Send Orders for Reprints to reprints@benthamscience.ae

Control Software Modeling in Production Systems

Sabah Al-Fedaghi[1,*] and Faisal Al-Shahin[2]

[1]*Department of Computer Engineering, Kuwait University, Kuwait*

[2]*Department of Information System Audit, State Audit Bureau of Kuwait, Kuwait*

Abstract: To meet current challenges, manufacturing industries must develop precise designs for interactions of various components of production control systems, and in particular for behavior of the software controlling them. Several diagrammatic techniques, e.g., SDL, statecharts, and UML, have been utilized to represent the static and dynamic aspects of control systems. Still, a conceptual framework is lacking that would integrate components and allow assembly of applications from shared processes. This paper addresses the issue by providing schemata to be used in the conceptual modeling of production control systems. A flow-based specification that has been utilized in software engineering is proposed as a good vehicle in this area. The resulting description achieves uniformity by using a single flow system to represent all production system elements. The methodology is demonstrated by recasting the diagrams of two specific research projects in unified flow models.

Keywords: Conceptual Model, Flow Model, Production Control System, PLC, SDL, Statecharts, UML Diagrams.

1. INTRODUCTION

This paper builds a *conceptual model* in the domain of *production control systems*. Generally speaking, the term *model* refers to an abstract representation of some part of the "real world" developed as a means of communication among stakeholders when a complex system is built. Desirable features of models include concise capture of relevant and significant aspects of a real phenomenon, understandability, and completeness of activity specifications. A conceptual model uses diagrammatic notations to provide a high-level representation of essential concepts and their interrelationships in a real-world domain. Its purpose is to convey a common description without technological aspects, and it serves as a guide for the subsequent systems design phase. In this context, Unified Modeling Language, UML [1, 2], has been utilized for at least two purposes: object-oriented software design, and conceptual modeling. "However, UML's origins in software engineering may limit its appropriateness for conceptual modeling" [3].

Production systems for industrial goods are multifaceted systems involving many aspects, including cells with parallel machines, machine failures, assembly cells, timing, and processing. A *production control system* links all parts of a manufacturing environment to ensure coordination of production and distribution activities to achieve specific delivery reliability at minimum cost. It provides the mechanism necessary to execute production in order to achieve some goal.

*Address correspondence to this author at the Department of Computer Engineering, Kuwait University, Kuwait; Tel: +965-99939594;
E-mail: sabah@alfedagi.com

1.1. Motivation

According to Nickel *et al.* [4],

Today's manufacturing industry faces big challenges. … In order to meet these challenges today's production control systems become more and more decentralized [5]… One major problem in building those decentralized systems is to develop a detailed and precise plan how those different components interact and in particular how their control software has to behave. There is no specification approach which allows to define, analyze and simulate such a production control system upfront before it is actually built, physically.

To develop this type of specification, several diagrammatic techniques have been used, including Specification and Description Language (SDL) [6, 7] and UML. UML has been utilized to build conceptual models representing the static and dynamic aspects of control systems by using class diagrams, collaboration diagrams, activity diagrams, and statecharts [4, 8-11].

1.2. Problem

In large-scale systems, processes are typically large in scale and great in complexity, especially with currently emerging networked applications such as those used in control of production control systems, grids, and integrated supply chains. Development of such systems is becoming more and more multidisciplinary, with many stakeholders, including client, architect, engineers, designers, contractors, and consultants, all of whom are dependent on each other for information. The impacts of failures in the design of these systems are quite costly. Difficulties in communication as well as in integration of subsystems have arisen from variations in representations of notions used by the various disciplines.

1874-4443/15

A key problem in this context is *lack of a unifying framework for development and management*. "Lack of consideration of engineering projects as systems-of-systems has led to methodological challenges in creation of integrated tools and techniques for better analysis and management of complex projects" [12]. A great deal of current research has focused on methodological approaches to functional specifications, flow descriptions, and system structure definitions [13][14][15] such as the model-based process [16], in which the system is specified at various levels of granularity. Nevertheless, an underlying tool for expressing the unified totality of a system's processes and concepts is lacking.

1.3. Outline of Proposed Solution

This paper addresses the need for a conceptual foundation for system modeling with a focus on production control systems. A flow-based model (called the Flowthing Model, FM) that has been utilized in software engineering is proposed as a viable modeling apparatus in this area. Without loss of generality, the paper focuses on specific research that has developed actual diagrammatic representations in the field of production control systems. Specifically, the paper focuses on two projects in this area:

- A simulated production process developed by Köhler *et al.* [8] is one of the most complete simulations and model of a factory production system in the field and reflects current approaches.
- In many production lines, manufacturing operations are generally controlled by Programmable Logic Controller programs (PLCs). Park *et al.* [17] proposed a diagrammatic modeling methodology that produces several integrated diagrams.

The approach of this paper of targeting specific works for viewing through the lens of FM is appropriate for the nature of diagram-based modeling and can be justified according to the following discussion.

Consider, for example, the following ways in which diagrammatic models are used in Unified Modeling Language (UML) [18]:

- As a sketch to communicate some aspect of a system. "These [UML] sketches can be used in both forward (i.e. devising new systems) as well as reverse (i.e. understanding existing systems) engineering" [18].
- As a blueprint by which to build (possibly with a more detailed design first). "Blueprints are about completeness" [18].
- As a programming language with diagrams compiled down to executable code.

This paper deals with uses of the first two types; however, there is no fundamental obstacle to extending the work to the third use when the field of programming of diagrams has become more mature. Accordingly, focusing on existing diagrammatic representations has the advantage of providing *immediate* contrast between views.

2. AUTONOMOUS PRODUCTION AGENTS

Köhler *et al.* [8] propose using various UML diagrams as visual programming language in a modeling approach to autonomous production agents in a decentralized production control system. Their proposal is suitable for the purpose of this paper of describing a proposed alternative to UML-based depiction, because the closer the diagrams are to the programming stage, the more precise they are in exemplifying the processes of the modeled system.

The original problem discussed by Köhler *et al.* [8] begins with difficulty in specifying production agents using available methodologies such as System Description Language (SDL) [6, 7] and statecharts [19, 20]. According to Köhler *et al.* [8], SDL process diagrams and statecharts "lack appropriate means for the specification of the actual actions triggered by the received signals."

To overcome this problem, Köhler *et al.* [8] propose incorporating several different UML diagrams into one drawing. They introduce UML activity diagrams as high-level control flow notations for graph rewrite rules:

In order to facilitate the use of graph rewrite rules for object-oriented designers and programmers, we additionally adapted UML collaboration diagrams as a notation for object-structure rewrite rules. For this combination of activity diagrams and collaboration diagrams we use the name story-diagrams… for the specification of complex application specific object structures [8].

Köhler *et al.* [8] have also developed a formal and analyzable specification language for manufacturing processes. The initial phase of such a process can be described as follows:

After the requirements engineering and analysis work, our approach proposes an *analysis consolidation phase*, where the topology and the block diagrams have to be specified. From this … topology and overall behavior one may derive an SDL block diagram by identifying participating processes and communications.

Still, we observe that a conceptual framework is missing that would integrate the components while allowing assembly of applications from shared processes, as will be demonstrated next.

3. MOTIVATIONAL EXAMPLE

Köhler *et al.* [8] introduced a simulated production process based on a factory production system [9]. This production process models a factory with multiple manufacturing sites and shuttles that transport goods from one site to another.

3.1. Description of the Production Process

Fig. (**1**) shows the topology of the factory. In this factory, one main track has multiple transfer gates to reach assembly lines.

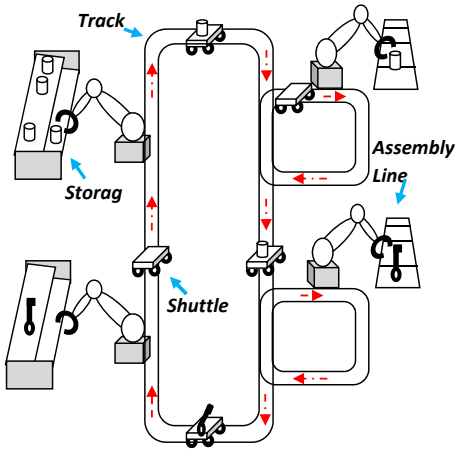

Fig. (1). Partial view of factory production system (partial view, redrawn from [8]).

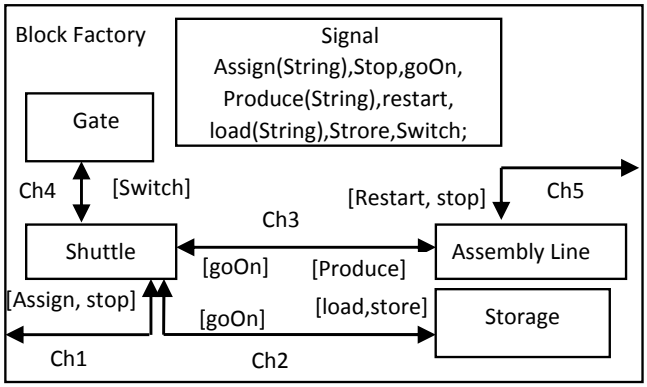

Fig. (2). SDL block diagram of sample factory (partial view, redrawn from [8]).

The assembly lines on the right as well as the storage areas on the left are called *stations*. Stations and their related robot arms build an operating unit. Assembly lines are able to manufacture various kinds of products. In the track system, each shuttle executes a defined working task. The task starts when a shuttle is assigned to produce a certain good.

3.2. Producing Keys

In this production-system cycle, pieces of metal are obtained from raw material storage (which we can also consider a station in the production system) and moved to an assembly line. The process starts when the shuttle reaches an assembly line and requests its assigned good (key). The related robot takes metal from the shuttle (e.g., upper right assembly line), and manufactures the good, using various tools (e.g., lower right assembly line). Later, the produced good is put on the shuttle surface and brought to finished goods storage (lower left storage). After the shuttle has moved to the storage area and the good is stored, the task is ended and the shuttle starts again from the beginning. The shuttle will continuously execute this task until it receives a new assignment or is stopped.

3.3. SDL and UML Diagrams

From this topology and overall behavior, one can derive an SDL block diagram by identifying participating processes and communications between them. In the example, the processes are shuttles, gates, assembly lines, and storages. These processes can be derived from the text description in which shuttles, the central parts, communicate with other parts in the factory in order to complete their tasks. Fig. (**2**) shows the corresponding SDL block diagram for the factory example. It contains the four processes Gate, Shuttle, Storage, AssemblyLine, and communication channels (Ch1...Ch5). While such a diagram does not present a fair description of SDL methodology, it is sufficient for our purpose of contrasting it with our own diagrammatic methodology.

From SDL block diagrams, one can derive an initial class diagram with process classes for each identified process as well as signal methods for each signal understood by these process classes. In addition, the class diagrams are used to generate code that initializes the production control system, creates the production agents, and establishes the communication channels. Each process in the SDL block diagram generates a class in the initial class diagram. Fig. (**3**) shows part of a UML class diagram for the production process example. It contains classes like Shuttle and Gate derived from the SDL block diagram.

The translation of class diagrams to an object-oriented programming language is straightforward and provided by most current Object-Oriented Case (OO-CASE) tools [21, 22]. Generation of code for UML class diagrams provides

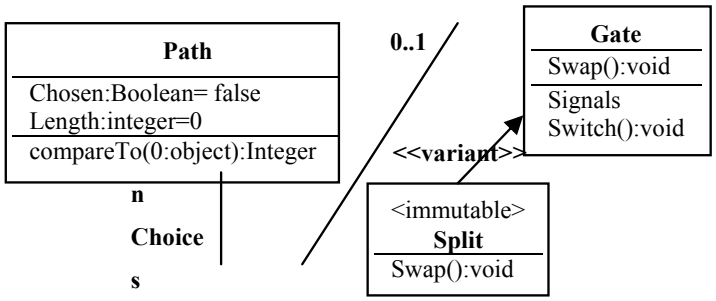

Fig. (3). Class diagram (partial, from [8]).

for the translation of behavior diagrams. Statecharts are used to specify the reactive behavior of process classes derived from SDL block diagrams.

In order to simulate production processes, code is generated from statechart modeling; therefore, Köhler *et al.* [8] propose combining statecharts and collaboration diagrams, with statecharts specifying complex control flows and collaboration diagrams specifying transition actions. To use collaboration diagrams as a visual programming language, one must add many details showing how the participating objects are found and how object structure modifications are executed .

Köhler *et al.* [8] also utilize collaboration diagrams and combine statecharts and collaboration diagrams to produce so-called story charts. Combining high-level control flow specifications (statecharts or activity diagrams) and high-level object structure (collaboration diagrams) results in a visual programming language that can be used as a means for specifying the reactive behavior of a flexible production agent.

4. SCRUTINY

The methods detailed in the previous section reflect laudable efforts to provide a complete methodology, from a text description of the production process to generation of code through stages of diagrams; nevertheless, there is a sense of missing a global specification of the system, like having detailed drawings of various systems in a high-rise building, including diagrams of electrical wiring and plumbing pipes, but no overall blueprint of the building itself.

An architectural blueprint describes the underlying framework, the shared processes, and the standardized components used to build the new architecture. It is highly integrated in a component-based style that allows assembly of applications from shared process functions when available. In our modeling context, the blueprint can be based on a model to produce an *infrastructure architecture* comprising various types of operations in all spheres and subspheres (e.g., manual vs. automated processes).

With regard to the SDL diagram shown in Fig. (**2**), we can observe the sketchy description rendered in this type of diagram. It includes programming-based notations (e.g., *Assign(String),Stop,goOn*, …), and the arrows seem to denote relationships and operations (e.g., *stop*, *go*). What does *Switch* mean on the bidirectional arrow between Gate and Shuttle? Processes are not clear; are they commands? (*go*, *stop*, *load*). To whom are these commands directed?

Nor can the statechart diagram serve as a conceptual description of the production control system. For example, the unlabeled arrow from *Fetch* to *GoProduce* seems to be a "flow control" mechanism that denotes performing *Fetch*, then performing *GoProduce*; however, the directed arrow labeled *Reached* seems semantically different from a mere "flow control". Similarly, the beginning points, one labeled *Active* and one with no label, seem conceptually hazy.

To show the contrast between the new FM process specification and other methodologies, e.g., UML, in the next section we review the methodology of our Flowthing Model, as described in several publications (e.g., [23-30]), before reconstructing the example and its extension. The example in the next section is a new contribution. Section 6 is a complete revision of first published materials [27], and sections 7 and 8 are new contributions.

5. FLOWTHING MODEL

The Flowthing Model (FM) depicts the way a system is structured by providing a roadmap of its components and conceptual flow. Components are termed *spheres* (e.g., a company, robot, human, assembly line, station), that may enclose or intersect with other spheres (e.g., the sphere of a house contains rooms, which in turn include walls, ceilings). Or, a sphere embeds flows (called *flowsystems*; e.g., walls encompass pipes of water flow and wires of electrical flow).

Things that flow in a flowsystem are referred to as *flowthings* (e.g., money, data, products, cars, parts). The lifecycle of a flowthing is defined in terms of six mutually exclusive *stages*: creation, process, arrival, acceptance, release, and transfer. Within a certain sphere:

- *Creation* means the appearance of a flowthing in the totality of the system of a sphere for the first time (e.g., creation of a new user in a computer system).

- *Process* means applying a change to the form of an existing flowthing (e.g., painting a product).

- *Release* means marking a flowthing "to be output" while the flowthing remains in the sphere (e.g., a product marked "to be shipped").

- *Transfer* denotes the input/output module of the sphere (e.g., the interface component (port) of a device with a communication channel).

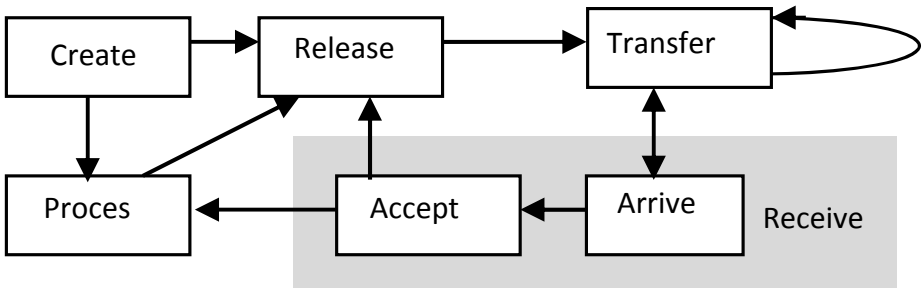

Fig. (4). Flow system.

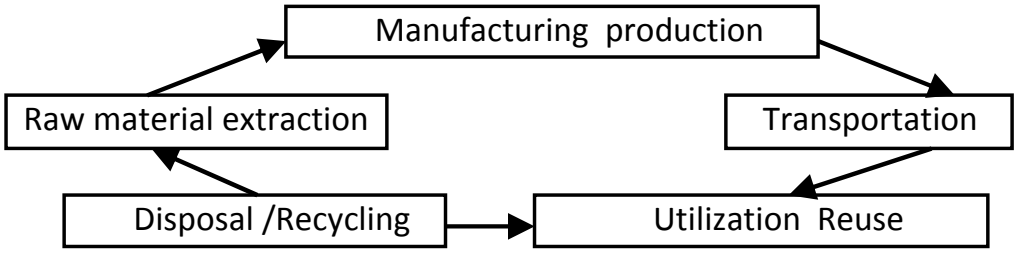

Fig. (5). Lifecycle of products (from [26]).

- *Arrival* means that the flowthing reaches the sphere but is not necessarily allowed to enter it (e.g., a letter addressed to the wrong receipient, to be returned).

- *Acceptance* means allowing the arrived flowthing to enter the new sphere.

Fig. (**4**) shows a flowsystem with its stages, where it is assumed that no released flowthing flows back to previous stages (e.g., DHL returns a package because an airport is closed after a disaster). This is a typical assumption in systems where the exceptional cases are ignored. For example, in a model of the crude oil refining process, different stages progress in producing residual, heavy gas oil, lubricating oil, diesel, kerosene, gasoline, and naphtha. The assumption, here, is continuous heating of the mixure to about 1112 °F (600 °C).

If for some reason the boiler fails, then some substances may revert to their previous states – but this is not mentioned when the refining process is depicted. Similarly, a flowsystem depicts a situation of one-way flow; flowthings move forward.

The reflexive arrow indicates flow to the *Transfe*r stage of another flowsystem. For simplicity's sake and where appropriate, the stages Arrive and Accept can be combined into a single stage called *Receive*.

The *stages* in the lifecycle of a flowthing are mutually exclusive (i.e., the flowthing can be in one and only one stage at a time). All other states of flowthings are not exclusive states. For example, we can have stored created flowthings, stored processed flowthings, stored received flowthings, etc. Also, flowthings can be released but not immediately transferred (e.g., the channel is down), or arrived but not accepted, …

In addition to flows in a flowsystem, *triggering* is a transformation (denoted by a dashed arrow) from one flow to another, e.g., a flow of electricity triggers a flow of air.

Example 1: Product lifecycle is "the entire lifecycle of a product from its conception, through design and manufacture, to service and disposal" [31]. Fig. (**5**) is a typical representation of a product lifecycle.

Viewing such a cycle from the FM perspective, we notice that the five spheres in Fig. (**5**) include two flowthings: raw materials and products (manufactured materials). Fig. (**6**) shows the flow stream of these two things and reflects the following components:

Raw materials are produced (created) in a raw material extraction sphere (see circle 1), released (2), and transferred (3) to the Manufacturing/Production sphere. Note that raw materials may stay in the released state (e.g., waiting to be shipped) until being transported. Note also that since the raw material extraction sphere has only one flowsystem, we need only one box to represent the sphere and its flowsystem, not two separate ones.

The Manufacturing/Production sphere has two flowsystems: one for raw materials and the other for products; thus received raw materials are processed, triggering (4) the creation of products that move to the Transportation sphere (5). Products then flow to the Utilization/Reuse sphere (6), where they are processed/used (7) and then at the end flow to the Disposal /Recycling sphere (8).

The Disposal/Recycling sphere has two flowsystems: one for product and the other for raw materials. Used products are processed (circle 9) either to flow back to the utilization/reuse sphere (10), or to trigger (11) the creation of raw materials (12) that flow to the raw material extraction sphere (13), to be released (14) back to the Manufacturing/Production sphere.

Example 2: Bock [32] gives an example (adapted from [33]) that shows the dependencies among some subfunctions of an automobile (see partial view in Fig. **7**). The first step in the activity is to TURN KEY TO ON.

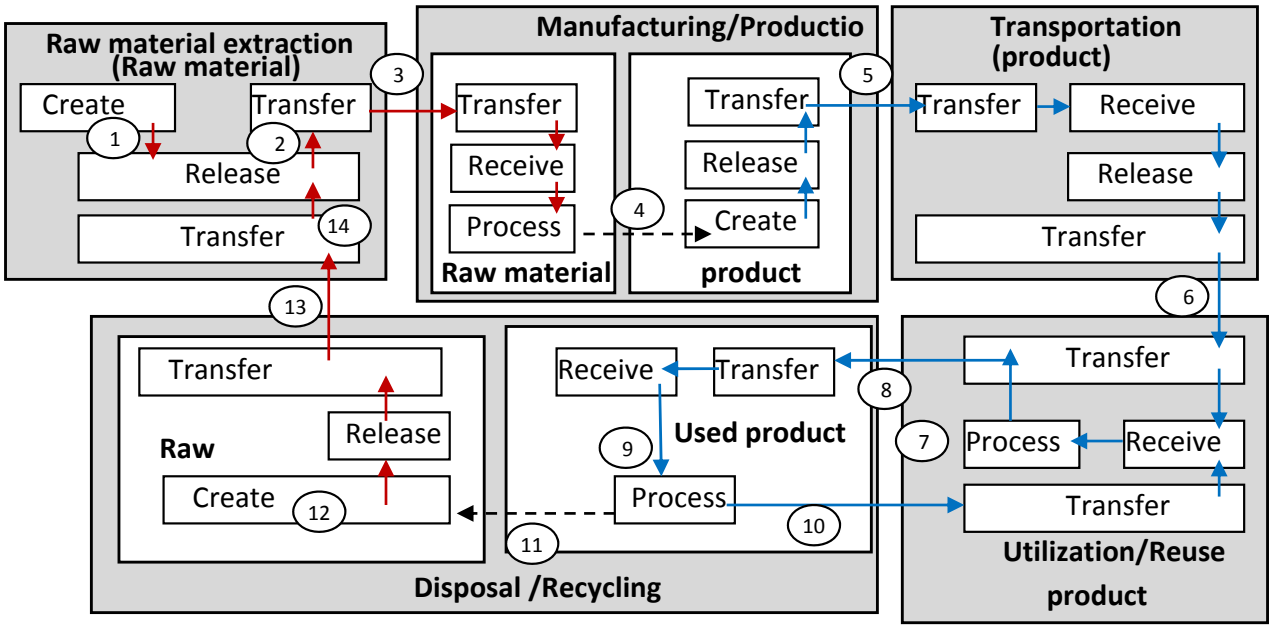

Fig. (6). FM description of a "product lifecycle".

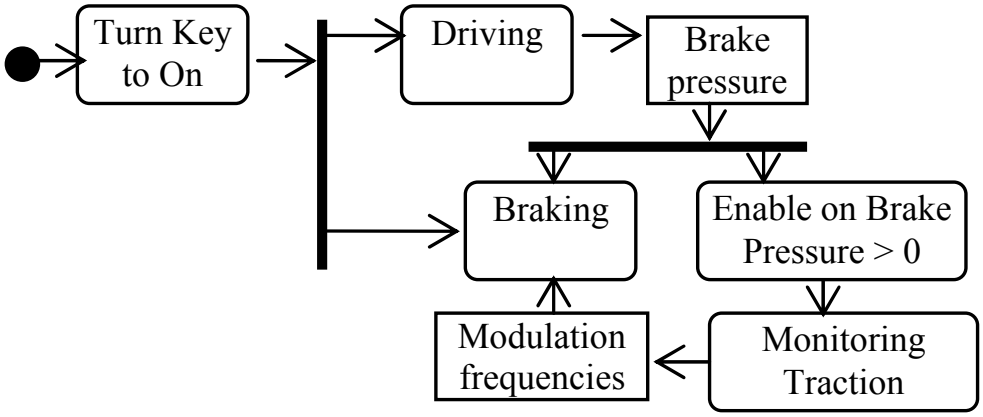

Fig. (7). Example (partial view, from Bock [32]).

Brake pressure *information* is passed from Driving to Braking (see the figure). The information is sent concurrently to Braking and to Monitoring Traction.

When brake pressure is greater than zero, it outputs an enabling control value to monitor traction, otherwiseMonitoring Traction emits Modulation frequencies to Braking as necessary to maintain traction and disable control value. Traction is monitored only when pressure is applied to the brake.

Fig. (**8**) shows an FM representation of this example. Since the purpose of the example is to illustate the nature of the diagramming methodology, we ignore several issues in the original activity diagram such as constraints. Numbered circles are used to explain the FM diagram.

Fig. (**8**) shows four spheres: Key, Driving, Braking, and Anti-lock braking system (ABC) – circles 1, 2, 3, and 4, respectively. ABC includes the monitoring system and is present in the original example given by SysML Partners [33] and adapted by Bock [32].

Note that *creation* in FM denotes a new flowthing that emerges (flows) or comes into *existence* from non-existence. Accordingly, *states* (in the engineering sense) are flowthings that can be *created*. In this sense creation is a *flow*. In general, states can also flow fron the creation stage to the processing stage, etc.

In the Key sphere, in the *State* flowsystem, the created state ON (circle 5) triggers (6 and 7, respectively) the creation of ON states in Driving and Braking spheres (2 and 3, respectively).

The state ON means putting Driving and Braking into the execution (activated) condition. This triggers (8) the generation of Brake pressure (9) which flows to the Braking sphere (10) and the ABS sphere (11).

In the ABS, and according to our understanding, the ABS is triggered (12) to the ON state when Brake pressure arrives and it is > 0. Of course the FM description can be changed if this understanding is incorrect. Accordingly, the Monitoring module is triggered (13) to activate, go to the ON state, (14)

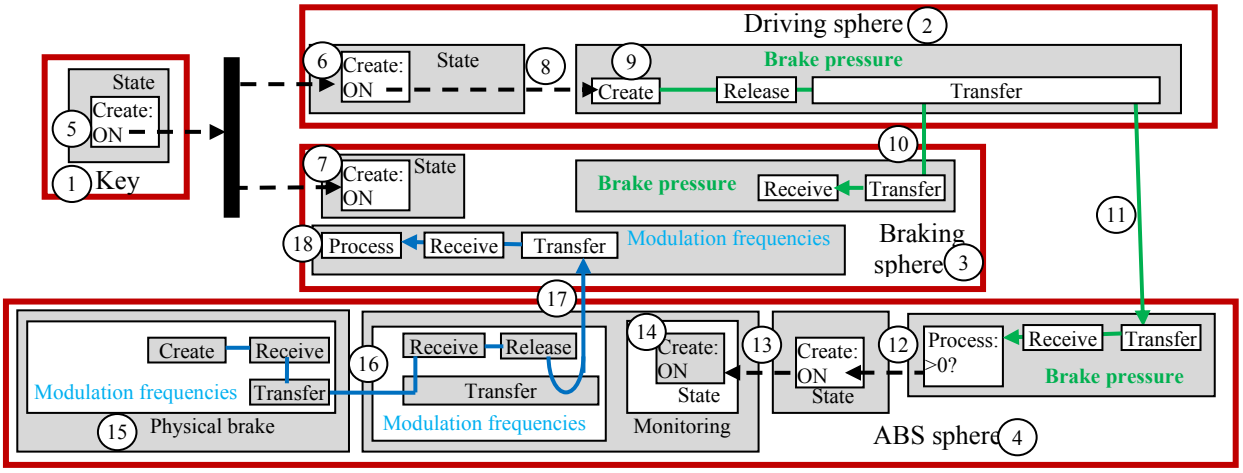

Fig. (8). FM representation of automobile subfunctions.

and be ready when the physical brake (15) sends (16) Modulation frequencies. Modulation frequencies flow (17) to the Braking sphere to be processed (18).

Note the contrast between Figs. (**7** and **8**). Fig. (**7**) is a mix of heterogeneous notions: the arrows may mean control or data flow and the boxes may denote a process, e.g., Driving, or a decision, e.g., Pressure > 0.

Activities as modeled in Fig. (**7**), e.g., turn, enable, brake, … seem, in principle, to embrace any and all English verbs, thus include unlimited "types" of vertices selected from English verbs by the designer. This gives an imprecise connotation of a mixture of meanings for different implementations of the same system. It would be like defining a graph in graph theory by selecting an infinite number of node types according to personal preference.

In FM, the diagram produces a systematic representation in which streams of flow trigger each other and processes are built on repeated application of the five stages. This idea will be demonstrated further by applying FM to a production control system.

6. APPLYING FM

Returning to the factory production system discussed in section 3, we claim that the FM representation provides a conceptual foundation for the modeling of such a system. Fig. (**9**) shows FM with a production site, *Source Station*, where goods arrive to be loaded on shuttles, and a *Target Station*, where goods can be stored. Each shuttle executes a defined working task: to transport goods from the Source Station to storage. Initially, all the shuttles wait (inactive and not loaded) in a garage.

When a shuttle is assigned a new task by a *Start* command (circle 1), a shuttle is released from the queue and *Shuttle State* is switched to *Active*. According to the task assignment parameters, a shuttle will be transferred to a specified *Source Station* (2), where a product is waiting to be loaded. When a shuttle "arrives" at the *Source Station*, some checking procedure is followed to "accept" the shuttle, for example, ensuring the shuttle is not loaded and is in the right position to be loaded.

If the shuttle is "accepted" (3), it triggers the product to be "released" and loaded onto the shuttle (4). The shuttle is now loaded with the product and ready for transfer to the *Target Station* (5).

When the shuttle "arrives" at the *Target Station*, some checking procedure is again followed to "accept" the shuttle by ensuring the shuttle is loaded and in the right position to be unloaded. If the shuttle is "accepted" (6), a product is released from the shuttle and transferred to the store (7). After the shuttle is unloaded, it is ready to be transferred back to the "garage" (8). When the shuttle arrives at the garage, checking procedures will again be followed to "accept" the shuttle, such as making sure the shuttle is not loaded and has delivered the product. If the shuttle is "accepted", it enters a queue where it waits for another task assignment (9) to start the process again with the same cycle.

At any time the shuttle can be halted with the Emergency Stop signal and switched to an Inactive state, then reactivated with the Activate signal and switched to the Active state (10,11,12).

FM can also be used as a foundation at a more detailed level along a flow, or at a higher level with an abstract description to form a conceptual representation of flow between spheres. Fig. (**10**) shows an abstract-level FM for a factory production system derived from Fig. (**9**). From the FM shown in Fig. (**9**), we can derive a language to represent the flow model for a factory production system, as shown partially in Fig. (**11**).

7. APPLYING FM IN PROGRAMMABLE LOGIC CONTROLLERS

In many production lines, manufacturing operations are generally controlled by Programmable Logic Controllers (PLCs) [34]. A PLC is a specialized computer that, with input from sensors, controls functions at different levels of complexity. Before PLC, control logic was represented by an

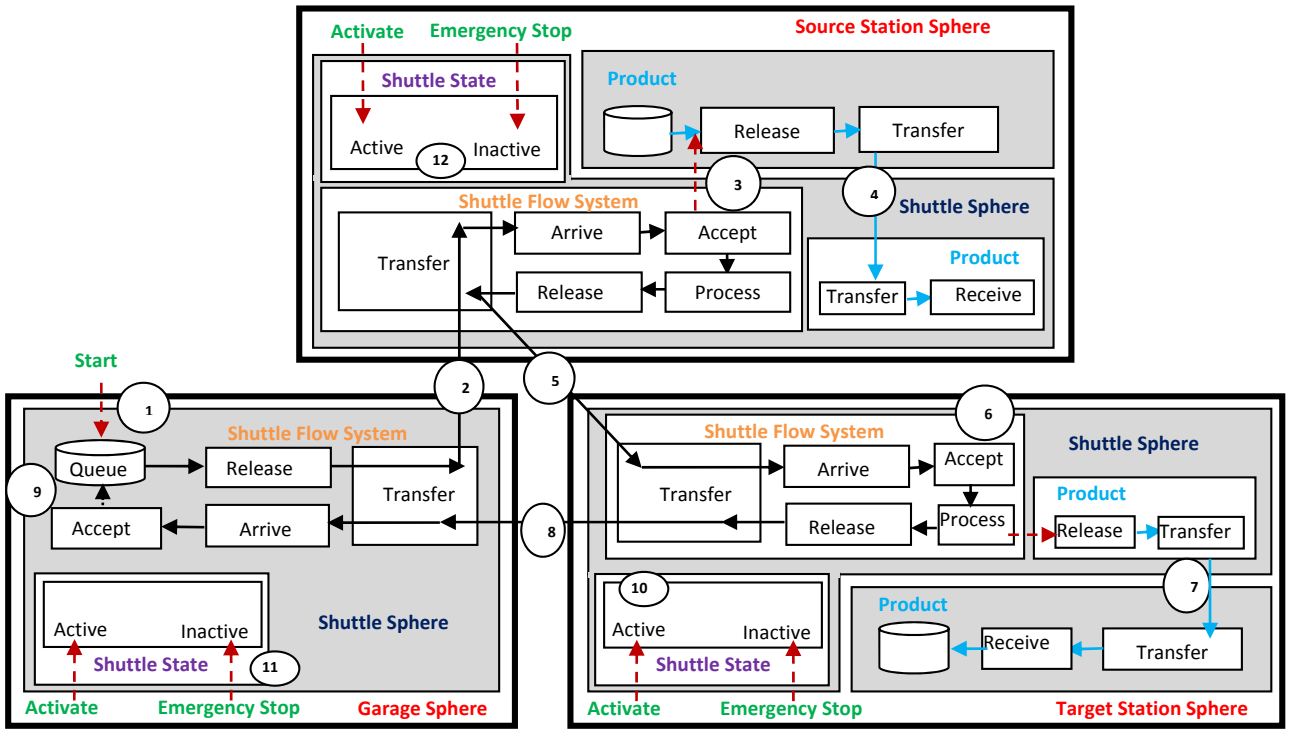

Fig. (9). FM of factory production system.

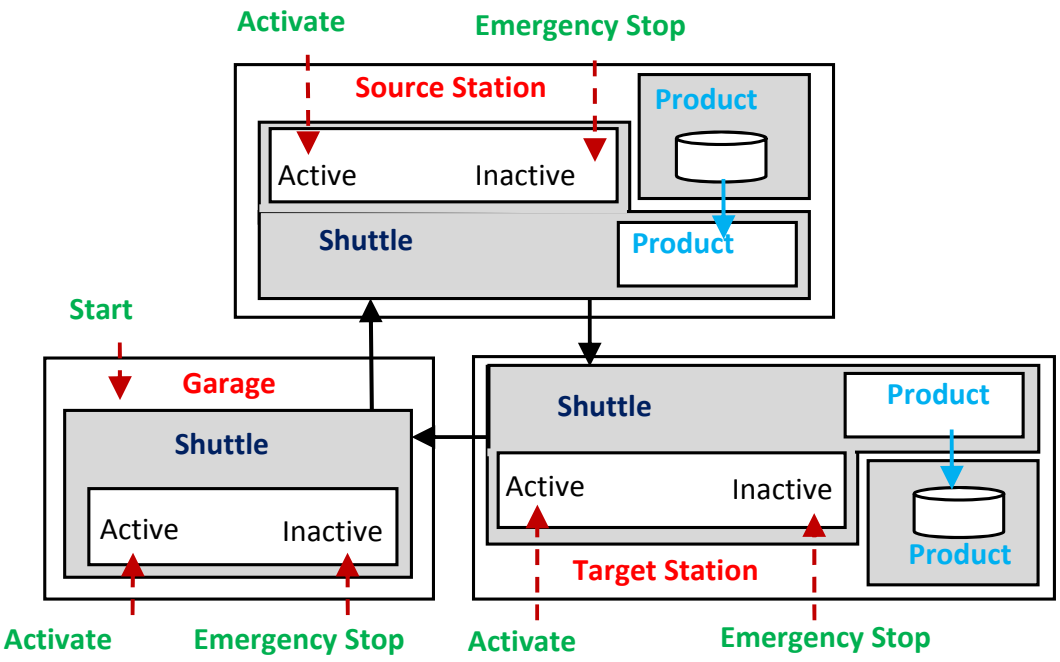

Fig. (10). An abstract-level FM for a factory production system corresponds to Fig. (**9**).

electrical circuit diagram; since then, PLC has replaced the hard-wired logic circuit with a software-like control box [17].

PLC is widely employed in industrial systems for process control [17, 34-36]. Data are continuously gathered by sensors in scanning cycles and trigger output operations according to logic specified in the PLC. Diagrammatic representations such as ladder diagrams [37, 38] are used in logic specification.

According to [35], steps to develop an industrial production control system include the following:

1. Define the process to be controlled

2. Make a sketch of the process operation

```
GarageSphere
{
        //different attibutes with their states in class
GarageSphere
        Shuttle
        {Shuttle:release,transfer,arrive,accept,
         State: Active,Inactive,}
  //emergency condition
   IF      Emergency      Stop=Active      then
Shuttle.State=Inactive
   ELSE
  //If there is no emergency then all the process will
be proceeded
    Shuttle.relase then
    Shuttle.transfer
  // transfer contains all the further steps till almost
end
    {
        SourceStationSphere
        {
  //different attibutes with their states in class
SourceStationSphere

            Product{ Product: relase, transfer}

   Shuttle{Shuttle:release,transfer,arrive,accept,
            State: Active,Inactive,
           Product: transfer,recieve}
  //emergency condition
            IF     Emergency     Stop=Active     then
        Shuttle.State=Inactive
            ELSE …
```

Fig. (11). Partial possible language specification that corresponds to Fig. (**9**).

3. Create a written sequence of the process

4. On the sketch, add the sensors needed to carry out the control sequence

5. Add the manual controls needed for the process setup or for operation checks

6. Consider the safety of the operating personnel and make additions and adjustments as needed

7. Add the master stop switches required for a safe shutdown

8. Create a ladder logic diagram that will be used as a basis for the PLC program

9. Consider the possible points where the process sequence might go astray.

Park *et al.* [17] have introduced a modeling methodology using multidiagram techniqes to generate PLC code. This methodology provides an opportunity to contrast their approach with FM in the context of PLC. According to Park et al. [17], in the early design phase of a manufacturing system, the control aspect is not intensely considered. Consequently, verifying and modifying this aspect of the system is costly. PLC simulation methods can be utilized for PLC verification [39-41]. As discussed by Park *et al.* [17], the problem is that such an approach is not realistic enough to be applied to detail design. "It is necessary to create a much more detailed simulation model that can forecast not only the production capability of the system but also the physical validity and efficiency of co-working machines and control programs" [42].

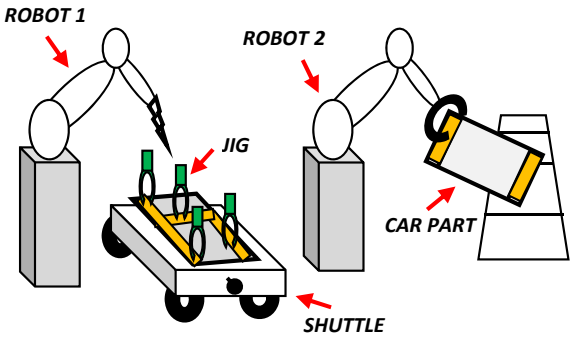

Fig. (12). Welding station of car manufacture (redrawn from a figure in [17]).

In this context, Park *et al.* [17] point out three requirements:

The first is defining states for each device, which is triggered by PLC output signal. The second is giving operations (state) sequence in each process unit. In the last, the modeling methodology has to be possible to define the hierarchical position of a device bearing whole factory aspect.

Accordingly, they propose a modeling methodology comprising the following main steps:

1. Define the hierarchical structure of all devices in the factory

2. For each device, define States to be controlled by PLC.

3. Generate PLC symbols and I/O model

4. In each station defined in Step 1, give the process sequence by selecting the device state defined in Step 2.

Park *et al.* [17] apply the proposed modeling methodology to the case of a manufacturing cell shown in Fig. (**12**), which is part of a car body assembly line.

Suppose that the line name is 'BODY', the process name is 'WELD', and example cell is consisting of welding robot(RBT1), handling robot(RBT2), incoming shuttle(SHTL1) and jigs(JIG) on the shuttle. The process sequence is as follows.

a) shuttle comes into the WELD station (station start).

b) jigs clamp to grasp the body part for welding.

c) welding robot starts to weld.

d) jigs unclamp to be unloaded by the handling robot.

e) handling robot unloads the finished part. [17].

Next, this modeling methodology is briefly described, not with the aim of complete understanding, but to use as an example of a depiction that is conceptually extensive and diagrammatically involved. This motivates introducing FM as a comprehensive representation to form an underlying framework for diagrammatic process modeling.

As a first step in Park *et al.'s* [17] approach, a hierarchical structure is introduced for a portion of the car body assembly line, as shown in Fig. (**13**). The *WELD station* is

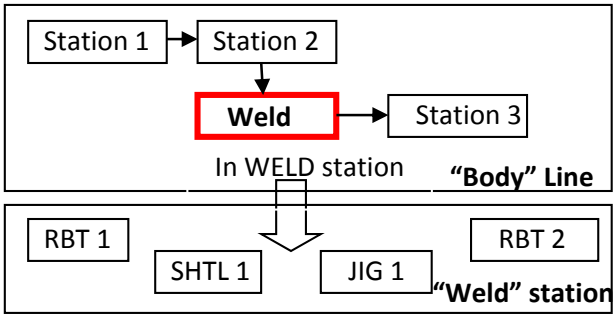

Fig. (13). Modeling the "device hierarchy" (redrawn from [17]).

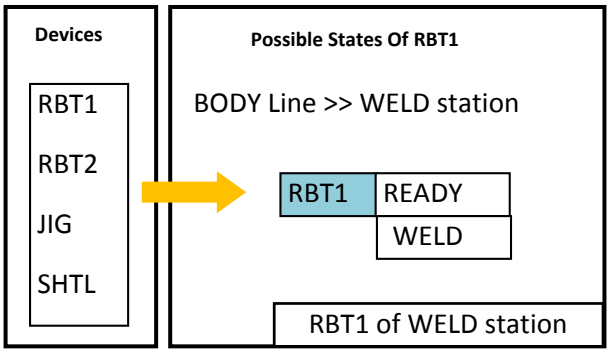

Fig. (14). Defined states of RBT1 (redrawn from [17]).

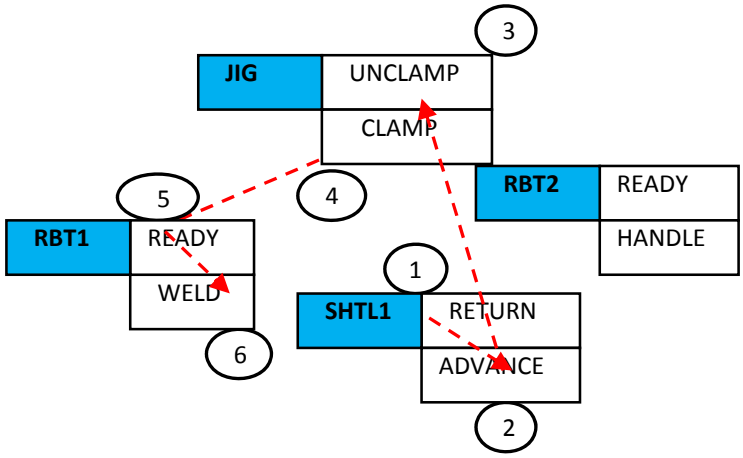

Fig. (15). Process sequence in Step 3 (redrawn from [17]).

part of the BODY line, with four devices belonging to the WELD station.

The *states* of each device in the WELD station are then defined. For example, RBT1 is in the READY state before the station process begins. RBT1 begins to weld when the SHTL comes into the station and the JIG is clamped. Therefore, the states of RBT1 are defined as READY and WELD, as shown in Fig. (**14**). The states of the other devices are defined in the same way.

From Figs. (**13** and **14**), the related symbols for RBT1 can be generated to produce an I/O model. One of the possible states is WELD, in which there can be one pair of PLC symbols: the triggering PLC output signal to begin welding, and an input signal that reports whether the Robot has completed that welding.

From Figs. (**13** and **14**), the related symbols for RBT1 can be generated to produce an I/O model. One of the possible states is WELD, in which there can be one pair of PLC symbols: the triggering PLC output signal to begin welding, and an input signal that reports whether the Robot has completed that welding.

From this modeling process, the device I/O model and PLC symbols are generated. Accordingly, the process sequence incorporating all devices and their predefined states is represented in Fig. (**15**).

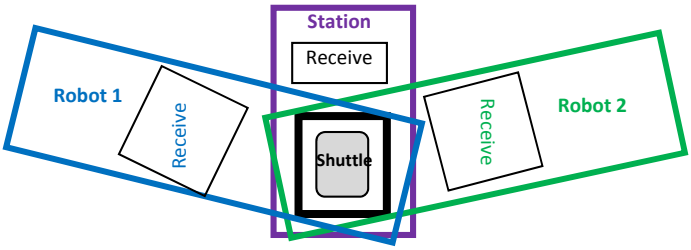

Fig. (16). Three spheres receive a part simultaneously when a shuttle arrives at the station.

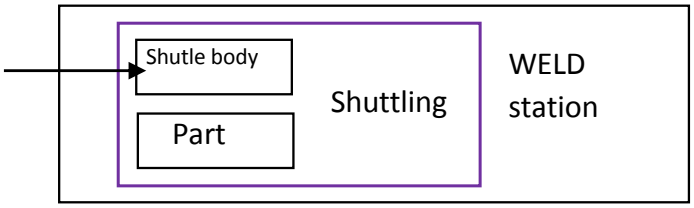

Fig. (17). Conceptual representation of the shuttle entering WELD station.

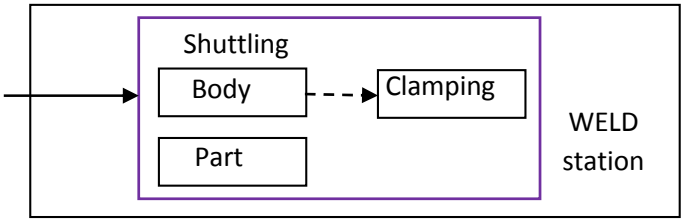

Fig. (18). Triggering a process of "clamping".

This extensive utilization of multiple diagrams raises the issue of the need for a conceptual framework for an overall representation of the system. In their depiction, Park *et al.* [17] start with a narrative description, then expand on it in successive waves of diagrams in an attempt to model the system by pulling it apart and inserting diagrammatic gadgets (e.g., state relationships) to represent its dynamics. The result is loss of a higher-level perspective or systemic *understanding* that would enable seeing the overall "forest" to improve modeling of the intricate web of conceptual connections between subprocesses. Models are utilized to demonstrate the design of physical products with the aim of increasing *understanding* of real-world objects while disregarding insignificant details. They are blueprints that allow us to gain the big picture, comprehend how parts fit together, develop vocabulary necessary for conducting discussions, and bring structure to the development process. These aims seem not to be achieved in the approach under consideration.

FM presents such qualities in modeling of physical systems. The FM structure consists of assembled blocks (spheres) that are superimposed flowsystems, the same way an architectural structure incorporates blocks of flow systems, such as an electrical system, a communication system, and a water system. The conceptual flow in FM provides a view of simultaneous processes acting on the same physical object, as shown in Fig. (**16**).

Notice that one physical flow is occurring in this picture: movement of the shuttle, carrying the part into the station. When it arrives at its position in the station to be processed, the part (not the shuttle) conceptually flows into the spheres of Robot 1 and Robot 2. The robots can process the part simultaneously in a synchronized way.

8. FM REPRESENTATION OF WELDING SYSTEM

According to the description given by Park *et al.* [17], processing proceeds as follows:

1. "(a) Shuttle comes into the WELD station" [17]. This is the first stream in FM, in which the shuttle *flows* into the sphere of the WELD station. An implicit understanding of the given phrase indicates that the shuttle carries a (shuttled) car part to be welded. Conceptually, the Shuttle and the Part are subspheres of one item we call Shuttling. Thus, we have the conceptual picture of spheres shown in Fig. (**17**).

This process of "implicit understanding" is analogous to saying *a car entered the garage* to mean *a driver drove a car into the garage*. FM forces us to explicitly distinguish between the Shuttle and the Part, because what is welded is the Part, not the Shuttle, and this is made even clearer with further development of the diagram.

2. "(b) Jigs clamp to grasp the body part for welding" [17], indicating that the flow of Shuttle *Body* triggers the process of "clamping". "Clamping", according to our understanding, involves the placing of jigs in certain physical places on the *Part*. Conceptually, clamping is performed in the sphere of Shuttling. Suppose that, by error, the Shuttle arrives with no Part mounted on it. The jigs are still placed in the designated places in the space of the missing part, regardless. Thus, we have the conceptual picture shown in Fig. (**18**).

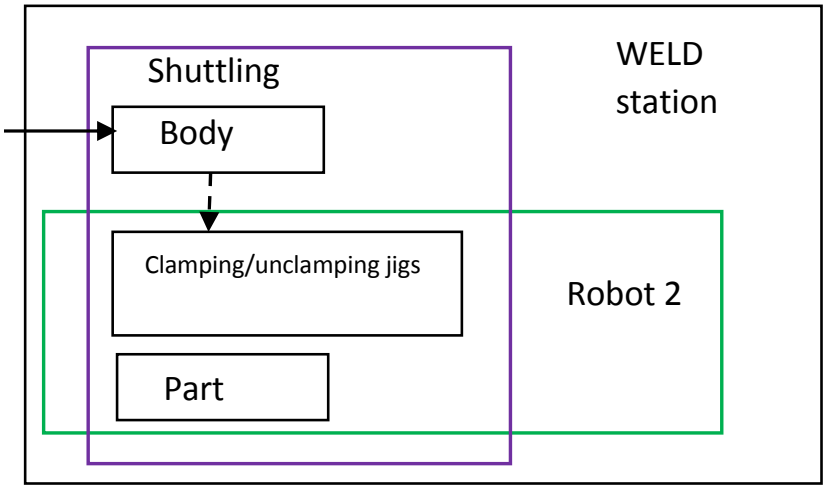

Fig. (19). Jigs and Part in Robot 2 sphere.

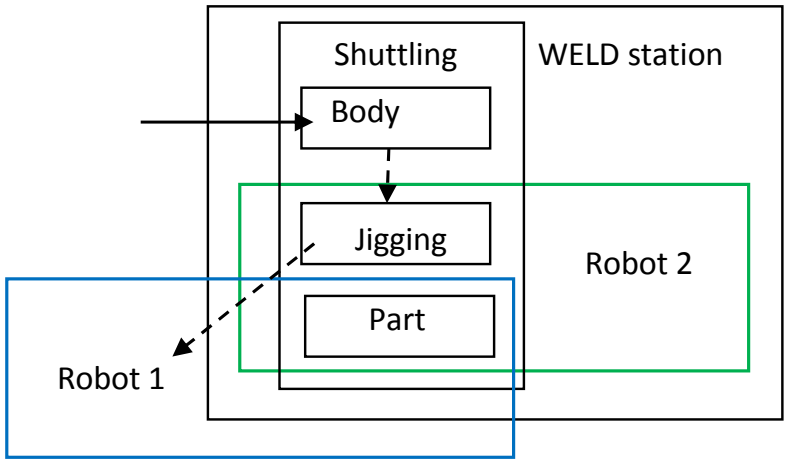

Fig. (20). Jigs and Part sharing both Robots' spheres.

3. "(c) Welding robot starts to weld" [17]. Information is missing at this step of the description, which is: who places the jigs for welding? Does this step have its own control mechanism? According to our understanding, Robot 2 performs such action because Park *et al.* [17] mention that this robot removes the jigs. Of course, if such an understanding is not correct, the FM description can be changed. Accordingly, Jigging and Part (which will be removed by Robot 2) are also in the sphere of Robot 2, as shown in Fig. (**19**). Also, "placing the jigs" triggers turning ON Robot 1 to Weld the Part. Accordingly, we get the representation shown in Fig. (**20**).

Fig. (**21**) shows the complete FM representation of the welding system. It starts at circle 1 in the figure, where the Shuttle, coming from a previous station (station 2), enters the welding station. Its acceptance for processing triggers (2) clamping, which triggers (3) turning ON Robot 1 (3). Robot 1 triggers the welding process (4). At the end of this welding, two processes are activated:

a) Turning OFF Robot 1 to be in Ready state (5)

b) Triggering Robot 2 (6) to unclamp the jigs (7)

As a result, Unclamping triggers unloading the Part at the outside welding station (8). Transferring the *Part* (9) to the next station (station 3) will trigger turning OFF Robot 2, putting it in READY state again. This triggers releasing the shuttle (10) to go back to the previous station (station 2).

Fig. (**21**) presents a schema of the portion of the car body assembly line discussed in this section. It is built upon a systematic methodology that uniformly applies an FM of spheres, flowsystems, stages, flows, and triggering. The result is a picture of the conceptual space in which each sphere handles its internal processes, separately, in a synchronized scenario. The description serves as an initial blueprint base for any further details in the design.

According to Chuang *et al.* [35], steps to develop an industrial production system controller include the following:

1. Define the process to be controlled

2. Make a sketch of the process operation

3. Create a written sequence of the process

4. On the sketch, add the sensors needed to carry out thecontrol sequence

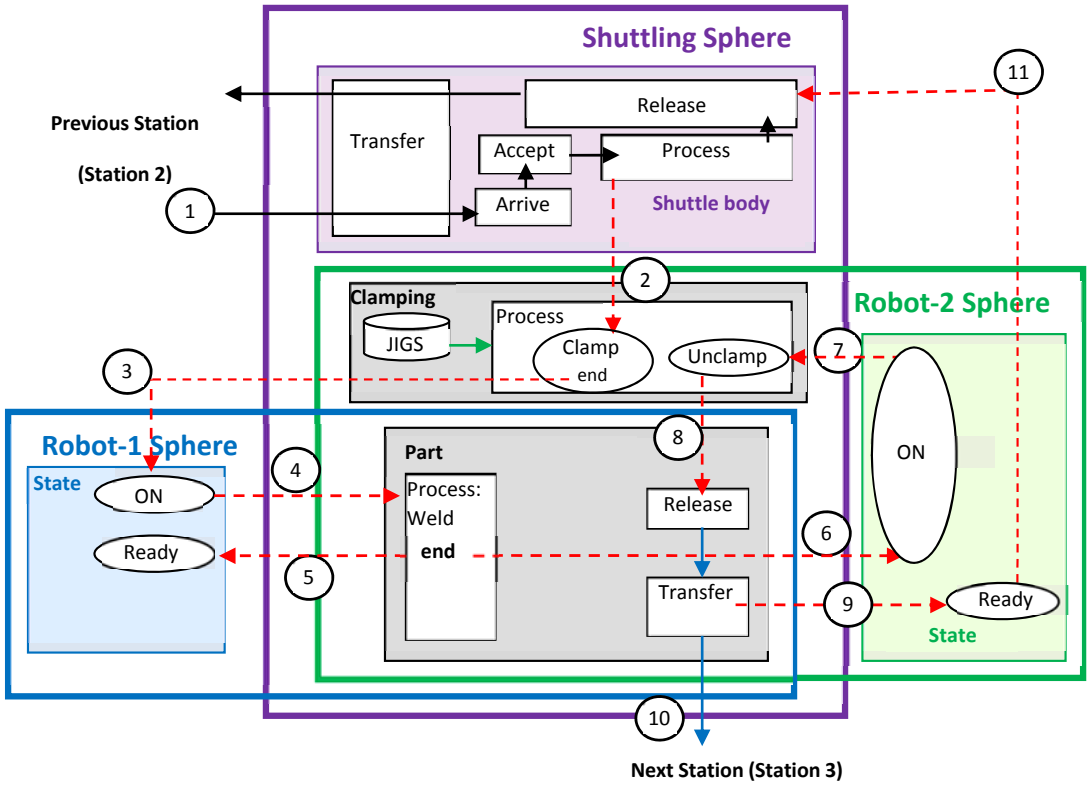

Fig. (21). FM representation of the welding system.

- Sensors can easily be added to an FM schema since they can be modeled as spheres with flows of their own.

5. Add the manual controls needed for the process setup or for operation checks

- The process setup and any type of checks and constrants can be added to the FM schema to control the flow by using additional notations such as those for synchronization or for logic operatons (AND, OR, …). FM provides the underlying draft upon which other cababilites and notations can be superimposed, including the next step: safety considerations.

6. Consider the safety of operating personnel and make additions and adjustments as needed

7. Add the master stop switches required for a safe shutdown- Similarly, switches can be modeled in FM and integrated into the FM depiction.

8. Create a ladder logic diagram that will be used as a basis for the PLC program.

FM can provide a more precise description of the total system, to be used in such diagramming methods as the ladder logic diagram. The process of converting FM schemata into such a diagram is beyond the scope of this paper and will be investigated as a future research objective.

9. CONCLUSION

Several diagrammatic techniques, e.g., SDL, statecharts, and UML, have been utilized to represent the static and dynamic aspects of control systems. This paper addresses the issue by providing schemata to be used in the conceptual modeling of production control systems. A flow-based specification, denoted FM, that has been utilized in software engineering is proposed as a good vehicle in this area.

The methodology is demonstrated by recasting two specific research projects involving actual diagrammatic representations in the field. In the first project, UML diagrams are used as a visual programming language to describe a modeling approach to autonomous production agents used in production control systems. The second project introduces a modeling methodology that uses multidiagram techniques to generate PLC code, providing an additional opportunity to compare a methodology with FM in the context of programmable logic controllers.

The resultant conceptual specification demonstrates that the methodology is capable of modeling various types of processes. It can be applied uniformly at different levels of detail to provide a central description analogous to complex engineering diagrams, e.g., electrical and mechanical schemata. Additionally, it supports the flow-down of requirements from initial planning to a user's view of finished automated systems, assuming an underpinning framework that is applied at each level down the left-hand side of a known V-lifecycle, from the whole system to its subsystems.

Contrasting the resulting FM description with the diagrammatic specification in these two projects shows that FM achieves uniformity with a single flow system used to represent all production system elements. At this level of

specification, the FM representation seems to provide continuity in the description of an episodic sequence of events, linking these events by flows and triggering. The movement of flowthings in FM flowsystems is analogous to the flow of water in a river, running from its origin and across boundaries without any "dry" sections along its stream.

Other representations such as UML fail to furnish a nucleus around which the various phases of a development process can evolve. In contrast to a UML-based approach of multifaceted textual and graphical descriptions, the contention of this paper is that the very nature of modeling demands a "master representation" around which global features can be identified and designed, analogous to the role of a graphical description (blueprint) in the construction of a high-rise building, where the drawing is the core around which functional features such as the structure itself, an electrical system, a water system, and interior furnishings are built by their various specialists, in addition to representation of global features such as aesthetics and themes, cost, and security.

Any methodology of representation has its own advantages and weaknesses, especially in regard to the features of understandability and simplicity. The FM representation is based on the notion of flow and characterized by uniform application of the basic structure of a flow system. One possible weakness of FM is the potential size and scope of the resulting picture, which may be too much to follow and control if all flows are shown along with diverse types of triggering; however, this drawback seems insignificant in light of large schemata of high-rise buildings and multifaceted specifications of airplanes and similar huge technical projects. Only further experimentation with FM would uncover potential advantages and difficulties. Another weakness of FM is that it is still in the exploratory phase of development compared with a modeling methodology such as UML which has a well-established body of research and investment. Nevertheless, FM seems to present a promising direction and to make a viable contribution to the area of system specification.

Further research is needed to apply the FM methodology directly to production control systems. Also, additional investigation is needed to develop tools and supporting apparatus in this direction. Specifically, additional synchronization, constraints, and logical notation need to be superimposed on the base FM description. These can be imported from other schematics such as Petri net diagrams.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] Object Management Group, *OMG Unified Modeling Language Specification, Version 1.5*, March 2003. http://www.omg.org/spec/UML/1.5/

[2] Object Management Group, *OMG Unified Modeling Language Specification, Version 2.4.1*, August 2011. http://www.omg.org/spec/UML/2.4.1/

[3] S. Lu and J. Parsons, "Enforcing ontological rules in UML-based conceptual modeling: Principles and implementation," *Proc. of 10th Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'05)*, pp. 451–462. Porto, Portugal: FEUP, 2005.

[4] U. Nickel, J. Niere, W. Schäfer, and A. Zündorf, "Combining statecharts and collaboration diagrams for the development of production control systems," *Proc. of Object-Oriented Modelling of Embedded Real-time Systems (OMER) Workshop, Technical Report 1999-01*, University of Armed Force München, May 1999.

[5] J. Gausemeier, H.-J. Buxbaum, S. Förste, and G. Gehnen, "Decentral control architecture for modular flow systems," *Proc. CAD/CAM Robotics and Factories of the Future*, London, 14–16 August 1996.

[6] International Telecommunication Union (ITU), *Z.100 - Specification and Description Language - Overview of SDL-2010*, 2012. http://www.itu.int/rec/T-REC-Z.100-201112-I

[7] International Telecommunication Union (ITU), *Z.101 - Specification and Description Language, Basic SDL-2010*, 2012. http://www.itu.int/rec/T-REC-Z.101-201112-I

[8] H. Köhler, U. Nickel, J. Niere, and A. Zündorf, "Integrating UML diagrams for production control systems," *Proc. of the 22nd International Conf. Software Engineering (ICSE), Limerick*, Ireland, pp. 241-251. ACM Press, 2000.

[9] H. J. Köhler, U. Nickel, J. Niere, and A. Zündorf, "Using UML as a visual programming language," Technical Report tr-ri-99-205. University of Paderborn, 1999.

[10] J. Niere and A. Zündorf, "Using Fujaba for the development of production control systems," *Proc. of Int. Workshop and Symposium on Applications of Graph Transformations with Industrial Relevance (AGTIVE), Kerkrade*, Netherlands, LNCS. Springer Verlag, 1999.

[11] F. Chi Kit, "*Design and analysis of agent-based FMS control systems*," PhD thesis, Department of Industrial and Manufacturing Systems Engineering, University of Hong Kong, 2005.

[12] J. Zhu and A. Mostafavi, "Towards a new paradigm for management of complex engineering projects: a system-of-systems framework," *8th Annual IEEE Systems Conference (SysCon)*, March 31-April 3, 2014.

[13] G. Finance, *SysML modelling language explained*. http://www.omgsysml.org/SysML_Modelling_Language_explained-finance.pdf

[14] Y. Xu, G. Zhang, and F. Wang, "The flow simulation in the fluidic amplifier," *The Open Automation and Control Systems Journal*, vol. 3, pp. 8-12, 2011.

[15] J. Zhi-yang, L. Zhong-xing, J. Hong, S. Li-qin, and X. Wen, "The software design of natural rubber plantation temperature and humidity monitoring system based on ZigBee," *The Open Automation and Control Systems Journal*, vol. 6, pp. 9-16, 2014.

[16] F. O. Hansen, *SysML – a modeling language for systems engineering.* 2010. [slides] http://staff.iha.dk/foh/Foredrag/SysML-SystemEngineering-DSFD-15-03-2010.pdf

[17] H. Park, J. Kwak, G. Wang, and S. Park, "A modeling methodology for process control in the automated manufacturing system," *Summer Computer Simulation Conf.*, 2010, pp. 439-445.

[18] M. Fowler, *UML Distilled: Brief Guide to the Standard*, 3rd ed. Addison-Wesley, USA, 2003.

[19] D. Harel and E. Gery, "Executable object modeling with statecharts," *Proc. 18th Int. Conf. Software Engineering (ICSE '18)*, Berlin, pp. 246-257, 1996.

[20] D. Balasubramanian, C. S. Pa sa reanu, M. W. Whalen, G. Karsai, and M. R. Lowry, "Polyglot: Modeling and analysis for multiple statechart formalisms," *ISSTA*, pp. 45–55, 2011.

[21] ILogix, *The Rhapsody Case Tool Reference Manual; Version 1.2.1*. http://www.ilogix.com/

[22] The Rational-Rose Realtime Case-Tool, http://www.rational.com

[23] S. Al-Fedaghi, "A conceptual foundation for data loss prevention," *International Journal of Digital Content Technology and its Applications*, vol. 5, no. 3, pp. 293-303, 2011.

[24] S. Al-Fedaghi, "States and conceptual modeling of software systems," *International Review on Computers and Software (IRECOS)*, vol. 4, no. 6, pp. 718-727, 2009.

[25] S. Al-Fedaghi, "System-based approach to software vulnerability," *IEEE Symposium on Privacy and Security Applications (PSA-10)*, Minneapolis, USA, 2010. ftp://ftp.computer.org/press/outgoing-/proceedings/SocialCom%202010/data/4211b072.pdf

[26] S. Al-Fedaghi, "Pure conceptualization of computer programming instructions," *International Journal of Advancements in Computing Technology (IJACT)*, [Accepted]

[27] S. Al-Fedaghi and F. Al-Shahin, "How to diagram a production control system", *The 2012 12th Int. Conf. Control, Automation, and Systems (ICCAS 2012)*, Jeju Island, Korea. 2012, pp. 1065-1070.

[28] S. Al-Fedaghi and F. Faihan, "A conceptual visualization of industrial control systems: Electrical power system," *International Review of Automatic Control (IREACO)*, [Accepted].

[29] S. Al-Fedaghi and A. Abdullah, "A new approach to component-based development of software architecture," *International Review on Computers and Software (IRECOS)*, vol. 8, no. 1, pp. 1-10, 2013.

[30] S. Al-Fedaghi, "Alternative representation of aspects," *10th IEEE Int. Conf. Information Technology: New Generations*, IEEE ITNG 2013, 15-17 April, Las Vegas, Nevada, USA.

[31] K. Roebock, *Product Lifecycle Management (PLM): High-impact Strategies - What You Need to Know*, Lightning Source Incorporated, 2011.

[32] C. Bock, "Systems engineering in the product lifecycle," *International Journal of Product Development*, vol. 2, no. pp. 1-2, 2005.

[33] SysML Partners, *Systems Modeling Language: SysML*, 2004. http://www.omg.org/syseng/SysML-Presentation-SEDSIG-040427.pdf

[34] J. Jang, P. H. Koo, and S. Y. Nof, "Application of design and control tools in a multirobot cell." *Computers and Industrial Engineering Journal*, vol. 32, no.1, pp. 89-100, 2010.

[35] C. P. Chuang, X. Lan, and J. C. Chen, "A systematic procedure for designing state combination circuits in PLCs," *Journal of Industrial Technology*, vol. 15, no. 3, pp. 2-5, 1999.

[36] M. Koa, S. Parka, J.-Ju Choib, and M. Chang, "New modeling formalism for control programs of flexible manufacturing systems," *International Journal of Production Research*, vol. 51, no. 6, pp. 1668-1679, 2013.

[37] IEC, *International Standard IEC 61131-3: Programmable Controllers*, second ed. International Electrotechnical Commission, 2003.

[38] E. A. da Silva Oliveira, L. D. da Silva, K. Gorgonio, A. Perkusich, and A. F Martins, "Obtaining formal models from ladder diagrams," *9th IEEE Int. Conf. Industrial Informatics (INDIN)*, pp. 796–801, 2011.

[39] C. M. Park, S.M. Bajimaya, S. C. Park, G. N. Wang, J. G. Kwak, K. H. Han, and M. Chang, "Development of virtual simulator for visual validation of PLC program," *Int. Conf. Intelligent Agent Web Technologies & Internet Commerce*, Australia, 2006.

[40] H. T. Park, J. G. Kwak, G. N. Wang, and S. C. Park, "Plant model generation for PLC simulation", *International Journal of Production Research*, vol. 48, no. 5, pp. 1517-1529, 2010.

[41] A. Anglani, A. Grieco, M. Pacella, and T. Tolio, "Object-oriented modeling and simulation of flexible manufacturing system: A rule-based procedure," *Simulation Modeling Practice and Theory*, vol. 10, no. 3-4, pp. 209-234, 2002.

[42] S. Park and J. S. Jang, "Virtual plant for control program verification," *Int. Conf. Circuits, System and Simulation (IPCSIT)*, vol. 7, 2011.

Received: August 02, 2014 Revised: December 06, 2014 Accepted: December 29, 2014