Motivating Use Cases for the Globalization of DSLs

Betty H.C. Cheng[1] and Thomas Degueule[2] and Colin Atkinson[3] and Siobhan Clarke[4] and Ulrich Frank[5] and Pieter J. Mosterman[6] and Janos Sztipanovits[7]

[1] Michigan State University, USA,
[2] University of Rennes and Inria, FR
[3] University of Mannheim, UK
[4] Trinity College, Dublin
[5] University of Duisburg-Essen, Germany
[6] MathWorks, USA
[7] Vanderbilt University, USA

Abstract. The development of complex software-intensive systems involves many stakeholders who contribute their expertise on specific aspects of the system under construction. Domain-specific languages (DSLs) are typically used by stakeholders to express their knowledge of the system using dedicated tools and abstractions. In this chapter, we explore different scenarios that lead to the globalization of DSLs through two motivating case studies – a command and control wind tunnel and a smart emergency response system – and outline the concrete engineering challenges they raise. Finally, we list some of the general research challenges related to the globalization of DSMLs and discuss some promising approaches for addressing them.

Keywords: multi-model integration, language integration

1 Introduction

Languages have been used in the development and evolution of software systems since the beginning of the computer industry. For many years, software engineers only needed to concern themselves with one language, the one general purpose programming language typically used to program an application (e.g. FORTRAN, COBOL, C or Pascal). Today, however, software engineers have to cope with a vast array of languages, supporting descriptions of different aspects and parts of software systems from the viewpoints of multiple stakeholders, levels of abstraction and concerns. This creates many new challenges for languages engineers beyond the traditional need to define a language's syntaxes (concrete and abstract) and semantics. Now languages engineers have to cope with multiple semantic interactions between the entities and concepts described by languages and cope with the co-evolution and management of families of interrelated languages.

Traditionally these challenges have been addressed in ad hoc ways, primarily at the level of the entities and concepts described by languages (i.e. at the level of

statements made in a language) rather than at the level of the languages themselves (i.e. at the level of language definitions). However, as domain-specific modeling technologies have matured, and the problems of engineering single-purpose languages solutions have been mastered, new conceptual tools and technologies for engineering heterogeneous solutions at the language level have started to emerge. Researchers now have the means to create new methodologies and tool ecosystems for combining independently-developed language fragments into new solutions that can leverage the vast amounts of knowledge and experience embedded within language definitions. The challenge of language globalization is to realize this vision.

Language globalization therefore aims to improve the way software systems are developed and evolved with a view to raising software quality and reducing costs. A major source of motivating use cases for this technology can be found in the realm of software systems engineering. Language globalization can be expected to improve the way in which key systems engineering challenges can be addressed, such as enhancing the functionality and information content of software systems, as well as adding new views and view types by which stakeholders can visualize them. This chapter illuminates the key language globalization challenges by exploring the language exploitation opportunities occurring in typical software development use cases. In particular, we present a set of concrete use cases in the context of two real-world example applications. After introducing these two applications and presenting the language engineering challenges they raise, we discuss the resulting research challenges.

2 Command and Control Wind Tunnel (C2WT)

The C2WT[8] is a model-integrated distributed simulation environment developed at the Institute for Software Integrated Systems at Vanderbilt University for complex, multi-modeling simulation tasks frequently required in virtual prototyping, end-to-end mission simulation and resilience studies. The application scenario below led to its original development as a part of an Air Force Office of Scientific Research (AFOSR) project.

2.1 Application: Evaluation of command and control architectures in mission scenarios

The evaluation of emerging command and control (C2) architectures necessitates a sophisticated modeling and simulation infrastructure that supports the concurrent modeling, simulation, and evaluation of (1) the C2 system architecture (advanced system-of-systems modeling), (2) the mission environment (scenario modeling and generation), and the (3) human organizations and (group and individual) decision-making processes (human performance and man-machine interaction modeling). Using simulated C2 environments to evaluate design concepts, validate new systems and components, and explore hazardous as well as

[8] https://wiki.isis.vanderbilt.edu/OpenC2WT/

ambiguous scenarios is easily justified from both cost and practicality perspectives.

An example of this application is shown in Figure 1. The mission scenario focuses on flying teleoperated Unmanned Aerial Vehicles (UAVs) in an urban environment for finding and tracking moving vehicles on the ground [15]. Each UAV is controlled by a human operator who inspects a video stream from the on-board sensor and remotely controls the UAV. The operators coordinate their track and search activities amongst each other and with a remote command center. Mission success is measured in terms of the time required for finding a moving vehicle with specific characteristics and the length of time tracking the vehicle without losing it. The specific evaluation scenarios examine the impact on mission success UAV characteristics (such as mobility, level of autonomy, sensor resolution), network attacks, allocation of decision responsibilities in the C2 architecture, and others.

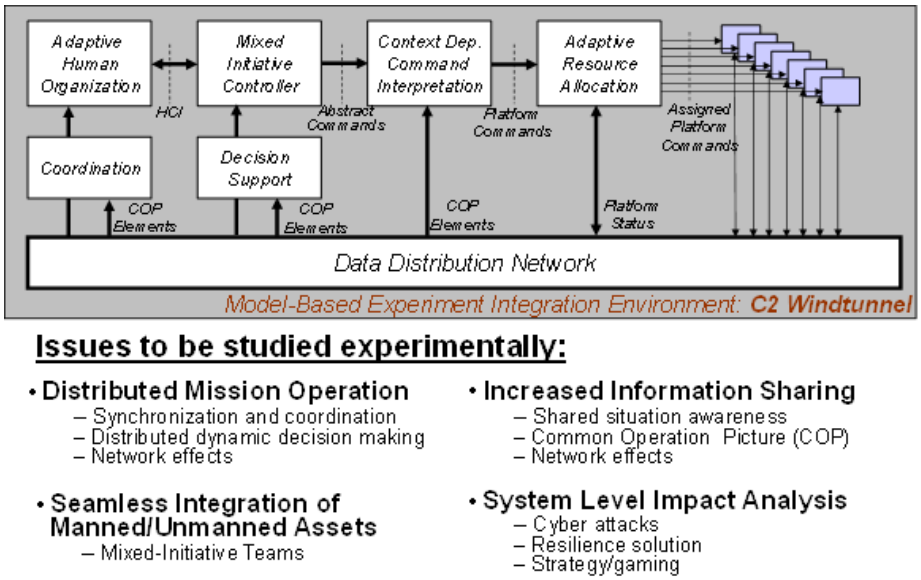

Fig. 1. Command and Control (C2) Architecture Analysis

The scenario identifies a network of interacting simulations (Figure 2). The individual modeling languages were selected based on the relevance of their respective domains to the simulation goals: Colored Petri Net (CPN) for modeling decision processes and interactions in command and control organizations, Simulink/Stateflow for modeling vehicle dynamics and controller dynamics, DEVS for modeling abstract behavior of software components, OMNeT for modeling communication links, and Google Earth for modeling motion in specific 3D environments. In addition, the overall simulation was executed in real-

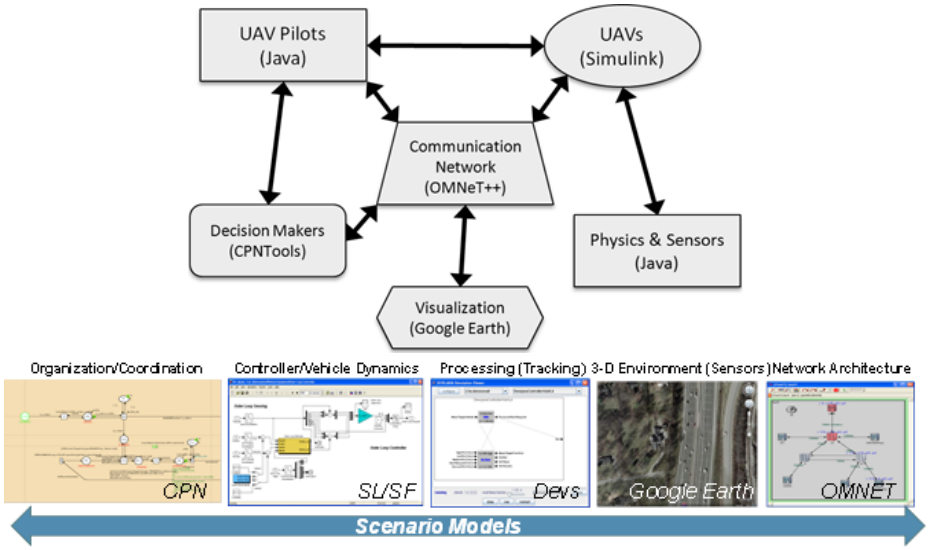

Fig. 2. Modeling Languages and Technologies Involved in the C2WT

time interacting with two UAV operators. The choice of modeling languages is strongly motivated by the need for reusing existing model libraries and existing simulators. However, neither of them was designed for incorporating all the other modeling abstractions.

2.2 Technical Challenges

Complex command and control environments have many disparate facets that need to be modeled and simulated. The constituent models include decision processes, dynamics of moving objects in a 3-D environment, sensors and information flows, communication networks, operator work stations, and mission scenarios. The modeling languages have different timing semantics (continuous time, discrete time, discrete event) and data semantics (3-D geometry, commands, physical variables). The simulations running on different simulation engines need to be coordinated, and the data needs to be routed. As a result, a heterogeneous collection of integrated simulations, all acting in a tightly coordinated environment, must be employed.

Individual simulations comprise two parts: a domain-specific model, such as a model of a flight control system in Simulink, and an underlying simulation engine, such as the Simulinks solver. Each modeling language has its own unique execution semantics as implemented by its simulation engine. All of the executions in federated (integrated) simulations must be coordinated in a meaningful way to ensure that the larger C2 simulation environment is useful. The problems in developing integrated simulations can be decomposed into two integration problems:

Model Integration ensures that the domain models (UAV dynamics model in Simulink, communication channel model in OMNeT, etc.) can be integrated in a semantically consistent manner.

Simulation Integration ensures that execution of simulations can be synchronized by a distributed global clock (that can be kept in synchrony also with the wall clock), the events generated by the individual simulators can be used for event-driven interaction among objects controlled by different simulators, and data can be routed among the simulators under the time constraints required by the progress of the distributed global clock.

Next, we explore these challenges in further detail.

2.3 Model integration challenge

There exist several ways to approach the challenge of integrating domain models in a semantically consistent way.

Modeling Language Embedding. Embedding requires an injective structure preserving mapping between one or more DSMLs and a host language. Accordingly, the semantic domain of the host language must be rich enough to provide a common semantic domain for all DSMLs. While from the point of view of time semantics, the continuous time (CT) semantics of Simulink can embed the discrete event (DE) semantics of CPN, DEVS and OMNeT, mapping many of their domain-specific languages constructs onto Simulink constructs would be impractical. Some variant of model embedding is facilitated by the external blocks allowed in some modeling languages (e.g. Modelicas external function calls, and FMU import, Simulinks S-function interface) but they require suppressing many details of an external model, transform it into a simple construct in the host language, therefore they cannot be regarded as an example of model embedding in the usual sense.

Formal Modeling Language Composition. There have been formally established methods for the precise composition of DSMLs in an algebraic/logic framework [15]. The approach introduces a range of composition operators (*includes*, *restricts*, *extends*, *as*, *pseudo-product "*"*, *pseudo-coproduct "+"*) with appropriate semantics. These composition operators have been introduced in the FORMULA tool [1] developed by Ethan Jackson at Microsoft Research. While precise and tool-supported, practical full formal treatment in this case study was restricted by the absence of a formal, FORMULA-based specification of the semantics of the constituent modeling languages.

Model Integration Language. In many (if not most) multi-modeling problems, physical or computational objects modeled in different modeling languages need to interact with each other. The interaction might have behavioral, structural, or conceptual meaning. If the semantics of the interaction is restricted only to some shared aspects of the semantics of the individual modeling languages, then the problem can be solved effectively by the specification of a model integration language [24] that includes the specification of a semantic interface for the individual modeling languages and the

specification of integration constructs that are not part of either of the integrated modeling languages but support the integration of the models across the semantic interfaces. In the C2WT case study, the purpose of the model integration is the coordination of timed behavior of objects in a 3D space using various forms of communication. The required interaction semantics is discrete event and the data semantics is based on a distributed (but partial) data model that need to be established for the scenario. Consequently, model integration requires the specification of a relatively narrow *semantic interface* for the individual DSMLs and a *Model Integration Language* that is built atop these semantic interfaces, extends them with integration-specific constructs and supports the rapid modeling of arbitrary model integration scenarios.

2.4 A Model Integration Language Solution for C2WT

The technical challenges for the C2WT modeling and simulation application offer a compelling case for using a model integration language. The individual modeling languages are rich and have complex semantics not defined formally (and in addition they evolve independently with new releases). The required interaction semantics across the models is relatively narrow and restricted to only some aspects of the rich semantics of modeling languages. However, these simplifying conditions only mean that the solution is feasible, but not that it is simple.

The primary difficulty in defining a model integration language for the C2WT application is that it requires runtime support for model and simulation integration on a distributed computing platform. Support for model integration means availability of services for data distribution (in real-time scenarios under time constraints), simulation integration means availability of services for distributed time management and interaction control. In fact, the development and use of these services represent the core technical challenges in this application. Fortunately, distributed simulation is important in many application domains, therefore well-developed standard frameworks are available, such as the High Level Architecture (HLA) [4] that was selected as simulation integration platform for C2WT. With this, our task was simplified to developing a C2WT model integration language and related tools. The High-Level Architecture is a standardized architecture for distributed computer simulation systems. Communications between different federates is managed via the Run-Time Infrastructure (RTI) layer – an implementation of the HLA standard.

The HLA standard focuses on three primary areas. The first is time coordination throughout the federation. The evolution of time is a key thread through each of the integrated simulators. Each simulation engine must slave its progression of time to that of the overall HLA clock. The HLA standard provides several methods to accomplish this. Second is coordination of inter-federate messages and shared data objects. The HLA standard provides a publish-and-subscribe mechanism for passing messages and object updates throughout the federation.

Third, the HLA standard provides for basic simulation execution control. A limited ability to start, pause, and stop the execution of a simulation is built directly into the HLA standard. The C2WT relies upon the services provided by the RTI during run-time. As HLA is an accepted standard, a number of commercial, academic, and alternate RTI implementations are available. Currently, we use the Portico RTI [2] which provides support for both C++ and Java clients and is compliant with version 1.3 of the HLA standard.

The integrated system is shown in Figure 3. The simulators are interfaced to the HLA RTI through the simulator federates. The federates use the HLA RTI API for time management, data distribution and execution control. Clearly, the federations capture all of the required code needed to implement the multi-model simulation. We explicitly model this information using the C2WT model integration language, and translate this integration model into federation code. Details, including the metamodels of the integration languages are described in [14]. We used the following strategy to design the C2WT Model Integration Language:

1. The constituent modeling languages remain unchanged,
2. The C2WT model integration language is used for describing the integration (i.e. the system-of-system) architecture,
3. The semantics of the model integration language is provided by the HLA services for time and data management,
4. The semantic interface is the simplest possible required for the integration tasks.

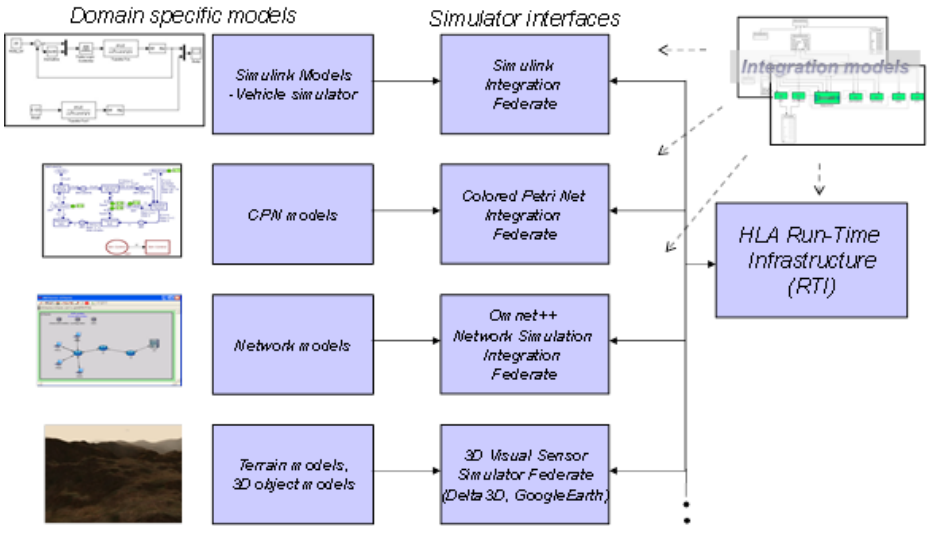

Fig. 3. Integration of DSMLs and Simulators using HLA

The semantic relationship between federates are defined in the model integration language using two main aspects: the data representation and the data flow. An integration model describes both data representation and data flow elements, and, in some cases, includes special elements as the placeholders for concepts specific to particular simulation engines. Data representation models consist of interaction and object models. Interactions are stateless, and can have parameters, while objects have states, which are represented as a set of attributes. Both interactions and objects are permitted to have inheritance hierarchies. These data representation models directly map to the HLA Federation Object Model (FOM). The federates (interfaces of simulators to HLA) are automatically generated from the integration models. The model integration language was specified by the usual means (informal (MetaGME) and formal (FORMULA) metamodels) and published in [14]. The full open source package is available on the OpenC2WT web site [3].

The set of integrated modeling languages and simulators is open, they do not influence HLA or each other. Currently the integrated modeling languages and engines include Simulink/Stateflow, CPN (Colored Petri Net), NS-2, OMNET, Delta3D, DEVSJava, Google Earth and Java-based custom federates (e.g. operator interface for user interaction in real-time simulation).

3 Smart Emergency Response System (SERS)

A MathWorks Summer Research Internship project [18] developed an automated emergency response system in order to dynamically manage emergency response personnel and equipment to handle emergencies on the roadways in the San Francisco area [20]. The system was then expanded into a Smart Emergency Response System (SERS) [26].

SERS coordinates the dispatch of flight systems (both rotorcraft and fixed-wing aircraft), ground supp:ort vehicles, and search and rescue dogs equipped with a harness to hold electronic devices. A smart device app enables emergency responders and survivors to share information in the field, learn about the current state of response operations, and request assistance. The information from the field as well as aid requests combined with available provisions (e.g., prescription medication, thermal blankets, defibrillators) and the configuration of the vehicle fleet serve as the input to a planning module that computes the time optimal mission. The deployment plan for each of the vehicles is then sent and executed autonomously by the respective vehicles (rotorcraft, fixed wing aircraft, and ground vehicles).

A sample scenario is as follows. An emergency occurs (e.g., a multiple vehicle accident) on a San Francisco roadway, and one or more persons at the scene who have a SERS app that lists all the services available can request provisions (e.g., defibrillator, medicine, oxygen masks, splints, etc.). Once the Command and Control (CC) receives the request(s), an analysis is made on the number, type, and location of requests to determine the best places (i.e., depots) to deploy their ground service vehicles. For each cluster of requests associated with a depot, a

number of rotorcraft will also be deployed to pick up provisions from the depot and drop off the respective provisions.

Model-Based Design (MBD) was used to develop the system, where several different types of models were integrated to deliver the overall system functionality. These models were provided by different stakeholders, involved different types of modeling languages, and supported numerous types of capabilities, such as simulation of real-world scenarios, visualization of vehicles in 2D space, virtual reality interaction with the flight systems atop Google Map of the area, etc., including support for incorporating the physical devices.

3.1 SERS as a Cyber-Physical System

SERS exemplifies the emerging paradigm of Cyber-Physical Systems as ensembles of collaborating embedded software systems [21]. The design of such systems challenges existing approaches for embedded systems such as the V design approach that is common in the aerospace and automotive industry. In the V approach, the system requirements first drive design by a top-down decomposition into subsystems and components, which is then followed by a bottom-up integration into a top-level system. Ultimately, the system integrator is responsible for the end product.

Increasingly, systems are being developed that follow a less rigid design process. These systems comprise systems in their own right and come into existence at runtime. Design and operation of such systems of systems is a challenging endeavor, the success of which is predicated on the use of models across systems, system perspectives, design stages, operational phases, and organizations. SERS serves as an example to highlight some of the key challenges and issues in successfully bringing systems of collaborating systems online.

Figure 4 shows the architecture of SERS [19]. At the core of SERS is Mission Command & Control. As shown along the top of the figure, interaction of emergency personnel with Mission Command & Control takes place via mobile apps, a mission user interface, video stream display, and virtual reality visualization. Along the bottom, devices are shown that communicate with Mission Command & Control and that operate in the physical world. These devices include ground vehicles to set up depots, delivery rotorcraft to deliver provisions in response to aid requests, fixed wing aircraft to provide situational awareness, sensory rotorcraft for use by emergency responders, network rotorcraft to setup an ad hoc network infrastructure [25], a robot arm to provide teleoperated manipulation, a humanoid robot to operate in dangerous environments, and search and rescue dogs with sensory equipment to find survivors and help with damage assessment.

As a reflection of the typical situation for collaborating systems, a broad range of organizations is involved in the design of SERS. These organizations provide expertise in the domains of operations, control, image processing, search and rescue dogs, robotics, communications, networks, and virtual environments.

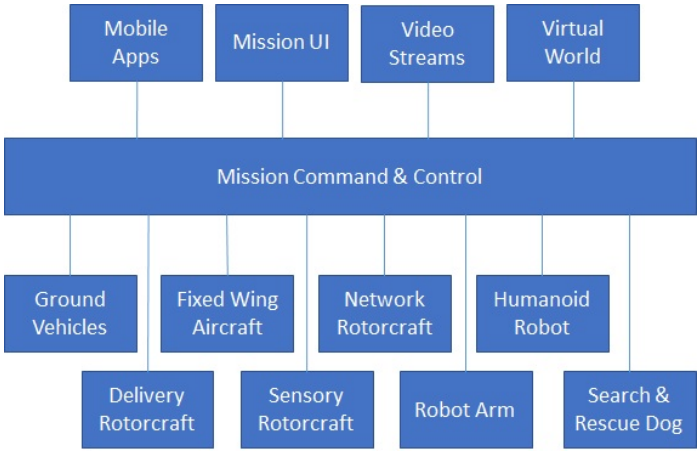

Fig. 4. Architecture of the Smart Emergency Response System

3.2 SERS Design

Figure 5 depicts the SERS elements and their relationships. The overall system synchronizes on time and location. For ground-based entities, location is further restricted to the midpoint of the roads, shown in the lower-left corner of Figure 5. These midpoints can be retrieved from curated *shape* files from government sites such as the City and County of San Francisco[9].

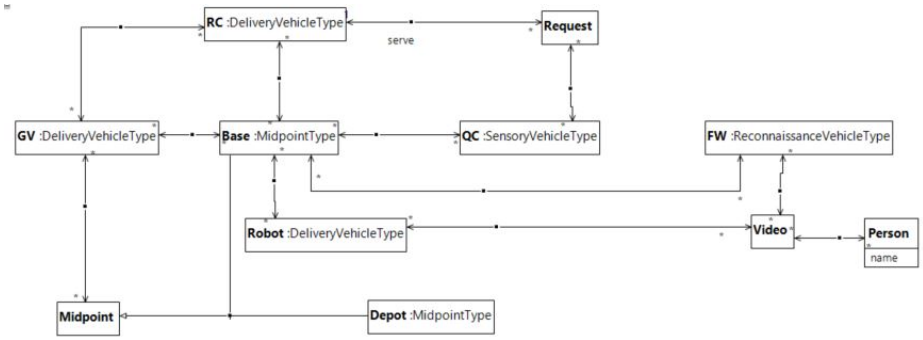

Fig. 5. The elements of SERS

The SERS *Base* station hosts Command & Control and is situated roadside, and is thus attached to a midpoint. Similarly, there are a number of locations (e.g., parks, schools, parking lots, etc.) where the ground vehicles may set up a

[9] https://data.sfgov.org/Geographic-Locations-and-Boundaries/Streets-of-San-Francisco-Zipped-Shapefile-Format-/wbm8-ratb

Depot, so these are also attached to midpoints. Clearly the *ground vehicles* (GV) must be attached to midpoints while they are originally stationed at the base. Each of the ground vehicles is capable of carrying *rotorcraft* (RC) that have a payload carrying capability so that they can deliver provisions to service aid *Requests*. Note that the rotorcraft may also remain stationed at the base. The list of possible provisions to deliver includes a *sensory rotorcraft* (QC) that can stream sensory information such as video from a high definition camera. This video may be streamed back to the Command & Control Center or to a mobile device, for example, operated by a first responder. As the mission unfolds, *fixed-wing aircraft* (FW) embark on *reconnaissance* sorties, for example, to determine the health of the infrastructure. To this end, video may be streamed back to the Command & Control Center as well as to a first responder. Finally, humanoid *Robots* that also support video streaming may be deployed.

In the overall design of SERS, the various elements are represented in many forms using many formalisms. In addition, abstract functionality such as optimizations must be represented. An overview of the range of representations is provided in Figure 6, where the formalisms used to represent the different elements are attached to these elements with a thick dotted line and shown in gray. The autonomous vehicles are at the center of the overall system and have various representations. In the form of a Computer Aided Design (CAD) model, the structure of the vehicles may be captured. This structure may be stored in a format such as the XML-based *collaborative design activity* (COLLADA) format or a *unified robot description format* (URDF).

From the structural model, a dynamics model of the physics can be automatically generated. Such a model may be represented by a block diagram with continuous time, differential equation, semantics (e.g., a Simulink model). Alternatively, a domain-specific representation such as SimMechanics multibody model may be automatically generated. In addition, a model of the control contributes to the dynamics. The low-level control model may be represented by a block diagram with continuous time or discrete time semantics. There may be different forms of low-level control such as for nominal operation and for system identification purposes. In addition, other forms of control such as supervisory control may be included, which is more appropriately represented by a discrete state formalism such as statecharts.

The models of the different vehicles are identified and calibrated against measurements derived from experiments. The corresponding data is represented in spreadsheets and multi-dimensional tables. Moreover, analysis and validation relies heavily on representation of data as graphs for convenient interpretation and documentation.

For optimization purposes, it is essential to characterize the various vehicles in terms of their longevity, payload, and wind speed. For example, a ground vehicle can set up a depot and operate for days. Rotorcraft, on the other hand, may only be able to fly for 15 minutes depending on the weight of the payload. Such characterization can be derived from the dynamics models where the data may be captured in the form of a spreadsheet.

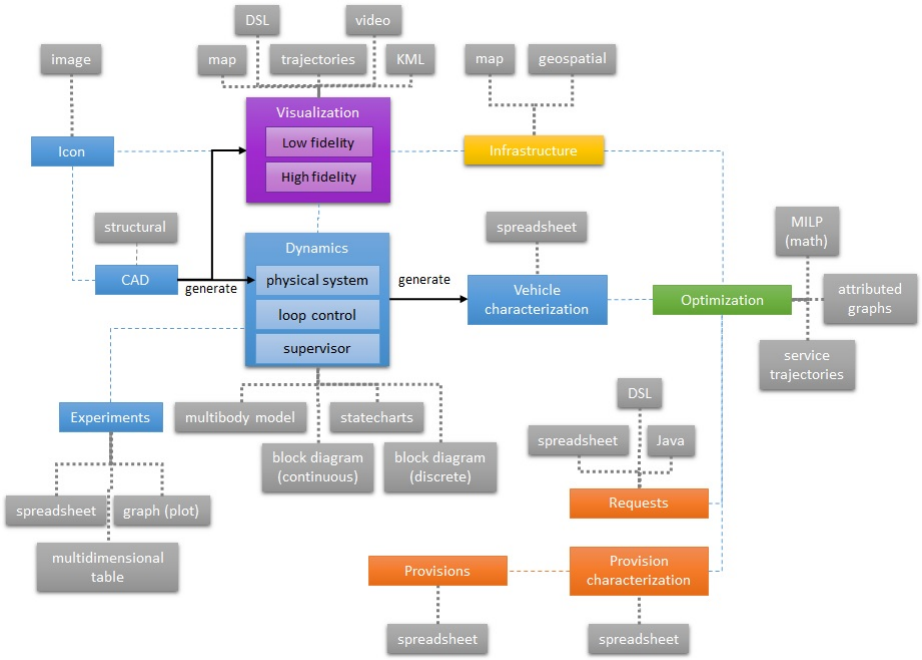

Fig. 6. The various SERS elements and formalisms (gray) used to represent them

Optimization input includes information about the infrastructure in terms of a combination of geospatial (road midpoints) and map information. These two pieces of data provide information as to how ground vehicles can be routed but also where there are potential locations to set up depots.

The remaining input to the optimization relates to the aid requests. First, there is the set of aid requests that come in from either the Command & Control Center or from mobile device apps in the field. To submit the requests, a Java data structure stores request information directly in the mobile device app or a spreadsheet representation may be used to share an underlying mapping. The Command & Control Center, in turn, may rely on a domain-specific language (DSL) to represent the various different requested provisions. Second, for the optimization process, it is necessary to characterize the provisions that can be requested (e.g., how much they weigh), which is information that is represented by a spreadsheet. This characterization is performed for the entire set of provisions, both of which are represented in a spreadsheet.

The optimization itself is then formulated in a mathematical representation. At the foundation are data structures that represent attributed graphs. Specific problems are then solved by using a mixed-integer linear programming (MILP) representation. The result of the optimization is a set of trajectories with service (delivery and pick-up) information attached. Each trajectory is an array of

waypoints that comprise geospatial data consisting of longitude, latitude, and altitude information as well as dwell time once a waypoint is reached.

A critical aspect of the overall system is visualization to help in design, analysis, test, training, and operation. The Command & Control Center has access to two different types of visualization. In a low fidelity representation, icons that represent the various vehicles are superimposed as images onto a representation of the region as a map. This visualization also shows the different types of aid requests on the map as they come in and at their requested location. The mission trajectories for each of the vehicles as computed by the optimization are also shown in the form of line segments between the waypoints that make up the trajectories. Alternatively, a high-fidelity visualization is available where the vehicles are shown performing their missions as realistic objects (based on the CAD models) in Google Earth as a virtual world. Motion of the objects is shown based on position and orientation information obtained from real-time simulations or physical measurements. Finally, live video data can be streamed to be displayed in the Command & Control Center.

3.3 A Smart Intersection

In a separate project to developing SERS, city planners may wish to install smart intersections that build on vehicle-to-vehicle and vehicle-to-infrastructure technology. A smart intersection system may consist of the elements shown in Figure 7. The smart intersection system may rely on a *Central Control* unit to coordinate use of the shared intersection surface area. The Central Control unit may have access to the *Streets* that intersect along with the *Traffic* in each of the streets. In addition, the Central Control unit may have access to a *Communication Unit* that allows vehicle-to-infrastructure communication via *communication units* of the *Lights* at the intersection and the *Vehicles* that wish to cross. Both the lights and the vehicle are equipped with a *Control Unit* to implement the functionality necessary to operate as part of a smart intersection. Finally, the central control unit has a network interface to communicate beyond the smart intersection, for example, with city infrastructure monitoring facilities.

3.4 Formalism Integration

At some point in time, the smart intersection infrastructure may wish to be integrated with the SERS. To design an integrated system, a domain-specific formalism would need to account both for the SERS specific elements in Figure 5 as well as the pertinent smart intersection elements in Figure 7. These smart intersection elements may include information about the traffic in the streets, and, therefore, only a subset of the smart intersection elements must be integrated with the SERS elements.

SERS involves the integration of at least seven different languages, many of which are domain-specific. Collectively, these languages involve different types of data, different granularity of data, continuous and discrete information, and different levels of abstraction. A wall clock is used to synchronize the activities

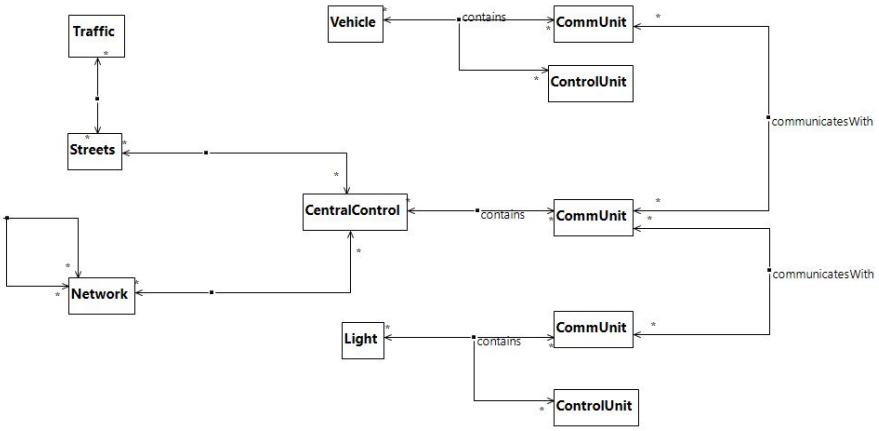

Fig. 7. The elements of a smart intersection

provided by the different components of the system – CAD diagrams, Block Diagrams (Simulink), Geospatial models, Mixed-Integer Linear Programming (MILP) models, Android app language, Shape models (map data in terms of longitude and latitude) and Excel spreadsheets.

4 Research Challenges

The development and analysis of DSMLs has been a research topic for some time. Thus far, the main focus has been on languages for supporting software construction in technical domains. There are convincing reasons why the relevance of research on DSMLs will substantially increase in the future. To date, only a few organizations, even in technical domains, are using DSMLs. Many domains have not been addressed so far. At the same time, languages and especially DSMLs are at the core of the digital transformation that many societies and organizations are facing. Among other things, the digital transformation is characterized by an increasing amount of reality being represented in information systems and by the fact that an ever increasing amount of services is performed by computers instead of human actors.

Globalization implies the demand for efficiently exchanging information with systems around the world. As a consequence, there is a need to support the economic creation and maintenance of application systems as well as their integration. Application systems are linguistic artifacts, i.e., they are constructed and used through language. DSMLs promise to facilitate the representation of the domain that is targeted by an application substantially, since they do not require modelers to build domain-specific concepts from scratch. At the same time, they promote system integrity, because they, to a certain extent, prevent inappropriate models from being created. Finally, DSMLs enable the creation of

models that represent complex real world objects and corresponding application systems, thereby empowering users.

The concepts in DSMLs should correspond directly to concepts with which users are familiar. In addition, they should feature a concrete syntax that fosters the intuitive understanding of models. As a consequence, users are not only supported in gaining a better understanding of the systems – and the environment in general – in which they work, but are also enabled to change the respective models and, in turn, the systems they represent. DSMLs also support the efficient exchange of information by allowing the description of complex objects in a more specific way, i.e. with more semantics, thereby substantially reducing the effort and risk caused by the need to reconstruct semantics. In addition to the increased demand for DSMLs, there are various challenges that researchers needs to address to foster the application of DSMLs.

4.1 Software Engineering challenges related to the formal foundation of languages

While separation of concerns demands an ever growing number of specialized DSMLs, many use scenarios require the integration of DSMLs. The current state of research includes various promising approaches, however, more challenges remain. Therefore, there is a need for advanced abstractions that enable more sophisticated integration technologies that support the integration or composition of languages and related tools. To take advantage of models during the entire lifecycle of systems (i.e., to support the idea of "models at runtime") future research needs to focus on the integration of modelling tools and application systems. On the one hand, more research on the notorious problem of synchronizing models and code is required, and on the other hand, it requires more research that is aimed at overcoming the principle limitations of current programming languages

Since these languages usually include one classification level only, (meta) classes that are manipulated in respective modelling tools have to be represented as objects on level M0 that creates the need to generate code. Alternatively, programming languages that allow for many classification levels would enable a common representation of models and code. With respect to maintaining DSMLs or family of DSMLs, powerful abstraction concepts are essential. Languages that focus on static information already have support for various abstraction concepts such as classification or generalization/specialization. Unfortunately, this is not the case for process modelling languages. These languages present a specific research challenge because dynamic abstractions are an obstacle to monotonic extensions, i.e., they do not allow a straightforward enforcement of the substitutability constraint [17]. Existing approaches that focus on relaxed versions of the substitutability constraint (e.g. [23, 5]) are insufficient, because they depend on premises that often cannot be satisfied.

4.2 Challenges related to the (re-) construction of domain-specific concepts

The construction of DSMLs cannot be restricted to their formal properties. Instead, they are intended to provide domain-specific concepts, the (re-) construction of which is not trivial. At the same time, it is usually not an option to leave this part of DSML development to domain experts, because they lack the knowledge required to develop appropriate abstractions. Against this background, future research on DSMLs has to take into account the peculiarities of reconstructing domain-specific concepts.

Methods that support the development of DSMLs. Existing methods for requirements analysis and system development are only partially suited to support the development of DSMLs. With respect to analyzing requirements, there is the problem that prospective users often lack a sufficient understanding of the artifact to be developed. In other words, they do not know what to expect. Hence, respective methods need to support users in developing a clearer picture of a DSML and/or the functionality it may provide. With respect to designing a DSML, methods should provide guidance for supporting various challenging decisions (see below). Research on specific methods for DSML development has recently emerged (e.g. [10]).

Distinguishing between a language and its application. The technical terms found in a domain are candidates for being included in a respective DSML. However, not all of them are suited for that purpose. The decision of whether a concept should be part of a DSML or should rather be specified with that DSML can be a remarkable challenge. The following example concepts illustrate this challenge: “Document”, “Goal”, “Product”, “Risk”, “ERP System”, “Department”. It is conceivable that all of these concepts are part of a DSML or can be defined using a DSML. Even though this decision depends on the specifics of a given case, it also depends on general criteria that have not been investigated sufficiently. They relate to economic and epistemological concerns.

Economics of DSMLs. The design of a DSML presents a fundamental economic challenge. On the one hand, a DSML should promote modelling productivity. For this reason it should provide specific concepts that fit the particular requirements of a domain. In other words, the concepts should reflect a high degree of domain-specific semantics (in the sense of information content). On the other hand, a DSML should enable a wide range of reuse in order to promote economies of scale. For this purpose, it should be built from concepts that abstract from specific features of domains. Since both objectives are of pivotal relevance for the economics of DSML and, as a consequence, for their use in practice, this conflict creates a research challenge that should be addressed. Figure 8 illustrates the problem.

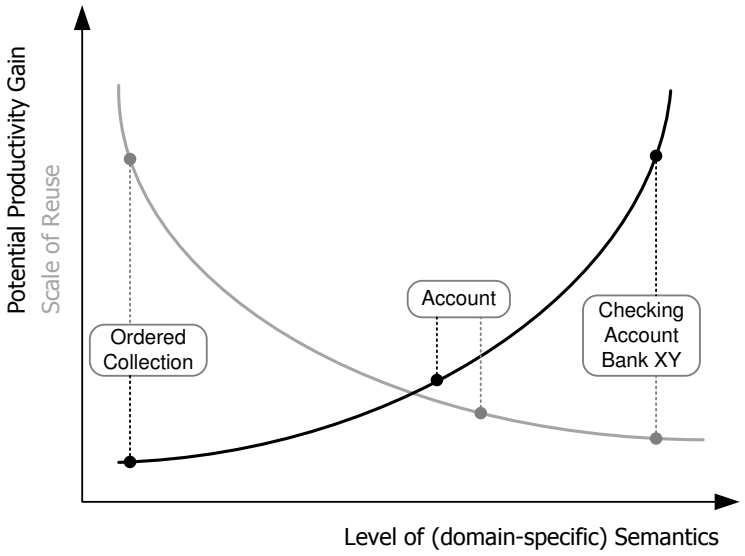

Fig. 8. Illustration of DSML design conflict

Note that the gradients of the curve will usually not be known precisely. The two example concepts show how the apparent conflict can be relaxed by including concepts on a higher level of abstraction (such as "Organizational Unit") that can be further refined to more specific needs (e.g. "University Department"). This situation reveals two research challenges. First there is the need to structure DSMLs into high level ("textbook") concepts and more specific ones that are tuned to narrower domains. Second, the semantics of the required refinement has to be specified. This is not a trivial task, since in many cases neither specialization nor inheritance alone is sufficient. Figure 9 illustrates this challenge. It represents concepts to model products. While the level of abstraction is apparently increasing from the bottom to the top, the relationships between the levels seem to combine characteristics of instantiation and specialization, which creates a serious problem, since there is a strict dichotomy between both refinement operations.

A further aspect of the above trade-off is related to standardization. With respect to the use of DSMLs, standardization is of pivotal relevance. It contributes to the protection of investments, fosters the dissemination of languages and, hence, economies of scale. Furthermore, decision makers appreciate standardization because it provides legitimization. However, standardization has a severe downside – it freezes a certain state and, because of the benefits it provides, creates a substantial obstacle to progress. Research on DSMLs can hardly ignore this conflict and therefore needs to aim at abstractions that are suited to develop standards that are open for evolution.

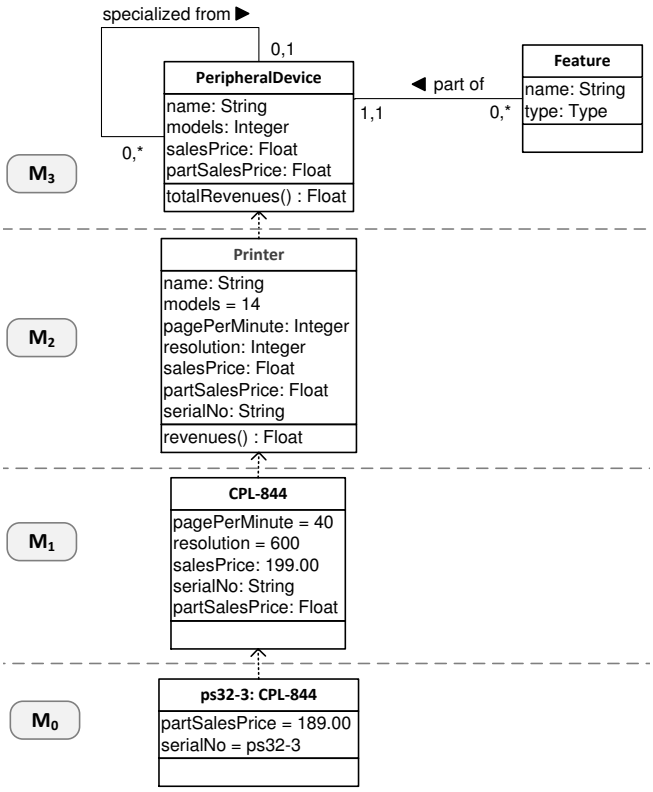

Fig. 9. Illustration of stepwise refinement of concepts over multiple classification levels

Top-down versus bottom-up development. To effectively address the conflicts outlined above it seems most promising to develop DSMLs in a top-down approach. That would allow similarities between different domains to be exploited. As a consequence, a top-down approach would be especially promising with respect to reuse and integration of languages. However, it is unrealistic to assume that existing languages can be easily replaced – people are accustomed to them and investments need to be protected. Therefore, research should focus on approaches that are suitable for combining aspects of top-down approaches with those of bottom-up approaches.

Epistemology of DSML design. Designing DSMLs for global use has to account for remarkable conceptual diversity. The technical languages used in certain related communities around the world not only vary with respect to designators, but also with respect to the semantics of concepts. From an epistemological perspective, there are two extreme interpretations of this conceptual diversity. On the one hand, one may assume that particular domains are actually different with respect to relevant objects, tasks, constraints etc. In that case, the diversity of technical languages would simply reflect ontological diversity and has to be accepted. On the other hand, one may regard conceptual diversity as the result of a cultural evolution that is characterized by chance and arbitrariness. Hence, actual technical languages are a contingent matter – they could be different and still serve their purpose. There is evidence for the latter assumption. The widespread use of ERP systems shows that organizations are able to adapt to a certain common conceptual foundation. As a consequence of this assumption, diversity could be reduced by proposing DSMLs that replace existing concepts with new ones in order to enable unified domains of discourse. While these considerations seem to be of a philosophical nature only, they are actually highly relevant for the design of DSMLs. They imply that it may not be sufficient to reconstruct the actual use of technical languages, but to ask whether they are appropriate and how conceptual diversity can be overcome.

Quality of DSMLs. If we follow Kant, who claims there is no recognition without concepts [16], and furthermore accept that certain concepts are more or less suited to structure a domain of interest with respect to a given purpose, then the construction of a DSML should not just aim at representing existing concepts, but eventually at reconstructing them with the intention to making them a better instrument for modeling. This corresponds to what Richard Rorty demanded for Philosophy – "Philosophers have long wanted to understand concepts, but the point is to change them so as to make them serve our purposes better." [22]. As a consequence, research on DSML needs to develop an elaborate, multi-perspective notion of language quality that not only comprises formal aspects, but also accounts for domain-specific concepts as well as for economic and cognitive aspects (for a respective approach see [10]).

Support for language evolution. The world is going to change. However, we lack powerful theories that do not allow for comprehensive predictions. As a consequence, it seems unavoidable that DSMLs have to be adapted from time to time. Therefore, future research needs to aim at supporting language evolution, including abstractions as well as respective tools that foster flexibility. Furthermore, the organization of the evolution is essential – should it be a managed process or would an agile approach be more suitable? What kind of incentives could be built to motivate stakeholders to contribute to language evolution and to migrate to new language versions.

Accounting for cultural challenges. The success of a DSML not only depends on its quality, it also depends on economic and political decisions that enable its use and dissemination. However, neither in organizations nor in the public is there sufficient awareness of the pivotal relevance of languages in general, and DSMLs in particular, for managing the digital transformation. As a consequence, funding for respective research projects remains relatively limited and organizations are reluctant to invest language technologies. If research on DSMLs is seriously interested in practical impact, it cannot ignore these obstacles. Instead, we need to put more emphasis on clarifying the tremendous economic and societal relevance of designing and disseminating DSMLs. This also includes the revision of university curricula.

Organization of research. The development of DSMLs and related tools requires substantial effort. That is even more so if one aims at languages and tools that should be disseminated in practice. Most research institutions, especially those in academia, do not have the resources to accomplish this objective on their own. This would require not only bundling of resources, but would also require the involved groups to agree on common concepts. Unfortunately, such an approach would contradict a core characteristic of scientific research – in order to promote progress, researchers are supposed to compete and challenge their peers. This conflict has to be taken into consideration.

4.3 Thoughts on possible future directions to pursue

The above considerations are intended to point at gaps in the state of the art that future research might address. Finally, we consider a few promising approaches for addressing selected challenges.

Multilevel language architectures. Traditional approaches to conceptual modeling are based on two classification levels: The M2 layer is used to define the metamodel of a modeling language and the M1 layer serves to represent corresponding models. However, such a language architecture is insufficient since it does not address the pivotal conflict of designing DSMLs – that is, it is not possible to specify a generic DSML that is subsequently refined into more specific

ones, thereby supporting both a wide range of reuse (on the more generic level) and high productivity of reuse (on more specific levels). Multilevel language architectures that enable an arbitrary number of classification levels [6] and support "deep instantiation"[7, 11] address this problem.

However, current programming languages that feature one classification level only prevent a tight integration of models and respective application systems at runtime. Elements of conceptual models, even though located conceptually at M1 or higher have to be represented within a tool at the M0 level. As a consequence, they cannot be further instantiated, which makes it mandatory to keep code and models in separate representations resulting in the notorious problem of synchronizing code and models. Programming languages that allow for multiple levels of classification, such as those that are based on the "golden braid" architecture [8] allow models to be represented at the intended level of classification and, as a consequence, allow for a common representation of models and code.

Establishing "open model" communities. The effort required to develop an elaborate DSML is not only beyond the capabilities of single research institutions, it also demands the involvement of stakeholders from practice, such as prospective users and tool vendors. At the same time, it seems reasonable to demand that DSMLs should be available to everybody in order to promote their dissemination and further development. Against this background, it seems promising to take the open source software initiative (which has been impressively successful in places) as a model for building similar communities that focus on the joint development and dissemination of DSMLs and respective models. Respective initiatives could also aim at fostering collaboration between various disciplines that are required to develop DSMLs. Furthermore, they can be seen as a new model for organizing research following the ideas of open science [13] and as a catalyst for disseminating research results to business practice. Early attempts to establish "open model" communities [9, 12] indicate that it is crucial to account for building effective incentives for participation, both for academics and practitioners.

Raising awareness. In various disciplines such as Philosophy, Psychology, and Sociology, the recognition that language is of pivotal relevance for almost all investigations resulted in "linguistic turns". As a consequence, the awareness for the foundational role of language has considerably grown in these disciplines. In Computer Science, language has always been a key foundation, especially formal languages and programming languages. However, with respect to the development of DSMLs, there is a need to increase the awareness of content, i.e. the relevance of reconstructing domain-specific concepts, which clearly requires to climb over the "firewall" that Dijkstra suggested. In addition to that, it will be important to start campaigns that aim at convincing funding agencies and politicians that artificial languages are not just a marginal instrument for pro-

moting automation, but lie at the core of the digital world and will thus play a crucial and essential role in our future lives.

References

1. `http://research.microsoft.com/en-us/projects/formula/`.
2. `www.porticoproject.org`.
3. `https://wiki.isis.vanderbilt.edu/OpenC2WT/index.php/Main_Page`.
4. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules. *IEEE Std. 1516-2000*, pages i –22, 2000.
5. Wil M. P. van der Aalst and Twan Basten. Inheritance of workflows: An approach to tackling problems related to change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
6. Colin Atkinson and Thomas Kühne. The essense of multilevel metamodeling. In Martin Gorgolla and Cris Kobryn, editors, *UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, volume 2185 of *Lecture Notes in Computer Science*, pages 19–33. Springer, Berlin and London, New York, 2001.
7. Colin Atkinson and Thomas Kühne. Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359, 2008.
8. Tony Clark, Paul Sammut, and James Willans. *Applied metamodelling: a foundation for language driven development*. Ceteva, 2 edition, 2008.
9. Robert France, Jim Bieman, and Betty H.C Cheng. Repository for model driven development (remodd). In Thomas Kühne, editor, *Models in Software Engineering*, volume 4364 of *Lecture Notes in Computer Science*, pages 311–317. Springer Berlin Heidelberg, 2007.
10. Ulrich Frank. Domain-specific modeling languages - requirements analysis and design guidelines. In Iris Reinhartz-Berger, Aron Sturm, Tony Clark, Yair Wand, Sholom Cohen, and Jorn Bettin, editors, *Domain Engineering: Product Lines, Conceptual Models, and Languages*, pages 133–157. Springer, 2013.
11. Ulrich Frank. Multilevel modeling: Toward a new paradigm of conceptual modeling and information systems design. *Business and Information Systems Engineering*, 6(6), 2014.
12. Ulrich Frank and Stefan Strecker. Open reference models – community-driven collaboration to promote development and dissemination of reference models. *Enterprise Modelling and Information Systems Architectures*, 2(2):32–41, 2007.
13. Andrés L. Guadamuz. Open science: Open source licences in scientific research. *North Carolina Journal of Law and Technology*, 7(2):321–366, 2006.
14. Graham Hemingway, Himanshu Neema, Harmon Nine, Janos Sztipanovits, and Gabor Karsai. Rapid synthesis of high-level architecture-based heterogeneous simulation: a model-based integration approach. *Simulation*, page 0037549711401950, 2011.
15. Ethan Jackson, Ryan Thibodeaux, Joseph Porter, and Janos Sztipanovits. Semantics of domain-specific modeling languages. *Model-Based Design for Embedded Systems*, page 437, 2010.
16. Immanuel Kant, Werner S. Pluhar, and Patricia Kitcher. *Critique of pure reason*. Hackett Pub. Co., Indianapolis and Ind, unified ed. edition, 1996.
17. Barbara H. Liskov and Jeannette M. Wing. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16:1811–1841, 1994.

18. Pieter J. Mosterman, David Escobar Sanabria, Enes Bilgin, Kun Zhang, and Justyna Zander. A heterogeneous fleet of vehicles for automated humanitarian missions. *Computing in Science & Engineering*, 12:90–95, August 2014.
19. Pieter J. Mosterman, David Escobar Sanabria, Enes Bilgin, Kun Zhang, and Justyna Zander. Automating humanitarian missions with a heterogeneous fleet of vehicles. *Annual Reviews in Control*, 38(2):259–270, 2014.
20. Pieter J Mosterman, David Escobar Sanabria, Enes Bilgin, Kun Zhang, and Justyna Zander. A heterogeneous fleet of vehicles for automated humanitarian missions. *Computing in Science & Engineering*, 16(3):90–95, 2014.
21. Pieter J. Mosterman and Justyna Zander. Cyber-physical systems challenges—a needs analysis for collaborating embedded software systems. *Software and System Modeling*, x(x):x–x, 2015.
22. Richard Rorty. Universality and truth. In Robert B. Brandom, editor, *Rorty and His Critics*, pages 1–30. Blackwell Publishing Ltd, Malden, MA and Oxford and Carlton, 2000.
23. Michael Schrefl and Markus Stumptner. Behavior-consistent specialization of object life cycles. *ACM Trans. Softw. Eng. Methodol.*, 11(1):92–148, 2002.
24. Gabor Simko, David Lindecker, Tihamer Levendovszky, Sandeep Neema, and Janos Sztipanovits. Specification of cyber-physical components with formal semantics–integration and composition. In *Model-Driven Engineering Languages and Systems*, pages 471–487. Springer, 2013.
25. Yan Wan, Shengli Fu, Justyna Zander, and Pieter J. Mosterman. Transforming on-demand emergency communication: Needs, analyses, and solutions. *Homeland Security Today*, 11(9):32–35, 2015.
26. Justyna Zander and Pieter J. Mosterman. Model-based design of a smart emergency response system. *Design News*, 2014.