A Virtual Environment for the Management and Development of Cyber-Physical Manufacturing Systems ⋆

DeJiu Chen [*] **Antonio Maffei** [**] **João Ferreirar** [**]
Hakan Akillioglu [**] **Mahmood R. Khabazzi** [*] **Xinhai Zhang** [*]

[*] *Mechatronics, Department of Machine Design, KTH Royal Institute of Technology, Sweden (e-mail: {chendj, khabbazi, xinhai}@kth.se).*
[**] *Technologies for Adaptable Production, Department of Production Engineering, KTH Royal Institute of Technology, Sweden (e-mail: {maffei, jpdsf, haaki}@kth.se)*

Abstract:
Modern machineries are often cyber-physical system-of-systems controlled by intelligent controllers for collaborative operations on the productions of complex products. To assure the efficiency and effectiveness, a consolidation of concerns across managerial levels, product lifecycle stages, and product lines or families becomes necessary. This calls for a common information infrastructure in terms of ontology, models, methods and tools. For industrial manufacturers subjected to increased cost pressure and market volatility, the availability of such an information infrastructure would promote their abilities of making optimized and proactive decisions and thereby their competitiveness and survivability. This paper presents a virtual environment that constitutes an information infrastructure for the management and development of evolvable production systems (EPS) in manufacturing. It adopts mature modeling frameworks through EAST-ADL for an effective model-based approach. The contribution is centered on a meta-model that offers a common data specification and semantic basis for information management across product lifecycle, models and tools, both for resource planning and for anomaly treatment. A prototype tool implementation of this virtual environment for validation is also presented.

Keywords: Evolvable Production Systems (EPS), Cyber-Physical Systems (CPS), Model-Based Development (MBD), Domain-Specific Modeling (DSM)

1. INTRODUCTION

Due to the increased cost pressure and market volatility, the ability of industrial manufacturers to support efficient, agile and robust processes is of vital importance for their competitiveness and survivability. As a consequence, the manufactures have to pay increasingly more attention to production planning, resource deployment, and operation control. Meanwhile, the integration of state-of-the-art information and communication technology, computer software and hardware into modern machineries are fostering many new opportunities in the areas of autonomy, robustness control, and situation-aware adaptations. For example, one scenario could be that a set of machines perceive and communicate their respective operational situations and thereby automatically plan and synchronize their actions taking into account the overall production goals and risks. As a consequence of such integration, modern machineries are evolving from being physical to being cyber-physical (Rajkumar et al. (2010), Wolf (2009)).

Form an engineering perspective, the success depends on the availability of appropriate methods and tools both for effective controller development and for process management. In particular, for advanced decision making, novel methods and tools are necessary to capture the following concerns: 1. the operational constraints in temporal and spatial domains, 2. Knowledge about the preferred optimizations of machine resources under different circumstances, and 3. the interdependencies of operational decisions with various business constraints such as production costs and time. Moreover, a systematic treatment of uncertainties of observed and planned behaviors, as well as various failure modes of the machines, is necessary for advanced proactive decisions, and robustness guarantee, error handling and fault treatment.

This paper presents a study on the development of a virtual environment that adopts well-known system description frameworks for an effective model-based approach to the management and development of manufacturing systems with advanced cyber-physical features. See Fig. 1 below. This model-based approach advocates the use of models, conforming to a common semantic meta-model, not only for system development, but also as a basis for the embedded knowledge inference as well as the higher level process management decisions. This allows effective communications and decisions across product lifecycles, models and tools.

⋆ This work is conducted within the EIT ICT-Labs Project 14386 (CPS for Smart Factories), with technical support by Volvo Trucks Technology AB/Sweden, and MetaCase/Finland.

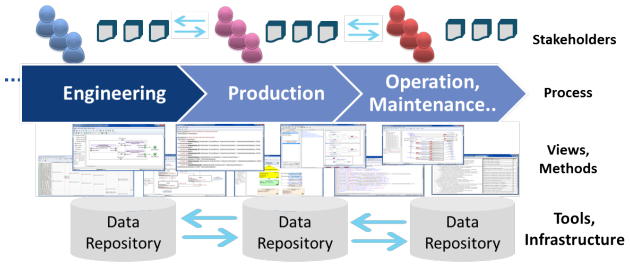

Fig. 1. Information exchange through social and technical means across the lifecycle stages of a system.

The content of this paper is organized into the following main sections:

- Section 2 introduces the key concepts of production systems being formalized as well as the EAST-ADL language being adopted as a fundamental framework for system modeling, information management, analysis and synthesis.
- Section 3 presents the meta-model underlying the design of a common modelling language for system description and information consolidation. By extending the EAST-ADL modeling framework, the approach exploits the support of several existing system description frameworks for the realization of an ontology for evolvable production systems (EPS).
- Section 4 presents a software tool implementation of the virtual environment, based on the DSM workbench MetaEdit+. Based on the meta-model, this tool provides the user-level support for an integrated specification of product&production features, abstract process, as well as the machine resources in terms of functional agents and physical equipments. We also introduce the related modeling support for requirements engineering, annotations of timing constraints, plausible errors/anomalies, as well as the approaches to analysis, design space exploration and optimization. This tool constitutes a prototype and a solid basis for further industrial validation.

2. SYSTEM CONCEPTS AND TECHNOLOGICAL PREDERENCES

2.1 System Concepts

An evolvable production systems (EPS) for product assembly consists of the following key aspects (See also Fig. 2)

- *Assembly Features* – relating to the parts of products to be assembled. For a product, the specification normally describes the number of parts and their corresponding geometrical and material characteristics as well as the preferred assembly methods. From a production point of view, each part together with its geometrical and material characteristics constitutes a *production feature* that can be of concern for the product composition.
- *Assembly Process* – relating to the plans of work *tasks* to be executed by the machine resources for the assembly of products. Each work task defines for one or multiple parts the required assembly operations or functions, such as moving and joining. In a process, the work task is also associated with constraints in regard to timing and synchronization, reliability and safety.
- *Assembly Equipment* – relating to the *machine resources* for the production. Normally, each machine resources offer some basic production skills, such as gripping and moving parts, or conducting negotiation with different machines for behavior coordination. Each resource has performance characteristics in regard to throughput and robustness.

A modern assembly line system is often implemented as a multi-agent system where a variety of interacting functional units, referred to as *agents*, collaborate dynamically to satisfy both local and global objectives (Maturana&Norrie (1996)). In order to achieve fully adaptability at the shop floor level, a novel design paradigm, referred to as *Evolvable Production Systems* (EPS) is first introduced in Onori (2002). One key notion is *mechatronic agent*, referring to a production unit (gripper, robot, etc.) that embeds a computer board on which agents are running to provide some production services in terms of skills (Onori et al. (2012)). The design depends the related process requirements (Onori&Barata (2009)). There are two types of skills; (1) Simple Skills; atomic services provided by the module which may or may not directly match a process, (2) Complex Skills; on the other hand abstracts and manages a composition of simple skills, necessary to the execution of higher level processes.

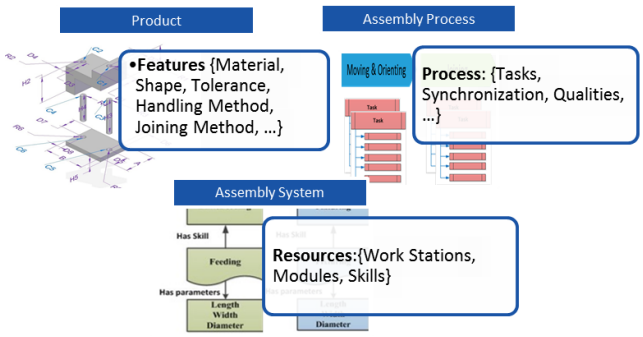

Fig. 2. Key aspects of an assembly system..

2.2 EAST-ADL as Base Technology to System Ontology

The EAST-ADL modeling framework (Electronics Architecture and Software Technology - Architecture Description Language), including a methodology and a modeling language, represents a key European initiative towards a standardized multi-viewed description of automotive electrical and electronics systems (EAST-ADL (2014)). It is a result of a series of consecutive projects: ITEA EAST-EEA, EU FP6 ATESST I and EU FP7 ATESST II, and EU FP7 MAENAD. We believe that EAST-ADL, by integrating many generic system description frameworks (e.g. SysML) and automotive specific methodological and technological considerations (e.g. RIF/ReqIF, ISO26262), provides a solid basis for enabling a model-based planning and management of vehicular assembly line systems. On the basis of EAST-ADL, language extensions and specializations will be developed to promote separation-of-concerns and thereby effective quality management of manufacturing systems.

The EAST-ADL language is highly modular, consisting of multiple packages (See Fig. 3). The core of system description is the *System Model* that specify the target system in multiple levels of abstraction. These levels are: *Vehicle Level*, *Analysis Level*, *Design Level*, and *Implementation Level*. The mappings across these abstraction levels are managed by *Realization* models. Of these system abstraction levels, the vehicle level model provides the topmost definition of a system by capturing the externally visible functionalities in terms of *features*. A vehicle level system model is refined by an analysis level description, which captures the underlying system I/O functions (e.g. for plant monitoring and actuation) and control functions (such as motion control algorithms) for each feature. For system realization, an analysis level model is further refined by taking the characteristics of available hardware resources into consideration. Such a refined design is captured by a design level system description, consisting of a *Functional Design Architecture* that specifies the grouping and partitioning of abstract functions, a *Hardware Design Architecture* that characterizes the target hardware platform, and an *allocation* matrix that captures the binding of the abstract functions to the platform. The implementation level model is a detailed specification of software components and the architecture configuration based on AUTOSAR (2013).

In system development, behavior modeling and analysis play a key role for assuring the correctness of decisions in regard to requirements engineering, architectural design and refinement, and the design of verification&validation tasks. As a generic modeling support, EAST-ADL also allows the users to explicitly capture and manage various behavioral concerns in terms of *behavior constraint* based on a *hybrid system model* (Chen et al. (2013a)). For example, a behavior constraint can be used to refine a textual requirement statement. Moreover, when targeting an error definition, a behavior constraint provides a formalization of the anomalies and error propagations. For the system functions, which have a synchronous execution semantics, external behavior models in off-the-shelf tools like SCADE, ASCET, Simulink, etc. can also be used for the specification. The language provides modeling support for specifying the related triggering and timing properties through

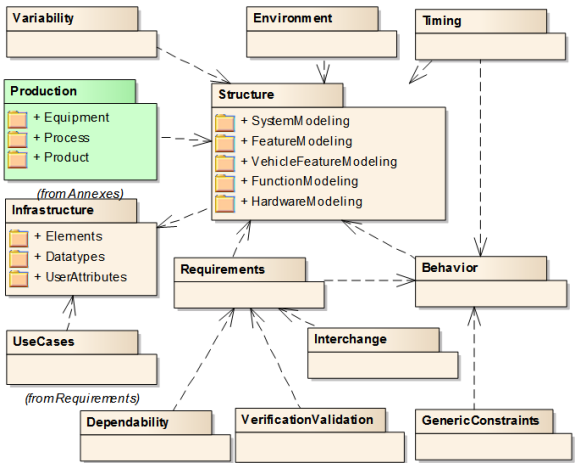

Fig. 3. The meta-model package *Production* and its dependency with other packages of EAST-ADL.

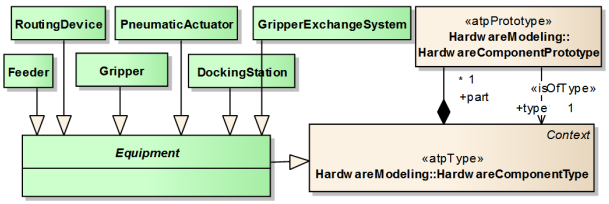

Fig. 4. The *Equipment* metaclass, extending the *HardwareModeling::HardwareComponentType*, for the description of machine resources.

its *Timing Model*. For requirement engineering, the EAST-ADL *Requirement Model* allows each requirement to be traced to the related design solutions, verification and validation cases, as well as to other interdependent requirements. EAST-ADL also provides comprehensive support for dependability engineering (Chen et al. (2011)) and safety analysis (Chen et al. (2013b)) through its *Dependability Model*.

3. EXTENSION OF EAST-ADL FOR PRODUCTION SYSTEMS

This extension of EAST-ADL meta-model (EAST-ADL (2014)) aims to support the specification of assembly line systems, by reusing the generic EAST-ADL concepts in regard to requirements engineering, architecture specification, and annotations of extra-functional constraints. This constitutes a basis for the stipulation of model libraries or patterns for further facilitating the system development and management. The approach emphasizes the formalisation of requirements and design considerations to enable knowledge-based system adaptations. This differs from other modeling frameworks or formalisms focusing on interoperability and operational control, such as PackML for packing line (PackML (2015)).

This meta-model extension is defined as an annex of EAST-ADL, referred to as *Production*. It has dependency with other EAST-ADL language packages as shown in Fig. 3. In particular, this new package extends with the existing EAST-ADL package, *Structure*, which supports the specification of system structure with multiple abstraction layers. The extension enables the specific structure specification for assembly line systems, while allows requirements, behaviors, timing, dependability and other constraints to be annotated to the structure entities as in the original EAST-ADL.

Within the *Production* package, there are three subpackages: *Equipment*, *Process*, and *Product*, for the specifications of assembly machine resources, assembly process, and assembly feature respectively.

The *Equipment* subpackage contains the meta-classes for the descriptions of machine resources. These meta-classes are shown in Fig. 4. The *Equipment* is an abstract class for relating a variety of specific machine resource types to the more generic notion of hardware components in EAST-ADL in terms of *HardwareComponentType*. In particular, through the specialization links, all properties of *HardwareComponentType* will be inherited by the machine resource types. For example, the machine resource types will have the same compositional structure as other hardware component types. In a composite, a part can be connected

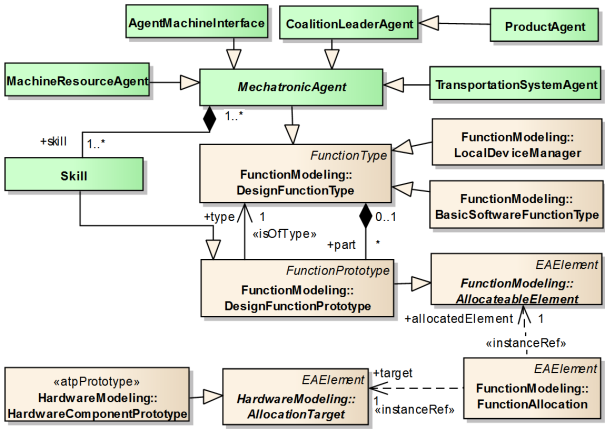

Fig. 5. The *Skill* metaclass, extending the *FunctionModeling::DesignFunctionPrototype*, for the description of machine services; The *MechatronicAgent* metaclass, extending the *FunctionModeling::DesignFunctionType*, for the description of control functions.

to other parts through the ports declaring its physical interfaces. The abstract *Equipment* is further specialized into *Feeder*, *Gripper*, *DockingStation*, etc.

In an assembly system, a variety of interacting functional units, referred to as *Mechatronics Agents*, collaboratively control the assembly process. See Fig. 5. Such agents are defined by extending the EAST-ADL metaclass *DesignFunctionType*, which represents the functionalities being allocated to particular hardware resources. For such functionalities, the specification is technical in the sense that sufficient platform specific details are taken into consideration for the mapping design, such as in regard to data size, hardware memory size, execution rate. Each agent offers their functionalities through its compositional parts, referred to as *Skills*. Defined as a specialization of *DesignFunctionPrototype* (representing an occurrence of the *DesignFunctionType* that types it in a composition as part), *Skills* are the constituent units of agents that provide the means for the agent execution and communication. For the same reason, each *Skill* is allocated to an Equipment, through an allocation relation (*FunctionModeling::FunctionAllocation*). For the allocation design, a *Skill* can be annotated with constraints in regard to states, behaviors, timing, synchronization, reliability, etc. As a modeling support for the domain concepts, the abstract *MechatronicAgent* is given by the following concrete functional units:

- *MachineResourceAgent* (MRA) – a metaclass for the functional units that are responsible for controlling machine resources. The skills of a machine resource agent have often a tight coupling with the underlying basic I/O software and hardware;
- *AgentMachineInterface* (AMI) – a metaclass for the functional units that provide the adaptation layer as well as basic application interface services for the interactions between functional units and their underlying machine specific software and hardware platforms;
- *CoalitionLeaderAgent* (CLA) – a metaclass for the functional units that coordinate the configuration and execution of skills. Such a functional unit can be used to monitor the status of involved machine resources and then adapt the modes or configurations for the purposes of quality-of-services or fault tolerance.
- *TransportationSystemAgent* (TSA) – a metaclass for the functional units that are responsible for controlling the transportation systems among machine resources. Such a functional unit provides localization, transport and positioning functionalities.
- *Product Agents* (PA) – a specialized for the functional units that are responsible for the process planning or adaption in regard to the emerging production requirements. Such functional units are specializations of coalition leader agent for strategic level decisions.

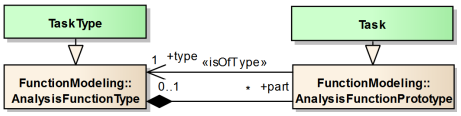

Fig. 6. The *Task* metaclass, extending the *FunctionModeling::AnalysisFunctionPrototype*, for the description of abstract work tasks; The *TaskType* metaclass, extending the *FunctionModeling::AnalysisFunctionType*, for the description of task types.

The specification of assembly process defines the work to be performed. Here, we extend the EAST-ADL constructs *AnalysisFunctionType* and *AnalysisFunctionPrototype* for the description of abstract, implementation independent, functions for such a specification. For an assembly system, each Task defines a piece of work to be done, or an activity to be performed, with the type definition given by its associated *TaskType*. See Fig. 6. The *Task* and *TaskType* inherit the properties of *AnalysisFunctionPrototype* and *AnalysisFunctionType* respectively. For example, a task may interact with other tasks through the function ports (*FunctionPort*) and connectors (*FunctionConnector*). Extra-functional constraints on behavior, timing and dependability can also be annotated to the tasks.

For an assembly system, the specification of product features is related to the characteristics of the products or product families that are of particular concern for the manufacturing. For example, such characteristics can be the compositional parts, their geometrical and material characteristics that affect the preferred assembly methods and process. To support the description of product features, a new metaclass, referred to as *ProductFeature*, is introduced. See Fig. 7. This metaclass extends the EAST-ADL construct *VehicleFeature* for the functional and extra-functional characteristics of automotive vehicle product line. Multiple product features (*ProductFeature*)

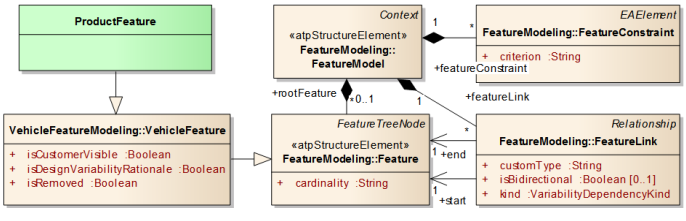

Fig. 7. The *ProductFeature* metaclass, extending the *VehicleFeatureModeling::VehicleFeature*, for the description of products being assembled.

and production system features (*Feature*), when combined through feature links (*FeatureLink*), form a feature model (*FeatureModel*). This model can be used to describe the variability and commonality within a product family, e.g. "product feature S requires assembly feature E" or "product feature S excludes product feature M". EAST-ADL currently provides a list of link types by the metaclass *VariabilityDependencyKind*, which can be further extended through the custom attribute.

The extensions of existing EAST-ADL constructs for assembly line allow the existing EAST-ADL support for the traceability of requirements and design refinements to be reused. For examples, the dependencies from a set of assembly line functions and resources to a particular requirement can be captured to the satisfy relation (*Requirements::Satisfy*). In regard to design refinements, the *Realization* relation (*Elements::Realization*), which connects some elements across boundaries of abstraction levels, is applied. For assembly lines, such design refinements exist as the mappings from abstract product features (*Product::ProductFeature*) to process tasks (*Process::Task*), as well as from process tasks (*Process::Task*) to the skills (*Equipment::Skill*) of the machine resources. See Fig. 8.
.

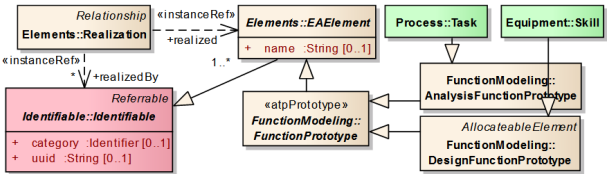

Fig. 8. The metaclass *Elements::Realization* captures the design refinements across abstraction levels given by two roles: *realizedBy* and *realized*. For production systems, it used to identify a task (*Process::Task*) that serves as a specification and the corresponding skills (*Equipment::Skill*) that realize this task specification.

4. THE TOOL IMPLEMENTATION

With the meta-model concepts introduced above, a prototype virtual environment for effective management of assembly line systems has been developed. In this section, we present the user level models and tool features. The target system (Fig. 9), built on a full industrial specification, is an assembly line for complex components in machines such as automotive vehicles. The tool implementation is done with MetaEdit+, a workbench tool for *DSM* (Domain Specific Modeling). Fig. 10 presents a top level package view of the system specification, including a requirements model, a system model, two realization models, a timing model, and an error model. The system model (*KTHSensair_AssemblyLineSystem*, Fig. 11), describes the machine

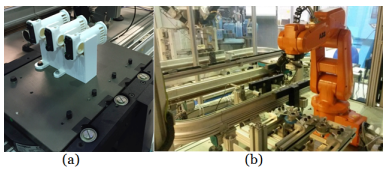

Fig. 9. (a) The components being assembled; (b) The machine resources of the assembly system.

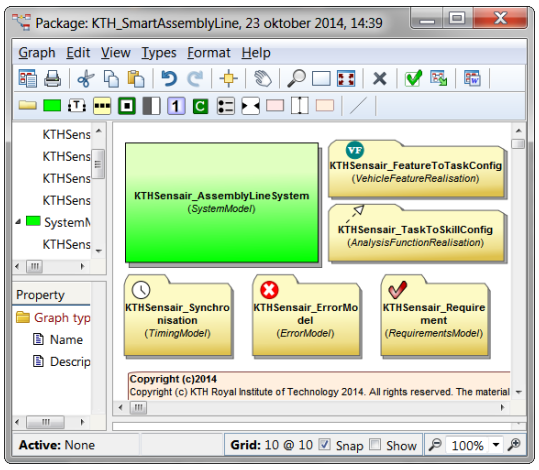

Fig. 10. A top level package view of the models currently given for the target assembly system specification.

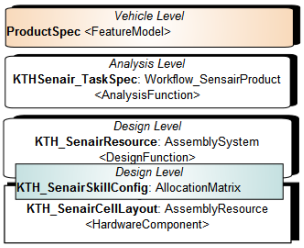

Fig. 11. The system model specifying the target assembly system in multiple levels of abstraction.

resources, assembly process, and assembly feature, following the EAST-ADL hierarchy for design refinement. With MetaEdit+, each model has a graphical representation as the concrete syntax and a corresponding textual representation as the abstract syntax (e.g. in XML).

The available machine resource is specified by the hardware model *KTH_SenairCellLayout*, which is located at the bottom layer. The content is shown in Fig. 12. In the hardware model, the components ports and connections denote the physical interfaces. These machine resources are controlled by a variety of mechatronics agents, specified by the design level function model *AssemblySystem* (Fig. 13). Here, the target system consists of four independent coalition leader agents that coordinate directly the execution of skills given by the machine resource agents *ABBRobot* and *Grippers* through their data-flow ports and connections.

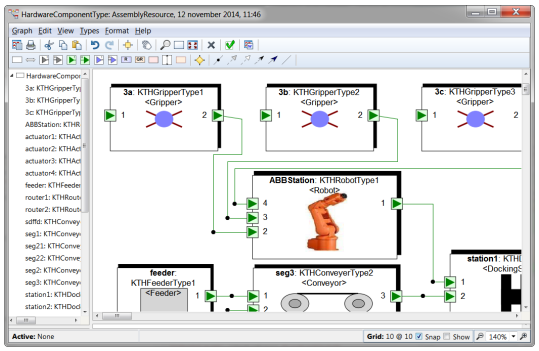

Fig. 12. An excerpt of the design level model *AssemblyResource* for hardware resource description.

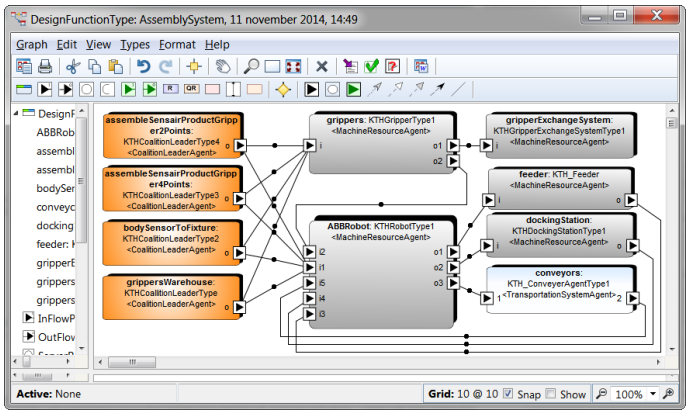

Fig. 13. The design level model *AssemblySystem* that provides a functional view of the mechatronics agents.

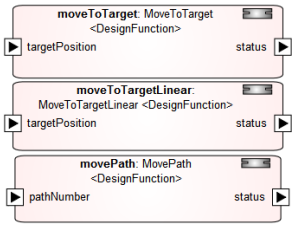

Fig. 14. The specification of *Skills* within the machine resource agent *KTHRobotType1*.

Depending on the status of execution, these two machine resource agents further regulate the executions of some subordinate machine resource agents, including *gripperExchangeSystem*, *feeder*, *dockingStation*, and *conveyors*. Each agent also declares their contained functionalities through the composition of some skills, as shown in Fig. 14.

With a given hardware architecture and a given functional specification of the mechatronics agents, the allocation matrix (*AllocationMatrix*) specifies the mapping from the skills to machine resources. Fig. 15 shows the specification of allocations for some of the skills with a matrix view. Coming back to the skills shown in Fig. 14, it can be noticed that the robot *ABBStation* (with the column name checked in the matrix) hosts now among others three skills: *movePath*, *moveToTarget*, *MoveToTargetLinear*. The tool currently supports the visualization of the mapping design directly in the hardware architecture model.

At the analysis level, a functional specification (Fig. 16) provides information about the abstract assembly tasks to be performed. Each Task defines a piece of work to be done, such as *MoveConveyer3*, *LockCarrier*, and *Attach-*

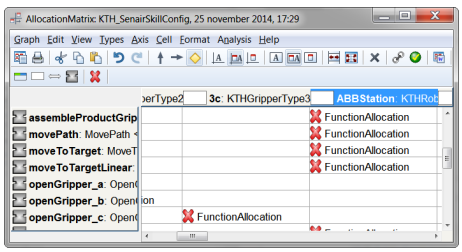

Fig. 15. An excerpt of the design level model *AllocationMatrix* specifying the mapping of skills to equipments.

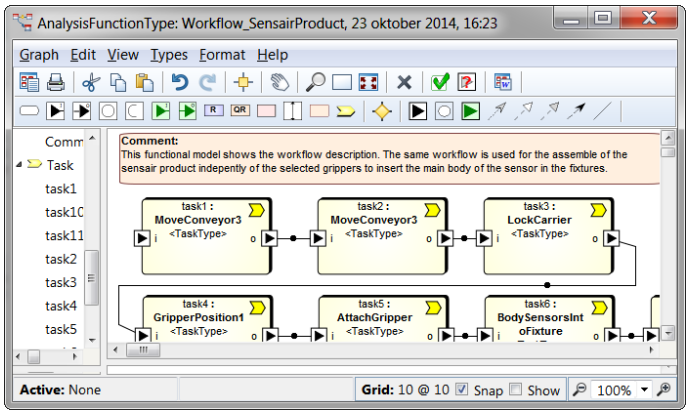

Fig. 16. An excerpt of the analysis level model *Workflow_SnesairProduct* for abstract process definition.

Gripper. Fundamentally, each elementary task follows the same *run-to-completion* semantics as for the EAST-ADL function blocks being extended (Chen et al. (2013a)).

The topmost level system description is given by a vehicle level feature model (Fig. 11) where the features represent the characteristics of a product as well as their production implications. Shown in Fig. 17, the product being manufactured by the target system consists of three components: a main body and two smaller parts that were attached to the sides. This structural property is given by the composition links that go from the product feature *AssemblyProduct_A* to the product features topLock, mainBody, and bottomLock. The topLock feature also owns a sub-feature given either by *TL_1* or by *TL_2*. These two features are linked by a MandatoryAlternative link, indicating an exclusive disjunction relation (i.e. *TL_1* XOR *TL_2*). These product features imply certain production features of the assembly system. For example, the topLock and bottomLock are two smaller components without sufficiently large surfaces on their sides for a mechanical gripper. Instead, a vacuum gripper has to be used to lift them. The mainBody can however be gripped by a mechanical gripper or a vacuum gripper. In the model shown in Fig. 17, such concerns are captured by

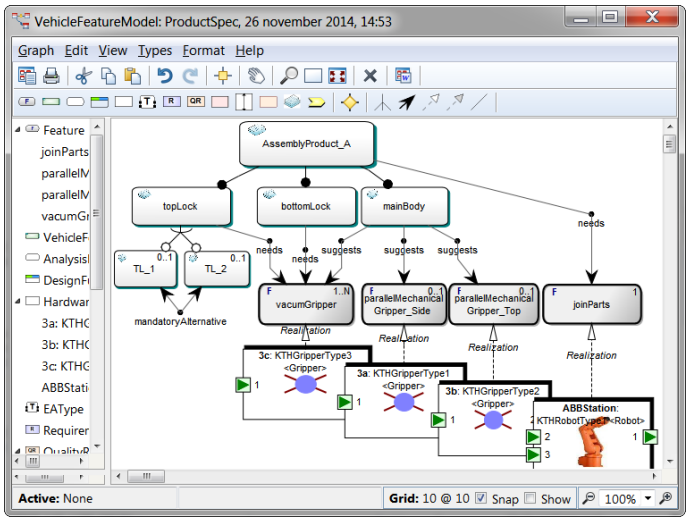

Fig. 17. The vehicle level feature model *ProductSpec* that specifies the features and their dependencies.

some EAST-ADL feature links (*needs* and *suggests* from product features to production features). In particular, the link needs indicates: if the start feature S is selected, then also its end feature E must be selected: not (S and not E). A weak form of needs link is the suggests link. Such product features can imply further technical constraints. For example, the production feature *parallelMechanicalGripper_Top* can only be realized by the gripper type 2 (*KTHGripperType2*), where mechanical grippers with fingers made in 3D-printed plastic support the pickup of the main body from tops. The production feature *parallelMechanicalGripperSide* can only be realized ty the gripper type 1 (*KTHGripperType1*), where mechanical grippers with fingers made in 3D-printed plastic provide support for picking up the main body from sides.

For the target system, two models are introduced for the realization mapping that goes from the assembly features, to the abstract assembly tasks, and then to the skills of available machine resources. A package view of these two models, referred to as *KTHSensair_FeatureToTaskConfig* and *KTHSensair_TaskToSkillConfig*, is given in Fig. 10. The model *KTHSensair_FeatureToTaskConfig* (Fig. 18) specifies the refinements from assembly features to the assembly tasks. These assembly tasks serve as functional means for the feature realization. For example, the task6 (with the column name checked in the matrix) realizes the production feature *joinParts*. A further refinement of the design is given by the mapping from abstract assembly tasks to the skills of machine resources. For the target system, this is specified by the model *KTHSensair_TaskToSkillConfig* (Fig. 19). For example, it is shown that the *task6* (with the row name checked in the matrix) is mapped the following skills of robot *ABBStation* for its realization: *bodySensorsIntoFixture*, *movePath*, *moveToTarget*, and *MoveToTargetLinear*.

EAST-ADL provides comprehensive modeling support for requirements engineering, including requirements description, verification&validation, refinements and traceability management. For production systems, the existing EAST-

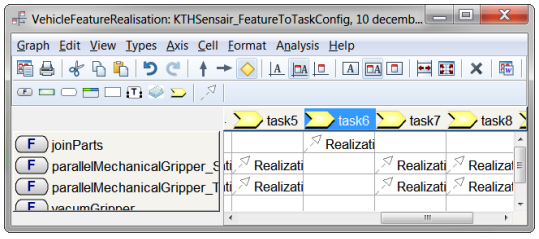

Fig. 18. An excerpt of realization model *KTHSensair_FeatureToTaskConfig* specifying the mapping from features to abstract tasks.

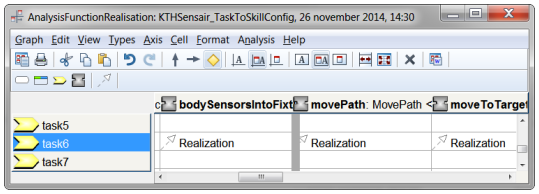

Fig. 19. An excerpt of realization model *KTHSensair_TaskToSkillConfig* specifying the mapping from abstract tasks to machine skills.

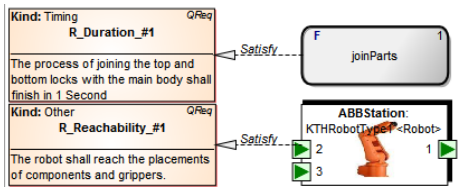

Fig. 20. An excerpt of the requirements model *KTHSensair_Requirement*.

ADL support is used. For example, the content of *KTHSensair_Requirement* is shown in Fig. 20. It captures two requirements, *R_Duration_#1* and *R_Reachability_#1*. With two *Satisfy* links, the model also defines the assembly feature *joinParts* and the machine resource *ABBStation* satisfying these requirements.

An EAST-ADL timing model is an explicit specification of the execution of active functional objects (see e.g. (Peraldi-Frati et al. (2012)). For the target production system, the model *KTHSensair_Synchronisation* contains the timing constraints as shown in Fig. 21. Through the "-||-" operators and execution events, these constraints specify the triggering of multiple agents (i.e. *Tolerance#1*), the delay of cross-agent communication (i.e. *Tolerance#2* and *Tolerance#3*), and the response time of an agent (i.e. *Tolerance#4*) on a time line.

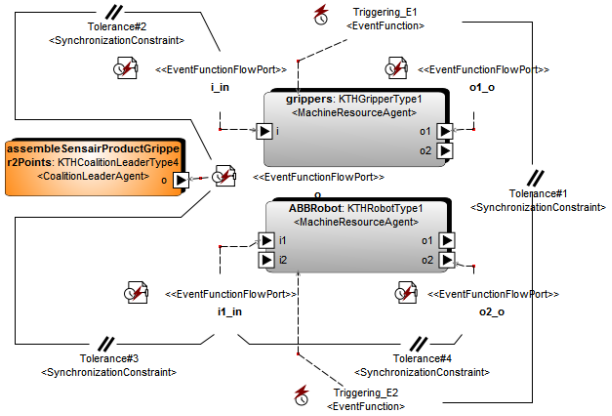

Fig. 21. An excerpt of the timing model *KTHSensair_Synchronisation*, where each "-||-" operators contains information about the preferred timing tolerance (e.g. 0 *msec* or 1 *msec*).

To support safety analysis, an EAST-ADL error model contains the annotations of estimated failure modes of system units as well as their transitions to/from other states. The tool provides model transformation support so that an initial error model can be produced automatically from a nominal architecture model defining the topology of error propagation according to the communication and hardware bindings. For a system unit, the error logic is given either by boolean logic expressions or by state-machine models. In Fig. 22, the error logic of the resource agent *ABBStation* is shown. The transition events are given by the faults and failures. With such a definition of error logic, automated safety analysis is supported through model transformation to external analyzers. For FTA&FMEA analysis, the transformation to the *HiP-HOPS* tool, described in Papadopoulos&McDermid (1999) and Mahmud et al. (2010), is currently supported. One analysis result

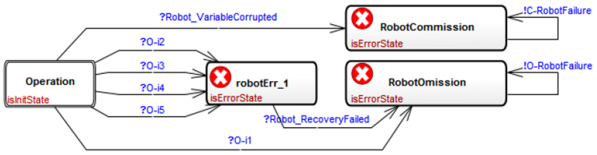

Fig. 22. An excerpt of the error model *KTHSensair_ErrorModel*. In the state-machine based description, a write event (!) is a failure occurring when the transition is fired. A read event (?) is a fault that must be available to fire a transition.

for the target system is shown in Fig. 23. See Chen et al. (2013b) for the details of modeling and analysis.

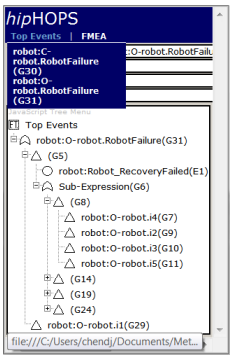

Fig. 23. An excerpt of the fault tree (FT) generated by the *HiP-HOPS* analyzer for the omission failure.

Assume that the target system (Fig. 9) operates in multiple modes with different cost and reliability levels. For the product feature *mainBody*, three grippers can be applied (Fig. 17). These grippers, due to their operational concepts, also imply different cost and reliability levels. The choice from the mode and gripper combinations becomes then a multi-objective optimization problem, which can be solved by algorithms as introduced in Deb (2001). Fig. 24 shows the assessment of configuration candidates given by the combination of *3* modes (*M1~3*) and *3* grippers (*G1~3*), where the non-dominating candidates in regard to cost and reliability constitutes the pareto front.

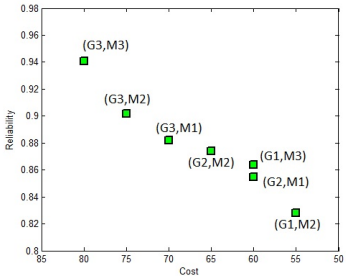

Fig. 24. The quality assessment for the configurations of operation mode M and gripper G combinations.

5. CONCLUSION

We have presented a virtual environment that provides an infrastructure in terms of ontology, modeling and tool support for advanced production systems. We believe that the exploitation of EAST-ADL framework as well as its related state-of-the-art technologies would be necessary for an effective model-based approach. As future direction, an explicit traceability between the EAST-ADL based system models and the business process models (e.g. based on *BPMN* - Business Process Model Notation) is currently under investigation. This would enable more comprehensive model-based decision-making even for other stakeholders like business and process managers of production. Such a seamless integration of system models and business process models would also facilitate the development of qualified controllers for highly autonomous process control or adaption of advanced production systems.

REFERENCES

R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution *Proceedings of the 47th Design Automation Conference*, pp. 731-736.Fundamenta Informaticae, 2010.

W. Wolf. *Cyber-physical systems*, Computer, vol. 42, pp. 88-89, 2009.

F. P. Maturana and D.H.Norrie. *A Generic Multi-Agent Mediator for Distributed Manufacturing Systems* Journal of Intelligent Manufacturing, Vol.7, 1996.

M. Onori. Evolvable Assembly Systems: A New Paradigm? *Proceedings of the 33rd International Symposium on Robotics*, Stockholm, Sweden, 2002.

M. Onori and J. Barata. Evolvable Production Systems: Mechatronic Production Equipment With Process-Based Distributed Control *9th IFAC Symposium on Robot Control*, Gifu, Japan, 2009.

M. Onori, N. Lohse, J. Barata, and C. Hanisch. *The IDEAS project: plug&produce at shop-floor level* Assembly Automation, vol. 32, pp. 124-134, 2012.

EAST-ADL. *EAST-ADL Domain Model Specification*, Version M.2.1.12. 2014. http://www.east-adl.info/

AUTOSAR. *AUTOSAR*, 2013. http://www. autosar.org

PackML. Packaging Machine Language *PackML*, 2015. http://omac.org/content/pack-ml-download

D. Chen, R. Johansson, et al. Integrated Safety and Architecture Modeling for Automotive Embedded Systems. *e&i*, vol. 128, Number 6. Springer. 2011.

D. Chen, L. Feng, et al. An Architectural Approach to the Analysis, Verification and Validation of Software Intensive Embedded Systems. *Computing*, Springer. DOI: 10.1007/s00607-013-0314-4. 2013. (2013a)

D. Chen, N. Mahmud, et al. Systems Modeling with EAST-ADL for Fault Tree Analysis through HiP-HOPS. *4th IFAC Workshop on Dependable Control of Discrete Systems*, York, U.K. September 4th- 6th, 2013 (2013b)

M.A. Peraldi-Frati, A. Goknil, J. Deantoni, Johan Nordlander. A timing model for specifying multi clock automotive systems: The timing augmented description language V2. IEEE 17th International Conference on Engineering of Complex Computer Systems, ICECCS 2012, Paris; 18 20, July. 2012.

Y. Papadopoulos, J.A. McDermid. Hierarchically performed hazard origin and propagation studies. *LNCS*, vol. 1698, pp.139-152. Springer. 1999.

N. Mahmud, Y. Papadopoulos, M. Walker. A Translation of State Machines to Temporal Fault Trees. *Proc. 40th Annu. IEEE/IFIP Int. Conf. Dependable Syst. and Netw.*, pp. 45-51, Chicago, Illinois, 2010.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley&Sons, p. 497, 2001.