The Concepts of IEC 61346 Applied to a Software Architecture for Automation

Technical Report IC/2004/20

Rodrigo García García
Software Engineering Laboratory
Swiss Federal Institute of Technology Lausanne (EPFL)
Ecublens
CH-1015 Lausanne, Switzerland
E-mail: rodrigo.garcia@epfl.ch

Esther Gelle
Information Technologies Dept.
ABB Switzerland Ltd
Corporate Research
Segelhof 1
CH-5405 Baden-Dättwil
E-mail: esther.gelle@ch.abb.com

Alfred Strohmeier
Software Engineering Laboratory
Swiss Federal Institute of Technology Lausanne (EPFL)
Ecublens
CH-1015 Lausanne, Switzerland
E-mail: alfred.strohmeier@epfl.ch

Abstract

The IEC 61346 standard establishes general principles for structuring the information of technical systems. The present document discusses the ideas shown in the standard, emphasizing the fact that some parts of it are ambiguous and can lead to different interpretations of the basic concepts. Consequently, we derive a concrete interpretation of the standard that tries to remove the ambiguities. We will apply this interpretation to the development of an industrial software platform for building automation applications.

The Concepts of IEC 61346 Applied to a Software Architecture for Automation

Technical Report IC/2004/20

I. The Standard IEC 61346

A. Introduction

The IEC 61346 standard [1][2][3][4] describes how to structure the information relative to a technical system. It is based on the idea that a system can be seen from different points of view and that the objects in a system can be organized hierarchically according to these points of view. The standard defines three key concepts among others. These are *object*, *aspect* and *structure*. It employs these concepts and a special notation to build up a reference designation system that identifies each object in a system.

A standardized description of a technical system such as a plant is useful in the context of large one-of-a-kind system projects where several integrators are involved. It supports a common understanding of the system itself and provides a basis on which integrators can discuss their proposed solutions. A common understanding of the technical concepts in turn eases integration of parts coming from different industries and it lays the grounds for standardized data exchange. Implicitly this requires the standard to be highly flexible in order to accommodate the description of a technical system under various aspects and viewpoints. From many industries the need for a flexible support during engineering, operation and maintenance has been pronounced. In the following, we often refer to the example of a plant as a highly complex technical system. In the engineering phase, the standard should help to manage the complexity of a technical system such as a plant in a top-down fashion. At the same time the result of the engineering process should be the documentation of the technical system. Engineers typically define and organize a technical system into subsystems and components before knowing their implementation. That is, functional modules are identified in the process, which later may be reused. The order fulfillment of a plant includes more than just product-oriented information, namely information on function but also on location of the various parts. Engineers have a need to switch between different types of information or structures that describe the different views or aspects of the system during engineering, installation and commissioning. In the processing of plant orders typically function, location and product-oriented information has to be managed simultaneously. In this context it is nevertheless important to maintain consistency between the various views. In the operation and maintenance phase of a plant, the distinction between occurrences, types and individuals is important [5]. It must be possible to retrieve and log operational information and attribute it to the correct occurrence and eventually also assign it to the correct owner of the information. A reference designation mechanism helps service engineers to identify the type and version of a specific occurrence and thus to provide the correct repair action with a fitting repair part. As a consequence, the standard has to provide open-ended structuring principles. An open-ended approach supports reuse of defined objects while maintaining their internal structure and designations. For example, a defined and documented water pumping system from an industrial plant could be reused in a power plant if the requirements on both subsystems are similar. One just would have to concatenate the defined system with the structure of the power plant. Note that such a reuse is not possible in the common power station designation system KKS (German for Kraftwerks-Kennzeichen-System [6]) for power plants, which defines absolute levels and has a fixed designation structure. Using KKS it is not possible to reuse the water pumping system from an industrial plant (where KKS is not applied) in a power plant, even if they would happen to be technically identical. The documentation would have to be re-done and the designations would have to be changed. Like IEC 61346, KKS defines a reference designation system for uniquely identifying equipment in a power plant. KKS defines three aspects for each object in a plant: the process related, the point of installation, and the location designation. As stated, KKS defines absolute codes, i.e. the function codes describing process related aspects are fixed, A standing for Grid and Distribution Systems, B for Power Transmission, etc.

B. Basic definitions of IEC 61346

An object, as defined by the standard, is an *entity treated in the process of design, engineering, realization, operation, maintenance and demolition*. We can summarize this definition by saying that an object is an entity of interest in the life-cycle of a technical system.

An aspect is defined as a *specific way of selecting information on or describing a system or an object of a system* (where a system is a set of interrelated objects). We can then see an aspect as a particular *view* of a system or object.

Structures are a way to organize the objects of a system by constituency relationships. Each structure is built around a particular aspect or view of the objects that compose the system. The standard considers three types of aspects in particular, yielding three corresponding structures: the function-oriented structure, the product-oriented structure and the location-oriented structure. The standard allows for concrete implementations to consider different aspects suitable for a specific technical field and to define the corresponding additional

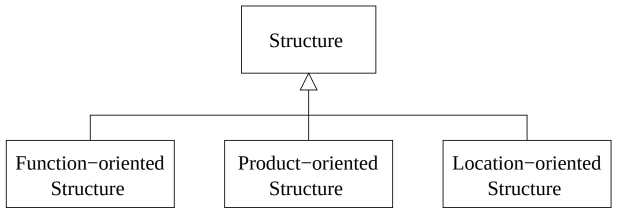
Structure
Function−oriented Structure
Product−oriented Structure
Location−oriented Structure

Fig. 1. The three structures defined in IEC 61346.

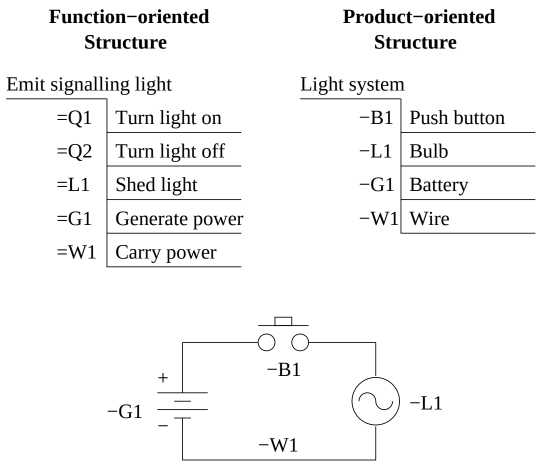

Fig. 2. Two views of a simple lighting system using structuring.

structures, but it only describes the three already mentioned (see Fig. 1).

Structures are graphically represented as trees, where the nodes of the tree that are up in the hierarchy are constituted by their children nodes. In Fig. 2, we see an example of a simple lighting system on which structuring has been applied. In this simple case, we have a one to one mapping between the two aspects of most objects: *Shed light* goes with *Bulb*, *Generate power* goes with *Battery* and *Carry power* goes with *Wire*. However, even in this simple example, there is a case of a two to one mapping. In effect, the product *Push button* performs two system functions, namely *Turn light on* and *Turn light off*. In general, there is a many to many mapping among the aspects in the strucures.

Contrary to views in other notations, the graphical representation of each one of the three structures is the same. In UML [7], for instance, different symbols and artifacts are used in the construction of the static view, the physical view or the use case view. In a standard about system architectures, the IEEE 1417 [8], the architectural description of a system is also composed of different views. A view in IEEE 1417 shows a system from the perspective of a set of concerns (areas of interest). A viewpoint specifies the set of conventions and artifacts used to develop an individual view. The standard IEEE 1417 provides us with the basis to build our own views of a system. In IEC 61346, all the views use tree-like structures showing constituency relations.

The reference designation of an object is built by concatenating the identifiers of its parent elements in the structures. Transitions among the structures allow the reference designation to cross structure boundaries. Each identifier is composed of a prefix, a letter code and a number code. Prefixes are established for the functional (=, equal sign), the product (-, minus sign) and the location (+, plus sign) structures. For an extended discussion about reference designations, see section 5 of IEC 61346-1. Starter **=W2M1Q1** is an example of a reference designation in Fig. 3.

II. Important concepts and their limitations

A. *The Definition of* Object

In an additional note, the standard explains that the *entity* mentioned in the definition of object may refer to a physical or non-physical “thing” or to a set of information associated with it. In our opinion, this definition is too vague. The standard aims at a wide variety of technical systems so it seeks for generality in its definitions. However, this particular definition suffers from a recursion problem. According to it, an entity can be a set of information related to itself. Conceptually, this is clearly a circular definition. The set of information associated to an entity is not the same thing as the entity itself. The representation of an object or the information related to an object should not be confounded with the *real-world* object it represents.

In an additional note to the definition of system (see note 2 in section 3.2 of IEC 61346-1), it is said that a system that is part of another can be considered as an object. From this note, it seems that subsystems and objects are interchangeable, so maybe there is no need of two separate concepts. In any case, we consider that the definition of system is useful to clearly state that one is talking about a group (or set) of objects.

B. *Content of the Structures*

The standard tries to clarify from the beginning what are the elements that the structures should hold. The examples shown in the different parts of the standard exhibit however disparate criteria for choosing the nature of the nodes that compose the tree-like structures.

The question is whether it is the aspects (views) of the objects which should be represented in the structures or whether the objects themselves should be the components of a structure. In the former case, each structure would hold the aspects that correspond to it (i.e. a function-oriented structure would hold function names). This option is supported by section 4.5 of IEC 61346-1, where it is said that an aspect of an object can be described in terms of the same aspect of other objects. In the latter case, objects should be organized in the structure according to a particular aspect (i.e. an object in a location-oriented structure would contain the objects that are placed inside it). This latter interpretation works well with product-oriented and location-oriented structures, but not with function-oriented structures. It is conceptually difficult to think that an object can be constituent of another object functionally. It is even more difficult if the function names are not shown explicitly. Some examples in the standard follow the first option whilst others use the second. There are even some examples which mix the two options altogether.

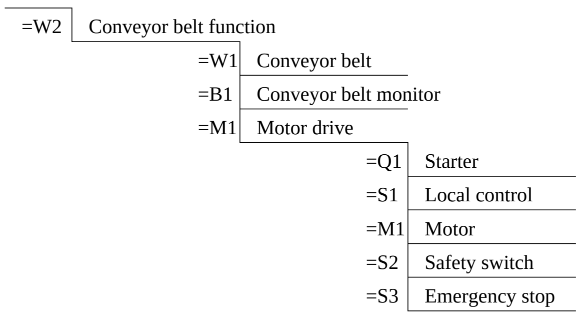

Fig. 3. Extract from Figure D.3 of IEC 61346-1.

Structure	**IEC 61346-1**	**IEC 61346-3**
Function-oriented	Functional aspect of an object	Object based on task or activities
Product-oriented	Product aspect of an object	Object based on equipment or devices
Location-oriented	Location aspect of an object	Object based on locations

Note: The examples shown in IEC 61346-1 do not conform to its interpretation.

TABLE I
INTERPRETATION OF THE ELEMENTS IN THE STRUCTURES.

The most prominent example of confusion due to these two ways of interpreting the contents of the structures is made explicit in figure D.3 of IEC 61346-1. In this figure (see extract in Fig. 3), we see the functional decomposition of a material-handling plant. This function-oriented structure holds, at the same time, the two types of structure elements we have seen: in the figure we can see a *Conveyor belt function* (functional aspect) and a *Conveyor belt* (physical object). Moreover, the figure is not only mixing the two kinds of elements, but it is also following a convention for naming the elements in the function-oriented structure that is against the recommendations of the standard itself. Indeed, there is an example for naming objects according to the function aspect in section 5.4.1 of IEC 61346-3, where it is recommended to *use "object for transporting from place A to B" instead of "conveyor belt"*. The advantage of this convention is that, in this way, the function aspect is not attached to a concrete product implementation, but it is exactly the opposite of what we find in the figure of IEC 61346-1. However, even if IEC 61346-3 recommends this advantageous naming convention, it promotes at the same time the use of objects to populate the structures. It talks about *objects based on tasks* as the elements that compose the function-oriented structure. This adds more confusion to what really are the elements of the structures, whether they are either aspects or objects.

The two different interpretations have a different impact on each of the three types of structure that the standard proposes. This is due to the diverse nature of the structures. Although the standard shows them as if they had a certain symmetry, they cannot be always treated in the same way. Therefore, we present a summary of the implications of choosing aspects or objects as the constituent elements of the structures (see also Table I).

1) *Function-oriented structure.*
   - **Aspects:** The functional aspect of an object is indeed a function (task or activity). The structures are thus populated with functions. The name of the function is explicitly shown.
   - **Objects:** The object is placed in the functional structure because the activity it performs is important for the plant. The name of the object is shown and the function it performs is implicit. A variant of this option appears in IEC 61346-3, where it talks about objects based on tasks, raising tasks to the category of objects. In that case, it is the name of the task which is shown.
2) *Product-oriented structure.*
   - In the product-oriented structure there is no significant difference if the elements of the structure are aspects or objects. They represent a product in any case.
3) *Location-oriented structure.*
   - **Aspects:** The location is the the name of the place or the coordinates where the object of interest is located.
   - **Objects:** The object is important because it determines a well defined space in the technical system, so it is placed in the location structure.

The possible misunderstandings are, in fact, recognized by the standard itself. A revealing note in section A.1 of IEC 61346-3 (Annex A) states that *both the object and the aspect of an object are often described with the same terms of function, product and location. This can sometimes lead to confusion.*

C. Transitions

An important part of the standard is the ability to relate elements of the different structures. One can navigate from one structure to another and thus designate an object by means of different aspects. In principle, transitions seem the ideal mechanism that would allow the engineer to obtain the products that are involved in the realization of a specific task or where they are located (see section 4.2 of IEC 61346-3).

However, the standard use transitions only as a method to build reference designations and not to show the relations between structures. The relations are supposed to exist and to be known (otherwise the transition would be impossible) but they are not directly shown. A transition in IEC 61346 goes from one aspect of an object to another aspect of the same object and then selects a child from the selected aspect (see Fig. 4). Our interpretation of transitions differs. We do not select a child of the destination aspect, but rather the aspect itself is considered the target of the transition. In this way, we use transitions to get the relations among structures. See section III-A for details.

The first part of the standard (IEC 61346-1) dedicates an extensive discussion to the way transitions should be done. Let us recall that, in this first part, the structures are supposed to hold the aspects of the objects. The discussion treats, among

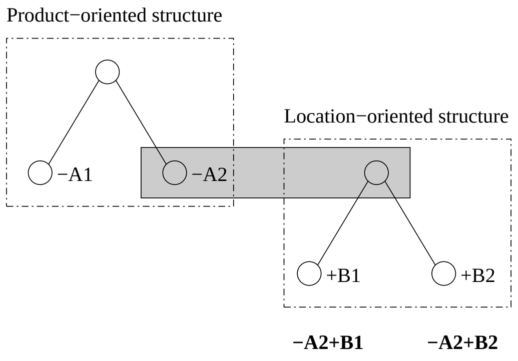

Fig. 4. Transitions as defined in the standard.

other issues, the case of transitions to a structure where an object has different aspects. For instance, an object can be represented by one aspect in the product-oriented structure and by two in the location-oriented structure. If a transition is made from the product to the location-oriented structure, the standard specifies how to unambiguously identify the concrete representation in the location-oriented structure (see Fig. 9).

Part three of the standard (IEC 61346-3) only discusses transitions briefly in section 6. There, the principle of constituency used to build structures applies somehow to the way transitions are done. For instance, a transition from the function aspect to the product aspect is only possible if the product completely implements the function. This kind of restriction is not stated in IEC 61346-1. In fact, this constraint eliminates the possibility of having ambiguous transitions like the ones addressed by IEC 61346-1 (see again Fig. 9). It reduces the applicability of transitions to those objects that have a many-to-one mapping from one aspect to another. In figure 12 of IEC 61346-3, where a transition is shown, it even seems that the product aspect that completely implements a function does not have to be a representation of the same object that is represented by the function aspect (it can be a higher-level object). This is in contradiction with IEC 61346-1, where transitions are supposed to occur between aspects of the same object. Moreover, the standard does not take into account that, if a product implements a function completely, all its superproducts will implement that function as well (see our proposed constituency rules in section III-B). As a consequence, a transition from the function-oriented to the product-oriented structure could reach any of those superproducts, as long as the function and the product aspects do not have to represent the same object [1]. We suppose that this is solved by not propagating the implementation of functions to the superproducts, but this point is not clarified in the standard.

Contrary to our interpretation, the interrelations between structures that require a many-to-many mapping are not taken into account as transitions in IEC 61346. These relations are supposed to be stored in a database in computer implementations of the standard (see again section A.3).

[1]We see here a consequence of the imprecise definition of *object* in the standard

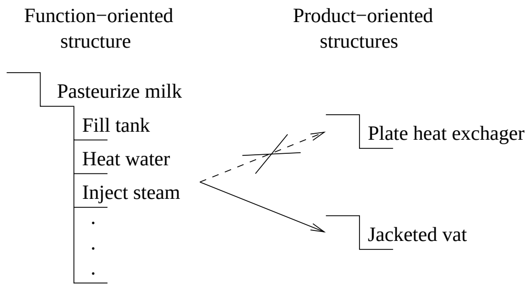

Fig. 5. Dependencies between structures.

D. Limitations of Structures

Structures, as defined in IEC 61346 only show constituency relations (see the definition of structure in section 3.6 of [1]). Transitions help to add more information by facilitating inter-structure navigation. Still, there are other simple relations among the elements of a function-oriented structure that cannot be expressed with structures. We are referring here to relations of precedence among functions. The fact that a function must be executed before another cannot be shown in a tree-like structure, which can only show the subfunctions. This kind of information has to be provided with additional documentation.

The structures do not take into account possible changes in the location of objects (i.e. objects that move from one place to another during system operation). The standard does not say anything about objects that can change its position in the system. The problem of how to model the products manufactured by the plant is related to this problem of product displacement. Moreover, the products which are being elaborated in the plant are not finished products and they go through several phases during fabrication.

The interdependency of structures is another limitation that is not treated in the standard. The standard seems to assume that each one of the structures can be developed independently and that is not always true: structures are not completely orthogonal. The contents of one structure will usually impose some constraints in the contents of the others. It is often the case that if one specifies functions in great detail, the lowest level functions will be determined by the products used to implement them. Let us see this with an example. Imagine that we have a plant for milk processing and we want to kill bacteria present in the milk with the help of pasteurization. We have in our plant a pasteurizing vat, which consists of a tank surrounded by circulating steam or hot water. The function *Pasteurize milk* can then be decomposed into subfunctions *Fill tank*, *Heat water*, *Inject steam*, etc. Imagine now, that we decide to change the pasteurizing vat by a plate heat exchanger. This device allows to pasteurize milk in a continuous way and it is quicker than the vat. In this case, the main function (*Pasteurize milk*) will remain the same, but its subfunctions (which depended on the use of a jacketed vat) will have to change.

The letter code system for reference designation shown in IEC 61346-2 has also one limitation. It classifies objects in terms of their main functionality and assigns them a letter code according to this functionality. It also suggests possible products that are suitable for performing the function. Thus, this approach is valid for the elements in the function-oriented structure and even for those in the product-oriented structure, but it is not adequate for the elements in the location-oriented structure. It would be good to have a better letter code assignment for the elements that specify locations.

III. Interpretation of Concepts in the Domain of Automation

As we have seen, the generality of the concept definitions in the standard and their multiple interpretations makes difficult for an implementation to claim conformance (see [9]). Before going any further, we decided to clarify the concepts we were going to use for our own implementation.

Although we try to lose as little generality as possible, we are limiting ourselves to a field of application of the standard. The platform we are building is intended to model automation plants, so we will impose some restrictions to the standard appropriate for this domain. For instance, in the construction industry environment, rooms could be considered as products. For us, rooms will just be space delimiters, so they can only be the location aspect of an object.

The standard supposes a certain symmetry among the structures it defines: they are all different views of the system. The function aspect, the product aspect or the location aspect of an object are not the object. They are just views of it. Although we agree up to some extent with this affirmation, we do not think that all three views can be equally related to a *real-world* object. It is obvious that the product view has a much closer relationship to the real object than the functional or the locational views. This is true up to the point that the two words are usually interchangeable in natural language: an object (or product) implements a function and is placed somewhere.

We think that the terms used in IEC 61346-3 when it speaks of *objects based on tasks* in order to refer to functional aspects, *objects based on equipment* in order to refer to products and *objects based on spaces* in order to refer to locations are an abuse of terminology (everything is an object). We also reject the idea of the object as a set of information, although when we model it, we deal with information related to the object. We conceive the object as something real that has an identity by itself and we do not want to confuse an object with a representation of it (as we said in section II-A) or with a conceptual idea that only lives in the mind of the engineer. This happens in section 5 of IEC 61346-4, for instance, where the life cycle of an object is described. The document discusses the creation of an object *motor*[2] in the function-oriented structure during the design process. In our opinion, the object is not created, it is only the functional specification (a representation) of the object. It seems that, in this case, the standard is not making a clear distinction between objects in the model and real-world objects.

For all the reasons explained above, we propose to modify the definitions in the standard. Remember, however, that we are limiting ourselves to a concrete field of application: automation plants.

- **Object:** Real-world entity relevant to the operation of an industrial plant.
- **Function:** Activity performed by the objects in the plant.
- **Product:** Information about the of an object in the plant.
- **Location:** Representation of a well delimited portion of space in a plant.

As opposed to the standard, which considers objects treated in the process of design, engineering, realization, operation, maintenance and demolition, we are only concerned with the objects which are important for the operation of the plant. It is up to the designer to decide which objects are relevant enough to put them in the plant model. It is important to note as well that we admit both physical and non-physical objects (such as software programs) in our definition of object. In any case, they refer to entities that exist in the real-world and not to entities of a model.

We think that, in natural language, a product is basically the same thing as an object. A possible difference is that a product is normally associated to something that can be purchased and typically identified by a serial number. For us, an object can be a single product or logical assembly of products. In the latter case, the object will not necessarily have an unique serial number and it can be built by plant engineers rather than purchased, but it is still a “logical product”. Because of this reason, we are thinking about the possibility of changing the name of the product-oriented structure and call it component-oriented structure or assembly-oriented structure. These names express better the real contents of the structure: not all the elements in the product-oriented structure have to be products but logical assemblies of products and parts of products. In this paper, we keep the term *product* in this general sense used by the standard. When we say that *a product implements a function*, we are extracting a relation from our plant model representing the fact that a real-world object is performing a certain activity. In this way, we avoid the confusion: we use the term *product* for the entities in the product-oriented structure (model) and *object* for the real-world entities.

It is also important to note that many automation plants generate manufactured objects as a result. When we talk about objects, we do not refer to this kind of objects. They will not be represented in the product-oriented structure since they do not participate in the plant operation but are the result of it. The figures related to the objects manufactured by a plant can be, however, important for management. Therefore, a software platform based on the standard should provide support for handling these figures, but not within the product structure because that is not its purpose.

A. Our Interpretation of Transitions

Transitions play a very important role in our interpretation of the standard. They are not only useful for building reference designations which include several structures, but also for showing other inter-structure relations (e.g. show the products that implement a function or the locations where products are placed). Some parts of the IEC 61346 standard consider this

[2]The name *motor* is used in IEC 61346-4 for designing an element in the function-oriented structure. As we have seen in section II-B, this is not a good choice for the name of an element in that kind of structure.

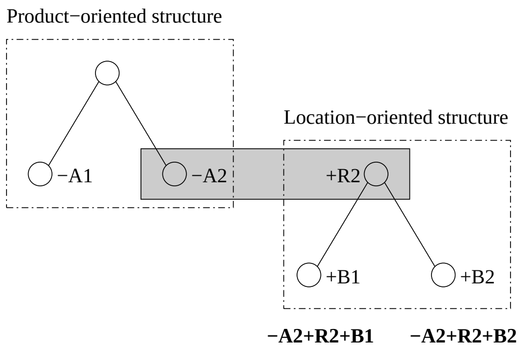

Fig. 6. The example of Fig. 4 with our interpretation of transitions.

latter functionality of the reference designation system apart from transitions (see II-C). However, we think that transitions are indeed the right mechanism to reflect these relations.

In IEC 61346-1, transitions are said to occur between two aspects of the same object. As we have seen in section II-C, the transition is finished when it designates a child node of the target aspect (see Fig. 4 or figures 27, 28 and 29 of IEC 61346-1). Transitions as in IEC 61346-1 cannot be used, for instance, to retrieve the product that implements a function, since they are considered by the standard as different aspects of the same object[3]. Instead, the transition from the function-oriented structure to the product-oriented structure will give, as a result, a reference designation for one of the subproducts of the product that implements the function. We think that this is limiting the full power of transitions, so our proposal is that transitions will be made between elements at the same level. That is, a transition from the function-structure to the product-structure will return a product that implements the function and not one of its subproducts. See Fig. 6 for an equivalent to Fig. 4 but with our own interpretation of transitions. Since we make transitions between aspects at the same level, the aspect +R2 of the example appears in the reference designations whenever we want to reference +B1 or +B2 starting from -A2.

As a result of our interpretation of transitions, one can go back and forth in a relation. In the example of Fig. 6, for instance, the transition can be made either from -A2 to +R2 or from +R2 to -A2. Nevertheless, reference designations should not contain loops. If we take the same example, that would mean that one should avoid the use of reference designations like -A2+R2-A2+R2+B1 because they are redundant.

In our opinion, and regarding their feasibility, transitions should always be possible from elements in the function-oriented structure to the product-oriented structure. The opposite would mean that there are some unimplemented functions in the plant. Only in the case that the plant model is unfinished may some of these transitions not be available. We differ from the IEC 61346-3 in this point, which has a much more restricted view about what transitions are possible (see again II-C). The same happens with transitions from the product-oriented to the location-oriented structures. All products must have been placed somewhere in the plant so

[3]It is not clear here whether the object should be given the same reference designation in both structures.

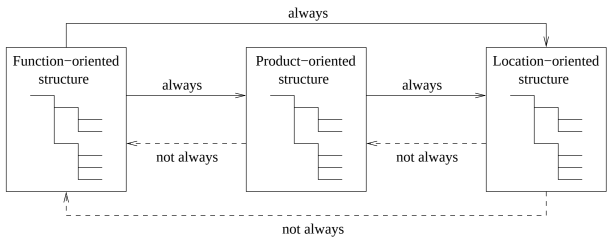
always
Function−oriented structure
always
Product−oriented structure
always
Location−oriented structure
not always
not always
not always

Fig. 7. Possible transitions between structures.

these transitions should always be possible as well.

A consequence of the asymmetry in our interpretation of the standard structures is that there is no direct transition between the function-oriented and the location-oriented structures. Transitions between these two structures are nevertheless allowed and transparent to the user. They are made indirectly by means of the product-oriented structure. Thus, when the user wants to know where are some functions located (a transition from the function to the location-oriented structure) we interpret that what he really wants to know is where are located the products that implement those functions. That is, we divide the transition function–location into function–product–location (see Fig. 7). The inverse happens with transitions from the location-oriented to the function-oriented structure.

However, there are some transitions that are not always possible. Let us imagine the case of a product delivered with a set of subproducts. Maybe not all the subproducts are useful to the plant operation but, since they come indivisibly with the product, they will appear in the product-oriented structure. If some subproducts do not implement any function, a transition from the product-oriented structure to the function-oriented structure will not be always possible. For instance, let us consider a simple circuit board with four NAND gates. The plant could use three of the gates for implementing a logical function and the fourth gate could be left unused. In that case, a transition from the fourth gate (in the product-oriented structure) to the function-oriented structure would be impossible. The same happens with the spaces defined in the location-oriented structure of the plant. Some of them can be empty of objects.

The IEC 61346 standard says that the single-level reference designation of an element must be unique within the same level in a structure. That allows the repetition of the same single-level reference designation for other elements only in other levels of the structure. The idea is that the full multi-level reference designation of the element will always be different. However, if the reference designation includes transitions, we can have the same full multi-level reference designation for different objects. Let us suppose, for instance, that the products `-T1-W1` and `-T2-W1` implement the function `=C2` in the model shown in Fig. 8. If we build a reference designation for these two products starting from the function-oriented structure, both would have the same: `=C2-W1`. In order to solve this problem, we suggest to go up into the hierarchy of the destination structure until we reach an element with a different single-level reference designation. All the hierarchi-

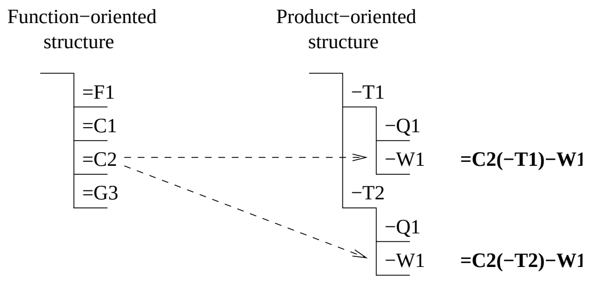

Fig. 8. Example of ambiguous multi-level reference designation.

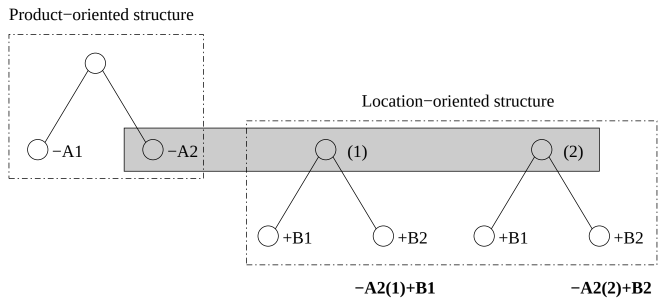

Fig. 9. Resolving ambiguous references in the standard.

cally superior elements needed for uniquely designating the transition target element will be enclosed in parenthesis. In our example, that would mean to write `=C2(-T1)-W1` for one of the products and `=C2(-T2)-W1` for the other. This guarantees the uniqueness of the reference designation since, by the construction rules of the structures, there will always be a parent element whose single-level reference designation will be different.

As discussed in section II-C, this problem is also treated in IEC 61346-1, suggesting the identification of the exact occurrence with a number in parenthesis (see Fig. 9 or figure 26 of IEC 61346-1). We based our solution on that idea. However, we have used the reference designation system itself to solve the problem instead of using a single number whose origin is not clear.

B. *Properties of Structures*

Following the discussion about transitions, we can state some properties of the structures based on the constituency ("is part of") relationship of their elements. The basic properties are the following:

1) If a product is located somewhere, all its superproducts are located (at least partially) in that same place.
2) If a product implements a function, all its superproducts are (indirectly) implementing that function.
3) If a function is implemented by a product, the implementation of its superfunctions also depend on that product.
4) If a location holds a product, all its superlocations hold that product as well.

The properties of functions related to products and vice versa can be derived from the basic properties. This is because functions and products are related by means of products, as seen in section III-A about transitions.

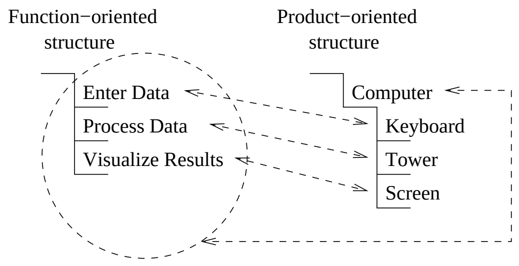

Fig. 10. Example of the properties of structures.

1) If a function is located somewhere, all its superfunctions will be located (at least partially) in that same place.
2) If a function is realized in one location, all the superlocations will realize that function as well.

As a guideline for building the plant model, we recommend to relate the lowest elements in the structures hierarchy. If possible, relate only the leaf nodes of different structures. The reason for this is that all the elements which are hierarchically superior to them will also be related to each other, as described by the properties shown above. The explicit relations will be thus the most concrete ones and the other relations will be derived by the system from the properties of the structures.

Let us explore one simple example. Imagine a *Computer* product composed by a *Screen*, a *Tower* and a *Keyboard* subproducts. Suppose that we also have defined the functions *Enter Data*, *Process Data* and *Visualize Results*, which are implemented by the computer (see Fig. 10, we do not show reference designations for clarity). Instead of assigning all three functions directly to the computer, it is better to be as much specific as possible and, for example, assign the function *Enter Data* to the *Keyboard*, *Process Data* to the *Tower* and *Visualize Results* to the *Screen* product. In this way, transitions can be made from the specific subproducts to their corresponding functions (something impossible if we assigned the functions only to the computer). But we keep the advantages of direct assignment as well. In this case, we also can make the transition from the *Computer* product to its three functions because, as shown in the properties of structures, a superproduct indirectly implements all the functions that its subproducts implement.

IV. Conclusion and Future Work

We have seen that the IEC 61346 standard is an ambitious document that sets the basis for modelling technical systems from a wide variety of domains. Due to this generality and to some ambiguities in its definitions and concepts, the standard is not applied as extensively as it should be. We have seen that a common standard would be useful for large projects where multiple vendors can participate and supply different products. By restricting ourselves to the field of automation and by removing most of the ambiguities, we have tried to improve the standard and make it suitable for the modelling of industrial plants.

We have started the development of a prototype software platform for developing industrial applications based on our interpretation of the standard. This platform will integrate different standards and concepts related to plant automation and offer a consistent view to plant engineers and operators alike.

V. Acknowledgements

Finally, we would like to thank Per-Åke Svensson from ABB and member of the TC3 (the IEC technical committee in charge of the publication and maintenance of IEC 61346). We greatly appreciate his help for answering our questions about the standard and for his extensive comments and suggestions during the writing of this paper.

References

[1] *Industrial systems, installations and equipment, and industrial products - Structuring principle and reference designations. Part 1: Basic Rules*, IEC Std. (6)1346-1, 1996.

[2] *Industrial systems, installations and equipment, and industrial products - Structuring principle and reference designations. Part 2: Classification of objects and codes for classes*, IEC Std. 61 346-2, 2000.

[3] *Industrial systems, installations and equipment, and industrial products - Structuring principle and reference designations. Part 3: Application guidelines*, IEC Std. 61 346-3, 2001.

[4] *Industrial systems, installations and equipment, and industrial products - Structuring principle and reference designations. Part 4: Discussion of concepts*, IEC Std. 61 346-4, 1998.

[5] P. Froehlich, Z. Hu, and M. Schoelzke, "Using UML for information modeling in industrial systems with multiple hierarchies," in *UML2002*, 2002, pp. 63–72.

[6] *KKS-Kraftwerk-Kennzeichen-System*, Verlag technisch wissenschaftlicher Schriften, VGB-Kraftwerkstechnik GmbH, 1988.

[7] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Booch, Jacobson, and Rumbaugh, Eds. Addison-Wesley, 1999.

[8] *IEEE Recommended Practice for Architectural Description of Software-Intesive Systems*, IEEE Std. 1471, 2000.

[9] R. García, E. Gelle, and A. Strohmeier, "A software architecture for industrial automation," in *Proc. Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC2003)*, Brisbane, Australia, Sept. 2003, pp. 315–320.