Security Verification of Third Party Design Files in Manufacturing

Alexander Giehl
Fraunhofer AISEC
Parkring 4
Garching b. München, Germany
0049 89 3229986 189
alexander.giehl@aisec.fraunhofer.de

Norbert Wiedermann
Fraunhofer AISEC
Parkring 4
Garching b. München, Germany
0049 89 3229986 141
norbert.wiedermann@aisec.fraunhofer.de

ABSTRACT
Customer-individual production in manufacturing is a current trend related to the Industrie 4.0 paradigm. Creation of design files by the customers is becoming more frequent. These design files are typically generated outside the company boundaries and then transferred to the organization where they are eventually processed and scheduled for production. From a security perspective, this introduces new attack vectors targeting producing companies. Design files with malicious configuration parameters can threaten the availability of the manufacturing plant resulting in financial risks and can even cause harm to humans. Human verification of design files is error-prone why an automated solution is required. A graph-theoretic modeling framework for machine tools capable of verifying the security of product designs is proposed. This framework is used to model an exemplary production process implemented in a wood processing plant based on the experiences of a real-world case study. Simulation of the modeled scenario shows the feasibility of the framework. Apart from security verification, the approach can be adopted to decide if a product design can be manufactured with a given set of machine tools.

CCS Concepts
• Security and privacy→Domain-specific security and privacy architectures.

Keywords
Security; Manufacturing; Modeling; Graph-Based Analysis; Production Planning; Industrie 4.0.

1. INTRODUCTION
Industrie 4.0 is the ongoing evolution of the manufacturing landscape in Germany that is also observed in other countries. It is characterized by the increasing integration of cyber-physical systems into the manufacturing process. One desired effect of this development is an increase in flexibility to address market demands. [1] identifies, among others, the possibility for strong incorporation of customer-individual features into the production cycle. Before production of these customer-individual goods can begin, the design for this product needs to be fixed. This is required for production planning to generate an efficient production schedule in terms of machine utilization. Offering tools to the customer for creation of the design files is a cost-effective way to collect customer requirements. The tools can be made accessible, for example, via an online service or in local subsidiaries. After the design is generated, it must be transferred to the production planning system to schedule and start the production.

Figure 1 provides an overview of the communication based on a functional model of different automation layers within a plant [2]. The model itself is based on the ISA 95 standard [3] with the corresponding numbers of the individual layers, or levels, given on the left side of Figure 1. The third party design file arrives at the top layer, "Business Planning & Logistics". This layer is, among others, responsible for the overall production planning within the plant. From here, the design file is transferred to the subjacent layer "Manufacturing Operations & Control", where detailed production planning takes place. The order for manufacturing of individual products is dispatched from here; manufacturing of the product takes place in the lowest layer. Within the Industrie 4.0 paradigm, a stronger interconnection of the different layers takes place and the boundaries between them are starting to disappear.

Information technology (IT) security concerns itself with the protection of computer systems by achieving protection goals [4]. Major protection goals are confidentiality, data integrity, and availability, typically referred to as CIA. Varying definitions are available in literature, in the context of this work the protection goals are interpreted as follows. Confidentiality means that information is disclosed only to authorized entities. Data Integrity describes the accuracy and consistency of stored or transmitted data and ensures that no manipulation or unauthorized alteration of the data occurs. Availability is a concept also related to the design of dependable systems [5]. It encompasses the availability for authorized and correct service. In manufacturing, the availability of the manufacturing environment is often seen as the most relevant protection goal by OT operators [6]. Accepting unverified third party design files leads to a security risk threatening the availability of the entire production cycle as seen in Figure 1. Trusted third party connections, e.g., to a vendor, are a possible entry point for external attacks to industrial controls systems (ICSs) [7]. This entry point is becoming more relevant due to new Industrie 4.0 business models that favor stronger inter-

ICCAE 2018, February 24–26, 2018, Brisbane, Australia

ACM ISBN 978-1-4503-6410-2/18/02…$15.00

DOI: https://doi.org/10.1145/3192975.3192984

connection of ICSs, also beyond company boundaries. Manipulation of design file parameters can lead to compromised product quality [8]. Furthermore, a manipulated design file can lead to an overall loss of machine availability and, therefore, production capacity [9]. Performing operations outside the processing capability of a machine can lead to machine downtime due to unplanned maintenance, for example, replacement of broken machine tools or clean up procedures.

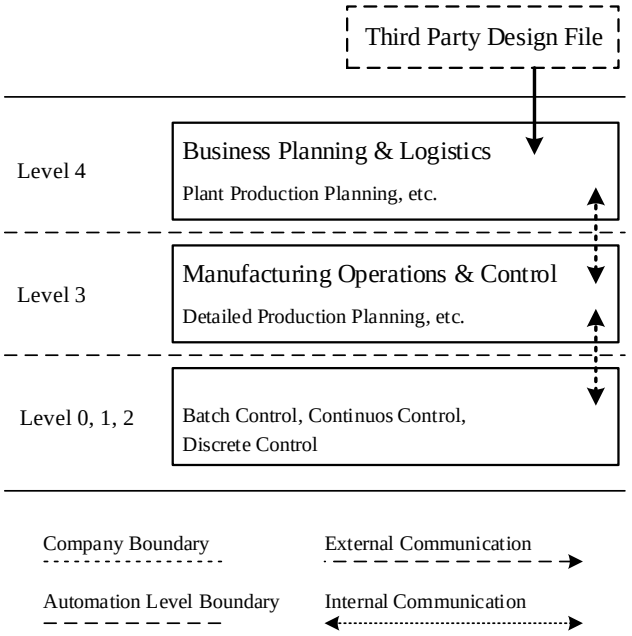

Figure 1. Layers of automation based on [2, 3].

Currently implemented production cycles are not well equipped to identify manipulated design files. Third party or other design files are checked by a process that incorporates a human-in-the-loop. Human subject experimentation with engineering students on identification of malicious design files show that humans are prone to error [9, 10]. While some of the participants in the conducted studies were able to identify malicious design files, intentional manipulation of the files, i.e., a cyber-attack, was not believed to be the cause. The authors of the studies suggest, among others, educational awareness and enhancements in the quality control process for mitigation. Awareness of engineering personal concerning security is important and is further aided with adequate tool support. Since quality control takes place after manufacturing of the potentially defective product, which can lead also to machine downtime, a verification step prior to production is beneficial towards the availability of the whole production environment.

An automated software solution to aid engineers in identifying malicious design files is proposed. The framework needs to be adoptable to encompass the wide variety of machine tools and their configurations present in real-world manufacturing plants. For this, the framework builds on well-established formal methods. To further aid in minimizing machine downtime, a simulation-based approach is proposed. The feasibility of manufacturing a product can be verified before production starts without involving real world machines. This helps in production planning. The focus of the framework is on security-related design file verification but can further be extended to verify if a product can be manufactured given a set of machines and machine configurations.

2. RELATED WORK

The exchange of product geometry as well as the verification of the specified geometry is subject of standardization [11]. Here, the deviation of a manufactured product from its ideal form according to product geometry is assessed. No automated approach is proposed, a human-in-the-loop is considered necessary in the verification step. Further, the standards offer no support in deciding if a certain product geometry or specification can be harmful to a manufacturing environment.

Verification of third party design files for manufacturing is a known problem in Additive Manufacturing (AM). AM, or 3D Printing, is the production of parts by successively adding material to the part until the target design is met. In contrast to this, Subtractive Manufacturing (SM) is the production of parts by successively removing material from the part until the target design specification is reached. For AM, a Stereo Lithography (STL) file is generated from the initial Computer-Aided Design (CAD) file containing the part specification. This STL file is then used in the further steps of AM. Existing work on identifying malicious design files for AM focus on STL file verification. [12] describes the possibility and effects of attacks on AM systems. The focus is on attacks where a void is introduced in the design of a STL file. A void is a hollow region of a part completely enclosed by material. Introduction of voids in the part's design can result in reduced structural strength and, therefore, in reduced product life of the product it is used to assemble. The authors recommend, among others, improved software checks on STL design files as protective measures. [13] examines the manipulation of part production in AM by assessing its intrinsic properties. Their work focuses on individual parts rather than the assembly line and does not take damage caused to the manufacturing environment into account.

SM processes do not include the usage of STL files. Rather, the CAD file is converted into machine code as specified, for example, in ISO 6983 [14]. Verification of design files for SM in the context of security is a field of research that has not received attention in the past and is addressed within this work.

3. CONCEPTUAL FRAMEWORK

In this section, a conceptual description of the proposed framework for security verification of design files is presented. The focus is on use cases for SM; however, the approach is flexible and can be extended to encompass AM scenarios as well.

The main design goal for the framework is adaptability. This is necessary to encompass technological progress in modern manufacturing environments. Further, the respective domain where manufacturing takes place needs to be considered. For example, an assembly line of a furniture manufacturing plant uses a different set of tooling machines as food packaging lines. Furthermore, within the same domain, tooling machines from a variety of vendors can be employed. These tooling machines vary in their range of operation and the included machine tools. In addition, a range of different products can be produced on an assembly line given a fixed set of tooling machines.

To achieve adaptability for this wide variety of possible applications and tooling machine configurations, a graph-theoretic approach is proposed [15]. Graphs are a tool of discrete mathematics used to model objects and their relations with each other. A graph G is defined as an ordered pair

$G = (V, E)$ where V is a set of vertices and E is a set of edges. The graph G here is the model of a manufacturing environment. A vertice $v \in V$ is a functional unit found in a manufacturing environment. This can include machine tools executing a subtractive process, e.g., cutting of material, or a non-subtractive process, e.g., pick-and-place or material refinement. A functional unit $v \in V$ is not a representation of a singular machine purchased from a tooling machine vendor. Rather, different functional units are found in modern tooling machines. V compromises all functional units of a manufacturing environment, thus, represents the overall capability of the modeled manufacturing environment G. An edge $e \in E$ is a directed connection between two vertices v_k and v_l denoted as $e = (v_k, v_l)$; the set of all edges v_n represents the path p of a workpiece W through a manufacturing environment. A workpiece can pass through the same set of vertices and edges several times to undergo successive processing. Thus, G is a cyclic, directed graph.

Figure 2 shows an example for the model of a simple manufacturing environment. The ingoing workpiece W_{in} arrives at vertice v_1 where it is processed. Then, it is transferred via edge e_1 to v_2 where further processing ensures. Similarly, it is transferred via e_2 to v_3. Here, the manufacturing process can be repeated iteratively via e_3. Once the target workpiece description of a design file is matched, the outgoing workpiece W_{out} leaves the manufacturing environment.

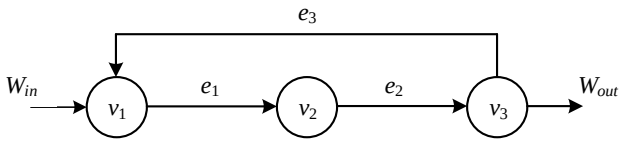

Figure 2. Representation of a directed graph.

The conceptual descriptions in this article are illustrated by a running example that is developed over the course of this article, implemented, and simulated in Section 5.

Example 1. *v_1 can be a machine tool designated to changing the orientation of the workpiece, v_2 can perform a subtractive operation on the workpiece, and v_3 executes a pick and place task putting the workpiece either back to v_1 for turning or towards the outgoing path. In this example, W_{in} can be a rectangular workpiece on which a subtractive action is performed on each side of W_{in}. After the subtractive actions are performed, a workpiece W_{out} fitting the target description of a design file, in this example, a rectangular part with certain dimensional properties, is created.*

The design file contains a description of the workpiece W_{out} to be produced. In general, W_{out} can be produced with a variation of raw materials, a different set of tooling machines, and via several paths in the manufacturing environment. A path p is a route through a manufacturing environment G that satisfies the workpiece description D, i.e., generates $W_{out} = D$ by traversal of G. Each of the connected vertices v that constitute p perform an action on the workpiece that is demanded by D. Considering the example above, this action can be a subtractive task where material is cut from the workpiece. This action is denoted as a transformation $t_v(W) = W'$ where W is changed to W' by the corresponding $v \in p$. The set of all actions that can be performed on W_{in} while traversing p is denoted T_p. A node v is capable of performing an action; each individual action is limited by bounds as illustrated by Example 2.

Example 2. *An action describes a transformation a workpiece can experience by traversing through a node, e.g., cutting, drilling, gluing, placing. Bounds describe the operational limitations of an action, e.g., the drilling tool can be placed by the machine's actuators within a certain area of the tooling machine for placing drill points.*

Given these definitions, it is possible to derive statements about the capabilities of the manufacturing environment modeled by G. First, a simple statement considering only the actions is provided:

> ***Statement 1 (Weak Statement).*** *The existence of a set of vertices $v_p \in V$ capable of performing all actions demanded by D and W_{out} is said to satisfy D and W_{out}.*

Note that the existence of v_p does not guarantee if the target design can be manufactured within the existing manufacturing environment, i.e., its manufacturability is undecided. This is because also the bounds of a tooling machine must be considered. For example, the existence of a drill alone does not guarantee that the required drill head is present or the drill point can be placed on the specified location of the workpiece. In order to decide if the target design can be manufactured, the following statement, which implicitly considers the bounds, is required:

> ***Statement 2 (Strong Statement).*** *The existence of a path p in G with $T_p(W_{in}) = W_{out}$ shows the manufacturability of D on a manufacturing environment G.*

If Statement 2 is true, the target design can be manufactured within the present manufacturing layout. By extension, this also includes that no incidents, security-related or otherwise, occur during manufacturing as the occurrence of an incident means a violation of certain bounds. Statement 2 is algorithmically computed in Section 5. For this, the following assumptions need to be met:

> ***Assumption 1.*** *The manufacturing environment G contains exactly one starting node v_s. W_{in} is always first processed by v_s.*

This is necessary for computation of p. Note that Assumption 1 can always be ensured by the addition of a virtual starting node.

> ***Assumption 2.*** *The processing or traversal of W_{in} in G is fixed. The order for traversing is implicitly given by D.*

It is possible that more than one path p exists in G. Therefore, the computation can be enhanced by optimization methods.

> ***Assumption 3.*** *The traversal order of Assumption 2 is reasonable.*

A manufacturing environment is set up to avoid unreasonable operations. For example, applying a refinement to the workpiece and then performing a subtractive action removing the refined part is unreasonable. Thus, production planning tends to reduce these operations, as they are harmful to the overall business.

In Section 4, modeling of the workpieces W_{in} and W_{out} is further described in more detail with a comprehensive example. Notes on the computation of a path through the graph, i.e., deciding the manufacturability, are provided in Section 5. In that section, Assumptions 1 - 3 are used to provide an algorithmic solution.

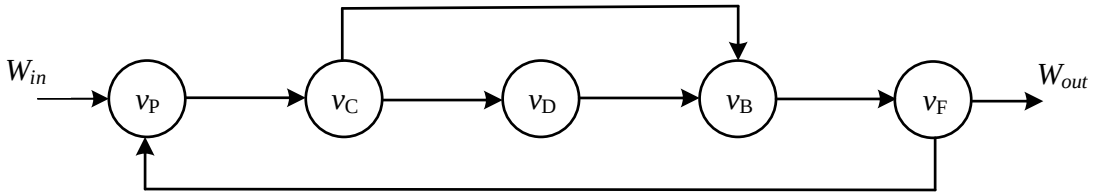

Figure 3. Model of simple manufacturing environment.

4. CASE STUDY: FURNITURE MANUFACTURING

In this section, a case study for the framework developed in Section 3 is presented. The framework is applied to the domain of furniture manufacturing. For this, a basic model of a manufacturing environment is assumed.

Typical shop-floor setups in furniture manufacturing are organized to provide an environment in which specific actions are performed. Among others, these actions are cutting, drilling, and border finishing [16]. Cutting involves the dimensional reduction of a wooden plate. In this case study, the wooden plate is the ingoing workpiece W_{in}. Drilling is the process of placing a drill point on the surface of the workpiece. Border finishing is comprised of several individual actions. In this article, a simplified process involving "border banding" and "border finishing" is assumed. Border banding is the process of attaching a border to a wooden plate; border finishing applies a finishing to an attached border. In addition, the placing operation is added. It changes the orientation of a workpiece according to a pre-defined frame of reference. To summarize, the set of actions consists of cutting, drilling, border banding, border finishing, and placing (see Example 3).

Modeling of W_{in} can be highly domain-specific why a reference model of W_{in} for furniture manufacturing is provided at this point. However, it can be argued that some of the aspects described here can be readily transferred to domains outside furniture manufacturing. Tooling machines employed in furniture manufacturing provide, inter alia, the actions described above. For dimensional reduction, i.e., cutting, the dimensions of the workpiece need to be included in the model. Therefore, a dimension vector $\boldsymbol{w}$ for W_{in} is added with $\boldsymbol{w} = (x, y, z)$ where x, y, and z are the three-dimensional coordinates in relation to a pre-defined coordinate system. The material of W_{in} is encoded by a descriptive entity M. Material descriptions need to be unique and documented in a well-defined list of descriptors relevant to the context of the scenario. Drilling is described by an unordered list L_D containing the coordinates of already placed drill points. Additional drill points specified by D are assumed to be known to the production planning system as it has information on D. Note that a reference frame for describing the orientation of the workpiece needs to be defined a priori. Borders are described by another unordered list L_B containing their properties, i.e., the type of the border itself and the applied border finishing. Thus, W is a 4-tupel $W = \{\boldsymbol{w}, M, L_D, L_B\}$.

Example 3*. Figure 3 shows a model of a simplified shop floor setup; for better readability, the edge labels are omitted. W_{in} is first processed by the node providing placing functionality (denoted as v_P), i.e., performs a change in orientation. From here, W' passes through the node designated to perform a cutting operation (v_C). Then, it is transferred to border banding (v_B) either directly or via an intermediate step through the node performing drilling operations (v_D). After border banding, border finishing is applied (v_F). Several loops can be performed through the manufacturing environment if further operations need to be applied to W'. In this case, it is transferred back to v_P where the described process begins anew. When no further loops are required to meet the demands of D, the workflow is completed and W_{out} is produced.*

In order to provide a description of a node's capabilities, additional information needs to be encoded in the model. For one, the input dimensions a node is capable of processing is required. This is necessary to decide if a workpiece can be processed by a node. Next, the bounds (see Section 3) of a node are encoded. For cutting and drilling, the bounds describe the range within cuts can be performed or drill points can be placed, respectively; for border processing, the bounds are the types of border placements and border finishings that can be applied. The data model of this information for computational processing is described in Section 5.

Given the model of the manufacturing environment above, security analysis can be conducted. It is possible to infer statements from the model that have a malicious effect on the availability of the production.

Example 4*. Consider a scenario where a drill is specified to be placed on a glass surface contained in Win. The glass surface can be part of a small door often found in kitchen or living room furniture. Thus, if the processing node assumes due to a manipulated Win that M is a wooden surface and places a drill point on it, the result will most likely be the destruction of Win. Further, it will be necessary to provide unscheduled maintenance to the processing node resulting in a loss of production capability.*

Example 4 is based on and motivated by real-world incidents experienced by project partners in IUNO (see Section 8). Verification was conducted by a human-in-the-loop. As described in Section 1, this is an error prone process. In Section 5, an automated solution to this problem is given and implemented with standard technologies.

5. EXPERIMENTAL EVALUATION

In this section, the experimental evaluation of the modeling framework described in the previous sections is provided. First, a reference implementation of the framework is provided in Section 5.1. Next, this implementation is used in Section 5.2 to simulate the case study in furniture manufacturing with varying parameters.

5.1 Implementation

In this section, a reference software architecture of the implementation is discussed.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <!DOCTYPE board SYSTEM "board.dtd">
3  <board>
     <size>
5      <xdim>1990</xdim>
       <ydim>1500</ydim>
7      <zdim>30</zdim>
     </size>
9    <material type="wood"/>
     <operations>
11   <holes tool="drill">
       <hole id="ID_drill_01" diameter="5" pos_x="100" pos_y="100" pos_z="10"/>
13     <hole id="ID_drill_02" diameter="5" pos_x="150" pos_y="100" pos_z="10"/>
       <hole id="ID_drill_03" diameter="5" pos_x="100" pos_y="150" pos_z="10"/>
15     <hole id="ID_drill_04" diameter="5" pos_x="150" pos_y="150" pos_z="10"/>
     </holes>
17   <edges>
       <edge id="ID_edge_01" type="a" finish="normal"/>
19   </edges>
     </operations>
21 </board>
```

Figure 4. Example for target design specification X_T.

As descriptive file format standard Extended Markup Language (XML) is employed [17]. This choice is due to the acceptance and widespread use of XML in modern OT environments [18]. In the following, X denotes a descriptive XML specification. In order to compute the manufacturability of a design file, certain information needs to be known a priori: descriptions of the tooling machines' capabilities X_M, the workpiece dimensions X_W derived from W_{in}, and the target furniture design X_T derived from W_{out}. X_M is ideally provided by the vendor of the tooling machines while X_W and X_T are best specified by the OT operator of the manufacturing environment.

Example 5*. Figure 4 shows the XML encoding of X_T given the manufacturing environment of Figure 3. The dimension vector w is seen in Lines 4-8, the material description in Line 9, the list of drill points L_D in Lines 11-16, and the border properties description L_B in Lines 17-19.*

Note that X_W uses the same structure and elements while X_M needs to provide matching descriptors, e.g., drill, for the capabilities of the tooling machine. The structure of the design files is validated against a predefined Document Type Definition (DTD) (see Line 2 in Figure 4), which specifies grammatical rules for XML files.

The individual operations performed by a tooling machine $v_n \in V$ are implemented via Extensible Stylesheet Language Transformations (XSLTs) [19]. They are used to transform XML files into other XML files. Furthermore, XSLT allows modification of an XML file by adding or removing individual XML nodes and by performing arithmetic operations. Those properties are used to update X_W while it traverses p.

Example 6. *Consider an X_W with dimensions $\boldsymbol{w}$ = (2000, 1500, 30) and an X_T as seen in Figure 4. When regarding only the dimensional vector, the value of x = 2000 needs to be reduced by 10 units in order to meet the target description and, therefore, manufacture the final product design. For this, a cutting action needs to be performed by one of the tooling machines $v_n \in V$.*

It is necessary to compute this difference between the target design X_T and the current state of the design X_W in order to proceed with design file verification. This difference is denoted as $\Delta \geq 0$. When $\Delta = 0$, the manufacturability of a target design on a manufacturing environment G is given. The transformation t_v (see Section 3) is expressed by a corresponding configuration c_v within the context of computing XML files. A $\Delta > 0$ is reduced by applying a set of configurations C where a $c_n \in C$ describes an individual configuration of a tooling machine v_n.

Example 7. *The corresponding configuration c is a dimensional vector with $\boldsymbol{c}$ = (10, 0, 0). In case of other operations (e.g., drilling), the computed tooling machine configuration is again represented as an unordered list. Next, a tooling machine capable of performing a cut within the specified bounds (see Section 3) needs to be found. Otherwise, the desired operation cannot be performed and X_T is not manufacturable on G as no path p through G is possible.*

Note that the knowledge on how to appropriately reduce Δ needs to be encoded within the framework. It is known to the framework that a cut reducing the dimensional vector $\boldsymbol{w}$ is beneficial towards meeting X_D as illustrated by Example 7. This knowledge is encoded internally in the framework. For future work, an externalized representation of knowledge, e.g., via an ontology, can provide benefits to scalability and extension of the framework towards further security studies [20].

The set of vertices that constitute a path p is computed by Algorithm 1. If a valid p with $\Delta = 0$ is found, the algorithm returns true, otherwise false. For better readability, implementation specific details, e.g., the use of temporary variables in Line 10 or data structure operations for p in Line 15, are omitted. The algorithm starts with confirming the trivial Weak Statement (see Statement 1) in Line 2. If it is confirmed positively, the manufacturability, i.e., the Strong Statement (see Statement 2), is verified. For this, the input graph G is mapped by function $f_{map}(v)$ (see Line 5). It returns all nodes reachable from the input node v.

This is possible since Assumption 2 holds. As input parameter, v_s from Assumption 1 is chosen.

Computation of Statement 2 starts with Line 6. Here, candidates of vertices v_i for inclusion in p are stored in V_i, which is initially filled by function f_{map}. The computation will continue as long as f_{map} returns further nodes v_i for processing (see Line 18). They are individually processed and evaluated towards their inclusion in p starting at Line 8. Prior to this, optimization of V_i can be performed by an optimization function f_{opt} due to Assumption 3, which is reasonable in more complex manufacturing layouts. In Lines 9-11, the configuration in the form of an XSLT X_{SLT} for a node v_i is computed by function f_{con}. Then, it is applied to X_W by function f_{app} resulting in an updated description X_C. X_C and X_T are then used to compute Δ. If $\Delta = 0$, the target design can be manufactured and the computation can be stopped (see Lines 12-13). Otherwise, a deviation function f_Δ is used in order to search for a better path p. p is then updated accordingly (see Lines 14-15). In Line 18, the next node is selected for expansion to continue the algorithmic computation. Finally, the Boolean return value based on Δ is determined in Lines 20-24.

```
Require: X_T, X_W, X_M, G
     {Initialize}
1:   Δ = ∞
     {Check for Weak Statement}
2:   if Weak Statement is not satisfied then
3:      return false
4:   end if
     {Check for Strong Statement}
5:   V_i = f_map (v_S)
6:   while V_i ≠ ∅ do
7:      V_i = f_opt (V_i)
8:      for all v_i ∈ V_i do
9:         X_SLT = f_con (v_i , X_M)
10:        X_C = f_app (v_i , X_SLT , X_W)
11:        Δ = f_Δ (X_C, X_T)
12:        if Δ = 0 then
13:           exit while loop
14:        else if Δ < f_Δ (p) then
15:           update p
16:        end if
17:     end for
18:     V_i = f_map (V, X_T)
19:  end while
20:  if Δ = 0 then
21:     return true
22:  else
23:     return false
24:  end if
```

Algorithm 1. Computation of *p*.

The algorithm presented in this section is found to perform well in the case study on furniture manufacturing (see Section 4). When used in other use cases, however, deadlocks can occur and the interpretation of the results can vary [21]. As the framework in this article is adopted within IUNO (see Section 8), future additions to the algorithm are following.

5.2 Simulation

In this section, the implementation and execution of different simulation runs as well as the evaluation of the simulation results are discussed.

Algorithm 1 is implemented in GoLang [22] along with other necessary software components such as XML decoding. The choice for GoLang is made because of the language's cross-compilation feature, which benefits the usage within heterogeneous manufacturing environments. Furthermore, required libraries and other dependencies are included in the binary file generated by GoLang's compiler allowing for easy distribution on existing systems. It is, however, also possible to implement a model of a manufacturing environment by using other programming languages that provide similar capabilities.

The framework can be called by a production planning system as a simple function call due to its Boolean return value. This call can be executed in Level 4 or Level 3 of the automation model provided by Figure 1. Note that is also possible to return the computed configurations *C*. Provided a dedicated compiler exists, the derived configurations can be translated in machine code for parametrization of the tooling machines in Levels 1-2.

As noted previously, Algorithm 1 requires several data as input. The running example developed over the course of the previous sections is used here for specifying the simulation use cases. Thus, the simulated manufacturing layout G^{Sim} is shown in Figure 3.

Example 8. *G^{Sim} is used to manufacture wooden cupboards with a glass door. The cupboards are of cubic dimension with five wooden sides and one glass side. For later assembly, drill points need to be placed on the glass surface. The glass doors are delivered by a supplier, as G^{Sim} is not capable of performing this action on its own. Performing drilling actions on glass requires specialized tooling that is not available in G^{Sim}.*

Following Example 8, two types of W_{in} are processed by G^{Sim}:

- W_{in}^{wood} = { (2000, 1500, 30), wood, ∅, ∅ } and
- W_{in}^{glass} = { (2000, 1500, 30), glass, ∅, ∅ }.

From them, the XML representations for W_{in}^{wood} and W_{in}^{glass}, namely X_W^{wood} and X_W^{glass} respectively, are derived in order to perform analysis on. X_T^{wood} is shown in Figure 4; X_T^{glass} is similar to Figure 4 but does not include Lines 11-16. The drill points specified in these lines are already placed by the supplier prior to the start of production. All XML files follow the syntactical structure shown in Figure 4 and are validated against a DTD. Furthermore, X_M^{Sim} specifies the actions required by the target designs and the corresponding bounds.

The attacker scenario based on a real world use case from Example 4 is described in the following example:

Example 9. *Two kitchens are separately designed by a customer (the third party) and issued electronically to the plant. In one case, the design file has been deliberately altered to include drill points on the glass surface.*

For evaluation, two simulation runs of the furniture manufacturing scenario are specified. They are denoted as Sim_1 and Sim_2 respectively. Sim_1 is the reference run with no manipulated design files present during processing of the workpieces. In Sim_2, however, X_T^{glass} has been manipulated as described by Example 9. For the sake of this example, $X_T^{glass} = X_T^{wood}$. Sim_2 is based on an observation by a plant operator, where no automated verification step is implemented. As indicated in Example 4, this results in the destruction of W_{in}^{glass}.

The simulation runs also confirm this outcome as the frameworks reports true for Sim_1 and false for Sim_2. The return values are derived from an implicitly encoded information base within the

framework as noted in Section 5.1. Here, the list of valid actions are stated for each material. As drilling is not specified for the material glass, the framework reports its value to the production planning system. Malicious design files, thus, can be detected and reacted to prior the physical production process is started. A human operator can be notified in a subsequent step in order to perform a detailed investigation on the outcome of Sim_2. As discussed in Section 1, a human-in-the-loop is prone to error. However, employing an automated solution simulating the outcome of manufacturing a target design supports the human operator. He can spend more time on the verification of identified potential critical designs and has the advantage of knowing the overall outcome of the verification. Therefore, he can focus on identifying the cause of the incident.

6. DISCUSSION

This work showed how to model and simulate manufacturing environments for security verification of third party design files. Concepts adopted from graph theory are used in order to develop a flexible framework for modeling purposes. Further, an extensive example based on modeling the production process in furniture manufacturing is provided. The presented approach to modeling is flexible and can be adapted to meet the requirements of other industries, as basic concepts are transferable. The framework definition is intended to provide a description that is independent of a specific architecture as the operational technology tends to be heterogeneous between companies and organizations [23]. Next, guidelines for implementation of the presented solution are provided. An algorithm for computation of the defined properties is presented. For implementation, technologies that can easily integrate into existing OT environments are used [18]. As markup language, XML is employed as it is standardized and widely used in manufacturing. For implementation, GoLang is used in order to provide a software component that can be easily distributed on heterogeneous environments. Two scenarios are specified and simulated. The results of the simulation show the feasibility of the framework.

7. CONCLUSION

This article provides a building block for the security of modern and future manufacturing environments within the context of Industrie 4.0. The discussed approach is not only relevant for security verification but, furthermore, relevant in production planning as it can be used to decide upon the manufacturability of a target design that is provided by a third party. In the context of large-scale customer-individual production, the automated solution presented here can offer a benefit within production planning. The authors work closely with tooling machine vendors and operators in order to provide a relevant contribution that can be adapted for usage in daily operations. The solution presented here is based on a real-life case study in a manufacturing plant.

8. PROJECT FUNDING

The presented work is part of the German national security reference project IUNO (http://www.iuno-projekt.de). The project is funded by the Federal Ministry of Education and Research (BMBF) and aims to provide building blocks for security in the emerging field of Industrie 4.0.

9. REFERENCES

[1] Kagermann, H., Helbig, J., Hellinger, A., and Wahlster, W. 2013. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry*. Technical report. Industrie 4.0 Working Group, Frankfurt/Main, Germany.

[2] Hollender, M. 2010. *Collaborative process automation systems.* Technical Report. International Society of Automation (ISA), Research Triangle Park, NC.

[3] ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod). 2010. *Enterprise-Control System Integration – Part 1: Models and Terminology*. International Society of Automation Standard (ISA), Research Triangle Park, NC.

[4] Eckert, C. 2014. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. De Gruyter, Oldenbourg, Germany.

[5] Avizienis, A., Laprie, J., Randell, B., and Landwehr, C. 2004. Basic concepts and taxonomy of dependable and secure computing, *IEEE transactions on dependable and secure computing*, 1, 11–33. DOI: https://doi.org/10.1109/TDSC.2004.2

[6] Sadeghi, A., Wachsmann, C. and Waidner, M. 2015. Security and privacy challenges in industrial internet of things. *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, 1–6. DOI: https://doi.org/10.1145/2744769.2747942

[7] Byres, E. and Lowe, J. 2004. The myths and facts behind cyber security risks for industrial control systems. *Proceedings of the VDE Kongres*s, 116, 213–218.

[8] Belikovetsky, S., Yampolskiy, M., Toh, J. and Elovici, Y. 2016. dr0wned - cyber-physical attack with additive manufacturing. arXiv:1609.00133. Retrieved from https://arxiv.org/abs/1609.00133

[9] Wells, L., Camelio, A., Williams, C. and White, J. 2014. Cyber-physical security challenges in manufacturing systems. *Manufacturing Letters*, 2, 74–77. DOI: https://doi.org/10.1016/j.mfglet.2014.01.005

[10] Turner, H., White, J., Camelio, J., Williams, C., Amos, B. and Parker, R. 2015. Bad parts: Are our manufacturing systems at risk of silent cyberattacks? *IEEE Security and Privacy*, 13, 40–47. DOI: https://doi.org/10.1109/MSP.2015.60

[11] Srinivasan, V. 2008. Standardizing the specification, verification, and exchange of product geometry: Research, status and trends. *Computer-Aided Design*, 40, 738–749. DOI: https://doi.org/10.1016/j.cad.2007.06.006

[12] Sturm, L., Williams, C., Camelio, J., White, J. and Parker, R. 2017. Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the STL file with human subjects. *Journal of Manufacturing Systems*, 44, 154–164. DOI: https://doi.org/10.1016/j.jmsy.2017.05.007

[13] Vincent, H., Wells, L., Tarazaga, P. and Camelio, J. 2015. Trojan detection and side-channel analyses for cyber-security in cyber-physical manufacturing systems. *Procedia Manufacturing*, 1, 77–85. DOI: https://doi.org/10.1016/j.promfg.2015.09.065

[14] ISO 6983-1:2009. 2009. *Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1: Data format for positioning, line motion and contouring control systems*. International Organization for Standardization Standard (ISO), Geneva, Switzerland.

[15] Bondy, J. et al. 1976. *Graph theory with applications*. North Holland, New York, NY.

[16] Suzić, N., Stevanov, B., Ćosić, I., Anišić, Z. and Sremčev, N. 2012. Customizing products through application of group technology: A case study of furniture manufacturing. *Strojniški vestnik-Journal of Mechanical*

Engineering 58, 724–731. DOI: http://dx.doi.org/10.5545/sv-jme.2012.708

[17] Bray, T., Paoli, J., Sperberg-McQueen, C., Mailer, Y. and Yergeau, F. 2008. *Extensible Markup Language (XML) 1.0 (5th edition)*. World Wide Web Consortium (W3C) Recommendation. Retrieved from https://www.w3.org/TR/xml/

[18] Sauter, T. 2005. Integration aspects in automation - a technology survey. *10th IEEE Conference Emerging Technologies and Factory Automation*, 2, 9-pp. DOI: https://doi.org/10.1109/ETFA.2005.1612688

[19] Kay, M. et al. 2017. *XSL transformations (XSLT) version 3.0*. World Wide Web Consortium (W3C) Recommendation. Retrieved from https://www.w3.org/TR/2017/REC-xslt-30-20170608/

[20] Ekelhart, A., Kiesling, E., Grill, B., Strauss, C. and Stummer, C. 2015. Integrating attacker behavior in IT security analysis: a discrete-event simulation approach. *Information Technology and Management*, 16**,** 221–233. DOI: https://doi.org/10.1007/s10799-015-0232-6

[21] Even, S. 2011. *Graph algorithms.* University Press, Cambridge, MA.

[22] Newmarch, J. 2017. *Network Programming with Go: Essential Skills for Using and Securing Networks* (1st. ed.). Apress, New York, NY.

[23] Thomesse, J. 2005. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93, 1073–1101. DOI: https://doi.org/10.1109/JPROC.2005.849724