Computer Standards & Interfaces 86 (2023) 103744

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Towards a method to quantitatively measure toolchain interoperability in the engineering lifecycle: A case study of digital hardware design

Jose María Alvarez-Rodríguez [a,*], Roy Mendieta [b], Eduardo Cibrián [a], Juan Llorens [a]

[a] Department of Computer Science and Engineering, Carlos III University of Madrid, Av. De la Universidad 30, 28911, Leganés, Spain
[b] The Reuse Company Inc, Avd. Margarita Salas, 28919, Leganés, Spain

ARTICLE INFO

Keywords:
Software tools
Software reusability
Web services
Software as a service
Internet

ABSTRACT

The engineering lifecycle of cyber-physical systems is becoming more challenging than ever. Multiple engineering disciplines must be orchestrated to produce both a virtual and physical version of the system. Each engineering discipline makes use of their own methods and tools generating different types of work products that must be consistently linked together and reused throughout the lifecycle. Requirements, logical/descriptive and physical/analytical models, 3D designs, test case descriptions, product lines, ontologies, evidence argumentations, and many other work products are continuously being produced and integrated to implement the technical engineering and technical management processes established in standards such as the ISO/IEC/IEEE 15288:2015 "Systems and software engineering-System life cycle processes". Toolchains are then created as a set of collaborative tools to provide an executable version of the required technical processes. In this engineering environment, there is a need for technical interoperability enabling tools to easily exchange data and invoke operations among them under different protocols, formats, and schemas. However, this automation of tasks and lifecycle processes does not come free of charge. Although enterprise integration patterns, shared and standardized data schemas and business process management tools are being used to implement toolchains, the reality shows that in many cases, the integration of tools within a toolchain is implemented through point-to-point connectors or applying some architectural style such as a communication bus to ease data exchange and to invoke operations. In this context, the ability to measure the current and expected degree of interoperability becomes relevant: 1) to understand the implications of defining a toolchain (need of different protocols, formats, schemas and tool interconnections) and 2) to measure the effort to implement the desired toolchain. To improve the management of the engineering lifecycle, a method is defined: 1) to measure the degree of interoperability within a technical engineering process implemented with a toolchain and 2) to estimate the effort to transition from an existing toolchain to another. A case study in the field of digital hardware design comprising 6 different technical engineering processes and 7 domain engineering tools is conducted to demonstrate and validate the proposed method.

1. Introduction

The notion of Cyber-Physical Systems (CPSs) was first introduced in 2006 by the United States "*to represent the Integration of computation, networking and physical processes where CPS range from minuscule (pace makers) to large-scale (e.g. national power-grid)*" [1]. Other authors [2] define CPSs as "*transformative technologies for managing interconnected systems between its physical assets and computational capabilities*". It is possible to find many examples of CPSs in domains such as Industry 4.0 (e.g. digital factories), aerospace and defense, automotive, environment control, critical infrastructures, or health care devices to name a few where there is a clear need of integrating software-intensive capabilities with physical processes and entities.

However, CPSs [2] are far beyond traditional embedded control systems since they pose specific characteristics: 1) inclusion of software capabilities in every physical component; 2) networking at a multiple scale; 3) dynamic configuration; 4) high degree of automation and 5) functions that must be dependable and, in many cases, certified. In this context, CPSs represent a type of information and communication systems characterized by several quality indicators such as performance,

* Corresponding author.
E-mail address: josemaria.alvarez@uc3m.es (J.M. Alvarez-Rodríguez).

https://doi.org/10.1016/j.csi.2023.103744
Received 28 February 2022; Received in revised form 28 February 2023; Accepted 18 March 2023
Available online 21 March 2023

dependability, security, safety, large geographical distribution, and very large scale of control. Given this context, some research works [3,4] have defined some challenges in the development lifecycle of CPSs. More specifically, bringing the principles of DevOps (Development and Operations and toolchains) to CPS development [5] is a cornerstone to improve both automation of the lifecycle processes and integration of tools across the design-operation time.

The engineering lifecycle of a CPS includes multiple engineering disciplines (methods and tools) that must be orchestrated building executable technical processes (e.g. requirements analysis in [6]) through toolchains increasing the complexity of the engineering process. To improve the engineering process and provide a holistic view of the system, a digital transformation process is being conducted in the industry to shift the engineering paradigm from a document-oriented to a digital asset approach. Automation, modelling, reuse, and simulation are considered the key techniques to provide a new collaborative engineering environment in which processes such as verification and validation [7] can be easily implemented, automated and reused through toolchains.

In terms of the engineering process and the system development lifecycle, traditional linear approaches, e.g. the Waterfall or Vee models, or iterative models like Concurrent Engineering have been re-designed to support the digitalization of the engineering (DE) process. A better management of relationships between original equipment manufacturers (OEM) and suppliers is also required to share (digital data package) requirements, descriptive and analytical models, etc. boosting collaboration. The Digital Thread [8] by the Boeing Company has introduced the concept of a diamond lifecycle, see Fig. 1, to provide a virtual version of the well-known Vee model linking the physical and virtual worlds. This new engineering lifecycle makes an intensive use of digital assets like models, as unit of exchange, and simulation techniques enabling the notion of digital twin through automation and collaborative engineering [9].

Other initiatives to improve the system engineering process and the development lifecycle can be found in the Future of Systems Engineering Roadmap [10, pp. 2019–2020] promoted by the SERC (Systems Engineering Research Center). This document establishes five major goals to strategically change the engineering process: 1) "model use for decision making", 2) "authoritative source of truth", 3) "technology innovation", 4) "collaborative environment" and 5) "digital engineering (DE) workforce and culture". More specifically, Goal 2 "Authoritative Source of Truth" includes the need of a data integration/interoperability framework, DE design process, semantic data links and digital twin innovation as a path to reach an augmented engineering process.

The definition of a strategy for DE strongly relies on technological support implemented through different tools that must be seamlessly orchestrated (toolchain) to provide new collaborative engineering environments, see Table 1. Furthermore, some key processes such as system traceability or quality management and, in general technical management processes, may also require an integrated view of the system under development including different types of work products through the access to the corresponding engineering tools.

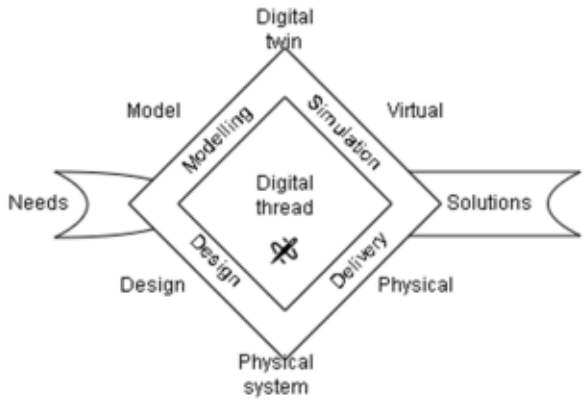

Fig. 1. Evolution of the Vee model lifecycle and notion of "Digital Thread" by the Boeing Company [8].

From a technical point of view, the implementation of technical engineering processes through toolchains is basically a problem of data interoperability and process integration. In the engineering lifecycle context, the efforts made by the family of standards ISO 10303-STEP (Standard for the Exchange of Product Model Data) or OASIS OSLC (Open Services for Lifecycle Collaboration) represent the major initiatives to cover data exchange of multiple work products within a federated environment of services (REST-based architectural style). Emerging standardization efforts such as the SysMLV2 API (Systems Modeling Application Programming Interface and Services specifications) [11, p. 0] also follow a similar approach in which data is encoded as a model and the communication among tools must occur under a set of REST services. Although these approaches are paving the way to unify the communication and exchange of data among tools, there is no more decision criteria than the use of well-recognized standards to evaluate which is the status of interoperability and the effort to transition from one technical approach to another.

In summary, there is a need to improve the engineering practice of CPSs to consider: 1) the digitalization of engineering work products and technical engineering processes, 2) the creation of collaborative engineering environments and 3) the automation and communication at different levels of abstraction: people (e.g. engineering teams), organizations (e.g. manufacturers-suppliers), engineering stages/activities and methods (e.g. requirements engineering and verification/validation) and tools (e.g. requirements management system and logical modelling tools). In this context, standardization remains as a key-enabler to implement the automation of technical engineering processes through toolchains. In fact, one of the means to reach a proper level of automation and communication within the development lifecycle relies on providing different levels of toolchain interoperability enabling the communication and exchange of data, information and knowledge between people, organizations, and tools. Table 1 shows examples of building toolchains to implement different scenarios that require the collaboration of domain engineering tools and different actors.

However, the implementation of interoperable engineering environments with toolchains does not come free of charge. It requires an effort to agree which parts must be interoperable and how this interoperability can be achieved from a technical and technological perspective, see Section 2.1. Most of the interoperability evaluation models that have been defined, see Section 2.2, are focused on the overall capacity of an organization. These models can be used to evaluate what is the current and target status of a software environment in terms of interoperability. However, they do not explicitly define a method to calculate the effort to transition from one level to another. Therefore, the implementation of interoperability mechanisms cannot be completely measured and estimated within an organization or engineering team.

That is why, in this paper, authors review existing interoperability evaluation models making an evaluation of their applicability to the context of system lifecycle and toolchains. Afterwards, an evaluation and estimation of effort method for interoperability is theoretically defined. To verify the applicability of the proposed method, a case study in the frame of the H2020-AHTOOLs (Arrowhead Tools for Engineering of Digitalization Solutions) project and the Use Case 3 (UC-3) is conducted to demonstrate its applicability to a real toolchain environment to design digital hardware. Finally, some conclusions and future research directions are also outlined.

2. State of the art

According to the ISO/IEC 2382-15-" Information technology —

Table 1
Use case scenarios combining different Technical Engineering processes, methods (tools) and actors.

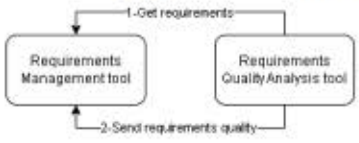

Figure 2 Use case scenario for checking the quality of requirements.

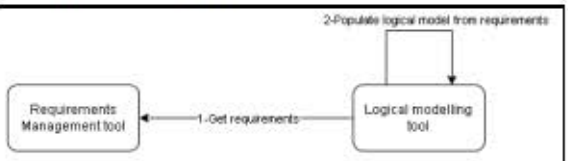

Figure 3 Use case scenario for populating logical models from requirements.

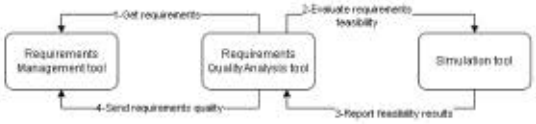

Figure 4 Use case scenario for checking the feasibility and quality of requirements with a simulation tool ("executable requirements").

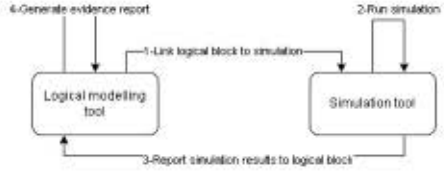

Figure 5 Use case scenario for linking physical blocks to logical models.

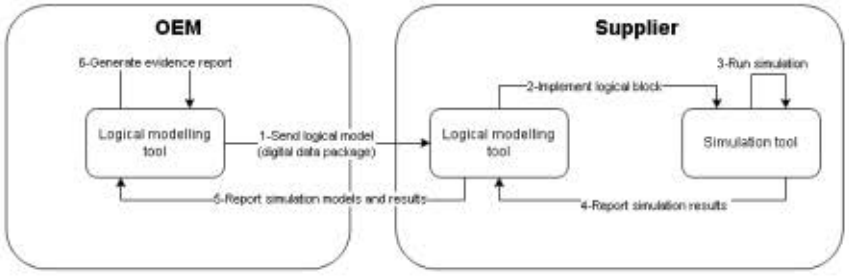

Figure 6 Use case scenario for sharing engineering information between an OEM and a supplier.

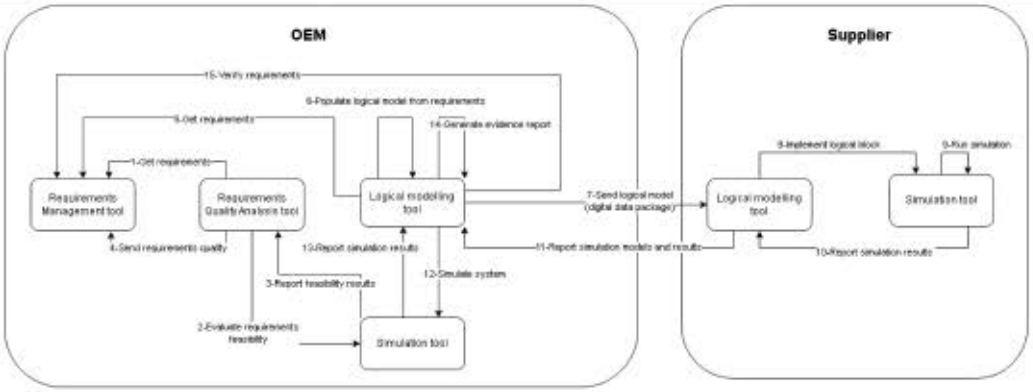

Figure 7 Use case scenario for sharing engineering information between an OEM and a supplier integrating the results in the OEM toolchain.

Vocabulary", interoperability can be defined as follows: "*The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.*" In this work, we take this definition as a reference for the required technical interoperability within a tool-chain to implement executable technical engineering processes.

2.1. Interoperability within the systems engineering process

In the frame of cyber-physical systems lifecycle development, the use of architectural frameworks, standardized languages, common data models and communication protocols are common practices to enable technical interoperability in both sides: development and operation. Model-based Systems Engineering (MBSE) [12] has emerged as a complete methodology to address the challenge of unifying the techniques, methods and tools within the development lifecycle. This means a "formalized application of modelling" to support the left-hand side in the *Vee* lifecycle model and the upper-side section of the diamond model, see Fig. 1, implying that any process, task, or activity will generate different system artifacts but all of them represented as a model.

The MBSE approach is considered a cornerstone [13] for the improvement of the current practice in the Systems Engineering

discipline since it is expected to cover multiple domains [14], to provide better results in terms of quality and productivity, lower risks and, in general, to support the concept of continuous and collaborative engineering. However, the MBSE approach considers that everything can be a model, e.g. a logical/descriptive SysML (System Modelling Language) model or a physical/analytical model, and this assumption is not always true in the development of a cyber-physical system. For instance, requirements are still specified as text statements, test cases description are usually defined using a restricted natural language and, in general, any piece of information that must be shared should also have a verbal representation since not everyone in an engineering team may understand or have the same interpretation of a model. Furthermore, the mere use of models as first-class members of the engineering process does not guarantee process automation and interoperability within the toolchain, it only unifies what is the type of artifact to be exchanged.

Technical interoperability initiatives for cyber-physical systems development such as the family of standards ISO 10303-STEP (Standard for the Exchange of Product Model Data) or the OASIS OSLC (Open Services for Lifecycle Collaboration) specifications, try to boost the implementation of technical engineering processes through an interoperable approach [15,16], easing the creation of federated environments of services (tools). Both define a collaborative engineering ecosystem through the definition of data shapes or schemes that serve us as a contract to get access to information resources. The Representational State Transfer (REST) software architecture style is used in both to manage information resources that are publicly represented and exchanged in different formats such as JSON or XML. Last version of the SysMLV2 API (Systems Modeling Application Programming Interface and Services specifications) [11, p. 0] is also following a similar approach, defining a REST API to consume SysML models.

All these efforts to improve the technical interoperability within the Systems Engineering process have been focused on providing technical solutions through different approaches like architectural frameworks, service-oriented computing [17] (e.g. OSLC-based toolchain [18]), unified data models, ontologies [19] and communication protocols or integration patterns [20]. However, the focus is mainly on the implementation of the different technical approaches (e.g. use of services, common data models, etc.). Although some technical tasks such as the automatic generation of service providers and clients can be automated to bring tools into a toolchain, the evaluation of interoperability and the estimation of effort to transition from one level of interoperability to another is not covered.

2.2. Interoperability evaluation models

According to the systematic review in [21] conducted in the frame of European research project FP7 ENSEMBLE, there are many types of interoperability evaluation or capability models that may change depending on their philosophy and implementation. Following, a summary of the main interoperability models is presented:

The LISI ("Levels of Information Systems Interoperability") model [22] was first developed by the US Department of Defense in 1988. This model provides a standard process for evaluating the interoperability of information systems. The LISI capabilities model comprises:

1) Five levels of interoperability maturity: 0-*Isolated* (manual integration of data and information), 1-*Connected* (peer-to-peer, basic exchange of data and information), 2-*Functional* (distributed, shared logical data models across systems), 3-*Domain* (integrated, shared domain data models across systems) and 4-*Enterprise* (universal: unified domain models and interpretation across systems) and
2) Four technical interoperability attributes (PAID), the key-enablers to reach a maturity level and to enable data/information exchange: *Procedures* (policies, standards, and procedures), *Applications* (set of applications for exchanging, processing and manipulation of data and information), *Infrastructure* (environment: networks, security, etc.) and *Data* (formats, protocols, or databases).

A LISI assessment process commonly follows the next stages: 1) fill a LISI questionnaire, 2) establish a system profile to assess the current interoperability level and the PAID attributes, 3) create a score card of the systems to assess the interoperability maturity, 4) establish a strategy for the improvement of the interoperability usually based on expert judgement and 5) apply the strategy to progress in the maturity level. In general, the LISI model offers us a method and a process to evaluate both the organizational and technical interoperability. This model can be used to measure the interoperability level before and after the implementation of interoperability mechanisms. Thus, it is possible to have a picture of the status for a toolchain environment and its evolution. There is also a review called "the Extended LISI model" that includes more abstract layers for command-and-control support. However, it does not include any mechanism to estimate the effort to transition from one level to another and it is hard to apply the model to a concrete environment like a toolchain (network of connections).

The LCIM ("Levels of Conceptual Interoperability Model") [23] emerged to provide a conceptual interoperability model beyond pure technical models like the LISI model with special focus on simulation connection problems [24]. It has also been applied to other domains like healthcare [25]. The LCIM model also establishes different levels of interoperability: 0-*Stand-alone* (no systems interoperability), 1-*Technical interoperability* (communication networks and protocols are established enabling data exchange), 2-*Syntactic interoperability* (a common data format is used to represent data and information), 3-*Semantic interoperability* (a common information schema is used to model data and information), 4-*Pragmatic interoperability* (interoperating systems are able to understand methods and operations using a common information schema), 5-*Dynamic interoperability* (systems are able to adapt their behavior to the changes in the environment, e.g. data model or operation definitions) and 6-*Conceptual interoperability* (conceptual models and methods are documented and specified under a common knowledge framework). As in the LISI model, the LCIM model provides us with an excellent tool to establish a level of interoperability but the estimation of effort to transition from one level to another is not completely defined.

The EIMM ("Enterprise Interoperability Maturity Model") [26] was created in the frame of the European research project ATHENA ("Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications") with the aim of providing a methodology, guidelines, and a reference architecture to enable cross-organizational collaboration and integration through interoperability mechanisms. They defined different interoperability profiles for the following domains: collaborative product development, networked collaborative product development, electronic procurement, and product portfolio management. Furthermore, they also established five levels of interoperability: 0-*Performed* (ad-hoc collaboration and integration), 1-*Modelled* (collaboration and integration is done in the same manner each time but not automated), 2-*Integrated* (the collaboration process is formally defined and documented), 3-*Interoperable* (enterprise models and process are dynamically adapted to new changes) and 4-*Optimizing* (interoperability is continuously measured and improved through the application of new technologies and frameworks).

The OIM (Organizational Interoperability Maturity model) represents another attempt mainly addressing organizational/business needs. It is focused on assessing the quality of interconnection of systems within the same organization. Likewise the LISI model, it does not offer us neither an estimation of the effort nor the specification of the technical details to improve or reach another level of interoperability. That is why it should be complemented with other type of model, e.g., the LISI model. In regard to the description of the defined levels, five different levels can be found: 0-*Independent* (manual integration of data and information), 1-*Ad-hoc* (some guidelines or frameworks to enable interoperability are envisioned), 2-*Collaborative* (guidelines and frameworks

are in place with a clear distinction of roles and responsibilities enabling cross-organizational interoperability), 3-*Integrated* (a common understanding of goals and business services) and 4-*Unified* (interoperability is properly deployed enabling exchange across organizations).

The Interoperability assessment methodology [27] was also developed after the LISI model in the context of military services. It includes nine components that are either a "yes/no" response and a mathematical equation. Leite further defined "degrees of interconnection" which included the availability, connectivity, understanding, interpretation, utility, feedback, and execution." [21].

The layered interoperability score (i-Score) [28] is a method to measure interoperability of any type. It is used in the operational process context, and it is based on assessing the current data architecture. The Service Measurement Index [29] also includes a metric for cloud service providers in terms of interoperability. Others based on the review in [30] such as the government interoperability maturity matrix, the business interoperability quotient measurement model [31], represent other efforts to measure interoperability in different contexts such as quality of service in service oriented architectures.

Other interoperability maturity models and reference architectures include the "NATO C3 Technical Architecture Reference Model for Interoperability", the ISO 11354-1:2011 "Advanced automation technologies and their applications. Part 1: Framework for enterprise interoperability" or the "European Interoperability Framework". These attempts collect similar characteristics to the previous models but oriented to different domains like defense, manufacturing or public services [32].

As interoperability definitions have been adapted to different domains and by several institutions, it is also possible to find recent reviews on interoperability assessment models [33]. In this work, authors conducted a systematic review to monitor and analyze a total of 38 INAS (Interoperability Assessment) models with the main objective of classifying the type of assessment, metrics, and interoperability barriers. They compared 22 assessment models, only those providing real case studies or illustrative examples. One of the main conclusions is that metrics are mainly qualitative (subjective) or those that are quantitative are merely a ratio (real/expected). In [34] the same authors defined the requirements for an enterprise interoperability assessment method while in [35] authors defined an ontology of automate the assessment of interoperability (process and metrics). In [36] authors also review the status of companies to support the development of cyber-physical systems (CPS) focusing mainly in the operational aspects of such systems and reaching as main conclusion that general interoperability frameworks do not cover all the needs of CPS development and operation. In [37] authors review the Federated Interoperability Framework (FIF) for the aerospace and defense sector to check how it has evolved in the last 20 years and how it can be customized for Product Lifecycle Management (PLM) interoperability. Although the main *-ilities* of a system are considered, they state that supporting virtual manufacturing will play a critical role in the PLM of the future.

In the context of this work and considering a toolchain as a set of orchestrated tools (services), the service-oriented architecture (SOA) assessment methods may be also relevant. In [38] authors proposed a method to evaluate the feasibility of a SOA architecture based on evaluating a set of requirements (functional and non-functional) in some qualitative scale. This type of approach is similar to others in that time like the Service Measurement Index (discontinued) but used in some previous works to evaluate cloud-based CRM (Customer Relationship Management) solutions [39]. In [40] authors make a systematic review of SOA maturity models (SOAMMs) in which interoperability is considered as another organizational and technical dimension. In [41] authors present the MeFSOAR framework based on Welke's SOA maturity model [42] with the aim of providing knowledge to developers in the adoption of SOA architectures. More interestingly, authors present in [43] a SOA architecture of Industry 4.0 and small companies collaboration considering the building blocks of an architecture to communicate tools in the industry. In many of these works, the concept of interoperability is mainly considered and evaluated as a non-functional requirement with a qualitative scale.

As a summary, some of the original maturity/capability models, see Table 2, have been largely studied, defined, reviewed [5,21], and extended to cover both dimensions: technical and organizational and to measure the level of interoperability within an organization or a business according to some scale. Starting from the LISI model, every defined model describes a set of categories (levels) to express and define the interoperability capabilities. In some of them, a set of guidelines and a description of the technical details to reach a new level of interoperability are also defined. In terms of evaluating the interoperability of a toolchain, it is possible to reuse concepts of both types, although the LISI and LCIM models fit better to the purpose of tool connection evaluation. However, these models do not reflect how to evolve from one level to another (apart from checklists). More recent studies considering the architectural style or different domains still follow a similar approach: interoperability as a major non-functional requirement that is evaluated on a qualitative scale at a high-level of abstraction.

2.3. Effort estimation models for software-based systems

In the previous section, a brief review of the interoperability capability models was presented. The ability of providing a toolchain within the Systems Engineering process as a set of interoperable services can be seen as a composed/orchestrated software system that should be estimated following a software effort estimation model.

Effort estimation models for software systems can be classified into three main categories: empirical, heuristic, and analytical. Models [44] like COCOMO, SLIM or Function Points have been historically applied to this end apart from others based on expert-judgement or analogy (expert opinion and experience). However, most of these models [45] generally fail to express the underlying complexity of building software unless formal methods and tools are used to generate the software code.

The development methodology also plays a key role in the estimation of effort. In the case of Agile methodologies, authors in [46] conducted a systematic review surveying 25 practitioners and extracting as conclusions that planning poker, expert judgement, use case points and story points are the main subjective estimation techniques while the "Mean

Table 2
Summary of the main Interoperability Maturity Models.

Interoperability maturity model	Scope	Levels	Attributes/ dimensions (technical aspects)	Effort estimation method
LISI	*Technical*	*0-Isolated* *1-Connected* *2-Functional* *3-Domain* *4-Enterprise*	Yes, PAID attributes	Not covered
LCIM	Technical	*0-No interoperability* *1-Technical* *2-Syntactical* *3-Semantic* *4-Pragmatic* *5-Dynamic* *6-Conceptual*	Yes, as an extension of the LISI model	Not covered
EIMM	Technical	*0-Performed* *1-Modelled* *2-Integrated* *3-Interoperable* *4-Optimizing*	Yes, a reference architecture based on services	Not covered
OIM	*Business*	*0-Independent* *1-Ad-hoc* *2-Collaborative* *3-Integrated* *4 Unified*	Not covered	Not covered

magnitude of relative error" MMRE metric is used to validate the estimation. Team skills and development experience remain a critical aspect to define a proper task size and get an accurate estimation. In [47], authors also conducted a systematic review and a survey to 53 practitioners in 7 different countries. They included different estimation techniques such as bucket system, dot voting, expert judgement, planning poker, team estimation game, swimlane sizing, use case points and story points. As main conclusions, story points are being used to mainly estimate of the entire effort necessary to develop the software system while more than 90% of the respondents were using planning poker and expert judgement as the most common estimation techniques (only one was using COCOMO). Regarding methods for effort estimation, in [48] authors propose a differential evolution algorithms to improve the adjust the parameters of models like COCOMO and COCOMO II. They validated the approach with two datasets from the Promise repository and using as a cost function the MMRE metric. In [49] authors propose an extension to the COCOMO model including cost drivers and other metrics to update the base model equation. Again, they validated the approach based on expert judgement and the MRRE metric.

In general, the conclusions about software estimation remain the same: "*no single technique is best for all situations*" [50] and, other factors like organizational (development methodology), experience or application type (e.g. web or mobile) are being considered as critical aspects to provide an accurate estimation of software development efforts. In fact, this is a research area in which effort can be estimated when using automated engineering techniques, but the estimation of human developed software is still open and subjected to variables such as experience (organizational and team), expert judgement, technology and application domain.

In the case of interoperability, the situation is like general software development. Some models can be found [51,52] to estimate the effort of implementing automated techniques (e.g. Model-Driven Development). Others are domain-specific, e.g. buildings and construction [53], have conducted surveys to know what is the cost of no interoperability [54] mainly for the stages of operation and maintenance.

Overall, the estimation of software development costs is not an easy task. There are different types of models but "*no one size fits all*" and specific considerations must be added to provide an accurate estimation. In the context of interoperability effort estimation, some studies can be found but focusing on the operation and maintenance of systems. However, it is not clear how to tailor an existing estimation model to calculate the effort of implementing interoperability mechanisms within a toolchain for Systems Engineering. A hybrid method considering expert judgement as well as a weighted set of parameters seems a feasible approach to estimate the effort of interoperability implementation within a toolchain. On the other hand, some of the existing parametric models are based on the analysis of previous projects and more subjective methods like Function Points seems to not fully consider the intrinsic complexity of a toolchain (many dependencies between tools in a non-linear way). Since the implementation of toolchain interoperability is not so common as other types of software, a method based on combining parameters established and scaled by the expert judgment is considered a feasible approach.

3. Definition of a method to evaluate and measure toolchain interoperability

3.1. Interoperability evaluation method

To establish a method to evaluate the interoperability within a toolchain, the next definitions must be considered:

A toolchain, T, is a set of software applications used to implement a technical process (e.g. verification) within the system development lifecycle. The set of software tools are usually executed in a linear manner, being the result of some tool the input of the following one. However, as it has been already stated [55], the design and execution of a toolchain is not always linear and, in most of cases, it creates a network of interconnections among tools, as an example Table 1 includes examples of different use case scenarios. It is important to remark that not all tools require bi-directional communication with all other tools. In the scope of this work, tools may only need to access artifact content, but other operations/methods offered through an interface are not considered.

Each tool, t_i, is a software application designed to perform some specific tasks within the development lifecycle. A tool t_i manages and produces a kind of work product (e.g. requirements, test cases, logical models, physical models, source code, etc.) that may be used by another tool t_j.

The definition, configuration, and integration of tools within a toolchain can be done manually or automatically within an ALM (Application Lifecycle Management) or PLM (Product Lifecycle Management). More specifically, the tool integration strategy can follow different enterprise application integration patterns such as an integration bus, a common database, a plugin architecture, or a queue of messages. The integration usually covers the communication level, but the syntax and semantics of message payloads must be implemented in each tool.

Given these initial definitions, an interoperability evaluation method for a toolchain in the context of cyber-physical systems development can be defined as follows:

A toolchain interoperability evaluation method establishes the required (input/output) connections between the different tools providing a quantitative value of interoperability at different levels of abstraction, see Table 3.

Building on these levels of interoperability, an interoperability evaluation method creates an implicit matrix, M, see Table 4, establishing two values, v_{ij} and v_{ji}, for each pair of tools ti and t_j. where v_{ij} and v_{ji} are not necessarily the same, v_{ij}, refers to connectivity from the tool t_i to the tool t_j, while v_{ji}, expresses the connectivity from t_j to the tool t_i. Interpreting the values on this matrix, it is possible to extract two main aggregated values.

The full interoperability value of a toolchain T is equals to $3*n^2$, being *n* the number of tools within the toolchain.

The current interoperability value of a toolchain T is equals to the sum of elements within the matrix v_{ij}

As consequence of this evaluation, the current level of technical interoperability within a toolchain can be calculated and compared to a target value providing a method to establish the degree of interoperability.

Table 3
Interpretation of the levels of technical interoperability based on the LCIM model.

Level	Value	Description
No exchange	0	There is no need of interoperability between the pair of tools (t_i, t_j).
Communication	1	There is a need of interoperability between the pair of tools (t_i, t_j) and, t_i and t_j share the same communication protocol.
Syntax	2	There is a need of interoperability between the pair of tools (t_i, t_j) and, t_i and t_j share the same data format.
Semantics	3	There is a need of interoperability between the pair of tools (t_i, t_j) and, t_i and t_j share the same meta-model.

Table 4
Evaluation of interoperability levels between tools within a toolchain.

Tool	t_0	…	t_i	…	t_j	…	t_n
t_0	0		v_{0i}		v_{0j}		v_{0n}
…		0					
t_i	v_{i0}		0		v_{ij}		v_{in}
…				0			
t_j	v_{j0}		v_{ji}		0		v_{jn}
…						0	
t_n	v_{nj}		v_{ni}		v_{nj}		0

3.2. Interoperability evolution: an effort estimation method

On the other hand, the mere calculation of a degree of interoperability only offers a picture of the current situation and, potentially, a target objective. However, the transition from one degree of interoperability to another may require an estimation of the implementation efforts.

As in other effort estimation software models, see Section 2.3, three different techniques can be used: empirical, heuristic, and analytical. In this case and with the aim of providing a practical method to estimate the effort of implementing an interoperability mechanism, an analytical technique considering both analogy/expert judgement and some parameters is defined. This effort estimation function is based on three main aspects: the existence of standards, libraries, and the development experience. Furthermore, it is also necessary to estimate which is the effort of implementing a new communication protocol or a new data and schema processor.

There is always a baseline effort, *be*, due to tasks related to configuration and integration. Furthermore, as Table 5 shows, there is a specific effort *e*, associated with the real implementation. This effort *e* depends on some scale factor associated with the type of implementation and the method to produce software, the expert judgement and experience strongly affects this factor. In the case of interoperability, three basic types of implementations and efforts are defined:

1 Implementation using an existing library based on standards with a scale factor k_i.
2 Implementation using an existing library not based on standards with a scale factor of k_j.
3 A new customized implementation with a scale factor of k_l.

The scale factor is expressed through a value k_m and it may depend on different factors such as organizational, human and technological aspects among others. In order to establish these values of k_m, a sequence of ascending numbers K $\{k_1,\ldots,k_i, k_j, k_l,\ldots, k_n\}$ can be used following some type of progression (e.g. linear, geometric, exponential, etc.).

For instance, assuming the *be* is 1 person/month (PM), *e* is 2 person/month (PM) and K $\{1, 2, 4\}$ (based on our experience developing tool connectors), the effort of developing an interoperable mechanism based on a standard library would be 3 PM, if there is no standard behind 5 PMs and, finally, if everything must be implemented from scratch, the estimated effort would be 9 PMs.

3.3. Example of application

To illustrate the presented approach, let's consider a toolchain, T, comprising 3 services (t1, t2 and t3) with the following description:

Table 5
Effort estimation depending on the type of implementation.

Type	Library and standard	Existing library but no standard	Custom implementation
Effort	*be* k_1e	*be* k_2e	*be* k_3e

t1 uses as communication protocol a standard protocol (p1, HTTP) generating a work product (wp1, "requirement") under the format (f1, XML) and standard (s1, ReqIF).
t2 uses as communication protocol a standard protocol (p1, HTTP) generating a work product (wp2, "logical model") under the format (f1, XML) and standard (s2, SysML). The tool t2 already has an implementation to connect to t1 at a syntax level (communication through HTTP and XML as data format) which implies that (t2, t1) 2 in Table 6.
t3 uses as communication protocol a standard protocol (p2, FTP) generating a work product (wp3, "physical model") under the format (f2, JSON) without any concrete semantics.

Furthermore, the target toolchain environment requires the following connections:

Tool t1 must be able to consume the information generated by tool t3 at a semantic level: (t1, t3) 3. The implementation requires then access to the tool through a new protocol (p2), processing data through a new data format (f2) and understanding data through a new schema (native).
Tool t2 must be able to consume the information generated by t1 and t3 at a semantic level: (t2, t1) 3 and (t2, t3) 3.
Tool t3, must also be able to semantically consume the information generated by t1: (t3, t1) 3.

Given this setting, the current degree of interoperability is calculated in Table 6, there is a need to increase this degree to interoperability to reach a target interoperability level as presented in Table 7.

According to Table 8 and following the proposed method, the total estimated effort can be calculated as follows: 6*(be k_1e) (be k_3e). More specifically, each pair (ti, tj) represents an estimated implementation effort:

0 (-) if no implementation is required
be k_1e: if there is an existing library based on standards.
be k_3e: if it is a new customized implementation.

4. Case study: a toolchain for digital hardware design

4.1. Context

The H2020-AHTOOLs project aims for "*digitalisation and automation solutions for the European industry, which will close the gaps that hinder the IT/OT integration by introducing new technologies in an open source platform for the design and run-time engineering of IoT and System of Systems. The project will provide engineering processes, integration platform, tools and tool chains for the cost-efficient development of digitalisation, connectivity and automation system solutions in various fields of application.*"

The core element of the project is the Arrowhead framework, previously implemented in other research projects and now part of the Eclipse ecosystem. From a conceptual perspective, the framework provides a complete platform of cross-cutting aspects like security or interoperability [56] to orchestrate services for different purposes under a bus integration pattern. Initially, the framework was focused on providing an execution platform for IoT applications [57,58] comprising sensors and software services. However, the H2020-AHTOOLs project focuses on applying and extending the platform to support the

Table 6
Example of an initial interoperability evaluation (value 2).

Tool	t1	t2	t3
t1	–	–	–
t2	2	–	–
t3	–	–	–

Table 7
Example of a target interoperability evaluation (value 12).

Tool	t1	t2	t3
t1	–	–	3
t2	3	–	3
t3	3	–	0

engineering phase of cyber-physical systems unifying the communication layer to orchestrate different tools within a toolchain. In this context, the Use Case-3 (UC-3) "Integration of electronic design automation tools with product lifecycle tools" has been proposed to make use of the Arrowhead framework supporting the engineering process of digital hardware.

More specifically, the UC-3 aims at providing means for improving the reuse of physical hardware models covering the abstraction, selection, representation, and customization of system artifacts for the whole development lifecycle. The reuse of any system artifact goes beyond the mere discovery of a potential reuse, and it must focus on evaluating what and how a system artifact can be reused (requirements, analytical models, descriptive models, test cases, etc.). To do so, quality also plays a role since it is assumed that high-quality system artifacts may help to improve the reusability factor of a system artifact. Furthermore, in this use case, there is another major objective focusing on the improvement of traceability to be able to automatically keep traces [59] from the very early stage of development to the final release of a complex product.

Both functions, system reuse and traceability, require access to different tools and artifacts to build an engineering knowledge graph that can be exploited to provide suggestions of traces or a set of artifacts to be reused. In other terms, from an architectural perspective, the implementation of the use case is based on the creation of a shared database in which any artifact is represented under the same data schema. Besides, there are two functions (reuse and trace) implemented on top of this database and results of these functions are populated in different tools, e.g. traces in the logical modelling tool or linked artifacts in the hardware design tool.

To do so and based on previous experiences [60,61], connectors between different tools may at least represent any artifact of the toolchain in this common database. As a common data schema, the SRL (System Representation Language) [16] is used and implemented within the CAKE ("Computer Aided Knowledge Environment") library (core component of the KnowledgeManager, Traceability Studio and Verification Studio tools).

Fig. 8 shows the process of accessing, representing, and indexing data coming from different tools and artifacts. The reuse and traceability functions are then implemented on top of the "Systems Assets Store" providing input for other tools under a specific data format.

The implementation of this use case [60,61], considers three types of partners: 1) academic (the Carlos III University of Madrid), to define the methodology and help in the implementation of the interoperability mechanisms, 2) a tool vendor (The Reuse Company, a software company with more than 20 years of experience providing engineering solutions), to implement the connectors and to provide the reuse and trace functions and 3) an end-user (ULMA Embedded Solutions, a hardware design company), to validate the solution of the provided artifacts. In this research work, we look for providing and showing the designed method to estimate the effort of this implementation based on the experience of the technology provider in the development of interoperable connectors. We evaluate the approach from the perspective of implementation efforts, but we do not include end-user validation.

The specific engineering process of this use case covers different technical engineering processes and engineering methods (supported by different techniques and tools) creating the next toolchain (following the standard ISO 15288:2015 "Systems and software engineering — System

Table 9
Engineering process, methods, techniques and tools for the UC-3.

Technical engineering process	Engineering method	Techniques	Tools
System Requirements Definition	Requirements Engineering	Text-based requirements	IBM Doors, Requirements Authoring Tool (RAT)
Architecture Definition	Logical modelling	Diagramming with SysML/UML	IBM Rhapsody
Design definition	Physical modeling	Diagramming	Altium designer
Implementation	Simulation	Programming, simulation configuration	Altium designer
Verification & Validation (Measurement process)	Quality management	Quality metrics	Verification Studio (VS)
Information Management	Knowledge engineering	Ontologies	Knowledge Manager (KM)
Information Management	Knowledge management	Traceability discovery	Knowledge Manager

Table 8
Example of implementation effort.

Tool/Element	p1 (HTTP)	p2 (FTP)	f1 (XML)	f2 (JSON)	s1 (ReqIF)	s2 (SysML)	s3 (Native)
t1	–	$be + k_3e$	0	$be + k_3e$	–	–	$be + k_3e$
t2	–	–	–	–	$be + k_3e$	–	–
t3	$be + k_3e$	0-	$be + k_3e$	0	$be + k_3e$	–	–

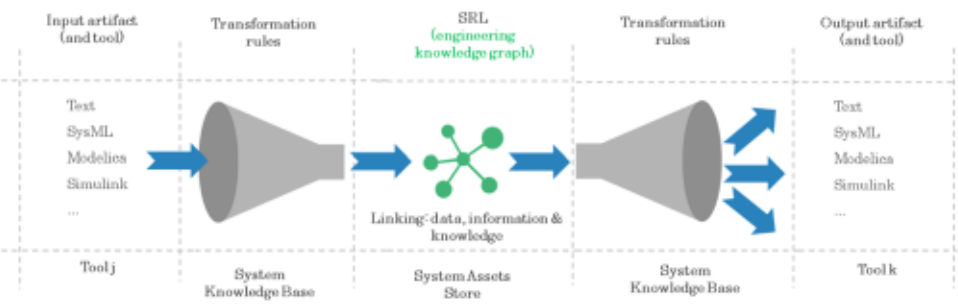

Fig. 8. Access and representation of data to provide reuse and traceability capabilities within the UC-3.

life cycle processes") (Table 9):

Fig. 2 shows the interconnection of these engineering processes as established in Table 10. As it has been introduced, the engineering process of a complex product should follow a liner approach (no feedback between processes and predefined inputs and outputs) (Figs. 3–7). However, a realistic toolchain comprises the interaction of different disciplines, people and tools creating an underlying graph of connections (Fig. 9).

In general, there are three tool providers: IBM, The Reuse Company and Altium. Most of them provide standardized ways of accessing (files and services) and consuming work products data and operations. However, the interpretation of standards (such as SysML or ReqIf) may differ from one tool to another and, in most of cases, the tools also manage more relevant information that is not exposed being critical for processes such as traceability or quality management.

In this context, it is possible to first define interoperability characteristics of the tools to then make a brief evaluation of the required integrations. To do so, the following table summarizes the data model and the access/communication models.

To reach the two major objectives of this use case (traceability management and reuse), the following integrations must be done (see Table 11) where ***x***: represents a connector or integration that already exists but it is not based in standards in both senses (communication and data model) and ***R*** a new standard-based connector is required to properly implement the use case.

4.2. Description and challenges

Building on the previous context, some research questions emerge:

1 *What is the degree of interoperability of the current toolchain?*
2 *What is the degree of interoperability of the target toolchain?*
3 *Can we estimate the effort of transitioning from the current to the target toolchain?*
4 *What is the role and the gain of using the Arrowhead framework?*

Table 10
Tool and system artifacts description for the UC-3.

Tool	System artifact	Data model	Access/ Communication model
IBM Doors	Requirement	Native ReqIF OSLC RM	Native API Native database File
IBM Rhapsody	Logical models	Native SysML/UML	Native API Native database File
RAT	Requirement	Native OSLC RM (Requirements Management) ReqIF	Native API WSDL-based Service OSLC/Rest Service File
Verification Studio	Quality metric	Native OSLC KM [16]	Native API WSDL-based Service OSLC/Rest Service File
KM (through Traceability Studio)	Trace	Native OSLC KM	Native API WSDL-based Service OSLC/Rest Service File
KM	Ontology, vocabulary, etc.	Native OSLC KM SKOS OWL	Native API WSDL-based Service OSLC/Rest Service File
Altium designer	Hardware model	Native	Native API WSDL-based Service

Applying the defined evaluation model, the first step is to establish the communication protocols, formats, and data models to be used by each tool.

IBM Doors: file, XML, ReqIF metamodel.
IBM Rhapsody: file, XML, SysML v1 metamodel.
RAT: HTTP, JSON, SRL ("System Representation Language") metamodel [59,15].
VS: HTTP, JSON, SRL metamodel.
KM: HTTP, JSON, SRL metamodel.
Altium designer: file, XML, native metamodel.

According to these dependencies, Table 12 shows the initial interoperability evaluation. The degree of interoperability has a value of 18. More specifically, there is a value of 3 between IBM Rhapsody and IBM Doors since there is already a connection that allows us to exchange and interpret requirements in IBM Rhapsody. In the same manner, RAT can access, interpret, and write requirements in IBM Doors while VS can also access, interpret requirements and write back quality checking values. On the other hand, other connections between tools are set to 0 (-) since no connection exists between tools or it is not required for this use case.

In the same manner, the target toolchain environment will have to accomplish integrations to consume and produce system artifacts generated in the different engineering methods and tools. The degree of interoperability of this target environment will have a value of 33. More specifically, the effort is focused on consuming information from Altium in IBM Rhapsody, VS and KM. In the same manner, Altium shall be able to consume the information generated in IBM Doors, VS and KM.

These two evaluations (initial and target) help us to understand which is the degree of interoperability of both environments providing information for tool vendors and engineering management processes (e.g. systems engineers).

4.3. Results and evaluation

Table 14 presents an estimation of the effort depending on the type of implementation required, see Table 4, at the three levels of technical interoperability (communication, syntax and semantics).

In the case of IBM Rhapsody, there are implementations of all required protocols and data formats but an implementation to understand the Altium metamodel is required. RAT and VS already connect to IBM Doors but some implementation is necessary to process ReqIF (a XML serialization of requirements). VS and KM also need to understand the Altium native metamodel and, in particular, KM requires the understanding of the SysML metamodel. The results produced by the different tools must be interpreted and displayed in Altium, so a connector to SRL is required. In conclusion, the effort relies on processing a data format (XML or JSON) and understanding the underlying metamodels: SysML, SRL, ReqIF or the Altium native metamodel.

According to Table 14, the total estimated effort can be calculated as follows: 5*(be k_1e) 1*(be k_2e) 3*(be k_3e) 9be 5k_1e k_2e 3k_3e.

As in the initial example, if we assume and assign the next values to the be 1PM, e 2PM and K {1, 2, 4}, the total effort of increasing the degree of interoperability would be: (9 1) (5 1*2) (2 2) (3 4*2) 47 PMs that is bit higher but more analytical and accurate that the initial estimation (48 PMs) of the main technological partners (Carlos III University of Madrid and The Reuse Company) in this use case.

Given these results, the proposed method to evaluate the toolchain interoperability provides us with a technique to understand and estimate the effort of transitioning from one toolchain environment to another. It is also possible to establish the current and target degrees of interoperability, see Tables 12 and 13. Furthermore, this method can also provide us with insights on the effort of implementing an interoperable toolchain, see Table 14.

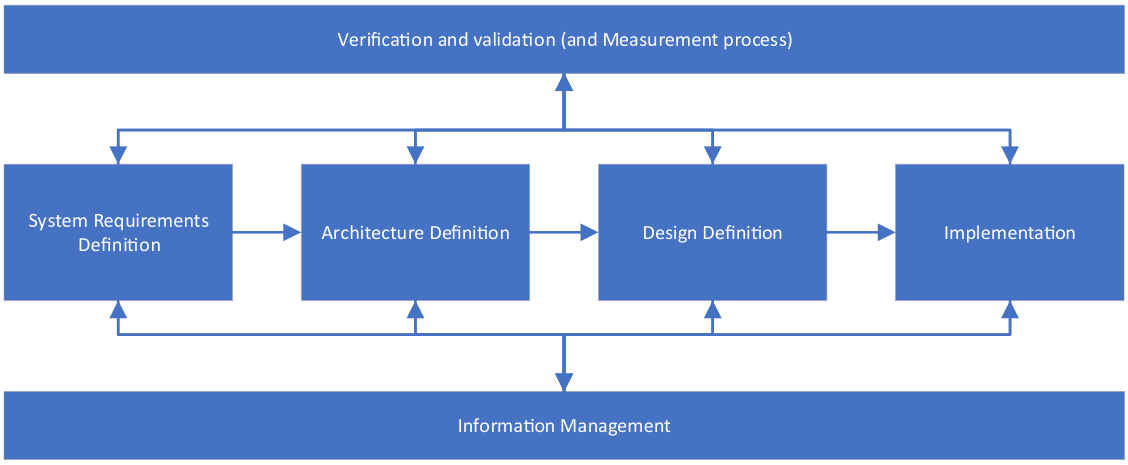

Fig. 9. Technical engineering and management processes for UC-3.

In terms of validation, the proposed method and the implementation of this use case brings the following conclusions:

Conceptual validation: the proposed method brings together two already agreed methods to evaluate the interoperability and to estimate software effort. More specifically, the proposed method takes the interoperability levels established in existing maturity models (the LCIM model) and defines an estimation of effort based on existing software estimation models (expert judgement, analogy, and analytical methods). This estimation of effort may contain some bias since it is only based on the opinion and experience of only one software company. However, this factor is mitigated since this company already has in the market tools integrating data and information from around other 20 third-party tools.

Technical validation: the implementation of this use case raised some technical challenges in terms of processing different types of information and data. However, once the implementation was verified and validated by the end-user, we had to demonstrate to what extent this approach was better than the initial environment. The application of the proposed method helped us to demonstrate both the feasibility, accuracy, and simplicity of the method and the added value of providing an interoperable environment for the Systems Engineering process. However, in terms of accuracy, the proposed method should be validated in more scenarios defining more complex toolchains (hundreds of tools collaborating in different organizations).

Finally, the application of the Arrowhead framework will help to decrease the effort of implementing the communication level, automating the exchange of system artifacts, and enabling the consumption of operations, but the interpretation of the different metamodels must be implemented in any case.

Table 11
Status of interoperability and integration between tools within the UC-3.

Source/Target tool	IBM Doors	IBM Rhapsody	RAT	VS	KM	Altium designer
IBM Doors		x	x	x	R	R
IBM Rhapsody	x		x, R			R
RAT	x	x, R		x	x	
Verification Studio	x	x, R	x		x	R
Traceability Studio	x	R	x	x	x	R
Knowledge Manager	R	R	x	x		R
Altium designer	R	R	R	R	R	

4.4. Research limitations

From a conceptual point of view, the proposed evaluation model is built on previous and validated definitions, especially the LCIM model. The proposed approach adapts and extends these definitions to provide a realistic and specific estimation of the degree of interoperability in an engineering environment.

However, as it has been reviewed in the state-of-the-art section, software effort estimation models are not accurate and depend on many factors that require a specific analysis of the problem. Here, authors propose the use of some analytical and parametrized estimation based on an increasing sequence of scale factors (K) that must be adapted to the background of an organization (e.g. software development experience, existence of libraries, complexity of standards metamodels, etc.).

The presented case study helps understand how to apply the evaluation method in a real toolchain environment. However, the interpretation of the method may change for a more complex product comprising several organizations (which internal toolchains could be unknown) and different engineering methods using specific protocols, data formats and metamodels.

Table 12
Initial interoperability evaluation within the UC-3 (value 18).

Tool	IBM Doors	IBM Rhapsody	RAT	VS	KM	Altium designer
IBM Doors	–	3	3	3	–	–
IBM Rhapsody	3	–	–	–	–	–
RAT	3	–	–	–	–	–
VS	3	–	–	–	–	–
KM	–	–	–	–	–	–
Altium designer	–	–	–	–	–	–

Table 13
Target interoperability evaluation within the UC-3 (value 33).

Tool	IBM Doors	IBM Rhapsody	RAT	VS	KM	Altium designer
IBM Doors	–	–	–	–	–	–
IBM Rhapsody	3	–	–	–	–	3
RAT	3	–	–	–	–	–
VS	3	–	–	–	–	3
KM	3	3	–	–	–	3
Altium designer	3	–	–	3	3	–

Table 14
Estimated effort to shift the degree of interoperability within UC-3.

Tool/Element	File	HTTP	XML	JSON	ReqIF	SysML	SRL	Native Altium
IBM Doors								
IBM Rhapsody								$be\ \ k_3e$
RAT					$be\ \ k_1e$			
VS					$be\ \ k_1e$			$be\ \ k_3e$
KM						$be\ \ k_1e$		$be\ \ k_3e$
Altium Designer	$be\ \ k_1e$			$be\ \ k_1e$			$be\ \ k_2e$	

5. Conclusions and future work

The digital transformation is gaining momentum in the engineering process of software-intensive systems. The need of delivering complex engineering products in a timely and cost-effective manner is now more relevant than ever. The engineering process of these systems comprises multiple disciplines and engineering domains that must be encompassed within a development lifecycle.

Digitalization of engineering through automation and simulation as means to implement a virtual version of a system also requires standardization and interoperability between the tools used by the different engineering disciplines. In this context, the ability of exchanging data, information and the knowledge embedded in the systems artifacts is a cornerstone to boost collaboration between organizations, people, and tools. To establish a degree of interoperability and to understand the effort to transition from one toolchain environment to another, it is necessary to measure the degree of interoperability between tools and provide a fine-grained view of the required tool integrations at different levels of abstraction (communication, syntax, and semantics).

Future research directions may include an improvement of the effort estimation function based on previous experiences developing interoperability adapters and extension of the evaluation to consider other types of application integration patterns. Furthermore, new toolchain interoperability evaluations in other sectors such as automotive or aerospace may also demonstrate the value of the model to estimate engineering efforts to implement executable technical processes through interoperable toolchains.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

Data availability

No data was used for the research described in the article.

Acknowledgments

The work leading to these results has received funding from the H2020-ECSEL Joint Undertaking (JU) under grant agreement No 826452-"Arrowhead Tools for Engineering of Digitalisation Solutions" and from specific national programs and/or funding authorities. Funding for APC: Universidad Carlos III de Madrid (Read & Publish Agreement CRUE-CSIC 2023).

References

[1] M.V. Cengarle, Characteristics, capabilities, potential applications of Cyber-Physical Systems: a preliminary analysis, CyPhERS (2013) (Accessed 20 January 2023). [Online]. Available: http://www.cyphers.eu/sites/default/files/D2.1.pdf.
[2] J. Lee, B. Bagheri, H.-A. Kao, A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manuf. Lett. 3 (2015) 18–23, https://doi.org/10.1016/j.mfglet.2014.12.001.
[3] M. Törngren and U. Sellgren, "Complexity challenges in development of cyber-physical systems," in *Principles of Modeling*, vol. 10760, M. Lohstroh, P. Derler, and M. Sirjani, Eds. Cham: Springer International Publishing, 2018, pp. 478–503. doi: 10.1007/978-3-319-95246-8_27.
[4] A. Mavridou, et al., The ten lockheed martin cyber-physical challenges: formalized, analyzed, and explained, in: 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, Aug. 2020, pp. 300–310, https://doi.org/10.1109/RE48521.2020.00040.
[5] Jerome Hugues, Joseph Yankel, John Hudak, Anton Hristozov, TwinOps: Digital Twins Meets Devops, Carnegie Mellon University, 2022, https://doi.org/10.1184/R1/19184915.
[6] A.M. Grubb, M. Chechik, Formal reasoning for analyzing goal models that evolve over time, Requir. Eng. 26 (3) (2021) 423–457, https://doi.org/10.1007/s00766-021-00350-8.
[7] A. Amjad, F. Azam, M.W. Anwar, W.H. Butt, M. Rashid, Event-driven process chain for modeling and verification of business requirements–a systematic literature review, IEEE Access 6 (2018) 9027–9048, https://doi.org/10.1109/ACCESS.2018.2791666.
[8] D. Seal, The System Engineering 'V' - is it still relevant in the digital age?, in: Presented at the Global Product Data Interoperability Summit, 2018 [Online]. Available: https://gpdisonline.com/wp-content/uploads/2018/09/Boeing-DanielSeal-The_System_-Engineering_V_Is_It_Still_Relevant_In_the_Digital_Age-MBSE-Open.pdf?pdf Boeing-DanielSeal-The_System_-Engineering_V_Is_It_Still_Relevant_In_the_Digital_Age-MBSE-Open.
[9] ISO/CD 10303-243: MoSSEC: Modelling and Simulation information in a Collaborative Systems Engineering Context (Under Development), International Organization for Standardization, Geneva, CH, Standard, 2020.
[10] Research Roadmaps 2019-2020, Systems Engineering Research Center (SERC, Hoboken, New York, US, 2020. Technical report[Online]. Available: https://serc uarc.org/wp-content/uploads/2020/01/ROADMAPS_2.3.pdf.
[11] Systems Modeling Application Programming Interface (API) and Services, Version 2.0, Object Management Group, Milford, MA, USA, Standard, 2023 [Online]. Available: https://github.com/Systems-Modeling/SysML-v2-API-Services.
[12] P. Micouin, Model-based Systems Engineering: Fundamentals and Methods, Hoboken, NJ, USA: iSTE; Wiley, London, UK, 2014.
[13] A. Madni, C. Madni, S. Lucero, Leveraging digital twin technology in model-based systems engineering, Systems 7 (1) (2019) 7, https://doi.org/10.3390/systems7010007.
[14] J. Lu, J. Wang, D. Chen, J. Wang, M. Torngren, A service-oriented tool-chain for model-based systems engineering of aero-engines, IEEE Access 6 (2018) 50443–50458, https://doi.org/10.1109/ACCESS.2018.2868055.
[15] J.M. Alvarez-Rodríguez, R. Mendieta, J.L. De la Vara, A. Fraga, J. Llorens, Enabling system artefact exchange and selection through a Linked Data layer, J. UCS JUCS In-Press (2018) 1–24.
[16] J.M. Alvarez-Rodríguez, J. Llorens, M. Alejandres, J.M. Fuentes, OSLC-KM: a knowledge management specification for OSLC-based resources, INCOSE Int. Symp. 25 (1) (2015) 16–34, https://doi.org/10.1002/j.2334-5837.2015.00046.x.
[17] M. Biehl, J. El-khoury, F. Loiret, M. Törngren, On the modeling and generation of service-oriented tool chains, Softw. Syst. Model. 13 (2) (2014) 461–480, https://doi.org/10.1007/s10270-012-0275-7.
[18] INCOSE, Systems Engineering Vision 2020, INCOSE, Technical INCOSE-TP-2004-004-02, 2004. Accessed: Nov. 24, 2013. [Online]. Available: http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.
[19] J. Lu, G. Wang, M. Törngren, Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain, IEEE Syst. J. 14 (1) (2020) 1297–1308, https://doi.org/10.1109/JSYST.2019.2911418.
[20] F. Asplund, M. Törngren, The discourse on tool integration beyond technology, a literature survey, J. Syst. Softw. 106 (2015) 117–131, https://doi.org/10.1016/j.jss.2015.04.082.
[21] R. Rezaei, T.K. Chiew, S.P. Lee, Z. Shams Aliee, Interoperability evaluation models: a systematic review, Comput. Ind. 65 (1) (2014) 1–23, https://doi.org/10.1016/j.compind.2013.09.001.
[22] C. A. W. Group and others, Levels of Information Systems Interoperability (LISI), US DoD, 1998.
[23] A. Tolk, J.A. Muguira, The levels of conceptual interoperability model, in: Proceedings of the 2003 Fall Simulation Interoperability Workshop 7, 2003, pp. 1–11.
[24] W. WANG, A. TOLK, W. WANG, The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S, 2009, https://doi.org/10.48550/ARXIV.0908.0191.
[25] M. Robkin, S. Weininger, B. Preciado, J. Goldman, Levels of conceptual interoperability model for healthcare framework for safe medical device interoperability, in: 2015 IEEE Symposium on Product Compliance Engineering

(ISPCE), Chicago, IL, 2015, pp. 1–8, https://doi.org/10.1109/ISPCE.2015.7138703.
[26] P. ATHENA, "ATHENA Interoperability Framework (AIF)." https://sintef-9012.github.io/athena-interoperability-framework/methodology/eimm.html (Accessed 20 February 2023).
[27] M.J. Leite, Interoperability Assessment, PRC INC ARLINGTON VA, 1998.
[28] T. Ford, J. Colombi, S. Graham, D. Jacques, The Interoperability Score, Air Force Inst Of Tech Wright, Patterson AFB OH, 2007.
[29] J. Siegel, J. Perdue, Cloud services measures for global use: the service measurement index (SMI), in: 2012 Annual SRII Global Conference, San Jose, CA, USA, 2012, pp. 411–415, https://doi.org/10.1109/SRII.2012.51.
[30] C. bCNRS, Towards a comparative analysis of interoperability assessment approaches for collaborative enterprise systems, in: Transdisciplinary Engineering: Crossing Boundaries: Proceedings of the 23rd ISPE Inc. International Conference on Transdisciplinary Engineering October 3–7, 2016 4, 2016, p. 45.
[31] A. Zutshi, A. Grilo, R. Jardim-Goncalves, The business interoperability quotient measurement model, Comput. Ind. 63 (5) (2012) 389–404.
[32] A. Kouroubali, D.G. Katehakis, The new European interoperability framework as a facilitator of digital transformation for citizen empowerment, J. Biomed. Inform. 94 (2019), 103166.
[33] G. da Silva Serapião Leal, W. Guédria, H. Panetto, Interoperability assessment: a systematic literature review, Comput. Ind. 106 (Apr. 2019) 111–132, https://doi.org/10.1016/j.compind.2019.01.002.
[34] G.S.S. Leal, W. Guédria, H. Panetto, Enterprise interoperability assessment: a requirements engineering approach, Int. J. Comput. Integr. Manuf. 33 (3) (2020) 265–286, https://doi.org/10.1080/0951192X.2020.1736636.
[35] G. da Silva Serapião Leal, W. Guédria, H. Panetto, A semi-automated system for interoperability assessment: an ontology-based approach, Enterp. Inf. Syst. 14 (3) (2020) 308–333, https://doi.org/10.1080/17517575.2019.1678767.
[36] G. Weichhart, H. Panetto, A. Molina, Interoperability in the cyber-physical manufacturing enterprise, Annu. Rev. Control 51 (2021) 346–356, https://doi.org/10.1016/j.arcontrol.2021.03.006.
[37] D. Tchoffa, N. Figay, P. Ghodous, H. Panetto, A. El Mhamedi, Alignment of the product lifecycle management federated interoperability framework with internet of things and virtual manufacturing, Comput. Ind. 130 (2021), 103466, https://doi.org/10.1016/j.compind.2021.103466.
[38] N. Ghoddosi, R.J. Rabelo, A Method for Evaluating the Feasibility of SOA Projects, 2015, pp. 1–6, https://doi.org/10.1109/ICSSSM.2015.7170188.
[39] R. Colomo-Palacios, J.M. Alvaraz-Rodríguez, Semantic representation and computation of cloud-based Customer Relationship Management solutions, in: Third Workshop on Industrial and Business Applications of Semantic Technologies and Knowledge-based information systems (INBAST 2014), 2014 [Online]. Available: http://www.onthemove-conferences.org/index.php/inbast-14.
[40] S. Pulparambil, Y. Baghdadi, C. Salinesi, A methodical framework for service oriented architecture adoption: guidelines, building blocks, and method fragments, Inf. Softw. Technol. 132 (2021), 106487, https://doi.org/10.1016/j.infsof.2020.106487.
[41] S. Pulparambil, Y. Baghdadi, Service oriented architecture maturity models: a systematic literature review, Comput. Stand. Interfaces 61 (2019) 65–76, https://doi.org/10.1016/j.csi.2018.05.001.
[42] R. Welke, R. Hirschheim, A. Schwarz, Service-oriented architecture maturity, Computer 44 (2) (2011) 61–67, https://doi.org/10.1109/MC.2011.56.
[43] Z. Liu, et al., The architectural design and implementation of a digital platform for Industry 4.0 SME collaboration, Comput. Ind. 138 (2022), 103623, https://doi.org/10.1016/j.compind.2022.103623.
[44] C.F. Kemerer, An empirical validation of software cost estimation models, Commun. ACM 30 (5) (1987) 416–429, https://doi.org/10.1145/22899.22906.
[45] D.V. Ferens, The conundrum of software estimation models, in: Proceedings of the IEEE 1998 National Aerospace and Electronics Conference. NAECON1998. Celebrating 50 Years (Cat. No.98CH36185), Dayton, OH, USA, 1998, pp. 320–328, https://doi.org/10.1109/NAECON.1998.710133.
[46] M. Usman, E. Mendes, F. Weidt, R. Britto, Effort Estimation in Agile Software Development: A Systematic Literature Review, 2014, pp. 82–91, https://doi.org/10.1145/2639490.2639503.
[47] R.C. Sandeep, M. Sánchez-Gordón, R. Colomo-Palacios, and M. Kristiansen, "Effort estimation in agile software development: a exploratory study of practitioners' perspective," in *Lean and Agile Software Development*, vol. 438, A. Przybyłek, A. Jarzębowicz, I. Luković, and Y. Y. Ng, Eds. Cham: Springer International Publishing, 2022, pp. 136–149. doi: 10.1007/978-3-030-94238-0_8.
[48] P. Singal, A.C. Kumari, P. Sharma, Estimation of software development effort: a differential evolution approach, Procedia Comput. Sci. 167 (2020) 2643–2652, https://doi.org/10.1016/j.procs.2020.03.343.
[49] J.A. Khan, S.U.R. Khan, T.A. Khan, I.U.R. Khan, An amplified COCOMO-II based cost estimation model in global software development context, IEEE Access 9 (2021) 88602–88620, https://doi.org/10.1109/ACCESS.2021.3089870.
[50] B. Boehm, C. Abts, S. Chulani, Software development cost estimation approaches — a survey, Ann. Softw. Eng. 10 (1/4) (2000) 177–205, https://doi.org/10.1023/A:1018991717352.
[51] P. Mork, W. Melo, S. Dutcher, C. Curtis, M. Scroggs, Cost estimation for model-driven interoperability: a canonical data modeling approach, in: 2014 14th International Conference on Quality Software, Allen, TX, USA, 2014, pp. 145–153, https://doi.org/10.1109/QSIC.2014.51.
[52] D. Gürdür, F. Asplund, J. El-khoury, Measuring tool chain interoperability in cyber-physical systems, in: 2016 11th System of Systems Engineering Conference (SoSE), 2016, pp. 1–4.
[53] D. Forgues, I. Iordanova, F. Valdivesio, S. Staub-French, Rethinking the cost estimating process through 5D BIM: a case study, in: Construction Research Congress 2012, West Lafayette, Indiana, United States, 2012, pp. 778–786, https://doi.org/10.1061/9780784412329.079.
[54] M.P. Gallaher, A.C. O'Connor, J.L. Dettbarn Jr., L.T. Gilday, Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry, National Institute of Standards and Technology, NIST GCR 04-867, 2004, https://doi.org/10.6028/NIST.GCR.04-867.
[55] vol. 9416 S. Imran, M. Buchheit, B. Hollunder, U. Schreier, Tool chains in agile ALM environments: a short introduction, in: I. Ciuciu, H. Panetto, C. Debruyne, A. Aubry, P. Bollen, R. Valencia-García, A. Mishra, A. Fensel, F. Ferri (Eds.), On the Move to Meaningful Internet Systems: OTM 2015 Workshops, Cham, Springer International Publishing, 2015, pp. 371–380, https://doi.org/10.1007/978-3-319-26138-6_40. vol. 9416.
[56] P. Varga, et al., Making system of systems interoperable – The core components of the arrowhead framework, J. Netw. Comput. Appl. 81 (2017) 85–95, https://doi.org/10.1016/j.jnca.2016.08.028.
[57] H. Derhamy, J. Eliasson, J. Delsing, System of system composition based on decentralized service-oriented architecture, IEEE Syst. J. 13 (4) (2019) 3675–3686, https://doi.org/10.1109/JSYST.2019.2894649.
[58] C. Paniagua, J. Eliasson, J. Delsing, Efficient device-to-device service invocation using arrowhead orchestration, IEEE Internet Things J. 7 (1) (2020) 429–439, https://doi.org/10.1109/JIOT.2019.2952697.
[59] J.M. Alvarez-Rodríguez, R. Mendieta, V. Moreno, M.Á. Sánchez-Puebla, J. Llorens, Semantic recovery of traceability links between system artifacts, Int. J. Softw. Eng. Knowl. Eng. IJSEKE 30 (11) (2020) 1–28, https://doi.org/10.1142/S0218194020011967.
[60] E. Cibrián, R. Mendieta, J.M.Á. Rodríguez, J. Lloréns, Towards the reuse of physical models within the development life-cycle: a case study of Simulink models, in: 2022 IEEE/IFIP Network Operations and Management Symposium, NOMS 2022, Budapest, Hungary, April 25-29, 2022, 2022, pp. 1–6, https://doi.org/10.1109/NOMS54207.2022.9789840.
[61] E. Cibrián, J.M.Á. Rodríguez, R. Mendieta, J. Lloréns, Discovering traces between textual requirements and logical models in the functional design of Printed Circuit Boards, in: 5th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS 2022, Coventry, United Kingdom, May 24-26, 2022, 2022, pp. 1–6, https://doi.org/10.1109/ICPS51978.2022.9816910.