Representing Function-Technology Platform based on the Unified Modeling Language

Alex Alblas, Linda Zhang, Hans Wortmann

Representing Function-Technology Platform based on the Unified Modeling Language

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2010-IJPR-0338.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	03-Nov-2010
Complete List of Authors:	Alblas, Alex; Eindhoven University of Technology Zhang, Linda; University of Groningen, Operations Wortmann, Hans; University of Groningen
Keywords:	MODELLING, PRODUCT DESIGN, RE-USE
Keywords (user):	

Representing Function-Technology Platform based on the Unified Modeling Language

Alex Alblas[1, 2], Linda L. Zhang[1*] and Hans Wortmann[1]
[1] Faculty of Economics and Business, University of Groningen, the Netherlands
[2] Department of Industrial Design, Eindhoven University of Technology, the Netherlands

Abstract:

Product platforms are used in many industries to allow a variety of products to be offered to the market while levering commonality in components. The reported methodologies to design product platforms assume mature and stable design and manufacturing technologies. Consequently, product platforms are not applicable in the semiconductor equipment manufacturing industries, where the technologies keep evolving and cannot be frozen in the product development process. In response to the application limitations of traditional platforms, a concept of function-technology (FT) platform is put forward to assist the semiconductor equipment manufacturers to efficiently design product families by reusing, in a structured way, functions and technologies. To shed light on the diverse constituent elements and the complex relationships inherent in an FT platform, this study focuses on its structural representation. A formalism of FT platform representation is developed based on the Unified Modeling Language (UML). It consists of a generic functional structure, a generic technology structure and the mapping relationships in between. An application case in a well-known semiconductor equipment manufacturer is also reported to present the structure of an FT platform and its representation based on the UML.

[*] *Corresponding author E-mail:* l.zhang@rug.nl

Keywords: Function-technology platform, Unified modeling language, Variant derivation.

1. Introduction

Product platforms have become a principal fundament and a prerequisite for profitable product development in many industries nowadays (Holmquist, 2004; Sawhney, 1998). The advantages of developing a number of related products (the so called a product family) based on a common platform include speeding up product development process, tailoring products to the need of different market segments and customers, reducing development costs, improving design quality, etc. (Skold and Karlsson, 2007). Competitive firms are increasingly requiring to reuse their assets across products because of the growing customization demanded in the market. This reuse is, however, often limited to physical components without reuse of other kinds of product information. Especially science based firms, that develop complex products and systems, require the reuse of product information that captures the scientific knowledge of products.

The definition of platforms reported in previous studies is effected by the type of product information that constitutes the platform. In accordance with the authors' expertise and focus, product platforms have been diversely conceptualized in the literature, ranging from being general and abstract (McGrath, 1995; Robertson and Ulrich, 1998) to being industry and product specific (Ericsson and Erixon, 1999; Sanderson and Uzumeri, 1995). Among the many definitions and descriptions, two common understandings of product platforms can be generalized. In one perspective, a product platform is viewed as a physical one, consisting of a set of well-structured tangible elements, such as parts and assemblies, which are common to all variants of the product family to be developed (Ericsson and Erixon, 1999; Muffatto and Roveda, 2002; Wilhelm, 1997). In contrast, the other perspective treats a platform as an abstract one formed by a set of intangible elements, such as common structures, knowledge about functions and core technologies; with these common intangible elements, a product family can be developed (Halman et al., 2003; McGrath, 1995; Meyer and Lehnerd, 1997; Robertson and Ulrich 1998; Skold and Karlsson, 2007). In these platform perspectives, however, a formal representation of platforms is lacking. Building on knowledge on problems with physical platforms (Alblas and Wortmann, 2009; 2010), the present research aims to further our understanding of the formal structure of platform. In particular, this research will more precisely examine the functional and technical elements that constitute the platforms together with the link between these elements. This new formalization of platforms is required because an increasing number of industries are highly complex and science-based and thus require to configure product design variants that can be developed further for specific customers.

Since early 90's, successful platform applications have been increasingly reported in the literature. However, most of these applications consider products in engineering-based industries, such as automobiles, airplanes, home appliances, personal computers, bicycles, etc. (e.g. Wilhelm, 1997, Sanderson and Uzumeri, 1997; Feitzinger and Lee, 1997; Whitney, 1993). Moreover, in most of these applications, the platforms are physical platforms, including common components, modules and their interfaces. For example, a platform developed by Volkswagen consists of a floor group, drive system, and running gear, along with the unseen part of the cockpit (Wilhelm, 1997). Based on this platform, Volkswagen has developed several models within different brands, such as Volkswagen, Audi, Seat, and Skoda. In comparison, platform applications involving products in science-based industries have not been found (Alblas and Wortmann, 2009; 2010).

Unlike engineering-based industries, science-based industries are characterized by rapid innovation speed, short time lags between scientific discoveries and their industrial implementations, complex and capital-intensive products (Chuma, 2006). One typical example is the semiconductor equipment manufacturing industries. In these types of firms, they both develop concepts that are on the fences what is scientifically known and available and implement them in commercial products. For example, to increase the accuracy of position measurement of a silicon wafer, ASML develops a new breakthrough mathematical algorithm. This fundamentally differs from firms that develop products based on known solution principles.

In these industries, on one hand, the development of new products highly depends on advanced technologies; on the other hand, these technologies keep evolving during product design and production. Consequently, some technologies cannot be frozen in product development process (Chuma, 2006). This not only impacts the mapping from the technologies to the physical components, but also prohibits the application of physical product platforms, which assume mature and stable technologies (Alblas and Wortmann, 2009; 2010). In addition, the extreme product complexity together with the frequent technology innovations during the life cycle of a product family impede defining stable physical modules that function independently. Consequently, reusing components/modules cannot be easily achieved. Nevertheless, with well-organized mechanisms, efficient reuse of functions and technologies is still possible.

Same as in other industries, in the semiconductor equipment manufacturing industries, product life cycles tend to be shorter; customers increasingly demand higher variety of options; and global competition becomes more intense (Chuma, 2006). To survive, semiconductor equipment manufacturers have to be able to quickly deliver customized products at low costs. In view of the above application limitations of product platforms, Alblas and Wortmann (2010) put forward a concept of function-technology (FT) platform, in attempting to assist structured reuse of intangible design elements (i.e., functions and technologies) in the semiconductor equipment manufacturing industries. An FT platform consists of the common functions associated with a product family and the technologies to deliver these functions. In addition, it also captures the interconnections between functions and technologies. Alblas and Wortmann (2010) have also reported the preliminary results of developing FT platforms, including 1) increased reuse of functions and technologies, 2) reduced development costs and time, and 3) improved capacity allocation.

While the notion of FT platforms is proposed and its contribution to product family development is empirically studied in the authors' early work (Alblas and Wortmann, 2010), the other issues pertaining to an FT platform, such as rigorous formulation, structural representation, design and development, etc. are left untouched. The previous work showed some ambiguity in the definition of platform elements which increased the complexity of managing versions and variants of these elements and the related products in a family. In this study, in attempting to facilitate solution development of these issues, we base on this work and use the ASML case to go a step further to analyze an FT platform from the structural aspect. Thus, our focus is on the structural representation of an FT platform in terms of the constitute elements and their relationships. This structural representation can be used by science-based firms to structure the versions and variants of functionally and technically defined products within a family. With the findings in terms of essential elements and their relationships involved in an FT platform, we believe this study would facilitate product development planning and structuring decisions and, consequently, influence the development of subsequent configuration management tools.

The rest of this paper is organized as follows: In Section 2, the requirements of representation tools are analyzed in accordance with the concept implications of an FT platform; and the unified modeling language (UML) is identified as the appropriate tool. An application case is introduced in Section 3 with focus on product information, such as the products themselves, the associated functions and technologies. Sections 4 and 5 detail the UML-based representation of the functional and technology structures of an FT platform, respectively. Also presented are the corresponding case application results. Section 6 provides the representation of an FT platform based on the UML and the case application result. This paper is concluded in Section 7 by pointing out the limitations and possible avenues for future research.

2. Representation Tool Identification

In developing products, the concept of *domain* is often used to organize the large amount of product information (Erens and Verhulst, 1997; Pahl and Beitz, 2003; Suh, 1990; Ulrich and Eppinger, 1995). Three typical domains are the functional, technology and physical domains (Erens and Verhulst, 1997). While the functionalities that products are expected to realize are modeled in the functional domain, the technology domain captures the technologies selected to provide solutions for the functions specified in the functional domain. The physical domain consists of component items which are to implement the technological solutions designed in the technology domain. Both functions and technologies are intangible design elements, whilst component items in the physical domain are tangible design elements. With the three domains, product development entails mapping processes between any two consecutive domains, as shown in Figure 1. Appropriate technologies are assigned to functions according to the desired values of functional performance; and physical components are specified to implement the technologies taking into account, e.g., manufacturing capabilities, material availability, economic factors. In such mapping processes, feedback is often used to refine elements in each domain.

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 1 here>>>>>>>>>>>>>>>>>>>>>>

An FT platform can be viewed as a conceptual structure and overall logical organization of a product family from the functional and technology viewpoints. Such a conceptual structure and overall logical organization acts as a generic umbrella, under which specific functions and the corresponding technologies can be determined to fulfill customer requirements for specific products in a family. In this regard, an FT platform includes all the functions associated with a product family, the possible technologies that can deliver the functions, and the interconnections in between. With such a conceptual structure, the derivation of appropriate technologies in accordance with specific functions of customized products can be achieved. Thus, from the architecture perspective, an FT platform involves two aspects: 1) a common function-technology (FT) structure within which variations in functions and technologies for diverse products belonging to a family can be differentiated; and 2) the derivation of specific technologies (or technology variants) in relation to the corresponding functions from the common FT structure. We first elaborate such concept implications of an FT platform as follows.

2.1 Concept implications of an FT platform

The common FT structure: The common FT structure refers to the generic FT structure organizing all intangible functional and technology elements in the functional and technology domains, respectively, that may occur in the targeted product family. It is formed by unifying a generic functional structure (GFS) and a corresponding generic technology structure (GTS).

As with the decomposition of a product into assemblies and parts at different levels of the product hierarchy, the function of a product can be decomposed into subfunctions and elementary functions. (The function of a product is the function that customers expect a product to perform. Take a bicycle as an example, its product function is to transport a cyclist from one location to another location.) While subfunctions are formed by child functions, be they subfunctions or elementary functions, the elementary functions cannot be decomposed further. The implication is that the realization of subfunctions depends on the realization of the child functions, whilst the realization of an elementary function is directly achieved by the corresponding technologies. In this regard, the decomposition entails an iterative process, stopping when all subfunctions have been decomposed into elementary functions. Thus, the GFS includes all product functions, subfunctions, elementary functions and their relationships that are necessary to deliver a product family[1]. Also included are the properties describing functions and the possible values that these properties can assume.

Similarly, the GTS captures all the possible technologies to deliver the functions in the GFS. In accordance with the product functions, subfunctions and elementary functions in the hierarchy of the GFS, the GTS organizes the system technologies and child technologies at different levels of abstraction. While a system technology is a composite one including child composite technologies and/or child basic technologies, a basic technology is an atomic unit having no child. Thus, a basic technology is directly implemented by components in the physical domain, a system technology is implemented by the composite implementation of its child technologies. Each technology is described by a number of attributes (e.g., air pressure and temperature as the attributes of a wafer handling technology) and the possible values that there attributes can assume. Moreover, the materials and/or tools that are necessary to implement the technologies are also specified.

Unifying the GFS modeling the functional domain and the GTS modeling the technology domain forms the common FT structure. Therefore, the common FT structure captures not only the data/information in each individual domain but also the interconnections (or mapping relationships) between the two domains. The mapping relationships can be classified into several categories. One basic category models the positive connections between functions and technologies, that is, the

[1] Note that a product can be decomposed in modules, and consequently, each product and/or module has its own functional three. This paper focuses on the configuration mechanism behind the FT platform; therefore, for illustrative simplicity we do not discusses the relationships among functional threes.

presence of one function calls for the presence of certain technologies. In contrast, the other basic category reveals the negative connections between functions and technologies (i.e., the specification of one function rules out the adoption of some technologies). The decision on using a certain technology is affected by, e.g., the functional property values, the technology attributes. In this regard, the common FT structure provides the positive connection between general functions (or function types) and general technologies (or technology types), whilst it reflects the relationships, be it positive or negative, among functional properties, property values, technology attributes and attribute values. In summary, an FT platform contains all data describing functions, technologies and the mapping relationships that is necessary for companies to design a product family in the functional and technology domains. With the presence of similarity, the same types of data and relationships are organized into a number of classes, such as functional element classes, technology element classes and relationship classes. In addition, the relationships can be classified into three types: class-to-class, class-to-member, member-to-member.

Technology variant derivation. For given functional specifications pertaining to a product, the technologies and technology details, such as technology attributes and attribute values, can be derived from the common FT structure. Such derivation is accomplished using a number of selection and design rules. These rules can be constructed by considering the mapping relationships between the GFS and the GTS at both the higher abstract level and the lower detailed level.

2.2 Identification of the representation tool

In the literature, many representation/modeling languages or tools have been reported to accommodate representation/modeling in both a generic context and a specific problem domain. In this study, we focus on the analysis of an FT platform from the structural aspect, more specifically, the representation of the constituent elements and relationships. Three commonly adopted structure representation tools include the UML (http://www.uml.org/), programming modeling languages involving predicates and algebraic relations (e.g., Neville and Joskowicz, 1993; Joskowicz and Neville, 1996) and graph grammars (e.g., Finger and Rinderle, 1989; Hoover and Rinderle, 1989). As noted by Vernadat (1993) and Sutton et al. (1990), programming modeling languages have strong limitations in readability and understandability since they demand expertise in computer systems. Graph grammars focus more on individual components rather than the system as a whole. Due to the above inherent limitations, both representation languages cannot handle diverse types, large volumes of data, which is fundamental to an FT platform. Moreover, the complex relationships among the many data classes and instances in an FT platform make these last two representation tools inefficient. To clearly represent an FT platform from the structural aspect, the tool should posses the ability to not only explicitly capture the multiple data classes and their specific member instances in terms of the data structures and characteristics but also sufficiently model the many relationships among these classes and instances at different levels of abstraction.

Researchers, such as Larsen et al. (2001) and Yang et al. (2008), have recognized the necessity of representing various components and their complicated relationships using the object-oriented (OO) technology. Developed based on the OO technology, the UML has been recognized as a promising tool to model complex systems, be they software or non software (Rumbaugh et at., 1991; Booch, 1994). It has thirteen diagrams, including class diagram, object diagram, activity diagram, use case diagram, sequence diagram, etc. that are tailored to specify a system from different perspectives. The class diagram is to model a system from the structural aspect. In addition, a standardized extension mechanism is defined, in attempting to adapt this general modeling language to specific application domains. While the UML class diagram can accurately represent classes and their relationships, the extension mechanism is able to model member instances and relationships among them. The OO technique assists in capturing relationships between classes and members. In view of the many classes of functions and technologies, their instances and the interconnections among these classes and instances, we thus adopt the UML to represent an FT platform.

3. An Application Case

As a well-known international corporation in the semiconductor equipment manufacturing industries, ASML (located in the Netherlands) offers a variety of machines to the semiconductor industry. In this study, we focus on the design of functions and technologies for a family of microlithography machines. (Note, due to the confidential concern, the original data is modified without losing the capability to highlight the characteristics of this study.) A microlithography machine is a special copying machine, printing integrated circuit patterns from a reticle onto a wafer through a projection lens. It consists of a number of modules, including a wafer handler, a reticle handler, an illuminator, a wafer stage, a reticle stage, a projection lens and several electronics cabinets, as shown in Figure 2. Each module fulfills certain functions, which, in turn, are delivered by a number of technologies. For example, the illuminator and the projection lens provide and condition the light for the imaging process; the wafer handler and the reticle handler properly position the reticle and the wafer, respectively, in the imaging process. Due to the variations among module variants in the microlithography machine variants, the associated module function variants differ from one another in their specific functional properties and the corresponding technology attributes.

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 2 here>>>>>>>>>>>>>>>>>>>>>

For illustrative simplicity with losing generality, we use the wafer handler module as an example to explain functions and the relevant technologies. The function of a wafer handler system is to exchange a preconditioned wafer from a wafer preproduction track to a lithography stage. In this exchange process, a wafer must be precisely positioned at the right speed and with the exact temperature. To meet such requirements, this function is decomposed into eight subfunctions, including Control wafer temperature, Buffer, Get from track, Center and pre-align, Return to track, Put on wafer stage, Pick from wafer stage and Control contamination, as shown in Figure 3(a). Hence, its realization relies on that of these eight subfunctions. While these functions remain common among all variants of the microlithography machine family, the specific values of the functional properties, such as speed and position accuracy, vary according to, e.g., different customer requirements.

<<<<<<<<<<<<<<<<<<<<<<<<*Insert Figure 3 here*>>>>>>>>>>>>>>>>>>>>>>>

We take the subfunction – Control wafer temperature – as an example to explain functional properties, the possible values and their effects on the technologies adopted. The control wafer temperature function is characterized by several properties, including 1) the non uniformity in the wafer and 2) the offset of the air temperature inside the wafer stage compartment, as shown in Table 1. The first property describes the quality of the uniformity of the temperature in the wafer and, can assume two values: 20 mK (mK = millikelvin) and 35 mK. The offset of the air temperature inside the wafer stage compartment defines the counterbalance of the temperature in the wafer stage compartment with respect to the lens cool water or the wafer table. It can take on two possible values: 20 mK with respect to lens cool water and 100 mK with respect to wafer table.

<<<<<<<<<<<<<<<<<<<<<<<<*Insert Table 1 here*>>>>>>>>>>>>>>>>>>>>>>

The delivery of this function involves four technologies, including 1) Air conditioning system, 2) Pump system, 3) Remote temperature sensing and 4) Temperature control system. In most cases, each technology is characterized by a number of attributes; each attribute, in turn, can assume several values. Table 2 gives some examples of attributes for the four technologies together with some possible values. (Same as the function properties in Table 1, the attributes and the attribute values are, by no means, exhaustive.) The technology variants differ from one another in different configurations of technology attributes and the specific values that they assume. For example, the Air conditioning system is characterized by three attributes, including Air supply flow, Temperature offset and the Pressure stability. The valid configurations of specific attribute values lead to different air conditioning system variants.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Insert Table 2 here>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

4. Representation of Generic Functional Structure

In the functional domain, the intangible design elements can be classified into a number of types (or classes), including functions, product functions, function clusters, elementary functions and properties. Along with their class-to-class, class-to-member and member-to-member relationships, these elements form the GFS.

4.1 Entities

- *Function* – An entity of the type *function* represents a function variant which can be a product function at the top level of the GFS, an elementary function at the lowest level or a function cluster at any arbitrary intermediate level. A function is denoted with a phrase consisting of a verb and a noun, e.g., Provide warm air (a function of an air curtain). While a specific function instance represents one function variant, a function class (or a generic function) models a number of function variants of the same type.

- *Product function* – An entity of the type *product function* represents a function variant at the top level of the GFS. It represents the overall function of the system that a customer wants a product to perform. Entities of the type *product function* are defined according to specific customer requirements.

- *Function cluster* - An entity of the type *functional cluster* represents a function variant at any arbitrary intermediate level of the GFS. It comprises a group of functions, be they child function clusters or elementary functions. The constitute elements of a function cluster are logically connected with respect to, e.g., overlap, synergy, interfaces. Together with sibling elementary functions and/or function clusters, they contribute to parent function clusters at a higher level or the product function at the top level.

- *Elementary function* - An entity of the type *elementary function* represents an indecomposable basic function at the lowest level of the GFS. In general, elementary functions are specified based on, e.g., the functions of existing products, the new customer requirements, the designers' expertise.

- *Property* - An entity of the type *property* describes a characteristic of an entity of the type *function*. It, thus, defines a performance indicator of the product/component in consideration. An example of function property is Accuracy, which defines the precision need of the function: Position wafer. In most cases, one function has multiple properties. A property specifies a range of values from which one value has to be chosen in creating a function variant.

- *Value* - An entity of the type *value* is an assignment of an entity of the type *property*. A number of possible values can be assigned to a property. A specific function (i.e., a function variant) has a value for each of its properties. In other words, after all specific values of the corresponding properties of a generic function are determined, a function variant is obtained.

- *Property value* - An entity of the type *property value* defines a relationship between a property and a value. It is the combination of an entity of the type *property* and an entity of the type *value*. A set of specific property value pairs defines a unique function (i.e., a function variant).

4.2 Relationships

Following the convention in the UML, the diverse class-member relationships in the GFS are identified as follows:

- *Instance_of*. An *instance_of* relationship shows the connection between an instance and its class. For example, speed can be an instance of the class *property*; reduce speed is an instance of the class *function*; 3 is an instance of the class *value*.

- *Association.* An *association* denotes a relationship between two classes that involve connections between their instances. For example, in Figure 4(a), the connection between two subclasses P_iV (the set of value instances of the *i*-th property) and P_jV (the set of value instances of the *j*-th property) of the super-class *property value* indicates certain relationships. These relationships are reified by the links between their instances. They may be incompatible or compatible. The association (in Figure 4(a)) between *property value* and *property* indicates that a number of property values can be mapped onto the same property, whilst a one-to-one mapping relationship exists between *value* and *property value*.

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 4 here>>>>>>>>>>>>>>>>>>>>>

- *Link.* A link represents the connection between two instances. It itself is an instance of an association. A link may exist in three situations. In the first situation, it connects two instances belonging to two different classes. In case one class is the composition of the other, the relationship between the two instances is "a_part_of", meaning multiple child functions form a parent function. As shown in Figure 4(b), FC^*_{jj} is *a part of* SF^*_j. If the two classes do not have the composition relationship, the connection between the two instances is either compatible (*AND*) or incompatible (*XOR*). For example, in Figure 4(a), $P_iV^*_{ii}$ (the *i*-th value of the *i*-th property of a function) is compatible with $P_jV^*_{ji}$ (the *i*-th value of the *j*-th property of the same function), while it is incompatible with $P_jV_{jj}^*$ (the *j*-th value of the *j*-th property of the function). In the second situation, a link connects two instances of the same class. Such a link can only be *XOR* since the instances are options of a generic function (e.g., the link between FC_{ii}^* and FC_{ij}^* in Figure 4(b), between F_j^* and F_m^* in Figure 4(c)). It means at one time, only one instance can be chosen to represent the class. In the third situation, a link exists between instances of *function* and *property*, *value*, *property value,* as shown in Figure 4(c). Such link has a keyword ***has***. It indicates a function has properties and property values.

- *Dependency.* A dependency indicates a semantic relationship between two classes. The dependency between *property* and *value* means properties must be bound to actual values in order to express certain meaning. After the binding, a new class *property value* is created.

- *Shared aggregation.* A shared aggregation is a special kind of aggregation. It represents the relationship between a part and a whole, where the part can be a part in a number of wholes. The shared aggregation between *product function* and *elementary function*, *function cluster* and *elementary function* indicates that a number of product functions and function clusters can share a set of common elementary functions. Figure 4(b) shows the shared aggregation between product function and function cluster. Note, each product or module can have its functional three on its own aggregation level.

- *Generalization.* A generalization is the taxonomic relationship between a more general class (the super-class) and a more specific class (the subclass). Such a subclass is fully consistent with the super-class while adding additional information. The connection between *product function* and *function* indicates a generalization connection (see Figure 4(b)).

4.3 Selection rules

The construction of the GFS depends on a set of selection rules. Representing design knowledge, selection rules specify the circumstance under which a function is a constituent of another function or a particular function variant can be specified. They are defined to present the designers with feasible functional options. The general form of selection rules is as follows:

(consequent) IF (antecedent),

where the relationship OR ($\vee$) and AND ($\wedge$) can be applied to both antecedent and consequent. For example, for $P_iV_{ij}^*$, $P_iV_{jj}^*$, and $P_iV_{ji}^*$:

$$\{ F_i \mid F_{ij}^* \} \wedge \{ F_j \mid F_{ji}^* \} \text{ IF } \{ P_jV \mid P_jV_{ji}^* \} \vee \{ P_iV \mid P_iV_{ij}^* \} \wedge \{ P_jV \mid P_jV_{jj}^* \} \quad (1)$$

$$\{ P_jV \mid P_jV_{ji}^* \} \text{ IF } \{ P_iV \mid P_iV_{ij}^* \} \quad (2)$$

In the above rules, F_x represents the x-th function class (or generic function); F_{xy}^* denotes the y-th variant of the x-th generic function; P_xV is the set of value instances of the x-th property; and $P_xV_{xy}^*$ is the y-th value instance of the x-th property. Rule (1) indicates that when $P_jV_{ji}^*$ or both $P_iV_{ij}^*$ and $P_jV_{jj}^*$ are specified, two function variants F_{ij}^* and F_{ji}^* have to be selected together to form the parent function. Rule (2) means the necessary condition for selecting $P_jV_{ji}^*$ is that $P_iV_{ij}^*$ is specified.

Selection rules are defined taking into account a set of constraints related to, e.g., environment concerns, economic factors, customer requirements. These rules describe the compatibility of function variants and property values. Hence, in collaboration with functional elements and their relationships, selection rules can determine specific functions pertaining to customized products.

4.4 UML-based representation of GFS

The GFS represented using the UML is given in Figure 5. As shown, a function can be a product function, a function cluster or an elementary function at an arbitrary level of the GFS. A product function may consist of elementary functions and function clusters. In turn, each function cluster may contain its own child function clusters and/or elementary functions.

One or more functions can have a set of common property values; and the differences among such functions lie in the optional property values specific to each function. No function can assume two property values that are incompatible with each other. In addition, at the same level of the GFS, two functions with incompatible property values cannot be specified together to contribute to a function cluster.

<<<<<<<<<<<<<<<<<<<<<<<<<*Insert Figure 5 here*>>>>>>>>>>>>>>>>>>>>>>

4.5 Microlithography machine family case

Based on ASML's practice and the designers' expertise, the product functions, function clusters and elementary functions pertaining to the microlithography machine family have been identified. Also identified are the relationships among these functions, function properties and property values. Subsequently, all these function-related design elements have organized as the GFS, as shown in Figure 6. Note, due to the space issue, we show the part associated with the function: Control temperature. The properties and their values used in the Figure are described in Table 1.

<<<<<<<<<<<<<<<<<<<<<<<<<*Insert Figure 6 here*>>>>>>>>>>>>>>>>>>>>>>

5. Representation of Generic Technology Structure

Similarly, in the technology domain, the design elements can be classified into a number of classes, including technology, system technology, technology cluster, basic technology, attribute, tool, materials and expert. Along with the many relationships, these technology-related design elements form a GTS.

5.1 Entities

- *Technology* – An entity of the type *technology* is involved in delivering one or more function entities with the perceived physical effects. Together with others, a technology ensures a proper operation of a product, independent from the product's physical shape. It can be a system technology at the highest level of a GTS, a technology cluster at an intermediate arbitrary level, or a basic technology at an elementary level. Examples of technologies are mechanical engineering, software engineering, opto-electronics, hydrodynamics, robotics, etc.

- *System technology* - An entity of the type *system technology* represents a technology variant at the top level of the GTS. It consists of various lower level technologies that together deliver the overall function of an end product.

- *Technology cluster* - An entity of the type *technology cluster* represents a technology variant. It embodies a group of technologies that is clustered based on competences, functionality and physical considerations. Together with others, a technology cluster contributes to the technology cluster at a higher level or the system technology at the top level.

- *Basic technology* – A basic technology is an elementary unit at the lowest level of a technology hierarchy and, cannot be decomposed further. Basic technologies are associated with elementary functions. The determination of basic technologies, that are compatible to one another, is influenced by the attribute values in association with the corresponding function property values.

- *Attribute* – An entity of the type *attribute* defines a characteristic of a technology entity and, can assume a number of values. Usually, a technology is characterized by multiple entities of the type *attribute*. Typical attributes in mechanical engineering include surface roughness, heat treatment and material needed.

- *Value* – A value entity represents an assignment of an attribute entity. The attributes together with their corresponding values define diverse technology variants.

- *Expert* – An expert entity is normally an engineer responsible for developing and/or applying technologies. Experts are clustered based on their capabilities and expertise in technologies and the corresponding function clusters as well.

- *Material* – A material entity provides the resources that are necessary for experts to implement technologies based on the supporting tools (see definition below).

- *Tool* - The entity *tool* is a supporting devise for testing a technology entity. A tool can be either a physical type, a software or the combination of both. It is used or controlled by experts and requires materials in implementing or testing a specific technology.

5.2 Relationships

- *Shared aggregation.* **A s**hared aggregation represents 'a part of' relationship between a part and a whole. For example, a basic technology is a part of technology cluster; and a technology cluster is a part of a system technology.

- *Dependency.* The dependency relationship between attribute and value implies that attributes require values to express specific meanings.

- *Instance_of.* The same as that in the GFS, an instance_of relationship between entities in the GTS represents the connection between a class and its instances. For instance, the class attribute value has a number of value instances.

- *Generalization.* A generalization relationship shows the connection between a more general class and a more specific class of the same types of technology elements. The relationship between a super-technology class (e.g., lithography) and its sub-technology class (e.g., microlithography) indicates generalization.

- *Association.* An association indicates certain relationship between two classes of technology elements. As shown in Figure 7, entities of the type technology cluster are implemented by tools, which are used or controlled by experts. In implementation, the tools require certain materials.

<<<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 7 here>>>>>>>>>>>>>>>>>>>>>>>

- *Link.* The same as that in the GFS, a link in the GTS represents the connection between two instances of technology elements. The connection between an instance of *attribute value* and an instance of *technology cluster* describes a link, indicating a technology cluster variant is characterized by attribute values.

5.3 Selection rules

As with the construction of the GFS, the construction of the GTS calls for selection rules. These rules determine whether or not 1) two technologies can be adopted together to implement one function, 2) two attributes characterizing two technologies are compatible, 3) two attribute values defining a single technology are compatible, and 4) two attribute values associated with two technologies are compatible. For example, if there is an XOR relationship between two attribute values associated with one technology, the two attribute values cannot exist at the same time when this technology is adopted. The rule form is the same as selection rules in the GFS.

5.4 UML-based representation of the GTS

Figure 8 shows the GTS represented using the UML. It shows that technologies can be system technologies, technology clusters or basic technologies. Each technology has multiple attributes with a number of alternative value instances. Technologies are developed based on knowledge and expertise of experts, which use tools and materials. The required experts, tools and materials support the development of technologies. Therefore, the arrangement of technologies in technology clusters and system technologies delineates the structure of the organization of experts, tools and material.

<<<<<<<<<<<<<<<<<<<<<<<<*Insert Figure 8 here*>>>>>>>>>>>>>>>>>>>>>>

5.5 Microlithography machine family case

The technologies involved in developing the microlithography machine family have been identified as a priori. With these technologies, the GTS is constructed. Figure 9 shows the UML representation of the GTS pertaining to the wafer handling family.

<<<<<<<<<<<<<<<<<<<<<<<<<*Insert Figure 9 here*>>>>>>>>>>>>>>>>>>>>>>>

As shown, as constituent technologies, Air conditioning system, Pump system, Remote temperature sensing and Temperature control system form the wafer handling temperature technology. Each such technology has a number of variants (or instances), each of which has a specific values for the corresponding technology attributes. For example, Air pressure is one attribute characterizing Pump system. It can assume different values (e.g., $P_{21}V_{211}^{*}$ denoting the values less than 2.75 x 10^{2} (normal type), $P_{21}V_{212}^{*}$ representing the values less than 3.00 x 10^{-1} (vacuum type). A pump system variant, T_{2j}^{*}, takes on a specific value $P_{21}V_{212}^{*}$ of this attribute. The experts, tools and materials are involved in developing/applying these technologies and, play different roles. Also shown in the figure are the compatibility (AND) and incompatibility (XOR) between instances. For example, $AT_{13}V_{131}^{*}$ (denoting 5mbar/s) of Pressure stability is incompatible with $AT_{12}V_{122}^{*}$ (representing the values less than 50 w.r.t LCW) of Temperature offset.

6. Representation of the FT Platform

Underpinning an FT platform, the common FT structure is formed by unifying the GFS and the GTS. The unification is accomplished based on the mapping relationships between the two structures at both the higher abstract level (i.e., at the class level) and the lower detailed level (i.e., at the property/attribute values level). To determine specific technology elements for given function elements, design rules are defined.

6.1 Mapping relationships

The mapping relationships between the two structures can be categorized as *association* between classes and *link* between instances.

- *Association.* The association exists between multiple class pairs in the common FT structure, thus having a number of different meanings. For example, the association between *function* and *technology*, as shown in Figure 10(a), indicates appropriate technologies are required to deliver the corresponding functions; the association between *property value* class and *attribute value* enables the specification of technology attribute values based on given function property values; experts can be grouped in accordance with function clusters.

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 10 here>>>>>>>>>>>>>>>>>>>>>>>>

- *Link.* Connections between function-related and technology-related entity classes are embodied by the specific links between their respective instances. For example, in Figure 10(b) $P_iV_{ii}^*$ (the i-th value instance of the i-th function property) selects $AT_iV_{ii}^*$ (the i-th value instance of the i-th technology attribute) and E_{ij}^* (the j-th expert).

6.2 Design rules

Design rules are defined in line with a set of constraints in connection with technical boundary conditions, economical factors, customer requirements, etc. In addition, certain *a priori* known geometric and/or software interface restrictions exist to ensure sufficient margins to realize variability in technology development. Design rules allow designers to select the appropriate technologies, materials, tools and experts based on property values of desired functions. In other words, design rules tie functions in the GFS and the corresponding technologies in the GTS coherently. The general format of the design rules follows the same format of the selection rules.

6.3 UML-based representation of common FT structure

The UML representation of the FT platform based on the union of the GFS and the GTS is shown Figure 11. Within an FT platform for a product family, the technologies can be determined in accordance with the given function specifications. It shows that function properties in combination with their values have an influence on the specification of technology attributes with the corresponding attribute values. Appropriate implementation of technologies can deliver the corresponding function property values.

<<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 11 here>>>>>>>>>>>>>>>>>>>>>>>>

6.4 Microlithography machine family case

With the GFS, the GTS and the identified connections in between, the FT platform for the microlithography machine family has been constructed. Figure 12 shows the part pertaining to the wafer handle family. A set of function properties and their values are employed to derive the right technologies and select the compatible resources. In such a derivation process, selection rules are involved in both configuring valid function elements and the corresponding technology details. Some examples of these rules are given below.

$$\{ F_{1i} | F_{11}^{*} \} \text{ IF } \{ PV_{11} | PV_{11}V_{112}^{*} \} \wedge \{ PV_{12} | PV_{12}V_{121}^{*} \} \quad (1)$$

$$\{ FC_{1i}^{*} | FC_{11}^{*} \} \text{ IF } \{ F_{1i} | F_{11}^{*} \} \quad (2)$$

$$\{ T_{1i} | T_{11}^{*} \} \text{ IF } \{ AT_{11} | AT_{11}V_{111}^{*} \} \wedge \{ AT_{12} | AT_{12}V_{122}^{*} \} \wedge \{ AT_{13} | AT_{13}V_{132}^{*} \} \quad (3)$$

Rule (1) indicates that when the maximum Non-uniformity of the wafer is specified with the value 35mK ($P_{11}V_{112}^{*}$) and the Offset is specified on 20 mK ($P_{12}V_{121}^{*}$) the first variant is selected of the function Control wafer temperature (F_{11}^{*}). Rule (2) shows that F_{11}^{*} is a child function of the function cluster FC_{11}^{*}. Rule (3) describes that the technology variant, T_{11}^{*}, must be selected if Air supply flow assumes $AT_{11}V_{111}^{*}$ (denoting the values greater than 400 m3/hr); the pump system has an Air pressure of $AT_{12}V_{122}^{*}$ (representing the values less than ≤ 3.00 x 10^{-1}); and a Remote temperature sensing of the type 'Vacuum' ($AT_{13}V_{132}^{*}$).

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Figure 12 here>>>>>>>>>>>>>>>>>>>>>>>

Based on the FT platform, for given customer requirements, ASML's designers first specify the desired functions, properties and property values and subsequently, derive the appropriate technology variants with respect to technology attributes and their specific values. Table 3 shows both the function and the corresponding technology details that are derived from the FT platform for a customized wafer handler. In addition, the supporting tools, materials and relevant experts are also determined. With these function and technology details, the customized wafer handler can be designed.

<<<<<<<<<<<<<<<<<<<<<<<<<Insert Table 3 here>>>>>>>>>>>>>>>>>>>>>>

7. Conclusions

This article introduces the concept of function-technology (FT) platforms, which we define as a strategy that seeks to increase development reuse efficiency by formalizing the intangible functional and technical elements. To help both the researchers and the practitioners with a better understanding, we focused in this study on the constituent elements and the relationships inherent in an FT platform. More specifically, we addressed the structural representation of an FT platform. The case illustration shows that managing design reuse is likely to be more difficult because of the complexity and science-based nature of the products. As noted already, science-based products require different elements that constitute the platform. The findings presented here clearly show that both the formal representation as well as the case illustration allows for better representation of the platform elements of the FT platform. FT platforms have been proposed as an effective means for the semiconductor equipment manufacturing industries to reduce product development costs and time and to improve capacity allocation/utilization. Such reuse eventually leads to the efficient design of tangible elements (i.e., physical components) by supporting product development with configuring design variants based on a FT platform. In addition, the study complements previous studies by providing a clear definition of FT platforms. Viewed from the architectural perspective, an FT platform is underpinned by a common FT structure; it enables the derivation of specific technology details in accordance with the given functional details pertaining to a customized product. The common FT structure is formed by unifying a GFS with a GTS, which organize the data and knowledge in the functional and technology domains, respectively.

It is important to recognize that there are limitations in this research. When market diversity can be aggregated by defining physical product variants, it is better for the firm to apply physical platforms in defining variants. As customer involvement in design and engineering increases, the application of FT platforms becomes increasingly justified. When the level of customer involvement in product design and engineering is high (by for example a lead customer), the strategy can be based on FT platforms. As such, the components that could constitute a physical platform are changed frequently. The FT platform approach is applicable in situations where customer orders require additional engineering. Thus, the FT platform concept enables firms that deliver products based on additional engineering to reuse design elements. The study is, however, based on data of industrial machinery manufacturing, and therefore more case comparison with other branches of industries is needed for further generalization. Future research therefore could extend this work to understand the impact of FT platforms in other industries. Furthermore, we based the representation in this study on one assumption: the mapping relationships between the GFS and the GTS are available. In practice, due to the complexity involved, it may not be easy to obtain these relationships; and in most cases, these relationships are implicitly embedded in the large volumes of data existing in companies' databases. In this regard, future research may be directed to identify the mapping relationships between the functional and technology domains using, e.g., data/text mining techniques. While structural representation of an FT platform can present the constituent elements and relationships from a static viewpoint, it is not able to shed light on the dynamics of an FT platform, i.e., how the functions and technologies are reused, how the specific technology details are derived. Thus, another avenue for future research might be the dynamic modeling of an FT platform with respect to, e.g., reuse of functions and technologies, derivation of technology details.

REFERENCE

Alblas, A. and Wortmann, J.C., 2009, The need for function platforms in engineer or order industries, *Proceedings of International Conference on Engineering Design*, August 24-27, 2009, Stanford, USA.

Alblas, A. and Wortmann, J.C., 2010, Product platforms improve R&D efficiency in science-based equipment manufacturing: evidence from a case study in Microlithography, *Research Policy*, Under review.

Booch, G., 1994, *Object-Oriented Analysis and Design*, Benjamin/Cummings publishing company, Inc.

Chuma, H., 2006, Increasing complexity and limits of organization in the microlithography industry: implications for science-based industries, *Research Policy*, 35(3), 394-411.

Erens, F., and Verhulst, K., 1997, Architectures for product families, *Computers in Industry,* 33(2–3), 165–178.

Ericsson, A., and Erixon, G., 1999, *Controlling Design Variants: Modular Product Platforms*, New York: ASME.

Feitzinger, E. and Lee, H. L., 1997, Mass Customization at Hewlett-Packard: the power of postponement, *Harvard Business Review,* 75(1), 116-121.

Finger, S. and Rinderle, J.R., 1989, A transformational approach to mechanical design using a bond graph grammar, *Proceedings of the 1st ASME Design Theory and Methodology Conference,* September 1989, Montreal, Canada.

Halman, J.I.M., Hofer, A.P. and van Vuuren, W., 2003, Platform-driven development of product families: linking theory with practice, *Journal of Product Innovation Management* 20(2), 149-162.

Holmqvist, T., 2004, *Managing product variety through product architecture*, Ph.D. Thesis, Chalmers University of Technology, Gothenburg, Sweden.

Hoover, S. and Rinderle, J., 1989, A synthesis strategy for mechanical devices, *Research in Engineering Design*, 1(2), 87-103.

Joskowica, L. and Neville, D., 1996, A representation language for mechanical behavior, *Artificial Intelligence in Engineering*, 10(2), 109-116.

Larsen, M.H., Sorensen, C. and Langer, G., 2001, Development of a production meta product state model, *Computers in Industry*, 46, 275-287.

Lin, Y. and Zhang, W.J., 2004, Towards a novel interface design framework: function-behavior-state paradigm, *International Journal of Human-Computer Studies*, 61(3), 259-297.

Liu, X., Zhang, W.J., Radhakrishnan, R. and Tu, Y.L., 2008, Manufacturing perspective of enterprise application integration: the state of the art review, *International Journal of Production Research*, 46(16), 4567-4596.

McGrath, M., 1995, *Product strategy for high-technology companies*. New York: Irwin Professional Publishing.

Meyer, M.H. and Lehnerd, A.P., 1997, *The Power of Product Platforms: Building Value and Cost Leadership*. Free Press.

Muffatto, M. and Roveda, M., 2002, Product architecture and platforms: a conceptual framework, *International Journal of Technology Management,* 24(1), 1–16.

Neville, D. and Joskowica, L., 1992, A representation language for conceptual mechanism design, *Proceedings of the 6th International Workshop on Qualitative Reasoning*, August, 1992.

Oxford English Dictionary, 1989, Oxford: Clarendon Press, New York: Oxford University Press.

Pahl, G. and Beitz, W., 2003, *Engineering Design: A Systematic Approach*, Springer.

Robertson, D. and Ulrich, K., 1998, Planning for product platforms, *Sloan Management Review* 39, 19-32.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy F. and Lorensen, W., 1991, *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ.

Sanderson, S.W. and Uzumeri, M., 1997, *Managing Product Families*, Irwin, Chicago, IL

Sawhney, M.S., 1998, Leveraged high-variety strategies: from portfolio thinking to platform thinking, *Journal of the Academy of Marketing Science,* 26(1), 54–61.

Sköld, M. and Karlsson, C., 2007, Multibranded platform development: a corporate strategy with multimanagerial challenges, *Journal of Product Innovation Management* 24(6), 554-566.

Sutton S., Heimbigner D. and Osterweil L.J., 1990, Language constructs for managing change in processes centered environments, *Proceedings of the Fourth SIGSOFT Symposium on Software Development Environments. Software Engineering Notes*, 1990, 15, 206–217.

Ulrich, K. and Eppinger, S., 1995, *Product design and development*, McGraw-Hill New York.

Vernadat, F., 1993, CIMOSA: enterprise modeling and integration using a process based approach, In: Yoshikawa H. and Goossenaerts J. (Eds.) *Information Infrastructure Systems for Manufacturing*, 1993, North-Holland, Amsterdam.

Whitney, D.E., 1993, Nippondenso Co. Ltd: a case study of strategic product design, *Research in Engineering Design,* 5(1), pp. 1-20.

Wilhelm, B., 1997, Platform and modular concepts at Volkswagen – their effects on the assembly process, In K. Shimokawa,U. Jürgens, andT. Fujimoto (Eds.), *Transforming automobile assembly*. Berlin Heidelberg: Springer-Verlag.

Yang, W.Z., Xie, S.Q., Ai, Q.S. and Zhou, Z.D., 2008, Recent development on product modelling: a review, *International Journal of Production Research*, 46(21), 6055-6085.

Zhang, W.J., Lin, Y. and Sinha, N., 2005, On the function-behavior-structure model for design, *Proceedings of the 2nd CDEB Conference,* CD ROM Version, 18-20 July 2005, Alberta, Canada.

A LIST OF FIGURES

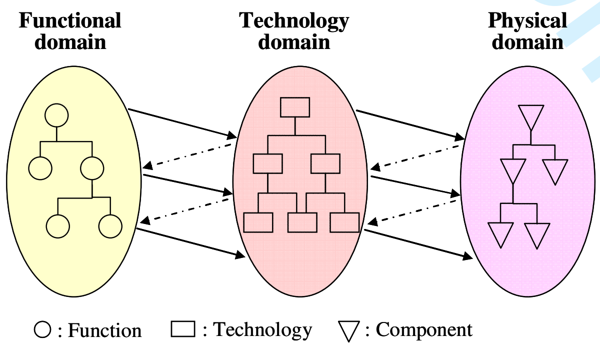


Figure 1. The three domains in product development (adapted from (Erens and Verhulst, 1997))

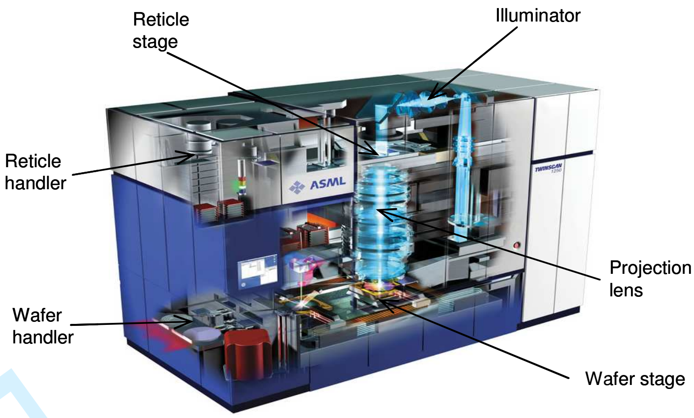


Figure 2. The major modules of a microlithography machine

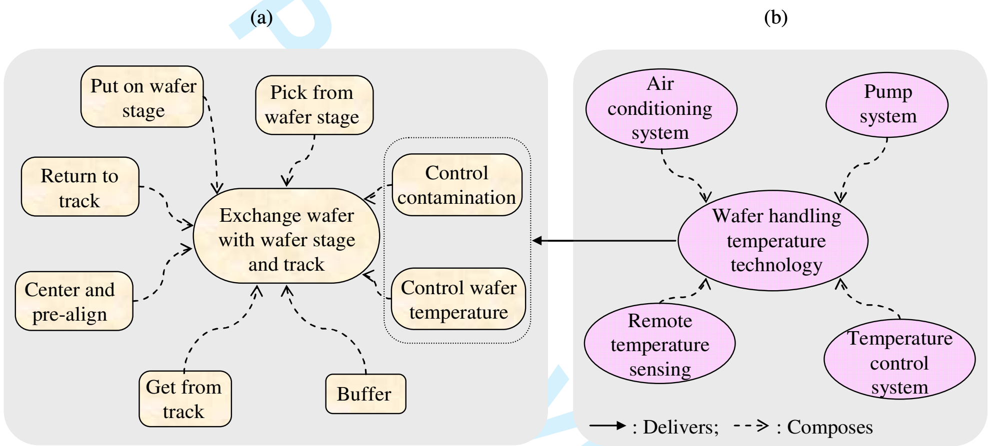


Figure 3. The functions and technologies associated with wafer handlers

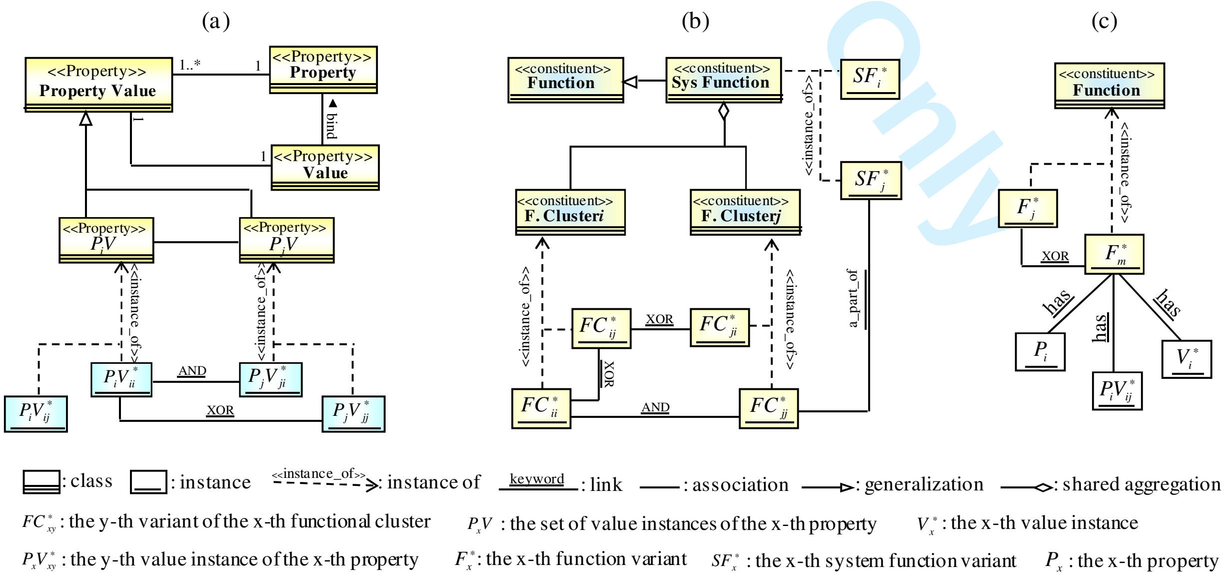


Figure 4. Associations and links in a GFS

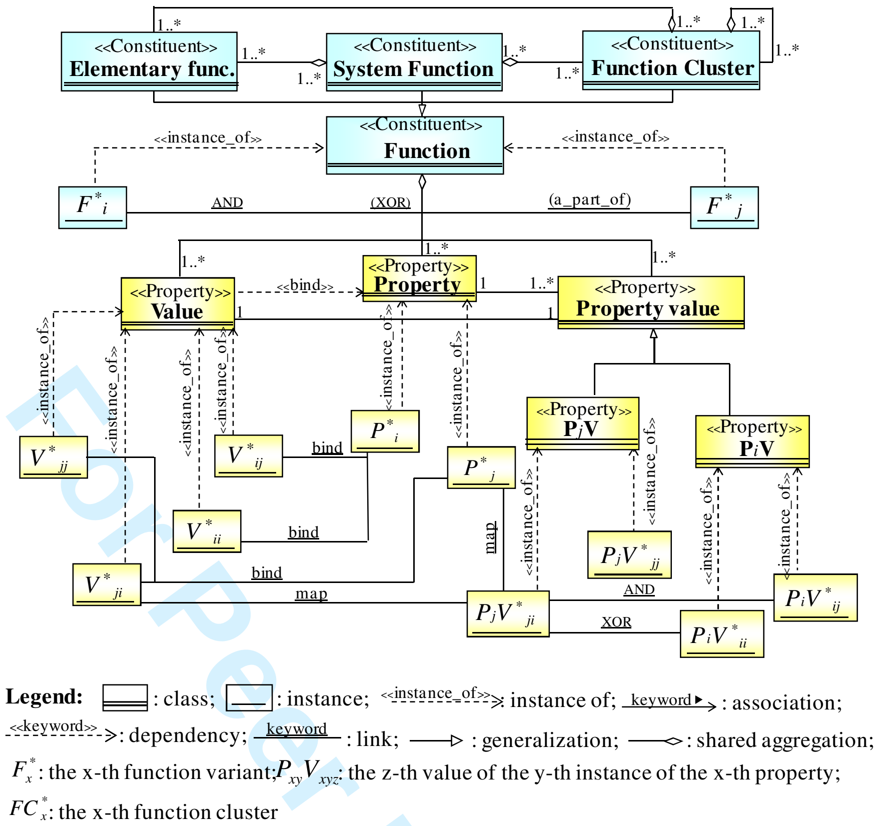


Figure 5. Representation of the generic functional structure

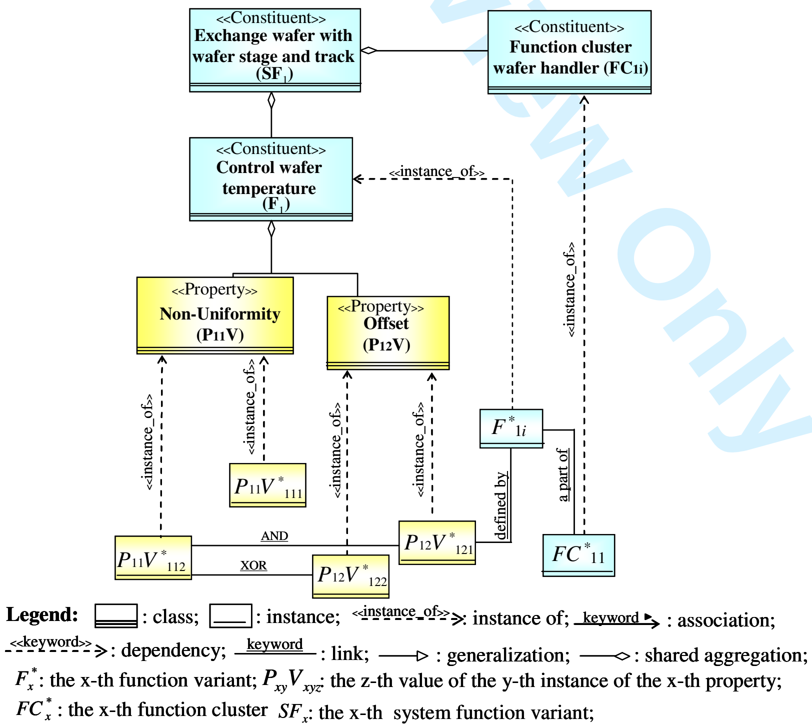


Figure 6. A partial GFS of the microlithography machine family

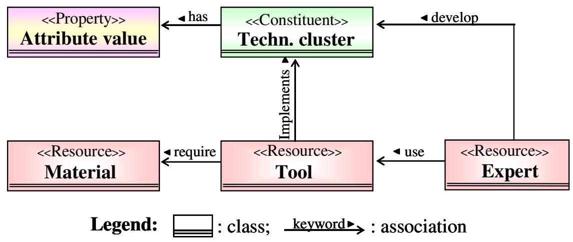


Figure 7. Association between technology elements

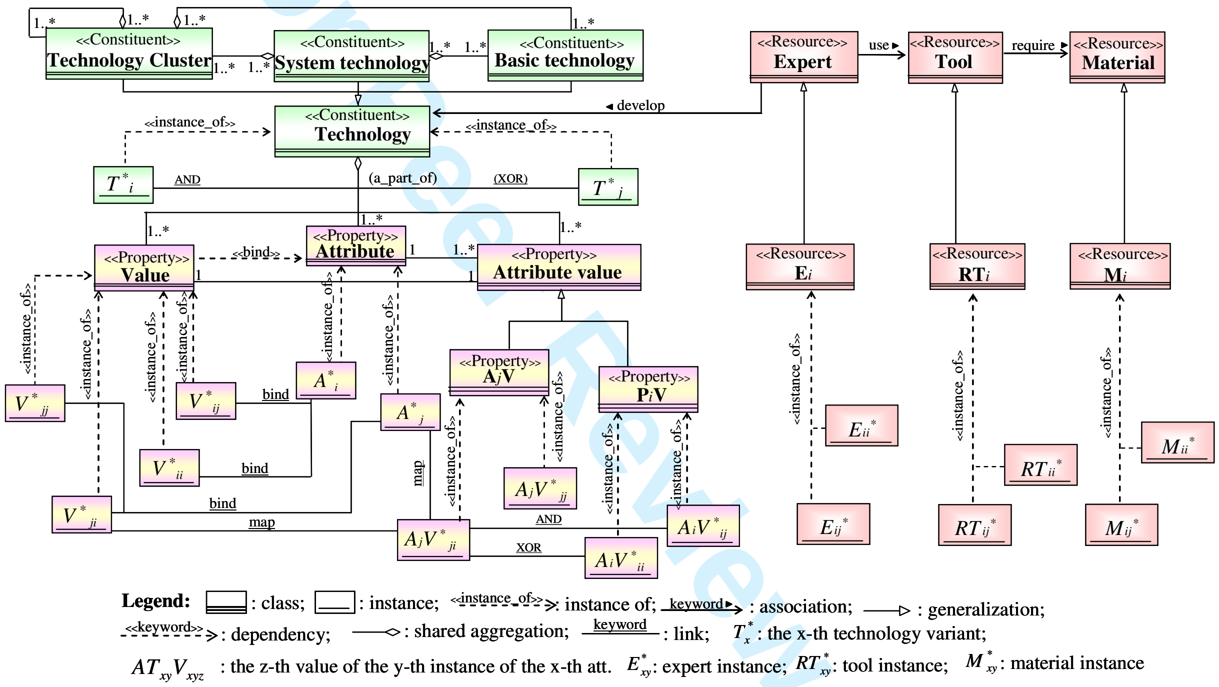


Figure 8. Representation of the generic technology structure

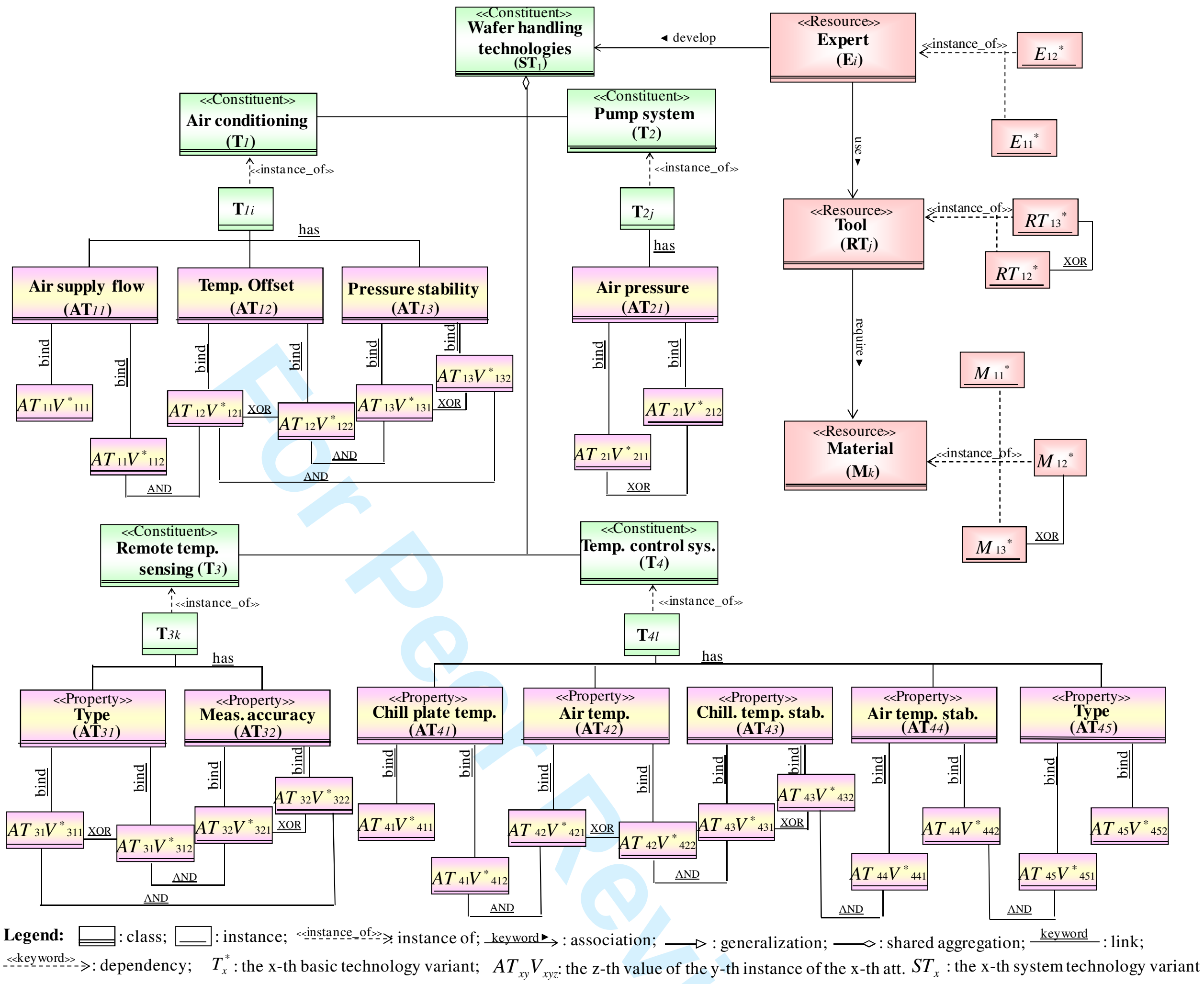


Figure 9. Partial representation of the GTS of the microlithography machine family

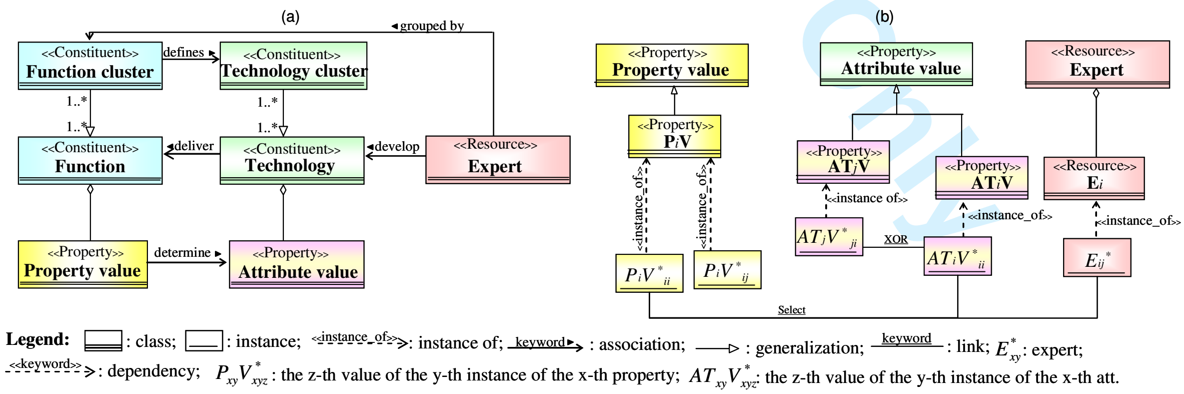
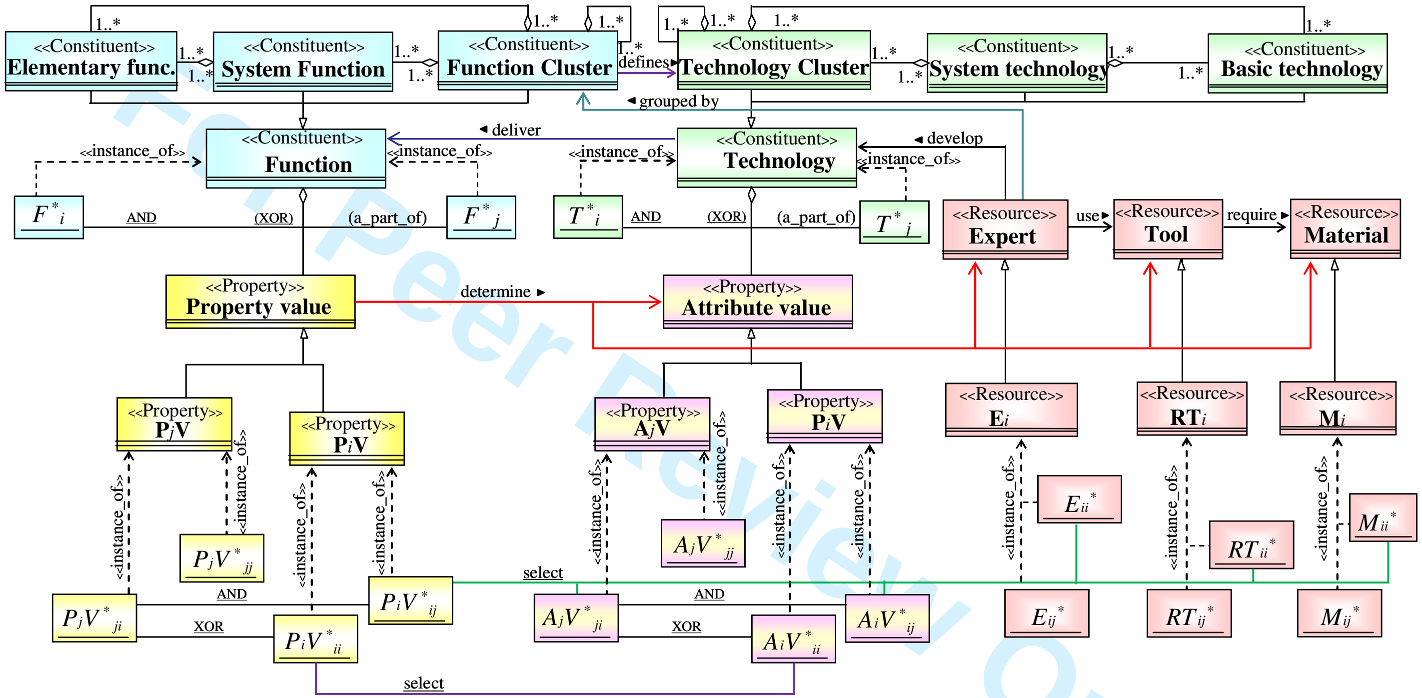


Figure 10. Associations and links between functional and technology elements



Legend: : class; : instance; <<instance_of>>: instance of; keyword: association; : generalization; : shared aggregation; keyword : link; <<keyword>>: dependency; F_x^*: the x-th function variant; T_x^* : the x-th technology variant; $AT_{xy}V_{xyz}$: the z-th value of the y-th instance of the x-th attributes; $P_{xy}V_{xyz}$: the z-th value of the y-th instance of the x-th property

Figure 11. Representation of entities and relationships in the FT platform

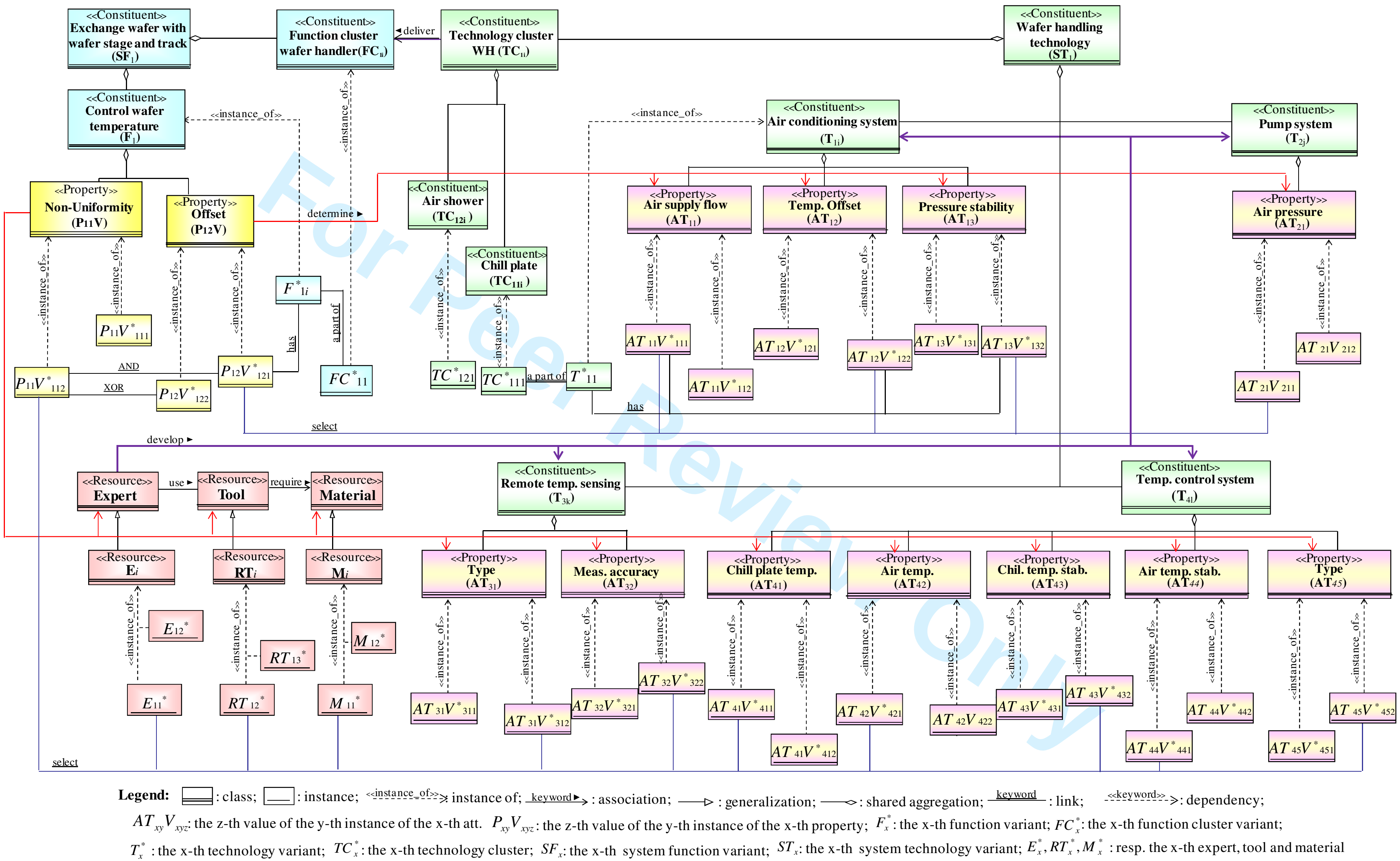


Figure 12. The partial FT platform for the wafer handler family

A LIST OF TABLES

Table 1. Properties associated with the function Control wafer temperature

Functions	Properties	Property descriptions	Possible values
Control wafer temperature	Maximum non-uniformity in the wafer.	The quality of being inconsistent uniformity among the temperature in the wafer.	20mK, 35mK
	Maximum offset of the air temperature inside the wafer stage compartment with respect to the wafer table.	The counterbalance of the temperature in the wafer stage compartment	20mK, 100mK
	Maximum offset of the air temperature inside the wafer stage compartment with respect to the lens cool water.	The counterbalance of the temperature in the wafer stage compartment	30mK, 110mK

Note: The unit of measure for both properties is mK (millikelvin).

Table 2. Technologies associated with the wafer handler function Control wafer temperature

Technologies	Attributes	Unit of measure	Possible Values
Air conditioning system	Air supply flow	Cubic meter per hour (m3/hr)	≥ 400, (300, 400)
	Temperature offset with respect to (w.r.t.) lens cool water (LCW) or wafer table (WT)	Millikelvin (mK)	≤ 100 w.r.t LCW, ≤ 50 w.r.t LCW, ≤ 50w.r.t WT, ≤ 25 w.r.t WT
	Pressure stability	Millibar per second (mbar/s)	5, 10
Pump system	Air pressure within the wafer handler	Pascal (Pa)	$\leq 2.75 \times 10^{2}$ (Normal), $\leq 3.00 \times 10^{-1}$ (Vacuum)
Remote temperature sensing	Type	Type	Normal, Vacuum
	Measurement accuracy	Millikelvin (mK)	0.1, 1
Temperature control system	Type	Type	Chill plate, Conditioned air flow
	Chill plate temperature	Celsius (C)	(22.5, 23.1), (21.5, 25.0)
	Air temperature	Celsius (C)	(21.5, 22.1), (20.5, 23.0)
	Chill plate temperature stability	Millikelvin per minute (mK/ min)	(12 mK/3min), (15 mK/5min)
	Air temperature stability	Millikelvin per minute (mK/ min)	(60 mK/3min), (70 mK/5min)

Table 3. Functional and technology details for a customized wafer handler

Function	Property	Values	Technology	Attributes	Values
Temperature control	Maximum non-uniformity in the wafer.	20 mK	Air conditioning	Air supply flow	≥ 300 m^3/hr
				Temperature offset	(400, 200)mK w.r.t. LCW
				Pressure stability	5 mbar/s
			Temperature control	Chill plate temperature	(22.5, 23.1)
				Air temperature	(21.5, 22.1)
				Chill plate temperature stability	≥ 2.4 mK/ min
				Air temperature stability	≥ 12 mK/ min
				Type	Chill plate
	Maximum offset of the air temperature inside the wafer stage compartment.	20 mK w.r.t. LCW	Pump system	Air pressure within the WH	≥ 3.00 X 10^{-1}Pa Vacuum
			Remote temperature sensing	Type	Vacuum
				Measurement accuracy	≥ 0.1 mK