The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-019-03823-x

ORIGINAL ARTICLE

A Function-Behaviour-Structure design methodology for adaptive production systems

David Sanderson[1] · Jack C. Chaplin[2] · Svetan Ratchev[2]

Received: 30 July 2018 / Accepted: 16 April 2019
© The Author(s) 2019

Abstract
Adaptive production systems are a key trend in modern advanced manufacturing. This stems from the requirement for the system to respond to disruption, either in the form of product changes or changes to other operational parameters. The design and reconfiguration of these systems are therefore a unique challenge for the community. One approach to systems design is based on functional and behavioural modelling, drawn from the field of design theory. Existing approaches suffer from lack of focus on the adaptive properties of the system. While traditional production systems design focusses on the physical system structure and associated processes, new approaches based on functional and behavioural models are particularly suited to addressing the challenges of disruptive production environments resulting from Industry 4.0 and similar trends. We therefore present a Function-Behaviour-Structure (FBS) methodology for Evolvable Assembly Systems (EAS), a class of self-adaptive reconfigurable production systems, comprising an ontology model and design process. The ontology model provides definitions for Function, Structure, and Behaviour of an adaptive production system. This model is used as the input to a functional modelling design process for EAS-like systems, where the design process must be integrated into the system control behaviour. The framework is illustrated with an example taken from a real EAS instantiation using industrial hardware.

Keywords Functional modelling · Function-Behaviour-Structure · Design methods · Adaptive systems · Manufacturing systems · Evolvable Assembly Systems

1 Introduction

1.1 Motivation

Mass customisation, shorter product lifecycles, smaller production batches, and higher product variability all lead to the requirement for manufacturing systems that are rapidly reconfigurable and self-adaptive in response to disruption [3]. This requires methods to aid in the design and modelling of such systems, both by humans and for automated self-design or self-reconfiguration by an agent control layer. In this work, we take an approach based on considering how system structure and behaviour relate to the intended system functions, drawing on the Function-Behaviour-Structure (FBS) formalisations by Gero, Rosenman, Umeda, and others [14, 16, 27, 34, 40, 48, 49]. The main driver is to be able to identify when a reconfiguration needs to occur and what that reconfiguration should address, and then "design the change" that would accomplish it.

1.2 Literature review—design frameworks

In general, the design process links requirements to components that are either to be freshly designed or selected from a set of existing components. For production systems, these requirements stem from the product and also a range of other quality requirements to optimise or evaluate against.

The range of approaches for design include the axiomatic approach [44], which uses domain models and rules to

✉ David Sanderson
David.Sanderson@nottingham.ac.uk

Jack C. Chaplin
Jack.Chaplin@nottingham.ac.uk

Svetan Ratchev
Svetan.Ratchev@nottingham.ac.uk

[1] Centre for Aerospace Manufacturing, University of Nottingham, Easter Park, Nottingham, NG7 2PX, UK

[2] Institute for Advanced Manufacturing, University of Nottingham, Jubilee Campus, Nottingham, NG7 2GX, UK

Published online: 15 May 2019

generate new designs, and the algorithmic approach [30] which goes step-by-step, decomposing the problem, solving the sub-problems, then re-synthesising the sub-solutions to form the final design solution. This allows the problem to be addressed at the correct level of granularity, appropriate to current practices.

The FBS approach of Gero, Rosenman, Kannengiesser, and others [14, 17, 34, 35] forms part of the research area in functional modelling [13, 51]. Functional modelling is concerned with how to represent knowledge about function, and can be applied to systems modelling and design. The FBS approach takes a holistic viewpoint looking at both the socio-cultural and techno-physical aspects of a system, clearly separating the subjective function of a system from its objective structure.

This function-based description of an object allows for a more effective decomposition according to the various uses an object must fulfil at any given time. When this is applied to modular and reconfigurable systems, the decomposition can be directed according to the available modules [23, 24]. Such reconfiguration must take into account the availability and connectivity of existing modules, and this problem is further complicated when a product-process-system view is taken [47]. In such a case, the reconfiguration from currently available capabilities to required capabilities may take place at any level. This requires a clear definition of task decomposition and assignment to modules or processes.

There are a number of integrated approaches for deriving the required process for a product representation. Some examples use graphs or petri nets [43] or a rule-based approach [31], but these often either operate at an abstract level in terms of the system or are extremely knowledge-intensive on the human side [52]. This is due to being focussed towards assisting system designers in system configuration, simulation, and evaluation.

There are some approaches for the computer-aided design and modelling of assembly systems [6, 20, 26, 36, 52] but they generally do not address system qualities such as self-adaptation, self-reconfiguration, and self-healing, more generally called the self-x qualities [28, 29]. Those approaches that do address self-x qualities in an automated manner [10, 25, 46, 50]—for example through agent-based systems—often focus primarily on one aspect, such as the software and configuration side, rather than a complete product-process-system approach that includes a formal description of the system from its function to the physical hardware.

Other works seek to address the design of specific sorts of production system in more specific detail, for example "Assembly 4.0" type systems [9], or large-scale aerostructure assembly [21], or are primarily concerned by the safety and ergonomics considerations at the heart of manual assembly system design [7, 8, 37].

Despite this existing body of work, we can therefore identify three main issues relating to design frameworks:

1. The majority of existing production system design frameworks only address single-product or single-family production, rather than a system that may have to drastically refocus on another product or process entirely, as well as being under a changing set of non-product requirements.
2. Existing FBS approaches do not address adaptive production systems; they focus on creating design artefacts such as the product itself, or a static system. There is a critical requirement for a method applicable to self-x production systems that change over time.
3. Many production system design frameworks are either extremely abstract or knowledge-intensive, making their automated application to self-x systems problematic. The majority of automation-focussed frameworks for self-x systems focus primarily on only one aspect of the system, missing a formal description of the whole system.

With this in mind, we propose a solution that is capable of addressing a wide range of dynamic requirements, takes advantage of the inherent self-x properties of the system components as a "top-level attribute" of the framework, and allows for a formal description of the whole system in order to enable its application to automated—in our case agent-based—control systems.

2 Overall approach

The design process for an adaptive production system starts with the products that are to be manufactured. These are defined in "*recipe files*" that are the starting point for the system's functional requirements. Other requirements are also considered: the requirements for adaptation, and the requirements relating to achieving and maintaining system quality. The other main input to the design process is a set of what Gero would call "design prototypes" [14]—the set of possible or available manufacturing resources for use in this system specified as "*blueprint files*". From these two main inputs, the design process generates the set of behaviours required of the recipes and generates a solution from the available set of resources that could express those behaviours. An overview of the approach is given in Fig. 1. Section 3 defines the design process and the ontology supporting it in more detail. Specific details of how recipe and blueprint files formalise product characteristics and system structure respectively

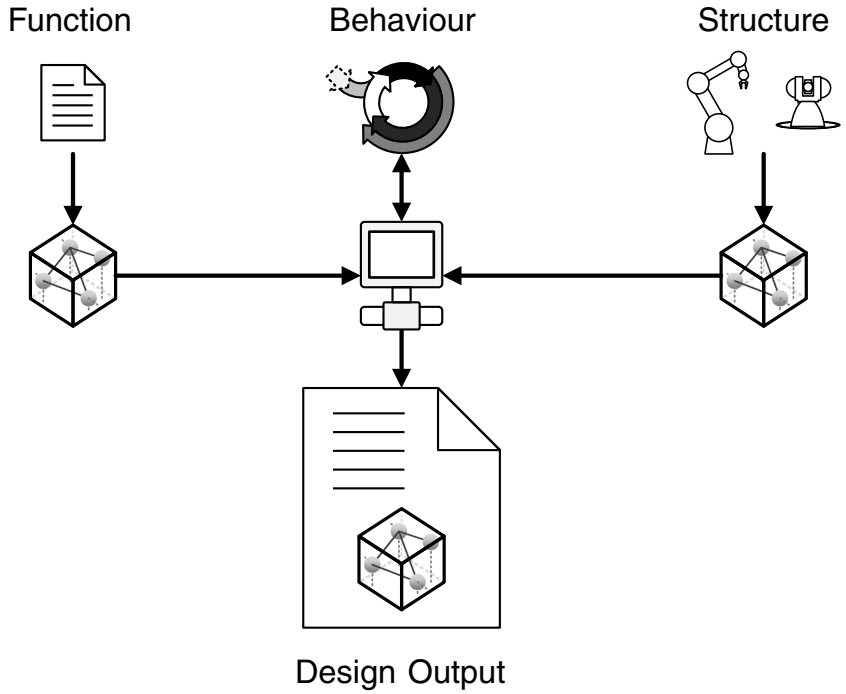


Fig. 1 Behavioural design process: overall approach

is described in Section 4, along with an overview of the implementation of behavioural evaluation that forms the core of the approach. Finally, Section 5 describes a real prototype implementation example.

3 General behaviour model

3.1 Views

A key concept in Umeda et al.'s work on FBS diagrams is their use of different "views" or "aspects" [48, 49] that are described as "a collection of all relevant entities, attributes, relations, and physical phenomena of the current interest".

In essence, this concept formalises the fact that it is not possible or feasible to represent everything about an entity (in our case a product, system, or sub-system) in a single encapsulated description. When you are designing a takeaway coffee cup for example, you are interested in how well it holds liquid—it is unlikely that you are concerned about the permittivity of the material to light. With this in mind, we use the concept of views to allow only the relevant aspects of function, behaviour, or structure to be considered.[1]

Gero et al. also refer to this concept of views (although not by the same terminology), by acknowledging that different actors will be concerned with different aspects of the functions, behaviours, or structure of an object. In [15], they give the example of a mobile phone: the user is interested in the structure as given by the screen, the keys, the size, and so on, resulting in a behavioural analysis on the use of the phone; an electrical engineer may find the electronic circuits of the phone to be relevant structural factors, which can then be used to account for behaviours relating to the operating system, the ring tone, or the ability of the phone to connect to various different kinds of wireless networks.

3.2 Function

Following the consensus in the literature, we maintain the definition of function as a subjective abstraction of purpose [12, 16, 49]. We can consider an Evolvable Assembly System to have a number of simultaneous functions.

As a production system, the main function is to produce a set of products. There are performance requirements associated with that function, both in terms of KPIs and in terms of complying with relevant regulatory frameworks.

In an EAS, there are two additional requirements on the real production system designed to achieve that function. First, the system should be flexible and reconfigurable to accommodate the production of a large and changing set of products. Second, the system should be able to adapt itself in response to disruptions such as failure and unexpected environmental changes.

[1] It is up to the system designer to ensure that all the relevant views are considered; in some cases, it is desirable for a cup to be transparent, in which case the permittivity to light is a highly relevant consideration.

The function of assembling a product is expressed in terms of the product to be assembled: informally "to assemble a product with this set of features and properties". Therefore, the function of an assembly system is related to the structure and behaviour of the products it is to assemble. However, the product designers should not be concerned with specifying the processes by which the assembly is carried out. As an example, the product designer would specify a hole should *exist* at a given location with a given accuracy (among other parameters), but would not be expected to specify the decision-making that occurs in the system in order to choose whether it is drilled, bored, or laser cut, or to ensure the selection of the correct drill bit, or the precision of the motion required to reach that point. These latter issues are related to the behaviour of the system; issues that the product designer should not be burdened with, and in many cases, may not be in a position to know.

In terms of requiring our systems to *adapt to changes*, this could occur at a high level, for example a change in product to be assembled, the failure or addition of a manufacturing capability, and so on. Alternatively, it could occur at a low level, for example compensating for the uncertainty, variability, deformation, or deflection of a part that is being worked on. This means that the adaptation requirements are defined in a multi-dimensional configuration space, according to the available and configurable variables in the system.

The remaining aspects of the system function encapsulate more traditional system performance indicators; for example, the system should achieve a certain throughput, not exceed a given power consumption, or must perform actions in a certain manner in order to comply with regulations.

With these three aspects in mind, we can define our concept of Function as follows:

Definition 1 (Function) $F = \langle F_{Pr}, F_A, F_Q \rangle$ where
$F_{Pr} = f(R_{Pr})$: functions relating to product requirements R_{Pr},
$F_A = f(R_A)$: functions relating to adaptation requirements R_A, and
$F_Q = f(R_Q)$: functions relating to achieving and maintaining system performance and quality requirements R_Q.

Further examining R_{Pr} allows us to express the requirements stemming from the product's behaviour (R_{Pr_B}) and structure (R_{Pr_S}), such that $R_{Pr} = \langle R_{Pr_S}, R_{Pr_B} \rangle$. The behaviour of the product Pr_B should be described using any relevant views. If a product's flexibility is relevant to the production, for example in the case of large panels that may deflect during assembly, then this should be modelled in Pr_B. Likewise, if the product is to be machined from a given material, the parameters that determine the deflection response of the material workpiece should be modelled in Pr_B. Another example is when the product is (or contains) liquids or gases—the behaviours of these substances are highly relevant to the design of the system behaviours and structures that will be used to manipulate and store them.

The structure of the product Pr_S may be expressed in any format that describes the product features, for example a CAD model. In the case of a product family, there are likely to be two views: one that describes the fixed attributes that are common to all variants (usually some sort of physical description), and one that describes the variables (this can be thought of as the "product recipe").

3.3 Behaviour

A key concept in behavioural approaches like FBS is the distinction between expected behaviour B_e and the actual behaviour expressed by the system's structure B_s. We maintain this distinction, but while previous work on FBS models was primarily concerned with fixed behaviour in a static system—which in our case would be a system resulting in a fixed assembly process—we are also concerned with adaptive behaviours. We therefore consider three types of behaviour in each case:[2]

Definition 2 (Behaviour) $B_x = \langle B_{x_{NAP}}, B_{x_{AP}}, B_{x_R} \rangle$ where $x \in \{e, s\}$,
B_e is expected behaviour—i.e. target behaviour,
B_s is structurally expressed behaviour—i.e. behaviour achieved in the real world,
$B_{x_{NAP}}$ is non-adaptive production behaviour,
$B_{x_{AP}}$ is adaptive production behaviour, and
B_{x_R} is reconfiguration behaviour.

$B_{x_{NAP}}$ *Non-adaptive production behaviour*: This is the type of behaviour considered in previous work on FBS models. It covers the "steady state" of production process in an assembly system and is often described as a simple sequence of actions.

$B_{x_{AP}}$ *Adaptive production behaviour*: The first addition of adaptivity is to the production behaviour above. Our systems are self-adaptive, leading to an adaptive function. This in turn implies adaptive behaviour. Behaviour cannot then be simple action

[2]The notation in this section can be applied either to the expected behaviours in B_e or the actually expressed behaviours in B_s.

sequences, but must instead encapsulate predictive decision-making.

B_{x_R} *Reconfiguration behaviour*: Finally, we explicitly require our systems to self-configure and self-reconfigure. This requires our systems to be able to integrate the "design process" into their own behaviours—to be able to identify when a reconfiguration needs to occur and what that reconfiguration should address, and then "design the change" that would accomplish it.

In terms of implementation, the production behaviours required of the system for a given product are determined by the set of tasks required to assemble it. In our work, these behaviours are specified as labelled transition systems (LTSs) that describe the flow of activity in each behaviour. For a given product to be produced, these behaviours are compiled into a recipe file that formalises the product recipe and fully captures the decision-making that may have to occur during the adaptive production of that product. To continue the hole-making example, where the function specified that a hole should *exist*, the behaviour would specify that a hole should be (for example) *drilled* at a given location, with a given tool, with a given accuracy (among other parameters). These parameters must be defined in relation to the structure of the system, otherwise the behaviour cannot be performed.

With these definitions of function and behaviour, we can see an overlap between the different aspects of each as shown in Fig. 2.

As system behaviour is either derived from function or generated by structure, there are a number of different views that can be taken into consideration, including motion definition based on kinematics [22], high-level production actions given by a Business-to-Manufacturing Markup Language (B2MML) description [42], or a "digital twin" simulation model [33]. Our model allows the designer to use the most appropriate perspective. Later in the paper, we present an example using the views of physical production topology and production processes.

3.4 Structure

An Evolvable Assembly System is an example of a modular, flexible, reconfigurable manufacturing system, and also a complex, context-aware, collective adaptive system of systems [38]. With this in mind, we must modify the definition of structure from the literature. The structure should deliver behaviours that will fulfil the desired system functions, and is defined by the components, modules, or sub-systems that comprise it, the connections between them, and the states that the system may be in. However, we must also consider a number of additional things:

- The system structure may change over time. System components may be added or removed, connections between components may change, and the system state will change. Although this last aspect of changeable state is not strictly new to the FBS literature—[49] notes that "structure" is informally just state that does not change very often—the changeability of the system components and connections is not explicitly addressed as a top-level object in existing frameworks.
- The system may have a set of "configurations" that it may be in at any given time. These configurations will encapsulate a set of components and connections and will often also include specific system state. For example, in a flexible and reconfigurable manufacturing system, the connections between system modules may change depending on the product to be manufactured, and the associated control code (which is part of the system state) may change to reflect this. The structure is a set of *possible* configurations, and the system must select a *single* configuration to be used at any given moment.

For the non-adaptive production process, the definition of structure from the literature given above is still mostly suitable. At this level of behaviour, we are only interested in the static assembly of a single product with a stable system in a fixed configuration. However, the addition of adaptivity to the process requires us to consider the other aspects of the system: that it is an open dynamic system made up of loosely coupled distributed autonomous resources. We must therefore consider the potential addition or removal of resources, that the connections between resources may change, and that a resource may even change its structure. As the structure of a system includes the hardware and software configuration of the system, the variables that make up the settings are part of our concept of structure for the system.

In terms of our hole-making example, the structure of the system needs to specify some resources that can exhibit the

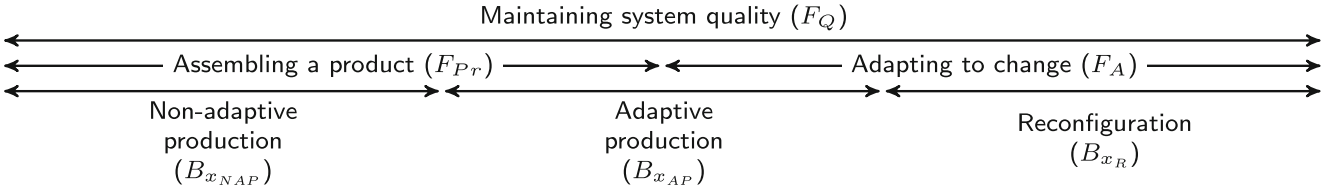


Fig. 2 The conceptual overlap between types of function (above) and types of behaviour (below) in an EAS

behaviours that we require. Therefore, it will need to have a drill and bits that are suitable for the material our product is made of (taking into account the system behaviours implied by the product behaviours). Along with this, it will also need some method of manipulating either the drill or the product such that the holes can be made in the correct positions with sufficient accuracy. Perhaps some metrology system could also be necessary to ensure this positional accuracy, or perhaps the manipulator and drill combined will have sufficient accuracy on their own—this is something to be decided either during the design process itself, or during operation.

We can therefore define structure as follows:

Definition 3 (Structure) $S = \{C_0, \ldots, C_n\}$ where $C_i = \langle M, Conn, State \rangle$ is a configuration, M is the set of system modules, components, or sub-systems, $Conn$ is the set of connections between elements in M, and $State$ is the state of the system.

As with function and behaviour, structure can also be described using different views. For example, each $m_i \in M$ may be described with a physical description of the hardware (most appropriate for traditional machining equipment), or in terms of its software architecture (most appropriate for a cloud manufacturing or agent-based control system). Likewise, where the $State$ of the system contains the "layout", this could be described topologically (in terms of the connections $Conn$ between modules), or geographically (in terms of the relative or absolute locations of modules). Similarly the $State$ may be concerned with the settings, or "configuration state", of a module, which for a software system may be the code being run or the values in a set of fixed variables, and in a hardware system may be various settings on the equipment.

3.5 Design process

The FBS framework of Gero et al. [14] describes both an ontological framework for Function, Behaviour, and Structure and a process for design using those concepts. We extend and adapt this process for the design of adaptive production systems. This is not only what Gero would call "creative design"—where all variables and their ranges can be modified, with no limits on the hardware that can be designed or included in the system—but also "innovative design" in terms of integration of existing hardware. In this case, the variables in R_A limit the adaptation by fixing some of the variables in the process and allowing some to be modified.

We can now consider the fully modified design process shown in Fig. 3:

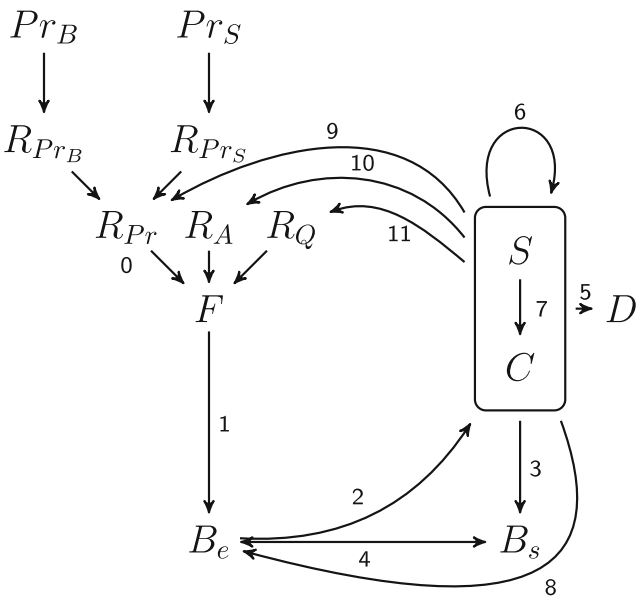


Fig. 3 EAS operational design process (see Section 3.5 for labels)

0. *Requirements gathering* takes the set of product requirements (R_{Pr} based on the product behaviour and structure $Pr_B \rightarrow R_{Pr_B}$, $Pr_S \rightarrow R_{Pr_S}$) and system adaptation and quality requirements (R_A and R_Q), and generates the system function (F) from them.
1. *Formulation* converts the required system function (F) into a set of behaviours (B_e) that are expected to result in the required function.
2. *Synthesis* generates a structure (S) containing a set of configurations (C) that are expected to exhibit the desired behaviours (B_e) and selects an initial configuration for the system.
3. *Analysis* is the process of observing or deriving the actual behaviour (B_s) exhibited by the currently chosen configuration of the structure (S).
4. *Evaluation* compares the two behaviours to determine whether the actual behaviour (B_s) matches the expected behaviour (B_e). The result of this comparison determines whether or not the "design" will be accepted, or if a reformulation is required.
5. *Documentation* involves the production of a design description or document (D) that describes the design to be implemented. This would be carried out if the result of the evaluation process is favourable. In our process, the output of this stage is a description of the changes to be implemented, in terms of the structure and configuration of the system.

6–11 *Reformulation* is the process by which changes can be made to the design if the result of the evaluation process is unfavourable. During reformulation, the process "loops back" to make a change to one of the properties of the design before the design process continues. Reformulation can take five forms:

6. *Type* 1_S uses changes to the structure or set of configurations to address unsatisfactory behaviour.
7. *Type* 1_C selects an existing configuration to address unsatisfactory behaviour.
8. *Type 2* uses changes to the (expected) behaviour to address unsatisfactory behaviour.
9. *Type* 3_{Pr} uses changes to the functional product requirements to address unsatisfactory behaviour or to respond to changes in product requirements.
10. *Type* 3_A uses changes to the system adaptation requirements to address unsatisfactory behaviour or to respond to changes in adaptation requirements.
11. *Type* 3_Q uses changes to the system quality requirements to address unsatisfactory behaviour or to respond to changes in quality requirements.

4 Implementation methodology

There are three main implementation challenges to address when considering our framework, as described in the following sections.

4.1 System functional requirements

The first challenge is formalising system functional requirements in a format that is machine-readable (stage 0 in the design process). The system requirements R_{Prs}, R_{Pr_B}, R_A, and R_Q are given as schemas defining a set of constants and variable attributes with associated ranges: $R = \langle \{constants\}, \{\langle variable, range \rangle\} \rangle$. The schema in R_{Prs} represents physical forms of the set of products that

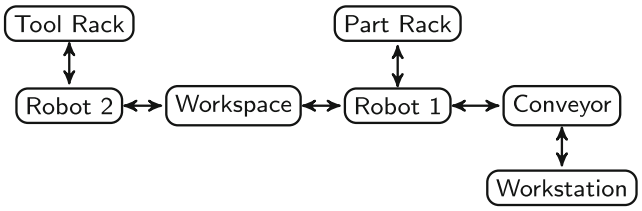
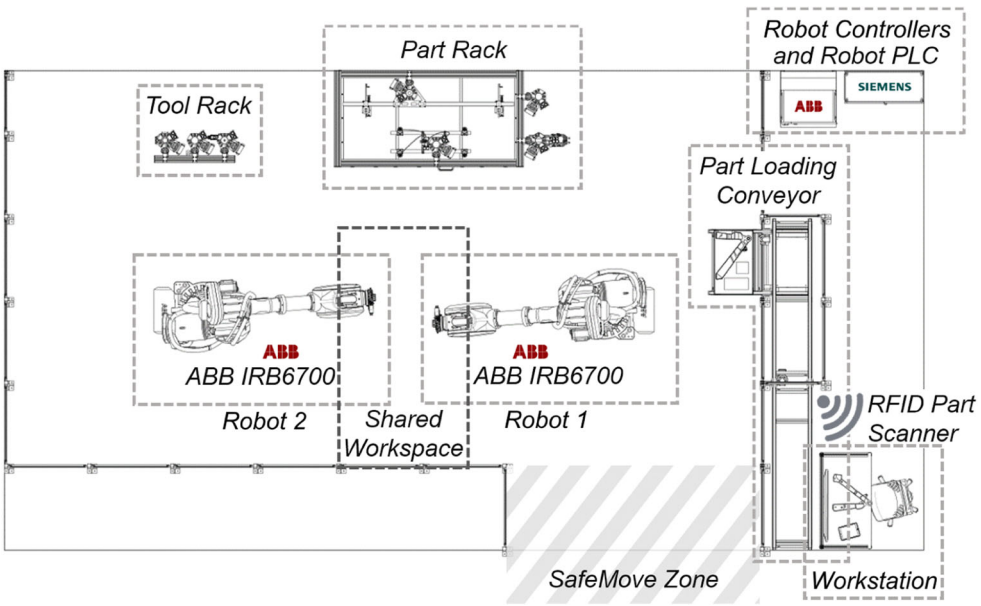


Fig. 5 Assembly cell connections

may be produced by the system: their shared attributes (the constants) and the ways in which they vary (the variables), as well as how much they can vary (the ranges). R_{Pr_B} represents the relevant aspects of product behaviour that should be considered—for example the flow characteristics of a liquid, or the relevant parameters for determining workpiece deflection during machining. Where R_{Prs} and R_{Pr_B} express potential product variation, R_A expresses allowable system variation in a similar manner. In the case of R_Q, these attributes are actually quality constraints on the system; the constants are binary invariants and the variables express ranged constraints. All of these schemas can be directly translated into XML schemas for machine readability. When it comes to producing a given product at a given time, the system function can be reduced to a single-product recipe. This is implemented as a recipe file that specifies the parts and operations required by the final product, the quality inspection tests to be carried out, and the possible responses to each test (i.e. rework, discard, etc). This recipe file is generated through stages 0–1, thereby corresponding to the required production behaviours of the system $B_{e_{NAP}}$ and $B_{e_{AP}}$, and takes the form of a labelled transition system that is stored in data structures specified in ANSI/ISA-95 [1] using B2MML (e.g. [2]).

Fig. 4 Assembly cell layout



The reconfiguration behaviours B_{e_R} are specified in the internal workings of the control system using a form of BDI architecture [11, 32].

4.2 System structure

The second challenge is formalising system structure in a format that is machine-readable. As described in Section 3, structure S is a set of configurations C where each $C_i = \langle M, Conn, State \rangle$: the production system is made up of a set of production resources (M: modules, components, or sub-systems) with state ($State$) connected together in some topology by their connections ($Conn$). Both the resources and topology are represented as labelled transition systems. Any previously existing resources are specified to the system by blueprint files, implemented using the same ANSI/ISA-95 and B2MML standards as the recipe files. Both the resources and topology can then be modified if necessary due to system adaptation.

4.3 Synthesis and evaluation of behaviour

The third challenge is around implementing methods for representing system behaviour during the design process. Stage 2 in the design process is the synthesis of a set of behaviours B_e that are expected to fulfil the system requirements. This is accomplished by a process of task simulation and controller synthesis [41] resulting in an annotated labelled transition system (mapping the recipe to the topology) that is then unfolded into an AND-OR tree. This AND-OR tree is represented by a B2MML "Operations Schedule" made up of several "Operations Requests" that can be interpreted by the agent control layer for execution on industrial hardware. While this controller is executed, the system tracks all attributes mentioned in F (Stage 3 of the design process, Analysis). As part of Stage 4, Evaluation, the system checks these attributes do not violate any associated ranges or constraints. A further pre-emptive check via virtual task simulation is done to compare the predicted system behaviour to the required system behaviour before actual execution [4]. This helps prevent a "try it and see" approach to the synthesis and reformulation steps. Likewise, once a suitable configuration is found by simulation, it can be saved for later use without having to generate from scratch.

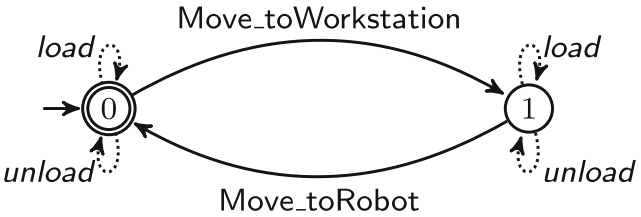


Fig. 6 Example labelled transition system for the cell conveyor generated from module structure and behaviour. Solid lines are processes and dotted lines indicate interaction with other resources

Table 1 Behaviours available to each module in the cell

Module	Behaviour	Sub-behaviour (optional)	Behaviour type
Conveyor	Move	To robot	$B_{x_{NAP}}$
		To workstation	$B_{x_{NAP}}$
Workstation	Load pallet		$B_{x_{NAP}}$
	Remove pallet		$B_{x_{NAP}}$
	Manual operation	(varies)	$B_{x_{AP}}$
Robot 1	Apply rivet		$B_{x_{AP}}$
	Move part	Conveyor → part rack	$B_{x_{AP}}$
		Conveyor → workspace	$B_{x_{AP}}$
		Workspace → conveyor	$B_{x_{AP}}$
		Workspace → part rack	$B_{x_{AP}}$
		Part rack → conveyor	$B_{x_{AP}}$
		Part rack → workspace	$B_{x_{AP}}$
	Change end effector	Pallet gripper	B_{x_R}
		Rib gripper	B_{x_R}
		Skin gripper	B_{x_R}
		Pneumatic rivet gun	B_{x_R}
Robot 2	Change end effector	Sealant gun	B_{x_R}
		Laser line scanner	B_{x_R}
Robots 1 and 2	Scan part		$B_{x_{AP}}$
	Apply sealant		$B_{x_{AP}}$

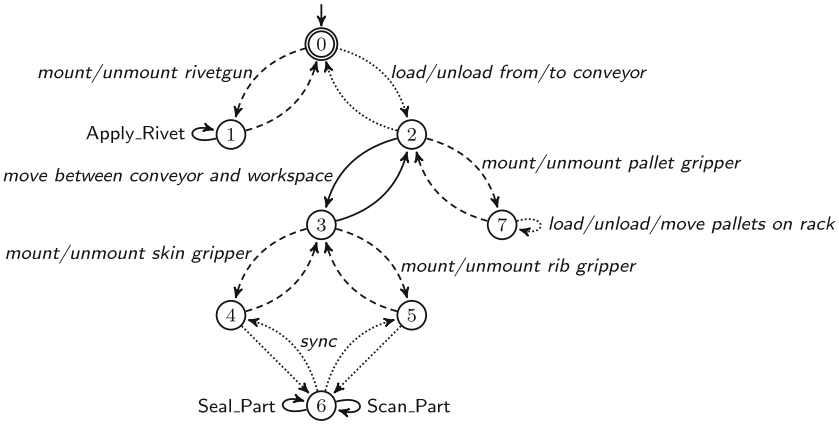


Fig. 7 Simplified example labelled transition system for robot 1 generated from module structure and behaviour. Solid lines are processes, dashed lines are configuration changes, and dotted lines indicate interaction with other resources. For readability, some labels have been combined and some states and transitions have been elided

5 Prototype implementation

The proposed design process has been verified using industrial data. To illustrate the approach, we present one use case based on an implemented proof of concept demonstrator at the University of Nottingham shown in Fig. 4. In this case, the view we are considering is that of the physical production processes, taking into account the production capabilities and topology of the cell. The cell's function is to inspect and assemble a variety of aerospace rib and skin components: this leads to requirements to handle the components, apply fasteners and sealant, and inspect the resulting products. Starting with some existing hardware, we use the presented approach in an "innovative design" manner to design the control approach and reconfigure the cell. The demonstrator consists of two ABB IRB6700 robots, a shared central workspace, a tool rack accessible by one robot, and both a shared tool/part rack and a part loading conveyor belt accessible by the other robot and by a human operator at a manual workstation. Each robot has access to a number of different end effectors on their respective rack and is equipped with an automatic tool changer.

The modules and connections of the structure can be seen diagrammatically in Figs. 4 and 5. Note that the end effector connections are ignored in Fig. 5 and will be presented in a simplified form here for clarity. This is because they can physically change location (in the rack or on the robot) and connection based on the reconfiguration behaviours. These different sets of connections, along with any drastic changes in state, result in the set of configurations C for the system.

This "first pass" over the system structure results in the set of processes (behaviours) shown in Table 1. Note that some behaviours are shared behaviours that require multiple modules to collaborate; for instance when robot 1 holds a part and robot 2 scans it or applies sealant. The structure and behaviour identified for each module is then used to generate an LTS for that module, as shown in Figs. 6 and 7 for example.

The final cell structure is given by $\langle M, Conn, State \rangle$ for each configuration, where the set of modules M and the interconnections $Conn$ are as follows:[3]

$$M = \left\{ \begin{array}{l} \text{Conveyor}, \\ \text{Workstation}, \\ \text{Robot 1}, \\ \text{Robot 2}, \\ \text{Part rack}, \\ \text{Tool rack}, \\ \text{Workspace}, \\ \text{Pneumatic rivet gun}, \\ \text{Sealant gun}, \\ \text{Laser line scanner}, \\ \text{Pallet gripper}, \\ \text{Rib gripper}, \\ \text{Skin gripper}, \\ \text{Parts} \end{array} \right\}$$

$$Conn = \left\{ \begin{array}{l} (\text{Workstation, Conveyor}) , \\ (\text{Conveyor, Robot 1}) , \\ (\text{Robot 1, Part rack}) , \\ (\text{Robot 1, } \textit{end effectors}) , \\ (\text{Part rack, } \textit{end effectors}) , \\ (\text{Robot 1, Workspace}) , \\ (\text{Robot 2, Workspace}) , \\ (\text{Robot 2, Tool rack}) , \\ (\text{Robot 2, } \textit{end effectors}) , \\ (\text{Tool rack, } \textit{end effectors}) \end{array} \right\}$$

The $State$ of the cell contains the program code and associated variables for each module, for example

[3] Although they are sets, M and $Conn$ are presented here in the style of vertical vectors for space considerations. Also note that this is for a view concerned with physical production capability only; if the system is analysed using a different view, the M and $Conn$ may be different. For example, when considering the design of the agent-level control system, the set of modules in M would be the set of agents in the system, and $Conn$ would be a fully connected network implemented via a databus technology.

program pointers, functional attributes, and generated data such as inspection results. This state will be distributed throughout the resource controllers, the Programmable Logic Controllers (PLCs), and the agent control system.

We will now sketch an example of how the design process is used to accommodate new products. To start, the system will be in a given configuration C_i to perform a sealant application on a rib. In this case, C_i will encompass the sealant gun, rib gripper, and pallet gripper (to place the rib on the pallet in the part rack to cure) in m_i but has no requirement for the skin gripper. This allows the system to continue functioning if the skin gripper needs to be reassigned to another cell if necessary, or removed for maintenance tasks. If the system then receives a new requirement in R_{Pr} to operate on skins, then type 3_{Pr} reformulation will trigger and the system will switch to the existing configuration, say C_j, that encompasses the skin gripper in m_j. As the set of behaviours already includes a skin gripper, no change in B should be required. In the case of a new skin geometry, this would entail behavioural analysis and evaluation to check that the existing gripper was capable of picking the skin. If a completely novel product is added to R_{Pr}, for example a spar, then the synthesis stage of the process will fail to generate a suitable structure and attempt to source a spar gripper by triggering type 1_S reformulation. In our real instantiation, this would notify an operator of the need for a spar gripper, and reject the operations request.

6 Conclusions

6.1 Discussion

The reported research addresses the need for behavioural approaches to the design and modelling of flexible and reconfigurable production systems of the sort studied in the Evolvable Assembly Systems project. A novel design process and associated definitions have been proposed for such systems, taking into account how system structure and behaviour relate to the intended system functions. The process has been developed in such a way as to enable automatic implementation by an agent control system. This has been demonstrated through a real example instantiation in the aerospace assembly domain.

The main advantages of the approach are that it enables a wide range of dynamic requirements, takes advantage of the inherent self-x properties of the system components, and can be applied to automated agent-based control systems. In order to aid robustness and applicability, the approach has a formal foundation in both design theory (functional modelling) and automated reasoning (labelled transition systems and task simulation), combined with a standards-compliant approach to implementation (using ANSI/ISA-95 data structures in B2MML).

6.2 Contribution and further work

The paper presents a novel extension of the Function-Behaviour-Structure (FBS) frameworks of Gero, Rosenman, Umeda, and others from the field of functional modelling. The presented approach extends the theoretical FBS frameworks not only to fully accommodate self-adaptive systems but also to accommodate production systems whose function is determined by the FBS of other artefacts (i.e. the products to be produced). This novel theory is supported by practical application on real industrial hardware through a standards-compliant implementation to demonstrate automatic design, integration, and reconfiguration in response to changing product requirements.

As discussed in this paper, new approaches based on functional and behavioural models are particularly suited to addressing the challenges of disruptive production environments resulting from Industry 4.0 and similar trends. Consequently, the contributions described here are highly relevant to the manufacturing industry. High-value manufacturing, such as in the aerospace domain, is of particular relevance. The ability of the system to reconfigure according to changing requirements allows for an increase in utilisation, and a reduction in cost and change over time resulting in increased productivity. The example instantiation is in a research demonstrator designed to be a "relevant environment" [5] representative of a real industrial environment, and work is ongoing to further develop and apply the approaches described here to industrial demonstrations at higher Technology Readiness Levels [5, 39].

Our future theoretical research focus will be on more complex integration of hybrid human-machine decision-making through a joint cognitive systems approach [18]. This should provide for easier integration of automated and manual operations in a production environment. A potentially interesting avenue of review is the automatic generation of certain aspects of the control or test code for the system, followed by the comprehensibility by a designer, tester, or operator of the system; previous work in the area has highlighted a number of challenges [19, 45].

Acknowledgements The authors gratefully acknowledge the support provided by UK EPSRC Evolvable Assembly Systems (EP/K018205/1), as well as our colleagues Lavindra de Silva, Paul Holmes, and Brian Logan for their invaluable assistance during the project.

Funding information This research was supported by EPSRC grant EP/K018205/1 "Evolvable Assembly Systems."

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. ANSI/ISA-95, Enterprise-control system integration, parts 1-5. https://www.isa.org/standards-publications/
2. Business to manufacturing markup language operations schedule version 6.0. https://services.mesa.org/ResourceLibrary
3. Manufuture: a vision for 2020, assuring the future of manufacturing in Europe (2004). http://www.manufuture.org/
4. Bakker OJ, Chaplin JC, de Silva L, Felli P, Sanderson D, Logan B, Ratchev S (2017) Toward process control from formal models of transformable manufacturing systems. Procedia CIRP 63:521–526
5. Bilbro JW (2007) A suite of tools for technology assessment AFRL technology maturity conference. In: AFRL technology maturity conference
6. Boër C, Pedrazzoli P, Sacco M, Rinaldi R, De Pascale G, Avai A (2001) Integrated computer aided design for assembly systems. CIRP Ann Manuf Technol 50(1):17–20
7. Bortolini M, Faccio M, Gamberi M, Pilati F (2017) Multi-objective assembly line balancing considering component picking and ergonomic risk. Comput Ind Eng 112:348–367
8. Bortolini M, Faccio M, Gamberi M, Pilati F (2018) Motion analysis system (MAS) for production and ergonomics assessment in the manufacturing processes. Comput Ind Eng
9. Bortolini M, Ferrari E, Gamberi M, Pilati F, Faccio M (2017) Assembly system design in the Industry 4.0 era: a general framework. IFAC-PapersOnLine 50(1):5700–5705
10. Brad S, Murar M, Brad E (2017) Design of smart connected manufacturing resources to enable changeability, reconfigurability and total-cost-of-ownership models in the factory-of-the-future. Int J Prod Res: 1–23
11. Chaplin JC, Ratchev SM (2018) Deployment of a distributed multi-agent architecture for transformable assembly. In: Proceedings of the eighth international precision assembly seminar
12. Dorst K, Vermaas PE (2005) John Gero's Function-Behaviour-Structure model of designing: a critical analysis. Res Eng Des 16(1-2):17–26
13. Erden M, Komoto H, van Beek T, D'Amelio V, Echavarria E, Tomiyama T (2008) A review of function modeling: approaches and applications. AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22(02):147–169
14. Gero JS (1990) Design prototypes: a knowledge representation schema for design. AI Mag 11(4):26–36
15. Gero JS, Kannengiesser U (2003) Function-behaviour-structure: a model for social situated agents. In: Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions, International Joint Conference on Artificial Intelligence, pp 101–107
16. Gero JS, Kannengiesser U (2007) A function–behavior–structure ontology of processes. AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 21(04):379–391
17. Gero JS, Kannengiesser U (2014) The Function-Behaviour-Structure ontology of design. In: An anthology of theories and models of design. Springer, London, pp 263–283
18. Golightly D, Sanderson D, Holmes P, Ratchev S, Sharples S (2016) Design requirements for effective hybrid decision making with evolvable assembly systems. In: Proceedings of the European Conference on Cognitive Ergonomics - ECCE '16. ACM Press, pp 1–7
19. Hashim NL, Ibrahim HR, Rejab MM, Romli R, Mohd H (2018) An empirical evaluation of behavioral UML diagrams based on the comprehension of test case generation. Adv Sci Lett 24(10):7257–7262
20. Hehenberger P, Vogel-Heuser B, Bradley D, Eynard B, Tomiyama T, Achiche S (2016) Design, modelling, simulation and integration of cyber physical systems: methods and applications. Comput Ind 82
21. Jefferson TG, Benardos P, Ratchev S (2015) Reconfigurable assembly system design methodology: a wing assembly case study. SAE International Journal of Materials and Manufacturing 9(1)
22. Kontovourkis O, Phocas MC, Lamprou I (2015) Adaptive kinetic structural behavior through machine learning: optimizing the process of kinematic transformation using artificial neural networks. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 29(04):371–391
23. Koren Y (2005) Reconfigurable manufacturing and beyond. In: CIRP 3rd International Conference on Reconfigurable Manufacturing
24. Koren Y, Heisel U, Jovane F, Moriwaki T, Pritschow G, Ulsoy G, Van Brussel H (1999) Reconfigurable manufacturing systems. CIRP Ann Manuf Technol 48(2):527–540
25. Leitão P, Barbosa J, Trentesaux D (2012) Bio-inspired multi-agent systems for reconfigurable manufacturing systems. Eng Appl Artif Intell 25(5):934–944
26. Lohse N (2006) Towards an ontology framework for the integrated design of modular assembly systems. Ph.D thesis
27. Mizoguchi R, Kitamura Y (2001) Foundation of knowledge systematization: role of ontological engineering. In: Industrial knowledge management. Springer, London, pp 17–36
28. Monostori L (2014) Cyber-physical production systems: roots, expectations and R&D challenges. Procedia CIRP 17:9–13
29. Onori M, Semere D, Lindberg B (2011) Evolvable systems: an approach to self-X production. Int J Comput Integr Manuf 24(5):506–516
30. Pahl G, Beitz W (1996) Engineering design: a systematic approach, 2nd edn. Springer, Berlin
31. Rampsersad HK (1994) Integrated and simultaneous design for robotic assembly. Wiley, New York
32. Rao AS, Georgeff MP (1995) BDI Agents: From theory to practice. In: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pp 312–319
33. Rosen R, von Wichert G, Lo G, Bettenhausen KD (2015) About the importance of autonomy and digital twins for the future of manufacturing. IFAC-PapersOnLine 48(3):567–572
34. Rosenman M, Gero J (1998) Purpose and function in design: from the socio-cultural to the techno-physical. Des Stud 19(2):161–186
35. Rosenman M, Gero J (1999) Purpose and function in a collaborative CAD environment. Reliab Eng Syst Saf 64(2):167–179
36. Rosenman M, Wang F (2001) A component agent based open CAD system for collaborative design. Autom Constr 10(4):383–397
37. Sadeghi L, Dantan JY, Mathieu L, Siadat A, Aghelinejad MM (2017) A design approach for safety based on Product-Service Systems and Function?Behavior?Structure. CIRP Journal of Manufacturing Science and Technology
38. Sanderson D, Antzoulatos N, Chaplin JC, Busquets D, Pitt J, German C, Norbury A, Kelly E, Ratchev S (2015) Advanced manufacturing: an industrial application for collective adaptive systems. In: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp 61–67. IEEE

39. Sanderson D, Turner A, Shires E, Chaplin J, Ratchev S (2019) Demonstration of transformable manufacturing systems through the evolvable assembly systems project. In: Aerotech Americas. SAE International
40. Sasajima M, Kitamura Y (1995) FBRL: A function and behavior representation language. In: Proceedings of the 14th International Joint Conferences on Artificial Intelligence (IJCAI), pp 1830–1836
41. de Silva L, Felli P, Chaplin JC, Logan B, Sanderson D, Ratchev S (2016) Realisability of production recipes. In: European Conference on Artificial Intelligence (ECAI)
42. de Silva L, Felli P, Chaplin JC, Logan B, Sanderson D, Ratchev S (2017) Synthesising industry-standard manufacturing process controllers. In: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, pp 1811–1813. International Foundation for Autonomous Agents and Multiagent Systems
43. Stadzisz P, Henrioud J (1998) An integrated approach for the design of multi-product assembly systems. Comput Ind 36(1-2):21–29
44. Suh NP (1998) Axiomatic design theory for systems. Res Eng Des 10(4):189–209
45. Sunitha EV, Samuel P (2016) Object oriented method to implement the hierarchical and concurrent states in UML state chart diagrams. Springer, Berlin, pp 133–149
46. Telgen D, van Moergestel L, Puik E, van Zanten A, Abdulamir A, Meyer JJ (2013) Automatic structured decomposition of manufacturing actions in an agent-based manufacturing system. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol 2, pp 155–162
47. Tolio T, Ceglarek D, ElMaraghy H, Fischer A, Hu S, Laperrière L., Newman S, Váncza J (2010) SPECIES — co-evolution of products, processes and production systems. CIRP Ann Manuf Technol 59(2):672–693
48. Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the function-behavior-state modeler. Artificial Intelligence for Engineering Design Analysis and Manufacturing 10:275–288
49. Umeda Y, Takeda H, Tomiyama T, Yoshikawa H (1990) Function, behaviour, and structure. In: Applications of artificial intelligence in engineering V, vol 1, pp 177–193
50. Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P (1998) Reference architecture for holonic manufacturing systems: PROSA. Comput Ind 37(3):255–274
51. Vermaas PE, Dorst K (2007) On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology. Des Stud 28(2):133–157
52. Zha X, Du H, Lim Y (2001) Knowledge intensive Petri net framework for concurrent intelligent design of automatic assembly systems. Robot Comput Integr Manuf 17(5):379–398

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.