Towards a Unified Architecture Methodology for Product Service Systems

Johan Cederbladh and Jagadish Suryadevara.

johan.cederbladh@.mdu.se

Abstract.

There is ongoing digital transformation in many industrial contexts including non-digital hardware-intensive domains such as heavy construction equipment machinery. This transformation is augmented by the latest technologies, increasing sustainability regulations, as well as integrated customer needs. This paradigm shift is causing transformation of current product-centric business models towards complex business models enabled by so-called Product Service Systems (PSS) offering advanced "capabilities" and "services". However, the development of PSSs requires a holistic approach to unify underlying solution domains such as mechatronic systems, embedded software control as well as information systems. In this paper, we describe a holistic "unified" architecture description methodology towards aligning the underlying domain-specific design and development contexts. The framework, developed in construction domain, consists of two main parts: a minimalistic reference model as the "common language" for cross-functional stakeholder communication; a modeling methodology based on existing standard modeling frameworks. We illustrate the methodology using examples from the construction equipment domain.

Introduction

Construction Equipment manufacturing is a heavy industry with major brands such as Caterpillar, Volvo, Komatsu, JCB etc. The equipment ranges from compact small machines to very large, heavy machines, e.g., road-paving equipment often requiring multiple operators. This equipment caters to major industrial segments such as construction, forestry, mining etc. Currently, the Construction Equipment domain is undergoing paradigm shifts from hardware-intensive machinery products towards software/data intensive integrated mechatronic information systems. The latter are complex System-of-Systems (SoS) (Maier 1998) referred to as Product Service Systems (PSS), where the physical product is part of a much larger entity, consisting of other physical products as well as information systems, providing "capabilities" and "services" beyond the scope of the functionality of a single product (Fakhfakh 2020). Capabilities can be summarized as "*the power or ability to do something"* and are attributed to systems or SoS (Martin et al. 2022). Capabilities are central to SoS as integrated systems form the foundation for the overall emerging SoS capabilities. A service instead can broadly be considered as "*an abstract product that is intangible and non-storable*" and PSS considers products that range from traditional "pure" products and "pure" services (Ashlin et al. 2016). The customers as well as the industry in entirety increasingly avail the benefit of integrated services and capabilities to enhance not only the profitability but also the sustainability through new business models that enable so-called "circular economy".

Historically the business models within the Construction Equipment industry evolved around the individual machine types so-called "product-families" offering customized machine

features or functionality. In this context, the standard Product Lifecycle Management (PLM) (Stark 2022) approach is based on implementing “Modular Product Architectures” (Bruun et al. 2013). In construction domain, the PLM approach is largely influenced by hardware and mechanical design as the major costs are attributed to the physical components of the machinery. The other technology domains such as electronics and software have been traditionally treated as “black-box” subsystems having independent development life-cycle besides standalone methodologies and tools often disconnected from the overall “architecture development" at machine level (latter in turn disconnected from the associated Information Systems). To enable the “digital transformation”, a unified common approach is needed across the various developmental teams, sometimes referred to “silos”. This work presents and evaluates a case where a unified common framework has been leveraged for cross-domain integration. To formulate a unified framework, a core ontology is presented for Construction Equipment. A core ontology centralizes knowledge by anchoring it in well-defined concepts in the domain (Martin et al. 2023). The presented core ontology is an enabler for a common architecture realized through the increased opportunities for collaboration and communication. This paper does not intend to propose an all-encompassing framework, but rather a “unifying” approach that aligns existing way-of-working across multiple domains. A minimalist “architecture language” is defined that can be mapped onto various solution domains such as hardware, systems, software, information, applications, and processes. Furthermore, a model-based development methodology proposed based on general systems engineering principles and existing modeling techniques is presented in this work. Based on a unified framework, the first steps towards a common architecture for PSS are taken, and we discuss the broader implications of such a framework in this work in the Construction Equipment domain.

Background

Systems Engineering (SE) has evolved as a holistic discipline ushering in broader “systems perspective” in product development as well as structured life-cycle management of corresponding system elements (Walden et al. 2015). The industry standard PLM methodologies are primarily based on ISO 15288 (Software and Systems Engineering) and ISO 26550 (Product line engineering) and deal with the following main areas such as systems engineering, product (and portfolio) management, manufacturing process management, and Product Data Management (PDM). The “unified” framework proposed in this paper is based on the “modularity principle” from methodologies described above. The standard ISO 42010 defines the concept of “architecture descriptions” in terms of *Views* and *Viewpoints* of a system-of-interest. An architecture “view” relates to specific system stakeholder concern and describes the system elements and corresponding relationships that are essentially impacted or governed by the concerns. The systems views are documented (or “modeled” using tool support) as part of the system specification. The standard ISO 42010 is the basis for evolution of several industrial frameworks (such as UAF (OMG 2022) and TOGAF (The Open Group 2023)), methodologies and tools for Systems and Software engineering domains. In the rest of this section below, we present an overview of current development methodologies in software, hardware, and emerging frameworks dealing with SoS development.

Software development methodologies have evolved over the past decades through object oriented (OO) design and analysis techniques. The use of abstraction techniques separates the specification and design processes from technical implementation, paving the way for reuse of

software designs. The standard Software Engineering methodologies employ the use of semi-formal languages like the Unified Modeling Language (UML) for specification and design of software architectures (OMG 2017). There exist domain-specific software Architecture Description Languages (ADLs). For automotive domain and embedded software development, the EAST-ADL has emerged as the de-facto standard language (EAST-ADL association 2021). The EAST-ADL language employs the general OO techniques and defines the software-centric "system model" through abstraction levels (i.e., architecture *views*) namely Technical Feature Model (Vehicle Level), Functional Analysis Architecture (Analysis Level), Functional Design Architecture and Hardware Design Architecture (Design Level). Model-Based Systems Engineering (MBSE) (Madni and Sievers 2018) is growing as a discipline where the holistic "system models" and the corresponding architecture views are the primary artefacts during the development (Suryadevara and Tiwari 2018). The term "model" loosely refers to any part or view of the system-of-interest that is in focus for development. The SysML has become the *de facto* industry standard language for systems engineering practices (OMG 2019). SysML is an extension of UML, for Systems Engineering domain. In SysML, views correspond to different types of diagrams, for example Requirements (*Requirement tables/diagrams*), Structure (*blocks*), Behavior (*use case, activity, statemachine*), and Parametrics (*attributes, valuetypes*).

As SE evolves further into SoS, both OMG and The Open Group have developed frameworks such as UAF and TOGAF respectively. These frameworks are supported by architecture methodologies and tools based on the underlying modeling languages i.e., SysML/UML and ArchiMate respectively (The Open Group 2022). There exist other languages and notations e.g., Business Process Modeling Notation (BPMN) (OMG 2023) supported by corresponding tools which can be easily mapped within enterprise frameworks such as UAF and TOGAF. For Construction Equipment domain, a well-defined enterprise framework can facilitate not only bridge the gap towards holistic development of PSS together with traditional machines but also support automation and digital transformation of the corresponding eco-system. There exist some established "unified" frameworks e.g., ARCADIA and CESAMES in the avionics domain (Roques 2016, CESAMES 2021). ARCADIA describes the stages of Operational analysis, System needs (functional and non-functional), Logical and Physical architectures. The shared architecture models that can be used by several domains. Similarly, the CESAMES framework describes operational (why), functional (what), and constructional (how) "views" of the system. RAMI 4.0 (Hankel & Rexroth 2015), the Reference Architecture Model Industry 4.0 is a reference model for structured description and management of architectures for cross-functional domains in a distributed industry 4.0. The framework defines a toolbox (RAMI 4.0-Toolbox) to implement a service-oriented architecture methodology for development in Industry 4.0.

SoS examples from Construction Equipment Domain

A SoS is a system consisting of interconnected collaborating Constituent Systems (CS) with operational and managerial independence, apart from the overall SoS goals and objectives (Maier 1998). The increased complexity of SoS compared to traditional systems introduces several challenges as domains should be aligned to collaborate efficiently. Requirements need to be adequately captured for several stakeholders and contexts, that is capture individual CS

and the SoS concerns. There is a need to enable interoperability, a common language, traceability, among others, between the involved stakeholders and domains. There is many potential SoS in the Construction Equipment domain, as it often involves large construction or quarry sites that operate for long periods of time. And many operational scenarios in above contexts can be considered as SoS perspective. Figure 1 provides a high-level view of many operational contexts related to the Construction Equipment domain and corresponding industry segments such as Roads and Building, Material Transport, Forestry, Quarry and Mining, Waste management etc.

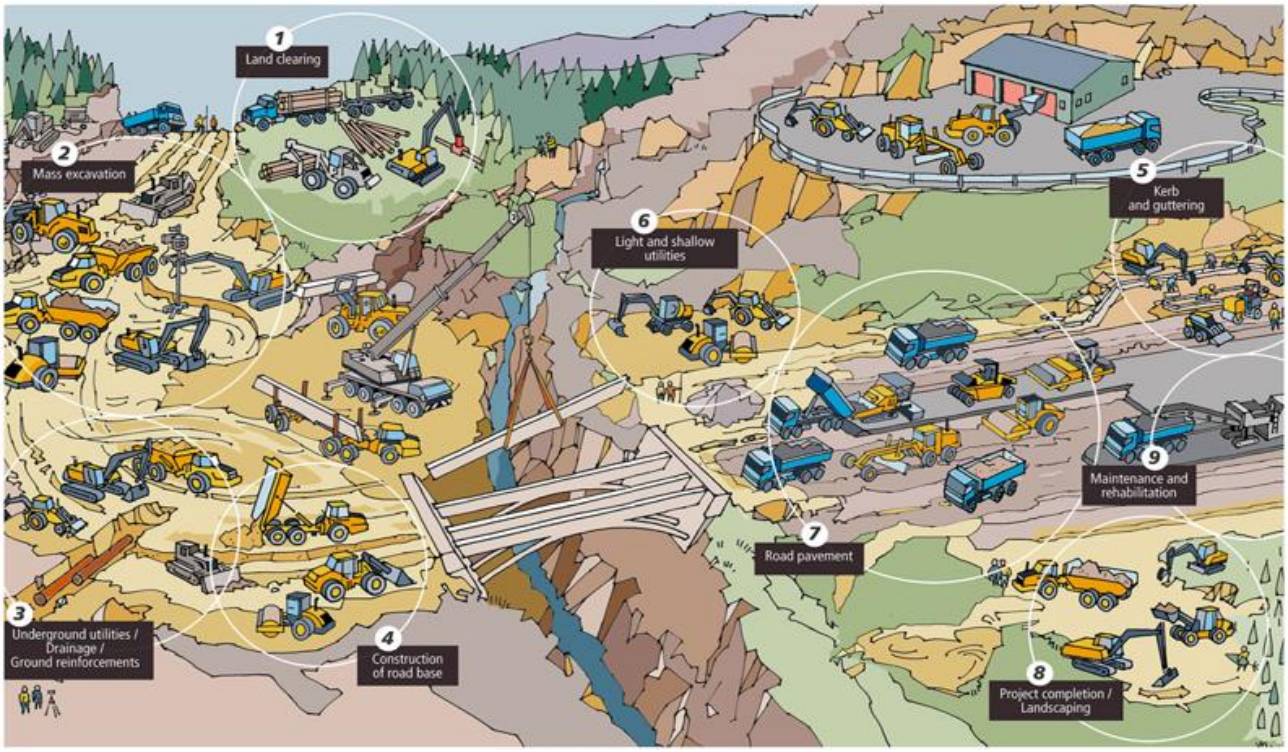

Figure 1 – Operational scenarios in construction domain: The SoS contexts

To further demonstrate SoS context in construction domain, and corresponding PSS "capabilities" and "services", we describe an ongoing *proof-of-concept* project. The example is a partially electrified autonomous material production site (E-site) (Sjöberg et al. 2017). Figure 2 illustrates the operational scenarios involved.

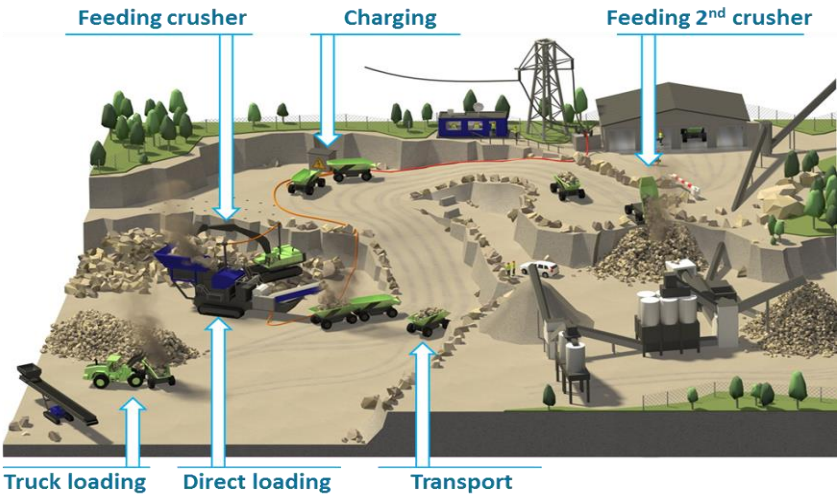

Figure 2 – Partially electrified autonomous production site (E-site)

The scenarios involve both conventional machines e.g., heavy-duty excavators, loader, and stone crushers as well as the electrified autonomous material transporters (i.e., compact autonomous haulers). The latter requires periodic charging at a dedicated location within the site. While the material (i.e., crushed stones) is autonomously transported to second crushing location within the site, the material loading (onto the autonomous units) is handled both conventionally (using a heavy loader) as well as automatically (through transport band). The operational scenarios are deceptively simplistic but involve several operational/performance requirements and constraints. For instance, the autonomous units need to transport material along the fixed paths in a non-collision, timely no-waiting mode interspersed with charging cycles as per individual machine needs. Further, work-safety and other security (e.g., cyber security) requirements need to be fulfilled. These constraints and requirements need to be considered during the development of the corresponding PSS responsible for the operational scenarios and ensure required productivity throughput. The main system components are the E-site Control (an operational station physically manned with supervisory role(s)), onboard autonomous control units communicating with the E-site Control. The manually operated units (i.e., heavy loaders, excavators) have secondary display/control units (co-pilot systems) to assist the operator with productivity cycles as well as other assist functionality (additional safety, diagnostic, v2v communication etc.). Thus, all the constituent systems and subsystems need to communicate and collaborate effectively to meet the productivity of the E-site.

A Unified Architecture Framework

In this section we present an overview of the proposed framework for development of PSSs. The framework defines an (abstract) architecture language consisting of a minimal set of architecture concepts for overall architecture and design management for a PSS of interest. As described later in following sections, the abstract architecture language can be mapped to concrete modeling languages and architecture methodologies in corresponding solution domains towards a unified model-based architecture methodology for a PSS.

The reference model of the framework, as shown in Figure 3.a illustrates multiple contexts involved in architecture development and the deployment of a PSS, its constituent systems and other system elements. The top of the model corresponds to enterprise “offerings” to customers i.e., capabilities and services provided by a PSS. A conventional machine product, as a constituent system, may represent either a capability, or a service or a mere physical product (i.e., a resource) in the PSS context. The other contexts and “layers” of the framework, and corresponding architecture views and main terminology, are briefly described below.

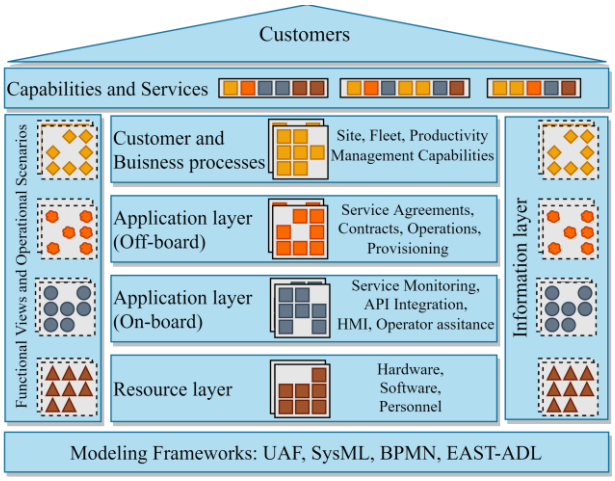

Figure 3.a – An Enterprise Reference model for PSS.

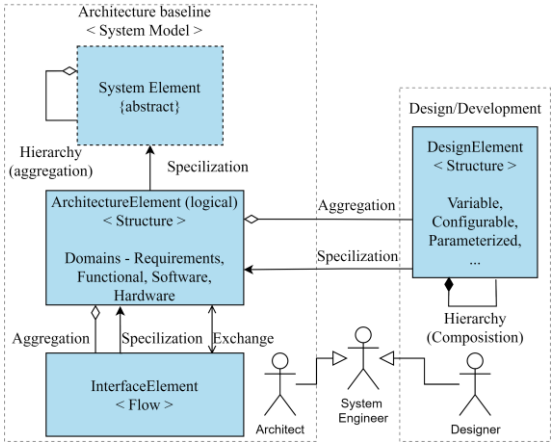

Figure 3.b – An Architecture Language – The Core Ontology.

Customers and Business Processes. The customer processes correspond to the operational scenarios of a SoS and the corresponding stakeholders' needs, whereas the business processes correspond to the capabilities (or services) "provided" by the PSS (using the resources already developed such as machines, systems, processes, personnel etc.). While a customer process is an external view (complimented further by detailed operational scenarios described later), the business processes represent the internal view of a PSS (and consists of elements from other layers of the framework described later). The clear separation-of-concerns prescribed by the framework enables architecture analysis and development of enterprise resources to meet the required capabilities.

Application Layer (Off-board). This layer represents the enterprise capabilities and services mostly in terms of enterprise Applications, to automate the business processes as well as support the functionality of both product (e.g., On-board applications) and non-product resources e.g., maintenance and support personnel etc. The application views describe the corresponding architecture in terms of life-cycle management of corresponding PSS, as shown in Figure 4.

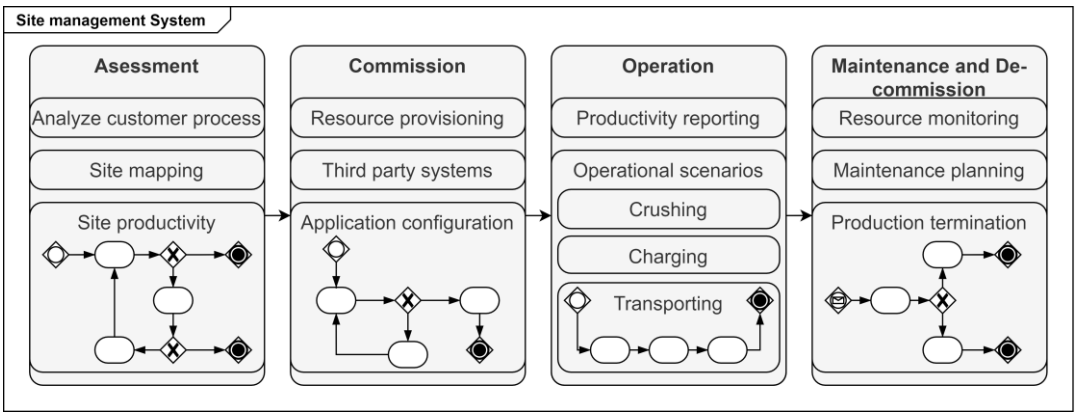

Figure 4 – The Architecture Life-cycle Management of a PSS (Business Process view)

Application Layer (On-board). This layer represents the "product" context within a PSS. For construction domain, this represents the traditional product-families and corresponding

machine “features” of onboard applications. The architecture views are mainly software-oriented whereas the resource layer described below deals with hardware elements.

Resource Layer. This layer represents the physical and tangible system elements including human resources (e.g., enterprise roles), as well as the third-party systems, that act as the “building blocks” for constituent systems of a PSS. Resource architecture deals with the detailed design management of a PSS.

The onboard application layer and the resource layer described above represent the traditional solution domains such as hardware, software, electronics etc. that constitute the main development silos and often contribute to the overall complexity of a PSS. It is critical to integrate these development domains within the architecture methodology for a PSS. It can be noted that a physical product may represent a sufficiently complex PSS as a “constituent system”, and thus suitable itself for application of the unified approach described in this paper. The layers described above constitute the “enablers” i.e., the building-blocks for ready deployment of a PSS. This also represents the “As-Is” architecture for current enterprise capabilities as well as the basis for future development of the corresponding enterprise layer. The vertical layers of the framework described below represent architecture information concerning all the other layers described above.

Information Layer. This layer overlaps all the layers of the framework described above as the information is both produced and consumed at several contexts during the life-cycle of a PSS.

Functional Views and Operational Scenarios. This layer represents the general “problem domain” for application of PSS capabilities and services. The functional views enable mapping the operational contexts to specific PSS elements as well as necessary for evaluation of “As-Is” architecture maturity and gap-analysis of current and future enterprise capabilities of a PSS.

Unified Language. The abstract architecture ontology, described in Figure 3.b, separates architecture concerns from detailed design issues. An architecture view consists of main architecture elements which represent specific capability or functionality that is realized in corresponding solution domains such as *process*, *application*, and *resources*. An architecture is realized in terms of design elements such as *hardware* and *software* which are outside the scope of the unified framework. The dependencies among architecture elements are represented by the logical interactions based on the corresponding resource exchanges and information flows. At the architectural level, the interactions do not constitute any constraints. The language incorporates a modularity approach based on the functional aspects of the corresponding architecture elements. For instance, a collection of architecture elements belongs to the same “module” to facilitate reuse in faster development and deployment of constituent parts of the corresponding PSS. These principles are same as the principle of “modularity” from product development (hardware engineering) context, as illustrated in Figure 6.

Model-based Methodology. For practical application of the unified framework described in this section, we propose to integrate existing MBSE techniques and modeling approaches. As described in the next section, the abstract architecture language can be mapped onto architecture concepts in the underlying solution domains. Further, the modeling techniques in the solution domains can be integrated into a unified architecture methodology as demonstrated in the following sections.

Unified Architecture Modeling – A Mapping Approach

In this section we describe architecture modeling techniques by mapping the general architecture concept model (Figure 3.b) onto general architecture concepts from solution domains described in previous section. Table 1 presents an overview of all the mappings in a "modeling-grid" form where each column represents general architecture concepts from corresponding solution domain context. The grid enables uniform dissemination of architecture information among PSS stakeholders. To model the architecture information, further mappings using standard modeling language SysML are also defined. We demonstrate the approach using PSS examples from the Construction domain.

Systems Engineering. Though this is an all-encompassing discipline in general, here we focus on the early phases e.g., architecture definition, in system development. This phase is gaining prominence with emerging MBSE methodologies. As shown in Table 1, the SE is further divided into architecture development in SoS, Systems and SW contexts, as each of these correspond to distinct life-cycle phase and different architecture development approaches.

Table 1 - Mappings to solution domain unified architecture concepts (abstract).

Domain	SoS	SE	SW	Process	Application	Information	Hardware
Core Ontology Concepts Mapping							
Architecture View	Operational Architecture	ContextDiagram, System Architecture	SW_Architecture, AnalysisFunction	Process, ProcessStep, Role	Application	Report	Assembly, Module
Interface Element	LogicalInterface	SystemInterface	SW_Interface, Device_Interface	ProcessInterface	API	Data (API) interface	Mechanical Interface
Flow Exchange	Energy / Material / Data Flow	Energy / Material / Data Flow	Signal	Trigger Information-Element	Information, Data	Information	-
Design Element	Constituent System	SubSystem, Function	SW_Function, SWComponent, DeviceFuntion	Constituent System, Role, SystemFunction	APIComponent, APIFunction	Data	Design Unit, Part

SoS Modeling. As shown in Figure 5, based on the E-site example described earlier, the operational architecture captures the "constituent systems" (modeled using SysML Block elements) e.g., Excavator, Charging station, Site Control, etc. The material, energy, and data flows between the constituent systems are modeled using the SysML Association relationships. Additionally, Figure 5 highlights a process activity diagram that represents the high-level process of the E-Site general operational scenario. The data flows are part of the "logical interface", also modeling using Association relationship between the constituent systems. Table 2 below shows all the mapped concepts to modeling elements, as annotated in Figure 5.

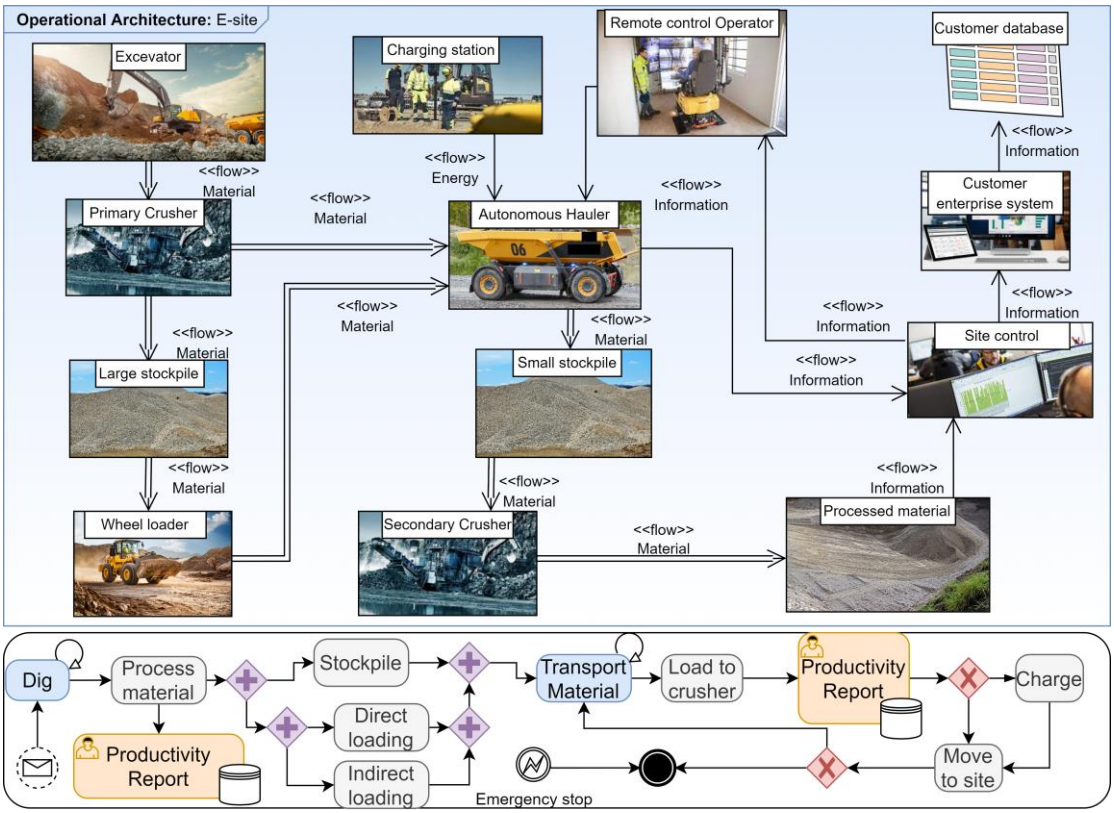

Figure 5 – SoS Modeling: Operational Architecture and an Operational Scenario.

Table 2 – SoS Architecture Modeling using SysML

Architecture Concepts	OperationalArchitecture	LogicalInterface	Energy/Material/Data Exchange	Constituent System
SYSML	BDD, IBD	Association	Flow	Block

Process Modeling. Figure 4 includes a process modeling example, based on the autonomous operation of the e-site example. The process model represents a BPMN-like notation. The different Processes (along with Roles and Actors) are modeled using Activities, Transactions, and Actions. Connecting the processes are directed associations and different decision gates (e.g, parallel). Specific Triggers are also modeled along with corresponding InformationElements using Events and Data Objects.

Table 3 – Process Modeling.

Architecture Concepts	Process, Role, Actor	ProcessInterface	Trigger, InformationElement	System Function
BPMN	Activity, Transaction, Action	Association	Event, Data Objects	Activity

Hardware Engineering. This is part of the Resource layer (hardware), a well-established architecting domain in PSS context. The learnings from this domain (namely, "modularity principle" in planning, development, production) forms the basis for the "unified" architecture framework and methodology. The architecture descriptions of this domain mainly focus on the physical "modularity" of the products (from planning, development, production perspective). The architectural elements are of kind *Machine, Module, Assembly.* The specialized design elements are *Design Unit, Key Component, Interface, Part* etc. Besides mainly the mechanical parts, this architecture view also contains physical parts of Electrical and Electronics i.e., ECUs, Sensors, Actuators etc. A specialized non-leaf level module (*Assembly*) namely "E&E

Architecture Module" can be instantiated to manage elements belonging to this subdomain. The leaf-level modules contain *Design Units* which are also "logical" grouping of the actual physical "parts". All the architecture blocks represent the reuse/sharing among the corresponding "product family" (managed through Governance process). The main stakeholders for this architecture view are from both development and production domains. This architecture view represents the "architecture" (a.k.a., "platform") of a given product kind within corresponding product family and guides the design and implementation phases downstream (e.g., A-build, B-build, C-build). Specific design elements (i.e., Design Units) selected to configure a specific product kind i.e., "machine model" (built on a production line in a factory, i.e., production domain).

Hardware Architecture Modeling.

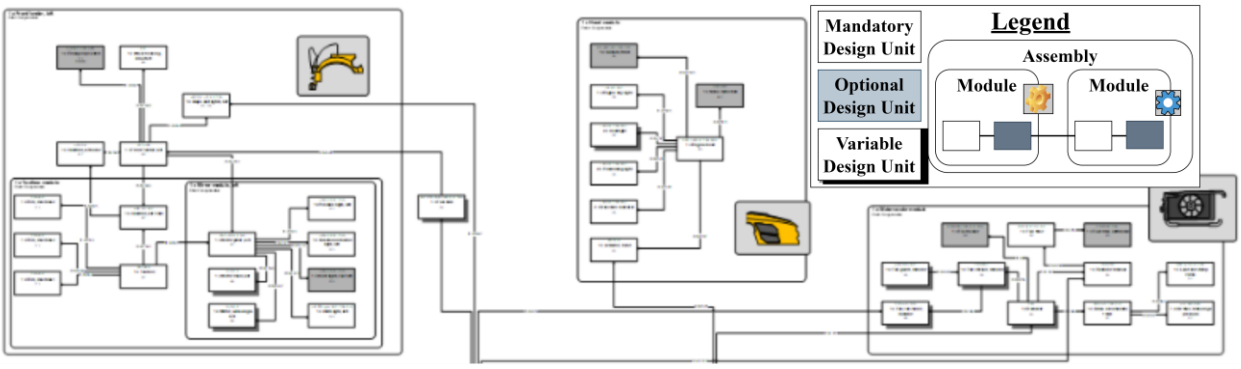

Figure 6 - Hardware Engineering - Architecture Modeling (Schematic)

As shown in Figure 6, the architecture view captures the "Hardware architecture" for the constituent system (Conventional Hauler), modeled using Microsoft Visio template (a plugin tool developed in-house). The main hardware architecture elements such as Modules and Design Units are visualized using Visio elements. An Assembly represents an aggregation of containing Module elements more logically than physical binding. Whereas a leaf-level Module element containing only Design Units constitutes a tightly coupled physical element that is built on an assembly line and integrated with containing Module or Assembly structure. A Module represents a reusable element capturing the "modularity principle". Although the example visualizes the Visio "legacy", the translation to SysML is quite straightforward and new models (or even automated translations of older models) can be expected in SysML. The mapping is shown in Table 4 below, also annotated in Figure 6.

Table 4 - Hardware Architecture Modeling.

Architecture Concepts	Assembly	Module	Design Unit (*Mandatory, Optional, Variant*)	Part	Mechanical Interface
SysML	Package	Package, Block	Block, Stereotypes	Block	Connector

Systems & Software Modeling. As shown in Figure 7 the architecture view represents the system architecture and software interfaces for a constituent system, modeled using SysML IBDs (Internal Block Diagram). SubSystem functions are modeled using SysML Activities. The data and signal "Exchanges" are modeled (using SysML InterfaceBlocks and Ports) between System and SubSystem elements (modeled using SysML Block element). Table 5 below shows the mapped concepts to modeling elements, annotated on Figure 7.

Table 5 - Systems Modeling

Architecture Concepts	ContextDiagram, SystemArchitecture	SystemInterface	Energy/Material/Data Exchange	Function	SubSystem Hardware
SYSML	BDD, IBD	Port	InterfaceBlock, Port	Activity	Block

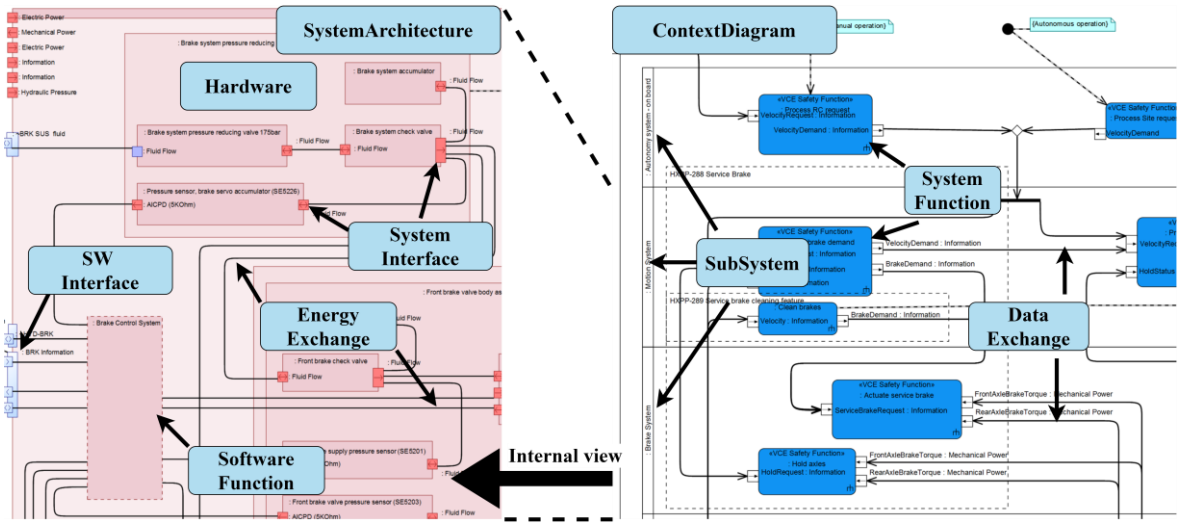

Figure 7 – Systems/Software Engineering - Architecture Modeling

Information Modeling. With digital transformation data is exchanged continuously between interconnected and interacting systems to fulfill enterprise capabilities and customer "digital services". The enterprise system manages huge amounts of data for both internal (i.e., development, production, deployment) as well as external purposes (integration with third-party suppliers and customer ecosystem). Information and data are not only critical for architecture concerns but are often associated with the direct business value. This requires architecture management of data sources, as well as information provisioning. As shown in Table 1, the main concepts identified for architecture management for information domain are *Reports*, *Information*, *Data,* and *Data*_API (data interface specifications). The *Reports* enable high-level capabilities (e.g., Predictive Maintenance and digital services providing by Fuel Utilization). A report consists of specific Information (e.g., Machine Position) which in turn made of other Data (e.g., Longitude, Latitude, Timestamp). Simple UML elements are used to realize the architecture descriptions (I.e., "Class", "Composite Class", Provided/Required Interface).

Table 6 - Information Modeling

Architecture Concepts	Report	Data (API) Interface	Information	Data
UML	(generated) Document	Provided/Required Interface	Composite Class	Class

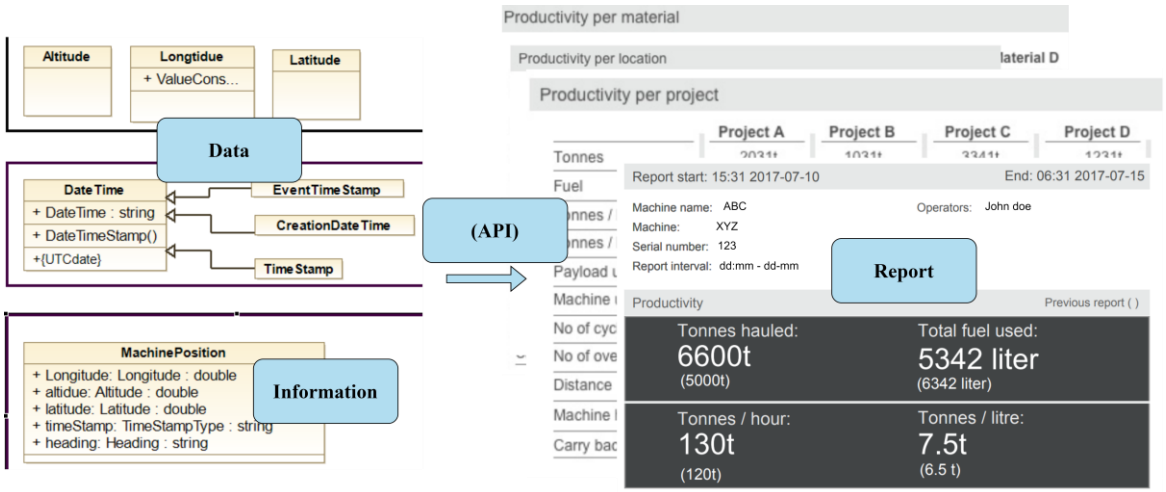

Figure 8 – Information Domain: The Architecture views

Domain Reference Architectures. In previous sections we have described the solution domains within an enterprise framework for the Construction domain. Also, based on a core ontology, a unified "architecture" development methodology proposed, which can be mapped onto existing solution domains. However, it is further required to complement the methodology with Domain Reference Architectures (i.e., the "functional views") (See Figure 3.a) corresponding to the solution domains mainly Process, Application and Resources. While the functional views are the starting point of systems development and part of general SE/MBSE methodologies, we emphasize an "architectural approach" and management of these views as part of the unified framework, as described further below.

For domain reference architectures, we can employ the generic principles of the framework as described earlier i.e., application of the proposed core ontology (Figure 3.b) and mapping to existing modeling techniques, as shown in Table 6. The main architecture concepts are Function, Functional Domain, and Capability, the latter two correspond to transparent grouping of functions. The function development traditionally applies SE/MBSE techniques such as operational scenarios, use case analysis, functional decomposition etc. The main architecture views are Functional Domains, and Capability view. A Functional Domain (i.e., functional views of the problem contexts) corresponds to one or more Capabilities (i.e., elements of corresponding solution domain) and vice versa. It can be noted that the Functional Domains represent the problem domain, where as the "capability" (logical groupings shown in Figure 3.a within solid boxes i.e. tightly-coupled) represent deployable solution element (e.g., application, resources). Both the architectural views described above facilitate stakeholder communication, effective use of solution domains as well enable overall architecture management. As shown in Table 6, the above architecture views can be modeled using the standard MBSE tools and modeling elements.

Table 6 – Domain Reference architectures: Concepts and Modeling Elements

CoreOntology Concepts	Architecture Block	Logical Interface	Design Element	
Reference Architecture Concepts	Capability Functional_Domain	Traceability	Stakeholder Need System Requirement	CustomerValueProposition, Function
SysML	Package	Association	Functional_Requirement	UseCase

	Block	Dependency	Non-Functional Requirement	Activity

Discussion

Traditionally construction domain has been hardware intensive with major costs of development involving hardware components. However, the complexity has been shifting towards software and electronics intensive solutions with increasingly data-intensive machine functionality and digital services. In previous sections we presented an enterprise framework based on an abstract architecture language that can be mapped into model-based architecture methodologies in underlying solution domains such as hardware, software, electronics, and information systems. This is a major step forward as it aligns both embedded control systems as well as information systems development through a common framework. As illustrated in Figure 3.a, the services and capabilities provided by a PSS is a composition of various architecture entities from different solution domains described in this paper. The enterprise behind a PSS represents a SoS involving many collaborating organizations providing the required capabilities and services as well as other constituent systems to be integrated. With increased complexity, PSS development and management will benefit from the application of the model-based architecting methodology described in this paper. The model-based techniques enable traceability and logical dependencies among architecture elements for efficient configuration and deployment of PSS architectures.

Despite the successful adoption of model-based techniques in certain solution domains, in particular software development, general application of MBSE lacks maturity due to development silos in product development. The unified approach proposed in this paper provides an architecture-centric approach to systems development that facilitates better integration of software and hardware development. As shown in Figure 7, the software context (Software Function) is explicitly modeled within the System Architecture view of the system model. The software architecture represents the software interfaces and interaction with the corresponding hardware components (see Figure 7). The detailed software behavior can be specified using behavior diagram (such as SysML Statemachine diagrams, Activity diagrams). With model-based techniques and simulation tools, software behavior can be verified and validated against the system requirements associated with the system model. In construction domain, manufacturing industries have adopted product line management techniques based mainly on mechanical modularity. For instance, Figure 6 represents a hardware (mechanical) architecture view of a machine product with no information about functional and software interfaces. However, using the unified architecture approach described in this paper, the hardware architecture (see Figure 7) can be aligned within the overall system model. With model-based approach, hardware designs can be verified and validated against the system requirements using the system models where hardware components are specified as FMUs (Functional Mock-up Units). A set of FMU (based on FMI standard) specifications for hardware components enables a simulation model for design trade-off analysis as well as early-phase analysis. A unified framework as described in this paper enables seamless development of both software and hardware. The main hardware components are represented within the system model and interfaces to software and systems behaviors are specified. Similar to Software simulation techniques described earlier, there is increasing support towards integrated tools-

based enterprise frameworks and modeling methodologies provided by major tool vendors such as PTC, Siemens, Dassault etc. The emerging standard OSLC (Open Services for Lifecycle Collaboration) is a welcoming step forward to integrate the domain-specific tools, design methodologies and processes towards rigorous SE processes at enterprise level.

Conclusion

In this paper we presented a unified architecture methodology for development of Product Service Systems. The framework is based on defining a core ontology, as a "common language" to document/model main architectural elements from several solution domains such as software, hardware, systems, applications, information, as well as business processes. The architectural methodology is also used to specify system and functional views enabling the traceability of corresponding design elements as well as to system and stakeholder requirements. For practical purposes and to achieve the systems engineering rigor, the methodology is mapped onto the standard methodologies and tools of the solution domains. As a first step, the approach is applied within the product context to bridge the traditional development silos e.g., software, electronics, hardware engineering, as well as IT systems domains. The proposed unified framework lays the foundation for structured planning of digital transformation and enables automation within integrated product and services contexts.

References

Ashlin, S. J., Cardow, I., Crawford, A., Davies, J. K., Farncombe, A., & Mason, P. (2016, July). Understanding Services: Understanding Stakeholders. In *INCOSE International Symposium* (Vol. 26, No. 1, pp. 2226-2240).

Bruun, H. P. L., Mortensen, N. H., & Harlou, U. (2013). PLM support for development of modular product families. In DS 75-4: *Proceedings of the 19th International Conference on Engineering Design (ICED13),* Design for Harmonies, Vol. 4: Product, Service and Systems Design, Seoul, Korea, 19-22.08. 2013.

CESAMES, 2021, CESAMES Systems Architecting Method (CESAM), URL: <https://cesam.community/method-and-tools/>, Accessed August 2023

EAST-ADL Association, 2021, EAST-ADL, URL: <https://www.east-adl.info/Specification.html> Accessed August 2023

Fakhfakh, S., Hein, A. M., Jankovic, M., & Chazal, Y. (2020, May). A meta-model for product service systems of systems. In *Proceedings of the Design Society: DESIGN Conference* (Vol. 1, pp. 1235-1244). Cambridge University Press.

Friedenthal, S., Moore, A. and Steiner, R., 2014. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann.

Hankel, M., & Rexroth, B. (2015). The reference architectural model industrie 4.0 (rami 4.0). *Zvei*, 2(2), 4-9.

IEEE, 2022, 42010-2022 - ISO/IEC/IEEE Systems and software engineering -- Architecture description, URL: <https://standards.ieee.org/ieee/42010/6846/> Accessed August 2023

(———), 2015, 15288-2015 - ISO/IEC/IEEE Systems and software engineering – System life cycle processes, URL: <https://standards.ieee.org/ieee/15288/5673/> Accessed August 2023

ISO, 2015, 26550-2015 - ISO/IEC Systems and software engineering – Reference model for product line engineering and management, URL: <https://www.iso.org/standard/69529.html> Accessed August 2023

Madni, A.M. and Sievers, M., 2018. Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, *21*(3), pp.172-190.

Maier, M.W., 1998. Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, *1*(4), pp.267-284.

Martin, J., Axelsson, J., Carlson, J., & Suryadevara, J. (2022, October). The Capability Concept in the Context of Systems of Systems: A Systematic Literature Review. In *2022 IEEE International Symposium on Systems Engineering (ISSE)* (pp. 1-8). IEEE.

(———). (2023, June). Towards a Core Ontology for Missions and Capabilities in Systems of Systems. In *2023 18th Annual System of Systems Engineering Conference (SoSe)* (pp. 1-7). IEEE

Modelica Association, 2022, Functional Mockup Interface (FMI) V3.0, URL: <https://open-services.net/specifications/> Accessed August 2023

Object Management Group (OMG), 2023, Business Process Model and Notation (BPMN) V2, URL: <https://www.bpmn.org/> Accessed August 2023

(———), 2022, Unified Architecture Framework (UAF) V1.2, URL: <https://www.omg.org/spec/UAF> Accessed August 2023

(———), 2017, Unified Modeling Language (UML) V2.5.1, URL: <https://www.omg.org/spec/UML/2.5.1/About-UML> Accessed May 2023

(———), 2019, OMG Systems Modeling Language (OMG SysML) V1.6, URL: <https://www.omg.org/spec/SysML/1.6/PDF> Accessed August 2023

OSLC Open Project, 2021, Open Services for Lifecycle Collaboration (OSLC) V3.0, URL: <https://open-services.net/specifications/> Accessed August 2023

Roques, P., 2016, January. MBSE with the ARCADIA Method and the Capella Tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.

Sjöberg, Peter, Lars-Olof Kihlström, and Matthew Hause. "An industrial example of using Enterprise Architecture to speed up systems development." In *INCOSE International Symposium*, vol. 27, no. 1, pp. 401-417. 2017.

Stark, J., 2022. Product lifecycle management (PLM). In *Product Lifecycle Management (Volume 1) 21st Century Paradigm for Product Realisation* (pp. 1-32). Cham: Springer International Publishing.

Suryadevara, J., & Tiwari, S. (2018, December). Adopting mbse in construction equipment industry: An experience report. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 512-521). IEEE.

The Open Group, 2022, ArchiMate V3.2, URL: <https://publications.opengroup.org/standards/archimate/c226> Accessed August 2023

(———), 2022, TOGAF, URL: <https://www.opengroup.org/togaf> Accessed August 2023

Walden, D.D., Roedler, G.J. and Forsberg, K., 2015, October. INCOSE systems engineering handbook version 4: updating the reference for practitioners. In *INCOSE International Symposium* (Vol. 25, No. 1, pp. 678-686).