Conceptual Modeling for Composition of Model-based Complex Systems – Ontological Means to capture Information Exchange and Constraints

Andreas Tolk, Old Dominion University, Norfolk, Virginia, USA
Saikou Y. Diallo, Old Dominion University, Norfolk, Virginia, USA
Robert D. King, Old Dominion University, Norfolk, Virginia, USA
Charles D. Turnitsa, Old Dominion University, Norfolk, Virginia, USA

Introduction

Conceptual models are often understood as efforts that happen before systems are being built or software code is written. They are primarily described as mental models that are in an early stage in the abstraction and as a simplification process that happens during the modeling phase. As long as the resulting implementation has an unambiguous interpretation for all users, this view of conceptual models may be sufficient.

However, the notion of conceptual models and their role presented in this chapter is different. When moving applications from the stand-alone mode into the realm of services in a service-oriented architecture, or into the domain of system of systems, it is essential for systems and services to be annotated with their respective assumptions, constraints, and simplifications. The resulting meta-data describing the conceptualization of systems and services allows machines or intelligent agents to identify applicable solutions, select the best available solution, compose services or systems, and orchestrate their execution to provide the functionality required for a given application. Thus, conceptual models must contribute to these machine-readable artifacts, and these artifacts must be part of the conceptual model. In this chapter, the necessity of such artifacts for valid compositions in the system of systems and service-oriented architecture domain will be shown.

In order to support the evaluation and analysis of composability and interoperability challenges for model-based complex systems, the Levels of Conceptual Interoperability Model (LCIM) was developed and applied successfully in different domains, such as defense, homeland security, and energy. The LCIM can be used as a guide for topics that have to be dealt with in order to support compositions of systems or services so that misalignments can be avoided in the different layers of interoperation: technical, syntactical, semantic, pragmatic, dynamic, and conceptual.

Within this chapter, the focus will be on artifacts needed for describing the information exchange requirements of systems and services that have to be captured during the conceptual modeling activities to ensure composability of models and interoperability of systems or services. This approach will be generalized into three engineering methods based on the application of systems engineering principles: data engineering, process engineering, and constraint engineering. The objective is to capture the results of conceptual modeling efforts in machine-interpretable form in order to enable and facilitate the composition of systems or services into system of systems or service oriented architectures.

The chapter starts by analyzing the interoperation challenges of model-based systems to motivate the LCIM. It shows how artifacts that belong to the conceptual modeling realm must be used to deal with these challenges. The chapter then focuses on the challenges inherent to data exchange between systems and introduces the notion of data engineering. The principles of data engineering are extended to analyze the requirements for processes that consume or produce data leading to process engineering and to understand the assumptions and constraints placed on systems and services leading to constraint engineering. All three engineering methods are described and artifacts that should result from the conceptual modeling process are identified. The common theme is the need for a standardized way to formally specify the conceptualization. A formal specification of a conceptualization, however, is the definition of an ontology, which leads to the evaluation of ontological means to express the results of the conceptual modeling phase in machine-understandable form. The chapter closes with a summary and description of the way forward to integrate conceptual models into the life cycle of systems in a service-oriented architecture or a system of systems.

Interoperation Challenges for Model-based Complex Systems

In order for two systems to interoperate, they need to fit together. The working definition of interoperation used in this chapter is simply: *two systems can interoperate if they are able to work together to support a common objective.* Traditionally, systems that are able to interoperate are referred to as interoperable systems. The Institute of Electrical and Electronics Engineers (IEEE, 1990) defines interoperability as *"the ability of two or more systems or components to exchange information and to use the information that has been exchanged."*

Unfortunately, multiple definitions of interoperability exist that are ambiguous. Petty and Weisel spawned very fruitful discussions on the differences and commonalities of interoperability and composability with their paper (Petty & Weisel, 2003). They show that the definitions are primarily driven by the challenges of technical integration and the interoperation of implemented solutions. Model-based complex systems are further adding a new category of challenges. The working definition for a model-based complex system results from the definition of the combined terms: *a system is made up of several components that interact with each other via interfaces; a complex system has many components that interact via many interfaces that represent typically non-linear relations between the components; model-based systems use an explicit formal specification of a conceptualization of an observed or assumed reality.* While complexity already plays a major role in the traditional view of interoperability, the model-based aspect is not often considered. To explicitly deal with challenges resulting from differences in the concepts the term composability is used.

As with interoperability, the definitions used for the term composability are manifold. Petty, et al. (2003) compiled the various definitions and used them to recommend a common definition embedded into a formal approach. Fishwick (2007) proposed, in his recent analysis, the restriction of the scope of composability to the model level, and follows the recommendations of Page, et al. (2004). Page recommends distinguishing between three interrelated but individual concepts all contributing to interoperation:

- *Integratability* contends with the physical/technical realms of connections between systems, which include hardware and firmware, protocols, networks, etc.
- *Interoperability* contends with the software and implementation details of interoperations; this includes exchange of data elements via interfaces, the use of middleware, mapping to common information exchange models, etc.
- *Composability* contends with the alignment of issues on the modeling level. The underlying models are purposeful abstractions of reality used for the conceptualization being implemented by the resulting systems.

The same recommendation is supported by Tolk (2006), where the importance of these categories for the domain of modeling and simulation is made obvious. An evaluation of current standardization efforts shows that the focus of these standards lies predominately on the implementation level of interoperability and doesn't consider the specialty of models sufficiently, which is in the modeling process as an activity that purposefully simplifies and abstracts reality and constrains the applicability of the model in the form of assumptions and constraints. While interoperability deals with simulation systems, composability deals with models, hence *Interoperability of Simulation Systems requires Composability of Conceptual Models!* Using the categories of Page et al., the case can be made that integratability assures the existence of a stable infrastructure such as a reliable network, interoperability assures that simulation systems can be federated with each other, and composability assures that the underlying conceptualizations are aligned – or at least not contradictive. Conceptual models capturing the information needed are therefore essential. This point is emphasized in (Benjamin, Akella, & Verna, 2007, p. 1082): "*The semantic rules of the component simulation tools and the semantic intentions of the component designers are not advertised or in any way accessible to other components in the federation. This makes it difficult, even impossible, for a given simulation tool to determine the semantic content of the other tools and databases in the federation, termed the problem of semantic inaccessibility. This problem manifests itself superficially in the forms of unresolved ambiguity and unidentified redundancy. But, these are just symptoms; the real problem is how to determine the presence of ambiguity, redundancy, and their type in the first place. That is, more generally, how is it possible to access the semantics of simulation data across different contexts? How is it possible to fix their semantics objectively in a way that permits the accurate interpretation by agents outside the immediate context of this data? Without this ability—semantic information flow and interoperability—an integrated simulation is impossible.*"

Following the objective of this chapter, the resulting engineering task is to capture the necessary conceptualization as they are utilized and annotated by systems *semantically accessible* to other systems so that they can make use of the information to select applicable solution, choose the best available, compose the part solutions to deliver the overall solution, and orchestrate the execution. For the remainder of this chapter, the terms “service” and “system” are used interchangeably.

In other words, the idea is to make the system semantically transparent for intelligent agents so that they can understand the objectives, inputs, outputs, content, assumptions, and simplifications captured in conceptualizations. Zeigler (1986) published a model explaining what is necessary for machine-based understanding. He identified three requirements that are applicable in the context of understanding the conceptualization and implementation as well:

- *Perception* – The observing system has a perception of the system that needs to be understood. In Zeigler’s model, perception is not a cognitive process but is simply capturing sensor input. Nonetheless, it implies that data and processes characterizing the observed system must be observable and perceivable by the observing system that shall help to identify, select, compose, and orchestrate the observed systems. It also implies that all data needed for these tasks are provided by the applicable systems that are observed.
- *Meta-Models* – The observing system has an appropriate meta-model of the observed system. The meta-models represent the categories of things the observing system knows about. As such, each meta-model is a description of data, processes, and constraints explaining the expected behavior of an observed system. Without such a meta-model of the observed system, understanding for the observing system is not possible. In Zeigler’s model, machine-based understanding can be defined as identifying an applicable meta-model. In the context of this chapter this means that the intelligent agent must have available meta-models that can be used to map perception in support of the tasks of identification, selection, composition, and orchestration.
- *Mapping* – Mappings between observations resulting in the perception and meta-models explaining the observed data, processes, and constraints do exist, are identified, and are applied in the observing system.

In summary, the artifacts that are derived during the conceptual modeling phase must therefore be perceivable and it must be possible to map them to meta-models that can be used to describe system behavior in the system-of-systems environment or service behavior in service-oriented architectures.

The related work on the challenges of interoperability and composability enabling the attainment of these objectives led to the development of the LCIM. As documented in Tolk (2006), the LCIM is the result of several composability and interoperability efforts. During a NATO Modeling & Simulation Conference, Dahmann (1999) introduced the idea of distinguishing between substantive and technical interoperability. In his research on composability, Petty (2002) enhanced this idea. In his work, he distinguished between the implemented model and the underlying layers for protocols, the communication layers, and hardware. Realizing the need to explicitly address the conceptual layer, Tolk and Muguira (2003) published the first version of the LCIM, which was very data-centric. The discussions initiated by the LCIM work, in particular the work of Page et al. (2004) and Hofmann (2004), resulted in the currently used version, which was first published by Turnitsa (2005). The following figure shows the evolution of layered models of interoperation resulting in the LCIM.

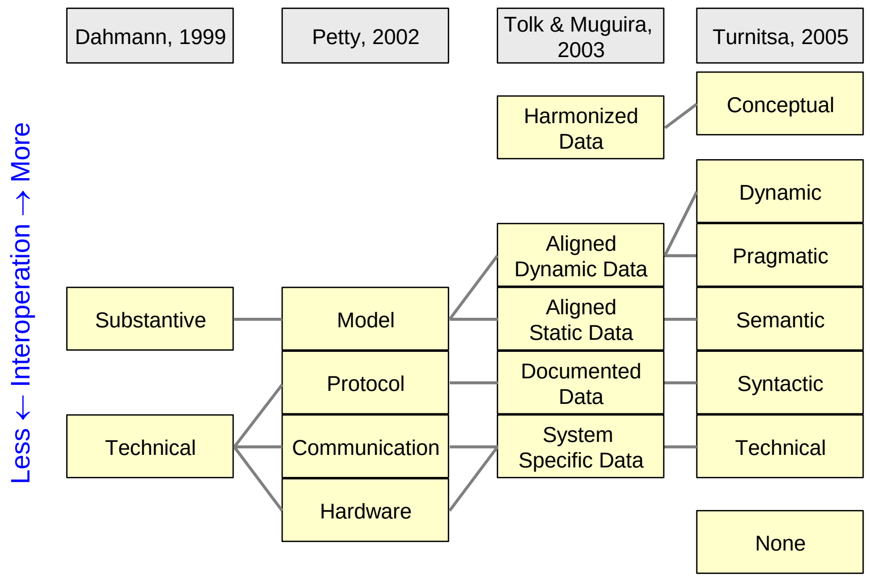

Figure 1: Evolution of Levels of Interoperation (Tolk, 2006, p. 57)

As shown, the LCIM exposes six levels of interoperation, namely:

- The *technical* level deals with infrastructure and network challenges, enabling systems to exchange carriers of information.
- The *syntactic* level deals with challenges to interpret and structure the information to form symbols within protocols.
- The *semantic* level provides a common understanding of the information exchange. On this level, the pieces of information that can be composed to objects, messages, and other higher structures are identified.
- The *pragmatic* level recognizes the patterns in which data are organized for the information exchange, which are in particular the inputs and outputs. These groups are often referred to as (business) objects.
- The *dynamic* level adds a new quality by taking the response of the system in form of context of the business objects into account. The same business object sent to different system can trigger very different responses. It is also possible that the same information sent to the same system at different times can trigger different responses.
- Finally, assumptions, constraints, and simplifications need to be captured. This happens on the *conceptual* level.

Wang, et al. (2009) show the descriptive and prescriptive potential of the LCIM and evaluate a first set of artifacts, in particular those defined by the simulation interoperability standards IEEE 1516, the High Level Architecture, and the Base Object Models (BOM) standard recently developed by the Simulation Interoperability Standards Organization (SISO). As documented by King (2007), the elusiveness of the conceptual level requires the orchestrated application of all existing part solutions. This orchestration requires a conceptual model that supports all views and all levels of interoperation.

The definition of conceptual models used in this chapter is based on Robinson (2007, p. 283): "*A conceptual model is a non-software specific description of the computer simulation model (that will be, is or has been developed), describing the objectives, inputs, outputs, content, assumptions and simplifications of the model.*" Robinson argues that the conceptual model establishes a common viewpoint that is essential to developing the overall model. Conceptual modeling captures aspects of interoperability that extend software definition. It specifically addresses understanding the problem, determining the modeling and project objectives, model content (regarding scope, structure and level of detail), and identifying assumptions and simplifications.

The consequence of these observations is that decisions made in the conceptual modeling phase will influence whether two systems can interoperate or not, even if these decisions are not explicitly obvious from the implementation. If two such systems are still composed with (or without) this knowledge, the resulting composition will be conceptually misaligned and will produce inappropriate results. In the worst case, the user will not be aware and use the results of such compositions in support of decisions, which can result in harm on loss for the user and the domain being supported.

This observation is not completely new. The importance of well-documented conceptualizations as a distinguishable product and the basis for simulation system specifications and implementations for the related domain of validation and verification is also emphasized by Sargent (2001). Brade (2000) introduced a series of artifacts allowing the tracing of results and recommendations back to the sponsor's needs thus capturing the resulting model developer's intention. While Brade's artifacts are still mainly designed for human consumption, Muguira and Tolk (2005) introduced at least partly readable models to capture the developer's intention to avoid the composition of simulation system with insufficiently aligned models.

As mentioned before, the necessity to support fully machine-understandable artifacts capturing the results of the conceptual modeling phase ensures that composability emerges from the idea to compose systems from existing systems (system of systems engineering) or from services (service-oriented architectures). In both application cases is it crucial to be aware of objectives, inputs, outputs, content, assumptions, and simplifications of the underlying models.

Yilmaz (2004) introduced the phrase *Contextualized Introspective Simulation Models* to describe the need of model-based applications to communicate their assumptions, constraints, inputs, outputs, and other elements of the conceptual model in order to improve reuse and composability. He shows that intelligent software agents can be used as placeholders and ambassadors to negotiate with other intelligent software agents representing other systems to find out whether a composition is possible. Again, this requires capturing the conceptual model in a machine-readable formal specification. Additional contributions and relevant related work can be found in the domain of agent-mediated interoperability and semantic web-services, such as described by Sycara (2002), Burstein and McDermott (2005), and Yilmaz and Tolk (2006).

This chapter recommends ontological means to capture the conceptual model. A common definition for an ontology is *a formal specification of a conceptualization* (Gruber, 1993). It can be argued that the use of ontological means to represent conceptual models in machine-understandable form is *quasi* motivated *per definitionem*: ontologies are the required specification of the conceptualization, and as they are formal, machines can read them and reason about them. However, there are very practical observations that support this decision as well. West (2009) gives an example for practical applications of ontologies in connection with modeling of data for a significant business. A more theoretic introduction to the use of ontological means has recently been published by Guizzardi and Halpin (2008) in their special issue on conceptual modeling in the Journal of Applied Ontologies. The discussion on ontological means in support of systems engineering are ongoing.

In summary, ontological means are at least a promising candidate to describe artifacts. These artifacts can be used to capture the results of conceptual modeling in form of objectives or developer's intent, business objects in form of inputs and outputs, content, assumptions and simplifications. As they are based on formal mathematics, they can be encoded in machine-readable form. These artifacts can be used to annotate solutions to enable their reuse in system of systems or as services in service-oriented architectures as they provide the means to describe the systems thus allowing the identification of applicable solutions and the selection of the best available solution. The artifacts ensure that candidate solutions can interoperate on the technical, syntactical, semantic, pragmatic, dynamic, and conceptual level of interoperation.

Within the following sections, the interoperation requirements for data-centric solutions will be presented, as this domain is the one best known in the community. This section will be followed by a presentation of current work in the field. The chapter will close with a short survey of ontological means and their application.

Data-centric Interoperation Requirements

In order to collaborate, systems need at least to be able to provide the required input parameters of services provided by partners and to make sense of their output parameters. These parameters set up the information exchange between two systems. The current practice of information exchange modeling is divided into two main categories:

- The first category uses *peer-to-peer agreements* between sending and receiving systems. The systems have to mutually agree to the matches between the information provided by the sender, and they have to agree to how this information is understood by the receiver. Such agreements only require the definition of a common protocol and format, such as an IEEE1516 Federation Object Model Template (OMT) or an extensible markup language (XML) schema. The content has to be agreed upon between the peers.
- The second category uses a *predefined model for the communication*. This model is

shared by the systems and they agree that information they provide will follow this model. Examples are all forms of predefined messages, the protocol data units (PDU) defined by IEEE1278 for Distributed Interactive Simulation (DIS) application, or reference models for the IEEE1516, such as the Real-time Platform Reference (RPR) Federation Object Model (FOM).

In both categories, the emphasis lies on data structure and context of an exchange of information between systems. In this view, a system is well and completely defined by its interface describing which structures which content can be produced or consumed in which context (leading to an "it's all about data movement" frame-of-mind in some M&S related communities).

The research of the authors of this chapter resulted in a recommendation to distinguish between information exchange requirements, information exchange capabilities, and information exchange needs. Too often developers assume that if information can flow from one system to another, this automatically ensures that the information can flow back as well. This, however, cannot be the general case. It is necessary to distinguish between the ability of a system to be the source or the target of a piece of information, i.e., the ability to produce the desired information or to consume the desired information. A simple example shows that both capabilities are not identical: if the desired information to be exchanged is the *sum* of all individual entities in a group and their *center of gravity* – as often used to aggregate vehicles into a military unit – a system that represents the original entities can produce the desired information, but it cannot consume the information for its own use. Furthermore, not all information that potentially could be exchanged needs to be exchanged in support of a common operation or business process. It would be a waste of resources to engineer all information exchange possibilities if it is already known that only a small subset will really be needed. These observations motivate the following definitions (Tolk et al., 2008):

- The set of related data elements in context needed to fulfill the operational or business requirements constitute *information exchange requirements*. In other words, this is the information required by the operational user of the system or system of systems.
- *Information exchange capability* is defined for each system. It is the set of related data elements, in context, that can be provided by the systems. This is the information that can be published or produced by a service or system.
- *Information exchange need* is also defined for each system. It is the set of data elements, in context, that can be consumed by the system. This is the information that can be subscribed to or consumed by a service or system.

Conceptual models must support the definition of information requirements, capabilities, and needs. The task is not trivial, in particular when looking at integration challenges of legacy systems into a system of systems.

The view of many system developers is that systems supporting the same domain naturally are using very similar, if not the same, conceptualization. However, the principle documented by Odgen and Richards (1923) still holds. Odgen and Richards distinguish between referents, concepts, and symbols to explain why communication often fails. Referents are objects in the real (or an assumed or virtual) world. When we communicate about the referents, we use perceptions or interpretations of these referents and capture them in concepts that reflect our viewpoint of the world as object, etc. and then use symbols to talk about our concepts.

The following figure shows the relation of this semiotic triangle to the M&S domain, as it is well known to M&S experts. They will easily recognize the similarity to the earlier mentioned validation and verification models as recommended by Sargent (2001) – where real world domain, conceptual model, and implemented model are distinguished – or the framework for M&S as recommended in Zeigler (2000) – the experimental frame with the source model, the model, and the simulator.

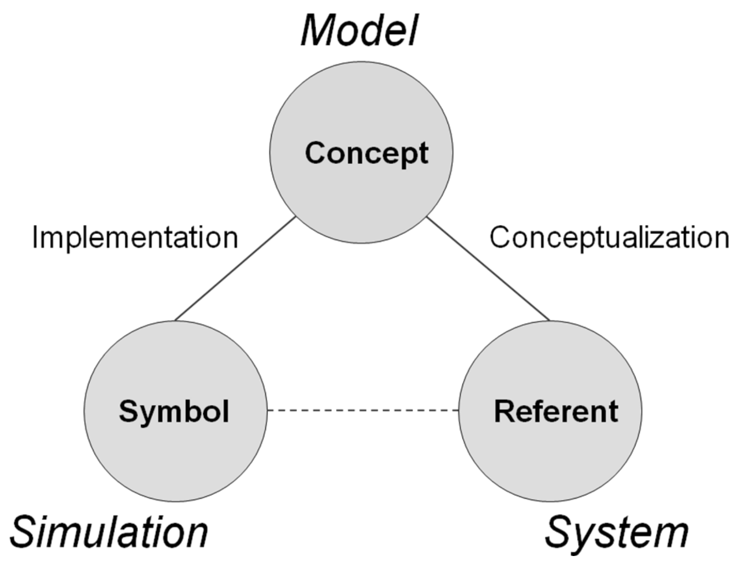

Figure 2: The Semiotic Triangle for M&S

It should be pointed out that the implementation does not reveal why the conceptualization was chosen, only which one was chosen. A common conceptualization can result in different implementations. In order to ensure composability, the conceptualization decisions need to be captured in addition to the implementation decision.

If the information exchange modeling category falls into the category of *predefined models for the communication*, the predefined model used serves as a common interpretation between two models when they exchange information. The PDUs defined in the DIS IEEE1278 Standard are a very good example of this case. This standard was developed in support of the armed forces to enable the exchange of information between simulators that collaborate on a virtual battle field, such as main battle simulators and infantry fighting vehicles. Whenever a preconceived event happens – such as one tank firing at another, two system colliding, artillery ammunition being used to shoot into special area, a report being transmitted using radio, a jammer being used to suppress the use of communication or detection devices, and more – the appropriate PDU is selected from the list of available PDUs and used to transmit the standardized information describing this event. The advantage of this solution is that every participant knows exactly what information needs to be provided and can be expected. The disadvantages are that every new event requires a new PDU, which means an extension of the standard, and – this is of particular interest for conceptual modeling – similar events will be reduced to the information used to describe them using the same PDU. What are only similar events in the simulators become equivalent events in the information exchange. If the system is sensitive regarding its initial conditions or exposes even *quasi*-chaotic behavior, such small adjustments can result in very different results.

The same observations made for IEEE1278 are true if other forms of predefined common information exchange models are used, such as the Real-time Platform Reference (RPR) Federate Object Model (FOM) as defined by the Simulation Interoperability Standards Organization (1999) or the Modeling Architecture for Technology, Research, and Experimentation (MATREX) FOM used by the US Army, as described by LeSueur , et al. (2006). The use of common *messages*, such as military messages in order to drive military command and control systems from a synthetic environment also falls into this category. Conceptually, it doesn't matter if the message is delivered in the form of a military text message or as a replication working on a military database: in both cases the information exchange is defined by the predefined model of these messages or replication instructions.

For conceptual modeling of information exchange, the main focus in the category of *predefined models for the communication* has to contribute to answering the question of whether the recommended – and in practical application often even mandated – predefined model for information exchange is sufficient and reflects all aspects of the operationally relevant information exchange requirements as well as the system specific representations as they are captured in the information exchange needs and information exchange capabilities. It should be pointed out that as – every other model – the predefined model is also an abstraction of reality with its assumptions, constraints, and simplifications. In the worst case, the information exchange capabilities and needs of participating systems will be *shoehorned* into a potentially inappropriate information exchange model.

In case the alternative category is used, which does not rely on predefined models but *peer-to-peer agreements* between sending and receiving systems, no predefined data must be used. This means, however, that every information exchange needs to be evaluated regarding its validity. With a predefined, and in some cases standardized, model as the backbone for information exchange, common understanding of which concepts are modeled and encoded by which symbols in the system is possible. For peer-to-peer agreements on the other hand, a formal way to capture the elements of the semiotic triangle for information exchange is needed, namely which concepts are used to capture the referents and how are they encoded in the implementation. As mentioned in last paragraph, this is needed for the predefined models as well, as it makes no difference if the information exchange is conceptually modeled between the information exchange means described by two systems or by one system and the predefined model.

Meta-data registries have been defined to support the consistent use of data within organization or even across multiple organizations. In addition, they need to be machine-understandable to maximize their use. Logically, the recommended structures for meta-data registries are strong candidates for capturing the results of conceptual modeling for information exchange. Part III of ISO/IEC 11179 Meta-data Registry (MDR) Standard defines structures that can be used for the purpose identified in this chapter. The following figure is based on the recommendations given in this standard as used in recent publications on model-based data engineering (MBDE) and extended in the following sections of this chapter as well. It shows four meta-data domains needed to capture data representation and implementation:

- The conceptual domain describes the concepts that were derived in the conceptualization phase of the modeling process. This domain comprises all the concepts that are needed to describe the referent or referents relevant to the information exchange.
- The property domain describes the properties that are used to describe the concept. Concepts are characterized by the defining properties. ISO/IEC 11179 refers to this domain as the data element concepts.
- The property value domain comprises the value ranges, enumeration, or other appropriate definition of values that can be assigned to a property. ISO/IEC 11179 refers to this domain as the value domain.
- Property instances capture the pieces of information that can be exchanged. They minimally comprise the value of one property, which can be interpreted as updating just one value, or they can become an *n*-tuple of n properties describing a group of associated concepts, which represents complex messages or updates for several objects. ISO/IEC 11179 calls these property instances data elements.

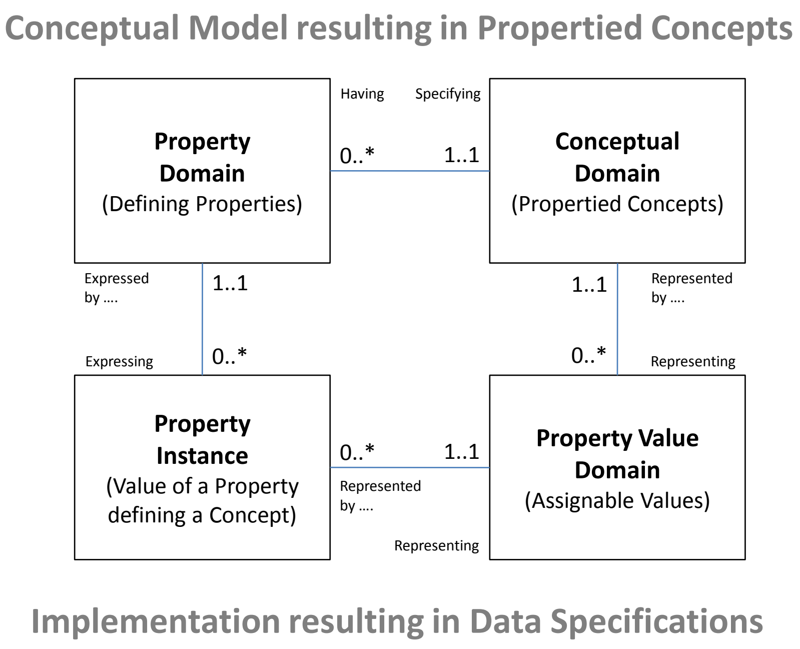
Conceptual Model resulting in Propertied Concepts
Property Domain (Defining Properties)
Conceptual Domain (Propertied Concepts)
Having
Specifying
0..*
1..1
Expressed by ....
Expressing
1..1
0..*
Represented by ....
Representing
1..1
0..*
Property Instance (Value of a Property defining a Concept)
Property Value Domain (Assignable Values)
0..*
1..1
Represented by ....
Representing
Implementation resulting in Data Specifications

Figure 3: Domains of Information Exchange Modeling

Traditionally, only the property instances are captured. From what has been specified in this chapter so far it becomes obvious that information needs to be specified in the context of the results of the underlying conceptualization to ensure the required semantic transparency. As the referent itself cannot be captured, because it is replaced by its conceptualization, the information exchange model must at least capture the conceptualization of the model used and cannot be limited to the symbols used for its implementation. Tolk, et al. (2008) introduce first ideas on how to use these structures to enable self-organization of information exchange, if machines are not only able to understand how the implementations are related to the conceptualization, but how conceptualizations of different models are related to each other. The following section introduces how these domains support data engineering.

Data Engineering

The notion of data engineering was introduced in the NATO Code of Best Practice (NCOBP) for Command and Control Assessment (NATO, 2002, p. 232) in support of the integration of heterogeneous data sources for common operations and operational analysis. While the NCOBP was written by international NATO experts, its application is not limited to military systems. The NCOBP was created more as an application-oriented introduction on how to conduct operations research studies on complex, complicated, and wicked problems, such as the command and control challenge in a multi-national organization with many independently developed information systems and not necessarily always aligned doctrinal viewpoints.
The NCOBP introduced data engineering as an engineering method to ensure that valuable resource data are best utilized. As defined in the NCOBP, data engineering consists of the following four main areas:

- *Data Administration*: Managing the information exchange needs including source, format, context of validity, fidelity, and credibility. As a result of the processes in this area the data engineer is aware of the data sources and their constraints.
- *Data Management*: The processes for planning, organizing, and managing of data including definition and standardization of the meaning of data as of their relations. Using the processes of this area, the data engineer unambiguously defines the meaning of the data.
- *Data Alignment*: Ensuring that data to be exchanged exist in all participating systems. Using the results of data management, target data elements needs and source data abilities can be compared. The data engineer identifies particular gaps that need to be closed by the system engineers responsible for the participating systems.
- *Data Transformation*: Technical process of mapping data elements from the source to the target. If all data are captured in the first three processes, data transformation can be automated by configuring XML translators (Tolk and Diallo, 2005).

MBDE introduces the notion of a *Common Reference Model* (CRM) in support of t data engineering processes. In MBDE, the definition of a common namespace captured in the form of a logical model is the starting point. The CRM captures the objects, attributes, and relationships that are susceptible of being exchanged during a federation. It is worth mentioning that in theory the four areas of data engineering are well-defined steps that can be conducted consecutively. In practical applications, however, systems and services are hardly ever documented as is needed for this application, so data engineering becomes an iterative process. Furthermore, it should be pointed out that the CRM is not a fixed model comparable to the predefined information exchange models described in the previous section, rather the CRM is gradually modified to reflect new concepts needed, requiring *extensions*, or to reflect new properties or additional property values, requiring *enhancement* of the CRM. The CRM is therefore the means-enabling mediation between the different viewpoints of the information exchange model as derived from the information exchange capabilities and needs of aligned services and systems based on the information exchange requirements derived from the operational use that need to be supported.

In the remainder of this section, we will show how to use the domains of information exchange modeling introduced earlier to directly support the areas of MBDE.

The first area of data engineering is data administration. If the data are not already structured and documented in form of an object or data model, this step is necessary. Conceptual modeling in support of data administration classifies each information exchange elements either as a value (V) that can be grouped with other property values of its domain (D) that can be assigned to property (P), or as a property can be grouped to identify a propertied concept (C), or as a concept that can be related to other concepts. At the end of this process, the domains of information exchange modeling for the information exchange capability (what can be produced as a data source) and the information exchange need (what can be consumed as a data target) for each system or service is documented. The following figure documents the result for two systems (or services) A and B:

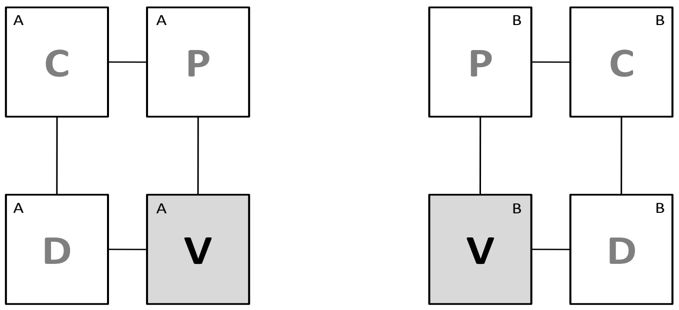

Figure 4: Data Administration

The second area of data engineering is data management. Using the logical information exchange modeling elements, the concepts (C) and the defining properties (P) of information exchange capability models and information exchange need models, common concepts and properties are identified. The result is a set of propertied concepts to which the elements of the information exchange capability models can be mapped and that can be mapped to the elements of the information exchange need models. In the case of MBDE, these propertied concepts build the CRM, which is the logical model of the information exchange that can take place between the systems or services. It is worth stating that this shows that such a logical CRM always exists, whether it is made explicit or not, as whenever a property from system A is mapped to a property of system B. This constitutes a property that makes sense in the information exchange between the two systems, which is expressed by the CRM. The concepts of the CRM serve two purposes:

- They build the propertied concepts of the properties of the CRM and as such help the human to interpret the information exchange categories better. In particular when a CRM is derived from the information exchange requirements, this is very helpful.
- They conserve the context of information exchange for the receiving systems, which is their information exchange need.

The following figure shows the result of the data management processes for two systems A and B:

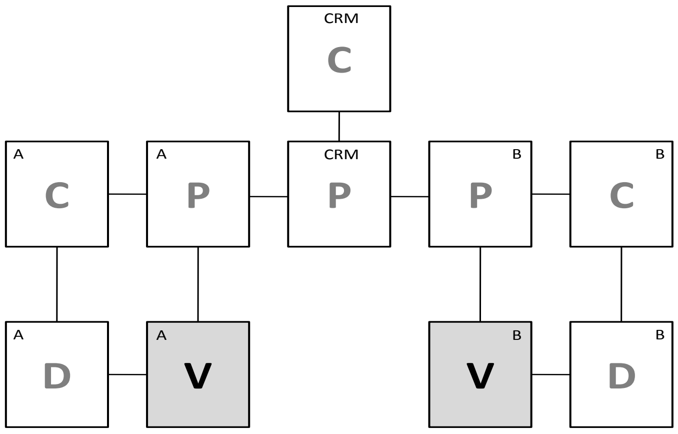

Figure 5: Data Management

The third area is data alignment. In this area of data engineering, it is necessary to distinguish between the logical and physical aspect of data alignment. Logically, information can be exchanged between two systems if the properties of system A have properties in system B that are logically equivalent, which means that there is a connection via the CRM. Not every property in system A will have a counterpart in system B, and only for those properties that are connected to a property of the CRM that is part of the information exchange requirements model, is the mapping operationally required. If a connection from system A to system B exists for every property of the CRM representing a piece of the information exchange requirement, system A and B are logically aligned under the CRM. It is worth mentioning that logical alignment is not mathematically symmetrical and dependent on the CRM.

The next step is the physical alignment. While two properties may be aligned under the CRM logically, their actual representations can be a challenge, as the property value domains must be equivalency classes, or at least the class of the information exchange need property value domain must be in the range of the information exchange capability property value domain. The modeling and simulation literature deals with problems derived from this step under the term multi-resolution modeling (Davis and Huber, 1992, Davis and Hillestad, 1993, Reynolds et al., 1997, Davis and Bigelow, 1998).

Within this chapter, we define system A to be aligned with system B under the CRM if they are logically aligned and an injective function can be constructed from the property value domain of the source property in system A to the target property in system B. The following figure shows the result of these steps.

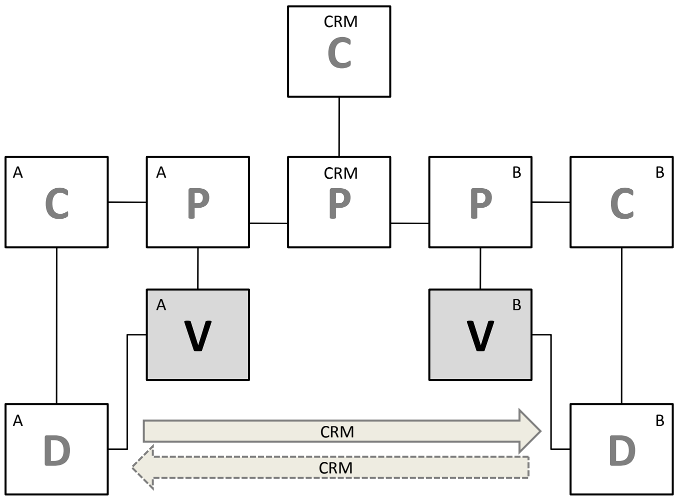

Figure 6: Data Alignment

Finally, data transformation can map the information exchange elements to each other. This process of data mediation between the different viewpoints represented in the participating systems is already specified by the other three areas of data engineering, so that this step can be automated by using the results of the first three areas:

- Data administration provides the needed structures of the information exchange capabilities (system A) and information exchange needs (system B).
- Data management provides the identification of common properties on the logical level, resulting in the CRM. The CRM can be constrained to those concepts and properties that satisfy the information exchange requirements of the operation to be supported.
- Data alignment evaluates the logical alignment and the physical alignment. For each property in the CRM representing a part of the information exchange requirement, a logical counterpart is needed in the information exchange capability of system A (data can be produced) and in the information exchange need of system B (data can be consumed). In addition, the property value domains used to implement these properties must be mapped to each other.
- The result is a function that maps all relevant information exchange elements of system A under the CRM to the logically correct information exchange elements of system B. If the relations between A and B or not only injective, but also surjective, an inverse function exists as well.

The following figure illustrates the final result of data transformation:

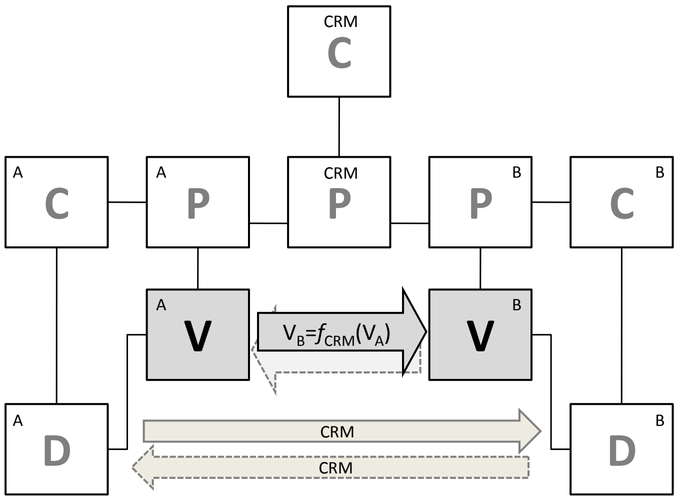

Figure 7: Data Transformation

Tolk and Aaron (2009) give application examples of MBDE using real-world CRM derived from projects conducted for the US Army and the US Joint Forces Command and show how these applications can be generalized. In their contribution, they emphasize in particular the necessity of an efficient engineering management in addition to the conceptual and technical constraints.

Process Engineering and Constraint Engineering

While the emphasis so far has been on the information exchange model in the form of conceptually understanding what information exchange elements are exchanged between systems, the LCIM described earlier in this chapter clearly motivates two additional methods, namely process engineering and constraint engineering:

- Data engineering focuses clearly on the syntactic and semantic layers of the LCIM. However, it only touches the pragmatic layer and is of only limited use for the remaining layers.
- To understand how the data are used within a system or what is needed to produce the data so it can be transmitted or how the system states are going to change once the data are produced or consumed requires a similar formalism for processes, similar to the one introduced for data in the previous section. This is the subject of ongoing research and will result in process engineering.
- The assumptions, constraints, and simplifications about a model are part of conceptual modeling, and are important elements that need to be evaluated if two systems or services are to be composable. These assumptions are often not reflected in the implementation and unfortunately are rarely documented. Spiegel et al. (2005) present a simple physics-based model of the falling body problem to show how many assumptions are implicitly accepted within models, and how hard it is for experts to reproduce them “after the fact.” Capturing constraints and assumptions in machine-understandable form is the subject of ongoing research and will result in constraint engineering.

While both engineering methods (process and constraint engineering) are currently not as mature as data engineering regarding applications, we will give an overview of the current state of research in the remainder of this section.
Tolk, et al. (2009) describe process engineering, similar to data engineering as an approach to align the process of separate systems. As with data engineering, process engineering follows a method of four areas:

- *Process Administration*: The processes that are included in the systems to be made interoperable must be identified, including the source of each process, and the operational context in which it acts.
- *Process Identification*: This involves organizing and managing the processes, and their specifications. Each process must be specified in order to enable the following steps. The requirements for such a specification will be defined in the next paragraph.
- *Process Alignment*: The defining attributes of the processes, as organized within process management, determine where, in the life of the system, the process will occur, and what affect it has internally on the system, as well as what the process means externally for interoperability. These attributes must be aligned so that the resulting effects of interacting processes can be determined.
- *Process Transformation*: Finally, if the resulting effects of interacting processes do not produce a desired outcome, it may be necessary to perform some transformation to one or more attributes of some of the processes in question. The data concerning the processes that results from the first three steps will enable this to happen.

In order to apply the steps of these four areas of process engineering, specifics concerning each process must be made available, namely a detailed enough description of the specific characteristics of each process is required. A *process description language* and a *process-algebra* designed in support of process engineering are objects of ongoing research, resulting so far in the identifications of first process-defining attributes:

- *Initialization Requirements:* Unambiguous definition of process-specific object-attribute values that must exist, including object-object relationships that must be in place, for the process to be feasible. Additionally, system operational requirements for the initialization of the process are specified, if required.
- *Time:* Capturing the dynamic characteristics of the process, such as when does the process start and how long it takes to complete the process, and in terms of complex processes, the rate of progress.
- *Effects:* A process affects some change in the attribution of an object in the system. Complex processes may affect more than one attribute in a single object, or perhaps more than one object. In this category of defining attributes, the range of these effects that take place, including how these effects occur, are captured. The changed attributes can identity attributes of the system, or can possibly be coincidental attributes.
- *Halting Requirements:* Some processes will terminate given a certain passage of operational time, others require specification of what conditions cause the process to halt. This specification is more likely to be required for a complex process (where more than one attribute or object change is part of the process specification) than for a simple

process.

- *Post-conditions:* The state of the system once a process has halted including specifying the nature of the process or related processes once it has expired, such as termination, waiting in idle mode, etc.

Once a process description language that captures these attributes is formally presented and used to define processes within a model of a system, then the application of process-algebra will lead to the four steps of process engineering that can be understood and supported by machines. Current research focuses on extensions into this direction using the web-ontology description for services, such as OWL-S (Martin et al., 2004). Related ideas have been published by Rubino et al. (2006) and others.

King (2009) describes the foundation for constraint engineering. In this work, he outlines a process for capturing and aligning assertions as well as a formalism to enable the envisioned machine support. Using the falling body problem described by Spiegel et al. (2005) mentioned earlier, he captures all applicable forces in an ontology using the Protégé Tool. He next develops a formalism to represent assertions, as the assertions must be encoded in a knowledge representation language to make them machine-understandable. Each proposition consists of its axioms and logical assertions that relate it to other concepts and propositions. Having the constraints and the assertion formalized as recommended here, it then becomes possible to compare the assertions, assumptions, and constraints using logical reason to identify incompatibilities on the conceptual level, as envisioned in King (2007). The idea is captured in the following figure.

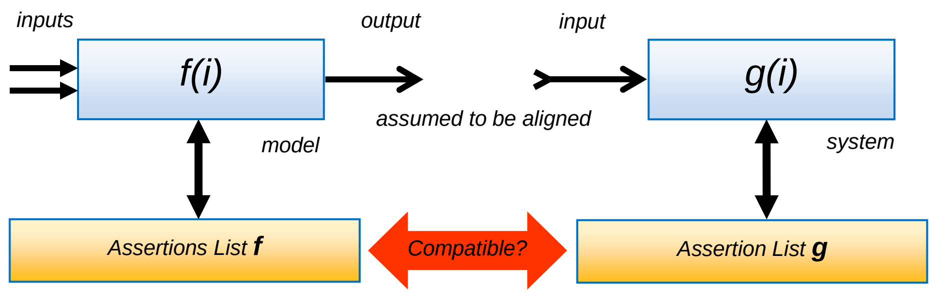

Figure 8: Evaluating Compatibility of Assertion Lists (King, 2009)

As stated ab[illegible] s the assertions, and a formalism for representing them. The formal model of assertion has four components that are described in the following list.

- *Use function:* The use function describes the role of the assertion in potentially modifying system behavior. The value will be the one of the terms in the following list: [uses | does not use | ignores | requires | denies]. The use function plays an important role in processing assertion lists. It establishes the relevance of the proposition with respect to the model and with respect to the role of the proposition when integrating the model with a system.
- *Referent:* The referent of an assertion is the entity to which it refers. A referent can be an object, a model, a process, a data entity, a system, or a property of one of these. When an

assertion acts as a constraint, the referent is what is being limited by the proposition.

- *Proposition*: The proposition of an assertion is the statement that it is making. Propositions are not restricted to simple concepts—they may encompass the content expressed by theories, books, and even whole libraries.
- *Scope:* This is an optional description that extends the portions of the overall system to which the assertion applies. Scope can limit consideration to a system component, the system as a whole, the environment of the system, or to combinations of these (e.g., *component-environment* scope means that the scope of assertion is the relationship between the component and its environment.) If scope is not specified, then the assertion has *component* scope. Scope can be stated explicitly or implicitly.

The formal description of an assertion is therefore a tuple structured as:

```
        Assertion <=> (referent useFN Proposition <scope>)
```

In order to apply these ideas, three steps need to be conducted, that are captured in the following list. The viewpoint is slightly different from the data and process engineering areas, but the results are comparable.

- *Capture assertions:* The first step is capturing the assertion propositions (assumptions, constraints, implemented considerations and competencies) for the model, system and environment that are known within the scope or that are otherwise important. Each proposition represents a concept that is initially expressed as a natural language statement about the problem, one or more of its components, or a particular solution. The result of this step is the formal identification of the main concepts that will form the basis of the ontology content.
- *Encode propositions:* The list of propositions expressed in natural language statements must be encoded in a knowledge representation language to make them machine-understandable. Each proposition will consist of axioms and logical assertions that relate it to other concepts and propositions. In this step, this encoding happens. King (2009) used the Protégé Tool, but every other tool supporting the encoding in logical form is applicable. The second step finally consists of assigning the use function, referent and scope to each proposition in both the model and the system lists. Experience in the research shows that the analyst should be prepared to make several iterations through this process step as the assertion lists are refined. The output of this step is list of statements encoded in a knowledge representation language—the list of assertions for the both component and system.
- *Compare assertion lists:* The task of comparing assertion lists requires a multi-level strategy that is described in detail in King (2009). The full details go beyond the scope of this chapter except to note that the comparison is steered by the use function assigned to each proposition in the previous step. The method is used to compare the list of assertions about the model to be composed with the statements about the system (see Figure 8). Each proposition represents a concept and there are different ways that concepts can match. The topic of semantic similarity—deciding if, and how closely concepts match—is the subject of much current study, particularly with respect to research into the Semantic Web. The issue is a complex process influenced by many different factors or characteristics, however analysis such as performed by Kokla (2006) reveals four

possible comparison cases between concepts: equivalence, when the concepts are identical in meaning; subsumption (partial equivalence), when one concept has broader meaning than the other; overlap (inexact equivalence), when concepts have similar, but not precisely identical meanings; and difference (non-equivalence), when the concepts have different meanings. The first three are potentially useful for comparing assertions lists. Testing for equivalence is straightforward and involves searching for text or label matches. Subsumption is handled by first order logic or a subset of FOL such as a Description Logics reasoner operating on an OWL-DL ontology. Determining overlap requires greater sophistication in reasoning, such as analogical reasoning described by Sowa and Majumdar (2003) or the agent-based metalevel framework for interoperation presented by Yilmaz and Paspuletti (2005).

King (2009) showes that with these steps it was possible to capture conceptual misalignments of services that did not show up on the implementation level. In other words: without capturing the results of conceptual modeling appropriately, these services would not have been identified as not composable, and a composition of them would have been technically valid but conceptually flawed, leading to potentially wrong results. Depending on the application domain such as simulation composition is used in, such conceptual misalignment can lead to significant harm and even the loss of human lives or economic disasters, in particular when decision makers base their decisions on flawed analysis.

In summary, conceptual modeling must produce machine-readable artifacts in support of the levels of interoperation identified in the LCIM. On the conceptual level, assertions need to be defined to avoid conceptual misaligned compositions. On the dynamic level, the system states governing the processes need to be captured. Process engineering lays the foundation for this by defining the artifacts for the pragmatic level. Data engineering focuses on the semantic and the syntactic level. Research work is ongoing and will hopefully soon lead to agreed terms and standardized artifacts on all layers of interoperation.

It is worth mentioning that the rigorous application of mathematics ensures the consistency of conceptualizations and their implementations and not that the conceptualizations are correct. In other words, it is possible to evaluate if two different conceptualizations can be mapped to each other and if transfer functions exists between resulting implementations. The three engineering methods described here support this evaluation and enable the support thereof by machine, if all needed artifacts are available and can be observed as discussed earlier in this chapter.

Ontological Means

In order to express all these findings in machine-understandable form, the means provided by current simulation standards, such as IEEE1278 and IEEE1516, are not sufficient. Ontological means have been applied for similar tasks in the context of the Semantic Web, so that they are at least potential candidates for support of semantic simulation interoperation as well. Obrst (2003) introduced a spectrum of ontological means that is captured in principal in the following figure:

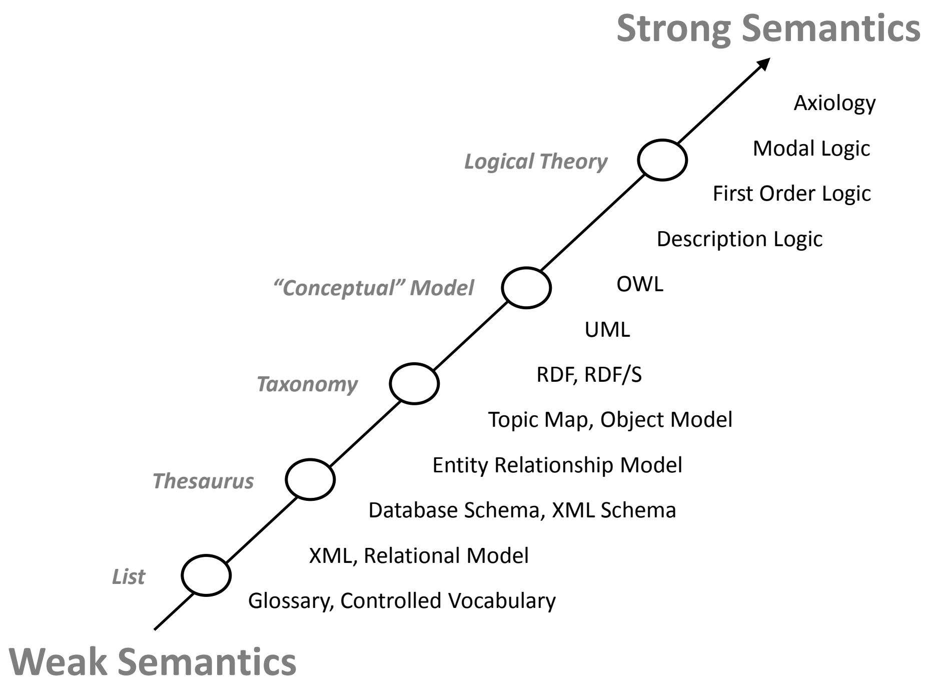

Figure 9: Ontological Spectrum and Methods

The ontological spectrum emphasizes the viewpoint that ontologies are not a radically new concept, but that they are a logical step in the process of increasingly organizing data: starting with pure enumeration, thesauri relate similar terms and taxonomies relate them in an order. Capturing assumptions and constraints adds the conceptual component. Formulating them in form of logical expressions and axioms leads to logical theory and makes the represented artifacts understandable for software systems, such as intelligent agents.

Recker and Niehaves (2008) broadened the scope, limits and boundaries regarding ontology-based theories for conceptual modeling beyond the pure technical discussions by focusing on three additional questions:

- What does it mean to engage in conceptual modeling?
- What does it mean to evaluate the outcome of conceptual modeling?
- What does it mean to achieve quality in conceptual modeling?

Following the work of Recker and Niehaves (2008), it is possible to distinguish between conceptual models that are defined to *collectively construct an artifact that reflects subjectivity and purpose* on one hand side, and conceptual models that are defined to *produce a direct representation of an external reality* on other hand. The viewpoint presented in this chapter favors the second viewpoint, as we request a machine-understandable description of the system or the service in order to identify applicable solutions, select the best one of these, compose the solutions and orchestrate their execution. This can be easier achieved by direct objective representation, although the subjective aspects can become interesting with that as well. Furthermore, Recker and Niehaves (2008) distinguish between testing of conceptual models by comparing them to a reference model of reality versus consensus building of experts. In the light of machine-understanding, only the first option makes sense. Finally, the quality needs to be measured by the degree to which properties of the modeled portion of reality are represented in the conceptual model. The alternative presented by Recker and Niehaves (2008) – its perception as a good model by a social community – is not objective driven. Nonetheless, remembering the lessons recorded in Tolk and Aaron (2009), a consensus between engineers and managers is mandatory for successful procedures for data engineering, including the conceptual modeling aspects. As such, the observations of Recker and Niehaves (2008) deserve special attention for setting up the necessary management structures.

Summary

In this chapter, we introduced ongoing research exploiting the concepts needed to annotate systems and services to enable to identify applicable solutions, select the best solutions of those available, compose the solutions, and orchestrate their execution. We showed that artifacts developed during conceptual modeling are necessary to enable composable solutions and that means provided by the ontological spectrum can be used to capture them in machine-readable form. Without such annotations that include conceptual modeling results, the concepts of system of systems and service-oriented architectures remain will incomplete.

Conceptual modeling for composition of model-based complex systems supported by the methods of data engineering, process engineering, and constraint engineering produces the artifacts necessary to enable the lossless mediation between viewpoints as captured in the conceptualization of different models. The concept of lossless mediation between viewpoints of models is superior to prescribing an enterprise wide data model, as it identifies shareable information between models that need to contribute to a common operation instead of shoehorning the underlying conceptualizations into yet another viewpoint.

The rigorous application of mathematics to produce artifacts for annotation is a necessary requirement to enable consistent systems of systems or service-oriented architectures as envisioned in the introduction of this chapter. The ultimate goal must be to capture also the system specifications and requirements in appropriate mathematical structures to allow agent to identify, select, compose, and orchestrate systems and services that satisfy these specifications and requirements. Therefore, this chapter is only the first step in setting a common research and alignment agenda for conceptual modelers, system architects, requirement developers for net-centric operations, and other communities that contribute already or that will contribute in the future to this common domain of interest.

References

- Benjamin, P., K. Akella, and A, Verma (2007), “Using ontologies for simulation integration,” *Proceedings of the Winter Simulation Conference*, pp. 1081-1089, IEEE Computer Society, Washington, DC
- Brade, D. (2000), “Enhancing M&S Accreditation by Structuring V&V Results,” *Proceedings of the Winter Simulation Conference*, pp. 840-848, IEEE Computer Society, Washington, DC

- Burstein, M. H., and D. V. McDermott (2005), "Ontology Translation for Interoperability Among Semantic Web Services," AI Magazine 26(1) : 71-82
- Dahmann, J. S. (1999): "High Level Architecture Interoperability Challenges," Presentation at the *NATO Modeling & Simulation Conference*, Norfolk VA, October 1999, NATO RTA Publications
- Davis, P.K., and Huber R.K. (1992). Variable-resolution combat modeling : motivations, issues, and principles. *RAND Notes*, Santa Barbara, CA
- Davis, P.K., and J.H. Bigelow (1998): *"Experiments in MRM,"* RAND Report MR-100-DARPA, Santa Barbara, CA
- Davis, P.K., and R. Hillestad (1993): "Families of models that cross levels of resolution: issues for design, calibration and management," *Proceedings of the Winter Simulation Conference*, pp: 1003-1012, IEEE Computer Society, Washington, DC
- Fishwick , P. A. (2007), "Handbook of Dynamic System Modeling," Chapman & Hall/CRC Press LLC, 760 pages
- Gruber, T. R. (1993), "A Translation Approach to Portable Ontology Specification," *Journal of Knowledge Acquisition*, 5:199-220
- Guizzardi, G., and T. Halpin (2008), "Ontological foundations for conceptual modeling," *Journal of Applied Ontology*, 3(1-2):1-12
- Hofmann, M. (2004): "Challenges of Model Interoperation in Military Simulations," *SIMULATION*, 80:659-667
- Institute of Electrical and Electronics Engineers (1990)" "*A Compilation of IEEE Standard Computer Glossaries,"IEEE Press,* New York
- Institute of Electrical and Electronics Engineers IEEE 1278 Standard for Distributed Interactive Simulation
- Institute of Electrical and Electronics Engineers IEEE 1516 Standard for Modeling and Simulation High Level Architecture
- International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) (2003): "Information technology meta-data registries part 3: registry metamodel and basic attributes," ISO/IEC 11179-3:2003
- King, R. D. (2007): "Towards Conceptual Linkage of Models and Simulations," Proceedings of the Spring Simulation Interoperability Workshop, IEEE Computer Society, Washington, DC
- King, R. D. (2009): "On the role of assertions for conceptual modeling as enablers of composable simulation solutions," Ph.D. thesis, Old Dominion University, Norfolk, VA
- Kokla M. 2006: "Guidelines on Geographic Ontology Integration," ISPRS Technical Commission II.
- LeSueur, K. G., K. Yetzer, M. Stokes, A. Krishnamurthy, and A. Chalker (2006): "Distributed tests: an army perspective," *Proceedings of the HPCMP Users Group Conference*, 26-29 June, IEEE Computer Society, Washington, DC, 337-346
- Martin, D., M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. R. Payne, E. Sirin, N. Srinivasan, and K. Sycara (2004): "OWL-S: Semantic Markup for Web Services," *Technical Report UNSPECIFIED*, Member Submission, World Wide Web Consortium (W3C)
- Muguira, J. A., and A. Tolk (2003): "The Levels of Conceptual Interoperability Model (LCIM)" *Proceedings Fall Simulation Interoperability Workshop*, IEEE Computer Society, Washington, DC
- Muguira, J. A., and A. Tolk (2006): "Applying a Methodology to identify Structural Variances in Interoperations," *Journal of Defense Modeling and Simulation*, 3(2):77-93
- NATO (2002): *"NATO Code of Best Practice for Command and Control Assessment,"* RTO-TR-

081 AC/323(SAS-026)TP/40, CCRP Press, Washington, DC

- Obrst, L. (2003): “Ontologies for Semantically Interoperable Systems,” *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, November 2003, pp. 366-369.
- Page, E. H., R. Briggs, and J. A. Tufarolo (2004), “Toward a Family of Maturity Models for the Simulation Interconnection Problem,” *Proceedings of the Spring Simulation Interoperability Workshop*, IEEE Computer Society, Washington, DC
- Petty, M. D. (2002): “Interoperability and Composability,” *Modeling & Simulation Curriculum* of Old Dominion University, Old Dominion University
- Petty, M. D., and E. W. Weisel (2003), "A Composability Lexicon", *Proceedings of the Spring Simulation Interoperability Workshop*, pp. 181-187, Orlando FL
- Petty, M. D., E. W. Weisel and R. R. Mielke (2003): "A Formal Approach to Composability," *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, pp. 1763-1772, Orlando FL
- Recker J., and B. Niehaves (2008): “Epistemological perspectives on ontology-based theories for conceptual modeling,” *Journal of Applied Ontology*, 3(1-2):111-130
- Reynolds Jr., P. F., Natrajan A, and Srinivasan, S. (1997). Consistency Maintenance in Multi-resolution Simulations. *ACM Transactions on Modeling and Computer Simulation*, 7(3): 368–392
- Robinson, S. (2007): "Conceptual modelling for simulation Part I: definition and requirements," *Journal of the Operational Research Society*, 59 (3):278-290
- Rubino, R., A. Molesini, and E. Denti (2006): “OWL-S for Describing Artifacts,” Proceedings of the 4th European Workshop on Multi-Agent Systems EUMAS'06, Lisbon, Portugal
- Sargent, R. G. (2001): “Verification and Validation: Some Approaches and Paradigms for Verifying and Validating Simulation Models,” *Proceedings of the Winter Simulation Conference*, pp. 106-114, IEEE Computer Society, Washington, DC
- Simulation Interoperability Standards Organization (1999): “Real-time Platform Reference Federation Object Model (RPR-FOM) Version 1.0,” SISO-STD-001.1-1999, IEEE Computer Society, Washington, DC
- Spiegel M., P. F. Reynolds Jr., and D. C. Brogan (2005): “A Case Study of Model Context for Simulation Composability and Reusability,” *Proceedings of the Winter Simulation Conference*, pp: 437-444, IEEE Computer Society, Washington, DC
- Sowa, J. and A. Majumdar (2003): "Analogical Reasoning," In de Moor, Lex, and Ganter (Eds.): *Conceptual Structures for Knowledge Creation and Communication*, Springer, Berlin, pp. 16-36
- Sycara, K. P. (2002): “Infrastructure and Interoperability for Agent-Mediated Services,” in *Proceedings of the 13th international Symposium on Foundations of intelligent Systems* (June 27 - 29, 2002). M. Hacid, Z. W. Ras, D. A. Zighed, and Y. Kodratoff, Eds. Lecture Notes In Computer Science, vol. 2366. Springer, London
- Tolk, A. (2006): “What Comes After the Semantic Web - PADS Implications for the Dynamic Web,” *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation,* pp. 55-62, IEEE Computer Society, Washington, DC
- Tolk, A., and R. D. Aaron (2009), “Model-based Data Engineering for Data-Rich Integration Projects – Case Studies Addressing the Challenges of Knowledge Transfer,” *Engineering Management Journal (in press)*
- Tolk, A., and S. Y. Diallo (2005): “Model-based data engineering for web services,” *IEEE Internet Computing*, 9(4): 65-70, July/August
- Tolk, A., S. Y. Diallo, and C. D. Turnitsa (2008): “Mathematical Models towards Self-Organizing Formal Federation Languages based on Conceptual Models of Information Exchange

Capabilities," *Proceedings of the Winter Simulation Conference*, pp: 966-974, IEEE Computer Society, Washington, DC

- Tolk, A., S. Y. Diallo, and C. D. Turnitsa (2009): "Data Engineering and Process Engineering for Management of M&S Interoperation," *Proceedings of the Third International Conference on Modeling, Simulation and Applied Optimization, Sharjah, U.A.E January 20-22 2009*
- Turnitsa, C. D. (2005), "Extending the Levels of Conceptual Interoperability Model," *Proceedings Summer Computer Simulation Conference*, IEEE Computer Society, Washington, DC
- Wang, W., A. Tolk, and W. Wang (2009): "The Levels of Conceptual Interoperability Model – Applying Systems Engineering Principles to M&S," in *Proceedings of the Spring Simulation Multiconference,* IEEE Computer Society, Washington, DC
- West, M. (2009): "Ontology Meets Business," in Tolk, A., and L. C. Jain (Eds): *Complex Systems in Knowledge-based Environments: Theory, Models and Applications*, SCI 168, pp: 229-260, Springer, Heidelberg
- Yilmaz, L. (2004): "On the Need for Contextualized Introspective Simulation Models to Improve Reuse and Composability of Defense Simulations," *Journal of Defense Modeling and Simulation,* 1(3):135-145
- Yilmaz, L. and A. Tolk (2006): "Engineering ab initio dynamic interoperability and composability via agent-mediated introspective simulation," in *Proceedings of the Winter Simulation Conference*, pp: 1075-1182, IEEE Computer Society, Washington, DC
- Yilmaz, L, and Paspuleti, S. 2005: "Toward a Meta-Level Framework for Agent-Supported Interoperation of Defense Simulations," *Journal of Defense Modeling and Simulation* 2 (3): 161-175.
- Zeigler, B. P. (1986): "Toward a simulation methodology for variable structure modeling," in Elzas, M.S. , T.I. Oren, and B.P. Zeigler (Eds): *Modeling and Simulation Methodology in the Artificial Intelligence Era,* pp: 195-210, Elsevier Scientific Pub., Amsterdam, The Netherlands
- Zeigler, B. P., H. Praehofer, and T. G. Kim (2000), *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*, Academic Press, Elsevier, Amsterdam, The Netherlands