INTEGRATED PRODUCT AND PRODUCTION MODEL – ISSUES ON COMPLETENESS, CONSISTENCY AND COMPATIBILITY

Stellan Gedell[1], Anders Claesson[2] and Hans Johannesson[1]
(1) Chalmers University of Technology, Sweden (2) Saab Automobile AB, Sweden

ABSTRACT
Product development of complex products and their corresponding production systems continue to provide challenges in industry as well as interesting and challenging research questions. Recent research in the area has aimed at increased understanding and development of an integrated product and production system-modeling framework supporting cross-functional collaboration and concurrency. In this context, a well-known challenge in industry is the problem of how to ensure correct and complete sets of parts for manufacturing of different product variants. In striving towards integrated modeling capabilities this is one of several fundamental problems to be addressed. Thus, this problem has been in focus for the research work reported on in this paper. The work includes a framing of the concepts of completeness, consistency, and compatibility. Based on this framing a case study is conducted exploring the possibilities and implications involved in using the modeling framework to include supporting functionality. The case study is ongoing and preliminary findings are included in this paper.

Keywords: product development, product model, systems theory

1 INTRODUCTION
Systems engineering is an important field of research. It aims at a more systematic development process, characterized by a method- and model-based cross-functional collaboration and concurrency, supported by information management tools. Information management tools are an important pre-requisite to enable the required information sharing as well as provide for necessary traceability. Furthermore, explicit information carried in formal information management tools are a fundamental pre-requisite and starting point for knowledge capture and reuse. The dependencies identified between product and production implies a need to strive for cross-functional collaboration, which highlights the need for information models and tools capable of describing the product and the production systems using one integrated model.
Based on a systems theory approach, recent research has resulted in an integrated modeling framework supporting collaborative design of product and production systems [1]. Included in that work dependencies and interactions within product and production systems has been elaborated and an integrated product and production systems model is presented. However, a deceitfully simple question to request the model to produce a complete list of parts required to manufacture a specific product variant reveals the need to provide additional thoughts on the concept of *completeness*.
The work presented here is a first step towards increased understanding of the issues involved and their implications on some additionally needed capabilities in the modeling framework. In other words, the scope of the paper is mainly to problematize on *completeness*, *consistency* and *compatibility*. The ongoing case study serves to contribute empirical data to the discussion of the problem and as a source of increased understanding of the validity of proposed solution approaches as well as potential hidden challenges and pitfalls. First, a short description of the modeling framework in [1] is presented. Then, the issues of completeness, consistency, and compatibility are presented and elaborated. An ongoing exploratory case study aiming to enhance the modeling framework is presented. Finally, some conclusions and reflections are provided.

2 INTEGRATED PRODUCT AND PRODUCTION MODEL

Some of the important aspects of the modeling framework [1] used as a starting point for the work presented in this paper are outlined below for convenience and in order to highlight some important features and aspects of the framework. This is done through describing three important cornerstones. The first cornerstone illustrates some important fundamental aspects of the framework. The framework was originally proposed by Claesson [2], and further enhanced by Gedell [3], aiming to support *structured development of complex, variant rich and platform-based design by means of re-use and information sharing*. It has some important advantages to point out:

- It can represent any system of interest. This capability can be used to represent any abstract system while not being limited to, for example, physical parts.
- Multiple similar designs can be represented by one parameterized model, a model with a design bandwidth defined by its parameters. This can provide an overview and definition of the product range the design is capable of supporting, which is an important aspect in platform design. With reduced duplication of work the workload is minimized. Quality is improved since the possibility for mistakes is reduced as a consequence of a reduced number of models to maintain.
- A system is described by means of *design solutions*. The amount of details, in other words, the granularity of the description depends of the purpose of the model. Consequently, there is no *right* amount of details, it depends of what the model is intended to support. The level of detail is to some extent guided by the need to provide sufficient description of the performances to be expected from a particular design solution. Furthermore, the design solution – in its context – will collaborate with other design solutions resulting in emergent properties. Comprehensive system models include these emergent properties as well as how these arise from the collaborating subsystems.
- A design rationale model is used to explain why a design solution is chosen, in terms of what set of requirements that the design has to meet. The design rationale model consists of design solutions, functional requirements, constraints and relation objects [3]. The relation objects carries the information why a design solution is considered a good choice to meet the requirements. When design rationale [4] is included within the model its usability in terms of modification and re-use in highly improved. Those that are interested in the design can easier understand why a design solution is chosen when the reasoning behind that choice is presented together with the functional requirements and constraints that it fulfils.
- Extensive designs can inadvisable be model as monolithic units. Though a design is not the sum of its parts, as will be discussed in the next chapter, it is practical to break complex phenomena into parts. For example, the parts may easier be identified as usable in multiple designs with the advantage of economy by scale as one driver. Another example is when multiple stakeholders, organizations and companies want to have a clear division of responsibility. Extensive designs can be described as systems composed of sub-systems, which – in their turn – are composed of sub-systems in a recursive fashion. Composition includes how a system presents its configurability to potential super-systems as well as how a system selects and configures sub-system.

The second cornerstone in the integrated product and production model [1] are the interactions between the product model and the production model. The framework is based on systems theory and Hitchins [5] gives some valuable input to the importance and effects of interacting in the statement:

> *"A system is an open set of complementary, interacting parts with properties, capabilities, and behaviors emerging both from the parts and from their interactions"*.

There are several important aspects that can be extracted from this sentence.

- *A systems behavior* is a consequence of the system itself *and* its interactions with other systems as well as its own *internal* structure and *internal* interactions. In other words, it is not meaningful, nor possible to describe and understand the behavior of a system without considering its *context* as well as its *internals*.
- Thus, the behavior of a system is not simply the sum of the behavior of its parts, as opposed to reductionism. Decomposing, without a mechanism to model emerging behavioral characteristics of the system is a simplification and will consequently have shortcoming.
- Finally, even though not explicitly mentioned in the citation above, systems behave differently in

different stages in their lifecycles (Figure 1). This can be illustrated if we consider a system model of a car. When a car is being produced in a plant it can be seen as two systems (the car and the plant) that interact with each other. The car is in its manufacturing lifecycle, whereas the plant is in its use lifecycle phase. Focusing on the system model of the car, this model of the car has previously been in its definition (or development) stage of its lifecycle. Then, after being in its manufacturing lifecycle stage it will be entering its supply and use cycles of its life.

The temporal duration of super-systems formed by interacting systems varies. Super-systems can be formed with the intention to have a relatively long duration like the use lifecycle phase of consumer products. A super-system, which is formed to describe the production lifecycle phase of a product, will have a short duration. For example, when parts are placed in a fixture, to be positioned before welding, they together can be seen as a temporary system. Similarly, every interaction that takes place during a products production phase and the production systems are possible to view as short-lived super-systems.

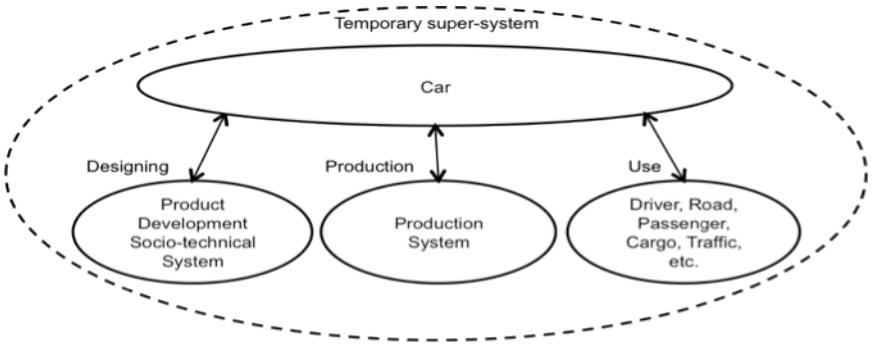

Figure 1, Super-systems formed of interacting systems during certain lifecycle phases.

The third cornerstone is the ability to allow multiple overlapping (partial) models. The interacting design solutions (Figure 2) can for modeling purposes be encapsulated in order to represent interacting systems for different purposes as indicated by the shaded areas with different colors to the right in the figure. Nothing restricts a design solution to participate in several different encapsulations.

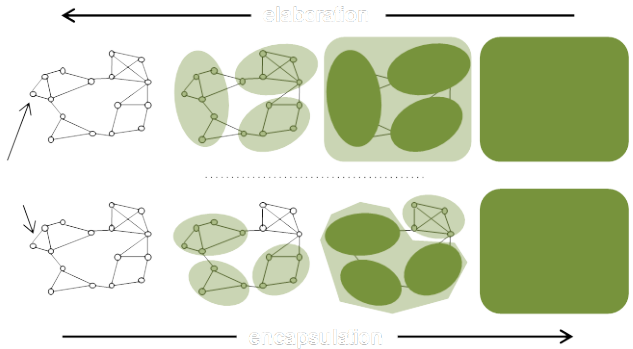

Figure 2, Two alternative elaborations and encapsulations [5].

Together these three cornerstones provide a foundation for the integrated product and production model [1]. How products and production processes relates to each other is facilitated by viewing the production processes as something going on within a production system, and similarly viewing the product (system) and production system as a temporarily formed super-system, i.e. a system in its own right.

Figure 3, which is borrowed from [1], is used to illustrate an integrated model. The *body-in-white* is composed of the *roof panel production system* (160) and the *roof panel*. That exemplifies how systems originating from different organizational parts of the company (product and production) seamlessly form an integrated model. The interaction *align with pin & hole*, describes how the production system's (the *fixture's*) positioning pin interacts with the product's (the *roof panel's*) positioning hole. Finally, the body-in-white and the *body shop* can exemplify a temporary system, as they together forms a system during the body-in-white's production phase.

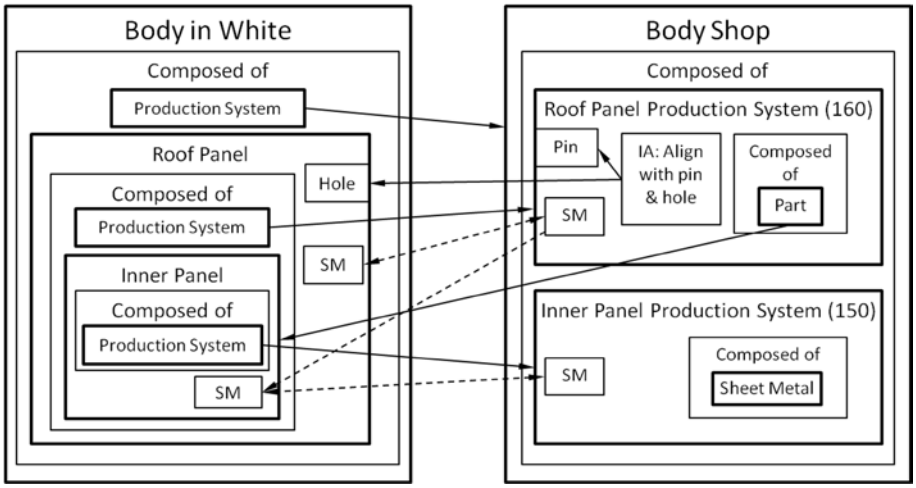

Figure 3. An integrated model representing a subset of a car and a subset of a plant [1].

3 COMPLETE, CONSISTENT AND COMPATIBLE

One of the key drivers behind the originally proposed framework [2] was the understanding of the product development process as a *journey* from an *incomplete* and *inconsistent* state of affairs towards a final gate "start of production" (or SOP) when the state of affairs should be characterized by a *complete* and *consistent* model capable of providing the information required to run large volume series production. Enabling a model to support an incomplete and inconsistent situation is rather easily achieved, for example, by simply avoiding putting any formal requirements on the model. The challenge, however, was not to just simply allow for an incomplete and inconsistent model. The challenge is rather to provide modeling mechanisms that allow the modeler to understand and identify when the model is incomplete and/or inconsistent in order to provide him opportunities to recover and correct such a state if it is of importance to do so. From an overall perspective, the speed of a product development process is equal to the speed of convergence from an incomplete and inconsistent model to a complete and consistent state.

The term *complete* can, deceitfully, be perceived as an absolute term, "When everything is there it is complete". However, an absolute definition vanishes at a closer look upon the issues involved. For example, using *Google define* we find two representative results from a search on complete: (1) *having every necessary (…) part*, and (2) *with all the necessary parts*. Instead of an absolute meaning, *complete* depends on a relative or subjective opinion of what is *necessary*.

Requirements and their solutions generally co-develop during the design activities. We initially focus on the solution part of a product description in order to provide a starting point for reasoning about completeness. The manufacturing of a product can be used to illustrate a concrete situation where completeness may be a problem. *When a product is manufactured as an assembly of parts, how do we know and ensure that exactly the required set of parts are selected and assembled?* The required set of parts required to assemble the product can be said to be *complete* in the sense that these parts are both required and sufficient to form the product – they are *necessary*. Adding or subtracting a part from this set would make the set either incomplete (one or more parts lacking) or redundant (one or more parts to many). This may seem as a trivial thing to achieve. However, when producing complex and variant rich products (like cars) there are a couple of thousand parts required per product and thousands of product variants possible to produce – each variant requiring a specific set of parts in order to be *complete*.

For the reasons mentioned above, the question – *What parts constitute a complete set of parts for the assembly of a product variant?* – is by far not trivial and most relevant. The situation described clearly shows that the answer to the question is that – *it depends*. It depends on which variant of a product that is to be produced. Clearly, different product variants will have different sets of parts depending on which features the product shall have as well as on the set of requirements it must fulfill. For example, a set of parts, that is complete for a low content product, will most certainly be incomplete for a high content product. The conclusion drawn from this is that *complete* is a relative concept that describes if some fundamental need is met. Our challenge is to explore if, and how, this issue can be handled using our modeling framework.

Thus, for the modeling framework to be able to make a statement about the *completeness* of a design, it is required to include a request, or expectation, of the modeled design. This request or expectation will establish a statement on what is *necessary*. The *criterion for completeness* is met when the modeled design solutions leaves none of the *explicitly stated requests* or *expectations* unanswered. The modeling framework, therefore, must include modeling mechanisms to formulate and establish these requests and expectations as well as modeling mechanisms allowing the modeled design solutions to provide *responses* to such requests and expectations.

Having provided a base for reasoning on completeness we now turn our interest to the concept of *consistency*. In logic, a consistent theory is one that does not contain a *contradiction*. The lack of contradiction can be defined in either *semantic* or *syntactic* terms. Consistency, in general, is also used in a slightly different way meaning a harmonious uniformity or agreement among things or parts or something of a regularly occurring, dependable nature. However, in this context we refer to consistency in the former meaning that consistency among a set of statements implies that no contradiction *logically* follows from these statements.

To include reasoning about consistency in our modeling framework we must examine it to identify where we are making statements that may lead to a contradiction. First we observe that unless we provide more than one statement about "the same thing" there is no possibility for a logical contradiction to arise. An example may be that we define a parameter, let's say a *length* of some design solution. As long as this parameter has no associated value, the model in a way could be said to be inconsistent, since a value is more or less required. However, it is probably more straightforward to view this situation as incomplete – we lack a value for a parameter. Then, someone assign this parameter a value. Now, unless the model contains other statements on this parameter and its value, the model is consistent and the value assigned to the parameter is simply a statement of fact – a kind of axiom. This clearly shows that in order to fruitfully discuss whether or not a model is consistent, the model must include mechanisms to make several different statements about the same entity. When all of these statements agree, we conclude that the model is consistent. Otherwise, we conclude that we have an inconsistency in our model. Looking at the model as a tool used to support product development during early phases on the journey from incomplete and inconsistent towards a more complete and consistent situation it is of no interest to just eliminate inconsistency, but rather provide support to identify inconsistencies in order to support understanding of the causes and thereby moving the sequence of design decision further.

As stated above, the modeling framework must allow that multiple statements can be made on the same entity (or fact). Another example of this might be allowing different stakeholders to use different evaluation methods to obtain a performance value. Even though the individually returned performance values may differ this is not a sufficient ground in itself to conclude that they are inconsistent. Provided that all these statements (performance values) according to *some criterion* agree with each other we still may conclude that the model is consistent. If they to some extent, again according to some criterion, disagree we may conclude that our model is inconsistent. Consistency as defined in our context can only be evaluated when there are more the one statement available about an entity, and a method as well as a criterion to determine whether there is a contradiction among these statements or not.

Moving the focus to the third issue – *compatibility* – we first recall that our integrated product and production-modeling framework is based upon a system oriented modeling approach. The framework provides several opportunities to represent complex and configurable systems that are defined and described as collections of collaborating *sub*-systems. The modeling mechanisms provided to define and describe these collaborations are primarily through *interfaces* and *interactions*. The modeling mechanism referred to as *composition* is used to identify which systems to include in such a collaborative collection. A consequence of collecting a set of systems with an expectation that they will collaborate is that the corresponding set of interfaces and interactions thereby obtained will – in a sense – *connect* and fulfill their expected behavior. Interfaces that in this way have been able to connect and fulfill their expected behavior can be viewed as *compatible*, i.e., they are capable of providing the requested and expected levels of collaboration. However, this is a conclusion that might not be possible to draw looking on a single interaction only. The reason for this is that we need to allow observation of emerging properties that we may have modeled on a "higher" system level. Even though an interface may seem locally ok, it may be the case that the results on important emergent properties are not as requested or expected. The implication here is that conclusions on compatibility

when viewing a system collection can be successively made starting at an individual interaction among a couple of interfaces and then successively propagating towards higher system levels ensuring that compatibility among the collaborating systems are maintained on all system levels. Another interesting issue regarding compatibility is that if offered capability meets or exceeds requested capability the request and response are compatible, otherwise offered capability is incompatible with request.

An important aspect to consider is that the quest for a complete, consistent and compatible model must not limit the models capability to handle incompleteness, inconsistency and incapability during the model's design lifecycle. In the design phase incompleteness and inconsistency must be allowed, for example due to conflicting design alternatives or stakeholders' prioritizations. To rigid processes or tools will hinder design activities and likely create frustration, for example due to reduced organization efficiency or deviation from prescribed processes and rules. To summarize, it is necessary for a design model to support incomplete and complete, inconsistent and consistent, and incompatibility and compatibility.

4 IMPLICATIONS FOR THE INTEGRATED MODEL

The modelling framework as it has evolved to the state described in [1] does not really include any mechanisms to support the kind of reasoning described above regarding completeness, consistency, or compatibility. In order to enable our modeling framework to include mechanisms to support issues on completeness, consistency, and compatibility we must provide some form of automated, or semi-automated, reasoning support. Formally, *automated reasoning* is a research area in its own right (e.g., see [7]). The objective of automated reasoning is to write computer programs that assist in solving problems and in answering questions requiring reasoning [8]. In a semi-automated reasoning such a program is used in an iterative fashion; that is, you can instruct it to draw some conclusions and present them to you, and then, based on your analysis of the conclusions, it can in the next run execute your new set of instructions. Alternatively, you can use such a program in a batch mode, that is, you can assign it an entire reasoning task and await the final result. The intention in our case is to enhance the modeling framework with some *basic capabilities* to provide for a first step towards a *semi-automated reasoning* with focus on our issues concerning completeness, consistency, and compatibility. An interesting overview of different forms of automated reasoning is provided in [9].

Reasoning is a process of drawing conclusions from facts. For the reasoning to be sound, these conclusions must inevitably follow from the facts from which they are drawn. In other words, reasoning is *not* concerned with some conclusion that has a good chance of being true when the facts are true. Indeed, reasoning as used here refers to *logical reasoning*, not of *common-sense reasoning* or *probabilistic reasoning*. The only conclusions that are acceptable are those that follow *logically* from the supplied facts.

This rather strict definition of reasoning is not really what we are aiming for in our ambition to provide a modeling framework capable of supporting concurrent product and production development. An engineering solution does not seek to claim that it is *logically right* – an engineering solution is one solution – among many potential solutions – that valued in the context of a set of expectations and requirements is *good enough*. What makes it so difficult is the vast amount of design parameters possible to decide upon and the many performances upon which expectations and requirements are placed. A further complication is that many of the most important performances upon which we place expectations and requirements are emergent properties on higher system levels and thus very difficult to attribute to any particular set of design parameters where the design decisions actually are taken. The consequences of the design decisions emerge from a whole range of design decisions rather than from any one decision in particular.

The mechanisms available to us within our modeling framework to start our journey towards providing some form of semi-automated reasoning along the thinking outlined above are primarily our parameters. In [2] three semantically different kinds of parameters were distinguished: *design parameters*, *performance parameters*, and *variant parameters*. The understanding of these are that *design parameters* are those parameters that a design engineer or a decision maker can influence and decide upon their values in order to form design solutions in accordance with their intentions. *Performance parameters* are additional parameters that provide information about the consequences or outcomes from the design solutions in terms of observable properties of interest. The understanding of how parameters depend on other parameters is captured introducing a new modeling element referred

to as a *parameter map*. *Variant parameters* are a kind of convenience mechanism that, for example, enables us to refer to huge sets of parameters with one simple statement of a value of a variant parameter. Conversely, the value of a variant parameter may be derived from an observation of the values of a set of other parameters, thereby providing a sort of automated categorization of which variant we currently are dealing with. With these basic modeling elements in place the first step towards an ability to provide a simple form of semi-automated reasoning is in place.
Another mechanism we will need to introduce is a possibility to define *expressions* and/or *constraints*. For example, we will need to establish a constraint expressing that a certain performance value must exceed a required value. Another example might be the performance parameter *weight* that we would like to minimize while also requiring it to be below a threshold value. Since we also want to support multiple opinions and allow for more than one statement on an entity, we furthermore need to provide explicit mechanisms supporting which statements we are taking into account during an evaluation of a constraint or expression as well as how we arrive at a certain conclusion. Yet, another issue to provide for in the modeling framework is how to initiate and trigger evaluations of constraints and expressions as well as how conclusions and results from parameter mappings are allowed to propagate forming a chain of successive mappings and conclusions. The steps taken regarding these concerns in the work reported here only touch upon these subjects in a basic and simple manner. More elaboration on this particular topic is beyond the scope of this work.

5 AN EXPLORATORY CASE STUDY

The purpose of the exploratory case study is to apply the thoughts on completeness, consistency, and compatibility outlined above. Further, the aim is to enhance the modeling framework in practice and examine the capabilities achieved through the enhanced framework. The approach taken in pursue of this exploration was to apply the framework in an attempt to model a car program (products) and the production system required to manufacture these products. The intent is to apply the framework and use it to describe and define current and next generation products and production system(s) as well as the platform(s) upon which these are based and derived. The study is conducted in collaboration between academic and industrial partners and based upon accumulated industrial experiences as well as research results obtained from many years of research in the area. The models created shall include solution bandwidths, architecture definitions, and definition of the platform(s). The approach is to define and maintain a complete and consistent holistic model while continually refining, detailing, and extending the model through elaboration and encapsulation.
As mentioned in the introduction, information management tools are required prerequisites for dealing with these models. Since the authors are unaware of any existing tool that can be used to capture, maintain and manage the information model defined in the framework, the study also include the creation of a prototype tool with enough functionality to work with the modeling aspects in focus of the study. Creating and using this prototype tool will provide valuable insights in itself and be a learning platform both in terms of modeling methodology and in terms of usability requirements on a future and more efficient tool.
From a scientific point of view the expectations on the case study are that it will provide both empirical validation of the proposed integrated product and production modeling framework and new insights in new questions for future research. From an industrial point of view the case study will enable an update on the modeling frameworks state-of-art and subject it to some relevant industrial issues in order to gain understanding about current modeling capabilities as well as experiences and knowledge about important issues to develop further in the future.

6 CONDUCTING THE CASE STUDY

The aim of the case study is to examine the capability of the enhanced modeling framework regarding the defined issues on completeness, consistency and compatibility. In order to achieve this, the model will have to include both system level aspects such as product variants and performance expectations and detailed design decision on design parameters and the consequences of these in resulting performances. Furthermore, it is of interest to include possibility to model both physical and functional interactions on physical part level as well as emerging performances on higher system levels. A choice to model the chassis system in a car was made while it provides all the above opportunities and also includes well-known system level performance expectations (*braking distance* of a car). A chassis system of a car also provides many opportunities to model product variation.

Besides the modeling of an example system using the framework, the case study includes creating a prototype information management tool. There are two main reasons. First, the authors are not aware of any existing tool with the functionality to host the modeling framework and its required functionality. Second, the expected extensions to the modeling framework required in order to address the described issues on completeness, consistency and compatibility are not known in detail as to what functionalities the modeling framework and the information tool must be capable of providing. The case study is expected to shed some more light and understanding on these aspects.
Creating an information management tool for the proposed framework is by no means a simple and straightforward task. The first issue to deal with is that the definition and documentation of the framework is provided through the description and references provided above. As a consequence, any missing or ambiguous elements must be given complementary and assumed definitions. The second issue to deal with is that this work in itself is of an exploratory character having the implication that it is not entirely known beforehand exactly what has to be included in the information management tool, nor what functionalities it is expected to be able to provide. Both issues combine to a very ambiguous, unclear, and incomplete situation and foundation for creating an information tool. Thus, if this tool were to be created by a third party, the amount of work required to bring clarity to these issues would be almost overwhelming and require a lot of time and resources to be spent on creating more formal requirements for this tool. The approach taken in this exploratory study is to define and create the tool in parallel with the ongoing modeling and conceptual work.
In order for this to be feasible the information tool is conceptually divided into two major areas of functionality: information capture and information visualization. The work conducted so far has been to enable information capture of all (or most) modeling entities defined in the modeling framework as well as extending the framework with some modeling entities discovered to be of vital importance in order to address the research questions on completeness, consistency, and compatibility.

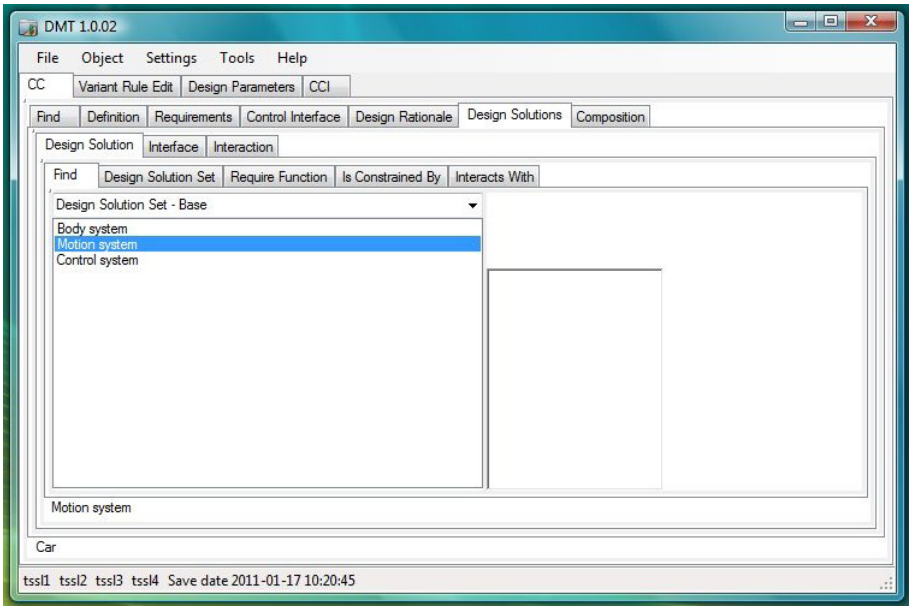

Figure 4. User interface to capture and define modeling entities.

The prototype information tool is developed using a C#-environment and is initially a single user and standalone application using simple files for data storage.
The user interface of the tool follows a more or less one-to-one mapping of the information model defined in the framework. Several sets of tabbed pages (Figure 4) have been used to provide an easy contextualization for each of the model entities to be defined.

The approach to create a product model including necessary elements to explore completeness, consistency, and compatibility takes is illustrated by Figure 5. The approach utilizes as a starting point, those physical parts of the chassis system that are required in order to manufacture a vehicle, illustrated by *rotor* and *brake pad* in the figure. Since the chassis exists in several different variants several sets of physical parts will be included in the model. The variability is represented in the figure by parameters, e.g. two rotor diameters. Depending on the product

variant to build, this starting point provides requirements on the model to define how product variants will utilize different sets of parts. Thus, the model must include several additional model elements representing higher system levels, exemplified by *chassis*, until the vehicle system level, i.e. *car*, is modeled and described. On vehicle level both performance expectations (e.g. *braking distance* in fr:braking) and vehicle variants (e.g. *sportiness* derived from *driving experience*) are added to the model, thereby providing a starting point for examination of completeness as well as consistency. Starting with adding model elements on the vehicle level requesting a certain level of performance (in this case exemplified by *braking distance*) a request for a performance response has been defined. Until such response is provided the model is incomplete. In order to resolve this incomplete state the model must include additional elements capable of providing a connection between part level performances and delivered performance on vehicle level.

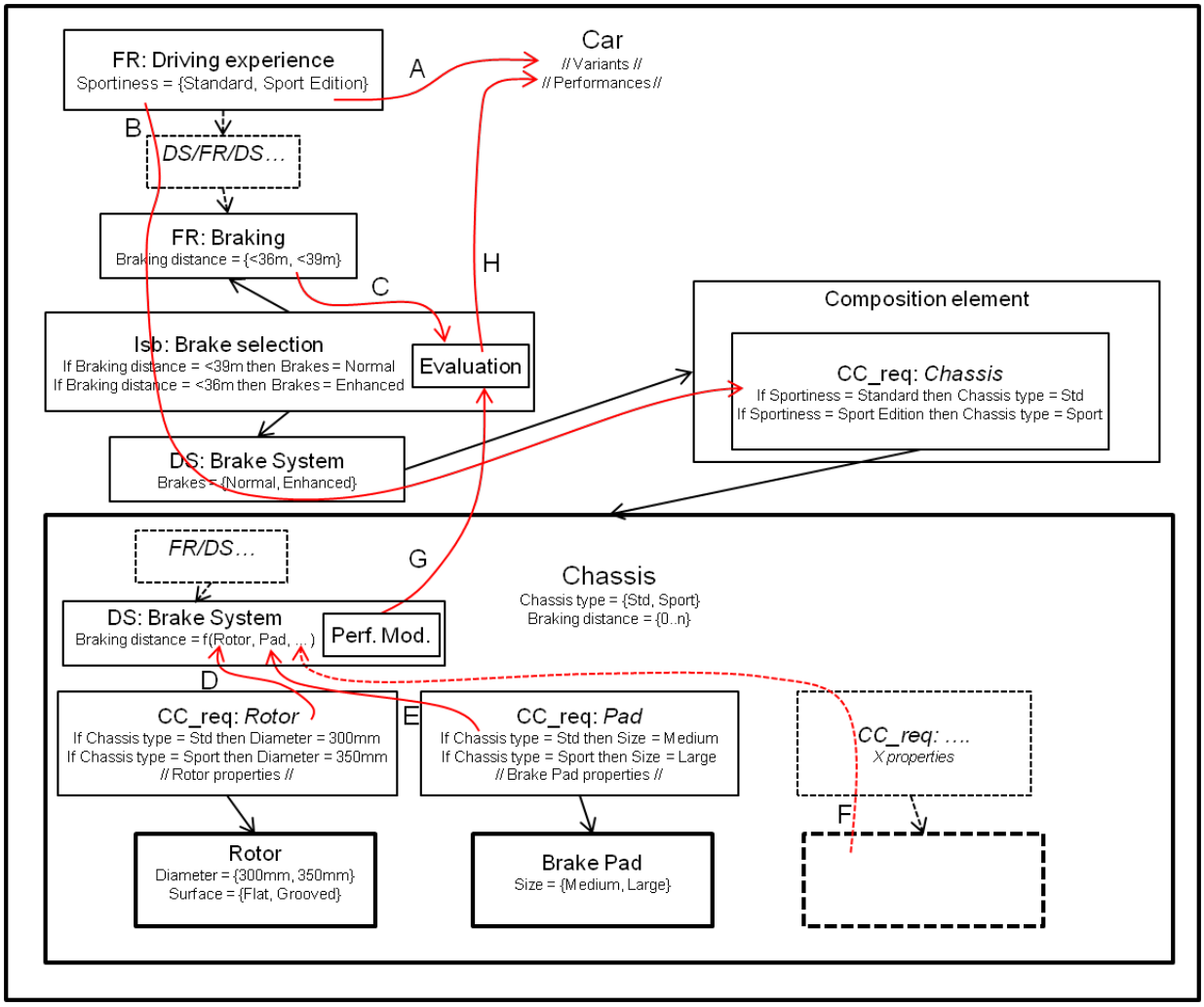

Figure 5. A subset of a car with the evaluation information flow (arrow D, E, F, G, C, H), the source for a variant parameter (arrow A), and a top-down design approach (arrow B).

Two elements in Figure 5 contribute in evaluating the designs performance relative the requirements, *evaluation* in *brake selection* and *performance model* (perf.mod.) in *brake system*. The information flows from the design solutions to the performance model (arrow D, E and F), to the *evaluation* (arrow C and G), and the result from the evaluation (arrow H) to *car*. Together these arrows form a bottom-up evaluation.

Attempting to establish this connection from design solutions to the car clearly showed that it is virtually impossible to form such a performance model by following a physically oriented product breakdown structure. As a consequence, the model must be capable of managing several overlapping modeling elements in order to provide for both a response on which parts to use for manufacturing and for calculation to evaluate achieved performance.

The *brake system's performance model* in Figure 5 is elaborated further in Figure 6, as a mean to model emergent properties. In order to model emergent properties, parameter mappings are utilized in design solution elements of abstract sub-systems, in this case *chassis*. This system representation provides the ability to host performance models that for example map the different angles of the chassis corners' (toe in, camber, caster etc.) contributions to performance measures on ride and handling. These and other similar design parameters contribution to the chassis behavior and performance requires an abstract dynamic chassis model (Figure 6) to be included.

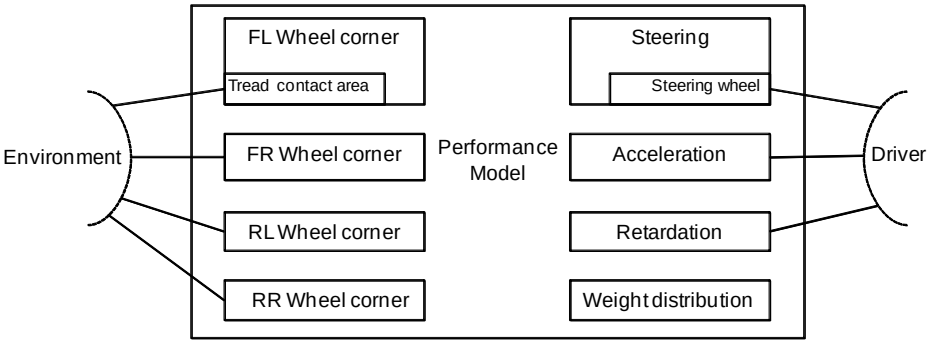

Figure 6, Illustration of some elements in a dynamic chassis model for Brake System.

Attempting to establish this connection from design solutions to the car clearly showed that it is virtually impossible to form such a performance model by following a physically oriented product breakdown structure. As a consequence, the model must be capable of managing several overlapping modeling elements in order to provide for both a response on which parts to use for manufacturing and for calculation to evaluate achieved performance.
The modeling elements outlined above constitute the prerequisites for a software agent to evaluate if necessary sub-systems are included. In other words, the modeling framework includes modeling mechanisms to formulate and establish requests and expectations as well as modeling mechanisms allowing the modeled design solutions to provide responses to those requests and expectations.

7 CASE STUDY RESULTS AND CONCLUSIONS

In performing the case study it is evident that deep insights of the design intent as well as the design itself are required in order to create appropriate descriptions and models. The focus for the case study was to model the vehicle chassis system and how it contributes to vehicle behavior through the different parts used. It was, however, an interesting experience for the researcher creating the model to realize that the task required knowledge way beyond his own, even though the researcher is an experienced senior engineer with long automotive experience. Even to briefly describe expected performances on car level and list a number of sub-systems requires deep and extensive knowledge. A respect for expert knowledge is a lesson to remember. In other words, our conclusion is that the product model preferably should be created and maintained as close to the source of knowledge as possible, i.e. by the designers themselves.
Further, concerning expert knowledge, modeling of performance will range from the complete vehicle down to individual parts. That range is seldom covered by a single person, but of a number of specialists that together covers the range from details to complete design. This puts even more emphasis on capabilities for supporting highly dynamic collaboration when using the described framework.
A strong benefit of creating a tool in parallel with the modeling research lies in the clarity that is required by the software tool in order to ensure proper capture and functionality for the modeling elements identified in the research. Furthermore, the requirement to actually capture the modeling entities using a software tool provides a higher level of clarity also in the approaches taken to the system modeling as such. In doing so, it becomes almost brutally clear where the modeling framework is supportive and where it has some weaknesses or missing elements or concepts.
The case study is still ongoing and results presented here are preliminary and based on the work done so far.

8 REFLECTIONS ON RESULTS AND CONCLUSIONS

In the early framing of the research scope presented a brief literature search was made in order to find a starting point and baseline. The outcome was rather disappointing and it was difficult to find a set of appropriate references upon which this research could be founded.

The completeness of a design is not absolute, but depends on the expectations of the design. The designs completeness can to a limited extent be evaluated against a list of expectations. There is, however, no known way to ensure whether this list in itself is complete or not. Obviously this is a recursive problem where it is only possible to state that completeness *depends*. The consequence is the need, presented above, to define a *criterion of completeness* in each and every case.

To illustrate and support the statement that the completeness of a design is not absolute, we can refer to Roozenburg and Eekels [6]. A design does not have functions (and thereby behavior and performance) on its own. Rather, a designs behavior (function) depends of the design itself (the four boxes in the upper left corner in Figure 7) in combination with its *mode and condition of use*. How a design is to be used is outside the control of the design itself. Actually the number of possible combinations of mode and condition of use is an infinite number. Based on this reasoning, the findings presented must be seen as initial steps in understanding and addressing issues on completeness, consistency, and compatibility.

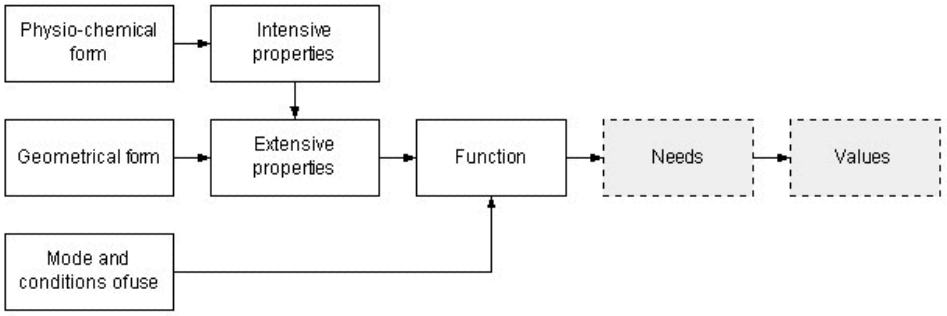

Figure 7: Product functioning [6].

A simplistic view and illustration of the problems dealt with above is presented in Figure 8. This generic feedback-loop shows the causality between expectations and performance responses from the design. It is our intention to apply the same approach of reasoning to the problems regarding consistency and compatibility.

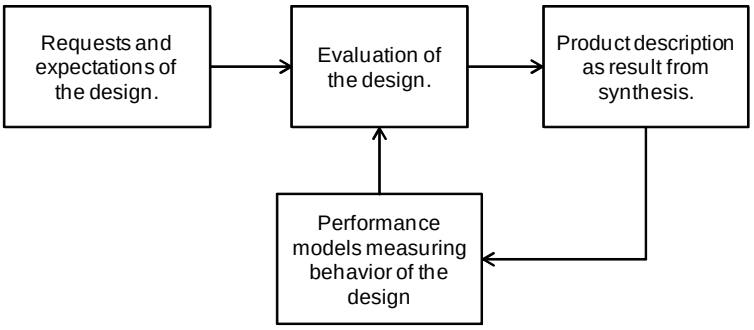

Figure 8: Conceptual illustration of the involved information blocks when evaluating completeness, consistency and compatibility.

The approach and findings presented has been found to provide a valuable starting point for further research on these topics.

ACKNOWLEDGEMENTS

This work was carried out at the Wingquist Laboratory VINN Excellence Centre within the Area of Advance – Production at Chalmers, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). The support is gratefully acknowledged.

REFERENCES

[1] Gedell, S., Michaelis, M., Johannesson, H., *Integrated Model for Co-Development of Products and Production Systems – A Systems Theory Approach*, 2010, (Accepted by Journal of Concurrent Engineering)
[2] Claesson, A., *A Configurable Component Framework Supporting Platform-Based Product Development*, 2006 (Doctoral Thesis, Division of Product and Production Development, Chalmers University of Technology, Göteborg, Sweden)
[3] Gedell, S., *Platform-Based Design - Design Rational Aspects within the Configurable Component Concept*, 2009 (Licentiate Thesis, Division of Product and Production Development, Chalmers University of Technology, Göteborg, Sweden)
[4] Andersson, F., *The Dynamics of Requirements and Product Concept Management*, 2003 (Doctoral Thesis, Division of Product and Production Development, Chalmers University of Technology, Göteborg, Sweden)
[5] Hitchins, D. K., *Advanced Systems – Thinking, Engineering, and Management*, 2003 (Norwood, MA: Artech House)
[6] Roozenburg, N. F. and Eekels, M. J., *Product Design: Fundamentals and Methods*, 1995 (Chichester: Wiley)
[7] Portoraro, F., *Automated Reasoning*, 2010 Winter Edition (The Stanford Encyclopedia of Philosophy, Zalta, E. N. (ed.)), http://plato.stanford.edu/archives/win2010/entries/reasoning-automated/
[8] Wos, L., Overbeek, R. Lusk, E., Boyle, J., *Automated reasoning: Introduction and Applications*, 1992 (McGraw Hill)
[9] Bonacina, M. P. and Martelli, A., *Automated reasoning*, 2006 (Intelligenza Artificiale, III(1-2):14-20, Marzo/Giugno)

Contact: Stellan Gedell
Chalmers University of Technology
Product and Production Development
412 96 Göteborg
Sweden
Tel: Int +46 (0)736 278525
Email: stellan.gedell@chalmers.se
URL:http://www.chalmers.se/ppd/SV/organisation/avdelningar/produktutveckling/personal/doktorander/gedell-stellan

Stellan Gedell is currently a PhD student at the Department of Product and Production Development at Chalmers University of Technology in Gothenburg, Sweden. He started his PhD studies at Chalmers University of Technology in 2008 focusing on integrated platform-based product development.