Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Information schema constructs for defining warehouse databases of genotypes and phenotypes of system manifestation features

Shahab POURTALEBI‡, Imre HORVÁTH

(*Faculty of Industrial Design Engineering, Delft University of Technology, Landbergstraat 15, 2628 CE Delft, Zuid Holland, the Netherlands*)

E-mail: s.pourtalebihendehkhaleh@tudelft.nl; i.horvath@tudelft.nl

Received Feb. 15, 2016; Revision accepted May 22, 2016; Crosschecked Aug. 8, 2016

Abstract: Our long-term objective is to develop a software toolbox for pre-embodiment design of complex and heterogeneous systems, such as cyber-physical systems. The novelty of this toolbox is that it uses system manifestation features (SMFs) for transdisciplinary modeling of these systems. The main challenges of implementation of the toolbox are functional design- and language-independent computational realization of the warehouses, and systematic development and management of the various evolving implements of SMFs (genotypes, phenotypes, and instances). Therefore, an information schema construct (ISC) based approach is proposed to create the schemata of the associated warehouse databases and the above-mentioned SMF implements. ISCs logically arrange the data contents of SMFs in a set of relational tables of varying semantics. In this article we present the ISCs necessary for creation of genotypes and phenotypes. They increase the efficiency of the database development process and make the data relationships transparent. Our follow-up research focuses on the elaboration of the SMF instances based system modeling methodology.

Key words: Cyber-physical systems, Software toolbox, Pre-embodiment design, System manifestation features (SMFs), Warehouses, Database schemata, SMF genotypes, SMF phenotypes, SMF instances, Information schema constructs
http://dx.doi.org/10.1631/FITEE.1600997 **CLC number:** TP391; TP311

1 Introduction

The objective of our research project is to develop a software toolbox for pre-embodiment design of complex and heterogeneous systems, such as cyber-physical systems (CPSs) (Hu, 2013). The novelty of this toolbox is that it uses system-level features (SLFs) to achieve an efficient transdisciplinary modeling of CPSs. In addition to the development of a robust theoretical framework for this toolbox, our research intends to convert the concepts and patterns of pre-embodiment design of CPSs into novel and efficient multidisciplinary modeling procedures and computational objects. This explains why an SLF-based pre-embodiment design approach has been considered. As specific design enablers and computational objects, system manifestation features (SMFs) have been devised (Pourtalebi and Horváth, 2016a). From now on, we use the acronym SMF-TB to refer to the proposed SMF-based toolbox. SMF-TB integrates two major constituents: (1) the platform, which supports creation and warehousing of SMFs, and (2) the workbench, which allows instantiation of SMFs and composition of system models. Its implementation is a multifaceted and effort-intensive undertaking that could be executed only in multiple phases and steps. Hence, our research and development work has been advancing through the phases of creating (1) a comprehensive theoretical framework, (2) an overall

‡ Corresponding author
ORCID: Shahab POURTALEBI, http://orcid.org/0000-0003-3482-5492; Imre HORVÁTH, http://orcid.org/0000-0002-6008-0570

information structure, (3) a computational methodology, and (4) a testable prototype. Each subsequent phase relies on the results achieved in the previous phases.

The concept of feature-based design is known for more than 30 years in various fields of computer-aided design and engineering (Shah and Rogers, 1988). According to the classical definition of Pratt (1991), a feature is a region of interest on the surface of a part. Henderson and Anderson (1984) interpreted features as geometric and topological patterns of interest in a part model, which represent high-level entities useful in part analysis. Feature technology provided an adequate basis for integration of design and other activities, such as engineering analysis, process planning, machining, and inspection (Case and Gao, 1993). The progress of feature technology is well documented in the literature (Parry-Barwick and Bowyer, 1993; Salomons *et al.*, 1993). Various semantic feature modeling approaches (Bidarra and Bronsvoort, 2000) and languages were constructed for different applications (Au and Yuen, 2000). Based on their semantics, feature-based technologies proved to be not only useful but also effective means of supporting the collaborative work of distributed multidisciplinary groups (Wang *et al.*, 2003). Efforts have been made for the standardization of features and technological approaches (VDI, 2003), making feature libraries exchangeable among commercial CAD/E/PP/M systems based on feature ontologies (Abdul-Ghafour *et al.*, 2014). However, there are some theoretical, methodological, and computational limitations that traditional feature technologies could not eliminate (Zhang *et al.*, 2016).

Right after the millennium, the idea of extending feature technology to domains other than mechanical engineering (e.g., to digital hardware and software development) has popped up has popped up (Zha *et al.*, 2005). In software engineering, a feature is defined as an increment in program functionality (Czarnecki and Eisenecker, 2000). The basic idea of feature-oriented software development is to decompose a large-scale software system in terms of the features it provides for satisfying a requirement, representing a design decision, and providing a potential configuration option (Apel and Kästner, 2009). The issue of distinction between problem domain features and features on the implementation level was raised in the context of feature interaction in composed software systems (Pulvermueller *et al.*, 2002). According to Czarnecki *et al.* (2005), a feature is a system property that is relevant to some stakeholder and is used to capture commonalities or discriminate among systems in a family. This definition does not differentiate between paradigmatic system features and system manifestation features. Batory (2005) observed that despite progress, "tools for feature models often seem ad hoc; they exhibit odd limitations and provide little or no support for debugging feature models". He argued that "this is to be expected when a fundamental underpinning of feature models is lacking".

As one of the beneficial application fields, physics-of-failure prognostic with system-level feature models has been proposed (Kacprzynski *et al.*, 2002). In the mechanical domain, a typical example of system-level features is system-level vibration features, which are usually associated with failure models of transmission systems. Theoretical and pragmatic issues such as using feature technology, semantic feature models, and feature ontology for embedded system design and development were also studied (Zha and Sriram, 2006). In the context of monitoring adaptive systems, a nonconventional interpretation of features has been proposed and four dimensions of using features were identified: (1) trying new features, (2) feature substitution, (3) feature combination, and (4) feature repurposing (Sun and Zhang, 2008). System-level features were considered in modeling information systems at the transaction level (Schirner *et al.*, 2010). Hardware architecture features, such as instruction count, cache miss, and bus traffic, have been used for sophisticated denial-of-service attack detections (Tao *et al.*, 2009). Recently, issues of feature-based data exchange have become popular for design and manufacturing (Wu *et al.*, 2015).

In the first part of the next section, the underpinning concepts of our work, the most important definitions, and some highlights of the completed work are briefly discussed. Unfortunately, many important technical details, such as the specific contents of the foundational knowledge frames, cannot be presented due to space limitation. Interested readers are encouraged to study our previous publications for further information (Horváth and Pourtalebi, 2015;

Pourtalebi and Horváth, 2016a; 2016b). Afterward, in Section 3, the procedural and information management aspects of creation and application of SMFs are addressed. Also, in this section, the information structures needed for creation of genotypes and phenotypes of SMFs are presented. Section 4 tries to give a detailed explanation on the information schema constructs (ISCs) designed for genotypes. Section 5 discusses the chunks of information and the ISCs designed for phenotypes. In Section 6, we reflect on the work and the results achieved so far.

2 From the theoretical framework to information schema constructs

2.1 Foundational concepts and definitions

The central concept of the reported work is SMF, which is a particular implementation of system-level features for the purpose of pre-embodiment design of complex and heterogeneous systems, such as CPSs. The significance of system-level features is that they imply specific design principles that are to be considered in the design process of the systems. As opposed to a paradigmatic characteristic, which concerns the purpose and functionality of the whole of an artifact, SMFs are defined to capture syntagmatic characteristics, resulting in particular materialization and implementation of a part of the whole.

SMFs are very high-level, transdisciplinary, adaptive, and smart modeling entities. They not only capture architectural and operational semantics but also smartly assist the collaboration of disciplinary designers in design time. They are considered finite regions of a system that have significance from both architectural and engineering operation (modus operandi) points of view (Pourtalebi *et al.*, 2014a; 2014b). They represent both logically based and physically based abstraction of the concerned region, and may be defined on various aggregation levels.

An information structure (IS) is a representation of some existing or conceived reality with different characteristics which are made explicit (Lee *et al.*, 2006). From a computational perspective, it is a pragmatic arrangement of data, describing the concepts and associations among them (Shenton and Hayter, 2006). The relationships between data can be either structural links or semantic links (Grzybek *et al.*, 2014). An IS captures units of information that lose all usefulness if broken down any further or taken out from the specified relationships. ISs can be arranged in compound structures such as linear, hierarchical, network or graph, and matrix.

The term 'construct' is frequently used in software engineering in the meaning of language constructs rather than as data organization or function operationalization constructs. In our reasoning, information schema constructs (ISCs) are regarded as blueprints for coding of the SMFs as software components and implementing warehouse databases. ICSs manage both the required chunks of information and their relationships. They are derived based on an overall operational scenario and an all-inclusive information structure. ISCs are designed to be logical content arrangements imposed over a set of relational database tables of varying semantics.

2.2 Preliminaries of the presented research

This research builds on and extends the results of preliminary research activities that have been conducted according to the workflow presented in Fig. 1. In the two preceding phases of our research, two underpinning theories have been developed. The first one, called Mereo-Operandi theory (MOT), supports transdisciplinary modeling and codesign of CPSs (Horváth and Pourtalebi, 2015). This theory suggests that it is necessary and sufficient to describe and characterize CPSs from both architectural and operational perspectives and that these can be achieved by integrating the principles and means of spatio-temporal mereotopology and those of engineering

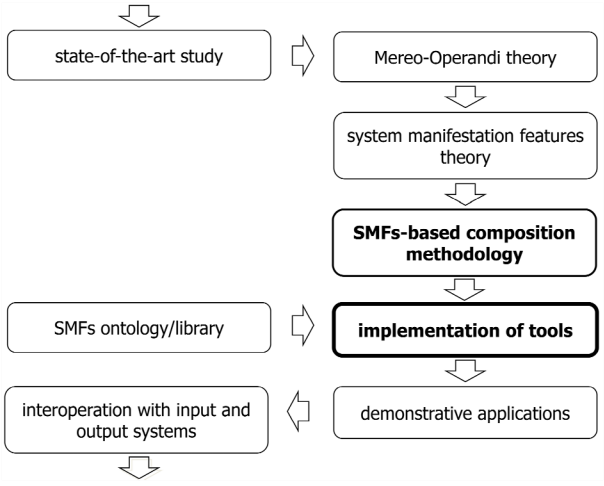

Fig. 1 Focus of the research reported in this paper

modus operandi (i.e., multi-physics). The second theory, called the theory of system manifestation features, creates an epistemological basis and a knowledge engineering framework for SMFs-based pre-embodiment design of CPSs (Pourtalebi and Horváth, 2016a). The proposal of this theory is that (1) every complicated heterogeneous system can be efficiently captured and uniquely modeled by a purposeful composition of SMFs, and (2) architecture and operation models of a CPS can be generated simultaneously and synergistically.

2.3 Some highlights of the presented work

As stated by Romero *et al*. (2015), the use of databases in computer systems has been on the rise in recent decades, and the computer community has worked intensively on systems using complex databases. Efforts have also been made toward schemata and code builders that provide the users with higher level semantics. Our current research focuses on the specification of the conceptual constructs that are needed for warehouse databases of SMFs. Methodologically, a multistage mapping should be supported, which involves (1) creation and storing of genotypes of SMFs in the warehouse, (2) derivation of specific phenotypes from genotypes, and (3) instantiation of SMFs based on the phenotypes in the course of structural and operational modeling of a particular CPS. A genotype determines the overall physical constitution (composition or makeup) of an artifact as distinguished from its physical and visual traits. A phenotype is a specific manifestation of a genotype in terms of its architectural and operational parameterization. An instance is a particular (evaluated) data representation of an element of a system, which satisfies the specified constraints.

In the process of system modeling, phenotypes are combined and adapted to each other in order to generate matching feature instances that a system model is composed of. An actual SMF instance is the result of this parametric matching/evaluation procedures. The originality (also uniqueness) of our work comes from the introduction of the concepts of genotype and phenotype (and coupled instances) as mediators in the SMFs-based modeling process. Implementation of genotypes and phenotypes and the hosting warehouse databases is based on the ISCs that facilitate structural and morphological/attributive parameterization of SMFs as computational modeling entities.

The advantages of using SMFs are in their transdisciplinary, comprehensive, and semantically rich nature, as well as in their potential of smartly supporting disciplinary collaboration. SMFs implement a synergetic processing of structural, architectural, morphological, physical, logical, operational, etc. chunks of information, and for this reason they need specific computational concepts and a systematic procedure. In a higher resolution, the information structures describing SMFs contain numerous compound entities and relationships, whose computational representation and processing necessitate a large number of ISCs. Being application-independent, SMFs can be classified based on their functionality and included in open libraries. SMFs as modeling entities are much more complex than the traditional part features. Therefore, implementation of an SMF-TB is much more demanding than that of a traditional feature-based part and assembly modeler.

3 Procedural aspects of processing SMFs

3.1 Workflow of creating SMFs

Prior to building a system's model, first the functional building blocks (i.e., SMFs) are created and specified from both architectural and operational aspects. We refer to this as 'SMF creation'. From a procedural perspective, the SMF creation process is preceded by the specification of feature concept ontology, as shown in Fig. 2 (Pandit and Honavar, 2010). Though this is the very first step of SMFs creation, we regarded this as a preparatory part of the process and did not specifically address it in our research completed so far.

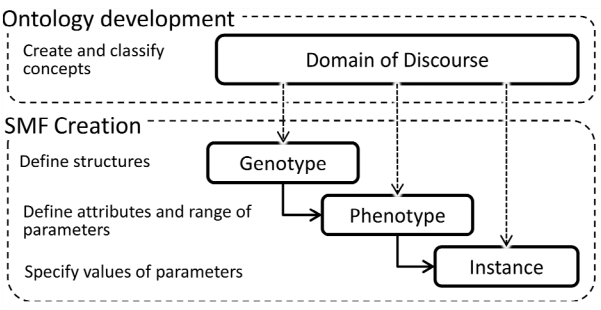

Fig. 2 Phases of system manifestation feature creation

We identified the major requirements for ontological classification of SMFs, and declared our assumptions concerning the contents to be provided by an SMF concept ontology. The necessary ontological concepts have been considered in our research, and the primary relations between SMFs and ISCs and the assumed ontologies and ISCs have been determined.

As discussed above, the SMF-based design process involves multiple transformations of information structures toward a system model. The database schemata of the genotype and the phenotype templates are derived from the combined information structures of architecture and operation knowledge frames, known as AKF and OKF, respectively (Pourtalebi and Horváth, 2016a). Every genotype lends itself to defining multiple different structural variants of an SMF. One of these structural variants is exemplified in a phenotype.

Phenotypes have user-defined morphological and attributive parameter values, in addition to those inherited from a genotype. Accordingly, to define a phenotype, additional chunks of information concerning the morphological and material properties and attributes are needed, as well as information about events, conditions, and constraints. The SMF creation task of knowledge engineers is supported by genotype editor and phenotype editor tools, which belong to the platform part of SMF-TB and will be explained in Section 3.3.

3.2 Workflow of modeling using SMFs

The overall workflow of conceptual modeling of a CPS with SMF-TB includes the following parts: (1) retrieval of SMF-phenotypes as modeling entities, (2) specification of coupling among phenotypes according to the need of the application at hand, (3) composition of the specified phenotypes into the system model, (4) instantiation of SMF-phenotypes as instances in the system model, (5) modification of the generated system model by adding, removing, or altering SMF instances, (6) validity analysis and storage of the system model, (7) placing the system model into application context by specifying operational parameters, and (8) simulation of the operation of the system model.

Instantiation of an SMF means deriving specific instances based on architectural (i.e., morphological) and operational coupling of phenotypes through their interfaces. To match the interfaces of phenotypes, the predefined input assumptions and output guaranties are used for architectural coupling. On the operational side, matching the interfaces of phenotypes is provided as event unification, state unification, and stream unification. These are the subjects of our follow-up paper (Pourtalebi and Horváth, 2016c). The parameterized nature of phenotypes allows adapting their parameter values to facilitate interfacing. The combination and numerical evaluation of phenotypes results in instances of SMFs, a composition of which makes up the evaluated system model.

3.3 Overall architecture of SMF-TB

The MOT and SMF theories have been used as a starting platform for the work presented in this paper. Actually, they formed a conceptual framework of the information technological specification and computational implementation of the SMFs-based preembodiment design methodology and toolbox. As shown in Fig. 3, the tools are integrated in two fundamental parts of SMF-TB: platform and workbench. The platform includes those software modules (seen as tools) for specification, warehousing, and retrieval

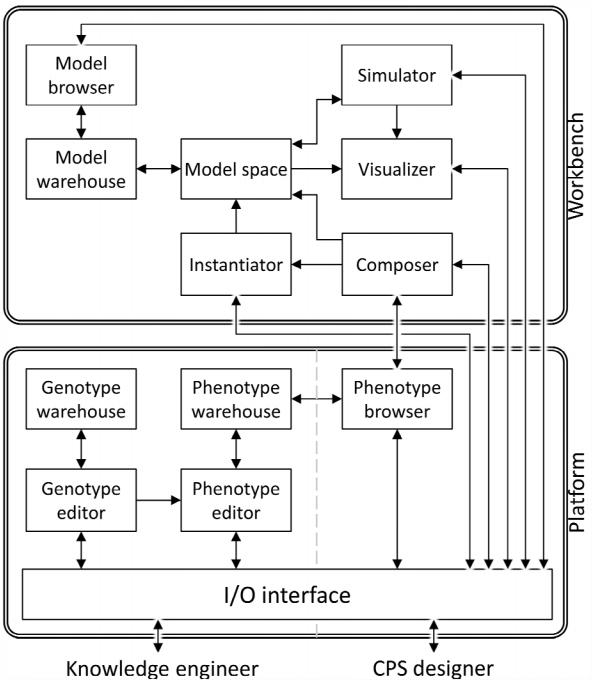

Fig. 3 Overall architecture of SMF-TB

of SMF implements. The workbench includes the tools for composition, instantiation, visualization, and simulation of SMF-based models of CPSs.

The overall architecture of SMF-TB and the specific software tools included in the platform part and in the workbench part are shown in Fig. 3. The platform provides user interfaces for both roles of knowledge engineer and CPS designer. At the backend of the platform, there are two specific warehouse databases for managing SMFs, one for genotypes and the other for phenotypes. SMF phenotypes can be specified directly by the phenotype editor based on an appropriate genotype, or alternatively searched for and retrieved from the phenotype warehouse repository.

In the workbench part of the toolbox, the composer tool checks for the constraints and the interface contracts of SMF phenotypes, as well as the architectural and operational relations between them. The instantiator tool assigns values to the parameters of phenotypes. The system model is built in the model space by coupling and instantiating selected phenotypes. The model data and relations are saved and stored in the database of the model warehouse. There is a model browser included that can be used to browse the saved models, to retrieve relevant ones, and to invoke the data belonging to the chosen model or SMFs thereof. The visualizer module, like a CAD system, produces visual images of architecture views, operation views, and simulation views. The simulator tool simulates the physical and computational operations of the system model according to predefined scenarios. Its functionality facilitates time-dependent analyses and simulations of the system model.

3.4 Chunks of information needed for defining genotypes and phenotypes of SMFs

The information structures underpinning the definition of genotypes and phenotypes are derived based on the chunks of information contained by the AKF and OKF (Pourtalebi and Horváth, 2016a). The interrelationships of the chunks of information needed for creating genotypes and phenotypes are summarized in Fig. 4. In this figure, the boxes with underlined texts contain ontological chunks of information, and the boxes whose names start with GT and PT contain chunks of information describing the genotype and phenotype, respectively. Below, we first focus on the chunks of information and the information schema constructs designed for creating genotypes, and then direct our attention to those designed for creating phenotypes.

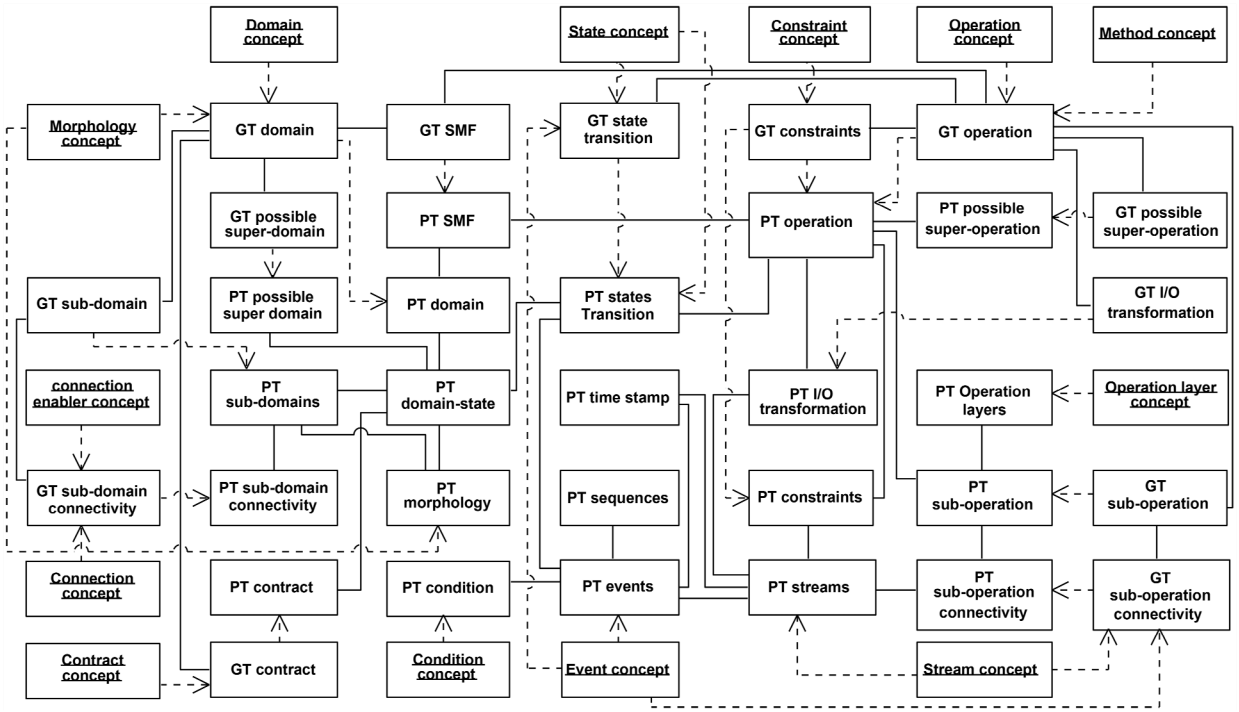

Fig. 4 Interrelationships of the chunks of information required for creation of genotypes and phenotypes

4 Information schema constructs for creating genotypes

4.1 Chunks of information needed for defining genotypes of SMFs

The ISCs of a genotype use the following ontological tables: (1) domain concepts, (2) morphology concepts, (3) connection concepts, (4) connection enabler concepts, (5) contract concepts, (6) stream concepts, (7) state concepts, (8) event concepts, and (9) operation concepts. The information sets taken over from an AKF are as follows:

Metadata:
<kind_of_possible_super_domain>;
<kind_of_domain>;
<description_of_domain>;
<associated_operation>;
<morphological_kind>;
<representative_state>

Domain entities:
<GT_entity_name>;
<GT_entity_description>

Entity types:
<kind_of_entity>

Entity morphologies:
<kind_of_entity_morphology>

Containment relations:
<list_of_GT_entities>

Connectivity relations:
<kind_of_connection>;
<connected_entities>;
<kind_of_connection_enabler>

Contracts:
<kind_of_contract>

The information sets taken over from an OKF are as follows:

Metadata:
<kind_of_possible_super_operation>;
<kind_of_operation>;
<description_of_GT_operation>;
<related_GT_domain>

States of domain:
<kind_of_start_state>;
<kind_of_end_state>;
<start_event>;
<end_event>

Streams:
<kind_of_external_stream>;
<kind_of_internal_stream>

Events:
<kind_of_event>

Transformative procedure:
<associated_stream>;
<stream_direction>

Methods of FoO:
<nil>

Time stamps:
<nil>

Scheduling conditions:
<nil>

Operational constraints:
<nil>

UoOs:
<name_of_GT_UoO>;
<description_of_GT_UoO>;
<kind_of_UoO>

Domain entities of UoOs:
<UoO_associated_entity>

Methods of UoOs:
<nil>

Operational layers:
<nil>

4.2 Constructs for defining the warehouse database scheme of genotypes

From the viewpoint of organizing data, genotypes are structures formed by entity-relationship tables (ERTs). The structure of ERTs can be depicted by graphical ERT diagrams such as the one shown in Fig. 5. Each rounded rectangle represents an entity table in the warehouse database. Contents of the rows of the tables in these diagrams are regarded as names of columns of tables in the database that can accommodate records as their rows. There are relations between the entity tables, as well as between the

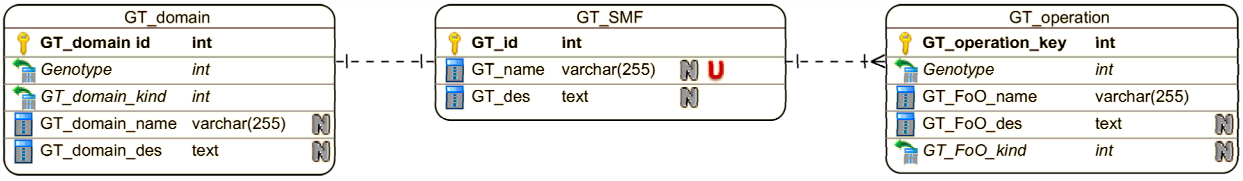

Fig. 5 Construct for arranging the primary relational tables for genotypes of SMFs

information entities. A relation needs to be specified if dependency coupling or semantic association exists between two (chunks of) information entities.

There are primary keys ( ) and foreign keys ( ) included in the tables. The primary keys (PKs) are unique identifiers of each row, and they also identify the records in the respective tables. It is possible to connect two tables by repeating the PKs of one table in another one as foreign keys (FKs). Based on PKs, contents of the rows of tables can be related to the records of other tables. PKs can be repeated once or multiple times in another table. Repeating PKs (which become FKs in that table) implies that multiple relations have been established between the tables.

The established relations do not indicate the direction of dependence per se—it needs semantic interpretation. The relationships between the entity tables are included in the main table of the genotype, denoted as GT_SMF. This establishes connections between the domain (GT_domain) and operation (GT_operation) tables (Fig. 5). There is a one-to-one relation between GT_SMF and GT_domain, and a one-to-many relation between GT_SMF and GT_operation.

A genotype is identified by its unique name and a short description. The PK of the GT_SMF table shown in Fig. 5 is a unique identifier GT_id. To facilitate a proper categorization of genotypes and prevent misinterpretation, the GT_name parameter does not allow entering similar names for multiple genotypes (i.e., accepts only unique varchar values). As FK, GT_domain_kind is imported from the domain concept table. GT_FoO_kind is imported from the operation concept table of the concept ontology.

The database management supports handling operational containment relations. Three levels of operations are defined: (1) super-operation, (2) flow of operation (FoO), and (3) unit of operation (UoO) (Fig. 6). FoO specifies the overall operations of the genotype, while UoOs describe the specific operations of components included in the genotype. The term 'super-operation' is used to refer to possible higher-level operation, of which an FoO of the genotype is a part. The information specifying these levels is included in the GT_possible_super_operation, GT_operation, and GT_UoO tables. They capture all chunks of information associated with the three operation levels.

UoOs can be aggregated in different compositions to form FoOs. What it means from a database management point of view is that it should support linking many UoOs to one particular FoO, and vice versa. UoOs are represented in the GT_UoO table, while FoOs are included in the GT_operation table. In the case of a many-to-many relation between them, an association table, called 'GT_operation_containment', establishes mutual relationships. The FKs indicated by the '-kind' suffix are imported from the operation concept table. In the corresponding relational data tables (RDTs) of the FoO and UoO tables, the 'name' and 'descriptions' specify the respective columns.

Representation of operational connectivity is an important aspect of organizing the database schemata. The reason is that the procedure of an FoO is established by procedural elements (UoOs) and their connections. The table GT_UoO_connectivity (shown in the middle of Fig. 7) is the basis of the related schemata. This table accommodates on the corresponding FKs. In the physical world, connections between UoOs are made through energy, information, and material streams. Therefore, the FK GT_stream_kind is used to refer to the stream connecting two UoOs. The FK GT_event_kind refers to the kind of event that triggers the connecting stream.

This schema identifies the sender UoO as well as the receiver UoO, in order to be able to capture both the source and the sink of the stream. Correspondingly, the FKs are imported from the GT_UoO table and included in the associated RDTs. These FKs are nullable, because of the possibility of having either

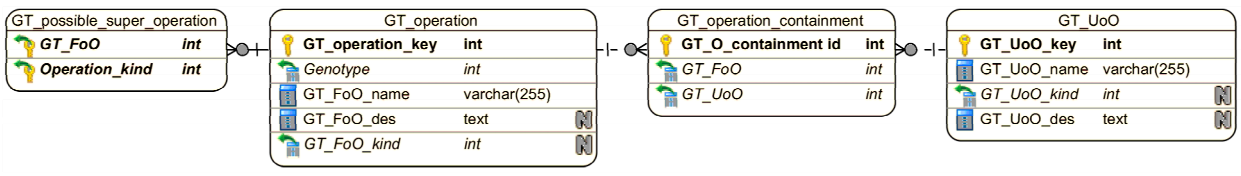

Fig. 6 Construct for arranging relational tables for capturing super-operation, operation, sub-operation, and operation connectivity by genotypes

incoming or outgoing streams. The FK FoO refers to a higher-level operation, in which the captured connection is a part of the procedure. We note that the FKs GT_event_kind and GT_stream_kind are supposed to be imported from ontological tables event_concept and stream_concept, respectively.

Each operation performed by an SMF is formally defined as a transformation. A physical transformation can be a state transition and an input-to-output transformation. Transformations operate on input and output streams. The construct that specifies the transformations is called I/O_transformation, and it creates relations with the genotypes' database tables. These relations are graphically shown in Fig. 8. The GT_IO_transformation table establishes one-to-many relations with the GT_input_stream and GT_output_stream tables. These tables assign kinds of streams as inputs and outputs to the transformation. The GT_stream_kind represents the FKs imported from the stream_concept table. The one-to-many relations are needed here because, from a physical viewpoint, one single operation may transform (convert) multiple input streams into multiple output streams.

The states of operational domains are changed by transformations. Physically, a start state of a domain is transformed to an end state as a result of completion of FoOs. A genotype describes a state change by considering two events as the start and the end of the state transition. For the computational realization the state_kind and event_kind concepts are imported as FKs from the state concept table and the event concept table. The construct specifying the relations between the domain and the operations is shown in Fig. 9. There is a many-to-many relation

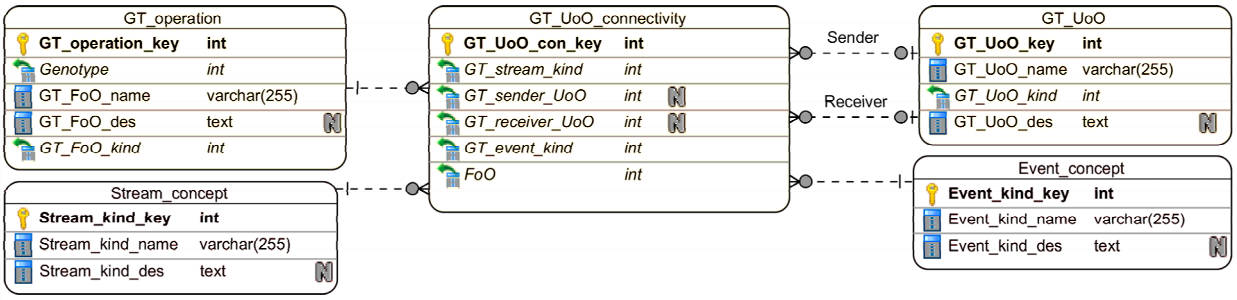

Fig. 7 Construct for arranging relational tables for capturing operation connectivity by genotypes

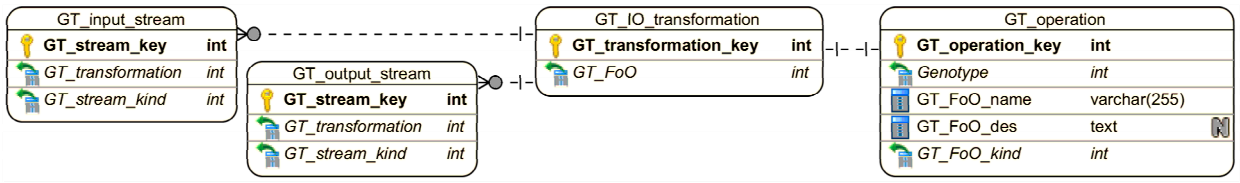

Fig. 8 Construct for arranging relational tables for capturing I/O transformations by genotypes

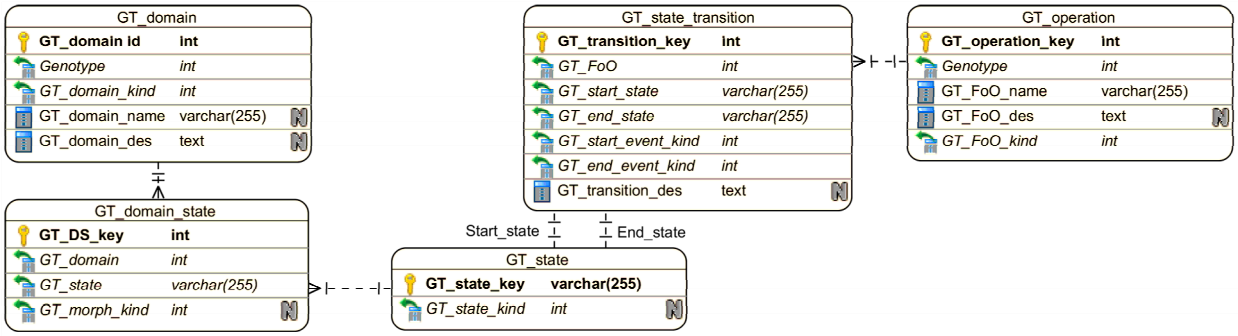

Fig. 9 Construct for arranging relational tables for capturing state transitions by genotypes

between the GT_domain and GT_state tables, which is described by the GT_domain_state table. This allows a domain to be in multiple states, and a state to associate with characteristics of several domains. This way, a state is linked to a domain through the GT_domain_state, which is a kind of virtual entity.

All tables concerning the (architectural) domain are connected to the GT_domain_state table. The advantage is that it makes possible to trace back the changes in the domain during simulation. The column named morph_kind in the GT_domain_state table is connected to the morphology concept table of the concept ontology, which assigns PKs from concepts included in morphological taxonomies (e.g., solid, gas, liquid) to GT_morph_kind. As GT_morph_kind can be different in two states, it is also connected to the GT_domain_state table.

The specification of architecture also uses containment relations, but slightly different from the specification of operations. The two reasons are that multiple subdomains (entities) may belong to a domain in a particular state, and that the state of the domain influences the possibility of a containment relationship. The dependence of the containment relations on the state (and consequently on the operation) is considered by linking GT_entity and GT_possible_super_domain tables to the GT_domain_state table. This enables monitoring the addition and removal of domain entities and super-domains due to operations on the domain. The relations associated with the GT_domain_state table are shown in Fig. 10.

Architectural connectivity is interpreted on both the entity (connection between entities of a domain) and domain (possible connection with other domains) levels. The possible connections are declared as input assumptions and output guarantees. The construct shown in Fig. 11 presents four database tables, from which GT_entity and GT_domain_state tables have been discussed above. The GT_entity_connection table accommodates several FKs that establish a self-many-to-many relation of the GT_entity table, which captures connections among domain entities, as well as a one-to-many relation between the GT_domain_state and GT_entity_connection tables assuming a relevant state. Since connections among domain entities change in various states, the GT_DS key specifies in which state a captured entity connection is valid.

There are two FKs (one for each) for connected entities, and two for connection enablers (for explanation, see Pourtalebi and Horváth (2016a)). In addition, an FK is there for specifying the kind of connection, referring to the connection concept table of the concept ontology. The column storing mereotopological notations captures all chunks of information needed to describe architectural connections (Pourtalebi and Horváth, 2016a). There is a one-to-many relation between GT_domain_state and GT_contract tables, because every domain may have several possible connections to other domains. The two columns included in the GT_contact table specify whether the contract is an input assumption or an output guarantee.

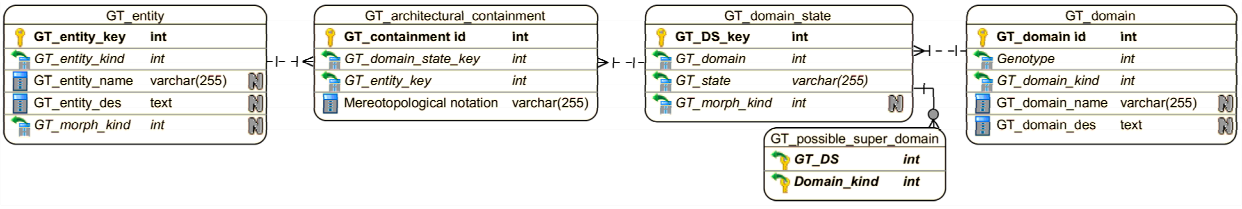

Fig. 10 Construct for arranging relational tables for capturing domain states by genotypes

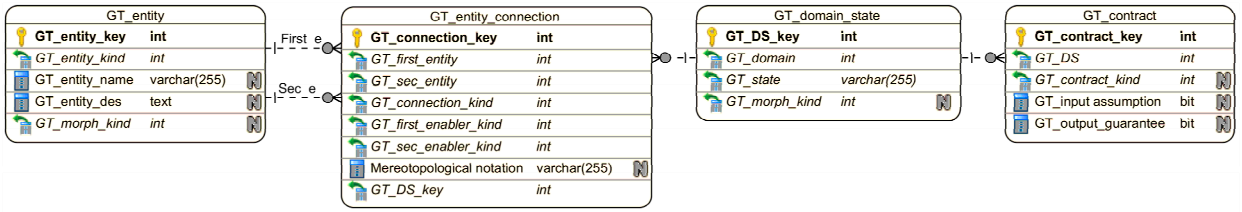

Fig. 11 Construct for arranging relational tables for capturing architectural connections by genotypes

The GT_contract_kind key refers to the contract concept table of the concept ontology.

Operations of a domain are described in terms of FoOs, while operations of domain entities are described by the UoOs included in the FoOs. This later entails that UoOs refer to the concerned domain entities. With regard to the corresponding construct, it establishes a many-to-many relation between the GT_architectural_containment and GT_UoO tables (Fig. 12). Instead of the GT_entity table, the GT_architectural_containment table is used in the construct, because it includes the GT_domain_state key, which is needed to specify the validity of the relation in different states of operation.

5 Information schema constructs for creating phenotypes

5.1 Chunks of information needed for phenotypes of SMFs

The ISCs of a phenotype use 21 ontological concept tables, namely (1) domain, (2) connection, (3) connection enabler, (4) morphology, (5) morphology element, (6) morphology parameter, (7) mate concept, (8) mate parameter, (9) domain attribute, (10) attribute parameter concept, (11) contract, (12) contract parameter, (13) protocol, (14) unit, (15) operation, (16) stream, (17) event, (18) operation attribute, (19) operation parameter, (20) state, and (21) operation layer.

These concepts create the semantical basis of defining the chunks of information and their relationships in the relational tables of the phenotype warehouse database. These read-only relational tables assign PKs to each 'kind' defined in the respective ontologies. These PKs are being imported to the relational tables of the phenotype database in order to specify kinds. The specific chunks of information extracted from AKF are as follows. The chunks of information inherited from genotype are indicated by the '*' sign.

Metadata:

<kind_of_possible_super_domain>*;
<name_of_possible_super_domain>;
<description_of_super_domain>;
<kind_of_domain>*;
<description_of_domain>*;
<associated_operation>*;
<morphological_kind>*;
<representative_state>*;
<architectural_attribute_of_domain>;
<architectural_attributive_parameter_of_domain>;
<value_of_attributive_parameter_of_domain>;
<morphological_element_of_domain>;
<parameter_of_morphological_element>;
<value_of_parameter_of_morphological_element>

Domain entities:

<name_of_entity>*;
<entity_description>*;
<kind_of_entity>*;
<architectural_attribute_of_entity>;
<parameter_of_architectural_attribute_of_entity>;
<value_of_parameter_of_attribute_of_domain>

Entity morphologies:

<kind_of_entity_morphology>*;
<morphological_element_of_entity>;
<parameter_of_morphological_element_of_entity>;
<value_of_morphological_element>

Containment relations:

<list_of_containment_relations>*;
<containment_in_state>

Connectivity relations:

<kind_of_connection>*;
<connected_entities>*;
<kind_of_connection_enabler>*;
<mating_element>;
<mating_parameter>;
<mating_value>;
<connection_in_state>

Contracts:

<kind_of_contract>*;
<parameter_of_contract>

Constraints:

<constraint_of_attributive_parameter>;
<constraint_of_morphological_parameter>;
<constraint_of_contract_parameter>

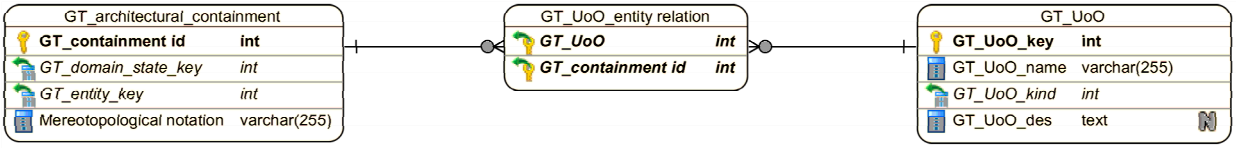

Fig. 12 Construct for arranging relational tables for capturing relations between domain entities and UoOs by genotypes

The specific chunks of information extracted from OKF are as follows:

Metadata:
<kind_of_possible_super_operation>*;
<kind_of_operation>*;
<description_of_PT_operation>*;
<PT_domain>*;
<operation_attribute_of_FoO>;
<parameter_of_operation_attribute_of_FoO>

States of domain:
<kind_of_start_state>*;
<name_of_start_state>;
<description_of_start_state>;
<kind_of_end_state>*;
<name_of_end_state>;
<description_of_end_state>;
<start_event>*;
<end_event>*

Streams:
<kind_of_external_stream>*;
<kind_of_internal_stream>*;
<name_of_stream>;
<description_of_stream>;
<start_time_stamp_of_stream>;
<end_time_stamp_of_stream>;
<halt_time_stamp_of_stream>;
<resume_time_stamp_of_stream>;
<parameter_of_stream>

Events:
<kind_of_event>*;
<name_of_event>;
<description_of_event>;
<stream_to_event>;
<state_of_event>;
<time_stamp_of_event>

Transformative procedure:
<streams_to_UoOs>*;
<direction_of_stream>*

Methods of FoO:
<method>;
<element_of_method>;
<parameter_of_method>

Time stamps:
<specification_of_time_stamp>;
<start_time_stamp>;
<end_time_stamp>;
<halt_time_stamp>;
<resume_time_stamp>;
<duration>

Scheduling conditions:
<conditions>;
<event_to_condition>;
<sequence>;
<event_to_sequence>

Operational constraints:
<maximum_value_of_parameter>;
<minimum_value_of_parameter>

UoOs:
<name_of_PT_UoO>*;
<description_of_PT_UoO>*;
<kind_of_UoO>*;
<operation_attribute_of_UoO>;
<parameter_of_operation_of_UoO>

UoOs' domain entities:
<list_of_entities_to_UoO>*

Methods of UoOs:
<method>;
<elements_of_method>;
<parameter_of_method>

Operational layer:
<list_of_layer_ID>;
<method_to_layer>

5.2 Constructs for defining the warehouse database schemata of phenotypes

The warehouse database of phenotypes is organized as entity-relation tables incorporating chunks of information. Recall that the tables are related to each other through standard connectors. A relation is valid if the PK of one table is represented as an FK in another table. A large number of relations are needed because of the large number of chunks of information relevant for phenotypes. Actually, the proposed database schemata of phenotypes contain 53 tables and 123 exchanged FKs. This complexity explains the many information schema constructs defined. Since phenotypes inherit various chunks of information from their parent genotype, some ISCs are reusable.

Like a genotype, a phenotype also captures information about a domain and the operations associated with it. Therefore, a construct has been designed for this purpose. As shown in Fig. 13, it includes the PT_SMF table that has a one-to-one relation with the PT_domain table, and a one-to-many relation with the PT_operation table. Directly or indirectly, all operational and architectural warehouse database tables are connected to the PT_operation and PT_domain tables, respectively.

The effects of operations on the architectural domains are modeled as state transitions. Therefore, the operation database tables are connected to the PT_state_transition table, which includes chucks of information about start_state, end_state, start_event, end event, and other descriptors of a transition (Fig. 13). States are defined by parameters such as PT_state_name, PT_state_kind, and PT_des(cription). Since one operation may cause multiple transitions in architectural states, there is a one-to-many relation between the PT_operation and PT_state_transition tables. The PT_state table is related to the PT_domain_state table through a one-to-many connector. The PT_domain_state table creates relations between an architectural domain (PT_domain) and its states (PT_state). The PT_state_transition table hosts two event keys and two state keys.

Operations can be looked at as transformations of input streams into output ones. The necessary entities and relations are included in the I/O transformation construct (Fig. 14). A one-to-one relation is defined between the PT_I/O_transformation and PT_operation tables. Using additional parameters and attributes, streams are defined in more detail in phenotypes than in their parent genotype. There are two many-to-many relations between the PT_stream and PT_I/O_transformation tables for both input and output streams, which results in two association tables, namely PT_input_stream and PT_output_stream.

Phenotypes concretize the flows of operations (FoOs), possible super-operations, and sub-operations (UoO) (Fig. 15). In this construct, the name of super-operation is the only extra information added to the PT_possible_super_operation table. The PT_UoO table captures pieces of information related to sub-operations (UoOs). Since one FoO may contain many UoOs, and one UoO may be involved in several FoOs, there is a many-to-many relation between the PT_operation and PT_UoO tables. This relation is captured in the PT_operation_containment table. The PT_UoO_connectivity table captures information about UoO_as_sender, UoO_as_receiver, and the streams between them. The connectivity and containment tables of genotypes and phenotypes are more or less similar. The only difference is that the event column is placed into the stream table of

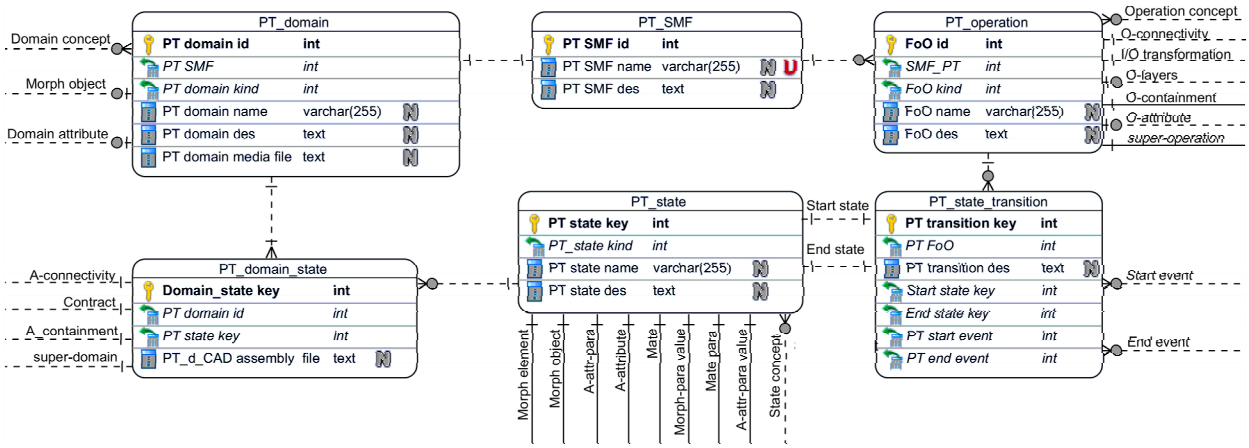

Fig. 13 Construct for arranging relational tables for capturing state transitions by phenotypes

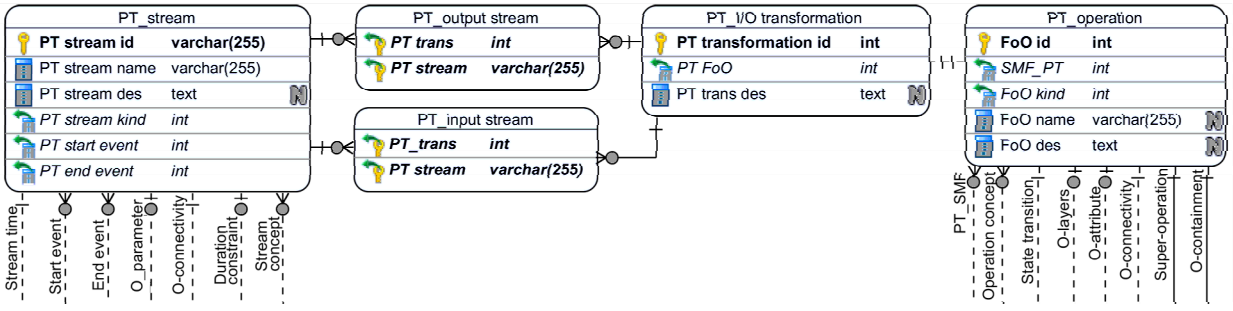

Fig. 14 Construct for arranging relational tables for capturing I/O transformations by phenotypes

phenotype, instead of the connectivity table.

SMFs define physical operations (transformations) in an analytic (numerical) manner and computational operations (transformations) in an algorithmic (logical) manner. Computing of these operations needs different methods. A set of architectural parameter (e.g., morphology, mate, architecture attribute, and contract) tables and operational parameter (e.g., operation attribute, stream, duration, and time) tables are needed to describe the computing methods of an SMF. The chunks of information captured by the tables may be related to each other in various ways. The construct designed for mapping them into the warehouse database includes (1) one table to store the methods, (2) multiple tables for storing different kinds of parameters, and (3) one table that creates relations between the associated parameters and the elements of the methods (Fig. 16). The construct establishes relations between the PT_operation_layers, PT_method, and PT_parameter_assigner tables. The PT_method table stores both numerical and logical methods. The PT_method_element table assigns parameters to the elements of methods. The PT_parameter_assigner table establishes a many-to-many relation between the PT_method table and the parameter table. One parameter may be involved in multiple methods, and one method may establish connection among many parameters (both architectural and operational). The content of the parameter table will be explained later.

Since certain operations may occur simultaneously, the opportunity of parallel computing is to be considered. A typical case where it may be needed is multi-physics analysis and simulation of components of CPSs. To support this from a data management point of view, the concept of layering is applied. Layers lend themselves to simultaneous computing of multiple operations that are logically and numerically interrelated. In the construct designed for this purpose, three kinds of layers are defined as subordinates of the main operation layer. These are specified in the PT_operation_layers table. This assigns computing methods from the PT_method table of the warehouse database to the layer of operation on which FoOs and UoOs are specified.

Streams connect UoOs, and attributive operation parameters describe the actual physical or computational transformations. In SMFs-based modeling, the attributive parameters of UoOs are not specified since they are used as black boxes. However, if an operation parameter is to be defined for a method, it can be done through a stream, or an attribute that may share

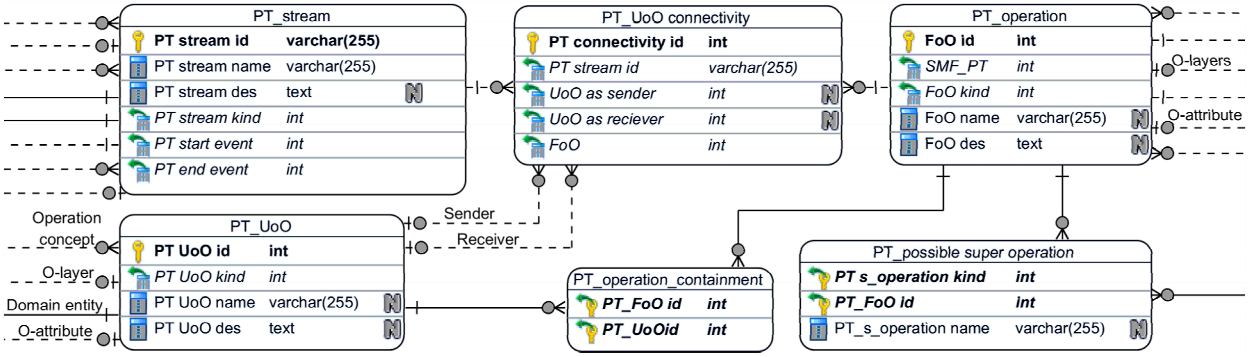

Fig. 15 Construct for arranging relational tables for capturing operation containment and connectivity connections by phenotypes

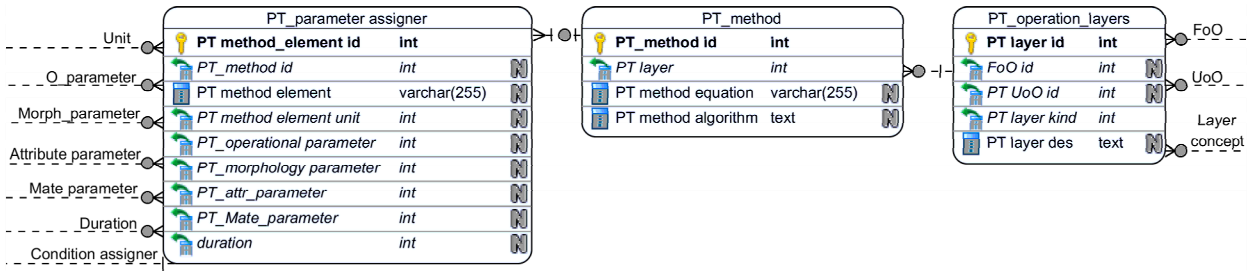

Fig. 16 Construct for arranging relational tables for capturing methods of operations by phenotypes

parameters. There is no direct connection between operation attributes and streams. The related construct has been designed based on these assumptions. As shown in Fig. 17, FKs from the PT_stream and PT_operational_attribute tables are accommodated in the PT_operational_parameter table. The connection with the PT_constraint table makes it possible to define constraints for each parameter. The PT_operational_attribute table receives FKs from the PT_operation and PT_UoO tables. Based on this, operational attributes can be defined for both FoO and UoOs.

As shown in Fig. 18, the PT_event table is connected to the PT_state_transition and PT_stream tables through two connectors: one for the start event and the other for the end event. These connectors establish a one-to-many relation, since each event may be related to multiple streams (e.g., the start event of one stream can be the end event of another stream). Events are typically subsequent with regard to operations. Sequencing relations can be 'before', 'after', 'coincident with', 'asynchronous with', etc. Pairs of related events can be used to define sequencing constraints for all events. The PT_sequence_constraint table makes it possible to define sequential relations by the two FKs of related events. The construct creates an opportunity to handle conditions of operation through events.

Since events are also connected to streams, all operations can be defined conditionally. Conditions are logical statements that are applied to make events dependent on the consequences of other events and on parameter values. These logical or mathematical statements are described in the PT_condition table in Fig. 18. The conditions are assigned to events and/or parameters in the PT_condition_assigner table. The PT_condition table has FKs in the PT_event and PT_parameter_assigner tables.

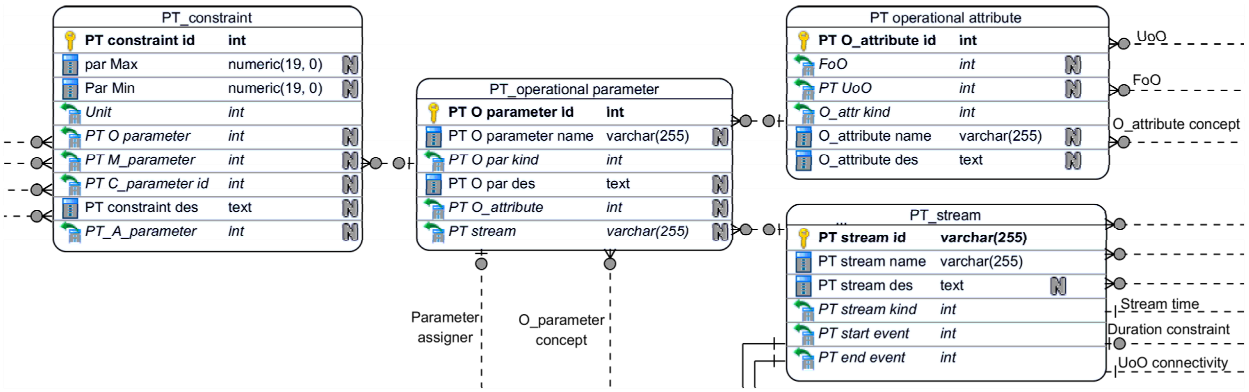

Fig. 17 Construct for arranging relational tables for capturing operation parameters by phenotypes

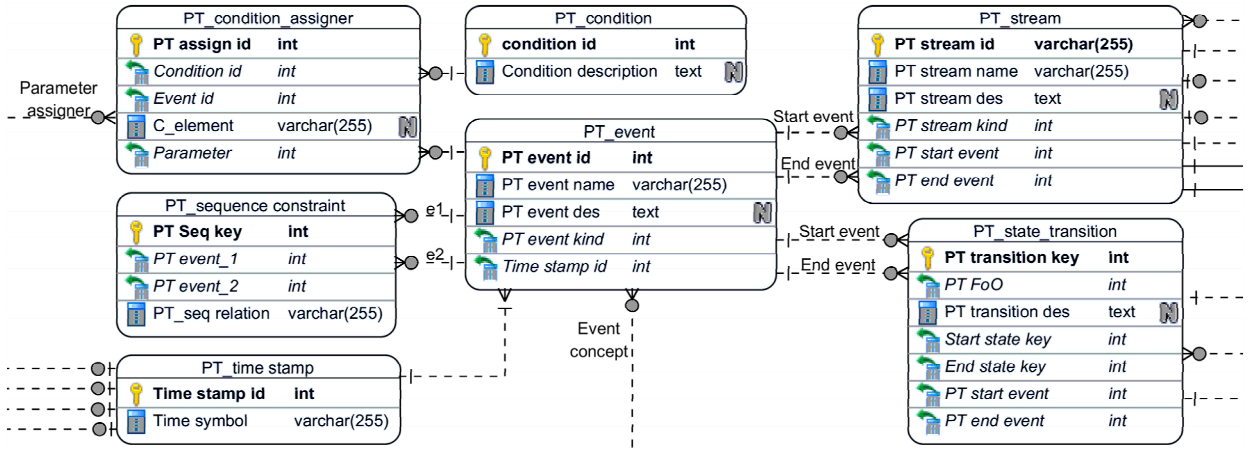

Fig. 18 Construct for arranging relational tables for capturing events, conditions, and sequence constraints by phenotypes

SMFs are supposed to be able to explicitly handle time aspects of operation, including existence. However, SMF instances, rather than phenotypes, are the entities used in modeling in specific application context. Nevertheless, temporal specifications in phenotypes provide part of the chunks of information that are needed to compute time-dependent operations of executable instances, assigning points in time, durations, and time-dependent attributes to streams, events, and entities of SMFs, called 'temporal specification' (TS) here. It is assumed that an event occurs in zero time, and that it is sufficient to assign only one time stamp. However, any change of a stream needs a particular duration, and for this reason a specification of at least two (begin and end) time stamps is necessary. Since streams may be stopped and recommenced, more time stamps may be needed as halts and resumes. The time stamps of the disruption points of streams are taken up by events.

In the construct shown in Fig. 19, three tables are used for temporal specification. The PT stream time table captures the start, halt, resume, and end time stamps of streams, which are assigned to the events by the PT_time stamp table. This makes it possible to assign one particular time stamp to multiple events. Together with the condition and sequence constraints assigned to events, this enables time- and condition-based simulation of operations (transformations of streams). The corresponding measurement units and the duration of streams are specified in the PT_duration constraint table. If there is no constraint on the duration of a stream, then its key will not be included in the PT_duration constraint table.

In line with its supposed operations, the architectural specification of SMFs needs the consideration of three granulation levels, namely (1) domain, (2) possible super-domain, and (3) sub-domain (domain entity) levels. Each domain is an aggregation of multiple domain entities, and can be referred to in several physical contexts (as super-domain). During the operation of an SMF, entities may be changed, added to, or removed from the aggregate represented by a domain. The physical context of a domain may also change (due to a change in super-domain). These are considered as state changes of the domain. In a real-life situation, a domain can be embedded in an aggregative hierarchy of multiple super-domains. These necessitate an explicit handling of the containment relations with regard to the PT_domain_state table.

The construct designed for this is shown in Fig. 20. It also captures the architectural connections with regard to the domain states, because they may change as a result of state changes. As the mediator, the PT_domain_state table is connected to the PT_domain table. Change of the state can be represented by including respective new keys in the PT_domain_state table, which is connected to all architectural tables. The PT_domain_state table has a one-to-many relation with the PT_DS_A_containment table. It means that all domain entities created by domain aggregation and deaggregation can be captured in each state of the domain. The possible connections between domain entities are captured in the PT_entity_connection table, which includes not only the references to domain entities but also the kind of connection and the kind of connection enabler. In the construct, a many-to-many relation is defined between the PT_domain_state and PT_entity_connection tables, through PT_DS_A_connectivity as the association table.

Morphology of domains and domain entities is explicitly captured in SMF-based modeling, no

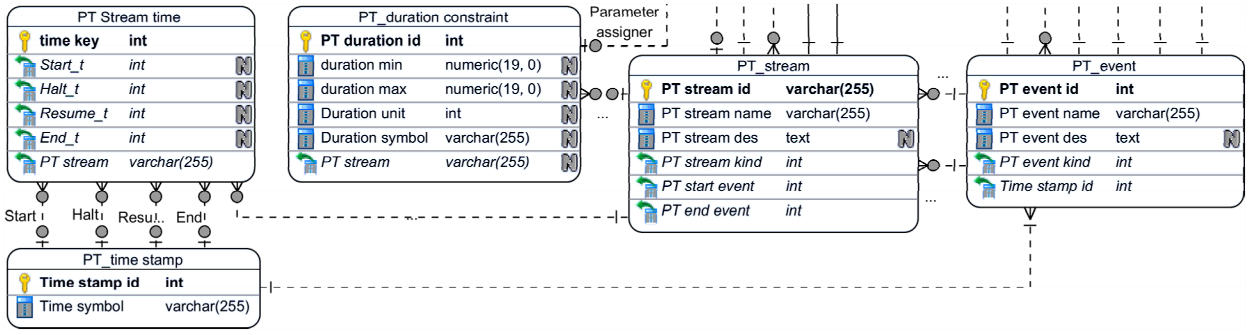

Fig. 19 Construct for arranging relational tables for capturing temporal specifications by phenotypes

matter whether they are hardware, software, or cyberware components. Morphology describes the space that domains occupy in the 3D space. The form of the space is the palpable shape of the domain. Morphological representation and its formats are high-level variables in morphological data management. With a view to the need of morphological representation in the pre-embodiment design phase of CPS development, combined skeleton-based and closure-box-based modeling is applied for solid-type domains.

This morphological representation can also capture the morphological characteristics of software and cyberware domains. The skeleton-based modeling exemplifies ports, based on which not only rigid but also liquid and gas domain entities can be modeled. The construct designed for handling morphologies is able to embed imported CAD files. All these concepts are stated in the morphology concept table of the concept ontology.

The relational tables arranged by this construct are shown in Fig. 21. The kernel is the PT_morphological_object table, which is connected to the PT_morph(ological)_element table. It includes the morphological kinds (e.g., solid, liquid, gas, script, code), in addition to PT_domain and PT_entity. The management of morphological models is based on the PT_morph(ological)_element (e.g., point, curve, surface) and PT_morphological_object tables, which are in a one-to-many relation. The options of element morphologies are stated by the morphology element concept table of the concept ontology. For decomposability, there is a reflective one-to-many relation defined for the PT_morph_element table. This relation is realized through the nullable parent element FK.

Description of the morphology of the domain elements needs several parameters (e.g., height, volume) and values (e.g., linear coordinate, angle). The options are stated in the morphology parameter concept table of the concept ontology. For many parameters default values are specified, e.g., based on entity relations. A morphological description is valid in the state in which it is described. The morphological elements of the PT_morph_element table are mapped to the parameters included in the PT_morphological_parameter table, and subsequently to the values in the PT_para(meter)_value table, to

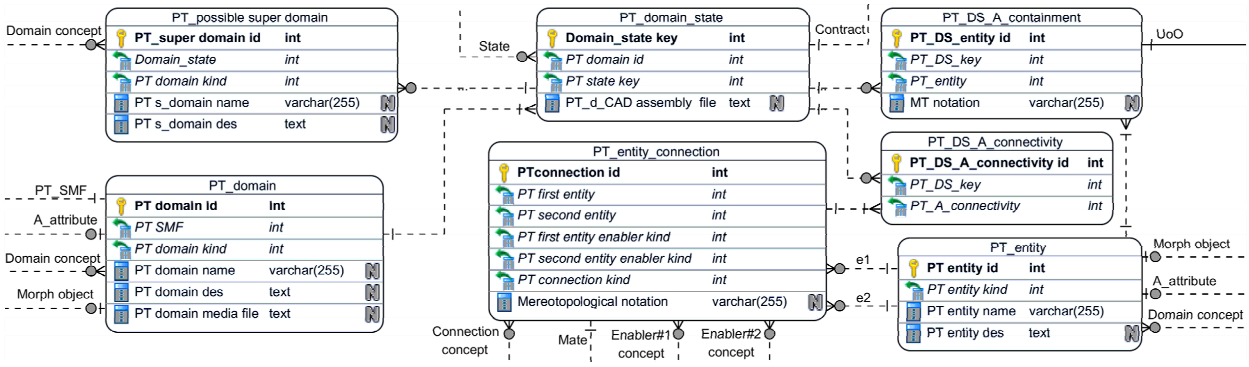

Fig. 20 Construct for arranging relational tables for capturing architectural containment and connectivity by phenotypes

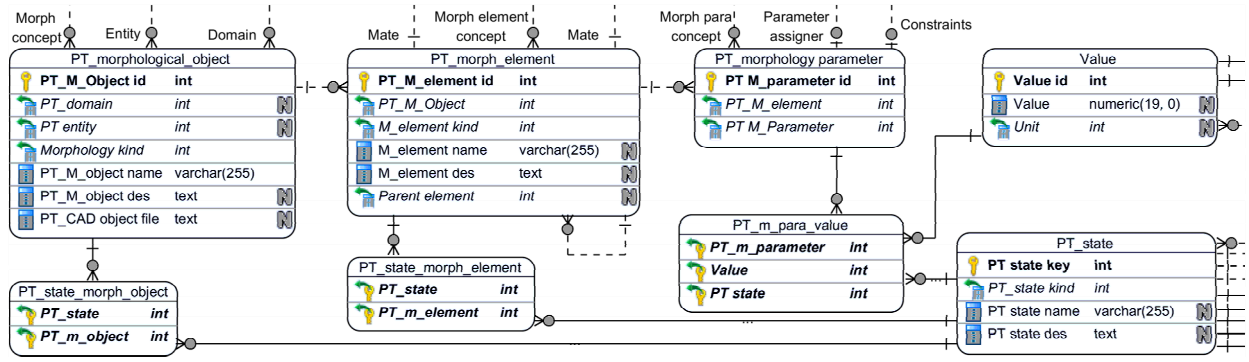

Fig. 21 Construct for arranging relational tables for capturing morphologies by phenotypes

which concrete values can be assigned. The morphology of a domain or an element thereof can change as a result of operation. Therefore, the morphological tables have connection with the operation tables through the domain state table.

Morphological connection between physical (material) domains and/or entities is called 'mate'. The possible morphological connections are stated in the architectural connectivity table of the concept ontology. Skeleton modeling offers ports for these morphological connections. In the construct shown in Fig. 22, the PT_morph(ological)_element table is the basis of specifying mate connections using the content of the PT_mate and PT_mate_parameter tables. The PT_entity_connection table is referred to by the PT_entity_connection key from the PT_mate table. Since mate is dependent on the domain states, there is a many-to-many connection between the PT_mate_state table and the PT_state table of the constructs. Specifying mate connection assumes the existence of an architectural connection between two domains or entities, as well as the selection of two morphological elements.

The mate_kind key of the PT_mate table specifies the mate situation (e.g., tangent mate, perpendicular mate, coincident mate, or concentric mate) between them. Some mate situations, such as distance mate or angle mate, are specified by respective parameters, which are given in the PT_mate_parameter table. The kinds of mate parameters are stated in the mate parameter concept table of the concept ontology. The values of mate parameters are defined in the PT_mate para(meter)_value table, which is connected to the value and PT_mate tables, and allows capturing mate parameter values as functions of states. There might be a situation in which the mate parameter values should be computed by specified methods. To support this issue, the PT_mate_parameter table is connected to the parameter assigner table to provide access to the method table.

The construct shown in Fig. 23 is dedicated to capturing architectural attributes (AAs). These attributes of a domain are varied (e.g., weight, material, cardinality, and flexibility). All possible attributes are handled by the PT_domain_attribute table, which uses keys generated based on the domain attribute concept table of the concept ontology. The kind of attribute parameter is complemented with parameters,

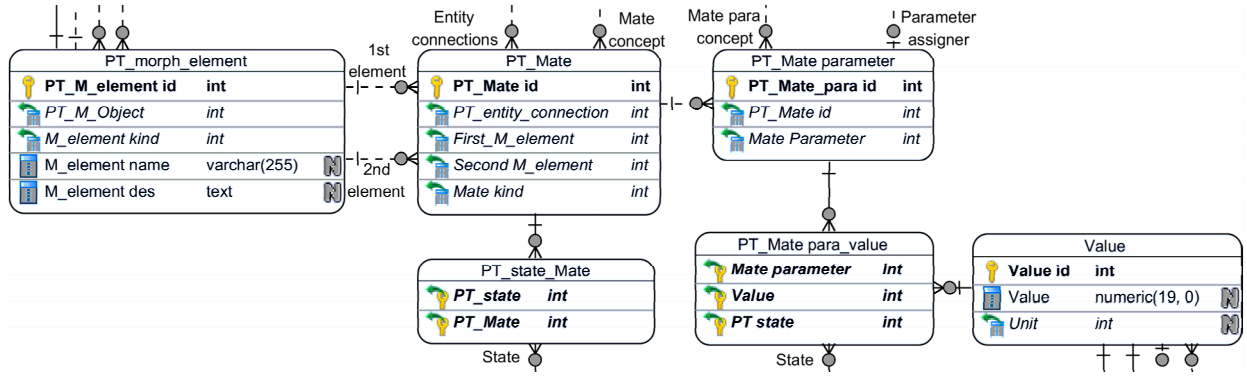

Fig. 22 Construct for arranging relational tables for capturing mates by phenotypes

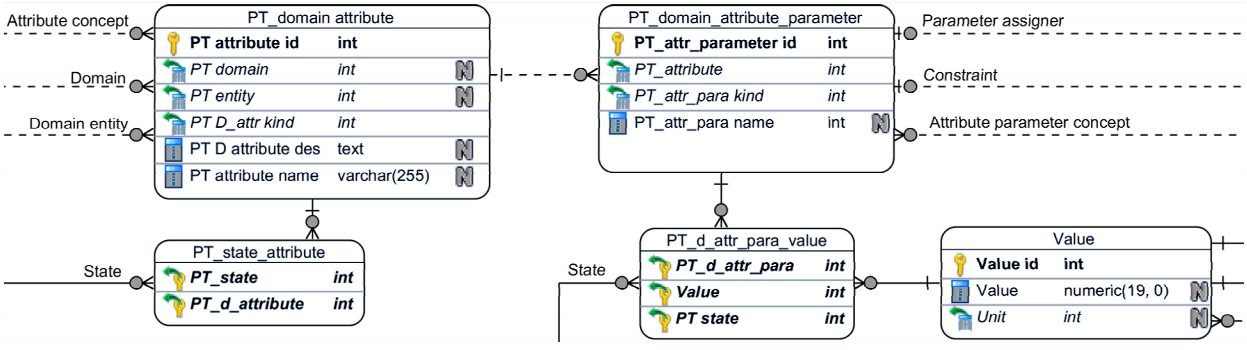

Fig. 23 Construct for arranging relational tables for capturing architectural attributes by phenotypes

units, and values. Since attribute parameters are used for computing values of operational parameters, the PT_domain_attribute table is connected to the PT_parameter_assigner table. This is used when parameters are assigned to methods of operation.

The kind of attribute and the attributive parameters may change in different states (over time). This explains the many-to-many relations between the abovementioned tables and the PT_state_attribute association table, which enables the connection between attributive parameters and states. In addition, the PT_domain_attribute parameter table is connected to the PT_constraint table in order to specify the minimum and maximum values of the attributive parameters.

The notion of contracts was adopted to inform about the possibility of establishing connectivity between architectural domains. Contracts are associated with domains and represent their architectural interfaces. The construct that has been designed to manage contracts is shown in Fig. 24. There can be parametric contracts and protocol contracts defined. Their concept is stated in the concept ontology.

Contract specification embraces the kind of contract, which can be either an input assumption or an output guarantee. These are included in the PT_contract table. No matter which hardware, software, or cyberware contracts are concerned, their parameters are captured by the PT_contract table. The possible range of parameters is specified in the PT_constraint table. Protocol contracts are defined in the contract protocol table. They can be operationalized based on the entries of this table. The kinds of applicable protocols are included in the protocol concept table. The contracts are also associated with the domain states through the PT_DS_contract association table.

Typical for phenotypes is that they specify a possible range of parameter values rather than nominal values. It is an instantiation issue to derive the best matching value for an SMF instance from the range of parameter values. Nevertheless, there are some constant parameter values (CPVs) included in the phenotype as input values. As a rule, CPVs are defined in the respective association tables (such as PT_morph_para_value, PT_mate_para_value, and PT_attr_para_value) of many constructs. The range of values of variable parameters is included in the PT_constraint table of the defined construct (Fig. 25). In different CPS modeling situations, it may occur that the range of variation of a parameter value should be specified depending on the range of variation of

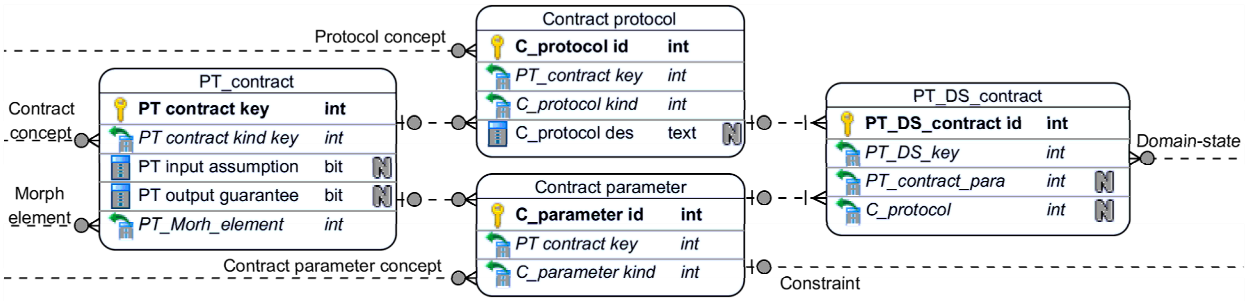

Fig. 24 Construct for arranging relational tables for capturing contracts by phenotypes

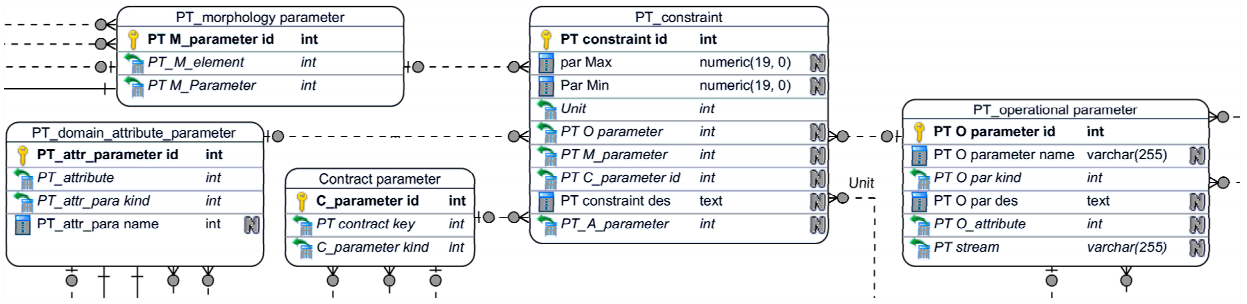

Fig. 25 Construct for arranging relational tables for capturing parameter constraints by phenotypes

another parameter value. This needs a specific type of constraint, which sets the range of one parameter according to the range of another parameter that it depends on. These have been called 'relative constraints' and are managed in the 'methods of operation' construct shown in Fig. 16. For all variable and constant parameters assigned by the PT_parameter_assigner table, the relative constraints upon ranges or values are determined by the mathematical or logical equations included in the columns of the PT_method table, respectively.

6 Discussion, conclusions, and future work

6.1 Discussion of the approach and findings

The presented SMF-based modeling methodology, the developed SMF-based toolbox, and the ISC-enabled warehouse database definition are the novel elements of our research work. Approaching the issues of pre-embodiment design of CPSs on this (or similar) basis is indispensable, since the traditional methodologies and tools have been found suboptimal for this purpose, though they can support many parts of the development process (Muth *et al.*, 2001). Their main deficiency is in the restricted capabilities of transdisciplinary integration, explicit handling of semantics, and relieving designers from the tasks of low-level logical and analytical specifications (Chaudron *et al.*, 2001; Wang *et al.*, 2016).

The conceptualized SMF-TB integrates two major constituents: (1) the platform, which supports creation and warehousing of SMFs, and (2) the workbench, which allows instantiation of SMFs and composition of (pre-embodiment) system models. In a physical point of view, an SMF resembles a component (e.g., hardware, software, cyberware, and aggregated-ware) in a real world, which could be used as a part in CPSs. From a knowledge structuring point of view, SMFs rely on two intertwined knowledge frames (AKF and OKF), describing the chunks of information needed for the specification of their architecture and operations.

The genotypes and phenotypes of SMFs include specific chunks of information that have been arranged based on ISCs, considering their associative relationships. The associative relationships and shared contents are inherited from genotypes through phenotypes to specific instances. This inheritance mechanism contributes to an effective architectural and operational processing of the SMFs information.

The proposed information schema constructs couple the chunks of operation and architecture information as relational data tables in the warehouse databases. At the highest aggregation level, operations are connected to their architectural domain through the PT_SMF table. A second connection is that state transitions generated by operations are referring to the states of the concerned architectural domain. This relation is established by means of the PT_state and PT_domain_state tables. A third connection is that units of operation are also connected to domain entities through the PT_UoO_PT_entity table. This connection also implies relation among subdomains and sub-operations. A fourth connection is captured by the PT_parameter_assigner and PT_method tables, which assign architecture and operation parameters to numeric and algorithmic methods.

6.2 Conclusions

Not considering the challenges that originate in complexity, transdisciplinary nature, and multifunctionality, one of the major technological challenges of the implementation of the SMF-based toolbox is functional design- and language-independent computational realization of the warehouses. Another challenge is that the process of SMF-based system design is based on various evolving implements of SMFs (genotypes, phenotypes, and instances).

Our intention was to achieve a high level of synergy in terms of architectural and operational specifications of SMF and CPS models. Toward this end, implementation of SMFs started out from the specified AKF and OKF. This intention was also supported by the introduction of the concept of information schema construct and operationalizing it in the context of specification of warehouse databases for handling genotypes and phenotypes of SMFs (Oliver, 1993).

In a broader research perspective, the present paper is a part of feasibility confirmation of the SMF theory. In our CPS-orientated modeling system, SMFs are the building blocks. The proposed unique compositional technique (aggregation instead of abstraction) makes it possible not only to effectively

generate original models but also to flexibly modify existing ones. The latter plays an important role in embedded customization of, e.g., cyber-physical consumer durables (Pourtalebi *et al*., 2013; 2014a; 2014b). As composition of conceptual information elements, ISCs enable executable code-level implementation of elementary functions or selecting software components which realize these functions.

Until the time of this reporting, the constituents of the proposed approach have been validated by (1) critical thinking, (2) pilot programming and tests, (3) case study and example analyses, (4) critical comparisons with other trademarked systems, and (5) logical functionality and interoperability analyses. It was scrutinized that the proposed overall procedure is able to manage the information elements that should be available for the SMF genotypes and phenotypes as input, and that should be available for instantiation of SMFs and composition of CPS models. This process has been reported by Pourtalebi and Horváth (2016b). Nevertheless, a final comprehensive practical validation will be possible only through an all-inclusive code-level programming implementation of the SMF-based modeling and simulation toolbox.

6.3 Future work

Recently, efforts were made to develop the set of ISCs that are needed for instantiation of phenotypes of SMFs and mapping the data of system models to the model database. The results of this work will be reported in the follow-up paper. Afterward, the development will continue with the implementation of the other modules of SMF-TB, namely the GT editor, PT editor, shared browser, instantiator, composer, visualizer, and simulator, as well as the integration of these components into a testable pilot system.

References

Abdul-Ghafour, S., Ghodous, P., Shariat, B., *et al*., 2014. Semantic interoperability of knowledge in feature-based CAD models. *Comput.-Aided Des.*, **56**:45-57. http://dx.doi.org/10.1016/j.cad.2014.06.001

Apel, S., Kästner, C., 2009. An overview of feature-oriented software development. *J. Object Technol.*, **8**(5):49-84. http://dx.doi.org/10.5381/jot.2009.8.5.c5

Au, C.K., Yuen, M.M.F., 2000. A semantic feature language for sculptured object modelling. *Comput.-Aided Des.*, **32**(1):63-74. http://dx.doi.org/10.1016/S0010-4485(99)00085-8

Batory, D., 2005. Feature models, grammars, and propositional formulas. *LNCS*, **3714**:7-20. http://dx.doi.org/10.1007/11554844_3

Bidarra, R., Bronsvoort, W.F., 2000. Semantic feature modelling. *Comput.-Aided Des.*, **32**(3):201-225. http://dx.doi.org/10.1016/S0010-4485(99)00090-1

Case, K., Gao, J., 1993. Feature technology: an overview. *Int. J. Comput. Integr. Manuf.*, **6**(1-2):2-12. http://dx.doi.org/10.1080/09511929308944549

Chaudron, M.R., Eskenazi, E.M., Fioukov, A.V., *et al*., 2001. A framework for formal component-based software architecting. Proc. Specification and Verification of Component-Based Systems Workshop, p.73-80.

Czarnecki, K., Eisenecker, U.W., 2000. Generative Programming: Methods, Tools, and Applications. Addison-Wesley, Boston, MA.

Czarnecki, K., Helsen, S., Eisenecker, U., 2005. Formalizing cardinality-based feature models and their specialization. *Softw. Process Improv. Pract.*, **10**(1):7-29. http://dx.doi.org/10.1002/spip.213

Grzybek, H., Xu, S., Gulliver, S., *et al*., 2014. Considering the feasibility of semantic model design in the built-environment. *Buildings*, **4**(4):849-879. http://dx.doi.org/10.3390/buildings4040849

Henderson, M.R., Anderson, D.C., 1984. Computer recognition and extraction of form features: a CAD/CAM link. *Comput. Ind.*, **5**(4):329-339. http://dx.doi.org/10.1016/0166-3615(84)90056-3

Horváth, I., Pourtalebi, S., 2015. Fundamentals of a Mereo-Operandi theory to support transdisciplinary modeling and co-design of cyber-physical systems. Proc. ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf., p.1-12. http://dx.doi.org/10.1115/DETC2015-46702

Hu, F., 2013. Cyber-Physical Systems: Integrated Computing and Engineering Design. CRC Press, Boca Raton, p.15-35. http://dx.doi.org/10.1201/b15552

Kacprzynski, G.J., Roemer, M.J., Byington, C.S., *et al*., 2002. Enhancing gear physics of failure models with system level vibration features. Proc. 56th Meeting of the Society for MFPT, p.263-277.

Lee, G., Eastman, C.M., Sacks, R., *et al*., 2006. Grammatical rules for specifying information for automated product data modeling. *Adv. Eng. Inform.*, **20**(2):155-170. http://dx.doi.org/10.1016/j.aei.2005.08.003

Muth, T., Herzberg, D., Larsen, J., 2001. A fresh view on model-based systems engineering: the processing system paradigm. *INCOSE Int. Symp.*, **11**(1):295-302. http://dx.doi.org/10.1002/j.2334-5837.2001.tb02306.x

Oliver, D.W., 1993. Descriptions of systems engineering methodologies and comparison of information representations. *INCOSE Int. Symp.*, **3**(1):97-104.

Pandit, S., Honavar, V., 2010. Ontology-guided extraction of complex nested relationships. Proc. IEEE 22nd Int. Conf. on Tools with Artificial Intelligence, p.173-178.

http://dx.doi.org/10.1109/ictai.2010.98

Parry-Barwick, S., Bowyer, A., 1993. Feature Technology. Technical Report, University of Bath, School of Mechanical Engineering, Bath.

Pourtalebi, S., Horváth, I., 2016a. Towards a methodology of system manifestation features-based pre-embodiment design. *J. Eng. Des.*, **27**(4-6):232-268. http://dx.doi.org/10.1080/09544828.2016.1141183

Pourtalebi, S., Horváth, I., 2016b. Procedures for creating system manifestation features: an information processing perspective. Proc. 11th Int. Symp. on Tools and Methods of Competitive Engineering, p.129-142.

Pourtalebi, S., Horváth, I., 2016c. Information schema constructs for instantiation and composition of system manifestation features. *Front. Inform. Technol. Electron. Eng.*, in press. http://dx.doi.org/10.1631/FITEE.1601235

Pourtalebi, S., Horváth, I., Opiyo, E., 2013. Multi-aspect study of mass customization in the context of cyber-physical consumer durables. Proc. ASME Int. Design Engineering Technical Conf. & Computers and Information in Engineering Conf., p.V004T05A006. http://dx.doi.org/10.1115/detc2013-12311

Pourtalebi, S., Horváth, I., Opiyo, E.Z., 2014a. New features imply new principles? Deriving design principles for mass customization of cyber-physical consumer durables. Proc. 10th Int. Tools and Methods of Competitive Engineering Symp., p.95-108.

Pourtalebi, S., Horváth, I., Opiyo, E.Z., 2014b. First steps towards a Mereo-Operandi theory for a system feature-based architecting of cyber-physical systems. 4th Int. Workshop on Advanced Design Concepts and Practice, p.2001-2006.

Pratt, M.J., 1991. Aspects of form feature modelling. *In*: Hagen, H., Roller, D. (Eds.), Geometric Modeling. Springer, Berlin Heidelberg, p.227-250. http://dx.doi.org/10.1007/978-3-642-76404-2_10

Pulvermueller, E., Speck, A., Coplien, J.O., *et al*., 2002. Feature interaction in composed systems. *LNCS*, **2323**:86-97. http://dx.doi.org/10.1007/3-540-47853-1_7

Romero, T.A., López, G.D., Torres, F.R., 2015. Dynamic SQL codebuilder. *Int. J. Latest Res. Sci. Technol.*, **4**(6):1-6.

Salomons, O.W., van Houten, F.J., Kals, H.J., 1993. Review of research in feature-based design. *J. Manuf. Syst.*, **12**(2): 113-132. http://dx.doi.org/10.1016/0278-6125(93)90012-I

Schirner, G., Gerstlauer, A., Dömer, R., 2010. System-level development of embedded software. Proc. Asia and South Pacific Design Automation Conf., p.903-909. http://dx.doi.org/10.1109/aspdac.2010.5419674

Shah, J.J., Rogers, M.T., 1988. Functional requirements and conceptual design of the feature-based modelling system. *Comput.-Aided Eng. J.*, **5**(1):9-15. http://dx.doi.org/10.1049/cae.1988.0004

Shenton, A.K., Hayter, S., 2006. Terminology deconstructed: phenomenographic approaches to investigating the term "information". *Libr. Inform. Sci. Res.*, **28**(4):563-578. http://dx.doi.org/10.1016/j.lisr.2006.10.003

Sun, H., Zhang, P., 2008. Adaptive system use: an investigation at the system feature level. Proc. 29th Int. Conf. on Information Systems, p.170.

Tao, R., Yang, L., Peng, L., *et al*., 2009. A case study: using architectural features to improve sophisticated denial-of-service attack detections. Proc. IEEE Symp. on Computational Intelligence in Cyber Security, p.13-18. http://dx.doi.org/10.1109/CICYBS.2009.4925084

VDI, 2003. Information Technology in Product Development: Feature Technology. VDI 2218. VDI-Richtlinien, Beuth Verlag GmbH, Berlin.

Wang, H., Zhang, Y., Cao, J., *et al*., 2003. Feature-based collaborative design. *J. Mater. Process. Technol.*, **139**(1-3): 613-618. http://dx.doi.org/10.1016/s0924-0136(03)00502-8

Wang, Y.N., Lin, Z.Y., Liang, X., *et al*., 2016. On modeling of electrical cyber-physical systems considering cyber security. *Front. Inform. Technol. Electron. Eng.*, **17**(5):465-478. http://dx.doi.org/10.1631/FITEE.1500446

Wu, Y., He, F., Zhang, D., *et al*., 2015. Service-oriented feature-based data exchange for cloud-based design and manufacturing. *IEEE Trans. Serv. Comput.*, in press. http://dx.doi.org/10.1109/TSC.2015.2501981

Zha, X.F., Sriram, R.D., 2006. Feature technology and ontology for embedded system design and development. Proc. ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf., p.701-714. http://dx.doi.org/10.1115/detc2006-99543

Zha, X.F., Fenves, S.J., Sriram, R.D., 2005. A feature-based approach to embedded system hardware and software co-design. Proc. ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf., p.609-620. http://dx.doi.org/10.1115/detc2005-85582

Zhang, D.J., He, F.Z., Han, S.H., *et al*., 2016. Quantitative optimization of interoperability during feature-based data exchange. *Integr. Comput.-Aided Eng.*, **23**(1):31-50. http://dx.doi.org/10.3233/ica-150499

Mr. Shahab POURTALEBI, first author of this invited paper, received his bachelor's degree in Industrial Design Engineering from Sahand University of Technology, Iran, in 2003. He received master's degree in Industrial Design Engineering from Art University of Tehran, Iran, in 2006. From 2006 till 2012 he was a lecturer in the Industrial Design Department of Art University of Tabriz, where since 2008 till 2011 he also

worked as head of the department. Currently, he is a PhD researcher at the Design Engineering Department of Faculty of Industrial Design Engineering in TU Delft. In his PhD research, he has been involved in the development of a system-level feature-based modeling toolbox to support the design of cyber-physical systems.

Imre HORVÁTH obtained MSc diplomas in mechanical engineering and engineering education from the Technical University of Budapest in 1978 and 1980, respectively. Then he worked for the GANZ Hungarian Shipyards and Crane Factory until 1983. He had various faculty positions at the Technical University of Budapest until 1996. He earned dr.univ. and Ph.D. titles from the Technical University of Budapest in 1987, and C.D.Sc. title from the Hungarian Academy of Sciences in 1993. He was nominated to a chair professor position at the Faculty of Industrial Design Engineering, Delft University of Technology in 1996. He has more than 350 publications. He acted as co-Editor-in-Chief of *Computer-Aided Design* in the period of 2004–2014, and is emeritus editor now. He is associate editor of *Journal of Engineering Design*. He compiled 29 journal special issues as guest editor and edited 15 conference proceedings. His current research interest is in designing smart cyber-physical systems, and tools and methods for modeling of reasoning mechanisms of cyber-physical systems. He is initiator of the series of International Tools and Methods of Competitive Engineering Symposia. He served in various positions on the Executive Board of the CIE Division of ASME and is a fellow of ASME.