Towards a methodology of system manifestation features-based pre-embodiment design

Pourtalebi Hendehkhaleh, Shahab; Horvath, Imre

DOI
10.1080/09544828.2016.1141183

Publication date
2016

Document Version
Final published version

Published in
Journal of Engineering Design

Citation (APA)
Pourtalebi Hendehkhaleh, S., & Horvath, I. (2016). Towards a methodology of system manifestation features-based pre-embodiment design. *Journal of Engineering Design*, *27*(4-6), 232-268. https://doi.org/10.1080/09544828.2016.1141183

Journal of Engineering Design

ISSN: 0954-4828 (Print) 1466-1837 (Online) Journal homepage: http://www.tandfonline.com/loi/cjen20

Towards a methodology of system manifestation features-based pre-embodiment design

Shahab Pourtalebi & Imre Horváth

To cite this article: Shahab Pourtalebi & Imre Horváth (2016) Towards a methodology of system manifestation features-based pre-embodiment design, Journal of Engineering Design, 27:4-6, 232-268, DOI: 10.1080/09544828.2016.1141183

To link to this article: https://doi.org/10.1080/09544828.2016.1141183

JOURNAL OF ENGINEERING DESIGN, 2016
VOL. 27, NOS. 4–6, 232–268
http://dx.doi.org/10.1080/09544828.2016.1141183

Taylor & Francis
Taylor & Francis Group

Towards a methodology of system manifestation features-based pre-embodiment design

Shahab Pourtalebi and Imre Horváth

Faculty of Industrial Design Engineering, Delft University of Technology, Delft, The Netherlands

ABSTRACT

The main assumption is that complicated systems, such as cyber-physical systems (CPSs), can be modelled by specific compositions of system manifestation features (SMFs). SMFs are regarded as architectural domains of a system having significance from an operational viewpoint. As system modelling entities, SMFs represent both physical and computing transformations of domains. Based on mereo-operandi theory (MOT), a computational framework for using SMFs in pre-embodiment design of CPSs is proposed. MOT offers a theoretical platform for concurrent modelling of architectural elements and their operations. The traditional 'application feature technology' has been generalised in order to provide a methodological basis. The computational formalisation captures state transitions and input/output streams, in addition to spatiotemporal, physical and/or computational attributes of domains. Domain transformations are represented by flows of operation (FoOs) that consist of time-sequenced and logically constrained sets of units of operations (UoOs), and processed by various computational methods as procedures. The domains of SMFs are aggregated into a feasible architecture, and their UoOs are combined into FoOs. An application case is used to explain the concepts and to demonstrate feasibility of the proposed approach. Further research will focus on implementation of an SMFs-based pre-embodiment design system and testing its feasibility and usability with designers of CPSs.

ARTICLE HISTORY
Received 7 April 2015
Revised 6 January 2016
Accepted 8 January 2016

KEYWORDS
Cyber-physical systems; pre-embodiment design; system manifestation features; mereo-operandi theory; architecture and operation knowledge frames

1. Introduction

1.1. Objectives of the presented work

Cyber-physical computing creates a high-level synergy among physical (analogue and digital hardware), middleware (control and application software) and cyber (media and knowledge) constituents of *cyber-physical systems* (CPSs) (Kurtoglu, Tumer, and Jensen 2010; Horváth and Gerritsen 2012). In combination with their broad functional spectrum, the heterogeneity of the constituents and the intricacy of their interactions pose many challenges for CPS developers (Bar-Yam 2004) (De Micheli 1996). Designers may come

CONTACT Shahab Pourtalebi s.pourtalebihendehkhaleh@tudelft.nl

across some of these challenges already in the *pre-embodiment design phase*, where off-the-shelf or custom-made heterogeneous constituents should be built together and their interoperability should be achieved (Abdul-Ghafour et al. 2014). This is typical for that family of CPSs which is referred to as complicated technical systems (Horváth and Gerritsen 2013). Efficient conceptualisation of this kind of systems needs multi-disciplinary tools and unified approaches (Sztipanovits 2012). As a novel approach, the idea of using *system-level features* for modelling these systems has been introduced (Liang et al. 2012). Nevertheless, system-level feature-based support of structural configuration and behavioural simulation of CPSs is still in its infancy even as a research issue. With the aim of keeping pace with the pioneers, our research efforts have been devoted to the development of the concept of *system manifestation features* (SMFs) for pre-embodiment design.

Our guiding assumptions were: (i) complicated technical systems, such as CPSs, can be modelled by using SMFs, and (ii) an SMFs-based system model can describe both the architecture and the operation of the system (Horváth and Pourtalebi 2015). The assumptions made by Tiihonen, Soininen, and Sulonen (1996) have also been considered, namely that system feature-based configurable systems are pre-designed systems, which: (i) need to be adapted according to the customers' requirements for each order or installation, (ii) have an operation-implied or a pre-designed system architecture, (iii) consist of a large number of off-the-shelf or only pre-designed components, and (iv) are adapted by a systematic configuration process or the producer, or by using the embedded (built-in) customisation options by the end-users (Da Silveira, Borenstein, and Fogliatto 2001). As a starting point for our research we presumed that: (i) the requirements of the engineering part of pre-embodiment design has been completed, (ii) the concepts concerning the overall architecture and operations of the system have been devised, and (iii) sufficient number of pre-designed SMFs are available as components or modules. The latter obviously minimises the need for creative or innovative design actions (Baldwin and Clark 2006).

In other research, the tools with such functionality were referred to as product configurators, or product customisers (Stjepandić et al. 2015). Though it is often named differently, e.g. as configuration design (Tiihonen et al. 1998), the challenges and specific issues of pre-embodiment design are well acknowledged in the literature (Hotz et al. 2014). The major novelty of our work is that we imposed a massively physical view on pre-embodiment design and modelling. We have realised that by doing so we can develop a tool that may go well beyond the affordances of traditional logic-based modelling tools or languages. This is important in a comparison with the currently commercially available system modelling and simulation tools. It is well known that some of them, e.g. Modelica®, use an analytic formulation (representation) of the functionality and behaviour of the physical components, and derive the architecture accordingly. Other tools, e.g. SysML®, support system-level representation by applying high-level logical abstractions and simplifications. A mathematical representation of components and systems is applied by MATLAB Simulink®, Ptolemy II® and LabVIEW®. The reason is that these systems have been developed based on traditional system modelling and simulation paradigms, which have made huge progress in terms of capturing the functional aspects, but did not intend to make the physicality of the system fully tangible from an architecting point of view.

The intention of our work was to model the *architecture and operations* of CPSs on a system level as well as on multiple component levels in their physicality. In addition,

the target tool was supposed to support both the development of SMF entities and their composition into a system. The development process should be guided by novel architectural and operational principles (Oreizy et al. 1999). Thus, the theoretical framework offered by the *mereo-operandi theory* (MOT) was used to guide the conceptualisation of SMFs and to transfer the theoretical concepts into a computational approach (Pourtalebi, Horváth, and Opiyo 2014b). In this paper, we present the results of the two recently completed phases of the research, which concentrated on the *theoretical fundamentals* of SMFs and the *computational framework* (Figure 1). The next subsection gives a concise overview of MOT, whose details can be found in Horváth and Pourtalebi (2015). Our first results indicate that SMFs can be used for both system configuration (Pourtalebi, Horváth, and Opiyo 2014a) and embedded customisation (Eckert, Clarkson, and Zanker 2004).

1.2. A brief overview of MOT

MOT has been developed to underpin transdisciplinary modelling and pre-embodiment design of complex heterogeneous systems. It includes numerous concepts to uniformly represent various parts of complex systems and to facilitate integration of inherently diverse hardware (HW), software (SW) and cyberware (CW) constituents (Gerritsen and Horváth 2015). It aims to represent not only the architectural relations and composition of these constituents as *aggregated ware* (AW), but also their operations and interactions as a synergetic whole. The four assumptions of MOT are: (i) a complex system is a heterogeneous composition of HW, SW and CW constituents and their aggregations, (ii) the components of a complex system should be captured on multiple levels, (iii) it is sufficient to consider the architectural and operational characteristics and their interdependencies for a non-exhaustive description (modelling) of systems and components, and (iv) a complex system can be de-aggregated into a finite number of semantic entities (Horváth and Pourtalebi 2015). It differentiates between homogeneous entities (constituents) and heterogeneous entities (components). MOT facilitates the representation and manipulation of

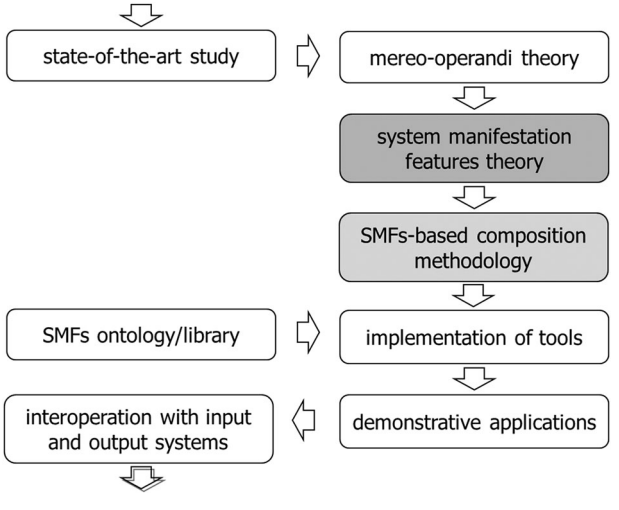

Figure 1. Implementation phases of SMFs-based support of pre-embodiment design of CPSs.

architectural domains and flows of operations on various aggregation (granularity) levels, as well as the simulation of system operations according to different application scenarios.

Bjørner and Eir (2010) raised the issue of compositionality of: (i) simple entities, (ii) operations, (iii) events, and (iv) behaviours. They used mereology to study composite entities and to address the compositionality problem of software constituents. These principles seem to be extendable to SMFs too. Composability can be used as a measure of the degree to which SMFs can be assembled in various combinations to satisfy specific user requirements (Doboli et al. 2014). The theory of SMF is seen as a complement of MOT towards a practical methodology. This complementing theory: (i) fosters the development of specific procedures, methods and algorithms for a unified handling of system-level features, (ii) interlinks their architectural and operational aspects, (iii) enables the development of SMFs ontologies and libraries, and (iv) facilitates the management of interaction of complex systems with the stakeholders (users) and the surrounding environment (Mcgrenere and Ho 2000). In order to form a system in an aggregative manner, SMFs should have proper interfaces that support meeting composability conditions.

MOT applies the principles of spatiotemporal mereotopology for identifying and representing the architectural elements of system and their mutual relationships (Asher and Vieu 1995; Kim, Yang, and Kim 2008). The physical entities are interpreted as proper parts of the whole, and defined as *domains* that lend themselves to certain operations (Kim and Yang 2008). Domains are abstracted physical manifestations of components and constituents of various aggregation levels. Though domains are defined as spatial entities, mereotopological abstraction makes it possible to describe them without considering their actual metrics, morphology and materialisation (Borgo, Guarino, and Masolo 1996). Thereupon, MOT is able to describe all HW, SW and CW constituents notwithstanding their inherent differences. MOT introduces two families of relationships, namely part-of and connected-to/with relations (also referred to as containment and connectivity relationships). Figure 2 introduces the possible architectural relations between components and constituents of a complex system. *Part-of* relations allow a logically hierarchical, multi-level de-aggregation of systems and a uniform handling of components and constituents (without differentiating between constituents). *Connected-to/with* relations

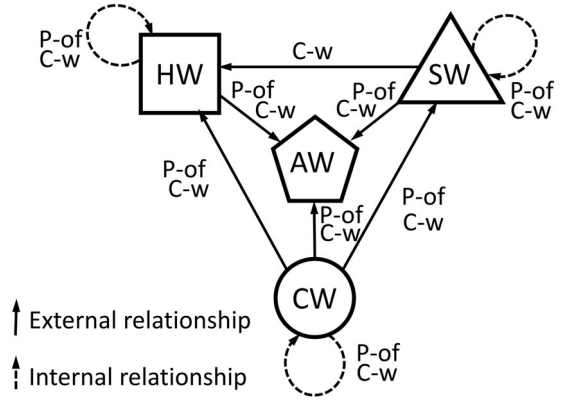

Figure 2. Internal and external relationships within and among constituents and components.

can describe neighbourhood, remoteness and functional relationships between components and constituents.

By imposing a purely physical view, MOT assumes that existence is an intrinsic (an indispensable) operation of all entities included in a system. When existent, each domain performs either physical or computational transformations, or concurrently both. Operations are described in terms of four generic aspects: (i) underlying principles of physical, computational, informing and/or synergic effects, (ii) morphological characteristics of the domains, (iii) conduct of the elementary operations of the domains, and (iv) flows of operations of the lower-level domains. Since HW, SW and CW have different morphologies and operate according to different principles, the concept of unit of operation (UoO) has been introduced in MOT. It provides a uniform scheme for representation of HW, SW and CW constituents, as well as for AW components. In fact, UoO creates consistency with respect to the material, energy and information streams, and the state transitions of the corresponding architectural domains.

We hypothesised that the concept of features and the idea of feature-based design can be generalised, adapted and reused as a methodological analogy in the context of multidisciplinary systems. Accordingly, the core principles of the classical feature theory have been imported into a specific system-level feature theory (SLFT) and have been extended with new principles that are entailed by MOT and the contexts of CPSs. Based on the considerations discussed below, the proposed SLFT is actually a complement of MOT. SMFs capture and integrate architectural and operational views, and synthesise the architectural and operational parameters on different levels. SMFs give the opportunity for system designers to focus their attention on those aspects of the system that are the most relevant and important at a given moment.

Throughout the paper, an application case is used to explain the concepts and to demonstrate the feasibility of the proposed approach. Actually, this is a commercialised

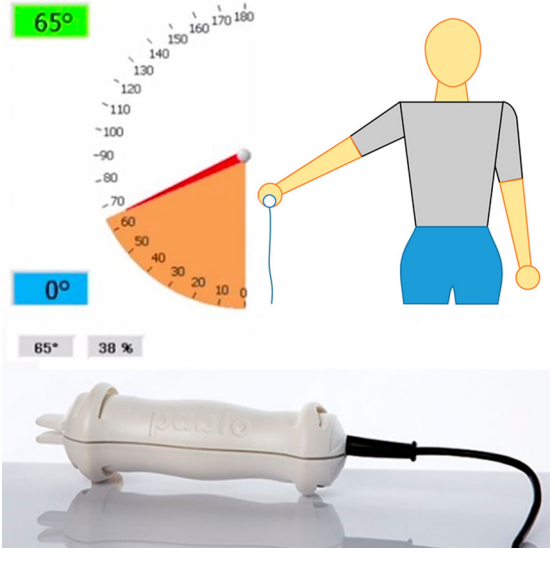

Figure 3. Side lift with the Pablo rehabilitation/training system.

product, marketed under the fantasy name Pablo®[1] (Figure 3). Developed for motor rehabilitation of shoulder, lower arm and hand functions, and for repetitive training, this basic system (device) can be connected to a PC via a USB interface. Including a hand loop, an element of the device is a sensor grip that makes it possible to perform exercises and contains strength and movement sensors to measure the forces of various (cylinder and pinch) grip patterns, and stretching and bending of the human hand. The device is able to measure the mobility range of the arm and the different characteristics of movements. The handle can determine which position the hand is in and when the exercise is finished. This simple system will be used to clarify the concepts introduced in association with both architecture and operation modelling.

2. Defining the architecture of SMFs

2.1. From traditional feature technology to SMFs

A *feature* is a notable property as well as a distinctive (prominent) characteristic of things (artefacts, systems, processes and phenomena) that sets them apart from similar items for humans or smart systems (Shah 1991). In the classical theory of engineering product features, a feature is something that has significance in a given context (Xie and Ma 2015). For example, *form features* capture regions of both prismatic and freeform industrial shapes that have a meaning for humans or smart system agents from a semantic point of view (e.g. chamfer, hole, rounding, sharp edge, depression, protrusion) (Bidarra and Bronsvoort 2000). This idea has been extended to applications where geometry, structure, materialisation, implementation, etc. imply some meaning in the context of a particular application (e.g. as is done by manufacturing features, assembly features, piping features, etc.) (Syaimak and Axinte 2009). They are often called *part features* or *application features*. Though it was addressed by many researchers, transformation between feature spaces and conversion of feature definitions between various applications remained a partially solvable problem due to the need for handling intrinsic meanings (Altidor et al. 2009). On the other hand, remarkable progress has been achieved in terms of parameterised and constraints-based feature-based design algorithms. In the last two decades, attention was paid to ontology-based feature definition (Horváth et al. 1998), information sharing (Kim, Manley, and Yang 2006), mapping (Tessier and Wang 2013) and feature conversion (Kim et al. 2009). Feature-based design has had more influence on the detailed design and planning activities of product development processes, than on the conceptual or pre-embodiment design activities (Brunetti and Grimm 2005). This comes from the dependence of meaning on the details of manifestation (Chen et al. 2006).

Theoretically, two categories of system-level features can be identified. The first category includes features that distinguish a particular type of system from other types of systems. We have called them *paradigmatic system features* (PSFs). They are generic and abstract, and do not have explicit relations with the engineering realisation/implementation of systems. The advantage of introducing paradigmatic features is to avoid mistakes in overall characterisation of systems based on specific details and attributes (the often-cited 'blind man and the elephant' situation). Examples of PSFs of CPSs are as follows: (i) open system boundary, (ii) run-time built architecture, (iii) non-linear behaviour, (iv) dynamic network management, (v) decentralised control structure, (vi) context-dependent

services, (vii) smart observing and reasoning, (viii) proactive operation scenarios, (ix) functional autonomy, (x) multi-scale composition, (xi) self-evolving capabilities and (xii) transdisciplinary framework. The second category includes features that exist only in a particular implementation of a system. Above, we referred to them as SMFs. They are concurrently genotypic and phenotypic in nature. In our context, genotypic means that SMFs can be used to determine the overall composition or makeup of a system, or a component thereof. Phenotypic means that they can also be used to determine specific physical and visual traits of a system/component. Technically, these are made possible by multi-level and multi-aspect parameterisation of SMFs. PSFs and SMFs represent two different levels of abstractions, but they are conceptually not independent from each other. For instance, having wheels is a paradigmatic feature of a car, but this implies a manifestation feature that the diameter of the wheels is, say, 18 inches. Considering these facts, we could draw demarcation lines between paradigmatic features and manifestation features.

SMFs can be uniquely characterised by spatiotemporal, architectural and operational attributes. Just like traditional part features (e.g. form∼, design∼ and manufacturing∼), SMFs can be represented by computational constructs containing a structured set of interrelated parameters, constraints, values and semantic annotations. A particular SMF may occur multiple times in a system in different instantiations (with different sets of values assigned to its parameters and annotations). SMFs may also appear on multiple aggregation levels. In the language of topology it means that various rougher and finer domain topologies can be interpreted over the overall domain of a system. For instance, a digital processor simultaneously represents an architectural domain and an operation performer. In a higher resolution, the domain of the processor is an aggregate of a number of interrelated digital elements, and the operation of the processor is the sum of the specific operations of these elements. This issue is called 'multi-granularity of SMFs' and will be revisited in the next subsections from different perspectives.

SMFs are seen as prefabricated 'components' that can be used in designing both the architecture and the operation of a system. From a computational point of view, an SMF is virtual entity parametrically representing a specific domain and the related operations. However, instead of working with two separate models for architecture and operations, the proposed approach uses SMFs as dual-aspect building blocks. SMFs capture the necessary and sufficient pieces of information about architecture and operation of a component. Using SMFs supports fast configuration and adaptation of the architecture and operations of systems. It also facilitates the reuse of components in pre-embodiment design of different system variants. Nevertheless, aggregation of SMFs brings the challenge of *feature interfacing*, which is a well-known issue for the traditional feature theory, into the forefront. In addition, SMFs should be made interoperable in order to realise the functional objectives of the designed systems (Demoly et al. 2011). As mentioned in the literature, features trigger ideas concerning the way of organising knowledge that facilitates computing and inferences (Ostrosi and Ferney 2007; Hotz et al. 2014). Below, we utilise this potential from both architectural aspects and operational aspects.

2.2. Multi-level interpretation of SMFs from an architectural point of view

In this subsection, we concentrate on the interpretation of SMFs from an architectural point of view, while in Section 2.3 we present examples for defining their prescriptive

contents. In Section 2.4, knowledge frames will be proposed to enable specifying SMFs on various aggregation levels. The starting point is that the information structures specifying a particular SMF should include: (i) description of domains of different physical extents and aggregation levels, (ii) capturing and representing operations happening on different aggregation levels, and (iii) interfacing between different architecture and operation levels. Thus, the computational representation should be flexible enough to allow both architecture and operation aggregation and interfacing even if the architectural and operational connections of SMFs are not standardised, or do not obey set contracts.

As interpreted by MOT, an architectural/operational domain is a spatiotemporal abstraction of the manifestation of an AW component, which may include HW, SW and CW constituents. Accordingly, the domains of an SMF, the parts of the domains and the relationships among them are described by spatiotemporal mereotopological operations introduced in Horváth and Pourtalebi (2015). As architectural relationships within a domain and between domains, *part-of* and *connected-to/with* relationships have been introduced. For *part-of* relationship of physical aggregates, specific graphical and symbolic representations have been applied. For instance, the graphical relation symbol A◇⟶B means that *B* is a part of *A*. Figure 4 shows various possible *part-of* relationships. Figure 4 should be interpreted in a physical view, as enforced by MOT. That is, the aggregate of all first-level components is the system itself. The aggregates of second-level components are the first-level components, and so forth-down to the levels of constituents. The various graphical entity symbols indicate the type of architectural entities on the respective levels of aggregation. The hierarchical relations between them indicate only logical relationships, while the physical relationships are captured by the abovementioned mereotopological expressions.

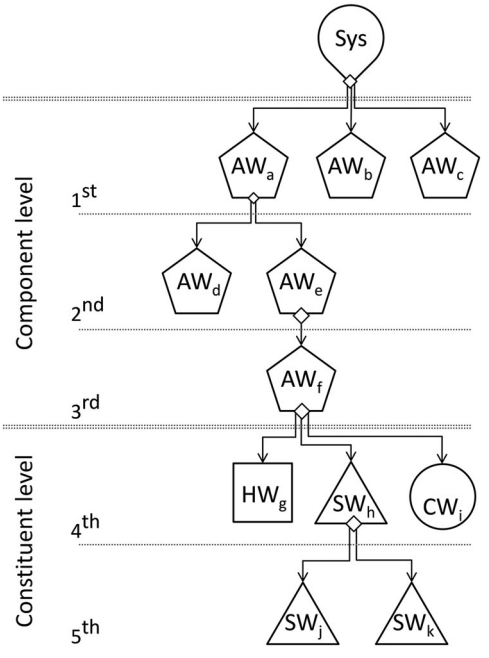

Figure 4. Various *part-of* relations on multiple levels of aggregation (de-aggregation).

Description of the hardware constituents of an SMF is more straightforward than that of the software constituents. The reason is that an SW constituent exists in two alternative forms: (i) in source (language) code (when it is a kind of white box, whose content is directly readable and changeable by programmers), and (ii) in compiled or binary code (when it is a kind of black box, whose machine instruction content is to be processed by processors). These two forms are discrete and need different treatment from both architectural and operational points of view. Notwithstanding that both forms should be considered in design, below we deal with only the run-time form of SW constituents. The possible containment relationships of components and constituents can be formally described by the following mereotopological expressions (relations):

- $PSysCom^{1}{}_{Ab}$ describes that component AW_b is a level 1 component and as such is directly *part-of* the system.
- $PCom^{1}{}_{Aa}\ Com^{2}{}_{Ad}$ represents that component AW_d is a level 2 component and is a *part-of* AW_a, which is a component of aggregation level 1.
- $PCom^{3}{}_{Af}\ Con^{4}{}_{Hg}$ means constituent HW_g is a level 4 constituent and is a *part-of* component AW_f, which is a component of aggregation level 3.
- $PCon^{4}{}_{Sh}\ Con^{5}{}_{Sj}$ means constituent SW_j is a level 5 constituent and is a *part-of* constituent SW_h, which is a constituent of aggregation level 4.

Other architectural relationships between entities are described by either *connected-to* or *connected-with* relations. These play a dual role as: (i) spatial neighbourhood relations and as (ii) carrier architectural relations of some operations. In the latter role, they can be: (a) one-directional relations (expressing dependence of B on A) or (b) bi-directional relation (expressing mutual operational interdependence of A and B). A *connected-to* relation is represented by an A●⟶B arrow, and a *connected-with* relation is represented by an A●⟷●B arrow. In the case of the latter (operation carrying) relationships, *proximity* and *directness* have also been considered. The graphical scheme, shown in Figure 5, represents two plates (H_{fm} and H_{fc}) of a capacitor that are remotely *connected-with* each other, and there is one connector (H_{ft}), which is in a direct connection with one of the plates, H_{fc}. In Figure 5, the type of *connected-to/with* relations is indicated by letters placed below the connection line (i.e. *r* stands for 'remote', and *d* is for 'direct' connection). The symbols above the arrows (at the ends of the connection line) indicate the enabler of the remote or direct connection of the components (i.e. g_1 and g_2 stand for electromagnetic fields, and e_1 for electric current).

The mereotopological relations of the entities H_{fm}, H_{fc} and H_{ft} are described symbolically as $r.CCon_{Hfm}{}^{g1}\ wCon_{Hfc}{}^{g2}$, and $d.CCon_{Hft}{}^{e1}\ tCon_{Hfc}$. Note that the codes can indicate different kinds of dependencies and connectivity (i.e. existence, processing, heat producing, energy consumption, information transmission, etc.).

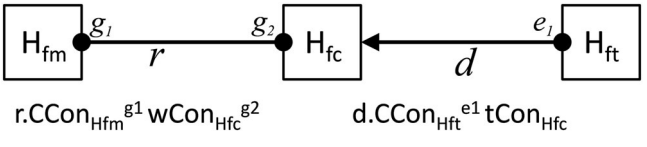

Figure 5. Examples of *connected-to/connected-with* relations.

2.3. Capturing architectural entities and relationships of SMFs

Below, we follow a 'from-concrete-to-abstract' reasoning in explaining how architectural entities and relationships of SMFs can be derived and represented. In other words, we will: (i) start out from a concrete existing system of moderate complexity, (ii) identify its SMFs, (iii) take one particular SMF, (iv) identify its components and constituents, (v) describe their spatiotemporal mereotopological relations, (vi) specify the operational procedures and computational methods, and (vii) define the computational knowledge frame of the SMF.

Let us revisit the Pablo system now. From an architectural point of view, this system as a whole has a physical part and a computational part. Both can be modelled as an aggregate of SMFs. Among others, the physical part includes an accelerometer sensor, which consists of an MEMS (micro-electromechanical system) detector, H_f, and an ASIC (application-specific integrated circuit) converter, A_s. These two parts will be used as representative SMFs in our further investigations. The lower-level architectural components and constituents of the MEMS detector and the ASIC converter, as well as their relationships, are shown in the lower part of Figure 6.

On a lower level (of de-aggregation), the elements of the accelerometer capture the acceleration of movements, and convert it into a capacitance change. As shown by its schematic architecture in Figure 7, the MEMS detector (H_f) implements this functionality

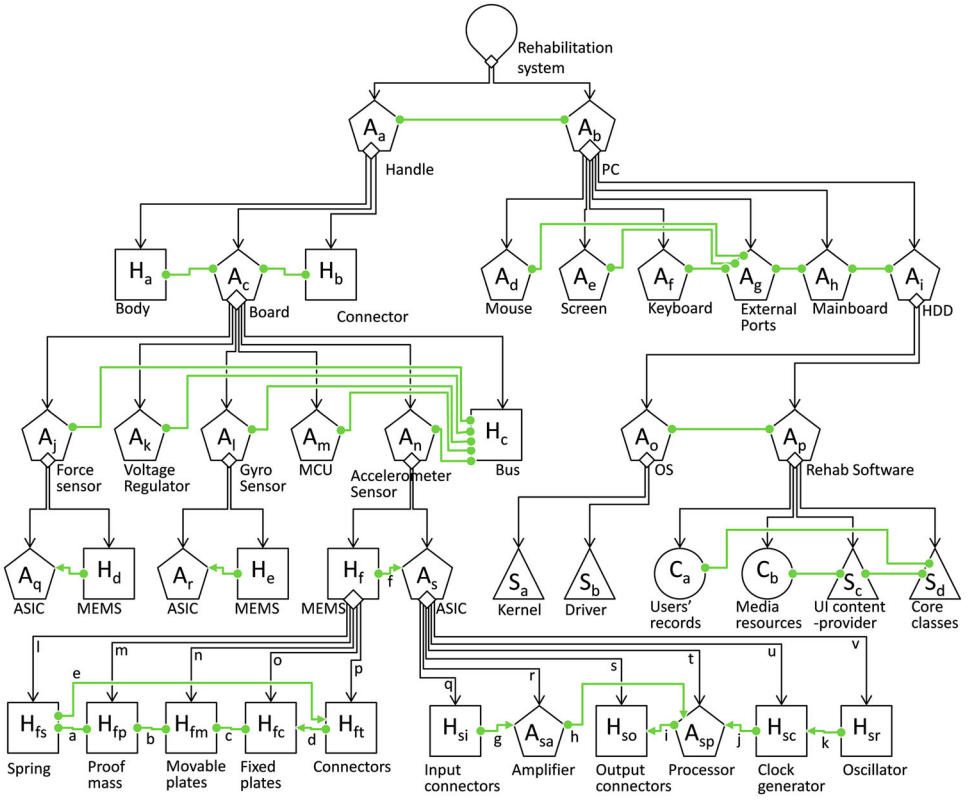

Figure 6. The overall architecture and the architectural elements of the Pablo system.

through a number of hardware constituents. The *proof mass* (H_{fp}) moves up and down due to acceleration. The *springs* (H_{fs}) at two ends of H_{fp} brake and reinstate it. There are two kinds of *fixed plates* (H_{fc}) connected to $V+$ and $V-$. Together with the *movable plates* (H_{fm}), which are connected to and move with H_{fp}, these plates create capacitors. The exerted acceleration changes the distance between each H_{fm} and H_{fc} plates, hence changes the electric current. The change of the electric current is proportional to the acceleration.

These *part-of* (containment) relations can be formally specified as given below. These symbolic expressions can be read as declarative statements. For instance, the meaning of expression (a) is: 'The spring (H_{fs}), which is a constituent on the fifth aggregation level, is a part of the MEMS (H_f), which is a constituent on the fourth aggregation level'.

(a) $PCon^4_{Hf}\ Con^5_{Hfs}$
(b) $PCon^4_{Hf}\ Con^5_{Hfp}$
(c) $PCon^4_{Hf}\ Con^5_{Hfm}$
(d) $PCon^4_{Hf}\ Con^5_{Hfc}$
(e) $PCon^4_{Hf}\ Con^5_{Hft}$

Figure 8 specifies and graphically visualises the connectivity and containment relations between the constituents of the MEMS accelerometer. The constituents are all hardware constituents, working based on different physical phenomena and effects. The following symbols are used to indicate the characteristic attributes of the included constituents: (i) *m*: movement, (ii) *r*: resistance against deformation, (iii) *e*: electric current, and (iv) *g*: electromagnetic field.

Considering these, the connectivity relationships of the constituents can be described by the connected-to and connected-with relations given below. For instance, the symbolic expression (a) means: 'The proof mass (H_{fp}) as a constituent is directly connected with the spring (H_{fs}) constituent'. The enabler on the spring side of the connection is resistance against deformation (r_1), while the enablers on the proof mass side of the connection

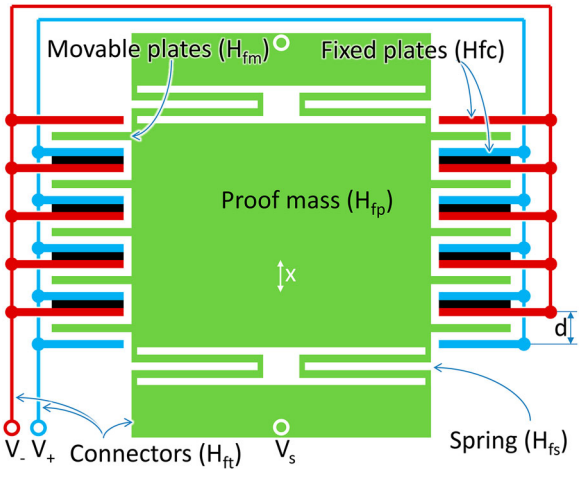

Figure 7. Schematic architecture of the MEMS detector (H_f).

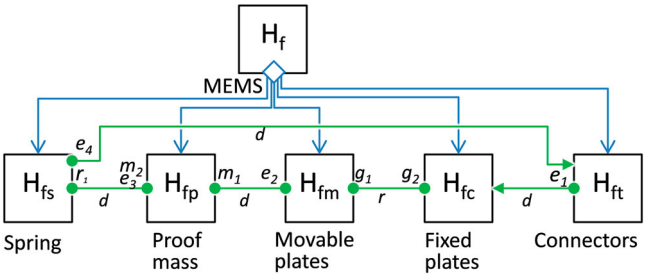

Figure 8. Connectivity relations among the constituents of H_f.

are movement ($\underline{m_2}$) and electric current ($\underline{e_3}$). The symbols indicating the enablers clarify what are shared in the mutual connections. For instance, the proof mass deforms the spring, and the spring exerts a reinstating force. Moreover, $\underline{H_{fp}}$ conducts electric current to $\underline{H_{fs}}$.

(a) $d.CCon_{Hfs}{}^{r1}\ wCon_{Hfp}{}^{m2,e3}$
(b) $d.CCon_{Hfp}{}^{m1}\ wCon_{Hfm}{}^{e2}$
(c) $r.CCon_{Hfm}{}^{g1}\ wCon_{Hfc}{}^{g2}$
(d) $d.CCon_{Hfs}{}^{e4}\ tCon_{Hft}$
(e) $d.CCon_{Hft}{}^{e1}\ tCon_{Hfc}$

This form of stating (by symbolic relations) of the containment and connectivity relationships supports their computational processing.

2.4. Formal specification of architectural entities and relationships of SMFs for computation

To capture the architectural entities and relationships of SMFs for computation, the concept of the so-called *architecture knowledge frames* (AKFs) has been introduced. Every AKF represents one particular domain of manifestation and, by doing so, forms the spatiotemporal basis of exactly one SMF. As a structured spatiotemporal abstraction, this domain may represent a whole system, or any one of its first-, second-, third-level, or so forth de-aggregates, which are treated uniformly as SMFs. The abstracted architectural domains of SMFs reflect one particular level of component aggregation. The system, as top-level formation, manifests itself as the aggregate of the first-level compound components (AW), while the lowest level domains of components are formed by aggregates of HW, SW and/or CW constituents. The domains on the intermittent levels are formed in the same aggregative manner. The specific contents of AKFs are the basis of the digital processing of the architectural entities and their relationships. It is to be noted that in the case of systems of higher complexity, additional levels of aggregation/de-aggregation can be considered, and thus AKFs can in principle be extended to any number of aggregation levels.

An AKF consists of the following sections: (i) header, (ii) domain metadata, (iii) domain entities, (iv) entity attributes, (v) entity morphologies, (vi) spatial positions, (vii) containment relations, (viii) connectivity relations, (ix) input assumptions, (x) output guarantees,

and (xi) auxiliary data. The header section contains the identifiers of all entities that can be referenced in containment and connectivity relations, in addition to the overall attributes and states of the domain. Altogether these form a part of the various computational information structures describing SMFs. During computation of operations, references are made to these fields of AKFs from *operation knowledge frames* (OKFs). These external structural connections of SMFs are included in the 'domain metadata' section of the AKFs. This way, any web-type arrangement of domains of SMFs can be captured. The names and purposes of the represented SMFs are also included in the metadata field, to support informing designers and keyword-based retrieval.

As examples, we present two instantiations of AKF for two different SMFs. The first AKF is instantiated for the MEMS accelerometer (H_f), which is an electromechanical component. The AKF shown in Figure 9 captures the pieces of information needed to describe the architectural domains of this electromechanical component, including the physical activating constituents, whose names and relationships are shown in Figure 8. Figure 10 shows an instance of another AKF, which is the knowledge frame of the ASIC converter of the accelerometer (A_s). This is a computational component. The AKFs of these SMFs include the specification of the lower-level entities making up their domains, assign reference-able identifiers to each of them and specify their types, architectural attributes, convex spatial closures and reference points relative to the reference system of coordinates of the domain. The containment and connectivity relations are specified for all included entities. In general, the morphology of the physical entities can be specified by *skeleton models* (if the entity geometries are still in the development process) or by computer-aided design (*CAD*) *geometries* (if they are standard or commercialised components).

3. Defining the operations of SMFs

3.1. Logical framework of specifying operations

Based on the reductionist stance taken in our work, we argue that flows and units of operations can be reasoned out from the overall operation of a system, and that the overall system operation can be aggregated from a finite set of discrete, but interlaced flows and units of operations. However, it does not entitle us to deal with non-linear, stochastic or random systems. In line with these, one of the foundational assumptions of MOT is that, similar to architectural domains, operations of SMFs can be generated by aggregation. It has also been assumed that the flows and units of operations are (i) discrete (individually identifiable), (ii) finite (in terms of cardinality and occurrence), (iii) definitive (implied by physical and computational laws), and (iv) orientated (i.e. not reversible in the operation process of a system). Below, we present the concepts and principles that help consideration of these assumptions in the computational specification of the operations of SMFs.

Theoretically, operation of an SMF can be only existence or (existence and) transformation. Every domain that is identified from an architectural point of view is supposed to be existent, but not necessarily conducting transformation. In the case of an *existence operation*, the attributes of the start state and the end state of a domain are identical, i.e. a 'no-change situation' is idealised. In the case of a *transformation operation*, the start

AKF // H_f //

header	*entity identifiers* MEMS accelerometer: H_f connectors: H_{ft} fixed plates: H_{fc} movable plates: H_{fm} proof mass: H_{fp} spring: H_{fs}		*domain attributes* digital hardware: DHW analogue hardware: AHW silicon: Si spring deformation limit: X_S		*domain states* acceleration: a supply energy: se output voltage: ov working temperature: TA	
domain metadata	*description:* *purpose:* *structure:*	MEMS detector hardware component capturing acceleration of movements and converting it into capacity change: *UoO*$_{CM}$ $A_a \diamond\!\!\rightarrow A_c \diamond\!\!\rightarrow A_n \diamond\!\!\rightarrow H_f$				
domain entities	*list of DEs:*	H_{ft}	H_{fc}	H_{fm}	H_{fp}	H_{fs}
entity types	*groups of ETs:*	AHW	AHW	AHW	AHW	AHW
architectural attributes	*list of AAs:*	Si	2 x (5 finger) Si	2 x (5 finger) Si	Si	X_S: 5 µm Si
entity morphologies	*size of closure box:* *model files of EMs:*	(520 x 80) µm -	(680 x 180) µm -	(500 x 170) µm -	(250 x 170) µm -	(250 x 85) µm -
spatial positions	*records of SPs (centre):*	(260 x 40) µm	(340 x 300) µm	(340 x 300) µm	(340 x 300) µm	(340 x 110) µm
containment relations	*list of COs:*	$PCon^4_{Hf}Con^5_{Hft}$	$PCon^4_{Hf}Con^5_{Hfc}$	$PCon^4_{Hf}Con^5_{Hfm}$	$PCon^4_{Hf}Con^5_{Hfp}$	$PCon^4_{Hf}Con^5_{Hfs}$
connectivity relations	*list of CRs:*	$d.CCon_{Hfs}{}^{r1}wCon_{Hfp}{}^{m2,e3}$ $d.CCon_{Hfp}{}^{m1}wCon_{Hfm}{}^{e2}$ $r.CCon_{Hfm}{}^{g1}wCon_{Hfc}{}^{g2}$ $d.CCon_{Hfs}{}^{e4}tCon_{Hft}$ $CCon_{Hft}{}^{e1}tCon_{Hfc}$				
input assumptions	*list of IAs:*	a: ± 50g se: 3.6V, 500 µA				
output guarantees	*list of OGs:*	ov: 800~0 mV				
auxiliary data	*list of ADs:*	–40°C < TA < 105°C				

Figure 9. Instantiation of AKF in the case of the MEMS detector, H_f.

state of an existent architectural domain is converted into a different end state under the effects of various physical phenomena. That is, manifestation changes are assumed, which also concern the conversion of the input events, quantities and qualities into the output events, quantities and qualities. The transformation can be physical or computing. The overall operation of a domain is described by a *flow of operation* (FoO), while the operation of the entities is described by a UoO. Since a domain is an architectural equivalent of a particular SMF, the overall operations of SMF are captured by FoOs that are aggregates of UoOs. Since domains may represent architectural manifestation of SMFs of various aggregation levels, an FoO plays the role of a UoO on a higher aggregation level, while a UoO plays the role of an FoO on a lower aggregation level. A practical implication is that specification of a higher-level SMF requires the specification of the operations of all incorporated lower-level SMFs. The involvement of SMFs in performing an operation can be conditional and temporal. These issues need attention at specifying the conduct of transformations performed by SMFs.

AKF // A_s //

header	*entity identifiers* Detector: A_{sa} in-connectors: H_{si} out-connectors: H_{so} processor: A_{sp} clock generator: H_{sc} oscillator: H_{sr}		*domain attributes* digital hardware: DHW analogue hardware: AHW silicon: Si copper: Cu		*domain states* acceleration: a supply energy: se output voltage: ov output signal: os		
domain metadata	*description:* *purpose:* *structure:*	ASIC converter aggregate-ware component converting input voltage into acceleration, velocity and displacement values: UoO_{CA} $A_a \diamond\!\!\rightarrow A_c \diamond\!\!\rightarrow A_n \diamond\!\!\rightarrow A_s$					
domain entities	*list of DEs:*	H_{si}	A_{sa}	H_{so}	A_{sp}	H_{sc}	H_{sr}
entity types	*groups of ETs:*	AHW	DHW	AHW	DHW	DHW	DHW
architectural attributes	*list of AAs:*	Cu	Si	Cu	Si	Si	Si
entity morphologies	*size of closure box:* *model files of EMs:*	- -	- -	- -	- -	- -	- -
spatial positions	*records of SPs (centre):*	-	-	-	-	-	-
containment relations	*list of COs:*	$PCom^4_{As}Con^5_{Hsi}$	$PCom^4_{As}Com^5_{Asa}$	$PCom^4_{As}Con^5_{Hso}$	$PCom^4_{As}Com^5_{Asp}$	$PCom^4_{As}Con^5_{Hsc}$	$PCom^4_{As}Con^5_{Hsr}$
connectivity relations	*list of CRs:*	$CCon_{Hsi}\,tCom_{Asa}$ $CCom_{Asa}\,tCom_{Asp}$ $CCom_{Asp}\,tCon_{Hso}$ $CCon_{Hsc}\,tCom_{Asp}$ $CCon_{Hsr}\,tCon_{Hsc}$ $CCon_{Hsi}\,fCon_{Hft}$					
input assumptions	*list of IAs:*	ov: 800~0 mV se: 3.6V, 500 µA					
output guarantees	*list of OGs:*	os: I^2C protocol					
auxiliary data	*list of ADs:*	-					

Figure 10. Instantiation of AKF in the case of the ASIC converter, A_s.

The reasoning model shown in Figure 11 has been introduced and applied as an attempt towards facilitating formal representation of FoOs of SMFs. According to this model, the operations related to a domain of any architectural level can be defined by seven conceptual elements, namely, by: (i) the description of the domain itself (D), (ii) start state of the domain (SS), (iii) end state of the domain (ES), (iv) input events and values (I), (v) output events and values (O), (vi) procedure of transformation (P), and (vii) methods that are associated with procedural elements (M). Symbolically, FoO = {D, SS, I, P, M, O, ES}. With respect to digital processing, transformations are described by duals of a *procedure* (logic of implementing the transformation) and *methods* (computational algorithms of implementing the transformation). As mentioned earlier, a transformation can be of physical or computational nature. A procedure determines: (i) the UoOs, which eventually transform the concerned material, energy and information streams, (ii) the timed logical sequences of these UoOs and (iii) the permanent or conditional constraints imposed on UoOs. Further explanation on these will be

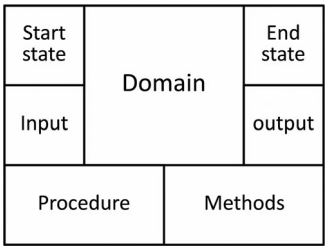

Figure 11. Reasoning model for specification and computation of operations.

given in the following subsections. The specification of the procedures depends on the aggregation level the domain of an SMF represents.

Due to the bijective relationship between a domain and the operations performed by it, whenever the architectural attributes of the domain are changed, the performed operation will also change, and vice versa. Interaction between domains and operations should be considered across the aggregation levels. The reason is that a lower-level UoO can contribute to multiple higher-level FoOs and/or UoOs as procedural element, or may have direct or indirect effects on other operations. Therefore, while a system can logically be de-aggregated into a containment hierarchy, its overall operation cannot be de-aggregated purely hierarchically. As a trivial example, a processor as an architectural domain is a unique part of the logic board and a unique aggregate of several subdomains, but it can contribute to the realisation of operations on multiple operational levels of the system.

An FoO is a logical and temporal (timed) arrangement of transformative actions (UoOs), whose digital processing is enabled by some associated computational methods. A domain may be involved in multiple different physical operations at a given moment in time, or over a time period (Gavrilescu et al. 2010). For instance, the main operation of a processor is digital instruction processing, but its operation also consumes energy, generates heat and so forth. In order to be able to handle multi-operations of a domain, *layers* have been considered for the representation and processing of elements of UoOs. The related issues and solution will be discussed in Section 3.5.

3.2. Defining the computational procedures of operations

The transformations made by SMFs extend to material (M), energy (E) and information (I) streams. In Figure 12, an example is given to show how higher-level operations are aggregated from lower-level operations. This figure also shows how the procedural relationships of UoOs are established on different de-aggregation levels. Like domains and parts thereof in the architectural realm, operations are in containment relationship with each other (that is, an operation can be specified as part of another operation). The architectural 'connected to/connected with' relations are carriers of operational relations of FoOs that specify the transformations of M–E–I streams. These streams are represented by arrows in Figure 12. These show the logical ordering of the related UoOs and FoOs, but temporal ordering or time sequences are not represented.

As shown in Figure 12, each UoO represents an FoO in a lower level. There are two types of M–E–I streams associated with each FoO, namely: (i) internal streams (between the

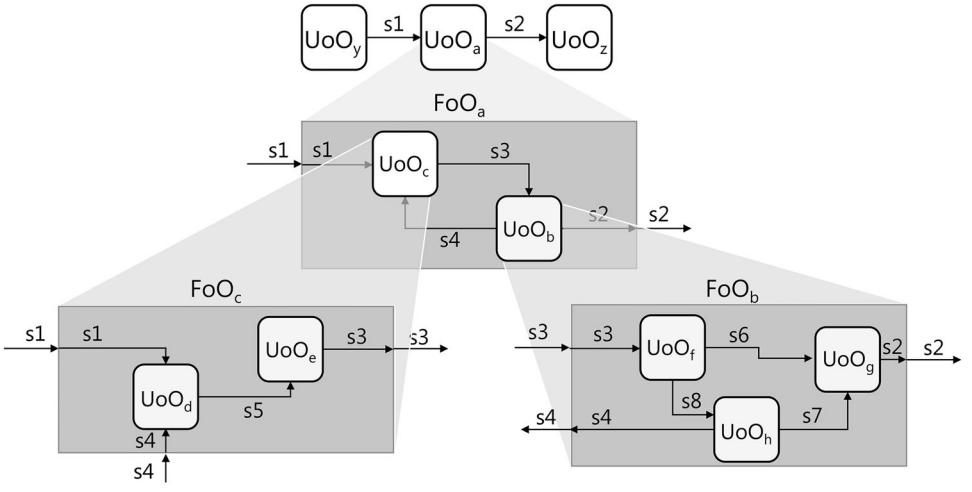

Figure 12. Containment and connectivity relations of FoOs.

UoOs) and (ii) external streams (that connect the UoOs of an FoO to other UoOs outside that FoO). External M–E–I streams are actually inputs and outputs for operations of an SMF. The external streams appearing in a lower-level FoO may be internal streams between UoOs on a higher level, or external streams of some UoOs. For instance, the two FoOs (FoO_b and FoO_c) in Figure 12 together form a higher-level FoO (FoO_a). In a graphical representation, their connectivity can be indicated by connecting the associated external streams that have the same identifiers. Figure 13 shows the aggregation of the mentioned FoOs.

For modelling the operations of a target system based on SMFs as building blocks, we have to determine not only the concerned FoOs, but also the interfaces between these FoOs. Interfaces should be able to connect SMFs according to both containment and connectivity relations. Specification of the interfaces is an important issue for three reasons. Firstly, they should allow an easy replacement of SMFs without paying attention to their internal operations and parameters. That is, they should support flexibility and interchangeability. Secondly, the interfaces make it possible to define SMFs as independent modelling building blocks. For example, if an off-the-shelf component is used, we are not interested in the details of its internal operation, but the interface should capture all

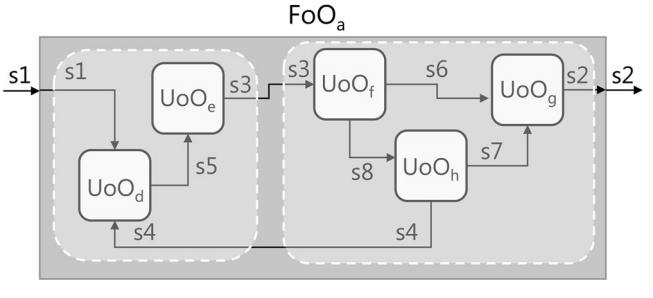

Figure 13. Aggregation of two FoOs.

pieces of information that are needed at building the component into a system as an SMF. Thirdly, the interfaces should capture that amount of information that is exactly needed for a correct architectural and operational embedding of an SMF. In fact, over-defined or incompletely defined interfaces make the matching of SMFs complicated, time consuming or even impossible.

From a computational point of view, aggregation of the operations of SMFs is facilitated by an indexing convention, which is implemented in and applied by the modelling system. As shown in Figure 13, the containment relationships can be rendered by comparing the indexes of UoOs and FoOs. It means that UoOs with the same index as a given FoO should be regarded as its de-aggregation on a lower level. The identifiers of the M–E–I streams also help establish patterns of operational connectivity. Repeating the same stream identifiers in two FoOs indicates that these FoOs are operationally connected. Based on the above formal specification, the operational interfaces can determine what kinds of information need to be captured for the computational modelling of SMFs. These are: (i) identifier of the FoO, (ii) identifiers of the UoOs making up that FoO, (iii) identifiers of the internal streams, (iv) the UoOs that are connected by the respective internal streams, (v) identifiers of the external (outgoing and in-coming) streams, and (vi) the UoOs that receive/send external streams. In order to capture all these pieces of information in a compact form, the concept of *matrix of streams* (MoS) has been introduced. A *MoS* provides a uniform computational representation of all transformations made by UoOs, or by any individual combination of them, on M–E–I streams.

Two groups of operations are associated with an FoO: (i) the operations of the UoOs making up the FoO and (ii) the operations complementing the operations of the FoO. As such, the latter operations are externally related to processing M–E–I streams by the FoO. This second group of system-related complementary operations is indicated by the symbol ∁ (*complement*). The complementing operations may be operations of the embedding environment, a coupled system, a system user, etc. All of these relationships and the other contents are represented in the MoS, as shown graphically in Figure 14. This figure depicts the arrangement of FoO_b and the included UoOs, exemplified in Figure 12, together with the related internal and external M–E–I streams. In the matrix representation, the operations complementing FoO_b are placed into the first position of the descending main diagonal of the matrix. It symbolises a kind of 'gateway' for both the input and output streams. In the rest of the positions in the main diagonal, the respective UoOs of FoO_b are placed. In this context, the complementing operation represents all operations of the embedding system, while the UoOs de-aggregated in the main diagonal represent the operations (FoOs) of the customising SMFs.

The arrows in Figure 14(a) show the orientation, while the circles contain the identifiers of the different M–E–I streams that are processed by the respective UoOs. The places of the identifiers will be the same in the matrix representation as in the graphical representation of an FoO. The size of an MoS matrix equals the number of UoOs in the FoO added by 1. As shown in Figure 14.a, all incoming streams of the concerned FoO are included in the top first row of the MoS, which is called *input row*. All outgoing streams of the FoO are included in the left first column of the MoS, which is called *output column*. That is, the first row and column of MoS always represent external streams, while the rest of the matrix describes the internal streams of an FoO. MoS can be scaled up to capture the aggregation of an arbitrary number of FoOs.

(a)

$$\begin{bmatrix} \complement FoO_b & s3 & - & - \\ - & UoO_f & s6 & s8 \\ s2 & - & UoO_g & - \\ s4 & - & s7 & UoO_h \end{bmatrix}$$

(b)

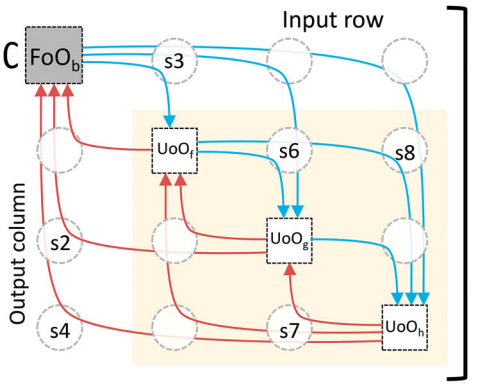

Figure 14. MoS of FoO_b: (a) graphical representation of operations and streams, (b) the matrix of operations and streams.

To demonstrate the applicability of MoS formalism to a real-life case, let us take the example of the Pablo rehabilitation device again. Through this example, we can clarify how application-oriented meaning can be assigned to the formalism and symbols used in Figure 15. We can also demonstrate how information about operations can be extracted from practical processes. Mentioned earlier was that a typical operation mode of the device is side lift. The device captures the range of the movements when side lift is exercised. The two components of the device that are involved in this FoO, namely, the MEMS detector and the ASIC converter, were discussed from an architectural perspective in Section 2.4. They transform acceleration into capacitance change (UoO_{CM}), and convert the value of capacitance change into value of displacement (UoO_{CA}), respectively. Since they have specific operational domain and distinct operations, we regard them as two SMFs.

Let us denote the FoO by FoO_C and all complementary operations of FoO_C by $\complement FoO_C$. With these, we can construct the MoS of the two SMFs of the rehabilitation device, as

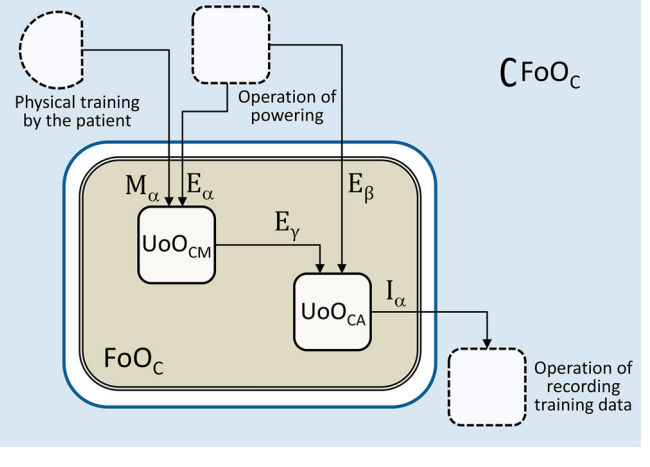

Figure 15. Incoming, internal and outgoing M–E–I streams in the FoO of the accelerometer.

shown in Figure 16(a). The descriptors of the M–E–I streams are shown in Figure 16(b). The units of operations of the two discussed SMFs of the device are included in the main diagonal of the instantiated MoS. The MoS describes that UoO_{CM} transforms an incoming material stream, M_α (acceleration produced by the patient during physical training) and an energy stream, E_α (input powering of the MEMS accelerometer SMF), without producing any outgoing stream. It also shows that UoO_{CA} transforms two incoming energy streams, E_β (input powering of the ASIC converter SMF) and E_γ (the voltage resulted from capacitance change), respectively, into an outgoing information stream, I_a, (pieces of information about acceleration, velocity and displacement).

Having discussed the computational representation of UoOs and the streams related to them (on different levels of granularity), we elaborate on some issues of specifying time dependencies of the operations of SMFs below. First, we discuss the concepts of timing and handling of operational conditions in the specification and computation of operations of the SMF represented by the ASIC converter in Section 3.3. Then, we use the example of the MEMS detector to discuss the computational methods in Section 3.4. Finally, we cast light on the contents of the knowledge frame describing its complex operations in Section 3.5, and discuss the interlacing of AKF and OKF for computational modelling and simulation of SMFs on multiple levels of aggregation in Section 3.6. This approach of information structuring and integral handling is suitable for conjoint handling of AKFs and OKFs of arbitrary number of SMFs.

3.3. Consideration of timing and conditional constraints in computation of operations

Obviously, operations of SMFs should happen in a controlled manner. Therefore, logical, temporal and conditional constraints should be considered in the computation of physical and computing operations. Towards this end, first of all, we need to operationalise the assumptions stated in Section 3.1 and to create opportunity for a purposeful logical arrangement, time scheduling and constraint management of FoOs and UoOs. In the literature, *time stamping* and *temporal logics* are widely used to capture temporal aspects of existence and transformative processes. While time stamping is typically used to assign date and time data to state changes (transformations), temporal logics is used to support procedural scheduling and synchronisation. For temporal management of operations, both event- and chronology-oriented approaches have been proposed (Eidson et al. 2012).

Consideration of time in the formal representation of SMFs has a dual perspective: (i) incorporation of time variables in operational processes, and (ii) time-oriented

(a)

$$\begin{bmatrix} CFoO_C & M_\alpha, E_\alpha & E_\beta \\ - & UoO_{CM} & E_\gamma \\ I_\alpha & - & UoO_{CA} \end{bmatrix}$$

(b) Descriptors:

M_α	: Moving of UoO_{CM} (± 5 g)
E_α	: Electrical energy (3.6V, 500 μA)
E_β	: Electrical energy (3.6V, 500 μA)
E_γ	: Electrical energy (Max. ±860mV)
I_δ	: Info (a, v, d)

Figure 16. Specification of the FoO of the accelerometer (UoO_C): (a) the contents of MoS, and (b) descriptors of the streams.

programming and execution of computation and control of operational processes. Systems having a large number of physical (analogue) and cyber (computing) components pose a challenge for real-time processing and control. In the case of large complexities, there can be a dissonance between the time required for digital computation and the time elapsed by physical operations. The specification of SMFs is supposed to support the resolution of this dissonance and to create a robust platform for time-sensitive computation. Towards this end, every time-dependent operational process needs to be represented as a *timed operation sequence* (TOS) by introducing and evaluation of time variables. The activation points in time and the durations of the units of operations of the concerned material, energy and information streams should be included in quantitative forms in a TOS.

In the context of the Pablo rehabilitation device, there are several UoOs that should be described as TOSs. Consider, for instance, the unit of operation UoO_{CA}, which converts value of the current change into value of acceleration, velocity and displacement. The procedure of this UoO includes five lower-level time-dependent units of operation:

- UoO_{CAA}: amplifying and measuring value of the input voltage (domain: in-connector H_{si}, detector A_{sa})
- UoO_{CAT}: measuring time variable (domain: oscillator H_{sr}, clock generator H_{sc})
- UoO_{CAC}: calculating and sending acceleration value (domain: processor A_{sp}, out connector H_{so})
- UoO_{CAV}: calculating and sending velocity value (domain: processor A_{sp}, out connector H_{so})
- UoO_{CAD}: calculating and sending displacement value (domain: processor A_{sp}, out connector H_{so})

The above-mentioned five lower-level units of operation need to be described as TOSs. This can be done by including the operation's start point in time, halt points in time, resume points in time and end point in time. This time-dependent parameterisation of operation flow (FoO_{CA}) is shown in Figure 17. The start point in time and the end point in time define the whole duration of this FoO. The change of a domain due to physical or computational operation is considered as an event (Tan, Vuran, and Goddard 2009). Every event has its own time duration – this explains why halt points in time and resume points in time are made explicit in the time-oriented parameterisation of the descriptors. For example, the stream E_γ starts at t_1, is halted at t_{C1}, is resumed at $t_{C1}+\Delta_2$ and stops at $t_1+\Delta_1$. These time dependences are specified for each stream in the table of descriptors, as shown in Figure 18. According to the applied convention, the first item in the descriptors block defines the start time and the last item defines the end time of an operation. As an example, the time specification for stream E_γ (in the fourth row of Figure 18) is as follows: t_1, $.t_{C1}$, $\wedge t_{C1}+\Delta_2$, $t_1+\Delta_1$, where the first item is the start time, the item after '.' is the halt time, the item after '$\wedge$' is the resume time and the last item is the end time. An operation descriptor with one timing item only means that it does not have duration.

It can be seen in Figure 17 that, as input of UoO_{CAA}, $E_{\beta 1}$ stands for the supply of energy, while E_γ and $E_{\gamma'}$ are the inputs generated by the MEMS that detects acceleration. The input energy streams $E_{\beta 1}$ and $E_{\beta 2}$ are operated on only one time. However, the stream $E_{\beta 3}$ is

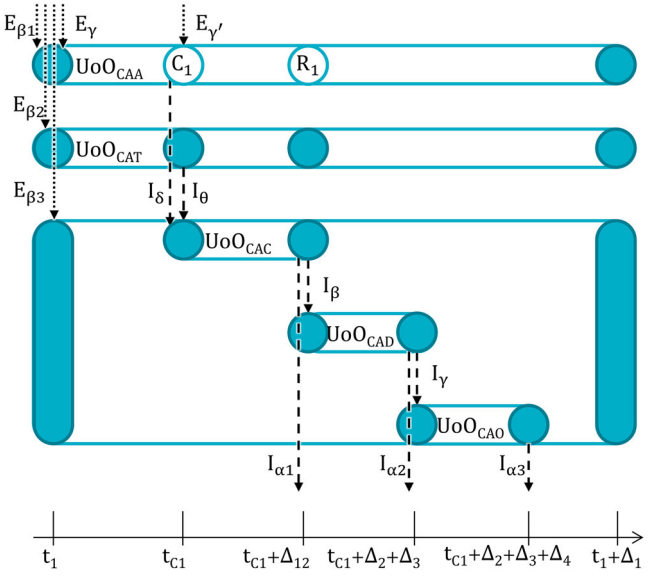

Figure 17. Time-oriented parameterisation of FoO_{CA}.

Descriptors:

Streams:

$E_{\beta1}$; $(t_1, t_1+\Delta_1)$:energy supply(3.6V 500 μA)

$E_{\beta2}$; $(t_1, t_1+\Delta_1)$:energy supply(3.6V 500 μA)

$E_{\beta3}$; $(t_1, t_1+\Delta_1)$:energy supply(3.6V 500 μA)

E_{γ} ; $(t_1, . t_{C1} \wedge t_{C1}+\Delta_2, t_1+\Delta_1-t_{C1}+\Delta_2)$:regular *V-in*(1.2*V*)

$E_{\gamma'}$; $(t_{C1}, t_{C1}+\Delta_2)$:varied *V-in* (800~0 *mV*)

I_{δ} ; $(t_{C1}, t_{C1}+\Delta_2)$:Info (value of the *V-in*)

I_{θ} ; $(t_{C1}, t_{C1}+\Delta_2)$:Info (time)

$I_{\beta} = I_{\alpha1}$; $(t_{C1}+\Delta_2)$:Info (*acceleration*)

$I_{\gamma} = I_{\alpha2}$; $(t_{C1}+\Delta_2+\Delta_3)$:Info (*Velocity*)

$I_{\alpha3}$; $(t_{C1}+\Delta_2+\Delta_3+\Delta_4)$:Info (*displacement*)

Events:

t_1 : Turn on the ASIC

$t_1+\Delta_1$: Turn off the ASIC

t_{C1} : Apply acceleration

Conditions:

C_1 : (t_{C1}) : if$(E_{\gamma'} \geq E_{\gamma}. 3/2)$, Then$(I_{\theta} \wedge I_{\delta})$

R_1 : $(t_{C1}+\Delta_2)$: repeat(C_1)

$t_{C1} \geq t_1$

$t_1+\Delta_1 \geq t_{C1}+\Delta_2+\Delta_3+\Delta_4$

Constraints:

–

Figure 18. Descriptors of the procedure of FoO_{CA}.

operated on three times by three different UoOs. Performed on the same architectural domain, these three UoOs form a group of operations with respect to $E_{\beta 3}$ and are captured by a TOS. The operation multiplicity of stream $E_{\beta 3}$ is reflected by the content of the top row of the MoS shown in Figure 19. In this sample case, the time constraints on the operation of the concerned SMF are the point of time of the input and the point of time of the output of the respective streams.

Many of the physical and computational operations of SMFs are typically executed under (i) space, (ii) time, (iii) attribute, (iv) logical and (v) recurrence constraints. These constraints should also be included in the computational specification of the operations of SMFs. Usually, space, time and attribute constraints specify a threshold (minimum) value, or a ceiling (maximum) value or a variation interval (min/max values) for the respected variables. The logical constraints express either propositional or production rules. Production rule-type conditional constraints are declared as: *if* (conditions) *then* (consequence) *else* (alternative). The consequences are the states of the M–E–I streams that are influenced by the conditional events. In order to express temporal sequences, time variables are assigned to these events in our approach.

Since the techniques of using value constraints are well known in the literature (Goldratt 1990) and rather frequently used in engineering computations (Benhamou, Jussien, and O'sullivan 2013), we do not address this issue here. We only touch upon the use and utility of logical (if–then) and recurrence constraints, which we refer to as *conditional constraints*. These are introduced at a given point in time, and either remain active in the rest of the operation or are deactivated at a given point in time or when a given time duration is elapsed. For instance, the empty circles in the upper part of the diagram in Figure 17 introduce two conditional constraints, C_1 (which specifies a logical condition) and R_1 (which specifies a recurrence condition) in the context of UoO_{CA}, These conditions are included in the condition field of the table of descriptors, which specifies the working variables associated with the streams and the timed events too.

3.4. Defining the computational methods of operation of SMFs

Other important resources for computation of operation of SMFs are *methods* that are represented by quantitative formulas based on which the transformation of the input streams of UoOs into output streams can be computed. An arrangement of methods is needed to transform the starting state of an SMF into the end state. The methods should be compliant with physical and computational laws and principles, as well as the constraints of operation. Methods are specified by mathematical expressions, systems variables and constraints. The physical and computational laws (e.g. gravity, electromagnetic field,

$$\begin{bmatrix} CFoO_{CA} & E_{\gamma}, E_{\gamma'}, E_{\beta 1} & E_{\beta 2} & E_{\beta 3} & E_{\beta 3} & E_{\beta 3} \\ - & UoO_{CAA} & - & I_{\theta} & - & - \\ - & - & UoO_{CAT} & I_{\delta} & - & - \\ I_{\alpha 1} & - & - & UoO_{CAC} & I_{\beta} & - \\ I_{\alpha 2} & - & - & - & UoO_{CAD} & I_{\gamma} \\ I_{\alpha 3} & - & - & - & - & UoO_{CAO} \end{bmatrix}$$

Figure 19. MoS of the procedure of FoO_{CA}.

friction, etc.) and principles are represented by equations or set of equations. System variables are groups formed by individual variables and values associated with an operation parameter, in order to reflect relatedness. System variables quantify input parameters (e.g. electrical DC current of 12 V, 2 mA, for 25 min), intermittent parameters and output parameters (e.g. vertical displacement of 173 mm). The methods have a one-to-one relation with the procedural elements of a UoO, but a particular method can be applied to multiple procedural elements. Below, we give a practical example for deriving the methods from real-life operation.

The MEMS accelerometer is an analogue electromechanical component that detects acceleration based on the movement of capacitor plates attached to a spring. As illustrated in Figure 20, it converts the detected acceleration into capacitance change, ΔC. The input acceleration produces a dynamic force in the proof mass ($F = m\ a$), which moves the plates against the spring. In turn, the spring deforms and causes a displacement $\times$ on the capacitive plates ($x = \frac{F}{k_s}$, where k_s is the spring constant (Young's modulus)). Knowing that $x = d\frac{\Delta C}{C_0}$ (where d is the distance between the constant plates, C_0 is the initial capacitance and ΔC is the change of capacitance proportional with displacement x), the change in capacitance is proportional to acceleration (Lyshevski 2002). These transformative steps can be described by one equation:

$$\Delta C = \frac{mC_0}{k_s d} a.$$

Many parameters of the MEMS accelerometer, such as Young's modulus of elasticity of the spring material, the proof mass, the distances between capacitor plates and the initial capacitance, are constant. Taking this into consideration, we can define a so-called accelerometer constant (k_a) that can be expressed as $k_a = \frac{k_s d}{mC_0}$. Using this, the relationship between the input and output quantities of the operational domain representing the MEMS accelerometer is $a = k_a \Delta C$. These equations are the basis of the computational method and algorithm.

The ASIC converter performs computational operations. It is a chipset that converts the value of capacitance change, ΔC, into a displacement value. As an operational domain, it

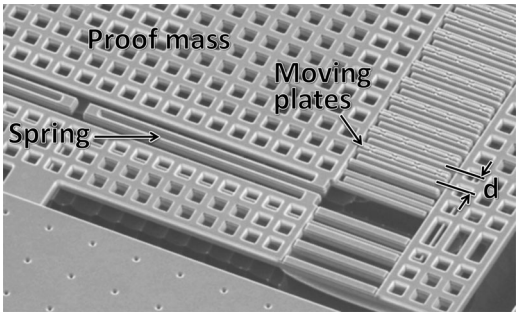

Figure 20. MEMS accelerometer (Fedder et al. 2008).

includes an (i) oscillator, (ii) a clock generator, (iii) computing units, (iv) detectors and (v) filters. The UoOs of the converter are described in Section 3.3. Knowing that $a = k_a \Delta C$, and $d = tv_0 + \frac{1}{2}t^2 a$, an equation describing the operation of the converter (i.e. converting the value of the input voltage to the distance) can be derived. The ASIC converter applies a Monte Carlo method when the acceleration is varying. Assuming that the value of acceleration in the second equation is constant, the displacement can be calculated by considering the time (t), accelerometer constant (k_a), value of current change (ΔC) and initial speed (v_0). Thus, the computational method can be described by the following equation:

$$a = k_a \Delta C,$$

$$\bar{v} = t k_a \Delta C,$$

$$d = tv_0 + \frac{1}{2}t^2 k_a \Delta C.$$

It should be taken into consideration at applying the method that it is accurate only in the case of low acceleration and short distances.

3.5. Formal specification of operations of SMFs for computation

Resembling the formalism of AKFs, OKFs have been defined and used as means of structuring the information needed to describe physical and computational operations of SMFs. An AKF captures domain-specific containment (architectural aggregation) and connectivity (architectural dependence) information for computing, whereas an associated OKF describes an FoO and multiple units of physical and computational operations on some aggregation level. OKFs have a complex and sophisticated architecture since they have been designed to capture all information necessary for computation. The information is used for: (i) creating a pre-embodiment artefact model, (ii) integration of the computational methods/algorithms and (iii) simulation of the behaviour of an SMF in various contexts. In addition to time-sequenced operations, the information structure of OKFs can describe concurrent multi-physics operations and can be used as the basis of controlling parallel computations of operations. From a computational viewpoint, every OKF is associated (exchanges data) with the AKF of a particular SMF. This will be discussed in the next subsection.

An OKF consists of the following sections: (i) header, (ii) operation metadata, (iii) domain states, (iv) domain transformations, (v) operation constraints, (vi) units of operation, (vii) operation layers and (viii) auxiliary data (Figure 22). The header section contains the identifiers of all entities that can be referenced by internal and external connectivity (operation) relations. It extends to the symbols (variables) used for the procedural elements (UoOs), M–E–I streams transformed by the domain, and the states and events of the domain. The metadata field of the OKF contains information about the FoO and the corresponding overall domain. The description of the operational state of the domain includes the start state and the end state, the specification of the timing of the streams, and the input and output events. The transformation section includes the contents of the MoS, and the information about the numerical and algorithmic processing of the concerned FoO. The operation constraints section specifies the overall time stamps, the

scheduling constraints for the FoO and the operation constraints with regards to the changes in the M–E–I streams. The section of units of operation includes the identifiers of the subdomains and the UoOs performed on them. This organisation of the content of the OKFs makes it possible to refer to one or more UoOs of an OKF from a higher-level OKF. Introducing the operation layer concept, and the related section, in the OKF makes it possible to handle multi-state, concurrent and hierarchical operations.

An important element of the OKF formalism is the use of layers for processing coinciding physical and computational operations. Actually, layering allows separating (computational) concerns, but also avoiding possible conflicts of computing concurrent operations (Figure 21). Introduction of layers not only rationalises the computational specification of operations, but also provides flexibility in processing concurrent elements of operation. For instance, layering makes the various states of software (language code, compiled executable, processor instruction) describable. Similarly, two or more different concurrent operations of components, whose computation needs different methods, can be captured (e.g. dynamic loading, stress calculation and thermal dilatation calculation can be combined into a multi-physics operation). In addition, layering also supports the integral treatment of primary, secondary and tertiary physical effects (e.g. rotational motion of a rubber wheel, the wear due to friction and the material behaviour on a molecular level). Each layer captures those alternative operations that are related to the same subdomain. Layers are not considered to be independent of each other. On the contrary, they are defined to share architecture and operation identifiers and variables. In this way, they can connect individual operations into a composite operation, which is needed in computation of multi-physics operations. For instance, an energy consumption variable used in a layer assigned to energy change can be specified as dependent on the temperature variable used in another layer assigned to heat change. The status of the individual variables and constraints described in the various layers can indicate if execution of an operation is possible, or not.

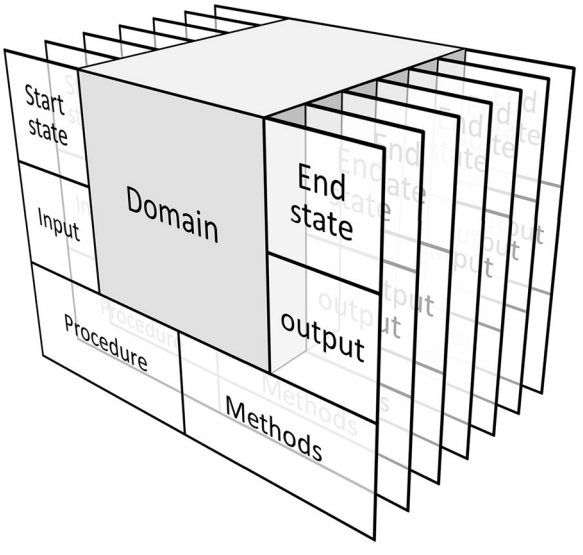

Figure 21. Multi-layer view of a UoO.

OKF // FoO_{CM} //

header	*procedural elements (UoOs)*		*domain streams*		*domain states and events*	
	UoO_{CMC}: conducting electricity into the capacitors' plates UoO_{CMA}: receiving acceleration and converting it to force UoO_{CMP}: pressing the spring by the force UoO_{CMD}: changing distance between plates of capacitors UoO_{CMO}: conducting electricity from capacitors to the MEMS output		E_α: energy supply M_α: applied acceleration M_β: applied force to H_{fs} M_γ: applied movement M_δ: displacement of capacitors' plates E_δ: potential electricity in capacitors E_γ: regular *V-out* $E_{\gamma\prime}$: varied *V-out*		e1: turn on the MEMS e2: receive acceleration e3: turn off the MEMS e4: send out regular *V-out* e5: send varied *V-out* S1: spring is not pressed S2: distance between H_{fm} and H_{fc} is maximum S3: *V-out* is 1.2V	
operation metadata	*FoO descriptor: domain*	FoO_{CM}: H_f				
start state of the domain	*list of SS variables:*	S1, S2, S3				
end state of the domain	*list of ES variables:*	S1, S2, S3				
list of streams	*list of streams and their starting time and duration:*	$E_\alpha\ (t_1,\Delta_1)$; $M_\alpha\ (t_2,\Delta_2)$; $M_\beta\ (t_2,\Delta_2)$; $M_\gamma\ (t_2,\Delta_2)$; $M_\delta\ (t_2,\Delta_2)$; $E_\delta\ (t_1,\Delta_1)$; $E_\gamma\ (t_1, .t_2, \wedge t_2+\Delta_2, t_1+\Delta_1)$; $E_{\gamma\prime}\ (t_2,\Delta_2)$;				
input events	*list of IE variables:*	e1: (E_α) e2: (M_α) = *a*				
output events	*list of OE variables:*	e4: (E_γ) e5: ($E_{\gamma\prime}$) = ΔC				
transformative procedure:	*list of physical and or computing units of operation:*	$\begin{bmatrix} CFoO_{CM} & E_\alpha & - & M_\alpha & - & - \\ - & UoO_{CMC} & - & - & - & - \\ - & - & UoO_{CMA} & M_\beta & - & E_\delta \\ - & - & - & UoO_{CMP} & M_\gamma & - \\ - & - & - & - & UoO_{CMD} & M_\delta \\ E_\gamma, E_{\gamma\prime} & - & - & - & - & UoO_{CMO} \end{bmatrix}$				
numerical processing of FoO	*order of numerical methods*	$\Delta C_{(E\gamma)} = \frac{m_{(Hfp)} C_{0(E\alpha)}}{k_{(Hfs)} d_{(Hfc)}} a_{(M\alpha)}$				
algorithmic processing of FoO	*order of algorithmic methods*	-				
time stamps	*start time, suspension times, duration, commencing times, end time variables:*	t_1 = t(e1) ≈ t(e4) $t_1+\Delta_1$ = t(e2) ≈ t(e5) t_2 = t(e3)				
scheduling conditions	*list of conditions*	$t_2 \geq t_1$ $t_1 + \Delta_1 \geq t_2+\Delta_2$				
operation constraints	*list of constraints*	–40°C < TA < 105°C ; 2.2 V < E_α < 3.6 V ; -5g < M_α < +5g ; E_γ= 1.2 V ; 0 mV < $E_{\gamma\prime}$ < 800 mV				
operation domain entities	*list of operation domain entities*	$H_{ft} \oplus H_{fs} \oplus H_{fp} \oplus H_{fm} \oplus H_{fc}$	H_{fp}	$H_{fp} \oplus H_{fs}$	$H_{fm} \oplus H_{fc}$	$H_{fm} \oplus H_{fc} \oplus H_{ft}$
UoOs	*list of units of operations:*	UoO_{CMC}	UoO_{CMA}	UoO_{CMP}	UoO_{CMD}	UoO_{CMO}
numerical methods of UoOs	*list of numerical methods*	Conducting E_α	$F_{(M\beta)} = a_{(M\alpha)} m_{(Hfp)}$	$x_{(Hfs)} = \frac{F_{(M\beta)}}{k_{(Hfs)}}$	$x_{(Hfs)} = x_{(Hfm)}$	$\Delta C_{(E\gamma)} = \frac{x_{(Hfs)} C_{0(E\alpha)}}{d_{(Hfc)}} a_{(M\alpha)}$
algorithmic methods of UoOs	*list of algorithmic methods*	-	-	-	-	-
operation layers of UoOs	*identifier and order of layers*	-		$L_{CMC,1}$	$L_{CMC,2}$	
	software manifestation	-	-	-	-	-
	concurrent operations	-	-	$x = \frac{F}{k_s}$	$\frac{k(T)}{k(To) + k(To)\zeta k \Delta T} = d = 2\left(\frac{F}{4}\right)\left(\frac{l^3}{k(T)l}\right)$	-
	subordinate operations	-	-	-	-	-
auxiliary data		-	-	-	-	-

Figure 22. Instantiation of OKF in the case of FoO_{cm}.

A typical example of using layers in computing the simultaneous changes due to multiple concurrently present physical changes is consideration of effects of temperature variations in a wide temperature range on the MEMS accelerometer, which need to be minimised in the process of optimisation (Liu et al. 2015). Because of the temperature variations thermal deformation occurs, which has a negative effect on the output performance of the accelerometer (Wang, Li, and Rizos 2009). The main operation (i.e. detecting the amount of acceleration) and the accompanying operation (i.e. deformation under varying temperature) can be described using two layers. The two layers contain the data needed for the computation of both respective operations, and the results of the computations on the layers can be blended when they are available. The assignment of computational methods to the layers of the springs of the detector, which connect the proof mass to the frame, can be seen in the OKF shown in Figure 22. Calculating the changes for both layers in the case of a sequence of sufficiently small Δt, we can find the deviations of the output capacitance and resonance frequency caused by fluctuations of temperature. The influence of temperature on Young's modulus of elasticity of the spring material is approximated by the following method:

$$k(T) = k(T_o) + k(T_o)\ \zeta_E\ \Delta T,$$

where $k(T)$ is Young's modulus of elasticity of the spring material at a temperature T, $k(T_o)$ is Young's modulus at the ambient temperature T_o, ζ_E is the temperature coefficient of Young's modulus and ΔT is the relative temperature change. The deflection of the beam parts of the spring can be calculated by the following method:

$$d = 2\left(\frac{F}{4}\right)\left(\frac{l^3}{k(T)I}\right),$$

where d is the deflection, F is the acceleration force, $k(T)$ is the actual Young's modulus of the spring material, and I is the moment of inertia of the spring beam. The data handled on the layers are derived partly from the general specifications in the domain states, domain transformations, operation constraints and units of operation sections of the concerned OKF and partly from the contents of the descriptive fields of the related AKF.

The indexes in the OKF shown in Figure 22 refer to domains and streams. Due to the fact that a UoO can be referred to from multiple different aggregation levels, attention is to be paid to parameterisation. The issue is that parameters used on different levels are not independent from each other. Certain parameters and parameter relationships

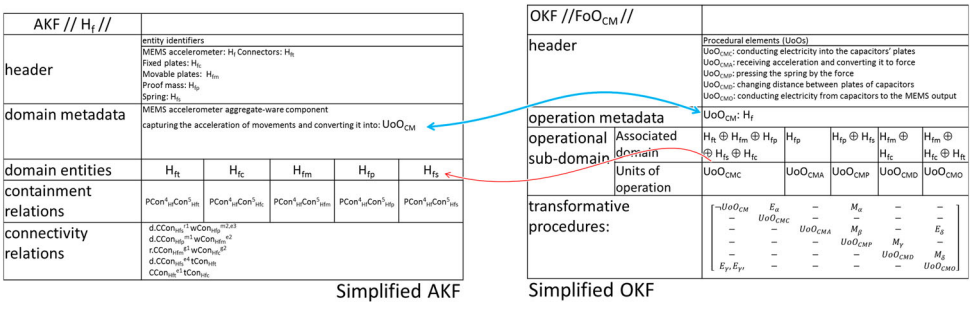

Figure 23. Interlacing the metadata of AKFs and OKFs.

may be shared by the different levels, but some parameters may be relevant just on a particular level of aggregation. In order to be able to handle the variables on different levels, the shared parameter variables are specified as two-tuples such that $V = \{v, n\}$, where v is a valuated variable used in an SMF and n is the aggregation level. However, in computing a mathematical equation or evaluating a logical relationship only those parameters are considered, which belong to the same aggregation level.

4. Interlacing AKFs and OKFs for computational modelling and simulation of SMFs on multiple levels of aggregation

4.1. Connections among AKFs and OKFs

We focused on the specification of the descriptive information structures in the preceding sections. Therefore, we separated the architectural and operational aspects and the architectural and the OKFs. However, they are integral elements of SMFs and should be processed in a conjoint manner when SMFs are used for artefactual modelling of a system, or when the operation of systems built up by SMFs is computationally simulated. We refer to the link between the architecture and OKFs as *interlacing*. From a data processing point of view, interlacing is enabled by a set of references (field pointers) among the OKFs and the AKFs associated with them. This creates a composite view whose advantage is that it involves both the architectural and the operational aspects when systems are developed by using SMFs as modelling entities. On the one hand, the introduced AKF and OKF specification formalism rationalises and simplifies the development and modification of SMFs, and, on the other hand, it supports a dynamic artefactual modelling and behavioural simulation of heterogeneous systems based on pre-programmed SMFs and structures thereof. A significant advantage of the described approach is its flexibility with regards to aggregation and de-aggregation.

There are data fields included in the AKFs and OKFs for the sake of representing lower-level entities. In AKFs, this data field is named *domain entities*. In OKFs, this data field is called *units of operation* (Figure 23). The lines show the mutual relations between the

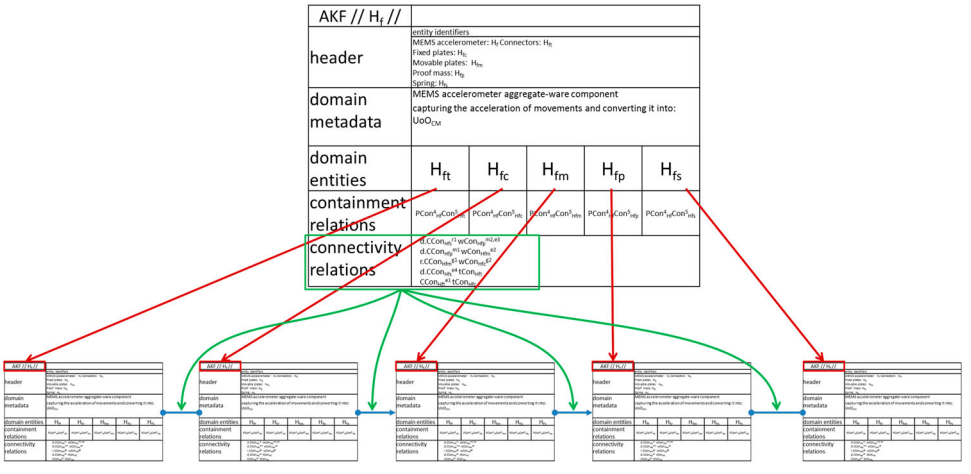

Figure 24. Containment and connectivity relations among AKFs of various levels.

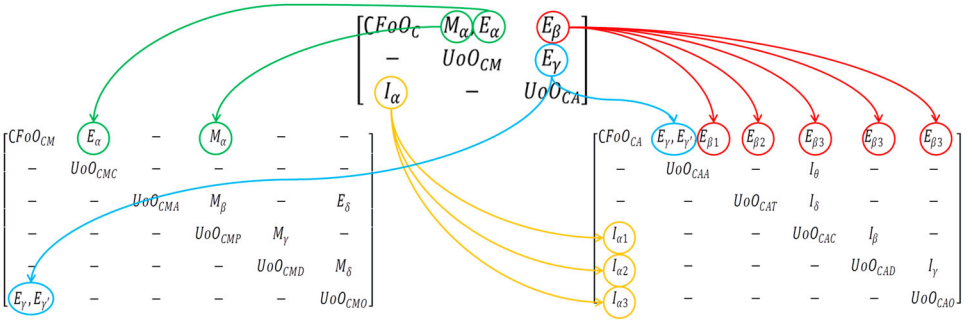

Figure 25. Streams establishing operational relations on two levels.

metadata fields of two complementary knowledge frames, as well as those among the domain entities and the associated units of operations. The reference identifiers used consistently in both types of knowledge frame helps to simplify the search in and link run-time associated frames. Considering that one architectural domain may be linked to several UoOs and vice versa, multiple reference identifiers are typically included in the metadata fields. The simultaneous reference to the domains entities and to the units of operation in the OKFs offers an opportunity for a direct checking of the correctness of the specification of the contents of the frames before computation.

4.2. Multi-level handling of knowledge frames

Interlacing happens not only horizontally, that is between the fields of AKF and those of the corresponding fields of the OKF, but also vertically, that is among the specific fields of AKFs and OKFs representing different aggregation levels. Figure 24 shows the relationships among AKFs describing unique architectural components of an aggregate SMF. The information about containment can be retrieved from the metadata fields of the predefined SMF. Connectivity relations are specified among the subordinate components within an AKF, and make it possible to establish operational relations among them. The so-called *permanent internal* containment and connectivity relations are coded when the contents of an SMF are specified. Therefore, these relations can be changed only by redefining the whole SMF. Other parts of the relations, called *dynamic external* containment and connectivity relations, are established when SMFs are combined into structures in the design process. The data concerning these relations are included in the respective aggregate level AKF and OKF in run-time. The connectivity data can be referred from the fields of the aggregate-level frames to the fields of any lower-level frames, as needed by the intended operation.

The *dynamic external* relations give the basis for describing time- and constraint-dependent operations, i.e. controlled transformations of M–E–I streams. An explanatory example is given in Figure 25. In this case, two aggregation levels of operations are considered. The top matrix (MoS_C) represents the streams of the accelerometer sensor (FoO_C), which is an aggregation of the MEMS acceleration detector (UoO_{CM}) and the ASIC converter (UoO_{CA}). The streams of the MEMS detector (MoS_{CM}) are shown on the left side of the figure and the streams related to the ASIC converter (MoS_{CA}) are

demonstrated on the right side of this figure. The two aggregation levels are interrelated by the transformed streams.

The internal and the external operational relations are shown in the respective matrices of streams (Figure 25). The higher-level UoO concerns one material stream, one information stream and three energy streams. These streams variously appear in the lower-level UoOs. For example, after being transferred into FoO_C, the material stream, M_α, is transformed by UoO_{CMP}. In MoS_C, the *external energy supply* stream, E_β, is received by UoO_{CA}. MoS_{CA} provides power for all lower-level UoOs. The appearance of the indices of a stream in a lower-level indicates that the concerned stream is de-aggregated. As an example, by de-aggregating the information stream (I_α) we get to the three lower-level streams. In the MoS_C, there is an internal energy stream (E_γ) from UoO_{CM} to UoO_{CA}. As explained before, the MEMS detector detects the acceleration and generates a proportional *output voltage*, E_γ, which is converted into information about the *acceleration*, $I_{\alpha 1}$, *velocity*, $I_{\alpha 2}$, and *displacement*, $I_{\alpha 3}$, by the ASIC converter. In the lower aggregation level, E_γ is an external stream from FoO_{CM} to FoO_{CA} that connects the two corresponding MoOs. These transformations can be modelled and processed through the multi-level handling of operational relations, as described above.

5. Discussion, conclusions and future work

5.1. Discussion of the new insights and the target application

MOT was presented in one of the authors' earlier papers (Horváth and Pourtalebi 2015); therefore only the details of the SMF theory and its implications for the system-level design methodology are discussed. The constructs proposed by MOT have been used for architecture and operation modelling of SMFs as building blocks in pre-embodiment design of heterogeneous systems. SMFs naturally implement what is called HW, SW and CW co-design (Wolf 1994). Though the idea of using parameterised features in CAD processes is well known and has reached a mature level in commercialised CAD systems, the idea of SMFs-based configuration and adaptation of heterogeneous systems has been considered just recently. This paper makes an attempt to contribute to the further development of this approach. Therefore, like in the 'classical' part feature theory, various issues of information structuring and parameterisation have been addressed. Parameterisation has been considered not only from geometric and structural aspects in the presented work, but also from a semantic aspect.

The main contribution of the presented work is a novel theory for SMF-based architecture modelling and operation simulation of heterogeneous systems (consisting of hardware, software and cyberware components) in the pre-embodiment phase of development. A strictly physical view is enforced, which applies the principle of aggregation in the generation of the architecture and the operations of a system, rather than that of abstraction. For a consistent specification of the architectural and operational aspects of SMFs, AKFs and OKFs have been introduced. Their information structures are the basis of the artefactual modelling of the domains of SMFs, and the simulation of the conceivable operations in the pre-embodiment phase of system development. The proposed SMF theory complements MOT and facilitates the realisation of a computer-supported

system configuration and adaptation methodology, as well as the development of a CAD tool with an intended application in embedded customisation (Elgh 2014).

During our research we obtained some new insights. For instance, due to aggregate complexity, the information structures needed to describe the architectural and operational characteristics of SMFs cannot be anything else but complicated. For this reason, it was inevitable to introduce some level of modularity with respect to entity specification and software programming. The modularisation of the architecture and operation information structures led to a large number of information constructs, and to a multitude of relationships among them. On studying practical cases, it was recognised that composability of SMFs was strongly influenced by the number and nature of relationships. In our approach, various constraints (e.g. temporal, logical) are applied to the relationships, the combination of which has an important role in fulfilling specific composability conditions. Composability is the degree to which SMFs can be assembled in various combinations to satisfy specific system and user requirements. A composition of SMFs should meet at least the major composability conditions in order to form an architecturally and operationally correct and feasible system. However, constraints cannot be predefined exhaustively when an SMF is created, since the need for them may emerge in the composition process. For the time being, we have not investigated the solution opportunities for this issue and if any systematic method exists. The same applies to dynamic constraints management and system compositionality analysis. To increase composability of SMFs, sufficient functional and interfacing information is to be included in the specification of the related information constructs. The currently proposed solution is based on formalised input assumptions and output guaranties.

We are currently working on the computational-level detailing of the SMFs-based methodology in order to be able to transfer it to a testable prototype tool. In this first phase of the work our, efforts were concentrated on guaranteeing feasibility and achieving efficiency in terms of the procedures, methods and instruments. Furthermore, it was also our goal to make the specification, operationalisation and exploitation of a large number of SMF possible. Even though we have recognised the advantages of a parallel specification or augmentation of the proposed computational approach with an SMFs-based, user-orientated pre-embodiment design methodology, we could not accomplish it yet because of the necessary experimental work. This will be done in the next phase of our research, together with the practical investigation of utility and usability issues.

Many published papers address the idea of transdisciplinary conceptualisation of CPSs and incorporating changeability into system architectures. The presented work may have an impact in this respect since architecture or operation configuration of consumer durables by end-users and embedded customisation of human-cantered heterogeneous systems are gaining importance in current days. As argued by Fricke and Schulz (2005), 'systems to be delivered must be designed not only to meet customer or market needs, but also increasingly to meet requirements and constraints of systems sharing its operational context and throughout their entire lifecycle' and 'Flexibility, agility, robustness, and adaptability as four key aspects of changeability will be defined and described'. The methodologies and the tools used in configuration and early design of systems must provide an opportunity for upgrading the architectural components and the functional scope of system already in service, or for releasing a new version of it in a fast, reliable and economic procedure (Chandrasegaran et al. 2013). The SMF approach outlined in

this paper enables achieving flexibility, agility, robustness and adaptability in system configuration and customisation.

5.2. Conclusions and propositions

As system modelling entities, SMFs may represent both physical and computing transformations of the domains. A system model can be formed by aggregating the domains into a feasible structure and combining their UoOs into FoOs. Parameterisation of SMFs comprising HW, SW and CW constituents represents a challenge because of the associated heterogeneity and complexity issues (Merali and Mckelvey 2006). Nevertheless, a straightforward and comprehensive constraints management and satisfaction are needed. To the best of our knowledge, no underlying mathematical theory has been developed for parametrised handling of SMFs. Being aware of the difficulties, we could strive after only a partial theory supporting parameterised computational processing of SMFs. We believe that the main advantages of the SMF-based approach are as follows:

(1) Connecting previously separated areas of system design such as designing analogue hardware, digital hardware, system and application software, knowledge structures, concept ontologies, information/data models and multimedia contents.
(2) Providing a uniform system development strategy and methodology for the pre-embodiment design phase of system development, by only incrementally deviating from the currently applied technologies, methodologies and best practices.
(3) Making possible for systems designers to focus on individual elements without losing sight of the overall architecture and operation framework, and without drowning designers in cognitive burdens.
(4) Allowing dynamic modelling and easy modification of the system concept by various aggregates of domains and FoOs on a relatively high level.
(5) Making system modelling transparent for each stakeholder representing different professional fields, balancing between different views and reaching a faster agreement this way.

5.3. On-going and future research

Involving multiple Ph.D. research projects, the ultimate targets of our work are the development of a practical methodology and a supporting software tool for a *systematic* pre-embodiment design and an *embedded customisation* of CPSs (Pourtalebi, Horváth, and Opiyo 2013). An SMFs-based workbench is being developed based on the theory presented in this paper. The workbench includes a suite of tools for: (i) development of SMFs of multiple aggregation levels, (ii) management of SMFs in indexed and extendable repositories, (iii) construction of SMF structures, (iv) modification of systems by embedding SMFs, and (v) simulation of the operation and behaviour of systems in the pre-embodiment design phase of development (Wache et al. 2001).

As indicated above, our future research will focus on: (i) further theoretical underpinning of the computational processing methodology of SMFs, (ii) development of a methodology for an SMFs-based pre-embodiment design of quasi-linear CPSs, and (iii)

validation of the utility of the computational framework and the SMFs-based pre-embodiment design methodology with end-users. Two papers are being written at this moment to discuss the computational constructs that are needed for computer processing of SMF procedures and data, and to present the methodology proposed for composition of SMFs and system modelling. More specifically, the first paper will concentrate on what pieces of information should be included in computational constructs and how the transition from genotypes through phenotypes to instances of SMFs should be handled by a software toolbox, also considering the issues of knowledge engineering. The second paper will explain how the phenotypes of SMFs are composed and instantiated in the process of modelling a specific CPS. It will also address methodological issues in order to support system designers to tackle composability challenges with the aid of the SMF-based modelling toolbox.

Note

1. See http://tyromotion.com/en/products/pablo.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Shahab Pourtalebi http://orcid.org/0000-0003-3482-5492
Imre Horváth http://orcid.org/0000-0002-6008-0570

References

Abdul-Ghafour, S., P. Ghodous, B. Shariat, E. Perna, and F. Khosrowshahi. 2014. "Semantic Interoperability of Knowledge in Feature-based CAD Models." *Computer-Aided Design* 56: 45–57.

Altidor, J., J. Wileden, Y. Wang, L. Hanayneh, and Y. Wang. 2009. "Analyzing and Implementing a Feature Mapping Approach to CAD System Interoperability." Proceedings of ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, San Diego.

Asher, N., and L. Vieu. 1995. "Toward a Geometry of common sense: A semantics and a complete axiomatization of mereotopology." *IJCAI* (*1*) Citeseer, 846–852.

Baldwin, C., and K. Clark. 2006. "Modularity in the Design of Complex Engineering Systems." In *Complex Engineered Systems*, edited by D. Braha, A. A. Minai, and Y. Bar-Yam, 175–205. Berlin: Springer.

Bar-Yam, Y. 2004. "Multiscale Variety in Complex Systems." *Complexity* 9 (4): 37–45.

Benhamou, F., N. Jussien, and B.A. O'sullivan. 2013. *Trends in Constraint Programming*. West Sussex: Wiley-ISTE.

Bidarra, R., and W. F. Bronsvoort. 2000. "Semantic Feature Modelling." *Computer-Aided Design* 32 (3): 201–225.

Bjørner, D., and A. Eir. 2010. "Compositionality: Ontology and mereology of domains." In *Concurrency, Compositionality, and Correctness*. Lecture Notes on Computer Science, 5930. Berlin Heidelberg: Springer, 22–59.

Borgo, S., N. Guarino, and C. Masolo. 1996. "A Pointless Theory of Space Based on Strong Connection and Congruence." *KR* 96: 220–229.

Brunetti, G., and S. Grimm. 2005. "Feature Ontologies for the Explicit Representation of Shape Semantics." *International Journal of Computer Applications in Technology* 23 (2–4): 192–202.
Chandrasegaran, S. K., K. Ramani, R. D. Sriram, I. Horváth, A. Bernard, R. F. Harik, and W. Gao. 2013. "The Evolution, Challenges, and Future of Knowledge Representation in Product Design Systems." *Computer-Aided Design* 45 (2): 204–228.
Chen, G., Y. S. Ma, G. Thimm, and S. H. Tang. 2006. "Associations in a Unified Feature Modeling Scheme." *Journal of Computing and Information Science in Engineering* 6 (2): 114–126. doi:10.1115/1.2194910.
Da Silveira, G., D. Borenstein, and F. S. Fogliatto. 2001. "Mass Customization: Literature Review and Research Directions." *International Journal of Production Economics* 72 (1): 1–13.
De Micheli, G. 1996. "Hardware/Software Co-design: Application Domains and Design Technologies." In *Hardware/Software Co-design*, edited by G. De Micheli and M. Sami, 1–28. Dordrecht: Springer.
Demoly, F., X.-T. Yan, B. Eynard, L. Rivest, and S. Gomes. 2011. "An Assembly Oriented Design Framework for Product Structure Engineering and Assembly Sequence Planning." *Robotics and Computer-integrated Manufacturing* 27 (1): 33–46.
Doboli, A., A. Umbarkar, V. Subramanian, and S. Doboli. 2014. "Two Experimental Studies on creative Concept Combinations in Modular Design of Electronic Embedded Systems." *Design Studies* 35 (1): 80–109.
Eckert, C., P. J. Clarkson, and W. Zanker. 2004. "*Change and Customisation in Complex Engineering Domains*." *Research in Engineering Design* 15 (1), 1–21.
Eidson, J. C., E. Lee, S. Matic, S. Seshia, and J. Zou. 2012. "Distributed Real-time Software for Cyber–physical Systems." *Proceedings of the IEEE* 100 (1): 45–59.
Elgh, F. 2014. "Automated Engineer-to-order Systems – A Task-oriented Approach to Enable Traceability of Design Rationale." *International Journal of Agile Systems and Management* 7 (3–4): 324–347.
Fedder, G. K., R. T. Howe, T.-J. K. Liu, and E. P. Quevy. 2008. "Technologies for Cofabricating Mems and Electronics." *Proceedings of the IEEE* 96 (2): 306–322.
Fricke, E., and A. P. Schulz. 2005. "Design for Changeability (DFC): Principles to Enable Changes in Systems Throughout Their Entire Lifecycle." *Systems Engineering* 8 (4): 342–359.
Gavrilescu, M., G. Magureanu, D. Pescaru, and A. Doboli. 2010. "Accurate Modeling of Physical Time in Asynchronous Embedded Sensing Networks." Intelligent Systems and Informatics (SISY), 2010 8th International Symposium on IEEE, 477–482.
Gerritsen, B. H., and I. Horváth. 2015. "Advancements in Advanced Modelling of Complex Products and Systems." *Engineering Computations: International Journal for Computer-Aided Engineering and Software* 32 (1): 1–4.
Goldratt, E. M. 1990. *Theory of Constraints*. New York, NY: North River Press.
Horváth, I., and B. Gerritsen. 2012. "Cyber-physical Systems: Concepts, Technologies and Implementation Principles." Proceedings of the International Tools and Methods of Competitive Engineering Symposium, Karlsruhe, Germany: Delft University of Technology, 19–36.
Horváth, I., and B. Gerritsen. 2013. "Outlining Nine Major Design Challenges of Open, Decentralized, Adaptive Cyber-physical Systems." Proceedings of ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR.
Horváth, I., and S. Pourtalebi. 2015. "Fundamentals of a Mereo-Operandi Theory to Support Transdisciplinary Modelling and Co-design of Cyber-physical Systems." Proceedings of ASME 2015 International Design Engineering Technical Conferences, Boston, MA.
Horváth, I., J. Pulles, A. Bremer, and J. Vergeest. 1998. "Towards an Ontology-based Definition of Design Features." Proceedings of the Workshop on Mathematical Foundations for Features in Computer Aided Design, Engineering, and Manufacturing, SIAM, 1–12.
Hotz, L., A. Felfernig, A. Günter, and J. Tiihonen. 2014. "A Short History of Configuration Technologies." In *Knowledge-based Configuration – From Research to Business Cases*. San Francisco, CA: Morgan Kaufmann, 9–19.
Kim, O., U. Jayaram, S. Jayaram, and L. Zhu. 2009. "An Ontology Mapping Application Using a Shared Ontology Approach and a Bridge Ontology." Proceedings of ASME 2009 International Design

Engineering Technical Conferences and Computers and Information in Engineering Conference, San Diego.
Kim, K.-Y., D. G. Manley, and H. Yang. 2006. "Ontology-based Assembly Design and Information Sharing for Collaborative Product Development." *Computer-Aided Design* 38 (12): 1233–1250. http://www.sciencedirect.com/science/article/pii/S0010448506001680.
Kim, K.-Y., and Yang, H., 2008. "The Role of Mereotopology and SWRL Rules to Represent Joint Topology Information for Design Collaboration." Proceedings of ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 131–139.
Kim, K.-Y., H. Yang, and D.-W. Kim. 2008. "Mereotopological Assembly Joint Information Representation for Collaborative Product Design." *Robotics and Computer-Integrated Manufacturing* 24 (6): 744–754. http://www.sciencedirect.com/science/article/pii/S0736584508000367.
Kurtoglu, T., I. Tumer, and D. Jensen. 2010. "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures." *Research in Engineering Design* 21 (4): 209–234. doi:10.1007/s00163-010-0086-1.
Liang, H., X. Nannan, K. Zhejun, and Z. Kuo. 2012. "Review of Cyber-physical System Architecture." Proceedings of 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), Shenzhen, Guangdong.
Liu, G., F. Yang, X. Bao, and T. Jiang. 2015. "Robust Optimization of a Mems Accelerometer Considering Temperature Variations." *Sensors* 15 (3): 6342–6359.
Lyshevski, S. E. 2002. *Mems and Nems: Systems, Devices, and Structures*. Boca Raton, FL: CRC Press.
Mcgrenere, J., and W. Ho. 2000. "Affordances: Clarifying and Evolving a Concept." *Graphics Interface*, Montreal, Canada.
Merali, Y., and B. Mckelvey. 2006. "Using Complexity Science to Effect a Paradigm Shift in Information Systems for the 21st Century." *Journal of Information technology* 21 (4): 211–215. doi:10.1057/palgrave.jit.2000082.
Oreizy, P., M. Gorlick, R. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, and A. L. Wolf. 1999. "An Architecture-based Approach to Self-adaptive Software." *IEEE Intelligent Systems*, 14 (3): 54–62.
Ostrosi, E., and Ferney, M., 2007. "Fuzzy Product Configuration in Advanced CAD Systems." In *Digital Enterprise Technology,* edited by P. Cunha and P. Maropoulos, 225–232. New York, NY: Springer.
Pourtalebi, S., I. Horváth, and E. Opiyo. 2013. "Multi-aspect Study of Mass Customization in the Context of Cyber-physical Consumer Durables." Proceedings of ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR.
Pourtalebi, S., I. Horváth, and E. Z. Opiyo. 2014a. "First Steps Towards a Mereo-Operandi Theory for a System Feature-based Architecting of Cyber-physical Systems." In *INFORMATIK 2014, Big Data – Komplexität Meistern*, edited by E. Plödereder, L.G., E. Schneider, and D. Ull (Hrsg.), 2001–2006. Stuttgart: GI.
Pourtalebi, S., I. Horváth, and E. Opiyo. 2014b. "New Features Imply New Principles? Deriving Design Principles for Mass Customization of Cyber-physical Consumer Durables." Proceedings of the TMCE, Budapest, Hungary, 95–108.
Shah, J.J. 1991. "Assessment of Features Technology." *Computer-Aided Design* 23 (5): 331–343.
Stjepandić, J., E. Ostrosi, A.-J. Fougères, and M. Kurth. 2015. "Modularity and Supporting Tools and Methods." In *Concurrent Engineering in the 21st Century*, 389–420. Cham, Switzerland: Springer International Publishing.
Syaimak, A., and D. Axinte. 2009. "An Approach of Using Primitive Feature Analysis in Manufacturability Analysis Systems for Micro-milling/Drilling." *International Journal of Computer Integrated Manufacturing* 22 (8): 727–744.
Sztipanovits, J. 2012. "Cyber Physical Systems—Convergence of Physical and Information Sciences." *It – Information Technology Methoden und innovative anwendungen der informatik und information-stechnik* 54 (6): 257–265.

Tan, Y., M. C. Vuran, and S. Goddard. 2009. "Spatio-temporal Event Model for Cyber-physical Systems." Proceeding of IEEE International Conference on Distributed Computing Systems Workshops, 44–50.

Tessier, S., and Y. Wang. 2013. "Ontology-based Feature Mapping and Verification Between CAD Systems." *Advanced Engineering Informatics* 27 (1): 76–92.

Tiihonen, J., T. Lehtonen, T. Soininen, A. Puikkinen, R. Sulonen, and A. Riitahuhta. 1998. "Modeling Configurable Product Families." *Modularity In Use-Experiences from Five Companies*. Proceedings of the 4th WDK Workshop on Product Structuring, October 22–23, 1998. Delft: Delft University of Technology, 1–22.

Tiihonen, J., T. Soininen, and R. Sulonen. 1996. "State of the Practice in Product Configuration – A Survey of 10 Cases in the Finnish Industry." *Knowledge Intensive CAD* 1: 95–114.

Wache, H., T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. 2001. "Ontology-based Integration of Information – A Survey of Existing Approaches." Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, WA: Citeseer, 108–117.

Wang, K., Y. Li, and C. Rizos. 2009. "The Effect of the Temperature-correlated Error of Inertial Mems Sensors on the Integration of GPS/INS." Proceedings of the Symposium of the International Global Navigation Satellite Systems Society, Australia: IGNSS.

Wolf, W.H. 1994. "Hardware-software Co-design of Embedded Systems [and Prolog]." *Proceedings of the IEEE* 82 (7): 967–989.

Xie, Y., and Y. Ma. 2015. "Design of a Multi-disciplinary and Feature-based Collaborative Environment for Chemical Process Projects." *Expert Systems with Applications* 42 (8): 4149–4166.