Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 44 (2016) 67 – 72

www.elsevier.com/locate/procedia

6th CIRP Conference on Assembly Technologies and Systems (CATS)

# Automated analysis of interdependencies between product platforms and assembly operations

Amir Hossein Ebrahimi*, Knut Åkesson*, Pierre E. C. Johansson*, Thomas Lezama*

[a]Department of Signals and Systems, Automation Research Group, Chalmers University of Technology, Gothenburg, Sweden
[b]Volvo Group Trucks Operations, Gothenburg, Sweden

* Corresponding author. Tel.: +0-000-000-0000; fax: +0-000-000-0000. *E-mail address:* amir.ebrahimi@chalmers.se

**Abstract**

Configurable products, like vehicles, face the challenge of handling all possible variants which are needed to answer the various customer needs. For these configurable products the support of all variants need to be addressed both by the design phase and the production phase. The product design phase and the production phase are linked together via operations. These operations model how each part of the bill-of-material is assembled to the final product. Operations also have inner relations among themselves, namely the precedence constraints, stating the order in which different parts can be assembled. Considering only the precedence constraints a product can generally be assembled in various different ways. It is through line balancing that the operations are assigned to different stations and/or assembly workers. The bill-of-material for each configurable product might be different with each variant, which will result in different set of operations. However, due to the precedence rules among the operations not all sets of operations might be possible to complete. The contribution in this paper is a automated method that can determine if all possible product variants can be successfully assembled while still satisfying precedence constraints between operations. The paper also includes an industrial example which further exemplifies the needed input for the method and the possible method outputs as a result of introducing a new variant to the product platform.

*Keywords:* Variability; Product platform; engineering Bill of Material; manufacturing Bill of Material; Assembly operation sequence

## 1. Introduction

In recent years there has been a dramatic increase in the number of product variants. This product variety can be seen across a wide range of products from simple products such as a light bulb through to extremely complex products like trucks and automobiles and airplanes. In some cases the number of product variants in the truck industry is around 500 variant families which lead to $10^{100}$ different valid product configurations. Some of the reasons behind this high increase in product variety include customers' demand for new product functions and features, different regional requirements, and large number of market segments with different need specifications [? ]. A product variant usually requires a number of manufacturing operations to be assembled. These manufacturing operations can include operations for machining, assembly, painting, finishing and packaging. These manufacturing operations usually have precedence constraint relationship which can be other manufacturing operations [? ]. Both in production management as well as in the supporting information technology, a central concept in product's design and manufacturing is each product's bill of material (BOM). Over the whole product lifecycle, a product will have more than one BOM describing its structure each from a different view point [? ]. Some of the most important BOMs include engineering BOM and manufacturing BOMs. The high variety in products will cause the creation and updating of the BOMs to be a time consuming and error prone job.

In this paper we propose, an automated method that can determine if all possible product variants can be successfully assembled while still satisfying precedence constraints between operations. It will be shown how to model the product platform in a way to capture all the inherent variability in the product platform. Then this model can be used as a satisfiability problem. The goal of this model will be defined in a way which means that an "UNSAT" result of the analysis means all operation sequences are executable. While a "SAT" result will give a counter example of an operation sequence which can't be executed.

The paper is organised as follows. In section II the terms used for describing problems with product variability and the necessary operations modelling actions that are necessary to assemble the products are defined, while an introductory example is used to illustrate the concepts. Section III explains the en-

doi:10.1016/j.procir.2015.12.132

gineering documents which are mainly effected by variability and the problems which are the result of variability in creating some of these documents. Section IV discusses the system implementation in two steps. Firstly the modelling formulas which are needed to model the product platform are given. Secondly the result of implementing the introductory example in this method is examined. In Section IV contains the conclusions.

## 2. Product and production system variability

For a product platform the different products may be realized through various *variants*. These variants can be either hardware or software variants. Variants which are similar due to their functionality and/or nature are grouped together to form *variant groups*. Each set of chosen variants from different variant groups is named a *configuration*. *Configuration rules* or *constraints* are specific rules for how configurations can be formed that result in a product that can be ordered by a customer. Any product instance which satisfies all the specified configuration rules is named a *valid product instance*. The configuration for each valid product generate the product's *bill-of-material*.

Each variant will need one or more manufacturing operations to be assembled. Hence for each valid product instance there will be a set of needed manufacturing operations for the assembly of the product. Considering that each product instance has a unique bill-of-material each product instance will also have its own set of manufacturing/assembly operations. However, not all sequences of operations are feasible, for example due to geometrical constraints. But among those that are feasible some are not desirable for other reasons, for example due to the need for extensive fixturing or the risk of part damage or because they result in sequences that causes fatigue or discomfort for the operator. In [**?** ] a method is presented for generating the desired precedence relations by considering the relations between parts, the liaisons, and by letting an engineer answers questions related to the desired sequences between operations realizing the assembly. In [**?** ] it is shown that the set of feasible and desired assembly sequences are not compactly represented by precedence diagrams, instead an AND/OR graph are proposed to implicitly define all possible and desired assembly sequences. The AND/OR assembly graph can be represented using the general precedence constraints between operations that are introduced in [**?** ]. In this work we will follow the operation concept as defined in [**?** ] but extend the approach to allow the analysis all precedence constraints for a set of possible product instances implicitly defined by product platform modelled as a feature diagram.

We will use an example to illustrate how to model the product variability and the corresponding operations. An example of product platform for trucks is introduced to illustrate the problem and the approach. The product platform consists of variant groups and variants and constraints relating these. These variant groups include both hardware and software groups. The hardware variant groups include *cab*, *frame*, and *accessories* with their relevant variants. While there is also a software variant group which includes software variants for handling the *Load indicator* and the *AI clock*. Table **??** shows the truck product platform where the corresponding feature diagram are represented using a table. Each variant group has a specific characteristic named *"Group Cardinality"* which shows the number of variants that should be chosen from that specific variant group for each valid truck. For example the *Cab* variant group has the group cardinality of *"choose exactly one"*, which means that for a valid truck one of the variants *Cab V1* or *Cab V2* should be picked. On the other hand, the variant group *accessories* has the group cardinality of *choose at least one*, meaning at least one variant from this variant group should be picked for each valid truck. Each combination of these variants from the different variant groups is one configuration, some of which will result in a valid truck instance. As an example consider the two possible configurations:

Table 1. Feature model expressed using a table of the truck product platform. VG denote a variant group, while V denote variants within a variant group.

| | Truck Platform | |
|---|---|---|
| ID: | Name: | Group Cardinality: |
| $VG_1$ | Frame | |
| $V_1$ | Frame rigid | Choose exactly one |
| $V_2$ | Frame tractor | Choose exactly one |
| ID: | Name: | Group Cardinality: |
| $VG_2$ | Cab | |
| $V_3$ | Cab VI | Choose exactly one |
| $V_4$ | Cab V2 | |
| ID: | Name: | Group Cardinality: |
| $VG_3$ | Accessories | |
| $V_5$ | Lower light bar | |
| $V_6$ | Head lamp protector | Choose at least one |
| $V_7$ | Wind deflector | |
| ID: | Name: | Group Cardinality: |
| $VG_4$ | Software | |
| $V_8$ | Load indicator | Choose exactly one |
| $V_9$ | AI clock | |

$$\text{Configuration 1 :}\{V_1, V_3, V_4, V_5, V_8\}$$
$$\text{Configuration 2 :}\{V_1, V_4, V_5, V_8\}$$

Configuration 1 is NOT a valid configuration according to the constraints in Table **??** as it has chosen both $V_3$ and $V_4$ while the *group cardinality* of variant group *cab* states that only one of the variants of this group should be present in any valid truck. But configuration 2 is a valid configuration as it satisfies all the stated configuration rules.

Table **??** shows some manufacturing oriented configuration rules which should be satisfied by each valid truck. As it can be seen each configuration rule is using at least one of the previously defined variant groups or variants. For example $C_1$ configuration rule states that each truck should have either the combination of *lower light bar* ($V_5$) and *load indicator* ($V_8$) or *head lamp protector* ($V_6$) and *AI clock* ($V_9$). Each configuration rule is a kind of a restriction which each valid truck should satisfy. If a configuration of a truck does not satisfy all of the defined constraints that configuration will not result in a valid truck.

Previously it was mentioned that Configuration 2 is a valid configuration according to the group cardinality constraints mentioned in table **??**. But if you consider the constraints in table **??**, then Configuration 2 is no longer a valid configuration as it contradicts $C_2$. This is because $C_2$ specifically states that

Table 2. Additional Constraints in the feature model of the product platform. Note that $R_1$ and $R_2$ relate variants in different variant groups to each other.

| ID | Constraint |
|---|---|
| $C_1$ | ($V_5$ AND $V_8$) OR ($V_6$ AND $V_9$) |
| $C_2$ | ($V_4$ AND $V_7$) OR ($V_3$ AND NOT $V_7$) |

if $V_4$ is picked then $V_7$ should also be picked, but this is not the case in Configuration 2 as $V_4$ is picked and $V_7$ is not picked. So now lets look at a new configuration.

$$\text{Configuration 3} : \{V_1, V_4, V_5, V_7, V_8\}$$

Configuration 3 does fulfil all the constraints mentioned in table **??** and **??**, so up to this point Configuration 3 is a VALID configuration.

We also need a model to describe the operations that will be used for a specific truck configuration. In this example we assume that each variant will have a corresponding operation, thus for each $V_i$ there will be an operation $O_i$. Operation $O_i$ will only be used if variant $V_i$ is selected in the configuration. In [**?** ] a three state model of an operation is used, three states are the *initial* ($O^i$), *executing* ($O^e$), and *finished* ($O^f$). Transition conditions are then used to restrict when the operation can transfer from the initial state to the executing state, and from the executing to the final state. In this work we will extend the model with a new state, called *unused* ($O^u$). This state will be used to model that the operation is not in use, this can be due to that some variant is not selected for a certain configuration and thus should not be assembled either. For the example, the precedence constraints expressed as preconditions can be seen in Table **??**. The precondition for operation $O_2$ is that $O_1$ should be in the state of $O_1^f$, this express that for $O_2$ to start, i.e. transfer from state $O_2^i$ to $O_2^e$, operation $O_1$ need to be finished, i.e. in state $O_1^f$. Now if you consider $O_3$, the precondition for this manufacturing operation is a bit more complex. Firstly, the precondition states that $O_1$ should be in state of $O_1^f$, i.e. $O_1$ needs to be completed before $O_3$ can start. Secondly, the precondition for $O_3$ states that $O_4$, $O_5$, and $O_6$ should be in their respective finished states before $O_3$ can start. In other words $V_5$, $V_6$, and $V_7$ need to be connected before you can drop the Cab V2.

The successful assembly of a product is when all the operations that are required for that particular product instance has been successfully completed, i.e. they have reached the *finished* state. Those operations that are not used, remain in the *unused* state. However, since the number of possible product instances are very high for complex products we have to avoid explicitly enumerating all possible product instances and the corresponding operations. In order to allow for one step analysis, for all possible configurations, if all operations can be successfully finished we will introduce additional constraints that will be used to set proper initial states on the operations depending on how the product is configured. Each variant $V_i$ is a boolean variable that can be true/false depending on if that variant is selected or not. Each state in operation $O_i$ will also be represented by a boolean variable that is defined accordingly. We will introduce constraints that if a variant is used the initial state of the corresponding is set to *initial*, if the variant is not used the operation

Table 4. Each operation state is a boolean variable that has to be initialised to a value that depend on the current configuration. Assuming operation $O_k$ should be used if and only if variant $V_k$ is selected in the configuration.

| Operation state constraints |
|---|
| $O_k^i \Leftrightarrow V_k$: (initial state) |
| $O_k^e \Leftrightarrow$ false: (executing state) |
| $O_k^f \Leftrightarrow$ false: (finished state) |
| $O_k^u \Leftrightarrow$ NOT $V_k$: (unused state) |

will start in state *unused*. Formally this can be expressed using the constraints in table **??**.

To summarize the models for the example we have constraints on the variants, expressed in Table **??** and **??**. However, we do also have precedence constraints between the operations that are found in Table **??**. The connection between the variants and the required operations is straightforward, i.e. each variant is supported by a corresponding operation. The initial state of the operations are defined as shown in Table **??**.

Consider now the previously mentioned Configuration 3 which was said to be a valid configuration. Considering this configuration the set of manufacturing operations which are needed for this configuration are as follows:

$$\text{Manufacturing operations needed for configuration 3} : \{O_1, O_3, O_4, O_6, O_7\} \quad (1)$$

Now according to the preconditions of $O_3$ this set of manufacturing operations can't be completed. This is because, as mentioned before, the precondition of $O_3$ states that $O_4$, $O_5$, and $O_6$ need to be finished before $O_3$ can start. But in the case of Configuration 3 there will be a problem with $O_5$. The problem is that because $V_5$ has not been picked, hence $O_5$ will be in the state of $O_5^u$ while it should be in the state of $O_5^f$ in order for $O_3$ to start. Hence it can be seen that although Configuration 3 was valid according to the constraints stated in tables **??** and **??** but it is not possible to successfully finish all the operations while obeying the precedence constraints in Table **??**. This is important knowledge for both a product designer and a product preparation engineer, because either the configuration should not be allowed or the operation precedence constraints should be changed. In an industrial setting we will have thousands of different variants, tens of thousands of configurations rules and thousands of operations. The industrial problem is further complicated because operations themselves have constraints on manufacturing resources that are necessary and also that operations have to be balanced between stations in order to achieve a desired cycle-time. To conclude the product designers and marketing people that work and update the configuration rules will have weak understanding of manufacturing operations and their requirements. On the other hand the product preparation engineers that update that develop the operation precedence constraints will not be in control of the product configuration constraints. However, as we saw in the example the two set of constraints affect each other. To the authors best knowledge it is unknown how to use computational techniques to efficiently analyse the consequences of for these situations.

Table 3. Assembly operations and the corresponding precedence constraints expressed using preconditions for the product platform in Table **??**.

| ID | Description | Precondition |
|---|---|---|
| $O_1$ | Drop frame rigid on automatic guided vehicle | No precondition |
| $O_2$ | Cab V1 drop | $O_1$ is in state of $O_1^f$ |
| $O_3$ | Day cab drop | $O_1$ is in state of $O_1^f$ AND $O_4$ is in state of $O_4^f$ AND $O_5$ is in state of $O_5^f$ AND $O_6$ is in state of $O_6^f$ |
| $O_4$ | Connect lower light bar | $O_2$ is in state of $O_2^f$ |
| $O_5$ | Connect head lamp protector | $O_2$ is in state of $O_2^f$ |
| $O_6$ | Connect wind deflectors | $O_3$ is in state of $O_3^f$ |
| $O_7$ | Install load indicator | $O_2$ is in state of $O_2^f$ |
| $O_8$ | Install AI clock | $O_3$ is in state of $O_3^f$ |

### 2.1. *Bounded Model Checking*

Bounded Model Checking (BMC) [**?** ], use SAT- and SMT-solvers for industrial property-based system verification. In general model checking, automatic techniques are used to verify if an implementation of a system satisfies system properties which are specified in logic. The answer to this verification can be "yes", indicating that the property holds, or "no", which will result in a counter-example for which the property does not hold. BMC is a way to implement model checking by taking advantage of SAT/SMT-solvers efficiency and to explore that state space iteratively for an increasing number of transitions. Both SAT- and SMT-solvers can be used to implement Bounded Model Checking algorithms. The main advantage of SMT-solvers is a more expressive modelling language allowing more general constraints, for example several SMT-solvers has the ability to reason about real-numbers. While SAT-solvers require the modelled to only be defined over boolean variables and the formula to be checked has to be in conjunctive normal form (CNF). Internally SMT-solvers typically use a SAT-solvers to solve relaxed problems. However, both SAT- and SMT-solvers has the ability to solve problems of large size and with high complexity. Safety properties are straightforward to encode using SAT/SMT-solvers, but different extensions to also handle liveness properties has been published, see for example [**?** ] that allow the use of properties expressed using linear temporal logic.

In order to model the system for property-based system verification in the BMC framework we will divide the model of the system into three parts, firstly the initial states and secondly the transitions between the states, and finally the property $P$

- $I$: is a formula that is true for the initial states ($S_0$) of the system.
- $T(s_i, s_j)$: represents the transitions of the system, which relation is true if and only if the system can make a transition between state $s_i$ and state $s_j$.
- $P$: represents a property to that we would like to hold for all states.

Using these variables and in order to check whether or not there exist a counter-example of length $n$ or smaller, the following formula can be used ($\wedge$ is a conjunction, $\vee$ is a disjunction, and negation is expressed by a bar over the expression):

$$I \wedge T(s_0, s_1) \wedge \ldots \wedge T(s_{n-1}, s_n) \wedge \overline{P} \tag{2}$$

Hence any satisfying assignment to the formula above represents a counter-example to the stated property.

The first effect of variability on equation **??** is seen in the formulation of the initial state $I(s_0)$, since the product is configurable the system might start in different initial states depending on how the truck is configured. A brute force algorithm would be to do the analysis in [**?** ] for all possible initial states. For complex products like trucks this is not tractable because the number of possible configure is very large. Thus, we will include the configuration problem including the constraints into the problem formulation.

In the initial state of a system with variability two important parts need to be defined. Firstly, all information on the different variation groups and variants need to be defined. Secondly, the configuration rules which state specific rules among variant groups and variants of the model also need to be defined in the initial set. All valid product instances need to satisfy all the stated configuration rules.

The second part of equation **??** which is affected by variability is the definition of the operation set. As previously mentioned each component of a product will need a series of defined operations for the assembly of the component. Hence as a result of different variations of product components, namely the defined variants, there will be different sets of operations which can be performed according to the variant which is chosen for a product. So instead of a fixed sequence of operations there will be different variations of operations which can be performed.

## 3. Effect of variability on engineering documents

The effect of variability can be seen in the structure of engineering documents which are needed to document product families. Some of the most important examples of these engineering documents include "engineering Bill of Material" (eBOM), "manufacturing Bill of Material" (mBOM), and "manufacturing Bill of Process" (mBOP). In this paper eBOM and mBOM refer to documents which explain the structure of the overall product family, hence differ from eBOM and mBOM which are based on individual single product. To make this distinction between these two types of eBOM and mBOM we will refer to our version of the eBOM and mBOM (containing the structure of the overall product family) as "master eBOM" and "master mBOM". The master eBOM and master mBOM both contain the structure of variant groups and variants which are contained in a product family but from different viewpoints. The master eBOM contains the variant and variant groups from a functional view point, while in master mBOM contains the variant group and variants from a manufacturing viewpoint. As an example consider the truck cabin, in the master eBOM this variant might have with it some smaller variants which are assembled together to build the cabin like doors and chairs. But when you look at the cabin variant in the master mBOM as this variant is in the assembly line as one single variant hence in the master mBOM will also be one single variant of Cabin. Valid product configurations are generated from the master mBOM and the list of configuration rules. Normally in industrial cases the number of valid configurations is extremely high (in some cases around $10^{100}$), making it impossible to create these valid configurations manually. Hence with the automatic analysis of the master mBOM and corresponding configurations rules the valid configurations can be created. These valid configurations are a part of the resulting mBOP. In the mBOP for each valid configuration the list of chosen variant can be seen. Another important part of the mBOP is the series of manufacturing operations which are needed to assemble each of the valid configurations.

## 4. System implementation

In this section we will show how to model and analyze a sample product family. The modeling of a product family has previously been shown in [**?** ]. Formula 7 to 12 are updated formulas from the previous work. We will not go into the modeling details of the system and refer the reader to [**?** ], but we will give a summary of formulas which are used to model the product family.

System model:

$$I(s_0) \wedge T(s_0, s_1) \wedge \ldots \wedge T(s_{n-1}, s_n) \wedge \overline{P} \tag{3}$$

Initial system state:

$$A \equiv \bigwedge_{i \in \{1 \ldots |VG|\}} A_i. \tag{4}$$

$$B \equiv \bigwedge_{i \in \{1 \ldots |R|\}} R_i. \tag{5}$$

$$C \equiv \bigwedge_{\forall j \in \{1 \ldots |V|\}} (O^i_{j,0} \Leftrightarrow V_j) \wedge \overline{O^e_{j,0}} \wedge \overline{O^f_{j,0}} \wedge (O^u_{j,0} \Leftrightarrow \overline{V_j}). \tag{6}$$

System transitions:

$$O^i_{k,j} \wedge Pre_{k,j} \Rightarrow O^e_{k,j+1} \tag{7}$$

$$\overline{O^i_{k,j} \wedge Pre_{k,j}} \Rightarrow (O^i_{k,j} \Leftrightarrow O^i_{k,j+1}) \tag{8}$$

$$O^i_{k,j} \oplus O^e_{k,j} \oplus O^f_{k,j} \oplus O^u_{k,j} = 1 \forall j \tag{9}$$

$$O^e_i \Rightarrow O^f_{i+1} \tag{10}$$

$$O^u_{k,j} \Leftrightarrow O^u_{k,j+1} \tag{11}$$

$$O^f_{k,j} \Rightarrow O^f_{k,j+1} \tag{12}$$

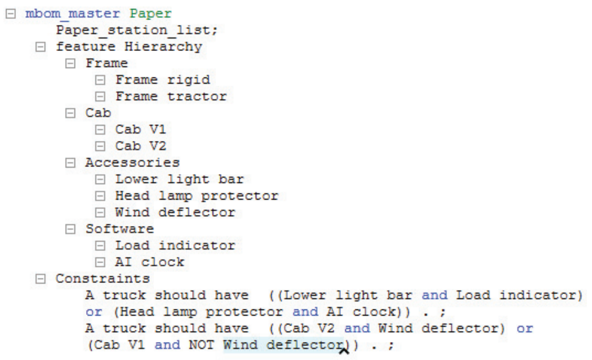
$$\bigwedge_{\forall k,k' \text{ s.t. } k<k'} \overline{(O^i_{k,j} \wedge \overline{O^i_{k,j+1}}) \wedge (O^i_{k',j} \wedge \overline{O^i_{k',j+1}})}. \tag{13}$$

System targets:

$$P^j_1 \equiv \bigwedge_{\forall k \in \{1 \ldots |V|\}} (O^f_{k,j} \vee O^u_{k,j}) \tag{14}$$

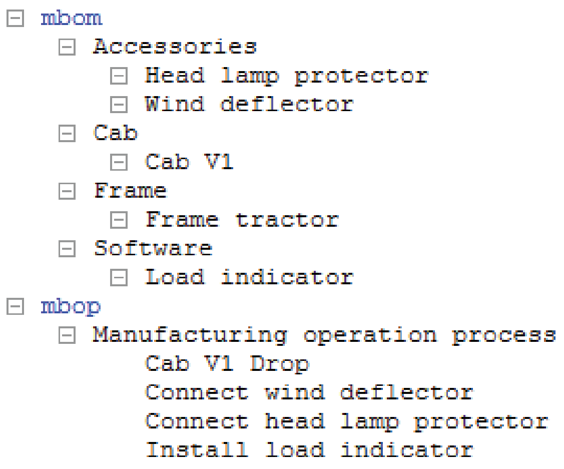
$$P^j_2 \equiv \bigvee_{\forall k \in \{1 \ldots |V|\}} ((O^i_{k,j} \wedge Pre_{k,j}) \vee O^e_{k,j}) \tag{15}$$

$$P^j \equiv \overline{P^j_1} \Rightarrow P^j_2. \tag{16}$$



```
⊟ mbom_master Paper
      Paper_station_list;
   ⊟ feature Hierarchy
      ⊟ Frame
         ⊟ Frame rigid
         ⊟ Frame tractor
      ⊟ Cab
         ⊟ Cab V1
         ⊟ Cab V2
      ⊟ Accessories
         ⊟ Lower light bar
         ⊟ Head lamp protector
         ⊟ Wind deflector
      ⊟ Software
         ⊟ Load indicator
         ⊟ AI clock
   ⊟ Constraints
         A truck should have  ((Lower light bar and Load indicator)
         or (Head lamp protector and AI clock)) . ;
         A truck should have  ((Cab V2 and Wind deflector) or
         (Cab V1 and NOT Wind deflector)) . ;
```

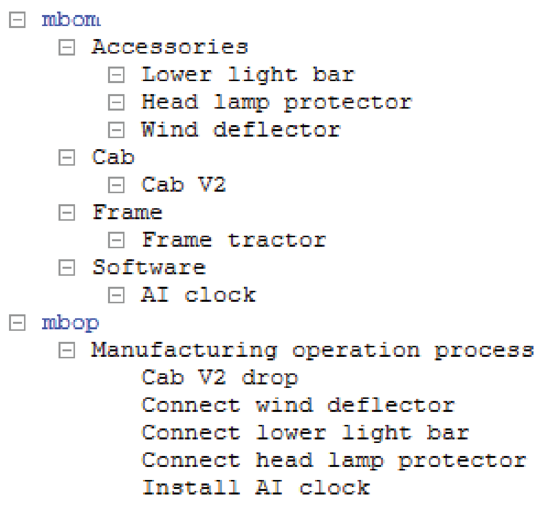
Fig. 1. master mBOM of the introductory example



```
⊟ mbom
   ⊟ Accessories
      ⊟ Head lamp protector
      ⊟ Wind deflector
   ⊟ Cab
      ⊟ Cab V1
   ⊟ Frame
      ⊟ Frame tractor
   ⊟ Software
      ⊟ Load indicator
⊟ mbop
   ⊟ Manufacturing operation process
         Cab V1 Drop
         Connect wind deflector
         Connect head lamp protector
         Install load indicator
```

Fig. 2. Automatically created mBOP

Before the master eBOM and master mBOM can be created or updated, some initial information are needed by the system. These initial information includes, list of manufacturing operations (including potential preconditions among manufacturing operations), and variant groups and variants (including the list of manufacturing operations which are needed for the assembly of the corresponding variant). It is using this initial information that the master mBOM can be analyzed and the mBOP can be



```
⊟ mbom
    ⊟ Accessories
        ⊟ Lower light bar
        ⊟ Head lamp protector
        ⊟ Wind deflector
    ⊟ Cab
        ⊟ Cab V2
    ⊟ Frame
        ⊟ Frame tractor
    ⊟ Software
        ⊟ AI clock
⊟ mbop
    ⊟ Manufacturing operation process
          Cab V2 drop
          Connect wind deflector
          Connect lower light bar
          Connect head lamp protector
          Install AI clock
```

Fig. 3. Automatically created mBOP

created. By analyzing the master mBOM the system can ensure that the manufacturing operations which are needed for each valid product can be executed. The meaning of the analysis result depends on the goal which is defined for this analysis. The primary goal which is defined in **??** is looking for operation sequences which can't be completed. Hence if the result of the analysis is ”UNSAT” it means that all operation sequences are safe and can be executed. On the other hand a ”SAT” result will give a counter example which is an operation sequence which can't be completely executed. The goal of the model can be defined in a way to reach all valid configurations which fulfil the given configuration rules.

Examples of master mBOM and mBOP showing the previously defined introductory example are shown in Figure **??**, **??**, and **??**. As it can be seen from Figure **??** the structure of the master mBOM is defined as it was specified in the introductory example. The master mBOM also includes a list of configuration rules using simplified normal sentences which can be better understood by the user. From the given master mBOM and the initial information which is specified, including definition of manufacturing operations and assembly stations, all possible valid configurations are calculated. Two of the calculated valid configurations are shown in Figures **??** and **??**. The valid configurations include two parts, firstly the variants which have been selected for this configuration are shown in the context of the mBOM for this specific product instance. Secondly, for this product instance the list of manufacturing operations which are needed for the assembly of this product is given in the mBOP part. This list is dependent on what manufacturing operations are needed for the assembly of each chosen variant. The main advantage of this method is the automatic creation of valid configurations which will enable the engineers to adjust the master mBOM information and see the immediate effect this change has on the list on valid configurations.

The presented model is currently implemented using a Z3 SMT solver. Initial results of this implementation will be presented in future work.

## 5. Summary

In this paper it is shown how variability effects product design and manufacturing process. In the design of the product engineering documents, i.e. master eBOM, master mBOM and mBOP, are effected by variability. A method is proposed to model the variability with in a product platform which can use initial information and the master eBOM and master mBOM to model the product platform. Detailed formulas for this modelling are given in previous work. Using this model and a SAT/SMT solver it is shown how to analyse the model to examine if all operation sequences are executable. Unsatisfiable analysis result will show operation sequences which can not fulfil the given constraints in the model. It is also shown how the model can also be used to create the valid configurations of the given product platform.

## Acknowledgements

This work was partially supported by the Know4Car project, grant agreement number 284602, funded by the EC Seventh Framework Programme theme FoF-ICT-2011.7.4.

## References

[1] Ebrahimi, Amir Hossein, et al. ”Formal analysis of product variability and the effects on assembly operations”. Emerging Technologies and Factory Automation (ETFA), 2015 IEEE 20th Conference on. IEEE, 2015.

[2] ElMaraghy, H., et al. ”Product variety management.” CIRP Annals-Manufacturing Technology 62.2 (2013): 629-652.

[3] Navaei, Javad, and Hoda ElMaraghy. ”Grouping part/product variants based on networked operations sequence.” Journal of Manufacturing Systems 38 (2016): 63-76.

[4] Xu, H. C., X. F. Xu, and Ting He. ”Research on transformation engineering bom into manufacturing bom based on bop.” Applied Mechanics and Materials. Vol. 10. 2008.

[5] T. De Fazio and D. E. Whitney, ”Simplified generation of all mechanical assembly sequences,” Robotics and Automation, IEEE Journal of, vol. 3, no. 6, pp. 640658, December 1987.

[6] L. S. Homem de Mello and A. C. Sanderson, ”AND/OR graph representation of assembly plans,” IEEE Transactions on Robotics and Automation, vol. 6, no. 2, pp. 188199, 1990.

[7] B. Lennartson, K. Bengtsson, C. Yuan, K. Andersson, M. Fabian, P. Falkman, and K. A kesson,”Sequence planning for integrated product, process and automation design,” Automation Science and Engineering, IEEE Transactions on, vol. 7, no. 4, pp. 791802, 2010.

[8] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, ”Bounded model checking,” Advances in Computers, vol. 58, no. 99, pp. 117148, 2003.

[9] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan,”Linear encodings of bounded ltl model checking,” Logical Methods in Computer Science, vol. 2, no. 5, pp. 164, 2006.

[10] A. Voronov and K. Akesson, ”Verification of process operations using model checking,” in 2009 IEEE International Conference on Automation Science and Engineering, 2009, pp. 415420.