreeze
valueMeasurement = ProductLT
valuePort = ProductOutput
MarketDrivenDesign : ValueAdd
propositionComponent = MarketDrivenDesignComponent
valueDefinition = MarketDrivenDesign
valuePort = InnovationOutput
FastInnovation : ValueAdd
propositionComponent = FastInnovationComponent
valueDefinition = FastInnovation
valuePort = InnovationOutput
: ValuePortValueAdd
ProductOutput : OutputPort
output = ProductFlow
outputDelegation = ProductOutputDelegation
portContainer = Manage Fulfillment
valueAdd = FairPrice, LateSpecFreeze
InnovationOutput : OutputPort
output = InnovationFlow
outputDelegation = InnovationOutputDelegation
portContainer = Manage Innovation
valueAdd = MarketDrivenDesign, FastInnovation
: PortContainerPort
Manage Fulfillment : Activity
appliedCapabilityOffer = FulfillmentManagment
capabilityRequirement = Fulfillment Management
collaboration = XTrailerBusiness
containedPort = ProductOutput, OrderFromStoreInput
delegationContext = ManageFulfillmentDelCntxt
performingRole = Manufacturer
Manage Innovation : Activity
appliedCapabilityOffer = InnovationManagement
capabilityRequirement = Innovation Management
collaboration = XTrailerBusiness
containedPort = InnovationOutput
delegationContext = ManageInnovationDelCntxt
performingRole = Manufacturer

Figure 96. Value proposition structure (objects)

The Manage Fulfillment activity contributes to two values: Fair Price and Late Spec Freeze. These are defined as Value Add objects, associated with the Product output of the activity. This means that these values are conveyed by the Product flow from the manufacturer (who performs this business network activity) to the transporter. Similarly, the Manage Innovation activity has the following two Value Add objects associated with its output port: Market Driven Design and Fast Innovation. These will be conveyed to the transporter by the Innovation flow in the business network.

Objects in the object diagram in Figure 96, are instances of classes in the meta-model diagram in Figure 97.

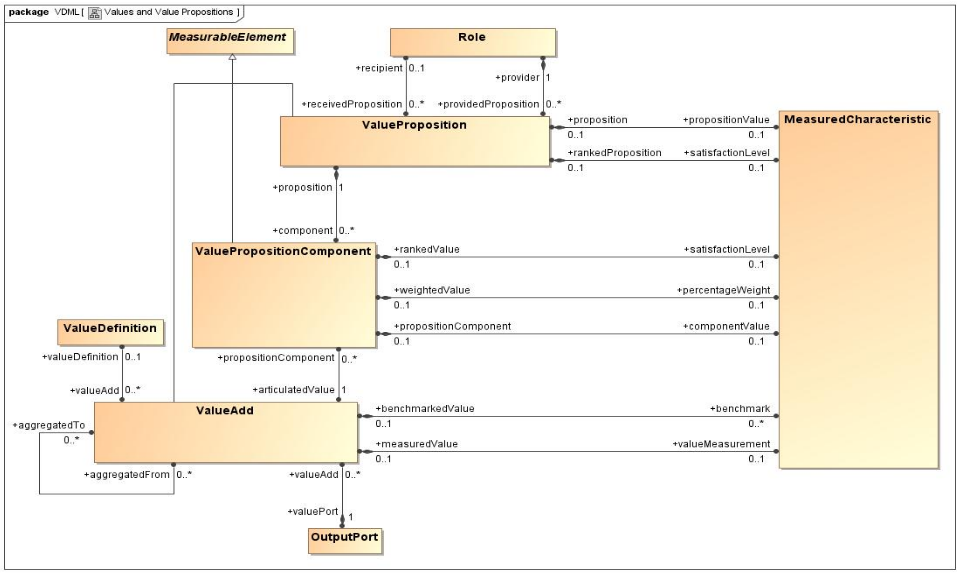


Figure 97. Values and Value Propositions metamodel

The various associations to Measured Characteristic, as well as the association between Value Add and Value Definition, will be discussed below.

The discussion of activity-related metamodel concepts has been completed with the discussion of Value Add, it might be useful to have a short reflection on the concept of Activity again. Figure 98 provides an informal and schematic representation of an activity, as it is defined in VDML. Most of the activity-related elements have been analyzed in earlier sections. The related Value Add object has just been introduced. As VDML also addresses activity modeling, and activity network modeling, the notations of which having some flavor of "process modeling", it is sometimes asked whether VDML is unnecessarily overlapping process modeling language such as specified in BPMN (2011). From the activity-related concepts, as shown in Figure 98, it is clear that, though both languages, VDML and BPMN, include the concept of Activity, their viewpoints are very different. Consider also other activity-related concepts in VDML, defining e.g. what capability an activity requires, and what capability offer is applied via it, and it is even more clear how different and complementary both viewpoints are. There is some overlap, but this overlap is rather useful, as it ensures alignment and integration between the two viewpoints (as will be discussed later).

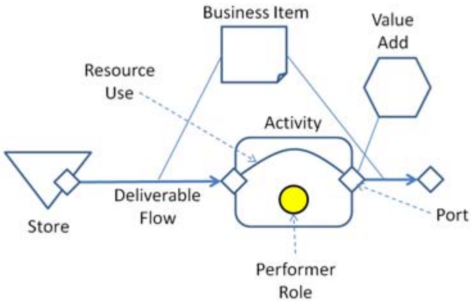
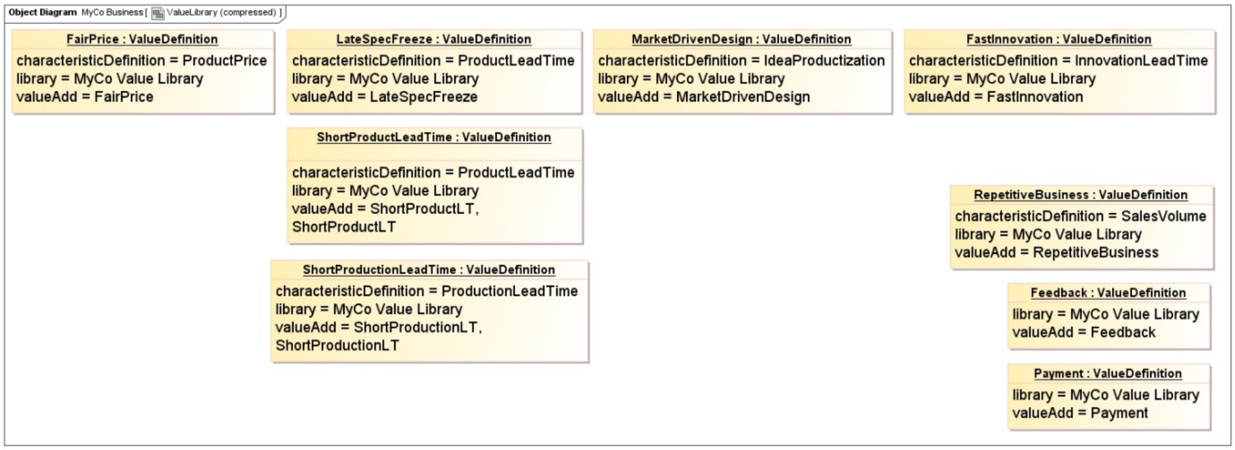


Figure 98. Schematic representation of activity

As the details of the Value Add objects in the object diagram in Figure 96 suggest, per value add a value definition is defined, which "types" the value add. This instantiates the association between the Value Add class and the Value Definition class in the metamodel diagram in Figure 97.

Value definitions are maintained in a value library. The objects in the object diagram in Figure 99 represent the content of the value library in the XTrailer use case.



Object Diagram MyCo Business [ValueLibrary (compressed)]

FairPrice : ValueDefinition
characteristicDefinition = ProductPrice
library = MyCo Value Library
valueAdd = FairPrice

LateSpecFreeze : ValueDefinition
characteristicDefinition = ProductLeadTime
library = MyCo Value Library
valueAdd = LateSpecFreeze

MarketDrivenDesign : ValueDefinition
characteristicDefinition = IdeaProductization
library = MyCo Value Library
valueAdd = MarketDrivenDesign

FastInnovation : ValueDefinition
characteristicDefinition = InnovationLeadTime
library = MyCo Value Library
valueAdd = FastInnovation

ShortProductLeadTime : ValueDefinition
characteristicDefinition = ProductLeadTime
library = MyCo Value Library
valueAdd = ShortProductLT,
ShortProductLT

ShortProductionLeadTime : ValueDefinition
characteristicDefinition = ProductionLeadTime
library = MyCo Value Library
valueAdd = ShortProductionLT,
ShortProductionLT

RepetitiveBusiness : ValueDefinition
characteristicDefinition = SalesVolume
library = MyCo Value Library
valueAdd = RepetitiveBusiness

Feedback : ValueDefinition
library = MyCo Value Library
valueAdd = Feedback

Payment : ValueDefinition
library = MyCo Value Library
valueAdd = Payment

Figure 99. Value library (objects)

A value library contains a taxonomy of values, consisting of value definitions and categories of them, and is applied to enforce consistency in the definition of value adds. Multiple value adds that are associated with the same value definition, are considered similar from the perspective of the library. As can be observed from the objects' details in the object diagram in Figure 99, the value definitions "Short Production Lead Time" and "Short Product Lead Time" are referenced from multiple value add objects.

The objects are instances of classes in the Value Libraries metamodel diagram in Figure 100.

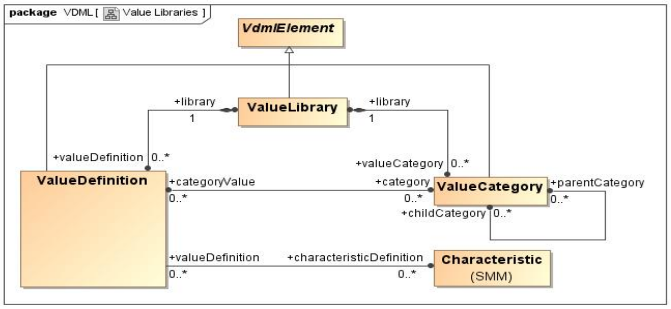


Figure 100. Value Libraries metamodel

Note that value categories have not been applied in the XTrailer use case.

We will now demonstrate, based on the XTrailer use case, how value is created by activities, accumulated over deliverable flows, from activity to activity, possibly via stores, as well as over levels of delegation, until the points where a value proposition is defined to articulate the values, and where the accumulated values are conveyed to the recipient of the value proposition. Figure 101 provides a (simplified) schematic representation of the concept of value contribution.

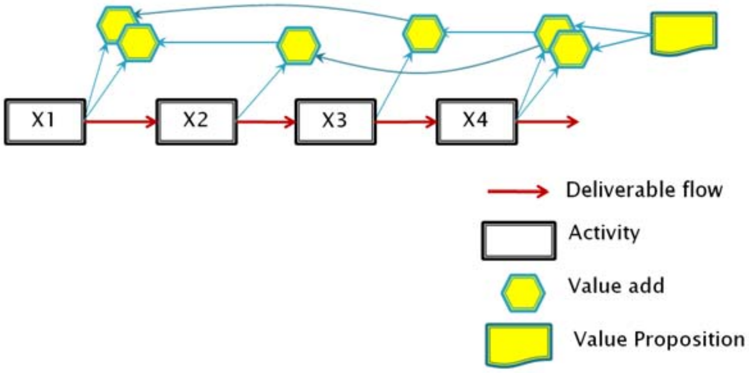


Figure 101. Value contribution

Note that the accumulation of value, over a "stream" of activities, as indicated in Figure 101, might be denoted as "value stream", though this is not a normative term in VDML. Generally known concepts, such as value chains, value streams and, particularly, value networks, can be applied as views on parts of value delivery models, as specified by VDML. VDML can support such views, but does not specify them normatively.

In this section we will deal with measures and measurements. A measure is "*a method that is applied to characterize an attribute of something by assigning a comparable quantification or qualification*". Attributes that are characterized through measures are called "measured characteristics". Many elements in VDML, such as activities, stores, ports, resource use objects, value add objects, value proposition components and value propositions are so-called measurable elements. Measurable elements can have measured characteristics, some of which are defined in the meta-model, and others can be defined as custom measured characteristics by the business analyst. A measurement is "*the result of applying a measure*". Measurements are associated with measured characteristics. A measured characteristic is "typed" by a characteristic definition in a measure library. The reader might again refer to the earlier discussed VDML Elements metamodel diagram in Figure 61, which shows some of these concepts and their relationships. Characteristics that are used in a measure library in the XTrailer use case are presented in the object diagram in Figure 62, which has been considered earlier as well. Figure 102 represents a metamodel diagram that shows some of the main concepts in SMM, and how they are related.

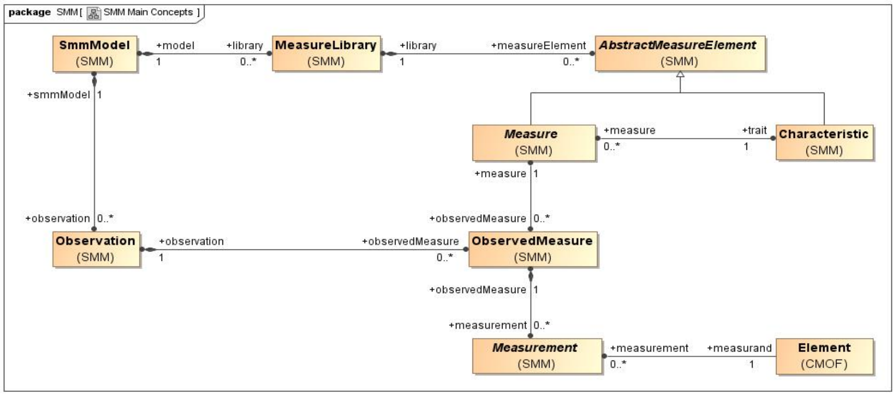


Figure 102. SMM main concepts metamodel

This metamodel diagram is abstracted from various details. For a detailed overview of the SMM metamodel, the reader should refer to SMM (2012). As the diagram shows, an SMM model may contain measure libraries, as well as observations. A measure library contains measures and characterstics, which serve as “traits” of the measures. A measure library does contain more detail than exposed in Figure 102, such as measure categories, but these can be looked up in SMM (2012). An observation contains “observed measures”, being applications of measures to establish measurements. Measures are contained in measure libraries, whereas measurements are contained in observations.

As the earlier discussed Scenario and Analysis Context metamodel diagram in Figure 51 indicates, an analysis context, and so any scenario and any delegation context may have one observation. This is the observation that contains the measurements that are specific to that context. Measurements that are not specific to a particular scenario, are assumed to be contained in the “default scenario”, which is defined in the same metamodel diagram. An observation can contain measurements for multiple measured characteristics. When a measured characteristic is associated with multiple measurements, these measurements are contained in different observations. When for instance a capability method is analyzed in a particular delegation context, the corresponding context observation provides the measurements for measured characteristics of the capability method and its contained measurable elements.

This SMM main concepts metamodel diagram in Figure 102, in combination with the earlier presented metamodel diagrams of VDML Elements in Figure 61 and Scenario and Analysis Context in Figure 51, provide a complete overview of the integration between VDML and SMM.

We will now demonstrate how measured characteristics of the value proposition (“XTrailer Proposition”) and its components can be measured, based on measurements of the underlying value add elements, and how these measurements can be derived from measurements of measurable elements throughout the business system in the XTrailer use case. As discussed earlier, analysis is context-based, whereby the context is defined by a scenario, and its tree of delegation contexts for the various underlying sub-collaborations and stores. Figure 103 shows again the model view mockup of the context tree in the XTrailer use case, but now with just a subset of it highlighted. In order to keep the model content as small as possible, and as large as just required to demonstrate essential concepts, we will only discuss measurements that are contained in the observations of “Overall Scenario”, and its sub-contexts “Manage Fulfillment Delegation Context” and “Manage Production Delegation Context”. And in relation to “Overall Scenario”, we will only consider measurements in relation to the fulfillment part of the business network, and exclude innovation relation measurements.

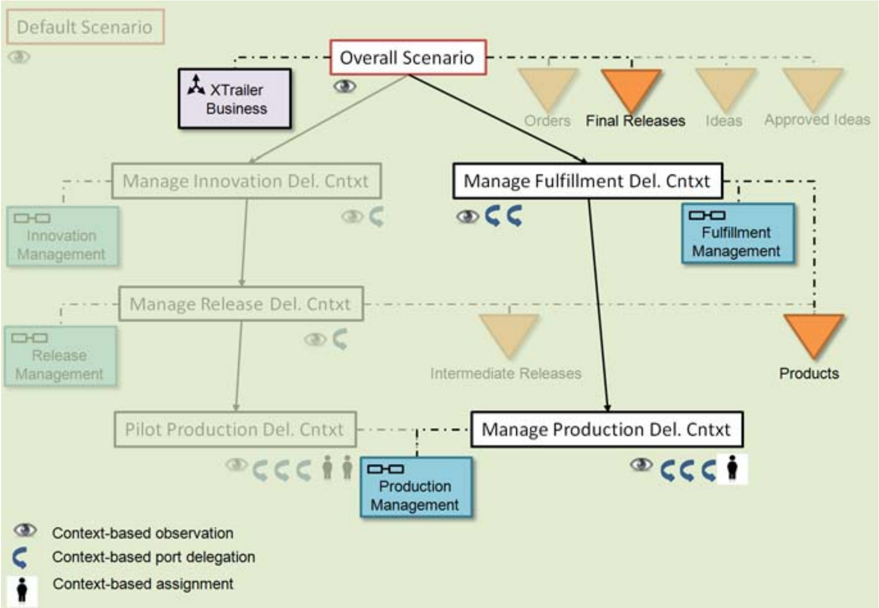


Figure 103. Context tree mockup: Fulfillment part partially highlighted

Figure 104 contains a model view mockup of a representative set of measurements that would all be relevant in the context of analysis of the XTrailer business network, its profit to the MyCompany, and the XTrailer Proposition as provided by its Manufacturer party role. These measurements together represent the measurement scope of the "Overall Scenario".

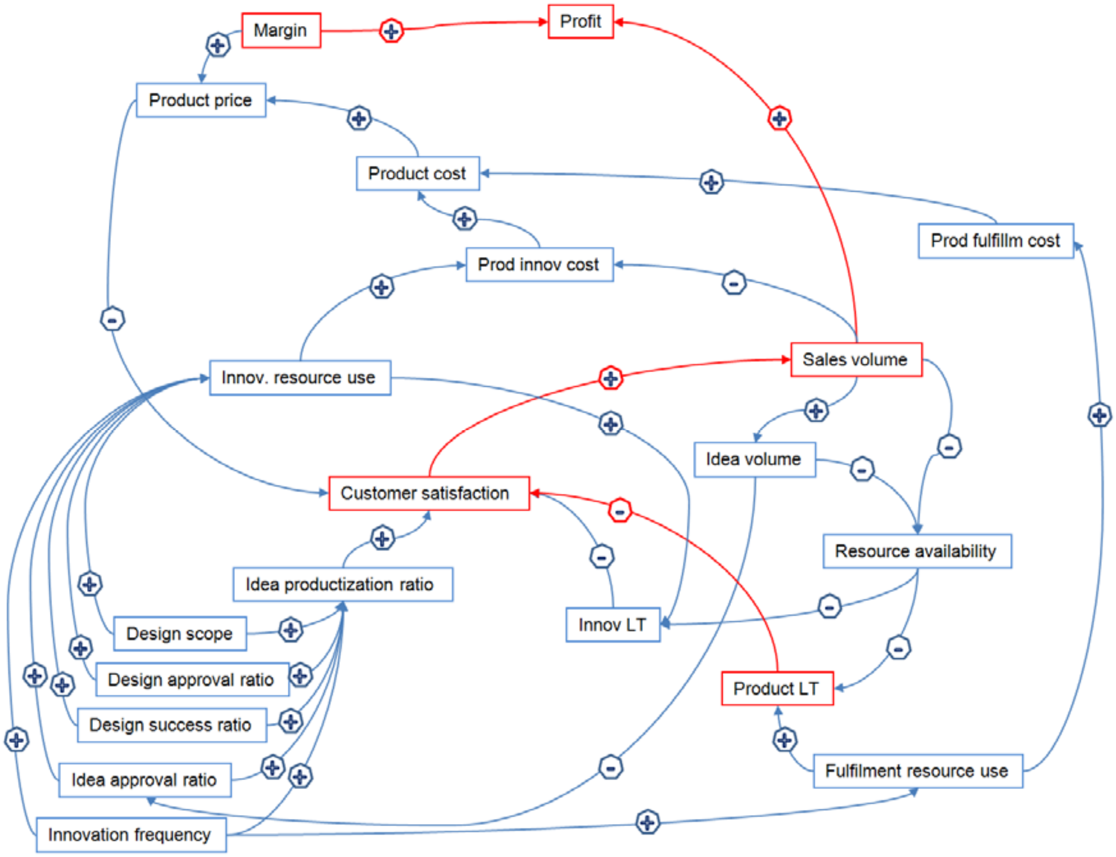


Figure 104. Measurements and measurement influences mockup

A subset of the measurements contained in the observations of the highlighted parts in the context tree in Figure 103 are highlighted in the mockup in Figure 104. Analysis in the remainder of this section will just focus on this highlighted subset of measurements. Note that the measurements that will actually be modeled in the use case are a bit more refined than the ones that are presented in the mockup.

As the mockup in Figure 104 suggests, measurements can be dependent on each other. Measurements can influence other measurements. Some influences are positive, others are negative. A positive influence indicates that increase of the value of one measurement leads to increase of the value of another. A negative influence indicates that increase of the value of one measurement leads to decrease of the value of another. For instance:

- Increase of Product LT (product lead time) leads to decrease of Customer Satisfaction
- Increase of Customer satisfaction leads to increase of Sales Volume, and hence a decrease of it leads to a decrease of Sales Volume.
- An increase of Sales Volume leads to an increase of Profit, though Profit is also positively influenced by Margin.
- An increase of Margin also increases Product Price
- Increase of Product Price leads to decrease of Customer Satisfaction

Note that the “value” of a measurement should not be confused with the concept of “value” in the sense of Value Add and Value Proposition. The “value” of a measurement is just a quantity specified, based on a measure, and against a unit of measure (see SMM (2012) for the detailed metamodel of SMM).

As will be discussed below, SMM supports measurement dependencies in various ways. Actually the indication of “structured” in the name “Structured Metrics Metamodel” (SMM), particularly relates to the fact that SMM specifies in detail how measures can depend on other measures, and how measurements can be calculated from other measurements accordingly.

As can be noticed from the mockup in Figure 104, measurement dependencies can easily lead, and in this use case does indeed lead, to circular dependencies, directly or indirectly. This implies that when measurements are quantified, and calculations take place, iterations will be required to achieve an equilibrium state in which all measurements have a value that is consistent with each other. Spreadsheet calculation engines do generally support iterative calculation. Supporting such calculations is in fact a basic form of model simulation.

Note also that the mockup in Figure 104 provides a convenient way to represent all measurement information, underlying a scenario, in a single diagram, from where the business analyst can drill down to related details, and understand the context of measurements in more detail. As the notation in the mockup is also rather consistent with causal-loop diagrams in System Dynamics (see e.g. Dooley (2002)), it might also provide an abstraction of underlying value delivery modeling data, which may serve as input for System Dynamics simulation.

In the remainder of this section we will discuss the modeling of measures and measurements, including dependencies between them, but will not discuss the actual simulation of the model, and will not consider any specific calculation results.

We will analyze and discuss the following in particular, in the remainder of this section:

- Measurement of Late Spec Freeze value, as discussed earlier, related to the Late Spec Freeze component in the XTrailer Proposition.
- Measurement of customer (i.e. transporter) satisfaction with the Late Spec Freeze value, as well as its relative weight to the customer.
- Measurement of overall XTrailer Proposition value, given its components and relative weights
- Measurement of customer satisfaction with the XTrailer Proposition.
- Measurement of the impact of customer satisfaction with the XTrailer Proposition on the sales volume of trailers in the XTrailer business network.
- Measurement of profit of the Manufacturer party in the business network.

The measures that will be used to support these measurements are represented by the objects in the object diagrams in Figure 105 and Figure 106. Together with their related characteristics, as earlier represented in the object diagram in Figure 62, they form the measure library content that is used in XTrailer use case.

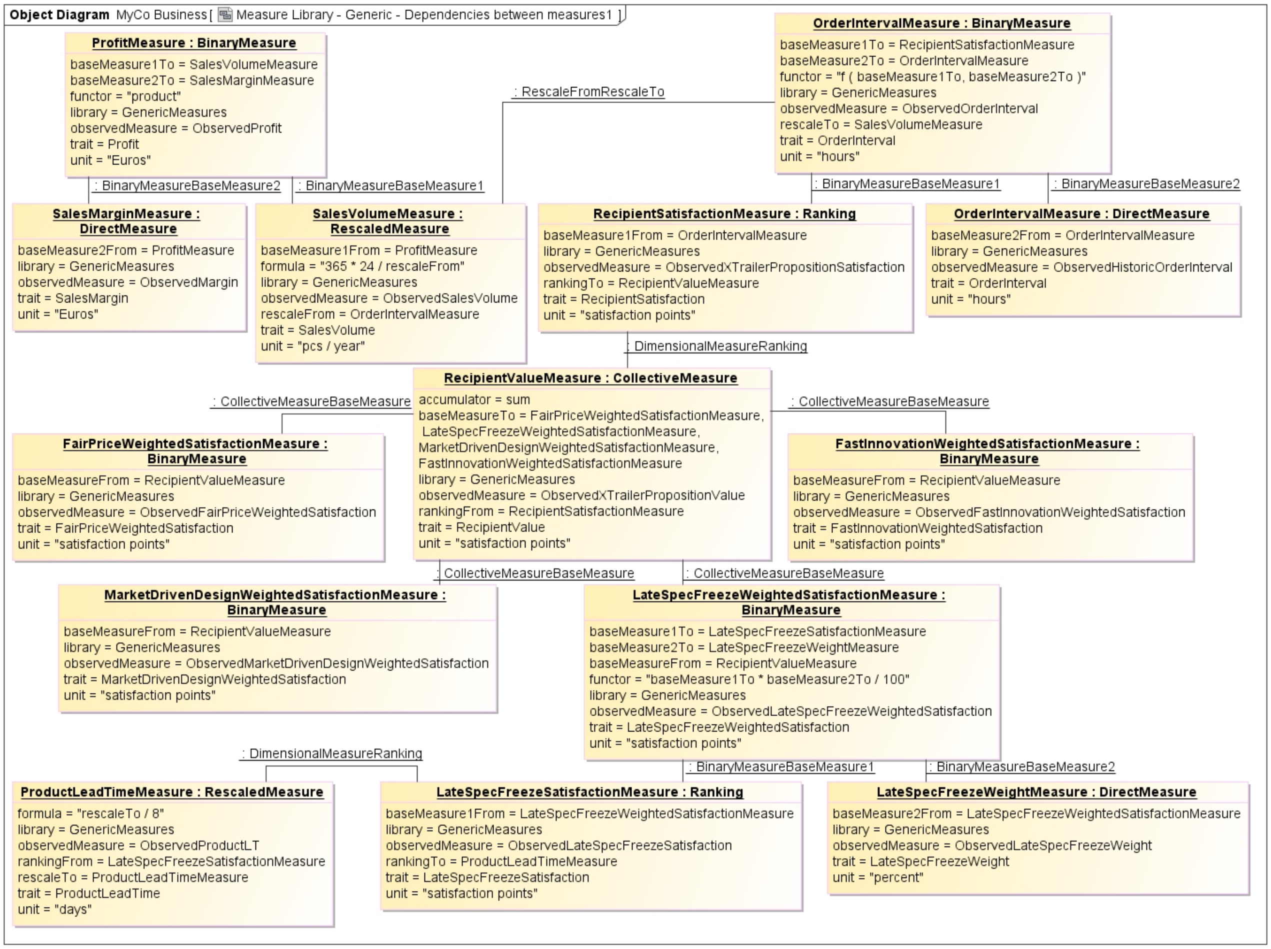

Object Diagram MyCo Business[Measure Library - Generic - Dependencies between measures1]
ProfitMeasure : BinaryMeasure
baseMeasure1To = SalesVolumeMeasure
baseMeasure2To = SalesMarginMeasure
functor = "product"
library = GenericMeasures
observedMeasure = ObservedProfit
trait = Profit
unit = "Euros"
OrderIntervalMeasure : BinaryMeasure
baseMeasure1To = RecipientSatisfactionMeasure
baseMeasure2To = OrderIntervalMeasure
functor = "f (baseMeasure1To, baseMeasure2To)"
library = GenericMeasures
observedMeasure = ObservedOrderInterval
rescaleTo = SalesVolumeMeasure
trait = OrderInterval
unit = "hours"
: RescaleFromRescaleTo
: BinaryMeasureBaseMeasure2
: BinaryMeasureBaseMeasure1
: BinaryMeasureBaseMeasure1
: BinaryMeasureBaseMeasure2
SalesMarginMeasure : DirectMeasure
baseMeasure2From = ProfitMeasure
library = GenericMeasures
observedMeasure = ObservedMargin
trait = SalesMargin
unit = "Euros"
SalesVolumeMeasure : RescaledMeasure
baseMeasure1From = ProfitMeasure
formula = "365 * 24 / rescaleFrom"
library = GenericMeasures
observedMeasure = ObservedSalesVolume
rescaleFrom = OrderIntervalMeasure
trait = SalesVolume
unit = "pcs / year"
RecipientSatisfactionMeasure : Ranking
baseMeasure1From = OrderIntervalMeasure
library = GenericMeasures
observedMeasure = ObservedXTrailerPropositionSatisfaction
rankingTo = RecipientValueMeasure
trait = RecipientSatisfaction
unit = "satisfaction points"
OrderIntervalMeasure : DirectMeasure
baseMeasure2From = OrderIntervalMeasure
library = GenericMeasures
observedMeasure = ObservedHistoricOrderInterval
trait = OrderInterval
unit = "hours"
: DimensionalMeasureRanking
RecipientValueMeasure : CollectiveMeasure
accumulator = sum
baseMeasureTo = FairPriceWeightedSatisfactionMeasure,
LateSpecFreezeWeightedSatisfactionMeasure,
MarketDrivenDesignWeightedSatisfactionMeasure,
FastInnovationWeightedSatisfactionMeasure
library = GenericMeasures
observedMeasure = ObservedXTrailerPropositionValue
rankingFrom = RecipientSatisfactionMeasure
trait = RecipientValue
unit = "satisfaction points"
: CollectiveMeasureBaseMeasure
: CollectiveMeasureBaseMeasure
FairPriceWeightedSatisfactionMeasure : BinaryMeasure
baseMeasureFrom = RecipientValueMeasure
library = GenericMeasures
observedMeasure = ObservedFairPriceWeightedSatisfaction
trait = FairPriceWeightedSatisfaction
unit = "satisfaction points"
FastInnovationWeightedSatisfactionMeasure : BinaryMeasure
baseMeasureFrom = RecipientValueMeasure
library = GenericMeasures
observedMeasure = ObservedFastInnovationWeightedSatisfaction
trait = FastInnovationWeightedSatisfaction
unit = "satisfaction points"
: CollectiveMeasureBaseMeasure
: CollectiveMeasureBaseMeasure
MarketDrivenDesignWeightedSatisfactionMeasure : BinaryMeasure
baseMeasureFrom = RecipientValueMeasure
library = GenericMeasures
observedMeasure = ObservedMarketDrivenDesignWeightedSatisfaction
trait = MarketDrivenDesignWeightedSatisfaction
unit = "satisfaction points"
LateSpecFreezeWeightedSatisfactionMeasure : BinaryMeasure
baseMeasure1To = LateSpecFreezeSatisfactionMeasure
baseMeasure2To = LateSpecFreezeWeightMeasure
baseMeasureFrom = RecipientValueMeasure
functor = "baseMeasure1To * baseMeasure2To / 100"
library = GenericMeasures
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction
trait = LateSpecFreezeWeightedSatisfaction
unit = "satisfaction points"
: DimensionalMeasureRanking
: BinaryMeasureBaseMeasure1
: BinaryMeasureBaseMeasure2
ProductLeadTimeMeasure : RescaledMeasure
formula = "rescaleTo / 8"
library = GenericMeasures
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasure
rescaleTo = ProductLeadTimeMeasure
trait = ProductLeadTime
unit = "days"
LateSpecFreezeSatisfactionMeasure : Ranking
baseMeasure1From = LateSpecFreezeWeightedSatisfactionMeasure
library = GenericMeasures
observedMeasure = ObservedLateSpecFreezeSatisfaction
rankingTo = ProductLeadTimeMeasure
trait = LateSpecFreezeSatisfaction
unit = "satisfaction points"
LateSpecFreezeWeightMeasure : DirectMeasure
baseMeasure2From = LateSpecFreezeWeightedSatisfactionMeasure
library = GenericMeasures
observedMeasure = ObservedLateSpecFreezeWeight
trait = LateSpecFreezeWeight
unit = "percent"

Figure 105. Measure Library objects (Measures, part-1)

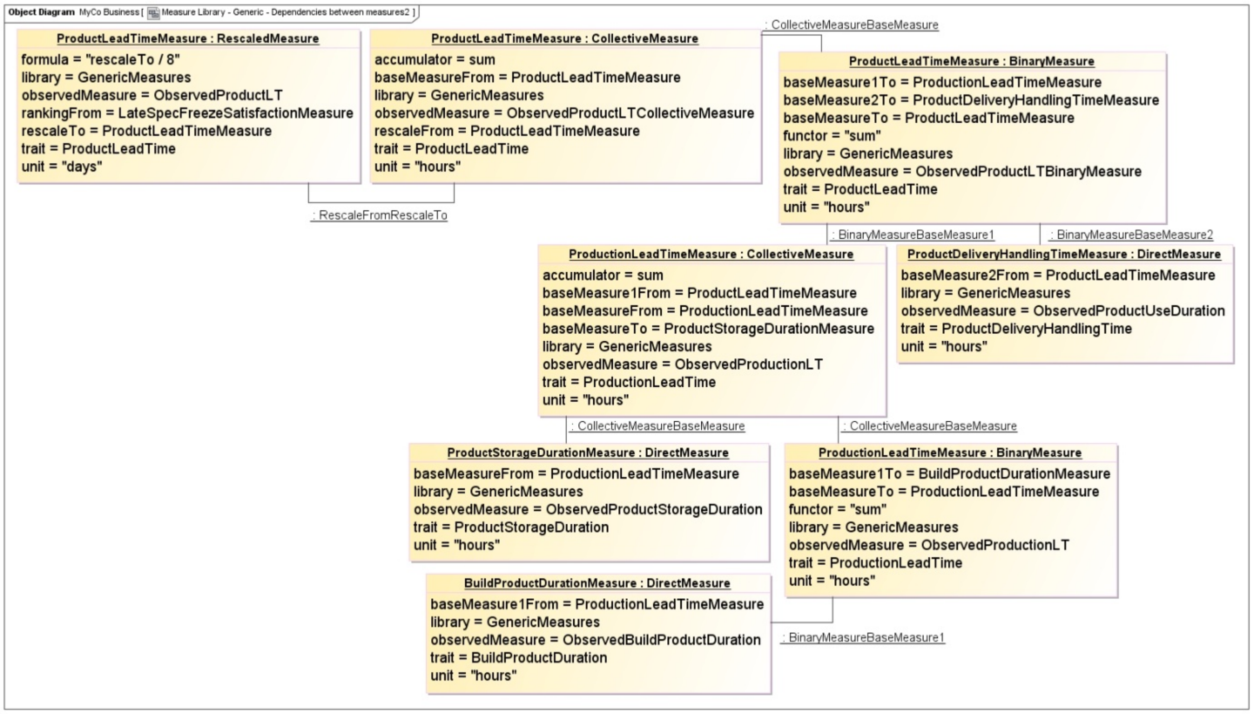

Object Diagram MyCo Business [Measure Library - Generic - Dependencies between measures2]
ProductLeadTimeMeasure : RescaledMeasure
formula = "rescaleTo / 8"
library = GenericMeasures
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasure
rescaleTo = ProductLeadTimeMeasure
trait = ProductLeadTime
unit = "days"
ProductLeadTimeMeasure : CollectiveMeasure
accumulator = sum
baseMeasureFrom = ProductLeadTimeMeasure
library = GenericMeasures
observedMeasure = ObservedProductLTCollectiveMeasure
rescaleFrom = ProductLeadTimeMeasure
trait = ProductLeadTime
unit = "hours"
: CollectiveMeasureBaseMeasure
ProductLeadTimeMeasure : BinaryMeasure
baseMeasure1To = ProductionLeadTimeMeasure
baseMeasure2To = ProductDeliveryHandlingTimeMeasure
baseMeasureTo = ProductLeadTimeMeasure
functor = "sum"
library = GenericMeasures
observedMeasure = ObservedProductLTBinaryMeasure
trait = ProductLeadTime
unit = "hours"
: RescaleFromRescaleTo
: BinaryMeasureBaseMeasure1
: BinaryMeasureBaseMeasure2
ProductionLeadTimeMeasure : CollectiveMeasure
accumulator = sum
baseMeasure1From = ProductLeadTimeMeasure
baseMeasureFrom = ProductionLeadTimeMeasure
baseMeasureTo = ProductStorageDurationMeasure
library = GenericMeasures
observedMeasure = ObservedProductionLT
trait = ProductionLeadTime
unit = "hours"
ProductDeliveryHandlingTimeMeasure : DirectMeasure
baseMeasure2From = ProductLeadTimeMeasure
library = GenericMeasures
observedMeasure = ObservedProductUseDuration
trait = ProductDeliveryHandlingTime
unit = "hours"
: CollectiveMeasureBaseMeasure
: CollectiveMeasureBaseMeasure
ProductStorageDurationMeasure : DirectMeasure
baseMeasureFrom = ProductionLeadTimeMeasure
library = GenericMeasures
observedMeasure = ObservedProductStorageDuration
trait = ProductStorageDuration
unit = "hours"
ProductionLeadTimeMeasure : BinaryMeasure
baseMeasure1To = BuildProductDurationMeasure
baseMeasureTo = ProductionLeadTimeMeasure
functor = "sum"
library = GenericMeasures
observedMeasure = ObservedProductionLT
trait = ProductionLeadTime
unit = "hours"
BuildProductDurationMeasure : DirectMeasure
baseMeasure1From = ProductionLeadTimeMeasure
library = GenericMeasures
observedMeasure = ObservedBuildProductDuration
trait = BuildProductDuration
unit = "hours"
: BinaryMeasureBaseMeasure1

Figure 106. Measure Library objects (Measures, part-2)

These objects are instances of the classes in the Measures metamodel diagram in Figure 107. Note that these classes are defined by SMM. SMM (2012) specifies the metamodel in detail. The diagrams below only shows the metamodel in part, and leaves out various details.

The SMM metamodel that is used here is an adaptation of the metamodel as is specified in SMM (2012). Relationships between classes are straight associations in Figure 107, whereas in SMM (2012) they involve relationship classes. This simplification has been applied here to avoid exposure of many objects that do not necessarily add to the understanding of the basic concepts.

As has been mentioned earlier, a revision of SMM is in progress. This revision addresses several improvements that are required in SMM, to facilitate integration with VDML, as well as to improve the maintainability and expressiveness of measures. **Error! Reference source not found.**) lists an overview of the most important changes that a forthcoming SMM revision will address, together with a short explanation of why these are required. The meta-model diagrams that we use in this document do incorporate just some of these changes, in so-far they are essential to demonstrate the application of SMM measures to the use case.

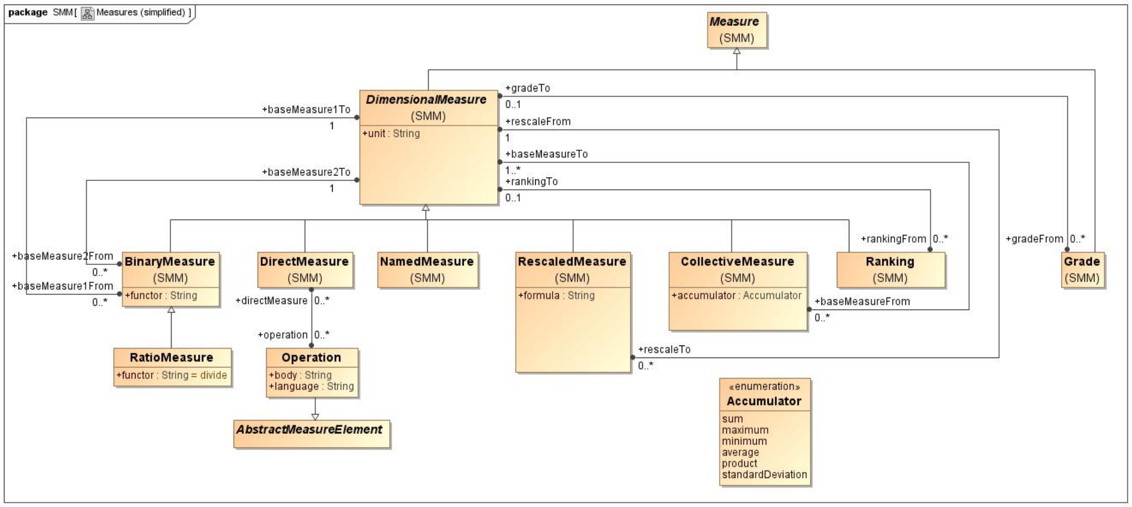


Figure 107. Measures metamodel (simplified and intermediate version)

The metamodel of measurements, being the application of measures, is structured similarly, and its diagram is provided in Figure 108.

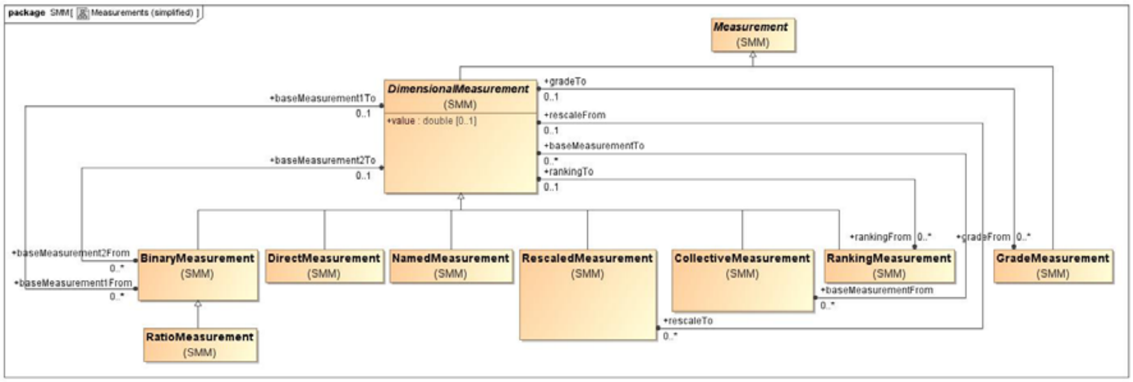


Figure 108. Measurements metamodel (simplified and intermediate version)

A dimensional measure, e.g. Product Lead Time, might be graded by a grade, being a non-dimensional measure (i.e. it does not have a unit of measure). Grade's are associated with intervals, not shown in the metamodel diagram in Figure 107, but part of the SMM meta-model in SMM (2012). For example, when Product Lead Time longer than 5 days, it may be graded as "inferior", whereas Product Lead Time shorter than 1 day maybe graded as "outstanding". The picture in Figure 109 suggests a graphical representation of such grading (see also Figure 94 above).

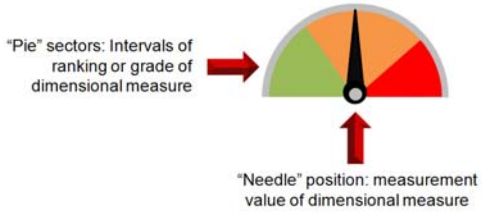

"Pie" sectors: Intervals of ranking or grade of dimensional measure
"Needle" position: measurement value of dimensional measure

Figure 109. Grading or ranking a dimensional measure

Though sometimes very useful, in the XTrailer use case example we will demonstrate the use of rankings, instead of grades (though both could have been applied in parallel). The difference is that ranking intervals are associated with values that are expressed against a unit of measure. A ranking is a dimensional measure, whereas a grade is not. The eye-catching indicator in the graphic in Figure 109 is the measurement of the underlying value. The level of the recipient's satisfaction with it is implied by the mapping on intervals ("pie" sectors). When rankings are applied, a more direct representation of the level of satisfaction would be possible, such as is suggested by the graphic in Figure 110, which provides a view on the value proposition components, as alternative to the one in Figure 94.

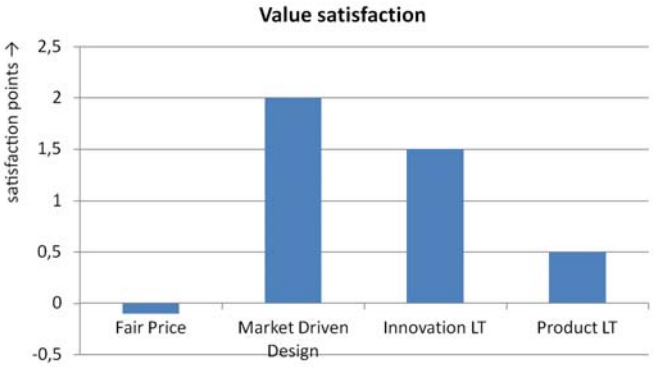


Figure 110. Satisfaction level as ranking of underlying value measurement

SMM version 1 (SMM (2012)) supports grades, but no rankings yet. As **Error! Reference source not found.**) indicates, rankings will be added as part of the revision of SMM. In the discussion in this section we anticipate on the support of rankings in SMM. The main advantage of a ranking is that it can itself be ranked or graded by other rankings or grades. This will be demonstrated below.

A "named measure" (see Figure 107), being a subtype of dimensional measure, is different from the other subtypes of measure, in that it does not specify in any structured way how the value of its measurements is constructed. After the revision to SMM has been applied, it might have a formal formula (string), but that would just be text that is interpretable by human readers. It is experienced in practice that many existing sources of measures, either in public or proprietary bodies of knowledge, do actually contain "informal" measures, i.e. measures that have a name, a description, some might have a formula, but not in machine-readable form, whereby such a formula might even relate to some operands that are not even available as measures, etc. Rigorous measurement, including automated aggregation of measurements, as well as, ultimately simulation, is not possible based on such measures. Such informal measures can be stored in SMM-based measure libraries as "named measures". When a "structured" measure is required as part of a value delivery model, a business analyst might select an "informal" measure of choice, and create one or more "structured" measures in another SMM-based measure library, whereby these measures refer back to the "informal" measure as source. Actual analysis will make use of the "structured" measures. In the remainder of this section we will only demonstrate the application of "structured" measures.

It is important to note the difference between direct measures and other dimensional measures, such as binary measures (including ratio measures), collective measures and rescaled measures. Binary measures are atomic, forming the leaves in a measure hierarchy. Measurements of direct measures are entered manually (e.g. estimates), or based on queries in a database, e.g. through query services. Binary measures, collective measures and rescaled measures are applying calculations to create aggregated or transformed measures, based on other dimensional measures. Binary measures do that based on two underlying dimensional measures (e.g. calculating their sum, ratio, difference or product), collective measurements based on a series of them (e.g. calculating their sum, product, maximum, average, etc.), and rescaled measures "rescale" or transform a single measure. All of these will be applied below.

We will now model the measurement of Late Spec Freeze value, related to the Late Spec Freeze component in the XTrailer Proposition (see Figure 94 and Figure 95 above), including how it is derived or aggregated from underlying measurements. Measurements of measured characteristics that are involved in this aggregation are define in relation to elements in the sub-scope of the XTrailer business system that is highlighted in the diagram in Figure 111. Product Lead Time will be constructed from Production Lead Time (duration of Build Product activity), duration of storage in the Products store, and the portion of lead time of the Deliver Product activity that is concerned with taking the product from the store and delivering it. This cumulative lead time will then be propagated further up the delegation stack, until the point where the Manage Fulfillment activity actually delivers the product to the customer, and even further up to the corresponding Late Spec Freeze component of the “XTrailer proposition”.

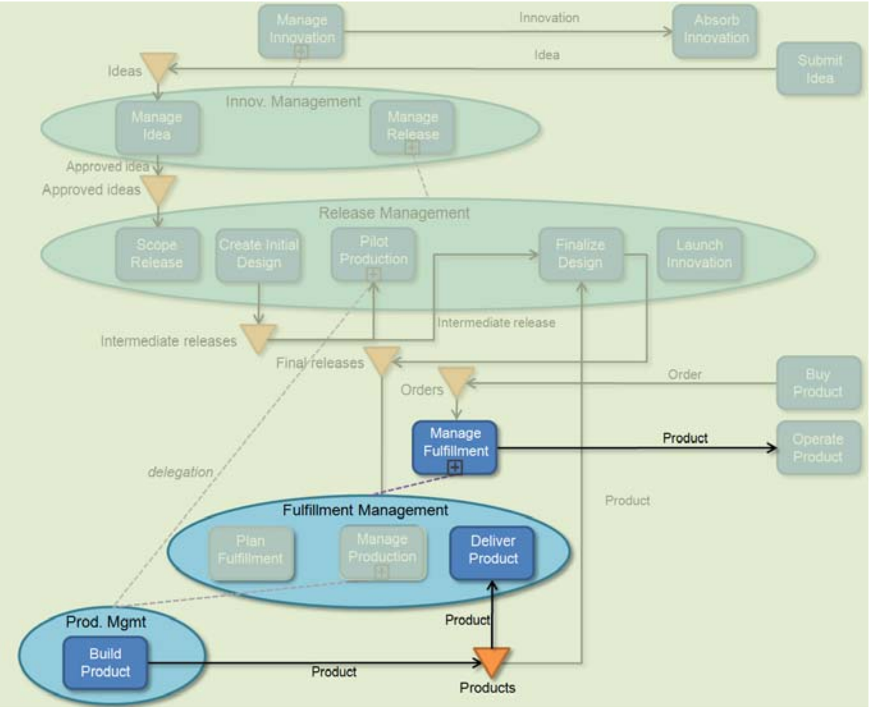


Figure 111. Use case overview: Basis for measurement of Late Spec Freeze value highlighted

Objects in the object diagram in Figure 112 show the bottom part of the measurement aggregation, starting with the duration of the Build Product activity. Note that we just consider a direct measurement for this duration, though we could have modeled even this in a more refined way, e.g. by making this duration itself dependent on availability of production capacity in the pools of the Trailer Plant, or even on certain custom defined measured characteristics of such pools, e.g. characteristics that express skill levels or certification levels in more detail. Note that custom measured characteristics of pools would also support refined cost measurements of activities, e.g. taking into account such pool characteristics as labor rates. This is all possible in real-world cases, but for sake of keeping the use case content small, we just focus on some highlights, selectively.

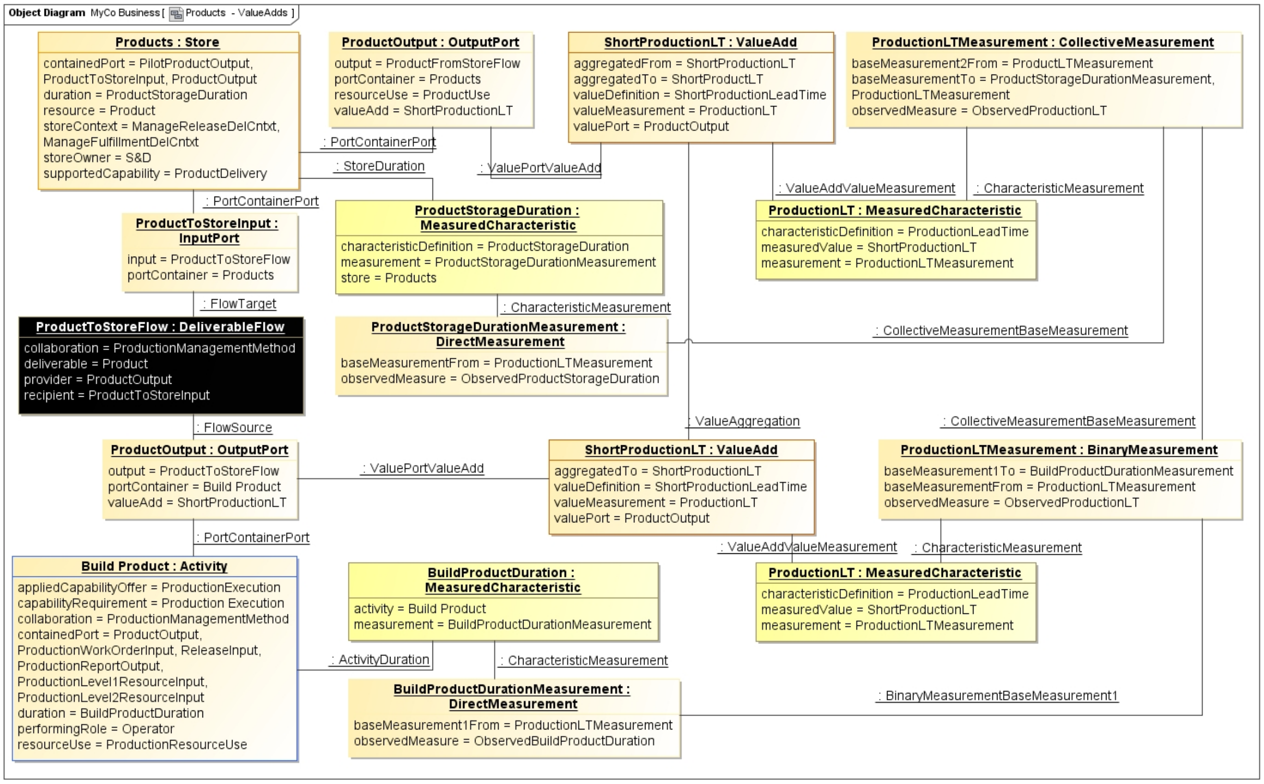

Object Diagram MyCo Business [Products - ValueAdds]
Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery
ProductOutput : OutputPort
output = ProductFromStoreFlow
portContainer = Products
resourceUse = ProductUse
valueAdd = ShortProductionLT
ShortProductionLT : ValueAdd
aggregatedFrom = ShortProductionLT
aggregatedTo = ShortProductLT
valueDefinition = ShortProductionLeadTime
valueMeasurement = ProductionLT
valuePort = ProductOutput
ProductionLTMeasurement : CollectiveMeasurement
baseMeasurement2From = ProductLTMeasurement
baseMeasurementTo = ProductStorageDurationMeasurement, ProductionLTMeasurement
observedMeasure = ObservedProductionLT
: PortContainerPort
: StoreDuration
: ValuePortValueAdd
: ValueAddValueMeasurement
: CharacteristicMeasurement
: PortContainerPort
ProductToStoreInput : InputPort
input = ProductToStoreFlow
portContainer = Products
ProductStorageDuration : MeasuredCharacteristic
characteristicDefinition = ProductStorageDuration
measurement = ProductStorageDurationMeasurement
store = Products
ProductionLT : MeasuredCharacteristic
characteristicDefinition = ProductionLeadTime
measuredValue = ShortProductionLT
measurement = ProductionLTMeasurement
: FlowTarget
: CharacteristicMeasurement
ProductToStoreFlow : DeliverableFlow
collaboration = ProductionManagementMethod
deliverable = Product
provider = ProductOutput
recipient = ProductToStoreInput
ProductStorageDurationMeasurement : DirectMeasurement
baseMeasurementFrom = ProductionLTMeasurement
observedMeasure = ObservedProductStorageDuration
: CollectiveMeasurementBaseMeasurement
: FlowSource
: ValueAggregation
: CollectiveMeasurementBaseMeasurement
ProductOutput : OutputPort
output = ProductToStoreFlow
portContainer = Build Product
valueAdd = ShortProductionLT
: ValuePortValueAdd
ShortProductionLT : ValueAdd
aggregatedTo = ShortProductionLT
valueDefinition = ShortProductionLeadTime
valueMeasurement = ProductionLT
valuePort = ProductOutput
ProductionLTMeasurement : BinaryMeasurement
baseMeasurement1To = BuildProductDurationMeasurement
baseMeasurementFrom = ProductionLTMeasurement
observedMeasure = ObservedProductionLT
: PortContainerPort
: ValueAddValueMeasurement
: CharacteristicMeasurement
Build Product : Activity
appliedCapabilityOffer = ProductionExecution
capabilityRequirement = Production Execution
collaboration = ProductionManagementMethod
containedPort = ProductOutput, ProductionWorkOrderInput, ReleaseInput, ProductionReportOutput, ProductionLevel1ResourceInput, ProductionLevel2ResourceInput
duration = BuildProductDuration
performingRole = Operator
resourceUse = ProductionResourceUse
BuildProductDuration : MeasuredCharacteristic
activity = Build Product
measurement = BuildProductDurationMeasurement
ProductionLT : MeasuredCharacteristic
characteristicDefinition = ProductionLeadTime
measuredValue = ShortProductionLT
measurement = ProductionLTMeasurement
: ActivityDuration
: CharacteristicMeasurement
BuildProductDurationMeasurement : DirectMeasurement
baseMeasurement1From = ProductionLTMeasurement
observedMeasure = ObservedBuildProductDuration
: BinaryMeasurementBaseMeasurement1

Figure 112. Aggregation of Short Production Lead Time value (objects)

Storage duration in the Products store is based on a direct measure as well (see Figure 107). The activity output port carries a value add object that "aggregates" the production lead time (though actually there is only a single underlying measurement object here). The measurement of the value add object on the output port of the store, aggregates from both production lead time and storage duration. Note also that the value add objects refer to value definitions in the same library that has been presented earlier, in Figure 99.

As can be concluded from the measurements in Figure 112 above, measurements do not refer to measures directly, but to "observed measures". As discussed earlier, observations contain measurements, via "observed measure" objects that refer to both a measure and their resulting measurement(s). This is expressed by the objects in the object diagram in Figure 113. It is important to notice that an observation is set by a context. As is visible in Figure 113, the measurements that relate to production, are contained in the "Manage Production Delegation context", which represent production management in the context of fulfillment. When we would analyze duration of production in the context of pilot production (as part of release management), a different observation would impose a different set of measurements to the same elements in production. Performance and value measurement is notable context-dependent.

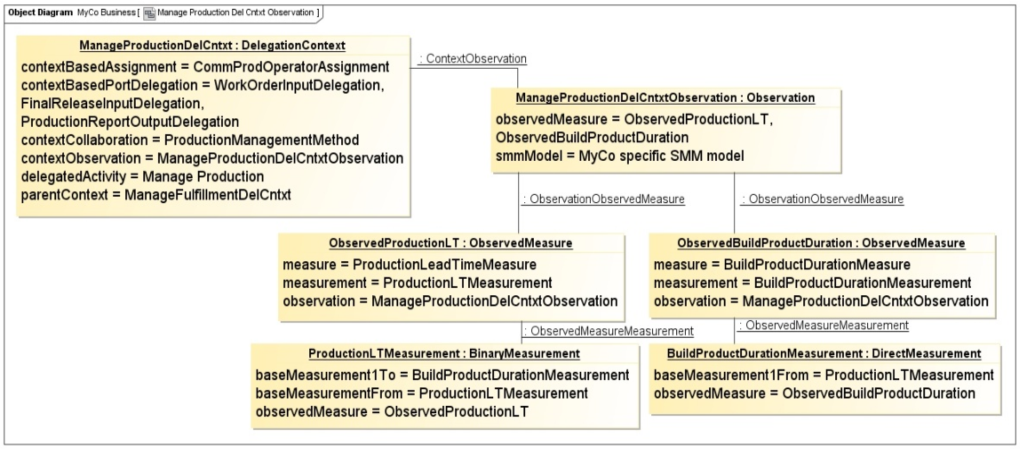

Object Diagram MyCo Business [Manage Production Del Cntxt Observation]
ManageProductionDelCntxt : DelegationContext
contextBasedAssignment = CommProdOperatorAssignment
contextBasedPortDelegation = WorkOrderInputDelegation, FinalReleaseInputDelegation, ProductionReportOutputDelegation
contextCollaboration = ProductionManagementMethod
contextObservation = ManageProductionDelCntxtObservation
delegatedActivity = Manage Production
parentContext = ManageFulfillmentDelCntxt
: ContextObservation
ManageProductionDelCntxtObservation : Observation
observedMeasure = ObservedProductionLT, ObservedBuildProductDuration
smmModel = MyCo specific SMM model
: ObservationObservedMeasure
: ObservationObservedMeasure
ObservedProductionLT : ObservedMeasure
measure = ProductionLeadTimeMeasure
measurement = ProductionLTMeasurement
observation = ManageProductionDelCntxtObservation
ObservedBuildProductDuration : ObservedMeasure
measure = BuildProductDurationMeasure
measurement = BuildProductDurationMeasurement
observation = ManageProductionDelCntxtObservation
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
ProductionLTMeasurement : BinaryMeasurement
baseMeasurement1To = BuildProductDurationMeasurement
baseMeasurementFrom = ProductionLTMeasurement
observedMeasure = ObservedProductionLT
BuildProductDurationMeasurement : DirectMeasurement
baseMeasurement1From = ProductionLTMeasurement
observedMeasure = ObservedBuildProductDuration

Figure 113. Production lead time measurement in context of Manage Production activity (objects)

Objects in the object diagram in Figure 114 indicate how observations, including the “Manage Production Delegation Context Observation” is contained in SMM models, here in the “MyCo specific SMM model”. The measure library that we use is assumably re-usable across multiple value delivery models, and is hence contained in a separate SMM model, called “Generic Library Model”. There is also the “Model specific Measures” library, contained in the “MyCo specific SMM model”, but that library is actually left empty. In 3.1.6 we will consider how the “MyCo specific SMM model”, containing the observations, is contained in the value delivery model itself, whereas the “Generic Library Model” model is not.

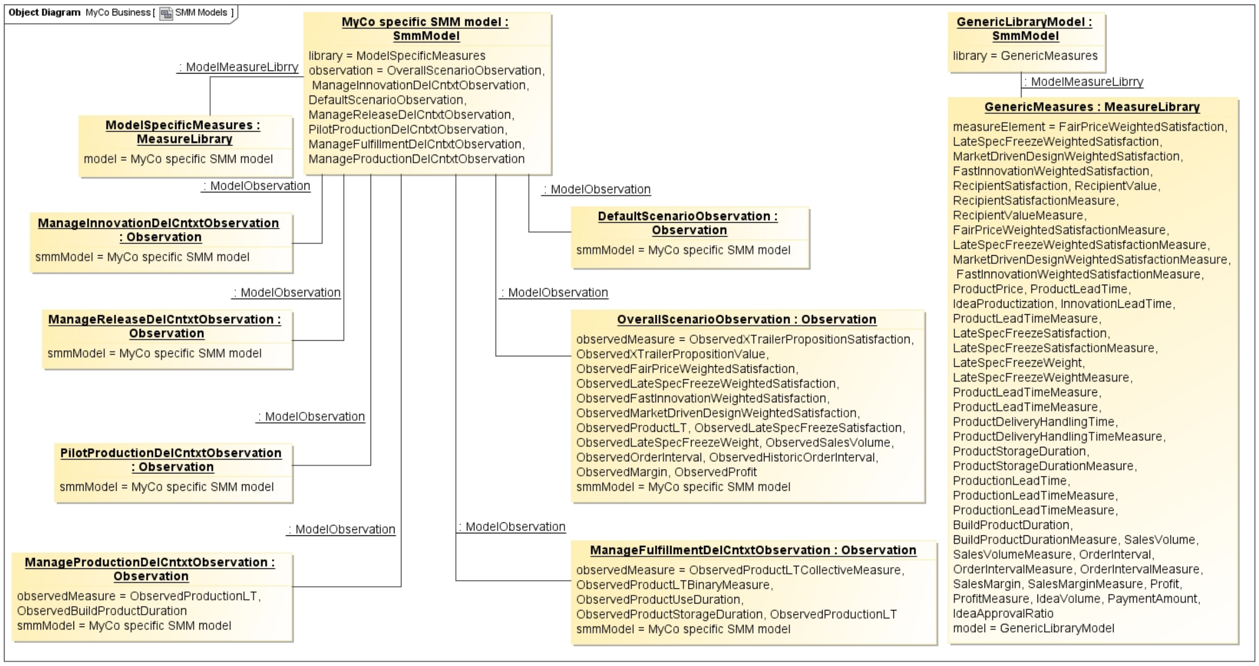

Object Diagram MyCo Business [SMM Models]
MyCo specific SMM model : SmmModel
library = ModelSpecificMeasures
observation = OverallScenarioObservation, ManageInnovationDelCntxtObservation, DefaultScenarioObservation, ManageReleaseDelCntxtObservation, PilotProductionDelCntxtObservation, ManageFulfillmentDelCntxtObservation, ManageProductionDelCntxtObservation
: ModelMeasureLibrry
ModelSpecificMeasures : MeasureLibrary
model = MyCo specific SMM model
: ModelObservation
ManageInnovationDelCntxtObservation : Observation
smmModel = MyCo specific SMM model
: ModelObservation
ManageReleaseDelCntxtObservation : Observation
smmModel = MyCo specific SMM model
: ModelObservation
PilotProductionDelCntxtObservation : Observation
smmModel = MyCo specific SMM model
: ModelObservation
ManageProductionDelCntxtObservation : Observation
observedMeasure = ObservedProductionLT, ObservedBuildProductDuration
smmModel = MyCo specific SMM model
: ModelObservation
DefaultScenarioObservation : Observation
smmModel = MyCo specific SMM model
: ModelObservation
OverallScenarioObservation : Observation
observedMeasure = ObservedXTrailerPropositionSatisfaction, ObservedXTrailerPropositionValue, ObservedFairPriceWeightedSatisfaction, ObservedLateSpecFreezeWeightedSatisfaction, ObservedFastInnovationWeightedSatisfaction, ObservedMarketDrivenDesignWeightedSatisfaction, ObservedProductLT, ObservedLateSpecFreezeSatisfaction, ObservedLateSpecFreezeWeight, ObservedSalesVolume, ObservedOrderInterval, ObservedHistoricOrderInterval, ObservedMargin, ObservedProfit
smmModel = MyCo specific SMM model
: ModelObservation
ManageFulfillmentDelCntxtObservation : Observation
observedMeasure = ObservedProductLTCollectiveMeasure, ObservedProductLTBinaryMeasure, ObservedProductUseDuration, ObservedProductStorageDuration, ObservedProductionLT
smmModel = MyCo specific SMM model
GenericLibraryModel : SmmModel
library = GenericMeasures
: ModelMeasureLibrry
GenericMeasures : MeasureLibrary
measureElement = FairPriceWeightedSatisfaction, LateSpecFreezeWeightedSatisfaction, MarketDrivenDesignWeightedSatisfaction, FastInnovationWeightedSatisfaction, RecipientSatisfaction, RecipientValue, RecipientSatisfactionMeasure, RecipientValueMeasure, FairPriceWeightedSatisfactionMeasure, LateSpecFreezeWeightedSatisfactionMeasure, MarketDrivenDesignWeightedSatisfactionMeasure, FastInnovationWeightedSatisfactionMeasure, ProductPrice, ProductLeadTime, IdeaProductization, InnovationLeadTime, ProductLeadTimeMeasure, LateSpecFreezeSatisfaction, LateSpecFreezeSatisfactionMeasure, LateSpecFreezeWeight, LateSpecFreezeWeightMeasure, ProductLeadTimeMeasure, ProductLeadTimeMeasure, ProductDeliveryHandlingTime, ProductDeliveryHandlingTimeMeasure, ProductStorageDuration, ProductStorageDurationMeasure, ProductionLeadTime, ProductionLeadTimeMeasure, ProductionLeadTimeMeasure, BuildProductDuration, BuildProductDurationMeasure, SalesVolume, SalesVolumeMeasure, OrderInterval, OrderIntervalMeasure, OrderIntervalMeasure, SalesMargin, SalesMarginMeasure, Profit, ProfitMeasure, IdeaVolume, PaymentAmount, IdeaApprovalRatio
model = GenericLibraryModel

Figure 114. SMM models (objects)

The objects in the object diagram in Figure 115 demonstrate how Product Lead Time is constructed from Production Lead Time plus a portion of the duration of the Deliver Product activity, measured as the duration of the “Product Use” object, being a resource use object as contained by the Deliver Product activity. This object connects the input port that denotes the input of the product from the Products store, to the output port that denotes the delivery of the product by the Deliver Product activity. As the Activities metamodel diagram in Figure 65 indicates, a resource use object can also contain a duration. The duration measurement of the “Product Use” object is taken as the measurement of the time it takes to physically handle the product in order to deliver it. The resulting Short Product Lead Time value is propagated up the delegation stack, from the output port of the activity, to the output port ("boundary port") of the Fulfillment Management method.

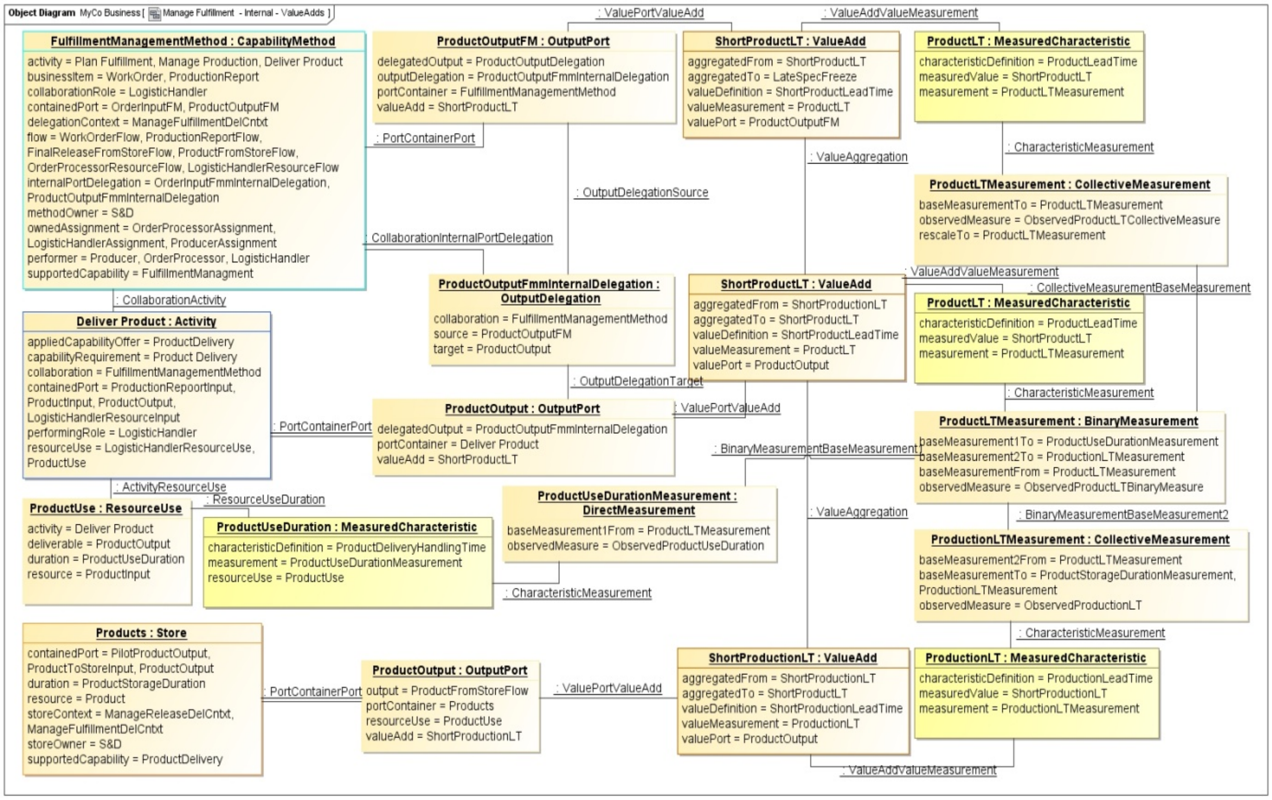

Object Diagram MyCo Business [Manage Fulfillment - Internal - ValueAdds]
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
ProductOutputFM : OutputPort
delegatedOutput = ProductOutputDelegation
outputDelegation = ProductOutputFmmInternalDelegation
portContainer = FulfillmentManagementMethod
valueAdd = ShortProductLT
: ValuePortValueAdd
ShortProductLT : ValueAdd
aggregatedFrom = ShortProductLT
aggregatedTo = LateSpecFreeze
valueDefinition = ShortProductLeadTime
valueMeasurement = ProductLT
valuePort = ProductOutputFM
: ValueAddValueMeasurement
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = ShortProductLT
measurement = ProductLTMeasurement
: PortContainerPort
: ValueAggregation
: CharacteristicMeasurement
ProductLTMeasurement : CollectiveMeasurement
baseMeasurementTo = ProductLTMeasurement
observedMeasure = ObservedProductLTCollectiveMeasure
rescaleTo = ProductLTMeasurement
: OutputDelegationSource
: CollaborationInternalPortDelegation
: CollaborationActivity
ProductOutputFmmInternalDelegation : OutputDelegation
collaboration = FulfillmentManagementMethod
source = ProductOutputFM
target = ProductOutput
: ValueAddValueMeasurement
ShortProductLT : ValueAdd
aggregatedFrom = ShortProductionLT
aggregatedTo = ShortProductLT
valueDefinition = ShortProductLeadTime
valueMeasurement = ProductLT
valuePort = ProductOutput
: CollectiveMeasurementBaseMeasurement
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = ShortProductLT
measurement = ProductLTMeasurement
Deliver Product : Activity
appliedCapabilityOffer = ProductDelivery
capabilityRequirement = Product Delivery
collaboration = FulfillmentManagementMethod
containedPort = ProductionRepoortInput, ProductInput, ProductOutput, LogisticHandlerResourceInput
performingRole = LogisticHandler
resourceUse = LogisticHandlerResourceUse, ProductUse
: OutputDelegationTarget
: CharacteristicMeasurement
: PortContainerPort
ProductOutput : OutputPort
delegatedOutput = ProductOutputFmmInternalDelegation
portContainer = Deliver Product
valueAdd = ShortProductLT
: ValuePortValueAdd
: BinaryMeasurementBaseMeasurement1
ProductLTMeasurement : BinaryMeasurement
baseMeasurement1To = ProductUseDurationMeasurement
baseMeasurement2To = ProductionLTMeasurement
baseMeasurementFrom = ProductLTMeasurement
observedMeasure = ObservedProductLTBinaryMeasure
: ActivityResourceUse
: ResourceUseDuration
ProductUse : ResourceUse
activity = Deliver Product
deliverable = ProductOutput
duration = ProductUseDuration
resource = ProductInput
ProductUseDuration : MeasuredCharacteristic
characteristicDefinition = ProductDeliveryHandlingTime
measurement = ProductUseDurationMeasurement
resourceUse = ProductUse
ProductUseDurationMeasurement : DirectMeasurement
baseMeasurement1From = ProductLTMeasurement
observedMeasure = ObservedProductUseDuration
: ValueAggregation
: BinaryMeasurementBaseMeasurement2
ProductionLTMeasurement : CollectiveMeasurement
baseMeasurement2From = ProductLTMeasurement
baseMeasurementTo = ProductStorageDurationMeasurement, ProductionLTMeasurement
observedMeasure = ObservedProductionLT
: CharacteristicMeasurement
: CharacteristicMeasurement
Products : Store
containedPort = PilotProductOutput, ProductToStoreInput, ProductOutput
duration = ProductStorageDuration
resource = Product
storeContext = ManageReleaseDelCntxt, ManageFulfillmentDelCntxt
storeOwner = S&D
supportedCapability = ProductDelivery
: PortContainerPort
ProductOutput : OutputPort
output = ProductFromStoreFlow
portContainer = Products
resourceUse = ProductUse
valueAdd = ShortProductionLT
: ValuePortValueAdd
ShortProductionLT : ValueAdd
aggregatedFrom = ShortProductionLT
aggregatedTo = ShortProductLT
valueDefinition = ShortProductionLeadTime
valueMeasurement = ProductionLT
valuePort = ProductOutput
ProductionLT : MeasuredCharacteristic
characteristicDefinition = ProductionLeadTime
measuredValue = ShortProductionLT
measurement = ProductionLTMeasurement
: ValueAddValueMeasurement

Figure 115. Aggregation of Short Product Lead Time value (objects)

The objects in the object diagram in Figure 116 demonstrate how the Short Product Lead Time value is propagated further, from the output port of the Fulfillment Management method to the Late Spec Freeze value on the output port of the Manage Fulfillment activity in the business network.

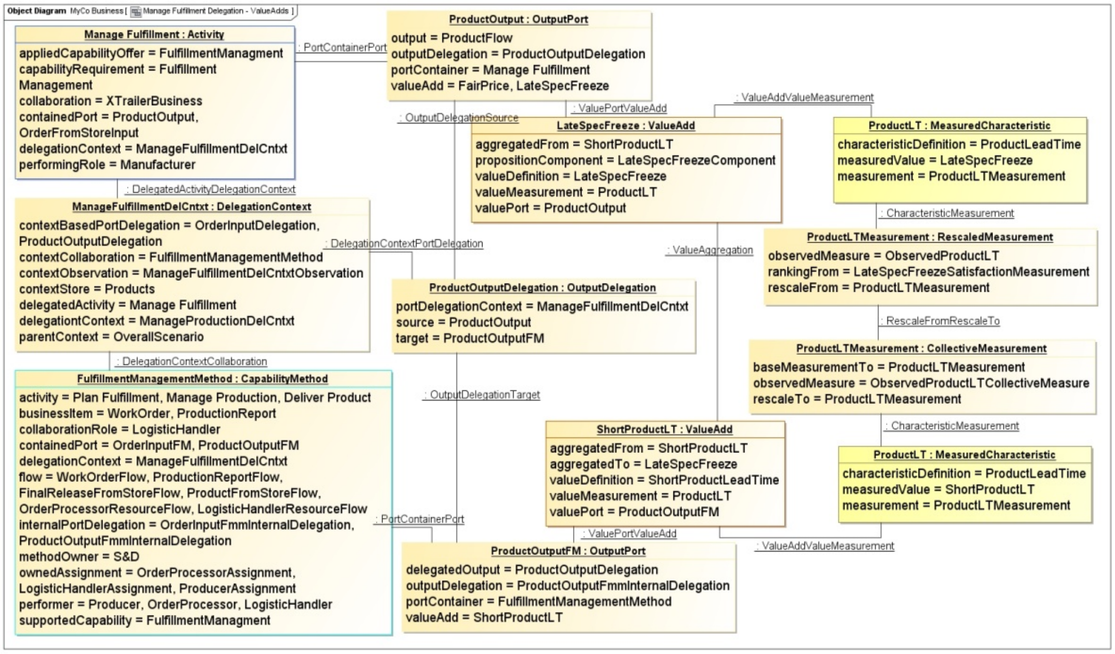

Object Diagram MyCo Business [Manage Fulfillment Delegation - ValueAdds]
Manage Fulfillment : Activity
appliedCapabilityOffer = FulfillmentManagment
capabilityRequirement = Fulfillment Management
collaboration = XTrailerBusiness
containedPort = ProductOutput, OrderFromStoreInput
delegationContext = ManageFulfillmentDelCntxt
performingRole = Manufacturer
: PortContainerPort
ProductOutput : OutputPort
output = ProductFlow
outputDelegation = ProductOutputDelegation
portContainer = Manage Fulfillment
valueAdd = FairPrice, LateSpecFreeze
: OutputDelegationSource
: ValuePortValueAdd
: ValueAddValueMeasurement
LateSpecFreeze : ValueAdd
aggregatedFrom = ShortProductLT
propositionComponent = LateSpecFreezeComponent
valueDefinition = LateSpecFreeze
valueMeasurement = ProductLT
valuePort = ProductOutput
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = LateSpecFreeze
measurement = ProductLTMeasurement
: DelegatedActivityDelegationContext
ManageFulfillmentDelCntxt : DelegationContext
contextBasedPortDelegation = OrderInputDelegation, ProductOutputDelegation
contextCollaboration = FulfillmentManagementMethod
contextObservation = ManageFulfillmentDelCntxtObservation
contextStore = Products
delegatedActivity = Manage Fulfillment
delegationtContext = ManageProductionDelCntxt
parentContext = OverallScenario
: DelegationContextPortDelegation
: ValueAggregation
: CharacteristicMeasurement
ProductLTMeasurement : RescaledMeasurement
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasurement
rescaleFrom = ProductLTMeasurement
ProductOutputDelegation : OutputDelegation
portDelegationContext = ManageFulfillmentDelCntxt
source = ProductOutput
target = ProductOutputFM
: RescaleFromRescaleTo
: DelegationContextCollaboration
ProductLTMeasurement : CollectiveMeasurement
baseMeasurementTo = ProductLTMeasurement
observedMeasure = ObservedProductLTCollectiveMeasure
rescaleTo = ProductLTMeasurement
FulfillmentManagementMethod : CapabilityMethod
activity = Plan Fulfillment, Manage Production, Deliver Product
businessItem = WorkOrder, ProductionReport
collaborationRole = LogisticHandler
containedPort = OrderInputFM, ProductOutputFM
delegationContext = ManageFulfillmentDelCntxt
flow = WorkOrderFlow, ProductionReportFlow, FinalReleaseFromStoreFlow, ProductFromStoreFlow, OrderProcessorResourceFlow, LogisticHandlerResourceFlow
internalPortDelegation = OrderInputFmmInternalDelegation, ProductOutputFmmInternalDelegation
methodOwner = S&D
ownedAssignment = OrderProcessorAssignment, LogisticHandlerAssignment, ProducerAssignment
performer = Producer, OrderProcessor, LogisticHandler
supportedCapability = FulfillmentManagment
: OutputDelegationTarget
: CharacteristicMeasurement
ShortProductLT : ValueAdd
aggregatedFrom = ShortProductLT
aggregatedTo = LateSpecFreeze
valueDefinition = ShortProductLeadTime
valueMeasurement = ProductLT
valuePort = ProductOutputFM
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = ShortProductLT
measurement = ProductLTMeasurement
: PortContainerPort
: ValuePortValueAdd
: ValueAddValueMeasurement
ProductOutputFM : OutputPort
delegatedOutput = ProductOutputDelegation
outputDelegation = ProductOutputFmmInternalDelegation
portContainer = FulfillmentManagementMethod
valueAdd = ShortProductLT

Figure 116. Aggregation of Late Spec Freeze value (objects)

From here on the product is actually delivered to the customer, as this output port connects to the actual deliverable flow in the business network.

The objects in the object diagram in Figure 117 show how measurements, as modeled above, are contained in the observation of the delegation context that is set by the Manage Fulfillment activity.

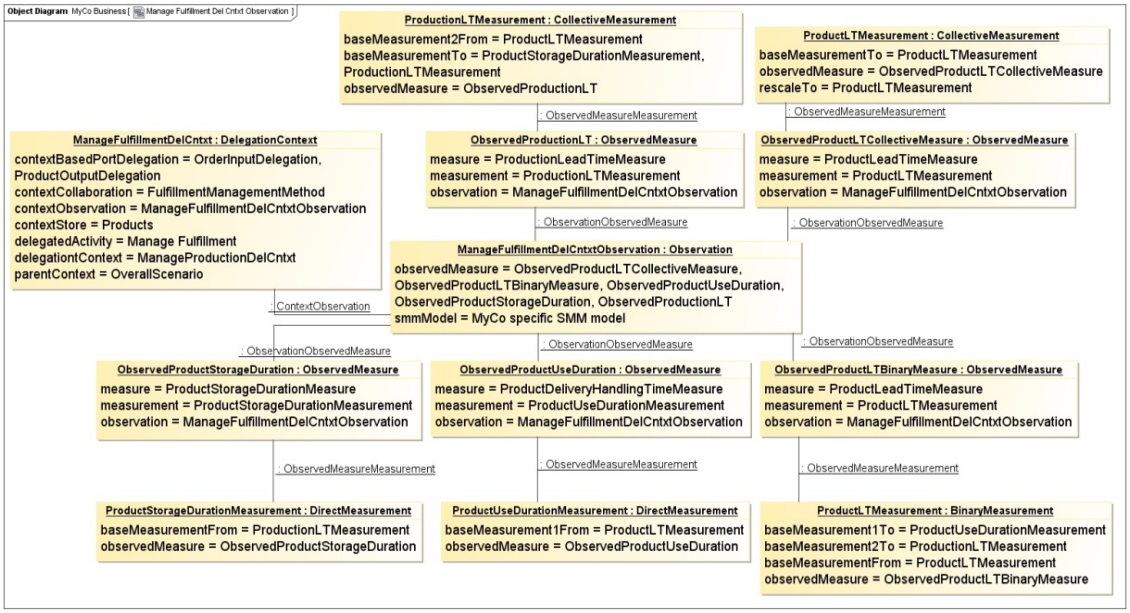

Object Diagram MyCo Business [Manage Fulfilment Del Cntxt Observation]
ProductionLTMeasurement : CollectiveMeasurement
baseMeasurement2From = ProductLTMeasurement
baseMeasurementTo = ProductStorageDurationMeasurement, ProductionLTMeasurement
observedMeasure = ObservedProductionLT
ProductLTMeasurement : CollectiveMeasurement
baseMeasurementTo = ProductLTMeasurement
observedMeasure = ObservedProductLTCollectiveMeasure
rescaleTo = ProductLTMeasurement
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
ManageFulfillmentDelCntxt : DelegationContext
contextBasedPortDelegation = OrderInputDelegation, ProductOutputDelegation
contextCollaboration = FulfillmentManagementMethod
contextObservation = ManageFulfillmentDelCntxtObservation
contextStore = Products
delegatedActivity = Manage Fulfillment
delegationtContext = ManageProductionDelCntxt
parentContext = OverallScenario
ObservedProductionLT : ObservedMeasure
measure = ProductionLeadTimeMeasure
measurement = ProductionLTMeasurement
observation = ManageFulfillmentDelCntxtObservation
ObservedProductLTCollectiveMeasure : ObservedMeasure
measure = ProductLeadTimeMeasure
measurement = ProductLTMeasurement
observation = ManageFulfillmentDelCntxtObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
ManageFulfillmentDelCntxtObservation : Observation
observedMeasure = ObservedProductLTCollectiveMeasure, ObservedProductLTBinaryMeasure, ObservedProductUseDuration, ObservedProductStorageDuration, ObservedProductionLT
smmModel = MyCo specific SMM model
: ContextObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
: ObservationObservedMeasure
ObservedProductStorageDuration : ObservedMeasure
measure = ProductStorageDurationMeasure
measurement = ProductStorageDurationMeasurement
observation = ManageFulfillmentDelCntxtObservation
ObservedProductUseDuration : ObservedMeasure
measure = ProductDeliveryHandlingTimeMeasure
measurement = ProductUseDurationMeasurement
observation = ManageFulfillmentDelCntxtObservation
ObservedProductLTBinaryMeasure : ObservedMeasure
measure = ProductLeadTimeMeasure
measurement = ProductLTMeasurement
observation = ManageFulfillmentDelCntxtObservation
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
ProductStorageDurationMeasurement : DirectMeasurement
baseMeasurementFrom = ProductionLTMeasurement
observedMeasure = ObservedProductStorageDuration
ProductUseDurationMeasurement : DirectMeasurement
baseMeasurement1From = ProductLTMeasurement
observedMeasure = ObservedProductUseDuration
ProductLTMeasurement : BinaryMeasurement
baseMeasurement1To = ProductUseDurationMeasurement
baseMeasurement2To = ProductionLTMeasurement
baseMeasurementFrom = ProductLTMeasurement
observedMeasure = ObservedProductLTBinaryMeasure

Figure 117. Storage and delivery lead time measurement in context of Manage Fulfillment activity (objects)

Having modeled so-far how the Late Spec Freeze value measurement is constructed, we will now consider how to measure the level of satisfaction of the customer (transporter) with that value, as part of the value proposition. Objects in the object diagram in Figure 118 represent the Late Spec Freeze component in the "XTrailer Proposition", together with its measured characteristics and their related measurement objects, as well as how the value proposition component articulates the “underlying” Late Spec Freeze value as carried by the output port of the Manage Fulfillment activity.

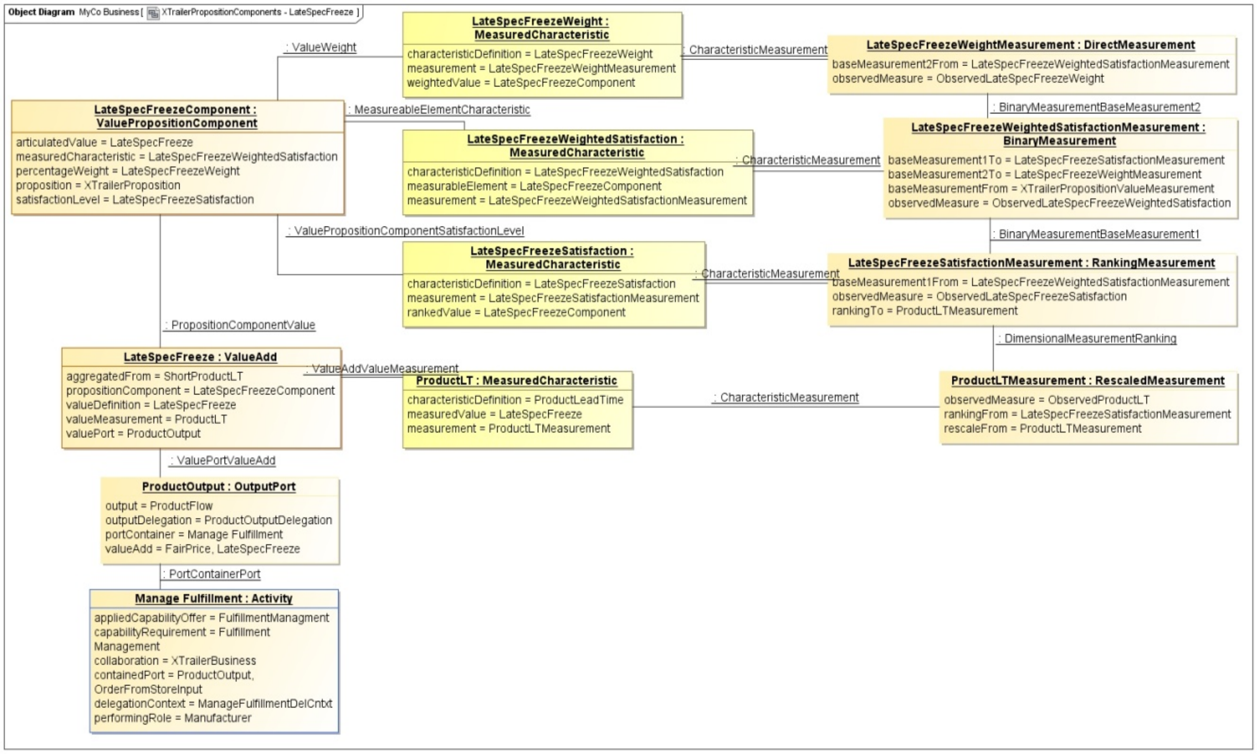
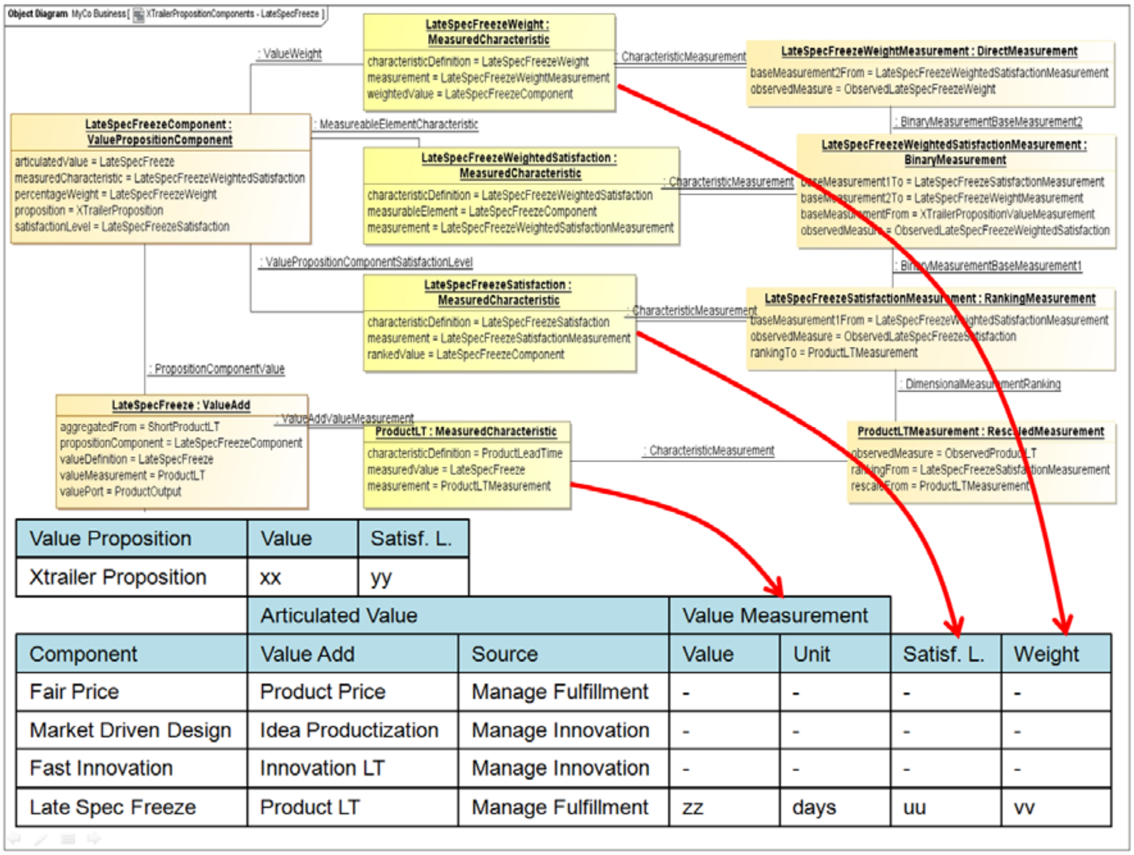

Object Diagram MyCo Business [XTrailerPropositionComponents - LateSpecFreeze]
LateSpecFreezeWeight : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeWeight
measurement = LateSpecFreezeWeightMeasurement
weightedValue = LateSpecFreezeComponent
: ValueWeight
: CharacteristicMeasurement
LateSpecFreezeWeightMeasurement : DirectMeasurement
baseMeasurement2From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeWeight
: BinaryMeasurementBaseMeasurement2
LateSpecFreezeComponent : ValuePropositionComponent
articulatedValue = LateSpecFreeze
measuredCharacteristic = LateSpecFreezeWeightedSatisfaction
percentageWeight = LateSpecFreezeWeight
proposition = XTrailerProposition
satisfactionLevel = LateSpecFreezeSatisfaction
: MeasureableElementCharacteristic
LateSpecFreezeWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeWeightedSatisfaction
measurableElement = LateSpecFreezeComponent
measurement = LateSpecFreezeWeightedSatisfactionMeasurement
: CharacteristicMeasurement
LateSpecFreezeWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurement1To = LateSpecFreezeSatisfactionMeasurement
baseMeasurement2To = LateSpecFreezeWeightMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction
: ValuePropositionComponentSatisfactionLevel
: BinaryMeasurementBaseMeasurement1
LateSpecFreezeSatisfaction : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeSatisfaction
measurement = LateSpecFreezeSatisfactionMeasurement
rankedValue = LateSpecFreezeComponent
: CharacteristicMeasurement
LateSpecFreezeSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeSatisfaction
rankingTo = ProductLTMeasurement
: PropositionComponentValue
: DimensionalMeasurementRanking
LateSpecFreeze : ValueAdd
aggregatedFrom = ShortProductLT
propositionComponent = LateSpecFreezeComponent
valueDefinition = LateSpecFreeze
valueMeasurement = ProductLT
valuePort = ProductOutput
: ValueAddValueMeasurement
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = LateSpecFreeze
measurement = ProductLTMeasurement
: CharacteristicMeasurement
ProductLTMeasurement : RescaledMeasurement
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasurement
rescaleFrom = ProductLTMeasurement
: ValuePortValueAdd
ProductOutput : OutputPort
output = ProductFlow
outputDelegation = ProductOutputDelegation
portContainer = Manage Fulfillment
valueAdd = FairPrice, LateSpecFreeze
: PortContainerPort
Manage Fulfillment : Activity
appliedCapabilityOffer = FulfillmentManagment
capabilityRequirement = Fulfillment Management
collaboration = XTrailerBusiness
containedPort = ProductOutput, OrderFromStoreInput
delegationContext = ManageFulfillmentDelCntxt
performingRole = Manufacturer

Figure 118. Measurements related to value proposition component and related value (objects).

Figure 119 shows how measured characteristics and related measurements from Figure 118 are exposed in the view mockup (the one that was introduced earlier in Figure 95).


Object Diagram MyCo Business [XTrailerPropositionComponents - LateSpecFreeze]
LateSpecFreezeWeight : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeWeight
measurement = LateSpecFreezeWeightMeasurement
weightedValue = LateSpecFreezeComponent
: ValueWeight
: CharacteristicMeasurement
LateSpecFreezeWeightMeasurement : DirectMeasurement
baseMeasurement2From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeWeight
: BinaryMeasurementBaseMeasurement2
LateSpecFreezeComponent : ValuePropositionComponent
articulatedValue = LateSpecFreeze
measuredCharacteristic = LateSpecFreezeWeightedSatisfaction
percentageWeight = LateSpecFreezeWeight
proposition = XTrailerProposition
satisfactionLevel = LateSpecFreezeSatisfaction
: MeasureableElementCharacteristic
LateSpecFreezeWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeWeightedSatisfaction
measurableElement = LateSpecFreezeComponent
measurement = LateSpecFreezeWeightedSatisfactionMeasurement
: CharacteristicMeasurement
LateSpecFreezeWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurement1To = LateSpecFreezeSatisfactionMeasurement
baseMeasurement2To = LateSpecFreezeWeightMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction
: ValuePropositionComponentSatisfactionLevel
: BinaryMeasurementBaseMeasurement1
LateSpecFreezeSatisfaction : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeSatisfaction
measurement = LateSpecFreezeSatisfactionMeasurement
rankedValue = LateSpecFreezeComponent
: CharacteristicMeasurement
LateSpecFreezeSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeSatisfaction
rankingTo = ProductLTMeasurement
: PropositionComponentValue
: DimensionalMeasurementRanking
LateSpecFreeze : ValueAdd
aggregatedFrom = ShortProductLT
propositionComponent = LateSpecFreezeComponent
valueDefinition = LateSpecFreeze
valueMeasurement = ProductLT
valuePort = ProductOutput
: ValueAddValueMeasurement
ProductLT : MeasuredCharacteristic
characteristicDefinition = ProductLeadTime
measuredValue = LateSpecFreeze
measurement = ProductLTMeasurement
: CharacteristicMeasurement
ProductLTMeasurement : RescaledMeasurement
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasurement
rescaledFrom = ProductLTMeasurement

Value Proposition	Value	Satisf. L.
Xtrailer Proposition	xx	yy

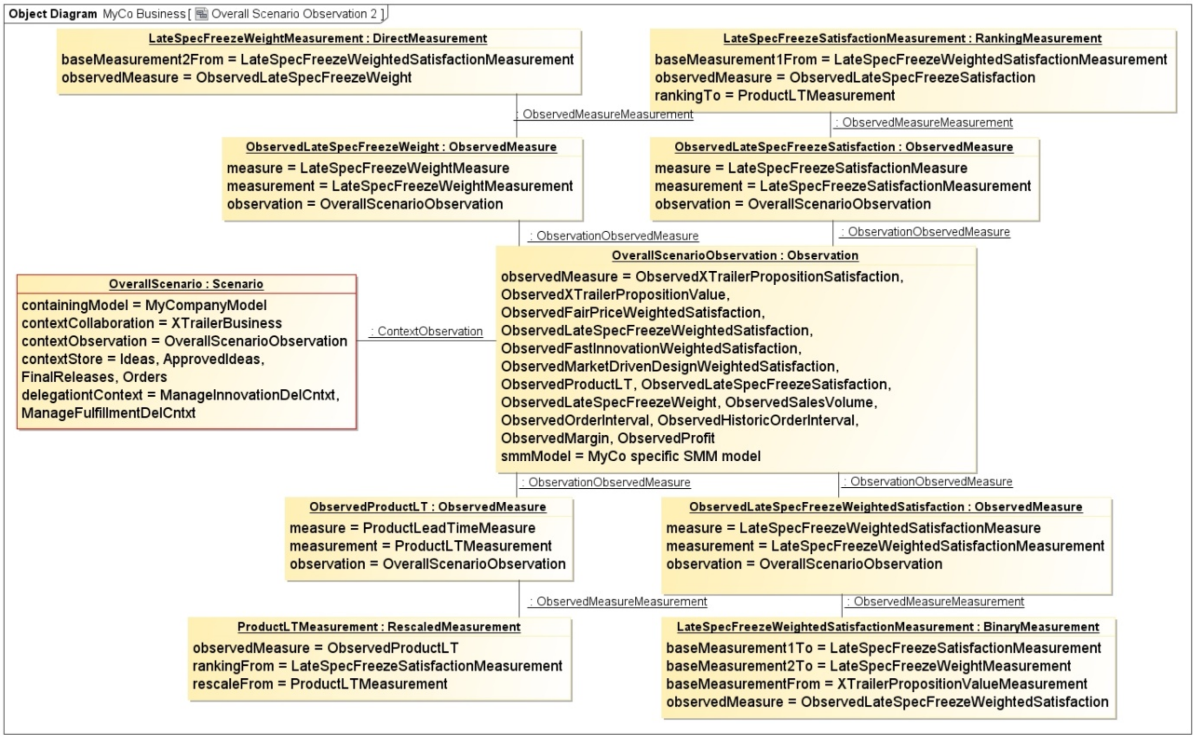
	Articulated Value		Value Measurement			
Component	Value Add	Source	Value	Unit	Satisf. L.	Weight
Fair Price	Product Price	Manage Fulfillment	-	-	-	-
Market Driven Design	Idea Productization	Manage Innovation	-	-	-	-
Fast Innovation	Innovation LT	Manage Innovation	-	-	-	-
Late Spec Freeze	Product LT	Manage Fulfillment	zz	days	uu	vv

Figure 119. Value proposition component, objects mapped on view

Note the distinction between four different measurements here:

- Measurement of the “underlying” value, via the Product Lead Time value measurement.
- Measurement of the satisfaction level of the customer with the “underlying” value, via a ranking measurement that ranks the Product Lead Time value measurement.
- A measurement of how important the value is to the customer, relative to the other values in the value proposition, via Late Spec Freeze Weight Measurement, being a direct measurement.
- A measurement of the aggregation (actually multiplication) of the both the previous measurements, via the Late Spec Freeze Weighted Satisfaction Measurement.

The objects in the object diagram in Figure 120 indicate how these four measurements, via their observed measure objects, are contained in the observation of the “Overall Scenario”.


Object Diagram MyCo Business [Overall Scenario Observation 2]
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
: ObservationObservedMeasure
: ObservationObservedMeasure
: ContextObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement

LateSpecFreezeWeightMeasurement : DirectMeasurement
baseMeasurement2From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeWeight

LateSpecFreezeSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = LateSpecFreezeWeightedSatisfactionMeasurement
observedMeasure = ObservedLateSpecFreezeSatisfaction
rankingTo = ProductLTMeasurement

ObservedLateSpecFreezeWeight : ObservedMeasure
measure = LateSpecFreezeWeightMeasure
measurement = LateSpecFreezeWeightMeasurement
observation = OverallScenarioObservation

ObservedLateSpecFreezeSatisfaction : ObservedMeasure
measure = LateSpecFreezeSatisfactionMeasure
measurement = LateSpecFreezeSatisfactionMeasurement
observation = OverallScenarioObservation

OverallScenario : Scenario
containingModel = MyCompanyModel
contextCollaboration = XTrailerBusiness
contextObservation = OverallScenarioObservation
contextStore = Ideas, ApprovedIdeas, FinalReleases, Orders
delegationtContext = ManageInnovationDelCntxt, ManageFulfillmentDelCntxt

OverallScenarioObservation : Observation
observedMeasure = ObservedXTrailerPropositionSatisfaction, ObservedXTrailerPropositionValue, ObservedFairPriceWeightedSatisfaction, ObservedLateSpecFreezeWeightedSatisfaction, ObservedFastInnovationWeightedSatisfaction, ObservedMarketDrivenDesignWeightedSatisfaction, ObservedProductLT, ObservedLateSpecFreezeSatisfaction, ObservedLateSpecFreezeWeight, ObservedSalesVolume, ObservedOrderInterval, ObservedHistoricOrderInterval, ObservedMargin, ObservedProfit
smmModel = MyCo specific SMM model

ObservedProductLT : ObservedMeasure
measure = ProductLeadTimeMeasure
measurement = ProductLTMeasurement
observation = OverallScenarioObservation

ObservedLateSpecFreezeWeightedSatisfaction : ObservedMeasure
measure = LateSpecFreezeWeightedSatisfactionMeasure
measurement = LateSpecFreezeWeightedSatisfactionMeasurement
observation = OverallScenarioObservation

ProductLTMeasurement : RescaledMeasurement
observedMeasure = ObservedProductLT
rankingFrom = LateSpecFreezeSatisfactionMeasurement
rescaleFrom = ProductLTMeasurement

LateSpecFreezeWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurement1To = LateSpecFreezeSatisfactionMeasurement
baseMeasurement2To = LateSpecFreezeWeightMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction

Figure 120. Overall scenario observation measurements, first part (Objects)

As Figure 118 indicates, the “Late Spec Freeze Satisfaction Measurement” is a ranking to the underlying measurement of Product Lead Time, based on “Late Spec Freeze Satisfaction Measure”, which is used as measure, according to the observed measure object “Observed Late Spec Freeze Satisfaction”. This measure, being a ranking, is a dimensional measure, and can itself serve as basis for further aggregation by other measures, in accordance to its unit of measure, being defined as “satisfaction points”, as the earlier presented measure library object diagram in Figure 105 indicates. Note that Figure 110 did already provide a representation of this ranking measurement. The ranking measurement is multiplied by “Late Spec Freeze Weight Measurement”, conform its related “Late Spec Freeze Weight Measure”, which is a direct measure, having “percent” as its unit of measure (see Figure 105 again). The multiplication itself is defined by “Late Spec Freeze Weighted Satisfaction Measure”, which is the measure that is used to establish the “Late Spec Freeze Weighted Satisfaction Measurement”, as is indicated by its related observed measure object in Figure 120. As the measure library object diagram in Figure 105 shows, we defined a “functor” that multiplies both measurements, and divides it by 100. Though this “functor” is technically valid according to the current SMM version (see SMM (2012)), it would have to be defined in a slightly further structured way according to the intended revision of SMM (see **Error! Reference source not found.**)), namely as “custom” functor, with a related custom functor operation that defines this multiplication and division. Alternatively, and equally valid, the measure could be replaced by two measures: a binary measure with standard functor “product”, doing the multiplication, and a rescaled measure that divides the result by 100. But the reader will get the idea.

So-far we have focused on the structured and detailed modeling of measurements associated with the Late Spec Freeze component in the value proposition. The measurement and measurement aggregation in relation to the other three components could have been modeled in similar ways, which would also involve consideration of characteristics of different type, such as cost-related, and related to the performance of transforming ideas into innovations. But in order to keep the model content as small as possible, we limit ourselves to just incorporating similar weighted satisfaction measurements for the other three proposition components in the model, and leave modeling of details of how these measurements are constructed out of consideration.

The object diagram in Figure 121 shows how the measurement of the overall XTrailer Proposition value, being a collective measurement, is based on the accumulation of the four weighted satisfaction measurements, associated with the four proposition components. The diagram also shows the ranking measurement, ranking the XTrailer Proposition value. The ranking measurement expresses customer satisfaction with the XTrailer Proposition.

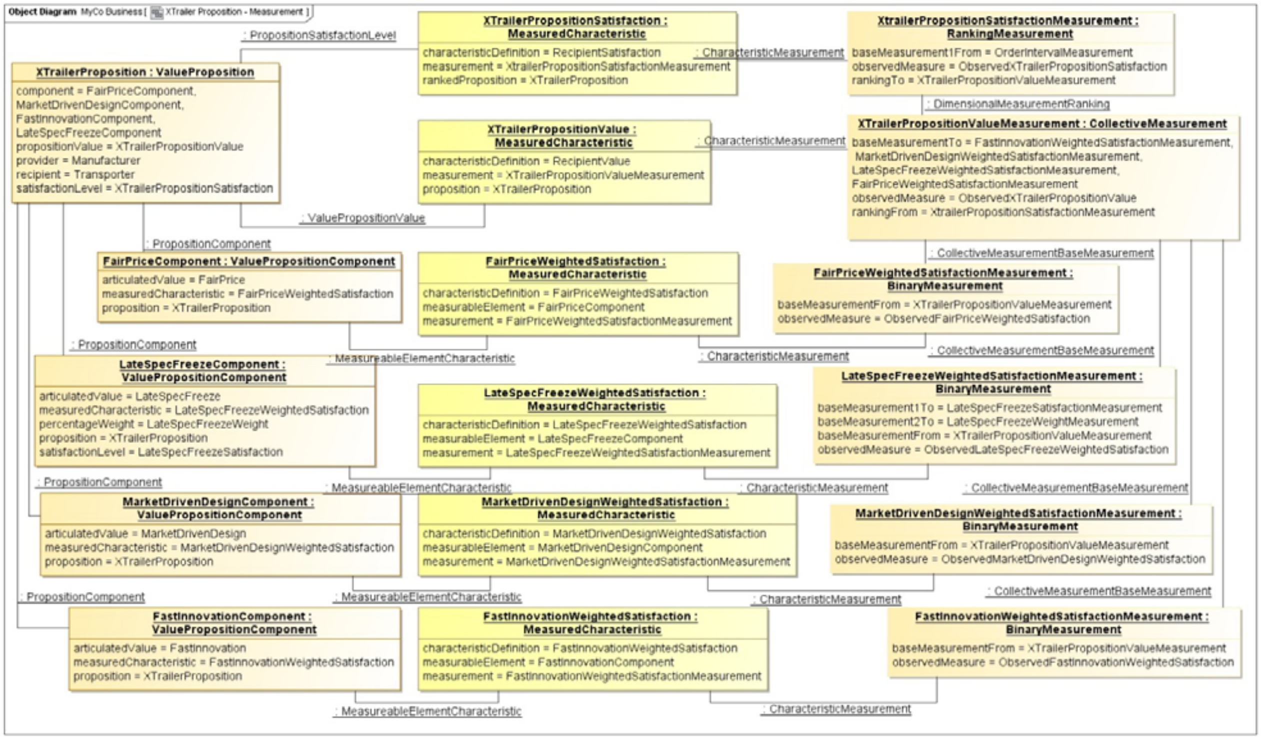
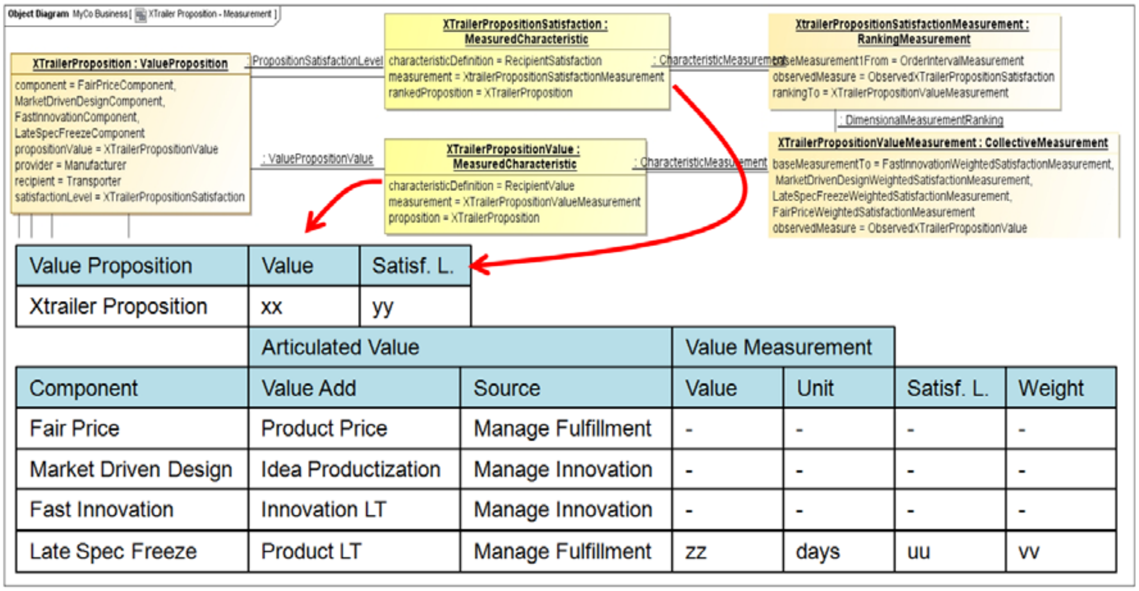

Object Diagram MyCo Business [XTrailer Proposition - Measurement]
XTrailerProposition : ValueProposition
component = FairPriceComponent, MarketDrivenDesignComponent, FastInnovationComponent, LateSpecFreezeComponent
propositionValue = XTrailerPropositionValue
provider = Manufacturer
recipient = Transporter
satisfactionLevel = XTrailerPropositionSatisfaction
: PropositionSatisfactionLevel
XTrailerPropositionSatisfaction : MeasuredCharacteristic
characteristicDefinition = RecipientSatisfaction
measurement = XtrailerPropositionSatisfactionMeasurement
rankedProposition = XTrailerProposition
: CharacteristicMeasurement
XtrailerPropositionSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = OrderIntervalMeasurement
observedMeasure = ObservedXTrailerPropositionSatisfaction
rankingTo = XTrailerPropositionValueMeasurement
: DimensionalMeasurementRanking
XTrailerPropositionValue : MeasuredCharacteristic
characteristicDefinition = RecipientValue
measurement = XTrailerPropositionValueMeasurement
proposition = XTrailerProposition
: CharacteristicMeasurement
XTrailerPropositionValueMeasurement : CollectiveMeasurement
baseMeasurementTo = FastInnovationWeightedSatisfactionMeasurement, MarketDrivenDesignWeightedSatisfactionMeasurement, LateSpecFreezeWeightedSatisfactionMeasurement, FairPriceWeightedSatisfactionMeasurement
observedMeasure = ObservedXTrailerPropositionValue
rankingFrom = XtrailerPropositionSatisfactionMeasurement
: ValuePropositionValue
: PropositionComponent
FairPriceComponent : ValuePropositionComponent
articulatedValue = FairPrice
measuredCharacteristic = FairPriceWeightedSatisfaction
proposition = XTrailerProposition
FairPriceWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = FairPriceWeightedSatisfaction
measurableElement = FairPriceComponent
measurement = FairPriceWeightedSatisfactionMeasurement
: CollectiveMeasurementBaseMeasurement
FairPriceWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedFairPriceWeightedSatisfaction
: PropositionComponent
: MeasureableElementCharacteristic
: CharacteristicMeasurement
: CollectiveMeasurementBaseMeasurement
LateSpecFreezeComponent : ValuePropositionComponent
articulatedValue = LateSpecFreeze
measuredCharacteristic = LateSpecFreezeWeightedSatisfaction
percentageWeight = LateSpecFreezeWeight
proposition = XTrailerProposition
satisfactionLevel = LateSpecFreezeSatisfaction
LateSpecFreezeWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = LateSpecFreezeWeightedSatisfaction
measurableElement = LateSpecFreezeComponent
measurement = LateSpecFreezeWeightedSatisfactionMeasurement
LateSpecFreezeWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurement1To = LateSpecFreezeSatisfactionMeasurement
baseMeasurement2To = LateSpecFreezeWeightMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction
: PropositionComponent
: MeasureableElementCharacteristic
: CharacteristicMeasurement
: CollectiveMeasurementBaseMeasurement
MarketDrivenDesignComponent : ValuePropositionComponent
articulatedValue = MarketDrivenDesign
measuredCharacteristic = MarketDrivenDesignWeightedSatisfaction
proposition = XTrailerProposition
MarketDrivenDesignWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = MarketDrivenDesignWeightedSatisfaction
measurableElement = MarketDrivenDesignComponent
measurement = MarketDrivenDesignWeightedSatisfactionMeasurement
MarketDrivenDesignWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedMarketDrivenDesignWeightedSatisfaction
: PropositionComponent
: MeasureableElementCharacteristic
: CharacteristicMeasurement
: CollectiveMeasurementBaseMeasurement
FastInnovationComponent : ValuePropositionComponent
articulatedValue = FastInnovation
measuredCharacteristic = FastInnovationWeightedSatisfaction
proposition = XTrailerProposition
FastInnovationWeightedSatisfaction : MeasuredCharacteristic
characteristicDefinition = FastInnovationWeightedSatisfaction
measurableElement = FastInnovationComponent
measurement = FastInnovationWeightedSatisfactionMeasurement
FastInnovationWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedFastInnovationWeightedSatisfaction
: MeasureableElementCharacteristic
: CharacteristicMeasurement

Figure 121. Measurements related to value propositions and related components (objects)

Figure 122 shows how measured characteristics and related measurements from Figure 118 are exposed in the view mockup (the one that was introduced earlier in Figure 95).


Object Diagram MyCo Business [XTrailer Proposition - Measurement]
XTrailerProposition : ValueProposition
component = FairPriceComponent, MarketDrivenDesignComponent, FastInnovationComponent, LateSpecFreezeComponent
propositionValue = XTrailerPropositionValue
provider = Manufacturer
recipient = Transporter
satisfactionLevel = XTrailerPropositionSatisfaction
: PropositionSatisfactionLevel
XTrailerPropositionSatisfaction : MeasuredCharacteristic
characteristicDefinition = RecipientSatisfaction
measurement = XtrailerPropositionSatisfactionMeasurement
rankedProposition = XTrailerProposition
: CharacteristicMeasurement
XtrailerPropositionSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = OrderIntervalMeasurement
observedMeasure = ObservedXTrailerPropositionSatisfaction
rankingTo = XTrailerPropositionValueMeasurement
: DimensionalMeasurementRanking
: ValuePropositionValue
XTrailerPropositionValue : MeasuredCharacteristic
characteristicDefinition = RecipientValue
measurement = XTrailerPropositionValueMeasurement
proposition = XTrailerProposition
: CharacteristicMeasurement
XTrailerPropositionValueMeasurement : CollectiveMeasurement
baseMeasurementTo = FastInnovationWeightedSatisfactionMeasurement, MarketDrivenDesignWeightedSatisfactionMeasurement, LateSpecFreezeWeightedSatisfactionMeasurement, FairPriceWeightedSatisfactionMeasurement
observedMeasure = ObservedXTrailerPropositionValue

Value Proposition	Value	Satisf. L.
Xtrailer Proposition	xx	yy

	Articulated Value		Value Measurement			
Component	Value Add	Source	Value	Unit	Satisf. L.	Weight
Fair Price	Product Price	Manage Fulfillment	-	-	-	-
Market Driven Design	Idea Productization	Manage Innovation	-	-	-	-
Fast Innovation	Innovation LT	Manage Innovation	-	-	-	-
Late Spec Freeze	Product LT	Manage Fulfillment	zz	days	uu	vv

Figure 122. Value proposition, objects mapped on view

The objects in the object diagram in Figure 123 indicate how the proposition component-related weighted satisfaction measurements, as well as the proposition-related measurements, via their observed measure objects, are contained in the observation of the “Overall Scenario”. As the diagram indicates, the observed measure of XTrailer Proposition Value refers to “Recipient Value Measure” as the measure that establishes the “XTrailer Proposition Value Measurement”. As the object diagram of the measure library in Figure 105 indicates, this is a collective measure, against the characteristic “Recipient Value” as “trait”, “sum” as accumulator, and “satisfaction points” as unit of measure. Its ranking measure, “Recipient Satisfaction Measure”, contained in the same measure library, has the same unit of measure and has “Recipient Satisfaction” as “trait”.

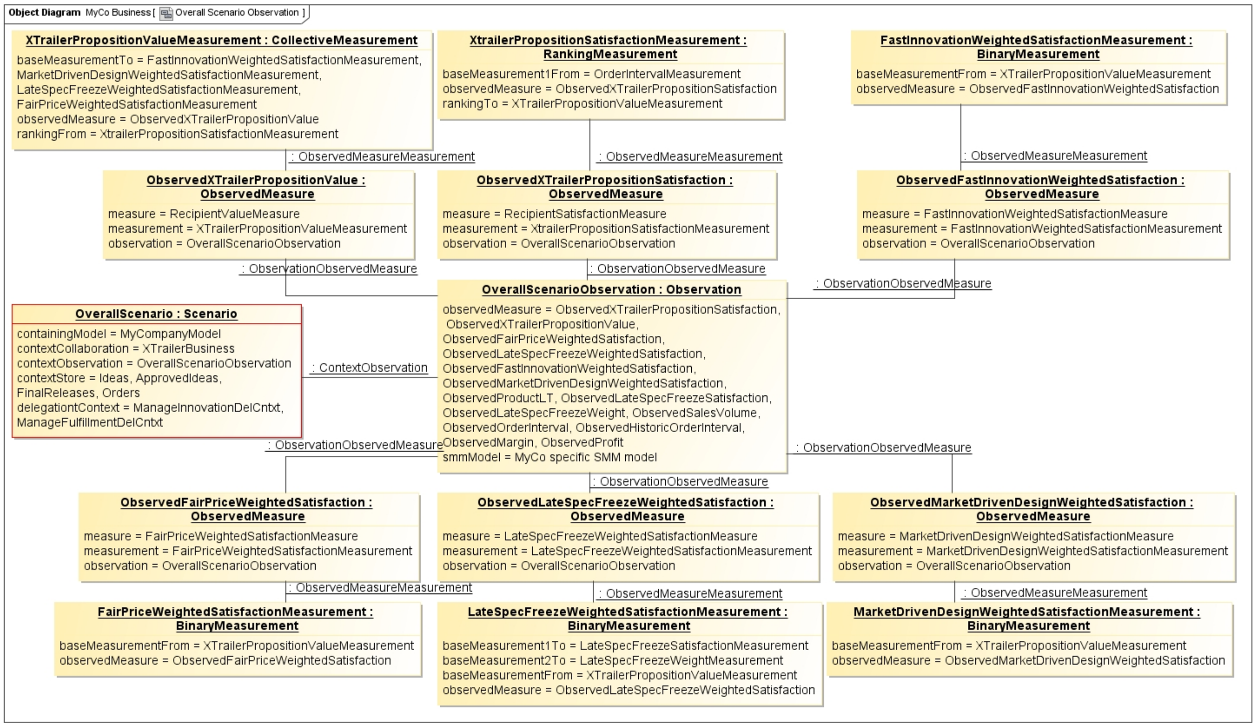

Object Diagram MyCo Business [Overall Scenario Observation]
XTrailerPropositionValueMeasurement : CollectiveMeasurement
baseMeasurementTo = FastInnovationWeightedSatisfactionMeasurement, MarketDrivenDesignWeightedSatisfactionMeasurement, LateSpecFreezeWeightedSatisfactionMeasurement, FairPriceWeightedSatisfactionMeasurement
observedMeasure = ObservedXTrailerPropositionValue
rankingFrom = XtrailerPropositionSatisfactionMeasurement
XtrailerPropositionSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = OrderIntervalMeasurement
observedMeasure = ObservedXTrailerPropositionSatisfaction
rankingTo = XTrailerPropositionValueMeasurement
FastInnovationWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedFastInnovationWeightedSatisfaction
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
ObservedXTrailerPropositionValue : ObservedMeasure
measure = RecipientValueMeasure
measurement = XTrailerPropositionValueMeasurement
observation = OverallScenarioObservation
ObservedXTrailerPropositionSatisfaction : ObservedMeasure
measure = RecipientSatisfactionMeasure
measurement = XtrailerPropositionSatisfactionMeasurement
observation = OverallScenarioObservation
ObservedFastInnovationWeightedSatisfaction : ObservedMeasure
measure = FastInnovationWeightedSatisfactionMeasure
measurement = FastInnovationWeightedSatisfactionMeasurement
observation = OverallScenarioObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
: ObservationObservedMeasure
OverallScenarioObservation : Observation
observedMeasure = ObservedXTrailerPropositionSatisfaction, ObservedXTrailerPropositionValue, ObservedFairPriceWeightedSatisfaction, ObservedLateSpecFreezeWeightedSatisfaction, ObservedFastInnovationWeightedSatisfaction, ObservedMarketDrivenDesignWeightedSatisfaction, ObservedProductLT, ObservedLateSpecFreezeSatisfaction, ObservedLateSpecFreezeWeight, ObservedSalesVolume, ObservedOrderInterval, ObservedHistoricOrderInterval, ObservedMargin, ObservedProfit
smmModel = MyCo specific SMM model
OverallScenario : Scenario
containingModel = MyCompanyModel
contextCollaboration = XTrailerBusiness
contextObservation = OverallScenarioObservation
contextStore = Ideas, ApprovedIdeas, FinalReleases, Orders
delegationtContext = ManageInnovationDelCntxt, ManageFulfillmentDelCntxt
: ContextObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
: ObservationObservedMeasure
ObservedFairPriceWeightedSatisfaction : ObservedMeasure
measure = FairPriceWeightedSatisfactionMeasure
measurement = FairPriceWeightedSatisfactionMeasurement
observation = OverallScenarioObservation
ObservedLateSpecFreezeWeightedSatisfaction : ObservedMeasure
measure = LateSpecFreezeWeightedSatisfactionMeasure
measurement = LateSpecFreezeWeightedSatisfactionMeasurement
observation = OverallScenarioObservation
ObservedMarketDrivenDesignWeightedSatisfaction : ObservedMeasure
measure = MarketDrivenDesignWeightedSatisfactionMeasure
measurement = MarketDrivenDesignWeightedSatisfactionMeasurement
observation = OverallScenarioObservation
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
FairPriceWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedFairPriceWeightedSatisfaction
LateSpecFreezeWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurement1To = LateSpecFreezeSatisfactionMeasurement
baseMeasurement2To = LateSpecFreezeWeightMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedLateSpecFreezeWeightedSatisfaction
MarketDrivenDesignWeightedSatisfactionMeasurement : BinaryMeasurement
baseMeasurementFrom = XTrailerPropositionValueMeasurement
observedMeasure = ObservedMarketDrivenDesignWeightedSatisfaction

Figure 123. Overall scenario observation measurements, second part (Objects)

Measurement of proposition value, related customer satisfaction, as well as, measurement of value per each component of the value proposition, its related customer satisfaction and relative weight of importance to the customer, are important in that they may guide the provider of the value proposition, here the manufacturer in the business network, in establishing priorities for improvement of capabilities that contribute to value delivery. Value delivery models support value-driven innovation of the business system.

The measurement of customer satisfaction with the value proposition can be taken further to analyze the impact of it on e.g. the sales volume of trailers in the XTrailer business network. This piece of analysis, based on structured models, would be very relevant to support broader what-if calculations and simulations of the business system, and is an example of how impact of value can be measured and analyzed. Impact of customer satisfaction on sales volume will further impact profit that the manufacturer realizes in the business network. Value-driven innovation of the business would be short-sighted, when impact of innovations on profit would not be taken into account. One decision might be to e.g. improve existing or develop new capabilities. Another decision would be to abandon a capability, or even withdraw from a business network, and therewith abandon a business model.

We will demonstrate below how the impact of customer satisfaction on sales volume can be modeled. We will also demonstrate the modeling of how the manufacturing party in the business network realizes profit. The latter aspect comes into what business model frameworks use to indicate as “profit formula”.

The objects in the object diagram in Figure 124 represent the modeling of how customer satisfaction with the XTrailer Proposition impacts sales volume of trailers in the XTrailer business network. Consider that the transporter provides “repetitive business”, as value, in reward, to the manufacturer. Figure 124 shows the corresponding value add object on the output port of the Buy Product activity. This activity is performed by the transporter in the business network. This value can be measured via the measurement of “sales volume”. The value add object has a measured characteristic “Sales Volume” accordingly, which is associated with the sales volume measurement object.

The sales volume measurement is a rescaled measurement, which rescales the underlying order interval measurement, associated with the order interval measured characteristic of the Buy Product activity. As the object diagram in Figure 124 shows, this order interval characteristic serves as “recurrence interval” to the Buy Product activity. Note also that a recurrence interval, especially when its measure’s operation denotes a sampler from a stochastic distribution, plays an important role in discrete event simulation. The measure, used to measure sales volume, has “pieces per year” as unit, and rescales from the order interval measure according to the formula “365 * 24 / order interval”, the order interval measures having “hours” as unit. These details can be looked up in the corresponding measure objects in the object diagram of the measure library as presented earlier in Figure 105. The measures that are used for these measurements in Figure 124 are the ones that are referenced by their related observed measure objects as contained in the object diagram in Figure 126 below. As sales volume is derived from the order interval, we can model the impact of customer satisfaction on sales volume, by modeling its impact on the order interval. It is reasonable to model the order interval measure as a function of the historic order interval, being the interval as measured over the recent past period, and the measure of customer satisfaction, being the measure of satisfaction of the recipient of the value proposition. This is indicated in the object diagram in Figure 124 by the Order Interval Measurement, having both the XTrailer Proposition Satisfaction Measurement and the Historic Order Interval Measurement as base measurements. Corresponding measures, related via their observed measure objects, can be found in the measure library diagram in Figure 105.

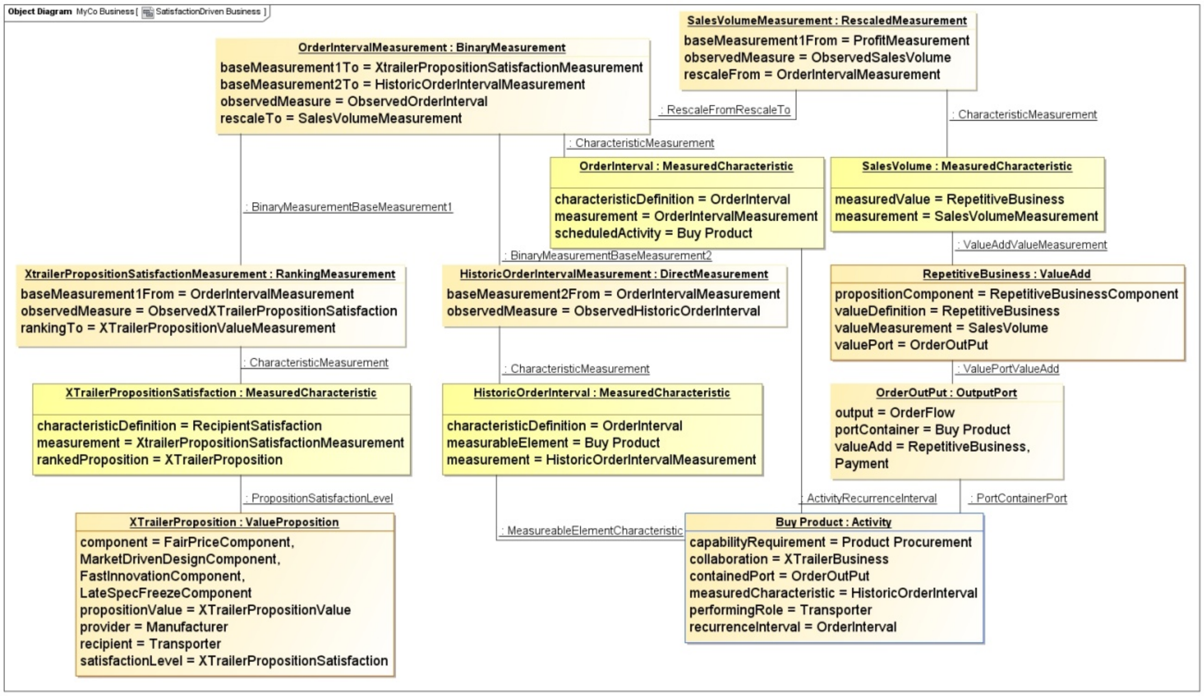

Object Diagram MyCo Business [SatisfactionDriven Business]
OrderIntervalMeasurement : BinaryMeasurement
baseMeasurement1To = XtrailerPropositionSatisfactionMeasurement
baseMeasurement2To = HistoricOrderIntervalMeasurement
observedMeasure = ObservedOrderInterval
rescaleTo = SalesVolumeMeasurement
SalesVolumeMeasurement : RescaledMeasurement
baseMeasurement1From = ProfitMeasurement
observedMeasure = ObservedSalesVolume
rescaleFrom = OrderIntervalMeasurement
: RescaleFromRescaleTo
: CharacteristicMeasurement
: CharacteristicMeasurement
OrderInterval : MeasuredCharacteristic
characteristicDefinition = OrderInterval
measurement = OrderIntervalMeasurement
scheduledActivity = Buy Product
SalesVolume : MeasuredCharacteristic
measuredValue = RepetitiveBusiness
measurement = SalesVolumeMeasurement
: BinaryMeasurementBaseMeasurement1
: BinaryMeasurementBaseMeasurement2
: ValueAddValueMeasurement
XtrailerPropositionSatisfactionMeasurement : RankingMeasurement
baseMeasurement1From = OrderIntervalMeasurement
observedMeasure = ObservedXTrailerPropositionSatisfaction
rankingTo = XTrailerPropositionValueMeasurement
HistoricOrderIntervalMeasurement : DirectMeasurement
baseMeasurement2From = OrderIntervalMeasurement
observedMeasure = ObservedHistoricOrderInterval
RepetitiveBusiness : ValueAdd
propositionComponent = RepetitiveBusinessComponent
valueDefinition = RepetitiveBusiness
valueMeasurement = SalesVolume
valuePort = OrderOutPut
: CharacteristicMeasurement
: CharacteristicMeasurement
: ValuePortValueAdd
XTrailerPropositionSatisfaction : MeasuredCharacteristic
characteristicDefinition = RecipientSatisfaction
measurement = XtrailerPropositionSatisfactionMeasurement
rankedProposition = XTrailerProposition
HistoricOrderInterval : MeasuredCharacteristic
characteristicDefinition = OrderInterval
measurableElement = Buy Product
measurement = HistoricOrderIntervalMeasurement
OrderOutPut : OutputPort
output = OrderFlow
portContainer = Buy Product
valueAdd = RepetitiveBusiness, Payment
: PropositionSatisfactionLevel
: ActivityRecurrenceInterval
: PortContainerPort
XTrailerProposition : ValueProposition
component = FairPriceComponent, MarketDrivenDesignComponent, FastInnovationComponent, LateSpecFreezeComponent
propositionValue = XTrailerPropositionValue
provider = Manufacturer
recipient = Transporter
satisfactionLevel = XTrailerPropositionSatisfaction
: MeasureableElementCharacteristic
Buy Product : Activity
capabilityRequirement = Product Procurement
collaboration = XTrailerBusiness
containedPort = OrderOutPut
measuredCharacteristic = HistoricOrderInterval
performingRole = Transporter
recurrenceInterval = OrderInterval

Figure 124. Measurement of impact of customer satisfaction on sales volume (objects)

The objects in the object diagram in Figure 125 represent the structure to calculate profit, in a straightforward way. This is the profit that the manufacturer makes in the XTrailer business network. The Business Networks metamodel diagram, as provided earlier in Figure 15, defines “profit” as a property of a providing party role in the business network. Profit is measured by a binary measure, which takes the product of margin and sales volume as its base measures. The object diagram in Figure 126 contains the observed measure objects that associate with the measurements in Figure 125. The observed measures refer to the corresponding measures in the library. Note for instance that the profit measure, as represented in the measure library diagram in Figure 105, has functor “product”, and “euros” as unit.

Margin is measured via a direct measure. The margin measurement is stored as measurement of the margin measured characteristic of the Final Releases store. Note that this store is the location in which the final release information of the product resides, and as it relates to the Overall Scenario as its context, margin measurement is global to the scenario.

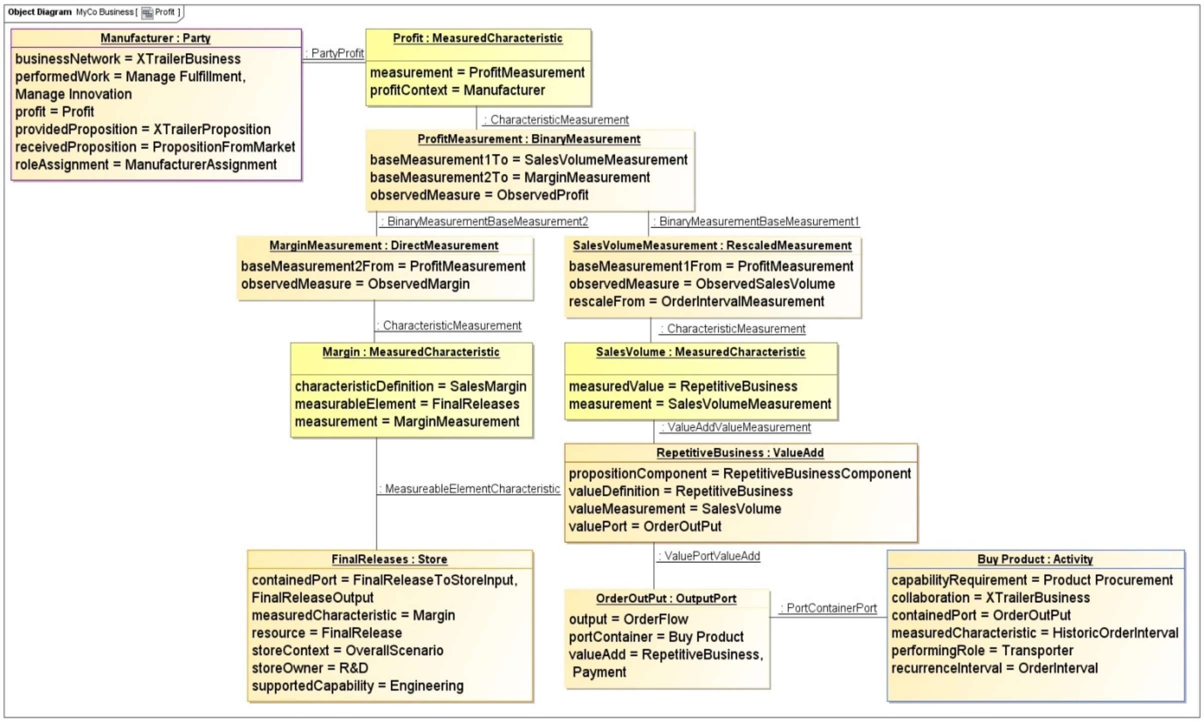

Object Diagram MyCo Business [Profit]
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
: PartyProfit
Profit : MeasuredCharacteristic
measurement = ProfitMeasurement
profitContext = Manufacturer
: CharacteristicMeasurement
ProfitMeasurement : BinaryMeasurement
baseMeasurement1To = SalesVolumeMeasurement
baseMeasurement2To = MarginMeasurement
observedMeasure = ObservedProfit
: BinaryMeasurementBaseMeasurement2
: BinaryMeasurementBaseMeasurement1
MarginMeasurement : DirectMeasurement
baseMeasurement2From = ProfitMeasurement
observedMeasure = ObservedMargin
SalesVolumeMeasurement : RescaledMeasurement
baseMeasurement1From = ProfitMeasurement
observedMeasure = ObservedSalesVolume
rescaleFrom = OrderIntervalMeasurement
: CharacteristicMeasurement
: CharacteristicMeasurement
Margin : MeasuredCharacteristic
characteristicDefinition = SalesMargin
measurableElement = FinalReleases
measurement = MarginMeasurement
SalesVolume : MeasuredCharacteristic
measuredValue = RepetitiveBusiness
measurement = SalesVolumeMeasurement
: ValueAddValueMeasurement
RepetitiveBusiness : ValueAdd
propositionComponent = RepetitiveBusinessComponent
valueDefinition = RepetitiveBusiness
valueMeasurement = SalesVolume
valuePort = OrderOutPut
: MeasureableElementCharacteristic
: ValuePortValueAdd
FinalReleases : Store
containedPort = FinalReleaseToStoreInput, FinalReleaseOutput
measuredCharacteristic = Margin
resource = FinalRelease
storeContext = OverallScenario
storeOwner = R&D
supportedCapability = Engineering
OrderOutPut : OutputPort
output = OrderFlow
portContainer = Buy Product
valueAdd = RepetitiveBusiness, Payment
: PortContainerPort
Buy Product : Activity
capabilityRequirement = Product Procurement
collaboration = XTrailerBusiness
containedPort = OrderOutPut
measuredCharacteristic = HistoricOrderInterval
performingRole = Transporter
recurrenceInterval = OrderInterval

Figure 125. Measurement of profit (objects)

Like various observed measures as encountered earlier, also the observed measures, for the measurements of sales volume, order interval, historic order interval, margin and profit, are contained in the observation of the overall scenario, as indicated in the object diagram in Figure 126.

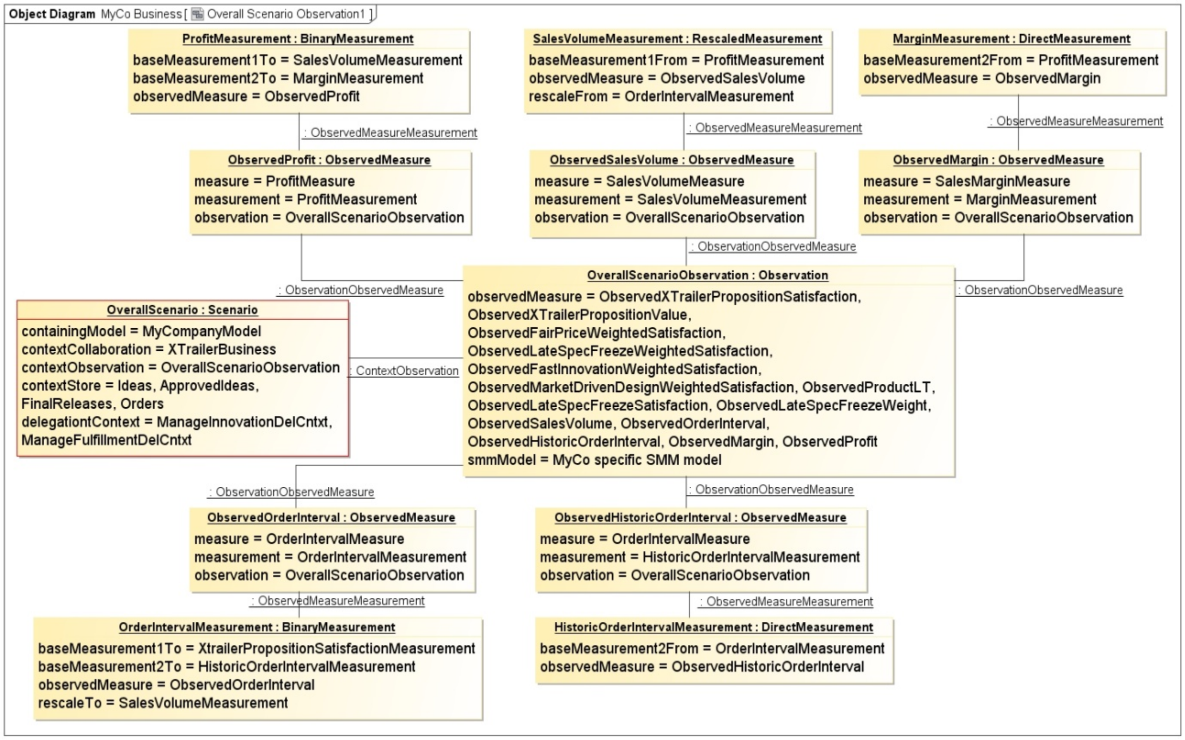

Object Diagram MyCo Business [Overall Scenario Observation1]
ProfitMeasurement : BinaryMeasurement
baseMeasurement1To = SalesVolumeMeasurement
baseMeasurement2To = MarginMeasurement
observedMeasure = ObservedProfit
SalesVolumeMeasurement : RescaledMeasurement
baseMeasurement1From = ProfitMeasurement
observedMeasure = ObservedSalesVolume
rescaleFrom = OrderIntervalMeasurement
MarginMeasurement : DirectMeasurement
baseMeasurement2From = ProfitMeasurement
observedMeasure = ObservedMargin
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
ObservedProfit : ObservedMeasure
measure = ProfitMeasure
measurement = ProfitMeasurement
observation = OverallScenarioObservation
ObservedSalesVolume : ObservedMeasure
measure = SalesVolumeMeasure
measurement = SalesVolumeMeasurement
observation = OverallScenarioObservation
ObservedMargin : ObservedMeasure
measure = SalesMarginMeasure
measurement = MarginMeasurement
observation = OverallScenarioObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
: ObservationObservedMeasure
OverallScenarioObservation : Observation
observedMeasure = ObservedXTrailerPropositionSatisfaction,
ObservedXTrailerPropositionValue,
ObservedFairPriceWeightedSatisfaction,
ObservedLateSpecFreezeWeightedSatisfaction,
ObservedFastInnovationWeightedSatisfaction,
ObservedMarketDrivenDesignWeightedSatisfaction, ObservedProductLT,
ObservedLateSpecFreezeSatisfaction, ObservedLateSpecFreezeWeight,
ObservedSalesVolume, ObservedOrderInterval,
ObservedHistoricOrderInterval, ObservedMargin, ObservedProfit
smmModel = MyCo specific SMM model
OverallScenario : Scenario
containingModel = MyCompanyModel
contextCollaboration = XTrailerBusiness
contextObservation = OverallScenarioObservation
contextStore = Ideas, ApprovedIdeas,
FinalReleases, Orders
delegationtContext = ManageInnovationDelCntxt,
ManageFulfillmentDelCntxt
: ContextObservation
: ObservationObservedMeasure
: ObservationObservedMeasure
ObservedOrderInterval : ObservedMeasure
measure = OrderIntervalMeasure
measurement = OrderIntervalMeasurement
observation = OverallScenarioObservation
ObservedHistoricOrderInterval : ObservedMeasure
measure = OrderIntervalMeasure
measurement = HistoricOrderIntervalMeasurement
observation = OverallScenarioObservation
: ObservedMeasureMeasurement
: ObservedMeasureMeasurement
OrderIntervalMeasurement : BinaryMeasurement
baseMeasurement1To = XtrailerPropositionSatisfactionMeasurement
baseMeasurement2To = HistoricOrderIntervalMeasurement
observedMeasure = ObservedOrderInterval
rescaleTo = SalesVolumeMeasurement
HistoricOrderIntervalMeasurement : DirectMeasurement
baseMeasurement2From = OrderIntervalMeasurement
observedMeasure = ObservedHistoricOrderInterval

Figure 126. Overall scenario observation measurements, third part (Objects)

As suggested earlier, in the XTrailer business network, the transporter party is assumed to provide value in return to the manufacturer. A value proposition can be defined in relation to this value as well. An example of its structure is represented in the object diagram in Figure 127. It consists of three components, articulating three values, one of which is the value of “Repetitive business” as was introduced earlier, during the discussion of measurement of sales volume (see Figure 124 above). The other values are “Payment”, being an obvious one, and “Feedback”. “Feedback” is the value that is conveyed based on the ideas that the transporter provides to the manufacturer. As Figure 127 indicates, “Repetitive Business” and “Payment” are conveyed based on the orders that the transporter places with the manufacturer. Note that a payment flow could have been modeled separately as well, but it is often not wanted to include more detail than is required for the analysis at hand.

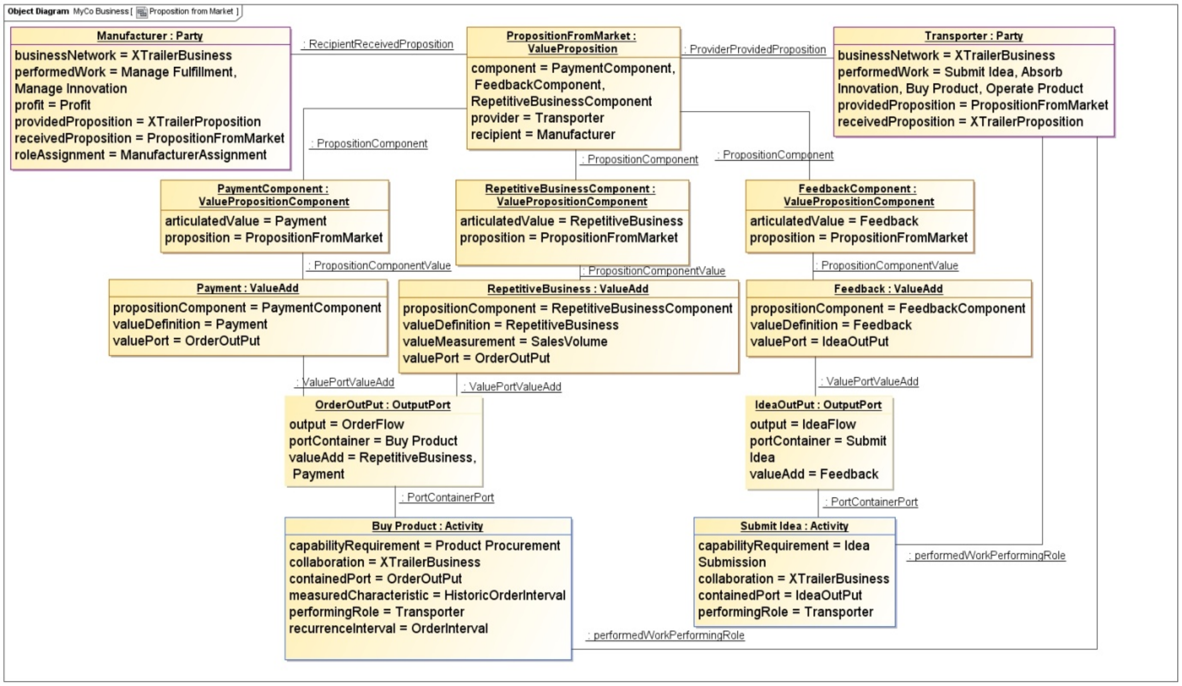

Object Diagram MyCo Business [Proposition from Market]
Manufacturer : Party
businessNetwork = XTrailerBusiness
performedWork = Manage Fulfillment, Manage Innovation
profit = Profit
providedProposition = XTrailerProposition
receivedProposition = PropositionFromMarket
roleAssignment = ManufacturerAssignment
: RecipientReceivedProposition
PropositionFromMarket : ValueProposition
component = PaymentComponent, FeedbackComponent, RepetitiveBusinessComponent
provider = Transporter
recipient = Manufacturer
: ProviderProvidedProposition
Transporter : Party
businessNetwork = XTrailerBusiness
performedWork = Submit Idea, Absorb Innovation, Buy Product, Operate Product
providedProposition = PropositionFromMarket
receivedProposition = XTrailerProposition
: PropositionComponent
: PropositionComponent
: PropositionComponent
PaymentComponent : ValuePropositionComponent
articulatedValue = Payment
proposition = PropositionFromMarket
RepetitiveBusinessComponent : ValuePropositionComponent
articulatedValue = RepetitiveBusiness
proposition = PropositionFromMarket
FeedbackComponent : ValuePropositionComponent
articulatedValue = Feedback
proposition = PropositionFromMarket
: PropositionComponentValue
: PropositionComponentValue
: PropositionComponentValue
Payment : ValueAdd
propositionComponent = PaymentComponent
valueDefinition = Payment
valuePort = OrderOutPut
RepetitiveBusiness : ValueAdd
propositionComponent = RepetitiveBusinessComponent
valueDefinition = RepetitiveBusiness
valueMeasurement = SalesVolume
valuePort = OrderOutPut
Feedback : ValueAdd
propositionComponent = FeedbackComponent
valueDefinition = Feedback
valuePort = IdeaOutPut
: ValuePortValueAdd
: ValuePortValueAdd
: ValuePortValueAdd
OrderOutPut : OutputPort
output = OrderFlow
portContainer = Buy Product
valueAdd = RepetitiveBusiness, Payment
IdeaOutPut : OutputPort
output = IdeaFlow
portContainer = Submit Idea
valueAdd = Feedback
: PortContainerPort
: PortContainerPort
Buy Product : Activity
capabilityRequirement = Product Procurement
collaboration = XTrailerBusiness
containedPort = OrderOutPut
measuredCharacteristic = HistoricOrderInterval
performingRole = Transporter
recurrenceInterval = OrderInterval
Submit Idea : Activity
capabilityRequirement = Idea Submission
collaboration = XTrailerBusiness
containedPort = IdeaOutPut
performingRole = Transporter
: performedWorkPerformingRole
: performedWorkPerformingRole

Figure 127. Value proposition as provided by the transporter and received by the manufacturer (objects)

Figure 91 suggests that activities do not only create, but also consume value. This is partly obvious, as deliverables, that flow from activity to activity or from store to activity, convey values. It is also possible that value consumption is defined with more precision, by defining how measurements of values received, influence measurements of values created. As a simplistic example, consider the value of "Feedback", as part of the value proposition in Figure 127. According to the measurements and measurement influences mockup in Figure 104, its associated value measurement of "idea volume" influences the "idea productization ratio", being the measurement of a value that is received by the market. When the volume of ideas would just rise, and activities would not perform adequately (i.e. do not properly consume the value), the value that is expected to be created would drop down. Activities will have to optimally "consume" feedback and apply it effectively, in order to create the value that is required. Note that this part of the use case has not been worked out in object models. More refined examples would be possible in the context of the use case as well. Imagine how the consumption of "feedback" could lead to measurable lower cost of resources as well as measurable shorter duration of innovation.

A business network is healthy when each party realizes a "gain" (value received is more than value provided). Though it might be an explicit strategy, that a party might "loose" (not gain) in one business network, whereas it gains in others. For instance, the same market or company might be recipient party (customer) in multiple business networks, some of which have low return, but others are highly profitable. It might not be a good strategy to eliminate the "low return" business networks, as the corresponding market or company might only stay in business when a "complete offering" is provided.

There are different ways to measure the "gain" (or lack of it) in a business network (or "business model"). The metamodel diagram of Business Networks in Figure 15 defines two measured characteristics, to measure "gain" per party:

- Profit. Above we discussed modeling and measurement of profit. A profit is realized when the price that the recipient pays is higher than the provider's cost.
- Value margin. Value margin is the margin that the recipient party in a business network might realize and can be defined as the difference between proposition value of value propositions received, and "fair market value", or as the difference between proposition value and the price that is paid to providers (providing parties in the same business network) of the corresponding value propositions.

Profit is defined from the perspective of the provider of a value proposition. Value margin is defined from the perspective of the recipient of it. Profit alone is not a sufficient indicator for healthiness of the business network ("business model"). It is possible that a party realizes profit, but is observing a negative value margin. For instance, when the recipient of the value proposition associates "equivalent economic value" to a value proposition, lower than the "fair market price", or in other words: when the recipient party judges price to pay as higher than the value it receives.

Measurement of value margin requires that proposition value is measured in economic (monetary) terms. As in the XTrailer use case example, the property "proposition value" was measured in non-economic terms, a custom-modeled measured characteristic would be required to express this value. And as such measured characteristics as "fair market value" (of a value proposition, and possibly its components) and "price" (of the underlying deliverables) are not enforced by the meta-model, custom-modeled measured characteristics would be required for these also. The VDML metamodel supports modeling of such custom measured characteristics of measureable elements, such as value proposition, value proposition component, business item and store (see VDML elements metamodel diagram in Figure 61).

So-far we explicitly analyzed customer value proposition, and value proposition as received to from the customer or market. It would also be possible to model other parties, and value propositions in relation to these. It would be useful to e.g. consider "enterprise value proposition", articulating such values as "fast innovation", "short fulfillment lead time", "high profitability", "low cost fulfillment", etc.

It would even be possible to model a 2-level business network, as indicated in Figure 128, where the top-level business network represents "the business", or "business model innovation", which contains party roles that are filled by the "actual" business networks, such as the XTrailer business network, which serve as "business models" to "the business".

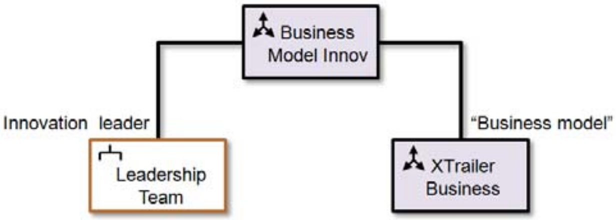


Figure 128. Two-level business network structure

The top-level business network may then also contain a leadership role, filled by the leadership team of the organization. The leadership provides "effort" (intangible) to the "business models", which provide "business improvement" (intangible) back to the leadership, conveying such values as just mentioned, which maybe be articulated by an "enterprise value proposition", provided by business model roles to the leadership role.

3.1.6 Model data organization and re-use

So-far we defined what a value delivery model is, we discussed its purpose, and the many model elements that are involved when doing value delivery modeling. But the metamodel diagrams so-far did not explicitly show an element called "value delivery model". As the discussions have clarified, a value delivery model might involved multiple collaborations, libraries and scenarios. Each of these contain various detailed model elements, but the set of collaborations, libraries and scenarios themselves, that are assumed to represent a value delivery model, are not contained in a single model object. For this reason VDML contains a "value delivery model" class, that serves as the top-level container of all model elements that are specific to the model.

As the earlier presented metamodel in Figure 102 indicates, SMM specifies a similar class, called "SMM model", containing measure libraries and/or observations.

The objects in the object diagram in Figure 129 represent the "MyCompany Model", which is the value delivery model in the XTrailer use case. It shows how the various collaborations, libraries and scenarios are contained in it, as well as how it contains the SMM model that contains the various observations that the value delivery model uses.

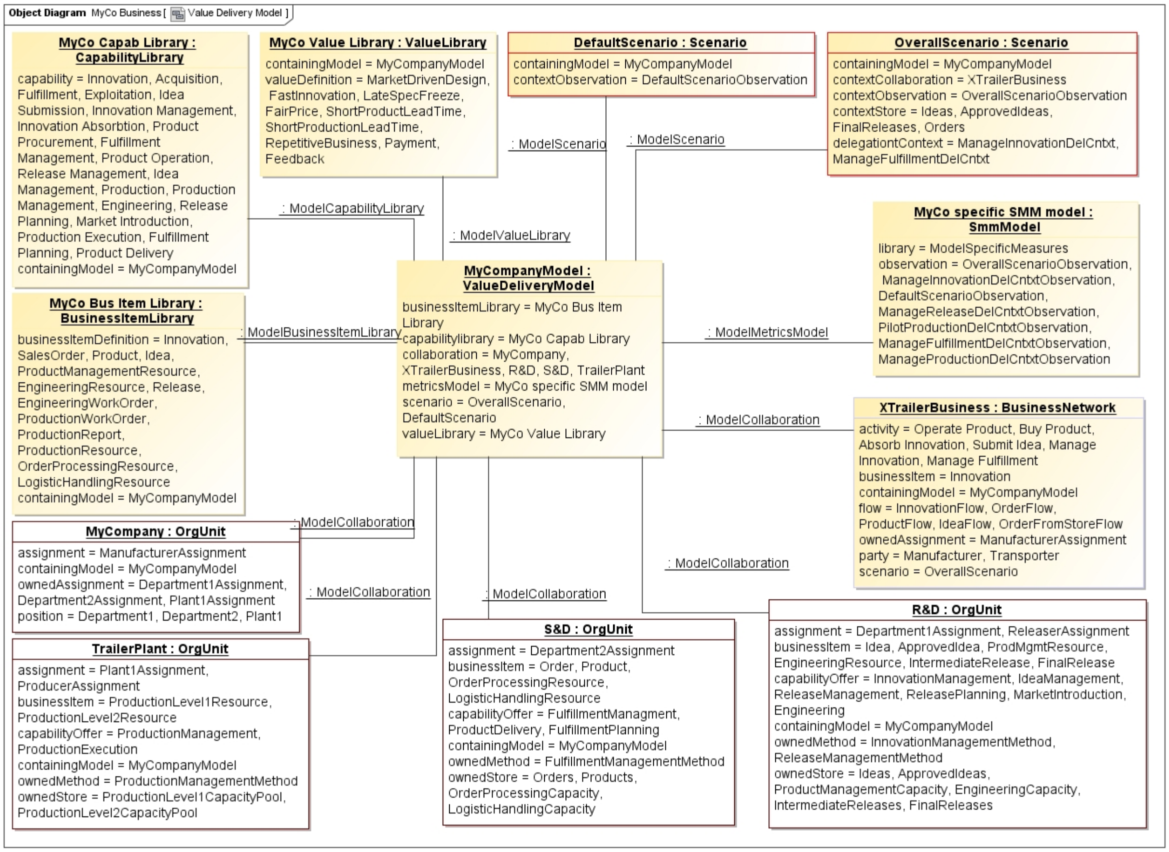

Object Diagram MyCo Business [Value Delivery Model]
MyCo Capab Library : CapabilityLibrary
capability = Innovation, Acquisition, Fulfillment, Exploitation, Idea Submission, Innovation Management, Innovation Absorbtion, Product Procurement, Fulfillment Management, Product Operation, Release Management, Idea Management, Production, Production Management, Engineering, Release Planning, Market Introduction, Production Execution, Fulfillment Planning, Product Delivery
containingModel = MyCompanyModel
MyCo Value Library : ValueLibrary
containingModel = MyCompanyModel
valueDefinition = MarketDrivenDesign, FastInnovation, LateSpecFreeze, FairPrice, ShortProductLeadTime, ShortProductionLeadTime, RepetitiveBusiness, Payment, Feedback
DefaultScenario : Scenario
containingModel = MyCompanyModel
contextObservation = DefaultScenarioObservation
OverallScenario : Scenario
containingModel = MyCompanyModel
contextCollaboration = XTrailerBusiness
contextObservation = OverallScenarioObservation
contextStore = Ideas, ApprovedIdeas, FinalReleases, Orders
delegationtContext = ManageInnovationDelCntxt, ManageFulfillmentDelCntxt
: ModelScenario
: ModelScenario
: ModelCapabilityLibrary
: ModelValueLibrary
MyCo specific SMM model : SmmModel
library = ModelSpecificMeasures
observation = OverallScenarioObservation, ManageInnovationDelCntxtObservation, DefaultScenarioObservation, ManageReleaseDelCntxtObservation, PilotProductionDelCntxtObservation, ManageFulfillmentDelCntxtObservation, ManageProductionDelCntxtObservation
MyCompanyModel : ValueDeliveryModel
businessItemLibrary = MyCo Bus Item Library
capabilitylibrary = MyCo Capab Library
collaboration = MyCompany, XTrailerBusiness, R&D, S&D, TrailerPlant
metricsModel = MyCo specific SMM model
scenario = OverallScenario, DefaultScenario
valueLibrary = MyCo Value Library
MyCo Bus Item Library : BusinessItemLibrary
businessItemDefinition = Innovation, SalesOrder, Product, Idea, ProductManagementResource, EngineeringResource, Release, EngineeringWorkOrder, ProductionWorkOrder, ProductionReport, ProductionResource, OrderProcessingResource, LogisticHandlingResource
containingModel = MyCompanyModel
: ModelBusinessItemLibrary
: ModelMetricsModel
: ModelCollaboration
XTrailerBusiness : BusinessNetwork
activity = Operate Product, Buy Product, Absorb Innovation, Submit Idea, Manage Innovation, Manage Fulfillment
businessItem = Innovation
containingModel = MyCompanyModel
flow = InnovationFlow, OrderFlow, ProductFlow, IdeaFlow, OrderFromStoreFlow
ownedAssignment = ManufacturerAssignment
party = Manufacturer, Transporter
scenario = OverallScenario
: ModelCollaboration
MyCompany : OrgUnit
assignment = ManufacturerAssignment
containingModel = MyCompanyModel
ownedAssignment = Department1Assignment, Department2Assignment, Plant1Assignment
position = Department1, Department2, Plant1
: ModelCollaboration
: ModelCollaboration
: ModelCollaboration
R&D : OrgUnit
assignment = Department1Assignment, ReleaserAssignment
businessItem = Idea, ApprovedIdea, ProdMgmtResource, EngineeringResource, IntermediateRelease, FinalRelease
capabilityOffer = InnovationManagement, IdeaManagement, ReleaseManagement, ReleasePlanning, MarketIntroduction, Engineering
containingModel = MyCompanyModel
ownedMethod = InnovationManagementMethod, ReleaseManagementMethod
ownedStore = Ideas, ApprovedIdeas, ProductManagementCapacity, EngineeringCapacity, IntermediateReleases, FinalReleases
TrailerPlant : OrgUnit
assignment = Plant1Assignment, ProducerAssignment
businessItem = ProductionLevel1Resource, ProductionLevel2Resource
capabilityOffer = ProductionManagement, ProductionExecution
containingModel = MyCompanyModel
ownedMethod = ProductionManagementMethod
ownedStore = ProductionLevel1CapacityPool, ProductionLevel2CapacityPool
S&D : OrgUnit
assignment = Department2Assignment
businessItem = Order, Product, OrderProcessingResource, LogisticHandlingResource
capabilityOffer = FulfillmentManagment, ProductDelivery, FulfillmentPlanning
containingModel = MyCompanyModel
ownedMethod = FulfillmentManagementMethod
ownedStore = Orders, Products, OrderProcessingCapacity, LogisticHandlingCapacity

Figure 129. Value delivery model as top-level model elements container (objects)

Note that the “MyCo specific SMM model”, containing the observations, is contained in the value delivery model itself, whereas the “Generic Library Model” (see Figure 114) model is not. The latter library is meant as generic library, re-usable across potentially many value delivery models, and is hence not specific to the “MyCompany Model”. Note however that we did not consistently apply this rule to all libraries. The capability, business item and value libraries, though they are assumably re-usable, are directly contained in the “MyCompany Model”. In a real world modeling situation there would be more conscious library maintenance, keeping model specific libraries separate from generic libraries.

The objects in Figure 129 are instances of classes in the value delivery models metamodel diagram in Figure 130.

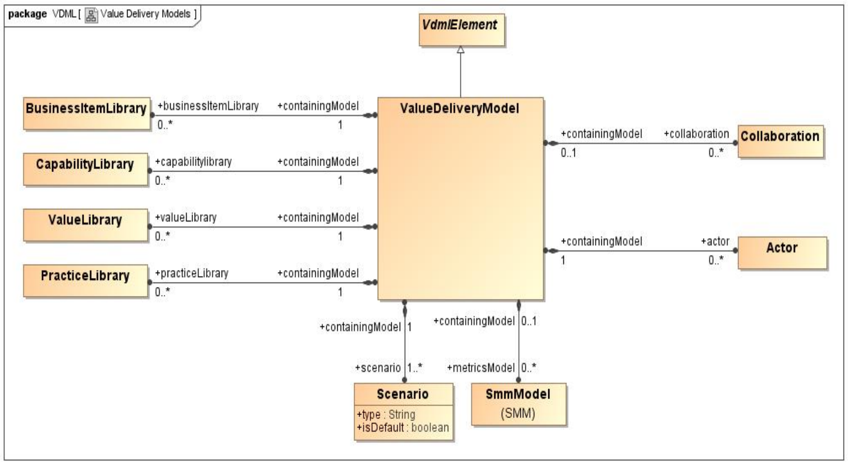


Figure 130. Value delivery models (as top-level containers) metamodel

Packaging value delivery modeling elements into value delivery models, does not imply that the entire business system has to be defined by a single value delivery model. As there might be different people (business analysts) responsible for modeling and analysis of different sub-scopes of the business system, and models of different sub-scopes of the business system might have their own life cycles, versions, etc. It is possible therefore that elements in one value delivery model, representing a part of the business system, are associated with elements in other value delivery models.

Various associations in VDML can be across value delivery model boundaries. Examples of these associations are:

- Libraries. One value delivery model might use elements from libraries that are contained in other value delivery models and/or SMM models.
- Collaborations. A delegation context in one value delivery model might relate to a collaboration in another.
- Stores. A deliverable flow in one value delivery model might associate to a store, via one of its ports, in another value delivery model

3.2 VDML advanced elements

In 3.1 we discussed the basic structure of value delivery models, based on the XTrailer use case. We briefly mentioned some advanced elements, but didn't apply them to the use case. These elements have been incorporated in the metamodel, in particular to support model-based business simulation.

Automated simulation support is not an achievable outcome of NEFFICS. It will require ongoing research beyond NEFFICS. But as, ultimately, VDML should support simulation, it is required to assure that the metamodel is robust. Making VDML "simulation-complete" in a later stage should not require fundamental refactoring of the metamodel. It is required therefore that the metamodel is designed with its application to simulation in mind, and that key elements to support simulation, are covered by it.

Most of these elements aren't exclusive to simulation, but are useful for other purposes as well, such as:

- Improving the opportunities for measurements-based manual analysis based on VDML models.
- Enable more extensive transformation of VDML models, or parts of it, to executable models, such as application models and process models, and enable resource management in real-world implementation (in Table 5 we refer to this as "implementation")
- Extend the business know-how that can be captured in VDML models.

Table 5 provides an overview of advanced elements, whereby we indicate per element what its purpose is in relation to simulation, as well as how it may serve other purpose. For definitions and detailed semantics of the elements the reader can refer to the VDML 1.0 specification.

Most of the advanced elements as listed in Table 5 are required in particular for Discrete Event Simulation. In 2.4 we discussed the various types of simulation. As far as VDML is concerned, Monte Carlo simulation would not require more than its basic structure provides. Monte Carlo simulation does require stochastic enabling of SMM though. In **Error! Reference source not found.**) we will indicate how a revision of SMM will support this. More research is required on the relationship between VDML and System Dynamics simulation, although a basic fit is provided by the basic structure of VDML, and VDML-as-integrated-with-SMM. The basic patterns of System Dynamic, such as "stock and flow" and "causal loop" can be recognized in basic patterns in VDML (and SMM), such as decoupling of activity networks by stores, and the measurement dependencies in SMM (see Figure 104), though VDML (and SMM) provide more refined structures. In **Error! Reference source not found.**) we will indicate how a revision of SMM will provide support for System Dynamics-style causal loops (see Figure 104).

Element	Reference	Simulation purpose (mainly Discrete Event)	General purpose
Calendars of participants and pools	Collaborations metamodel in Figure 16	Determine resource availability	Implementation
	Stores metamodel in Figure 36		
Business item being "fungible", "sharable"	Business Items metamodel in Figure 18	Proper tracking of business item instances through stores;	General business know-how
Input Port correlation expression	Port Containers metamodel in Figure 45		
Release Control, as defined by Capability Offer, and as imposed by Scenario	Organizations and capabilities metamodel in Figure 35	Determine priority of work to be performed by activities to which the capability offer is applied	Implementation
	Scenario and Analysis Context metamodel in Figure 51		
Inventory level of Store (and Pool) and pool size of Pool	Stores metamodel in Figure 36	Determine resource availability	Manual analysis based on average (i.e. non-stochastic) measurements
Port offset	Port Containers metamodel in Figure 45	Duration calculations	Manual analysis based on average (i.e. non-stochastic) measurements; in particular network planning and critical path analysis; Implementation (understanding whether e.g. an input arrives as intermediate event)
Duration of Activity, and Resource use	Activities metamodel in Figure 65		
Duration of Store	Stores metamodel in Figure 36		
Duration of Deliverable Flow	Deliverable Flows metamodel in Figure 17		
Recurrence interval of Activity	Activities metamodel in Figure 65	Proper generation and handling of business item instances throughout networks of	Manual analysis based on average (i.e. non-stochastic) measurements;
Port batch size	Port Containers		

	metamodel in Figure 45	activities and stores	
Port condition		Proper routing of business item instances	Implementation (flow-based decisions)
Scenario horizon	Scenario and Analysis Context metamodel in Figure 51	Simulation horizon	Manual analysis based on average (i.e. non-stochastic) measurements
Specification of resource (per Resource Use) as “input driven” versus “output driven”	Activities metamodel in Figure 65	Amongst others specifying the proper coordination between activities and stores, at the resource instance level	General business know-how
Resource Use quantity			
Specification of resource (per Resource Use) as “exclusive” versus “inclusive”		Proper handling of alternative resources, at the resource instance level	
Resource Use condition			
Ordered set of resources per Resource Use			
Specification of resource (per Resource Use) as “consumed” versus “used”		Proper handling of business item instances by activities	

Table 5. Some advanced VDML elements

Error! Reference source not found.) lists some functional changes that will be applied to the November 2012 VDML submission, and which have not been applied in this document.

4 How VDML addresses requirements

4.1 How VDML addresses specific requirements

Table 6 indicates, per each of the requirements as described in 2.2 and 2.3.8, how VDML addresses the particular requirement, based on the explanation of VDML in 3.1 and 3.2. A detailed mapping of elements of business model frameworks (as discussed in 2.2.2) will be provided in **Error! Reference source not found.**.

Value Delivery Model			
"Motivated by Business Values"			
		Requirement	**Resolution (VDML)**
	Value identification		
		Identify any form of value, financial and non-financial, tangible and non-tangible	Values, defined as Value Add objects (and "typed" by value definitions), conveyed by deliverable flows, tangible or non-tangible
		Definable types of value	Value definitions in Value Library
	Value flow, intra and inter-enterprise		
		Value creation, distribution (or "delivery") and consumption	Activities use / consume resources and produce deliverables, received and sent by deliverable flows, which also convey associated values. Measurements of values used or consumed can influence measurements of values produced.
		By and between an unlimited number of roles, attached to an unlimited number of organizations	Roles perform activities, and may be assigned to different organizations (Org Units), or are part of different organizations or capability methods of different organizations.
		Link value flow within the business to value flow with customers and business partners	Flows of deliverables and associated values can occur between activities and/or stores, whereby different organizations (or parts of them) perform these activities and own these stores
	Activities and activity networks		
		Value is created, distributed and consumed by activities	See "Value creation, distribution (or "delivery") and consumption"
		A value may depend on a single activity or on a combination (or network) of activities	Value (Value Add) can be the result of a single activity, but can also aggregate results from multiple activities, typically forming a network, and contained by possibly different collaborations
	Value measurement		
		Value and aspects of value creation, distribution and consumption, should be measurable, based on definable types of measures	Characteristics of values (Value Adds), and of measureable elements of the value creating, distributing and consuming system, such as activities, deliverable flows, resources uses, stores and other elements, can be measured based on definable types of measures in measure libraries
"**Supports Business Model"**			
	Customer, market segments and other stakeholders		
		Identify customers, market segments and stakeholders, other than customers, and business relationships with them, to which parties the business provides value	Business networks, as collaborations, define party roles, which can be assigned to organizations (e.g. companies), communities (e.g. market segments) or consumers (modeled as actors, such as persons). Their patterns of collaboration can be defined via the exchange of propositions and/or deliverables.
	Value proposition		
		Define value propositions, related to the products and services that are offered to	Value propositions, related to deliverables that are exchanged, via the values (Value Adds) that

		these parties, and which articulate the values that are delivered to them, in terms that relate to how their needs are satisfied	are delivered based on these deliverables, whereby the value proposition expresses the importance of these values to the recipient, as well as the recipient's level of satisfaction with them.
	Resources and activities		
		Identify the resources and the activities that they perform or are used by, which are required to apply the functions that are needed to create the value that is required	Activities are associated with the capability offers that are applied, as well as with capability methods and/or resources (via stores or pools) that support these capability offers. It is also possible to define resource use in measurable terms.
	Network partners		
		Identify network partners, such as suppliers and others, and business relationships with them, via which these parties contribute value, that is part of the overall value that the business provides	Business networks, as collaborations, define party roles, which can be assigned to organizations (e.g. network partner companies), communities (e.g. supply markets) or persons (as individual suppliers). Their patterns of collaboration can be defined via the exchange of propositions and/or deliverables. Value delivered by network partners can be further aggregated (in measurable ways) into value delivered to e.g. customers.
	Profit and value calculations		
		Define the structures and related computation mechanics that can determine all relevant measurements of cost, revenue, as well as, more generally, value provided, value received, and, though maybe subjectively, "value margin", related to the operations of the business, in the context of the business model	Value delivery model scenarios enable a measurement framework, in relation to measurable elements. This enables measurement of performance or other characteristics, as well as measurement of value contributions (Value Add) based on these. Analysis of value provided versus value received is enabled via value proposition related measurements. Some measurements, such as a measurement of value contribution, recipient's satisfaction, profit, and value margin are predefined in the meta-model. The framework supports definition of any additional measurement.
"Discovers Process and Service Models"			
	Capabilities and interfaces of capabilities		
		Support services as mechanisms to enable access to one or more capabilities, via prescribed interfaces.	Capability offers can be supported by capability methods, which may have input and output ports. Capability methods may represent business processes. Capability method input and output ports can be interpreted "interface inputs and outputs". A delegation context (in which the capability method is used) delegates (maps) activity inputs and outputs to inputs and outputs of the capability method. Different contexts might relate to different subsets of inputs and outputs. An implementation of VDML, or integrated VDML modeling support, may provide transformation of delegations to capability methods to service interfaces, and might associate service interfaces, which are themselves complementary to the VDML specification, with capability methods, their input and output ports and with delegation context.
		Be able to use capabilities to identify needed services, and to organize them into	Capabilities are maintained in a capability library. Organizations provide capability offers, providing

		catalogs in order to communicate the needs and capabilities of a service area, and whereby participants that provide a service must have a capability to provide it, and whereby different providers may have different capabilities to provide the same service. Capabilities can be seen from two perspectives, capabilities that a participant has that can be exploited to provide services, and capabilities that an enterprise needs that can be used to identify candidate services.	and managing the capabilities in their particular way. Activities define their capability requirements (in terms of references to capabilities in a library), based on which matching capability offers can be found.
		Enable that the capability provider's business processes support the capabilities that provide the provider's services.	Capability methods can be considered abstraction of processes. When containing sufficient structure in their activity network, they may be transformed to and associated with business process models.
	Collaboration to engage capabilities		
		Capabilities offered are used through interaction between participants.	Collaboration defines how roles interact, whereby participants in roles apply their capabilities, through activities, which consume or use and produce deliverables that are exchanged.
		A collaboration defines a pattern of interaction between roles, whereby entities or participants (e.g. persons, organizations, or systems) "play a role" in a the collaboration	Collaboration in VDML supports this. Organizations are specialized collaborations by themselves, called org units. Persons are actors. Systems might be actors, or they are modeled as capability methods to support capability offers of organizations. All of these can play roles.
	Organizational alignment of capabilities, activities and resources		
		Capabilities and services are possessed and provided by organizations	Organizations (org units) provide capability offers, which might be supported by capability methods, which can be considered as exposed via service interfaces. The capability methods might be owned by the capability providing org units, or by other org units.
		Capabilities require a combination of organization, people, processes, and technology	Capability offers, being provided by organizations, and being supported by capability methods (which maybe abstractions of processes) and resources, modeled via stores, pools and associations of positions to pools. Capability methods might also be associated with method resources directly, which are resources that generally accelerate methods, and for which it is not useful to have individual activities accounting for their use. Examples are patents and technologies such as ERP suites and BPM technology.
		A process view, as complementary to a service view focuses on what activities parties perform to provide and use services	Capability methods, as collaboration, typically contain (a network of) activities, whereby the activities are performed by collaboration roles (typically capability method performers).
	Loose coupling of activity networks through stores		
		From above it is clear that processes or activity networks support capabilities, which are provided as services, and that capabilities are involved through activities as well. This implies many dependencies in the business system. These dependencies cannot all be activity-based however, as this would lead to a representation of a business as one vast activity network. This	VDML supports stores. Deliverables might flow from activity to activity, from activity to store and from store to activity.

		is not realistic. A business comprises many activity networks, which are loosely coupled, via decoupling buffers, in this document further denotes as stores	
	Monitoring-based scenarios and measurements		
		Processes and services, as discovered, and implemented, will be executed, and it is required to be able to feed measurements from monitoring of implemented processes and services back to the value delivery model, as input for next cycles of innovation	Capability methods maybe transformed into executable processes or interfaces to applications that are implemented in the real-world organization (“process automation”). Such processes and applications (including system functions in enterprise applications), when implemented and executed, will generate business data, including performance data. Measured characteristics of measureable elements in value delivery models can be measured based on “direct” measures, maintained in libraries of measures, which “direct” measures are associated with operations that maybe implemented as query services that query business data from operational business process and application databases.
		As-monitored measurements, based on as-implemented parts of a value delivery model should be distinguishable from e.g. to-be scenarios, with estimated, planned or simulated measurements. It is essential that scenario-based analysis can be applied, based on the same value delivery model	Different scenarios might analyze the same value delivery models (in total or in parts), whereby each scenario, directly or indirectly (via delegation contexts in its context tree) comes with its own set of measurements for measured characteristics of measurable elements in the value delivery models. Different measurements might be determined by different measures, or by the same measures under different circumstances. “Direct” measures are the “leaves” in the measurement hierarchy, which require “external inputs”, such as query results, or stochastic samples or just manual entry by business analysts. Other measures aggregate or grade or rank “direct” measures. VDML integrates with and applies a complete measurement framework, in a way that supports scenario-based analysis and simulation.
“Leveraging Existing Approaches”			
	Value flow through role collaboration		
		The concept of role collaboration and exchange of deliverables that convey tangible and intangible value	This is the concept of collaborations in VDML.
		The concept of reciprocity in collaborations or value exchanges (something received compensating for something provided)	The scope of a collaboration, typically a business network, defines the scope of reciprocity. Its roles provide value propositions to and receive value propositions from other roles in the collaboration. Each role (party in the business network), is supposed to get compensation for what it provides to others, and may only maintain its role in the collaboration sustainably, if it “gains” from the collaboration.
	Capability and value stream / chain analysis		
		Define capabilities and apply them in value chains or value streams	The activity network of a collaboration, possibly extended with activity networks of other collaborations, linked via stores, can be considered a value chain or value stream. The contribution of activities to value, and the aggregation of value (Value Add) over a stream of

			activities, completes the notion of a value stream. Capabilities (capability offers) are applied through activities.
		Libraries for standardized reference model elements, such as capabilities and related measures, resources or deliverables and practices)	VDML supports libraries of capabilities, business items (serving as resources or deliverables, dependent on the situation in which they occur), values, measures and practices. Elements in these libraries can be used to guide modeling, as well as enforce consistency in across models.
	Explicit modeling of resources, resource stores, resource use and deliverables		
		Explicit modeling of resources and deliverables	Business items that are conveyed via deliverable flows, and which may serve as resources (used or consumed by activities) or deliverables (produced by activities), and which maybe kept in stores or pools (the latter for re-usable resources). Stores or pools or resources may support capability offers. Pools of resources might be reconciled with individual actors or roles of actors (positions). Resources may also be defined at the level of capability methods, in case they cannot usefully be said to be used/consumed or produced by individual activities. Resources maybe also act as role resources, performing activities.
		Explicit modeling of resource stores and how resources are consumed and produced or received	
		Distinction between the role resource ("agent") and other resources that are actually transformed or exchanged by the role resource	
		The concept of stores, as buffers that decouple or link value streams	See "Loose coupling of activity networks through stores"
	Emphasis on measurement of performance and value, also applied in scenario-based analysis		
		Aggregation of performance measurement, with distinction between added value and waste (as lean value stream maps do with lead time)	VDML comes with an integrated and integral measurement framework, that is applied to measure performance and value. Measurements of performance characteristics, e.g. throughout a stream or network of activities, can be aggregated, whereby the aggregated result can be defined as Value Add. Value propositions can grade or rank (aggregated) value. Grade measurements or ranking measurements can clarify to which extent results (Value Adds) are denoting "actual value" or "waste". Waste might be represented a component of value proposition that's requiring significant effort from the provider, but is dissatisfying and/or is irrelevant to the recipient. VDML provides the structure to analyze these aspects.
		Support for rather complex aggregations of measurements (cost-related, and more)	This is supported by the integrated measurement framework of VDML (based on integration of VDML and SMM).
		Support for quantitative what-if analysis or simulation of profit, based on cost and revenue in complex exchanges (n-ary business relationships), also based on demand forecasts of the parties involved	Scenario-based analysis can be applied, whereby different scenarios may apply different (or partly different) measurements for the same measured characteristics of measurable elements in value delivery models. Scenarios might typically separate analysis results that originate from different measurements, such as from manual input ("guesstimates"), querying real-world performance, or automated simulations. Different scenarios might separate analysis results that have been created under different circumstances, such as different demand rates, different material or resource costs, different customer priorities or satisfactions, different behaviors and availabilities
		The explicit definition of scenarios, to enable what-if simulation of different scenarios, based on the same model	

			of resources, etc. (different simulations). Some measured characteristics are pre-defined in the meta-model, such as “profit”, “recurrence interval” (the inverse of “rate”, e.g. demand rate), “resource use”, “duration”, “value measurement”, “value satisfaction”, “proposition value”, etc. The user might define any other measured characteristic, of measurable elements, as useful for a particular analysis or concern. Libraries support the user in defining measured characteristics and applying measures that are meaningful and standardized in a particular industry or company or for a particular modeling concern.

Table 6. VDML response to requirements

Error! Reference source not found. provides a mapping between (or alignment of) VDML and other approaches in the value delivery modelling domain, as have been discussed in 2.3.

4.2 How VDML meets motivation

As indicated in Figure 3, business models motivate value delivery models, and support “business models”. Figure 131 suggests how value considerations are underlying VDML-based analysis, and how such analysis might typically start at the level of a business network, which also aligns with the notion of “business model” in business model frameworks (see the mapping in **Error! Reference source not found.**.

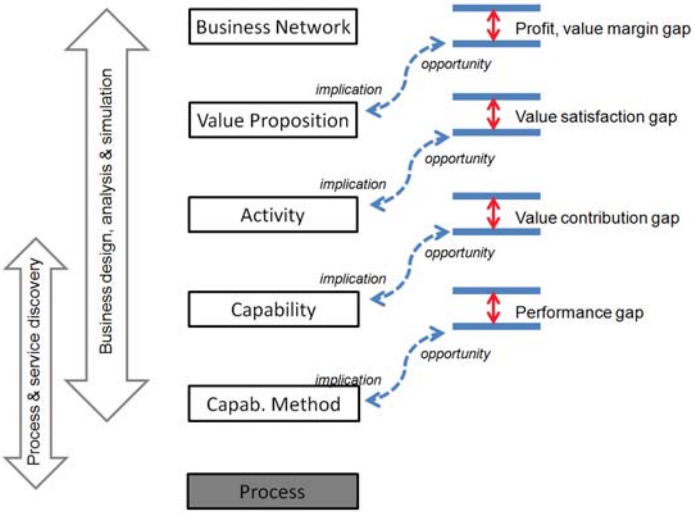


Figure 131. Value-driven innovation

Parties in a business network exchange value propositions, and, based on that observe profit (or loss), and a value margin (or lack of it). As explained in 3.1.5, a value margin is the net result that a recipient observes between the value received and the price paid for deliverables underlying the value proposition. When a party doesn’t get a “gain” from collaboration in the network, due to situation of loss or negative value margin, that party might accept this situation, as, in a portfolio of multiple business networks (“business models”) there is a balance, e.g. “gain” in some business models outweighs the “loss” in others. Or that party might withdraw from a business network (dismantle a “business model”), or seeks for opportunity to improve the results of the business network, which will lead to implications for value propositions.

For instance, in relation to the “enterprise value proposition” of the business network to the enterprise or leadership (see 3.1.5), that party might want to have a more satisfactory cost level. This would have implications for activities that contribute to cost. Or that party might want, in relation to value propositions provided, in the business network, to other parties, to increase the satisfaction of these recipient parties with values that are important for them, to increase their observation of value margin, or their willingness to pay for the value they get, or even pay more. Also this would lead to implications of activities that contribute to the values of concern. Activities that insufficiently or even negatively contribute to certain values, provide opportunities to abandon, subcontract, or improve capabilities, or even develop new ones. Improving capabilities is about transforming the way these capabilities are managed, so that the application of them leads to better performance, as far as the characteristics performance are associated with the values of concern. And this, in turn will typically lead to improvements of capability methods and/or resources, so that they contribute to the right levels of performance and so value. Improving capability methods, in many cases, is about improving or further innovation processes and services. As suggested by Figure 131, this all is about “value driven innovation”, starting from business model innovation, and translating down to process and service innovation. Analysis is also possible the other way round, bottom-up: given the place and purpose of e.g. a process, as part of a business system, it can be analyzed to which extent that process contributes to value in order to meet expectations.

As typically led by value and business model considerations, value delivery modeling supports model-based business design and analysis (and simulation to analyze with more rigor), both top-down and bottom up, whereby the business system might be (re)designed and analyzed in completion or, most often, in parts, dependent on where the innovation priority is.

Analysis and (re) design, based on value delivery models, that leads to improved or new capabilities, their interfaces and capability methods, can be considered “discovery” of new services and processes, or of opportunities to further innovate them.

5 Conclusion

5.1 Summary of results

VDML is a modeling language that supports business design, driven by analysis of business value.

VDML models (or value delivery models) can provide a detailed and structured foundation for “business models”. VMDL-based designs of the business may express how the business, supporting the “business models”, is structured (or should be structured) and operates (or should operate). In particular, VDML models may express how the business consumes, creates and delivers business value, both intra- and inter-enterprise.

VDML also provides a structured model-based approach for discovery of and/or alignment with process and service models. VDML bridges the gap between “business model” and the layer where parts of the operation are automated and managed based on process models and service models.

VDML models can include the measurements (of performance and value) required for quantitative analysis, and eventually also for automated business simulation.

Libraries of standardized model elements, such as capabilities, business items, etc., can facilitate common vocabulary amongst the business users that are involved in value delivery modeling or using value delivery models. A value delivery modeling tool can also use such libraries to guide the modeling user in productively discovering activities, capability offers, capability resources as well as inputs and outputs that capability methods need to handle, given the capability that they support. The linkage between capability offer and capability will also provide the opportunity to rationalize capability offers, by e.g. combining them to improve economy-of-scale.

Though VDML supports business design, which might be concerned with modeling both as-is, and scenarios of to-be states of the business, the “process of business innovation”, and its related elements of strategy and tactics, are not explicitly covered by VDML, but are complementary to it. Though VDML models can also be used to design and express the “system of business innovation” (its capabilities, collaborations, etc.).

VDML provides a more integral and multi-faceted and structured-modeling based approach than other, existing, approaches in the area of value delivery modeling. The modeling that is required for some of these approaches is subsumed by VDML. For other approaches, there is no 1-1 mapping, but what can be modeled by these other approaches, can also be modeled based on VDML, though with more universal and less dedicated means (and notation). VDML will provide practitioners of these other approaches a way to align with or migrate to a standard structured modeling approach, which can be implemented as integrated with existing main stream modeling approaches, such as for process and service modeling. VDML-based modeling may also provide disparate communities of these practitioners a common denominator or modeling kernel, based on an integrated set of modeling concepts that can be aligned with counterparts in their various approaches.

5.2 Future Work

Here are some future work that is being considered:

- Further extension and elaboration of user interface (and notation) paradigms for measurements-based analysis.
- Further efforts on tooling support
- Application of VDML to other use cases. Effort is ongoing to e.g. applying VDML to a use case in healthcare (which is also the domain that is covered NEFFICS).
- Mapping VDML to SoaML (see SoaML (2012), most likely as part of work in NEFFICS.
- Providing automated business simulation support, based on VDML models.

6 References

Allee, V., The Future of Knowledge: Increasing Prosperity through Value Networks, Butterworth-Heinemann 2003.

Allee, V., *Value Network Analysis and Value Conversion of Tangible and Intangible Assets*, Journal of Intellectual Capital, Volume 9, issue 1, pp 5-24, January 2008, http://www.vernaallee.com/images/VAA-VNAandValueConversionJIT.pdf .

Allee, V., *Value Networks and the True Nature of Collaboration*, ValueNet Works, 2011, http://www.valuenetworksandcollaboration.com.

Ballantyne, D., Varey, R.J., Frow, P. and Payne, A., *Service-dominant logic and value propositions: Re-examining our mental models*, Otago Forum 2, Paper no: 5, 2008, http://www.business.otago.ac.nz/marketing/events/OtagoForum/Final%20forum%20papers/Otago%20Forum%20Paper%205_Ballantyne.pdf .

Bolstorff, P. and Rosenbaum, R., *Supply Chain Excellence: A Handbook for Dramatic Improvement Using the SCOR Model*, AMACOM, a Division of the American Management Association, 2003.

BPMM, *Business Process Maturity Model*, Version 1.0, Object Management Group, Release Date: June 2008, http://www.omg.org/spec/BPMM/1.0/PDF/ .

BPMN, *Business Process Model and Notation*, Version 2.0, Object Management Group, Release Date: January 2011, http://www.omg.org/spec/BPMN/2.0/ .

Brodie, L. and Gilb, T., *Values for Value,* AgileRecord, October 2010, http://www.gilb.com/dl448

Cummins, F. A. and De Man, H., *Capability Analysis with the Value Delivery Modeling Language*, Cutter IT Journal, April 2011.

Dooley, K., *Simulation Research Methods, Companion to Organizations*, Joel Baum (ed.), London Blackwell, 2002, http://www.public.asu.edu/~kdooley/papers/simchapter.PDF .

Fowler, M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley, 2004

Joseph, F., *The IT Supply-Chain SCORcard*, BPTrends, March 2007, http://www.bptrends.com/publicationfiles/NINE-03-07-COL-ITSupplyChainSCORcard-Managing%20BPM-Francis-Final.pdf .

Gane, C. and Sarson, T, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall Software Series, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

Gilb, T., *Value Delivery in Systems Engineering,* October 2007, Published and used by INCOSE with permission, http://www.gilb.com/dl137

Gilb, T. and Gilb, K., *Done should mean value delivered to Stakeholders*, AgileRecord, October 2011, http://www.gilb.com/dl484 .

GoldSim, *Summary of Major New Features and Changes*, Version 10.1, GoldSim Technology Group, February 2010, http://www.goldsim.com/downloads/Documents/Version101Summary.pdf .

GoldSim, *Probabilistic Simulation Environment, User's Guide*, Version 10.5, GoldSim Technology Group, December 2010, http://www.goldsim.com/downloads/Documents/Version101Summary.pdf .

Gordijn, J. and Akkermans, H., *Value based requirements engineering: Exploring innovative e-commerce ideas*. In Requirements Engineering Journal, Vol. 8(2):114-134, 2003, http://e3value.few.vu.nl/docs/bibtex/pdf/Gordijn2003e3value.pdf, or a popular version of it: http://e3value.few.vu.nl/docs/bibtex/pdf/Gordijn2001e3value.pdf .

Harmon, P., *Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals*, Morgan Kaufman, 2007.

Hruby P., Kiehn J. And Scheller C.: *Model-Driven Design Using Business Patterns*, Springer-Verlag, 2006.

Hubbard, D. W., *How to Measure Anything, Finding the Value of “Intangibles” in Business*, John Wiley & Sons, New Jersey, 2010.

ITIL, *Service Design*, ITIL Version 3, August 2011, http://www.best-management-practice.com/Publications-Library/IT-Service-Management-ITIL/ITIL-2011-Edition/Service-Design/ .

Johnson, M. W., Christensen, C. M., and Kagermann, H., *Reinventing Your Business Model, Harvard Business Review on Business Model Innovation*, Harvard Business School Publishing Corporation, 2010.

Kaplan, R. and Norton, D., *The Balanced Scorecard*, Harvard Business School, 1996.

Krohn, D., *A Capability-Based Approach to Strategic Transformational Initiatives*, Cutter IT Journal, November 2011.

Mercer, T., Groves D. and Drecun, V., *BPTF Framework 2010, Part 1*, BPTrends, September 2010, http://www.bptrends.com/publicationfiles/09-14-10-ART-BPTF%20Mercer%20et%20al-Final.pdf .

Mercer, T., Groves D. and Drecun, V., Part II - BPTF Architectural Overview, BPTrends, October 2010, http://www.bptrends.com/publicationfiles/THREE%2010-05-10-ART-BPTF%20Framework-Part%202-final1.pdf .

Mercer, T., Groves D. and Drecun, V., Part III – Practical BPTF Application, BPTrends, November 2010, http://www.bptrends.com/publicationfiles/FOUR%2011-02-10-ART-BPTF%20Framework--Part%203-Mercer%20et%20al%20--final1.pdf .

NEFFICS DoW, *Description of Work*, Annex 1 to Proposal 258076 (STREP), April 2010.

NEFFICS D1.3, *Initial Release of the Virtual Extended Factory*, April 2011.

NEFFICS D3.2, *Definition of Business Values*, June 2012.

NEFFICS D3.3 – Part A, *Value Delivery Model and Methods*, June 2012.

NEFFICS D3.3 – Part B, *Value Delivery Modeling Language (VDML)*, Version 0.2, Object Management Group, Submission for May 21, 2012.

NEFFICS D4.1, *Baseline for Networked Innovation Models*, April 2011.

NEFFICS D4.3, *Models for Network-based Open Business Model Innovation*, June 2012.

Osterwalder, A., *The Business Model Ontology- A Proposition in a Design Science Approach*, Thesis, University of Lausanne, 2004, http://www.hec.unil.ch/aosterwa/PhD/Osterwalder_PhD_BM_Ontology.pdf .

Osterwalder, A. and Pigneur, Y., *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*, John Wiley & Sons, 2010.

PMBOK, *A Guide to the Project Management Body of Knowledge (PMBoK Guide)*, Project Management Institute (PMI), 2000, http://www.cs.bilkent.edu.tr/~cagatay/cs413/PMBOK.pdf

Porter, M. E., *Competitive Advantage: Creating and Sustaining Superior Performance*, The Free Press, New York, 1985.

Rother, M. and Shook, J., *Learning to See*. Lean Enterprise Institute, 1998.

SMM, *Software Metrics Meta-Model*, Version 1.0, Object Management Group, Release Date: January 2012, http://www.omg.org/spec/SMM/1.0/ .

SoaML, *SOA Modeling Language*, Version 1.0, Release Date: March 2012, http://www.omg.org/spec/SoaML/1.0/ .

SOA-RA, *SOA Reference Architecture*, The Open Group, December, 2011, https://collaboration.opengroup.org/projects/soa-ref-arch/.

SOA-RM, *Reference Model for Service Oriented Architecture 1.0*, OASIS, October 2006, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html.

Sowa, J. F. and Zachman, J. A., *Extending and formalizing the framework for information systems architecture*, IBM Systems Journal, vol 31, no 3, 1992, http://www.zachman.com/images/ZI_PIcs/ibmsj1992.pdf .

Ulrich, W. and McWhorter, N., *Value Streams: Business Architecture's Guidepost to Business-IT Transformation*, Cutter IT Journal, April 2011.

UML, *Unified Modeling Language, Superstructure*, Version 2.4.1, Object Management Group, Release Date: August 2011, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/ .

Vervest, P. H.M., Van Liere, D. W. and Zheng, Li (Eds.), *The Network Experience, New Value from Smart Business Networks*, Springer, 2009, http://www.erim.eur.nl/ERIM/publications/book_releases/Release?p_item_id=5157588&p_pg_id=93

Weill, P. and M. R. Vitale (2001). *Place to space: Migrating to eBusiness Models*. Boston, Harvard Business School Press.

Whittle, R., and Myrick, C.B., *Enterprise Business Architecture: The Formal Link Between Strategy and Results*, CRC Press, 2005.

Zachman, J. A., *A framework for information systems architecture*, IBM Systems Journal, vol 26, no 3, 1987, http://www.cesames.net/wp-content/uploads/2010/04/ibmsj2603e.pdf .

7 Glossary

7.1 Justification of terms and definitions

Some terms in VDML are specific to VDML. They denote VDML-specific, and merely metamodel-technical concepts. These aren't listed in the Glossary. Terms that are listed in the glossary denote commonly known concepts, that have origins from outside VDML. VDML does not intend to introduce new terms unnecessarily, but to re-use terms that people can relate to.

When these terms aren't normatively defined by VDML, we adopt definitions from outside VDML, and we provide a reference to an outside source. When VDML introduces normative VDML-specific definitions for these terms, we adopt these definitions, with reference to the VDML 1.0 specification.

The reason for VDML to introduce VDML-specific normative definitions for these terms is as follows: VDML applies and integrates several known concepts in new ways, introducing semantics that is tuned or specific to VDML. VDML then also introduces VDML-specific definitions for terms that indicate these concepts, to ensure sufficient alignment with VDML-specific semantics, and to ensure that definitions of terms of concepts that are integrated with each other in VDML are sufficiently consistent with each other.

7.2 Terms and definitions

Activity. Work contributed to a collaboration by a participant in a role of the collaboration. A role may contribute to multiple activities in the same collaboration (source: VDML; example of use of term outside VDML: Osterwalder (2004)).

Actor. An individual (indivisible) participant, which might be human (a person) or non human (e.g. a software agent or machine) (source: VDML; example of use of term outside VDML: Gordijn and Akkermans (2003)).

Analysis Context. An analysis context defines the set of measurements of a particular use of one or more collaborations or a store when used as a decoupling point between collaborations. When a collaboration is used by an activity, a context also defines the delegations of activity inputs and/or outputs to/from collaboration inputs and/or outputs, and may define assignments of roles within the collaboration (source: VDML).

Business Item. A business item is anything that can be acquired or created, that conveys information, obligation or other forms of value and that can be conveyed from a provider to a recipient. For example, it includes parts, products, units of fluids, orders, emails, notices, contracts, currency, assignments, devices, property and other resources (source: VDML).

Business Model. A business model describes the rationale of how an organization creates, delivers, and captures value (source: Osterwalder and Pigneur (2010)).

Business Network. A collaboration between independent business (or economic) entities, potentially companies, agencies, individuals or anonymous members of communities of independent business entities, participating in an economic exchange (source: VDML; example of use of term outside VDML: Vervest et al. (2009)).

Capability. Ability to perform a particular kind of work and deliver desired value (source: VDML; examples of use of term outside VDML: Osterwalder (2004), SoaML (2012), ITIL (2011)).

Capability Method. A reusable template for a collaboration configured for participants to perform activities to deliver a capability and to contribute value in a particular situation (source: VDML).

Characteristic. Distinguishing feature or quality that can be qualified or quantified by applying a measure (source: popularized version of definition in SMM (2012)).

Collaboration. Collection of participants working together for a shared purpose (source: VDML; examples of use of term outside VDML: SoaML (2012), BPMN (2011)).

Community. A collaboration of participants with similar interests that work together for some shared purpose such as sharing knowledge (source: VDML; examples of use of term outside VDML: Weill and Vitale (2001)).

Delegation Context. A specialized Analysis Context, set by an activity and in which the activity delegates its work to a collaboration. A delegation context also defines the delegations of activity inputs and/or outputs to/from collaboration inputs and/or outputs, and may define assignments of roles within the collaboration (source: VDML).

Deliverable. Product or service produced by an activity or delivered from a store that can be conveyed to another activity or store (source: VDML; example of use of term outside VDML: Allee (2008), ITIL (2011)).

Deliverable Flow. The transfer of a deliverable from a provider (or producer) to a recipient (or consumer) (source: VDML).

Intangible. Deliverable that represents something that is unpaid or non-contractual that makes things work smoothly or efficiently (as opposed to Tangible) (source: VDML; example of use of term outside VDML: Allee (2008)).

Measure. A method that is applied to characterize an attribute of something by assigning a comparable quantification or qualification (source: popularized version of definition in SMM (2012)).

Measurement. The result of applying a measure (source: popularized version of definition in SMM (2012)).

Organization. An administrative or functional structure normally interpreted as a network of Organization Units at a higher level in an organizational hierarchy (source: VDML; ; example of use of term outside VDML: ITIL (2011)).

Organization Unit (or: Org Unit). An administrative or functional organizational collaboration, with responsibility for defined resources, including a collaboration that occurs in the typical organization hierarchy, such as business units and departments (and also the company itself), as well as less formal organizational collaboration such as a committee, project, or task force (source: VDML; example of use of term outside VDML: Zachman framework, as introduced by Zachman (1987), and Sowa and Zachman (1992), though a formal definition of the term seems to be omitted).

Participant. Anyone or anything that can fill a role in a collaboration. Participants can be actors (human or automatons) or collaborations or roles of actors or collaborations (source: VDML; example of use of term outside VDML: Allee (2008)).

Pool. A store that contains re-usable resource, i.e. resource that is returned to the pool after having been used, so that it is again available for use (source: VDML; example of use of term outside VDML: PMBOK (2000)).

Practice. Proven way to handle specific types of work and that have been successfully used by multiple organizations (source: VDML; examples of use of term outside VDML: BPMM (2008), ITIL (2011)).

Process. A sequence or flow of Activities in an organization with the objective of carrying out work (source: BPMN (2011)).

Resource. Anything that is “used” or “consumed” in the production of a deliverable (source: VDML; example of use of term outside VDML: Hruby et al. (2006)).

Role. An expected behavior pattern or capability profile associated with participation in a collaboration—role participants working together with a common interest or purpose (source: VDML; example of use of term outside VDML: Allee (2008)).

Scenario. A scenario defines a consistent business use case of a VDML model by specifying a, possibly recursive, analysis context for elements in scope of that use case. The nesting of contexts allows a collaboration to be used as a sub-collaboration by more than one activity, each of which sets its particular context and measurements (source: VDML).

Service. A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description (source: SOA-RM (2006)).

Store. Represents a container of resource. The resource that is received or provided is identified by a business item (source: VDML; common concept in data flow diagrams (DFD), also known as Gane-Sarson diagrams, as proposed and applied by Gane and Sarson (1979); common construct in simulation systems, such as GoldSim, as explained in GoldSim (2010, 1) and GoldSim (2010, 2); data store in BPMN (2011) is a similar construct, though with a more narrow meaning).

Tangible. Deliverable that represents something that is contracted, mandated or expected by the recipient and which may generate revenue (as opposed to Intangible) (source: VDML; example of use of term outside VDML: Allee (2008)).

Value. A measurable benefit delivered to a recipient in association with a business item (source: VDML; example of uses of term outside VDML: Brodie and Gilb (2010), Gilb (2007), Gilb and Gilb (2011)).

Value Chain. Set of activities that an organization carries out to create value for its customers (source: Porter (1985)).

Value Delivery Model. Model that supports business analysis and design based on evaluation of performance and stakeholder satisfaction achieved through the activities and interactions of people and organizations using business capabilities to apply resources and deliver stakeholder values (source: VDML).

Value Network. Any set of roles and interactions in which participants engage in both tangible and intangible exchanges to achieve economic or social good (source: Allee (2008)). Or: Any web of relationships that generates both tangible and intangible value through complex dynamic exchanges between two or more individuals, groups or organizations (source: Allee (2003)).

Value Proposition. Expression of the values offered to a recipient in terms of the recipient's level of satisfaction (source: VDML; examples of use of term outside VDML: Ballantyne et al. (2008), Osterwalder (2004), Johnson et al. (2010)).

Value Stream. End-to-end collection of activities that create a result for a customer, who may be the ultimate customer or an internal end user of the value stream (source: Whittle and Myrick (2005)).