Flow-Based Specification of Time Design Requirements

Sabah Al-Fedaghi
Computer Engineering Department
Kuwait University
Kuwait

***Abstract*—This paper focuses on design requirements in real-time systems where information is processed to produce a response within a specified time. Nowadays, computer control applications embedded in chips have grown in significance in many aspects of human life. These systems need a high level of reliability to gain the trust of users. Ensuring correctness in the early stages of the design process is especially a major challenge in these systems. Faulty requirements lead to errors in the final product that have to be fixed later, often at a high cost. A crucial step in this process is modeling the intended system. This paper explores the potential of flow-based modeling in expressing design requirements in real-time systems that include time constraints and synchronization. The main emphasized problem is how to represent time. The objective is to assist real-time system requirement engineers, in an early state of the development, to express the timing behavior of the developed system. Several known examples are modeled and the results point to the viability of the flow-based representation in comparison with such time specifications as state-based and line-based modeling.**

Keywords—design requirements; conceptual model; time constraints; model-based systems engineering; requirements specification

I. Introduction

The product development life cycle in the engineering domain aims at achieving, among other goals, a design process with complete and precise specifications that satisfy all requirements. Requirements are descriptions of functions, features, and goals of the product. The requirements describe ‘what’ the intended product should do, but the ‘how’ is specified as design requirements during the design phase, where measurability and verifiability are of utmost importance. Design requirements (the focus of this paper) include the specifications that the intended product must meet in order to pass the acceptance test. Specifications consist of information that controls the creation of the intended product.

Early assurance of the correctness of design requirements is a major challenge in any system. Faulty design requirements lead to errors in the final product that have to be fixed later, often at a high cost. Reoccurring causes of failures include:

- Inadequate definitions and modifications of specifications
- Faulty interpretation and understanding
- Not meeting customer requirements
- Design not meeting manufacturing requirements
- Difficulties in specifying technical requirements
- Difficulties in interpreting and understanding specifications [1]

There are various methods for specifying real-time systems. For example, prototyping tools can be used by the designer and user to view the product in the development stage [2]. However, prototyping is a phase that comes after the specifications. If prototyping has produced unsatisfiable results, then the designer may have to re-specify the requirements. There are also formal specifications of real-time systems that should enable the system designer to verify mathematically that a system meets timing constraints. However, formal methods are still limited as a verification tool, especially for software systems, not to mention the complexity introduced by timing. Various specification languages for real-time systems with timing constraints can be expressed within the specifications (e.g., [3]), “but at the cost of restricting other features” [4].

The specifications of design requirements are usually formulated in a mixed of English, tables, graphs, screen shots, and unified modeling language (UML) diagrams. According to Palshikar [5], design requirements are examined in terms of:

- accurate reflection of the users’ requirements
- clarity, unambiguity, and understandability
- flexibility and feasibility for the engineers
- easily defined acceptance test cases
- an abstract and high-level manner of writing, away from design, implementation, and technology platforms

“Despite some help from modeling tools such as UML, the problem of ensuring the quality of requirements remains. The process is heavily manual and time-consuming, involving reviews and sometimes-partial prototyping. Using multiple notations (such as those in UML) introduces additional problems” [5].

Additionally, this paper is concerned with *design requirements* in *real-time systems* where information is processed to produce a response within a specified time. A real-time system interacts with the environment within certain timing constraints and the requirement specifications for such a system must include representation of timing which can guarantee meeting these constraints. The notion of *time* is an important element in such systems, especially if critical features (e.g., safety) are functionally required. The problems here are how to represent time, how to capture causality behavior, and how to integrate functional and timing activities [6].

Embedded systems where the software is completely encapsulated by the hardware that it controls are often real-time systems. An embedded system is a system that interacts continuously with its physical sphere via sensors and actuators. Nowadays, computer control applications embedded in chips have grown in significance in many aspects of human life (e.g., medicine, mobile phones, and vending machines). These embedded systems need a high level of reliability to gain the trust of users. Ensuring correctness in the early stage of the design process is especially a major challenge in these systems.

A crucial step in this process is modeling the intended system. Model-based design has been introduced as the method to deal with the design process where the requirements are specified in a systematic way before continuing with the design and implementation phases. A great deal of attention has focused on this, such as interest in the unified modeling language with its graphical notation, which is used for documentation, communication, and requirement capture, as well as being an abstraction base for implementation details. This paper explores the potential of the flow-based modeling [7–12] in expressing design requirements in real-time systems that include time constraints and synchronization.

This paper focuses on the *representation* of timing constraints. The objective is to assist real-time system requirement engineers, at an early state of the development, to express the timing behavior of the developed system. *Representation* here refers to humans' and machines' representation of knowledge for the purpose of communication and understanding and analyzing the embedded semantics (e.g., diagrams, formal notations). Representation is usually associated with reasoning (e.g., the computational understanding of human-level cognitive abilities). This concentrates on the representation aspect that can be used for manual or computation analysis, as in problem solving in artificial intelligence.

In preparation to recast the representation of several known design problems in terms of flow-based modeling, and to make this paper self-contained, the next section briefly reviews published materials describing the flow-based model. Several features of the model will be further illustrated.

II. FLOWTHING MODEL

The flowthing model (FM) is a uniform method for representing "things that flow," called *flowthings.* Flow in FM refers to the exclusive (i.e., being in one and only one) transformation among six *states* (also called stages): transfer (input/output), process, creation, release, arrival, and acceptance, as shown in Fig. 1. We will use *receive* as a combined stage of *arrive* and *accept* whenever arriving flowthings are always accepted.

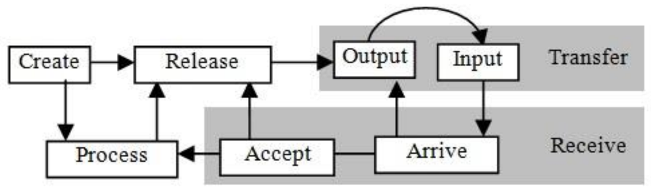


Fig. 1. Flowsystem

A flowthing has the capability of being created, released, transferred, arrived, accepted, and processed while flowing within and between "units" called *spheres*. A flow system (referred to as *flowsystem*) is a system with six stages and transformations (edges) between them. In FM, flows can be controlled by the progress (sequence) of the stream of events (creation, release, transfer, transfer within the next sphere, release, reception, …) or by a triggering (denoted by a dashed arrow) that can initiate a new flow. Spheres and subspheres are the environments of the flowthing, such as a company, a computer, and a person. A sphere can include the sphere of a flowsystem that includes the transfer stage. Triggering is the transformation from one flow to another, e.g., a flow of electricity triggers a flow of air.

Example: In studying a "successful" model checking for verifying requirements, Palshikar [5] used a simple pumping control system that transfers water from a source tank A into sink tank B using a pump P as shown in Fig. 2. Each tank has two level-meters to detect whether their levels are empty or full. The tank level is ok if it is neither empty nor full.

Initially, both tanks are empty. The pump is to be switched on as soon as the water level in tank A reaches ok (from empty), provided that tank B is not full. The pump remains turned on as long as tank A is not empty and as long as tank B is not full. The pump is to be switched off as soon as either tank A becomes empty or tank B becomes full. The system should not attempt to switch the pump off (on) if it is already off (on). [5]

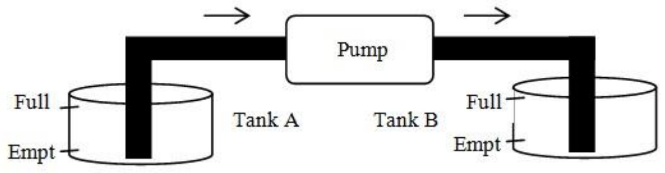
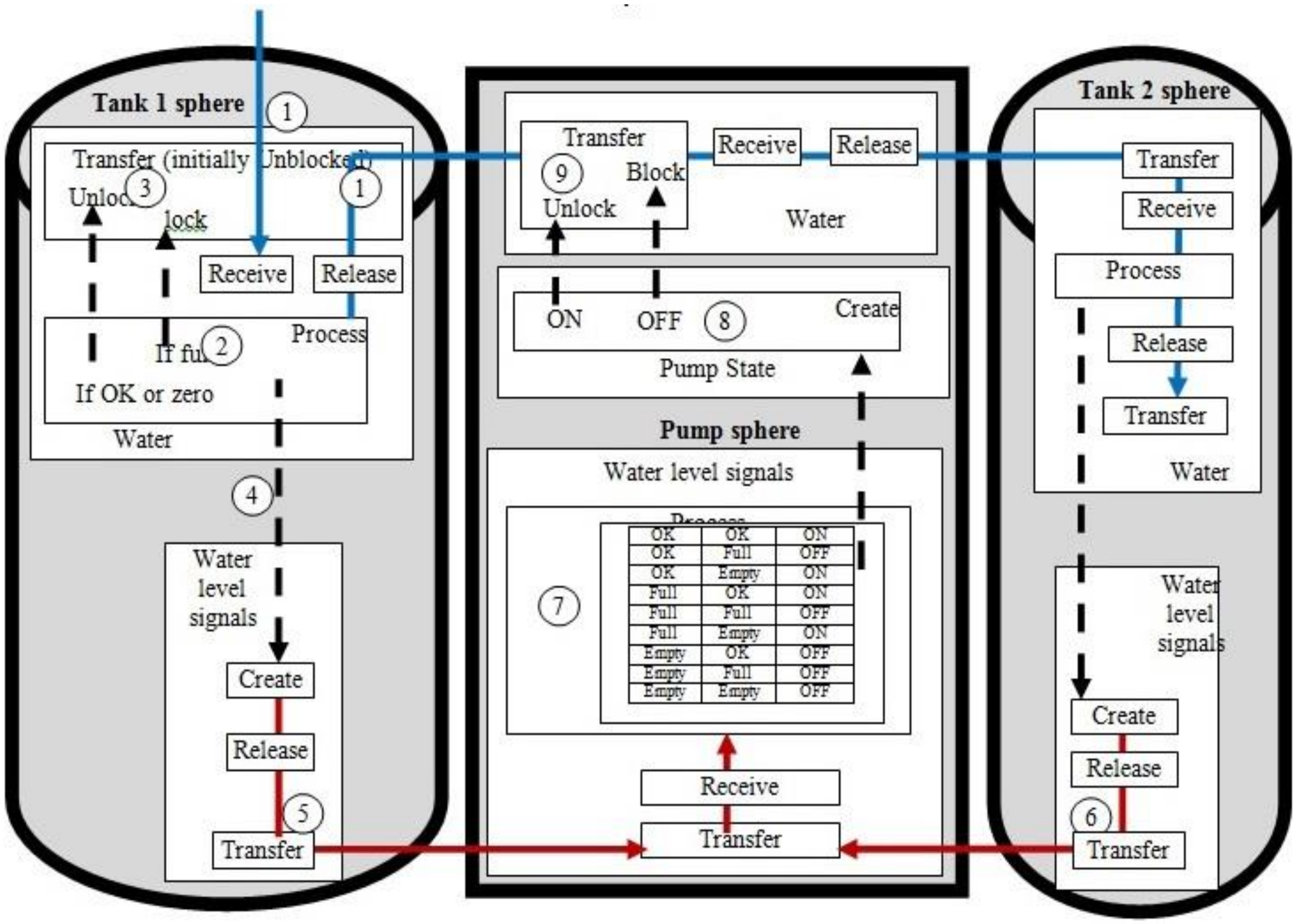


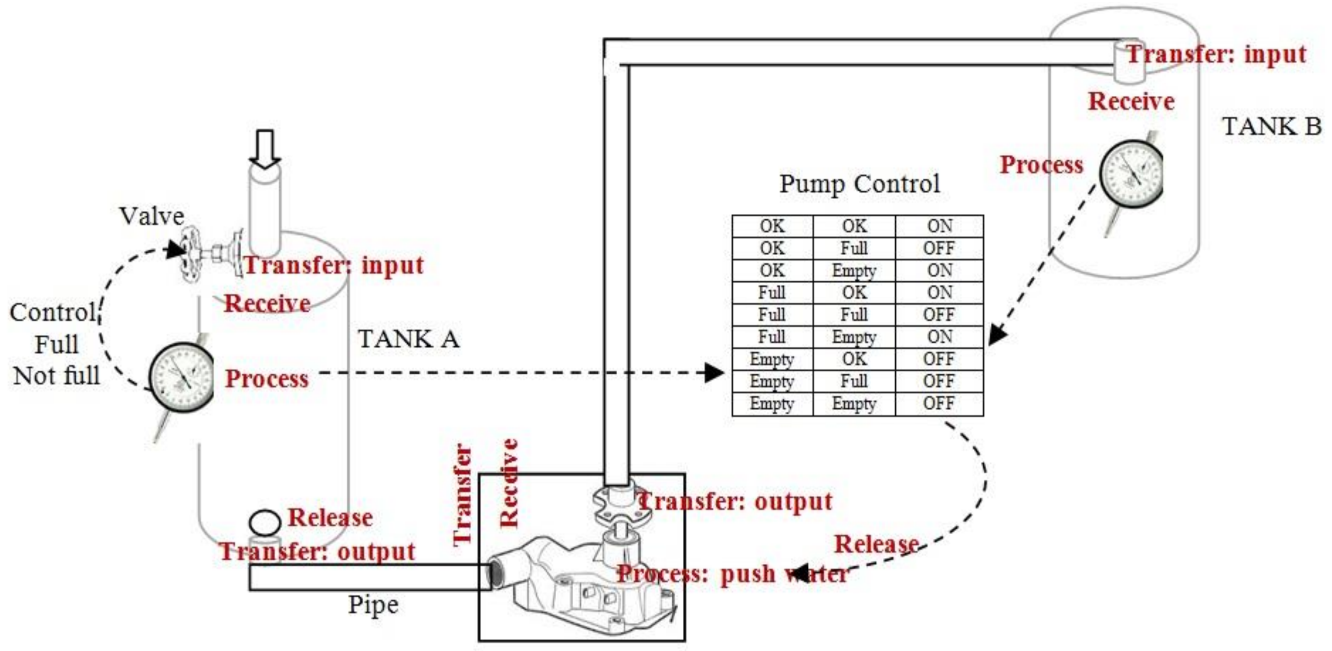
Fig. 2. A simple pumping control system (redrawn from [5])

Finite state machine (FSM) approach is utilized as an abstract notation for defining requirements and design. Fig. 3 shows the FM representation of this pumping control system. It impedes some assumptions which are illustrated in Fig. 4.



OK	OK	ON
OK	Full	OFF
OK	Empty	ON
Full	OK	ON
Full	Full	OFF
Full	Empty	ON
Empty	OK	OFF
Empty	Full	OFF
Empty	Empty	OFF

Fig. 3. FM representation of the pumping control system



OK	OK	ON
OK	Full	OFF
OK	Empty	ON
Full	OK	ON
Full	Full	OFF
Full	Empty	ON
Empty	OK	OFF
Empty	Full	OFF
Empty	Empty	OFF

Fig. 4. Illustration of assumptions in FM representation

In Fig. 4, it is assumed that water flows in tank A with the transfer of this flow controlled within that tank system. The water does not flow to tank B (and therefore tank B is drawn above tank A). Accordingly, the pump is installed between the two tanks to push the water toward tank B.

Tank A is a flowsystem with transfer, receive, process, and release stages. The transfer stage is drawn twice to simplify the drawing. The gate valve controls the *transfer* to tank A. As soon as the valve is opened the water is *received* in the tank. The *process* in tank A involves measuring the amount of water and, accordingly, the valve is opened or closed. At the bottom of tank A there is no control, hence *release* and *transfer* to the pump is immediate. This is analogous to passengers that proceed immediately to a waiting airplane after finishing passport processing. Imagine that this passport checking is on one end of the boarding bridge while the airplane is at the other end of the bridge. In this case, the bridge would correlate to the *release* stage as part of the airport system. Moving from the bridge end to the airplane door would be a flow between two *transfer* stages. Accordingly, in tank A's control system, the flow from the tank to the pipe (see the figure) leading to the pump is a flow between two *transfer* stages. It is possible to include each pipe in Fig. 3 as a flowsystem with transfer, release, and transfer stages. However, this is not shown in the Fig. 3.

The pump is similarly a flowsystem. The process stage involves pushing/not-pushing the water toward tank B. There is no need for valves because the water cannot flow to tank B without pushing.

There seems to be incompleteness in Palshikar [5]'s original description of this system in a case where both tanks are full. In this case, the valve to tank A is closed and the pump is off forever. Accordingly, an outlet has been added in the flowsystem in tank B.

Switching the description to Fig. 3, the water flows in (circle 1 in the figure) to be processed (circle 2, measuring its water level) and accordingly opens or closes the valve (3). Also, the processing triggers (4) the control flowsystem of tank A to send a signal (5) about the *current* level of water: empty, okay, or full to the pump control system. On the other hand, tank B also sends (6) such a signal. These signals are processed in the pump control flowsystem (7) to turn on/off the pump (8), which results in the stoppage or flow of the water to tank B (8).

The next section applies FM to the method known time representations in order to compare the two methods side by side.

III. Time and FM

Time requirements play a central role in understanding and designing systems. Timing is typically incorporated after tasks and software architectures are defined, when holistic scheduling algorithms and expected worst-case execution times are analyzed [13]. This paper does not involve such a detailed level of description; rather, it is concerned with a very high level of requirements specifications, e.g., the level of UML use-case, sequence, and activity diagrams. Accordingly, this section relates time to its representation in FM.

Philosophically, time can be conceptualized as a fourth-dimensional phenomenon. Such a conceptualization is inspired by Edwin Abbott's *Flatland*:

Dr. Abbott pictures intelligent beings whose whole experience is confined to a plane, or other space of two dimensions, who have no faculties by which they can become conscious of anything outside that space and no means of moving off the surface on which they live. He then asks the reader, who has consciousness of the third dimension, to imagine a sphere descending upon the plane of Flatland and passing through it. How will the inhabitants regard this phenomenon? […]

Their experience will be that of a circular obstacle gradually expanding or growing, and then contracting, and they will attribute to *growth in time* what the external observer in three dimensions assigns to motion in the third dimension. If there is motion of our three-dimensional space relative to the fourth dimension, all the changes we experience and assign to the *flow of time* will be due simply to this movement, the whole of the future as well as the past always existing in the fourth dimension. (Italics added.) [14]

The sphere (ball) is seen as constantly changing, and the whole change from birth to disappearance is the "lifetime" of the sphere. Applying the 3-dimensional world, this time must then be a 4th dimension.

Strachan [15]'s conceptualization of the same phenomenon is as follows:

Let's imagine a miniature world which is a cube. Now suppose that one of the faces of the cube—say the bottom face—is a little 2-dimensional world, a Flatland, inhabited by creatures called 'Toodies' (2-D) . . .

Since the Toodies' Flatland is infinitely thin . . . , then an infinite number of Flatlands could be stacked into the cube . . .

But let us now suppose that a Toody is subjected to some force which can lift him up the 3rd (up and down) dimension of the cube. So he is propelled out of his own paper-thin world, the bottom face of the cube, right up through the cube to its top face. As he does so, he will pass through all the 2-dimensional 'paper' Flatlands which lie in between. Since the whole cube exists, then all of these Flatlands exist, even though they won't exist for Toody until he reaches them. So they lie in Toody's future.

But change occurs, and can only occur, in time. So his movement in this 3rd (up/down) space dimension will seem like *the passage of time* to Toody: it is his time dimension. (Italics added.)

A. *Time as spheres*

Accordingly, from the FM point of view, these "flatlands" are *flowthings that flow through times spheres*: past1, past2, . . . , now, future1, future 2, . . . In this case, time is modeled as spheres. All of these spheres are projections of different times on flatlands. UML representation of this modeling of time is shown in Fig. 5, which includes slices of time with processes happening in them. Fig. 6 shows the corresponding FM representation.

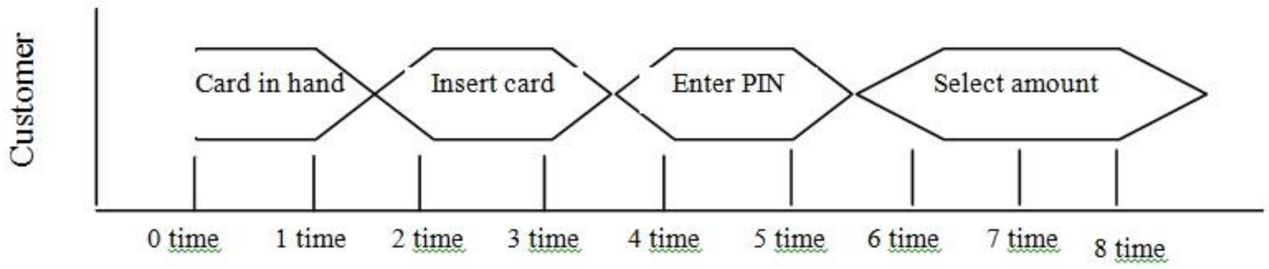


Fig. 5. Sample of UML representation of time (from [16])

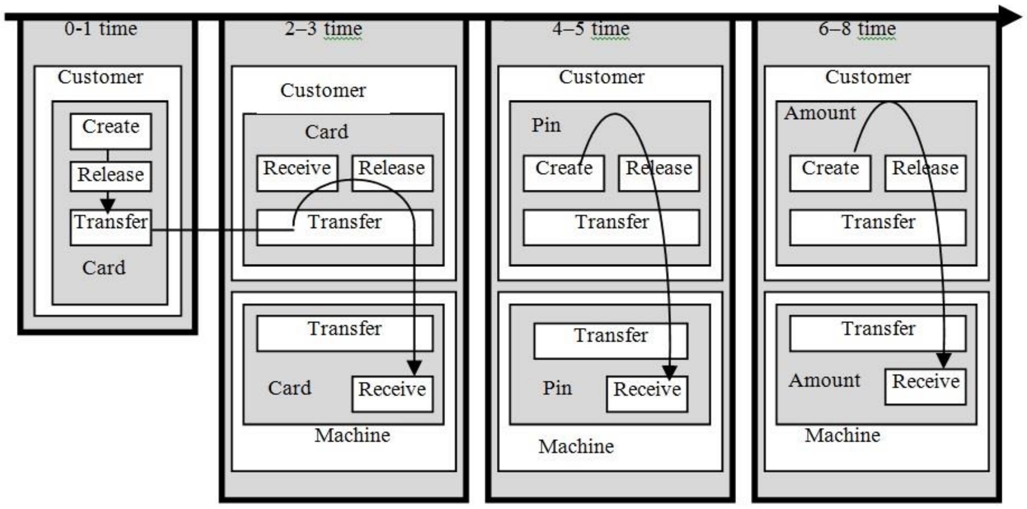


Fig. 6. Time spheres with "flatlands" flow through them

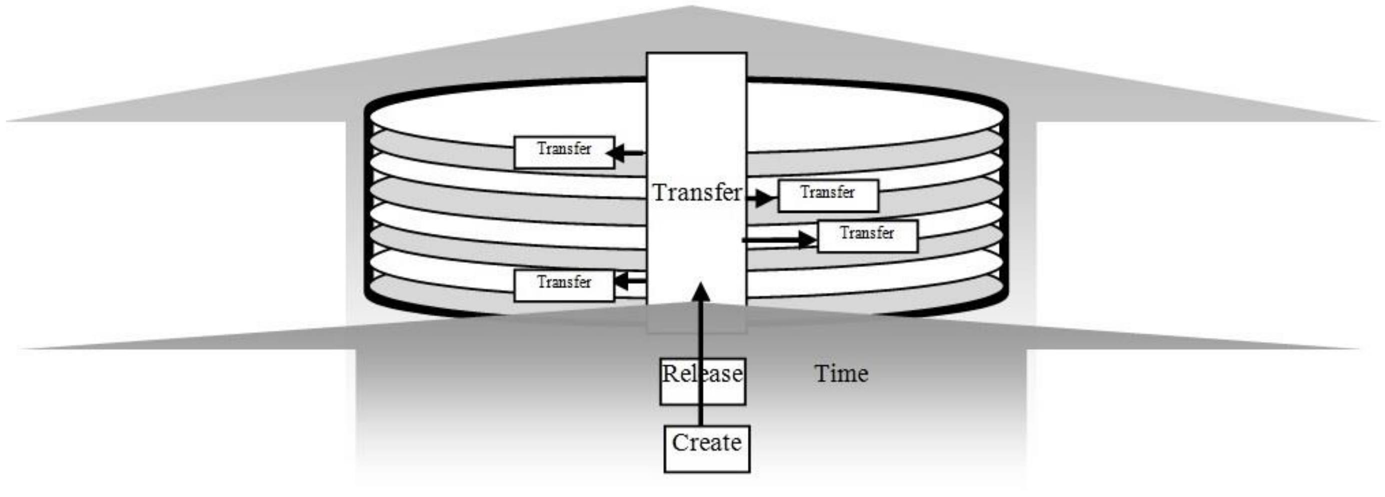


Fig. 7. Cylinder is used instead of a cube to illustrate time flows through "Flatlands"

B. *Time as flowthings*

Alternatively, time can be conceptualized as a *flowthing* that flows through "flatlands." In this case, Strachan [15]' s cube (though we prefer to use a cylinder instead of a cube; see Fig. 7) passes through all the 2-dimensional Flatlands, accomplishing the same result.In this case, time in FM is a flowthing that can be released, transferred, received, and processed. For each flowsystem, it is processed to count its passing though counting, as will be illustrated in the next section. In FM, time is something that flows contiguously from a fourth-dimension sphere to any other sphere, as shown in Fig. 8. If this is of relevance to flows or triggering in that sphere, it is represented by a flowsystem. This conceptualization of time as a flowthing will be utilized in the discussions in the next sections.

IV. Linear Time Diagrams

Timing diagrams "focus on conditions changing within and among lifelines along a linear time axis … on time of events causing changes in the modeled conditions of the lifelines" [17]. They utilize the notions of lifeline, state or condition timeline, destruction event, duration constraint, and time constraint. Timelines are one of the simplest means of representing the flow of events. In UML 2, timing diagrams are a special form of sequence diagrams where the axes are reversed and the lifelines are shown in separate compartments arranged vertically. These diagrams "aren't the most popular" [18].

According to the Web site [17], time duration constraint refers to the duration used to determine whether the constraint is satisfied. It is an association between a duration interval and the constructs that it constrains. For example, that ice should melt into water in 1 to 6 minutes can be represented as shown in Fig. 9. From the conceptual point of view, lining (putting in one category) ice, melting, and water is a categorical mix. Ice and water can be categorized as "states" of H_2O, but melting is certainly not. Also, it seems that H_2O is another name for water. Fig. 10 shows the FM representation.

There are three subspheres: time, ice, and water. The units of time are continuously received (1) and ignored. They are processed (2) as soon as the melting (a kind of process (3)) starts in the ice sphere until "counting" 6 units of time. When the ice starts melting (3), it triggers (4) the counting (processing (2)) of time. When the melting ends (5), the time is ignored again (6) and water is generated (7). The model reflects that time *always* flows through systems, and thus time constraint is awareness of this flow and alignment of events with the flowing time.

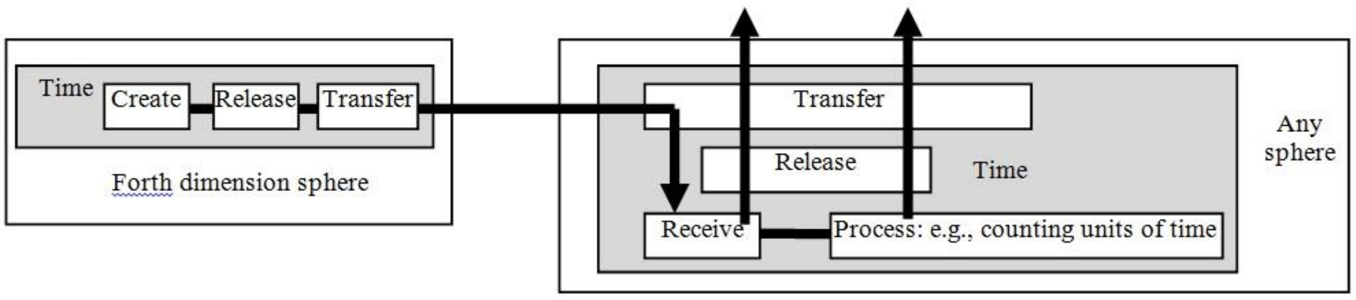


Fig. 8. Time conceptualization FM representation

In addition, a time constraint is time expression used to determine whether the constraint is satisfied. "All traces where the constraints are violated are negative traces, i.e., if they occur, the system is considered as failed" [17]. Fig. 11 is given as a representation of this constraint. It involves two states: sleep and awake. At the change from sleep to awake, the time period {5:40 a.m., 6 a.m.} passes to accomplish this change. The state (sleep or awake) is represented by a horizontal line: no line, no state. The change from a state to another is represented by a vertical line that connects the horizontal lines. The delay that corresponds to the change is represented by the diagonal line and the text {5:40 a.m., 6 a.m.} at the point of beginning the awake state.

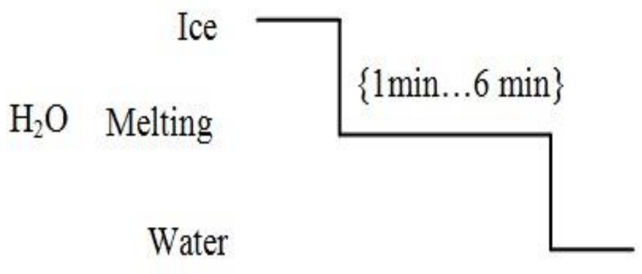


Fig. 9. Representation of how ice should melt into water in 1 to 6 minutes (from [17])

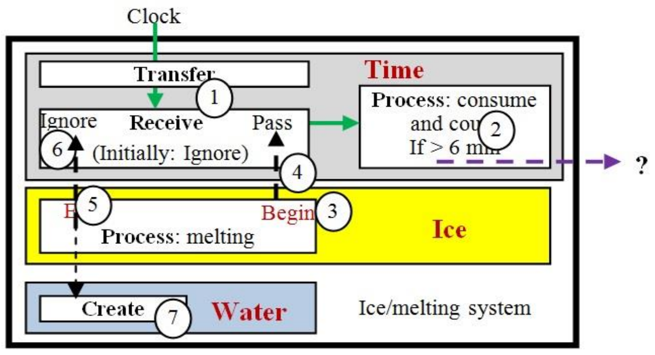


Fig. 10. FM representation

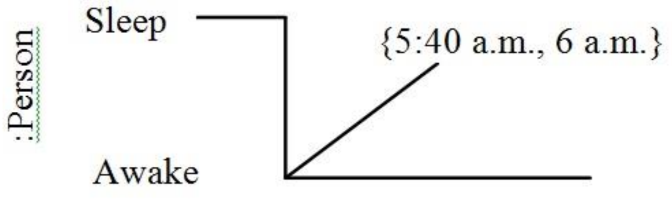


Fig. 11. Person should wake up between 5:40 a.m. and 6 a.m

Semantically, {5:40 a.m., 6 a.m.} is the "length" of sleep. Accordingly, the diagonal line and {5:40 a.m., 6 a.m.} look like a comment and not a modeling of the situation. If it is not a comment, then the representation is misleading because it gives the impression of a three-dimensional representation. Also, there is no indication of “failure” as mentioned in the given constraint. This example shows the limitations of the line representation of time.

Fig. 12 shows the corresponding FM representation. These are the spheres: time, sleep, awake, and the logical join. The clock performs the following:

- At 5:40 a.m., it triggers sleeping
- At 6:00 a.m., it triggers awaking
- At 6:00 a.m. it also triggers checking whether the awaking occurs

Time is generated by the clock and received by the sphere of the time in the system (circle 1). This sphere is the part of the total system that deals with time. The clock sends continuous signals, say 12:00, 12:01, 12:02, . . . , and these data arrive and are received and processed (2). This processing involves the recognition of 5:40 a.m. and 6:00 a.m. If it is 5:40 a.m. then this triggers the person to enter into sleeping (3). He/she is processed (absorbed) into sleeping (4). If it is 6:00 a.m., then this triggers:

- The release (5) of the person from sleeping to awaking (6)
- The checking of whether the person has arrived to the awaking state (7). If this is the case then this triggers success (For simplification sake, success is reported instead of failure; accordingly, the recipient of the report assumes failure if success does not arrive.)

Note that the horizontal joint bar can be represented in FM as shown in Fig. 13.

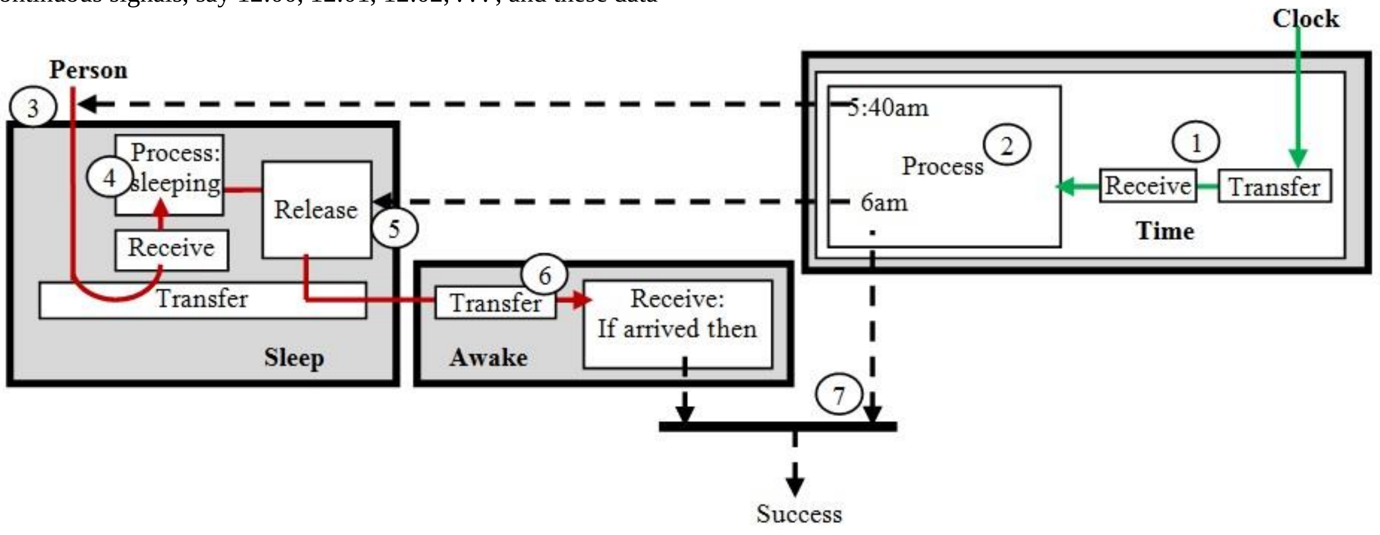


Fig. 12. FM representation

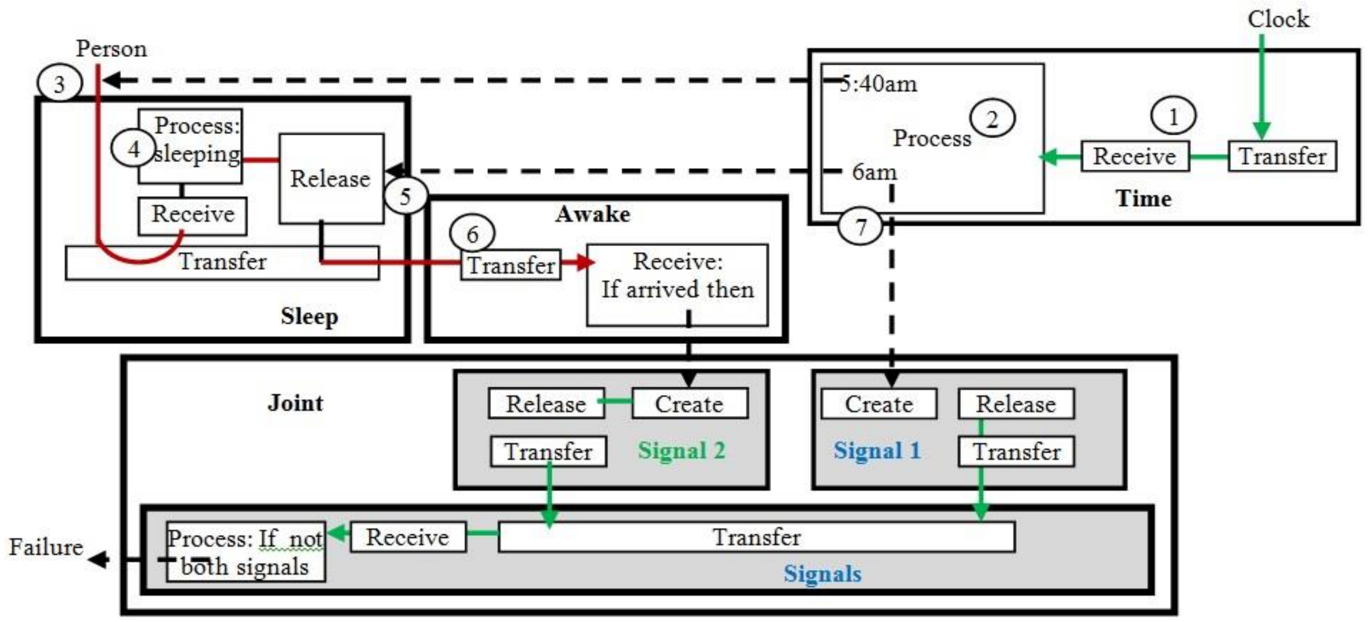


Fig. 13. FM representation without the joint bar

V. Timing and Real-Time Systems

Coffee machines have been used as a well-known example of modeling real-time systems using such languages as Uppaal and UML (e.g., [19–23]) In this section, we investigate the specification of design requirements for the coffee machine problem in the context of Uppaal, as it is described in many publications and course materials.

The coffee machine problem involves modeling the behavior of a system with three elements: a *machine*, *person*, and an *observer*. The person repeatedly inserts coins to receive coffee, after which he/she produces a publication. There is time delay after each such action. The machine takes some time for brewing the coffee.

It also takes a timeout if the brewed coffee is not taken before a certain upper time limit. The observer complains if more than 8 time units elapse between two consecutive publications.

In modeling the coffee machine in FM, we find that to complete the conceptual picture and flows, we need additional items (spheres) in addition to person, machine, and observer. One interesting aspect of FM description is the systematic application of the same generic stages for entities, subentities, and spheres. This repeatability of application creates specifications that are more complete. It is also possible to simplify the depiction by reducing the level of description in several ways. As an introduction, before giving the complete FM representation, Fig. 14 shows a brief description of the "waves" of flow and the new additional spheres.

In the figure, coin flow (A) triggers the coffee (B) and cup (C—a new sphere with an important role that will be explained later) flows as well as the flow of "counted time units" (D). As was mentioned previously, time flows continuously, but it is ignored until certain events (e.g., the arrival of coins) trigger counting units of time. Accordingly in the figure with the passing of the coffee preparation period, the coffee and the cup flow to the "filled cup compartment" (E) and start the "fill cup" flow (F). In this case, time is also counted (G), and if the filled cup does not flow (i.e., it is taken from the compartment), then this triggers dispensing. The flow of the filled cup outside the compartment (H) is supposed to trigger the flow of the coffee to the person (I—e.g., being drank). This in turn triggers producing publications (J) that flow to the observer (K). Upon the arrival of publications the observer starts counting time (L) and complains start to flow out (M) if time reaches its maximum without receiving new publications.

The completeness and continuity of events (technical and physical) are grounds for the validity of the model. Take the state-based modeling of the machine as given by Anderson [19] and represented in Fig. 15, according to Anderson [19–20]:

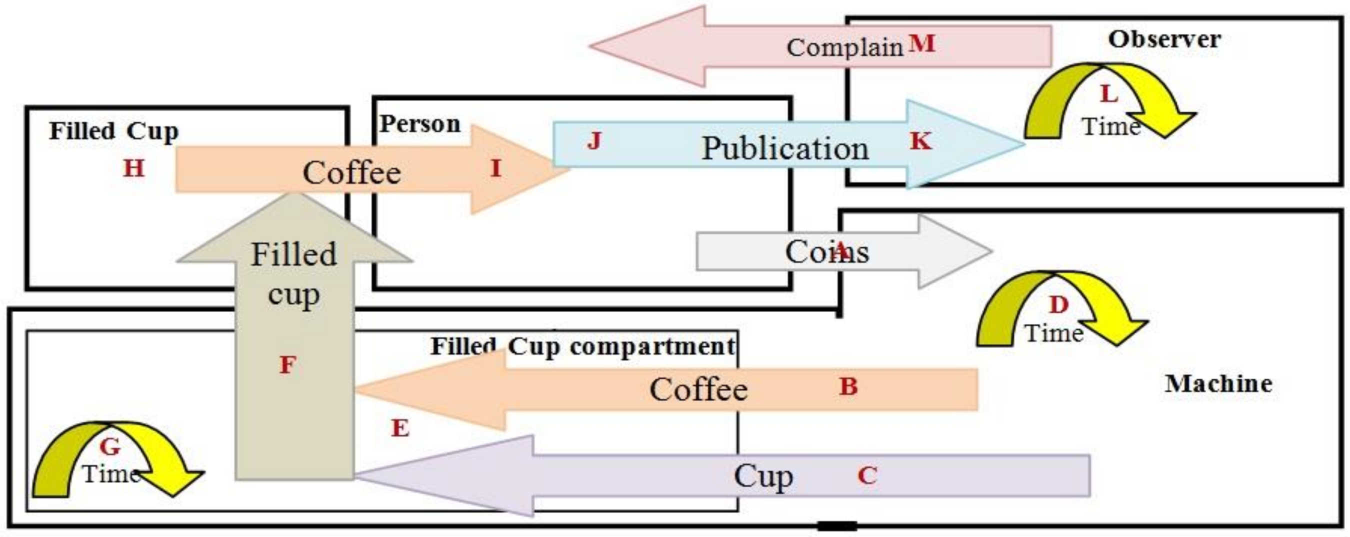


Fig. 14. Flows in the coffee machine problem

Coffee machine accepts a coin and then delays for some time (above it is 6 time units). It then sets a timeout timer, and either (to the right) dispenses coffee, or (to the left) times out and then dispenses coffee. The extra state on the left is because Uppaal does not allow both guards and synchronizing elements to appear on the same transition.

Note that this model assumes that the coffee *flows* outside the machine immediately, after the brewing process, just as water flows outside a pipe. This means the coffee does not *wait* to be taken outside the machine. The flow-based FM representation (see Fig. 16) forces introduction of a container for the coffee because there is waiting time. Thus, the items of *cup* and *filled cup* (cup+coffee) are necessary to convert the flowthing coffee from the state of liquidity (which makes its flow outside the machine compulsory) to the state of "handle-ability" (a thing that stands by itself waiting to be picked up). From the "state" perspective, Fig. 17 shows the two methods of conceptualization. On the left, the model is not based on flows, hence the conceptualization is represented by conceptual jumps from one state to another. On the right side, the FM model is casted in state jumps. In the figure, the two triggering arrows that come from outside the machine sphere change the waiting state.

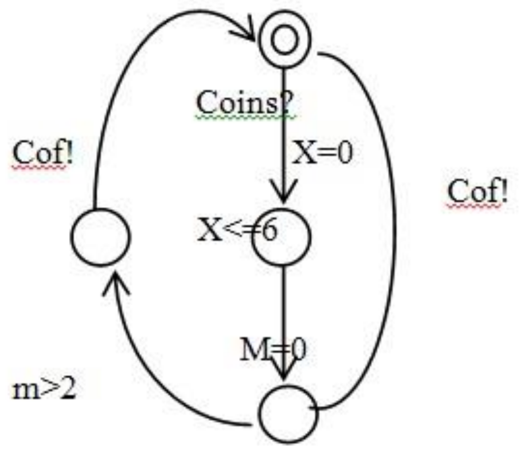


Fig. 15. Automata for machine (redrawn from [19])

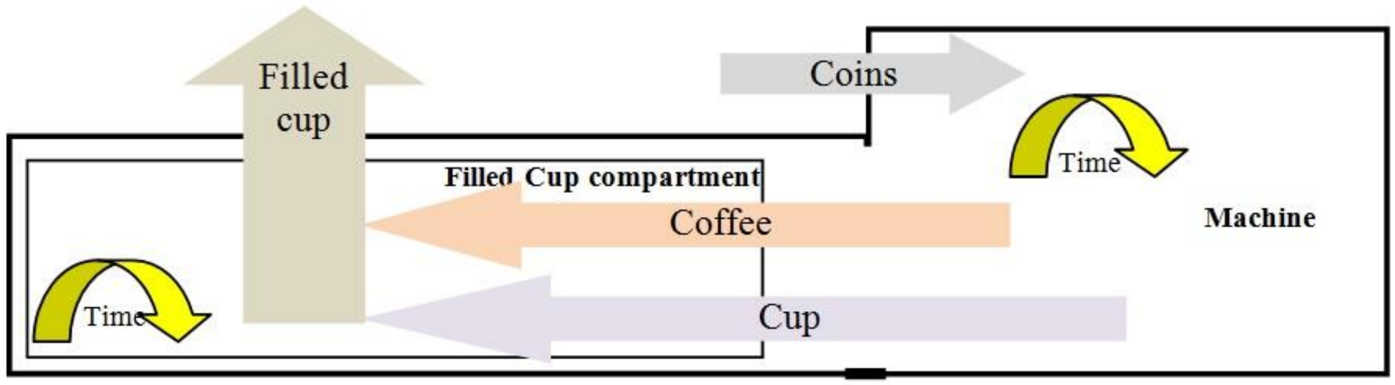


Fig. 16. Flows in the coffee machine problem

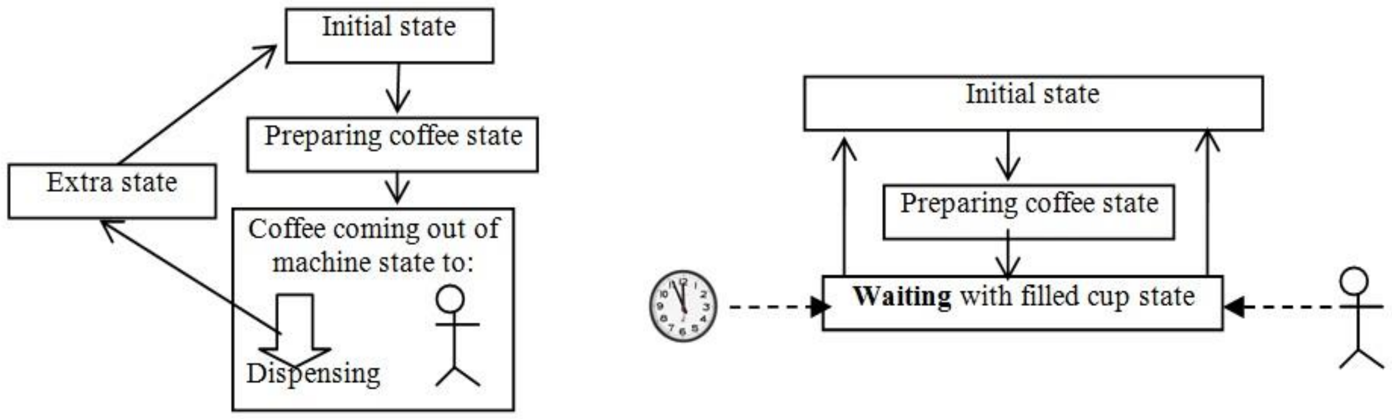


Fig. 17. The coffee problem described in terms of states

The point here is that the flow-based conceptualization "forces" continuity and completeness of the narration of events, thus identifying items (e.g., cups) and processes (e.g., waiting liquid).

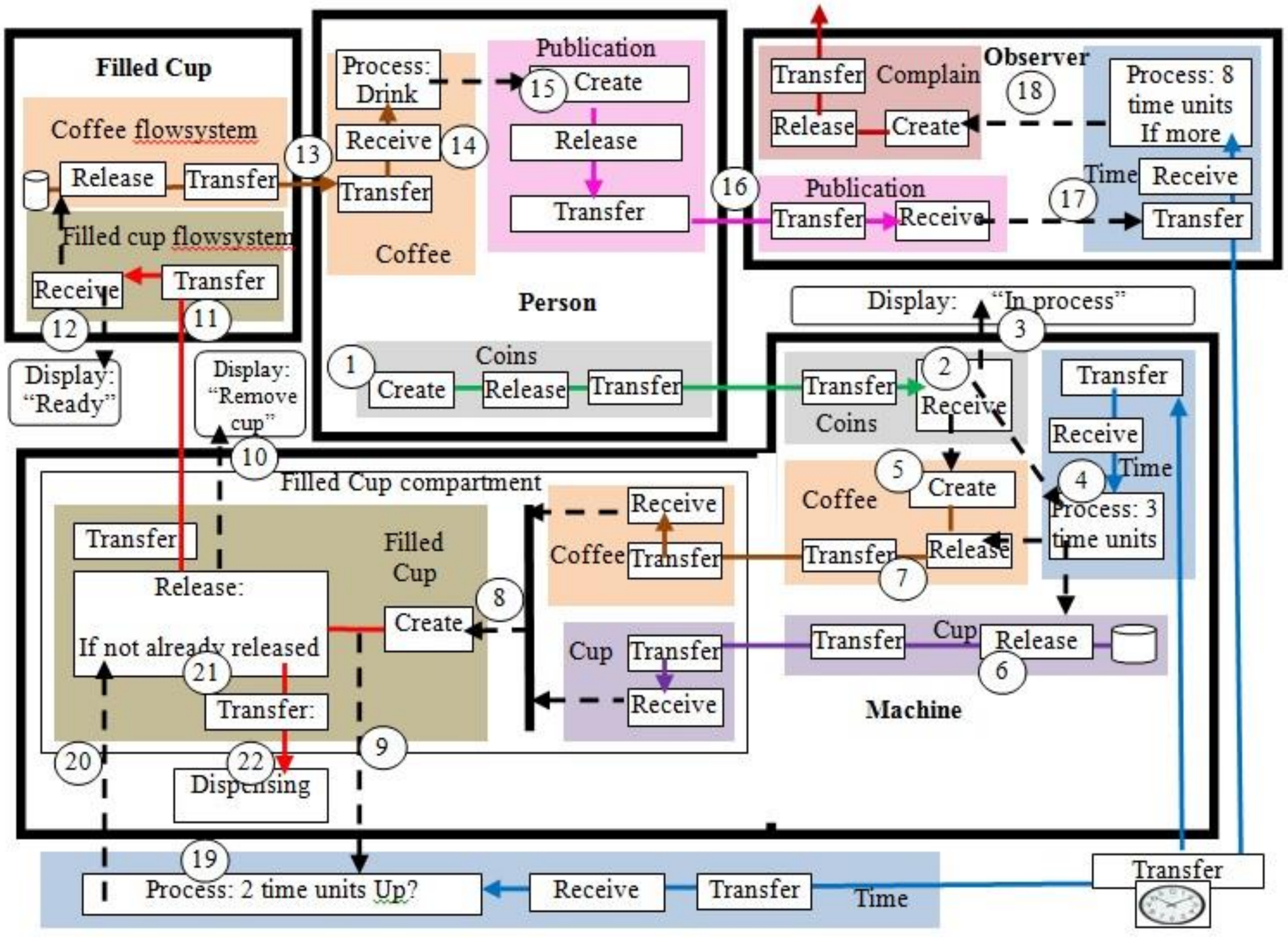


Fig. 18. The problem described in terms of states

Fig. 18 shows the complete FM representation of the coffee machine problem. We start with inserting the coins (circle 1). The *creation* here (in the figure) means the appearance of coins in the episode, just as the first appearance of a new character in a play in theater.

The coins flow to the machine (2) where they are received and trigger three events:

- Displaying "in process" to the user. Initially, we assume it displays "ready" (3).
- Triggering the time counter (4)
- Triggering preparing the coffee (5)

The machine is continuously receiving time units; however, the triggering makes it "pay attention" and count these time units. Note that the time sphere is represented by a clock picture for illustrative purposes, but it is really the flowsystem that creates time units. Also, it is possible to detail the coffee sphere by drawing flowsystems for the coffee powder and water separately to be processed and make coffee.

At the end of the coffee preparation time, the cup is dropped (6) and the coffee is released (7); this happens in the machine compartment subsphere to create the filled cup (8). Creating the filled cup and releasing it trigger waiting time (9) to pick up the cup and display that (10). If the person takes out the filled cup (11), this triggers displaying "ready" and triggers (12) the flow of coffee to the person (13).

Note that, in general, the filled cup sphere includes three subspheres: the filled cup (cup+coffee), coffee, and cup (see Fig. 19). In any sphere, we can focus on any of its subspheres. Accordingly, when the person removes the filled cup outside the compartment the "attention" (matters of interest) is on the filled cup and the coffee subspheres (flowsystems (11 and 12)), but the cup by itself is of no interest.

Continuing the flows, when the coffee is received by the person (14), he/she drinks it to trigger creation of publications (15) that flow to the observer (16), which in turn triggers initializing a waiting time period for the next publication (17). If no publication arrives, this triggers creation of a complaint (18). We assume that initially the waiting timing here is set to zero.

Returning to releasing the filled cup that triggers waiting time (9); if the waiting time is over (19), then this triggers (20) checking whether the filled cup has been already removed (21); if not, the filled cup is dispensed with (22).

VI. CONCLUSION

Methodologies of time representation can be based on states, UML, Petri nets, and other types of diagrams. Each has its own advantages and weaknesses, especially with regard to having the features of understandability and simplicity. This paper proposed a flow-based representation that is based on the notion of flow with a focus on exploring the representation of time. The new methodology was demonstrated through sample timing-related problems.

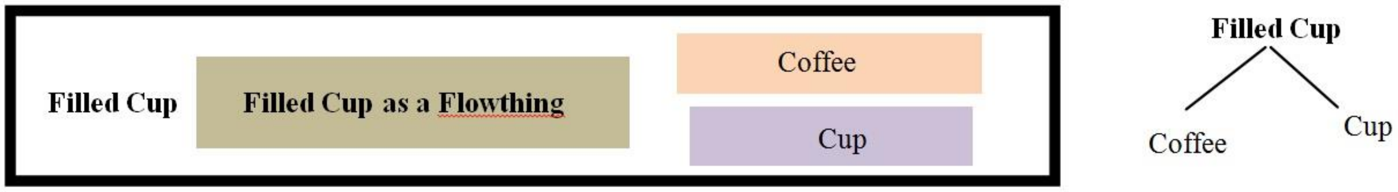


Fig. 19. The filled cup as a flowthing and its two flowthing components

FM can serve as an early system understanding and communication among stakeholders, including those without technical knowledge, and facilitate agreement between clients/users and designers. Additionally, it can be used as a base for system development and the design phase. The resultant FM representation avoids ambiguous textual language and heterogeneous diagramming. Of course, FM is still not well developed in comparison with such well diagram-oriented modeling methodology such as UML. Its weaknesses in terms of expressivity and complexity have to be studied more in different applications. Nevertheless, comparing FM diagrams side by side with other types of modeling techniques reveals it is a promising viable modeling tool.

We are currently exploring further time representation in FM, especially its relation to the actual design phase.

REFERENCES

[1] H. Personnier, M.-A. le Dain, and R. Calvi. Failures in Collaborative Design with Suppliers: Literature Review and Future Research Avenues. 21st Annual IPSERA Conference **(2012)** Italy.

[2] Luqi and V. Berzims, Knowlede-Based Support for Rapid software Prototyping, IEEE Express, pp. 9–18 **(1988)**

[3] E. Klingerman and A. D. Stoyenko. Real-Time Euclid: A Language for Reliable Real-Time Systems, IEEE Trans. Softw. Engng., SE-12, 9, **(1986)** , pp. 941-949.

[4] S. Berryman and I. Sommerville. Modelling Real-Time Constraints. Proceedings of the 3rd International Conference on Software Engineering for Real-Time Systems, **(1991)**; Cirencester, UK

[5] G. K. Palshikar. An Introduction to Model Checking. Available from www.eetasia.com/ARTICLES/2005FEB/B/2005FEB16_EMS_ST_TA.pdf. **(2005)**

[6] T. A. Henzinger and J. Sifakis. The Embedded Systems Design Challenge. Proceedings of the 14th International Symposium on Formal Methods (FM), **(2006)**

[7] S. Al-Fedaghi, Pure Conceptualization of Computer Programming Instructions. International Journal of Advancements in Computing Technology, 3, 9 (2011)

[8] S. Al-Fedaghi and A. Alrashed, Threat Risk Modeling. International Conference on Communication Software and Networks (ICCSN), (2010) February 26–28; Singapore

[9] S. Al-Fedaghi. Flow-Based Enterprise Process Modeling. International Journal of Database Theory and Application Compendex, 6, 3 (2013)

[10] S. Al-Fedaghi. Schematizing Proofs Based on Flow of Truth Values in Logic. IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC), (2013) October 13–16; Manchester, UK

[11] S. Al-Fedaghi. An Alternative Approach to Multiple Models: Application to Control of a Production Cell. International Journal of Control and Automation, 7, 4 (2014)

[12] S. Al-Fedaghi. System for a Passenger-Friendly Airport: An Alternative Approach to High-Level Requirements Specification. International Journal of Control and Automation, 7, 2 (2014)

[13] K. Tindell and J. Clark. Holistic Schedulability Analysis for Distributed Hard Real-Time Systems. Microprocessing and Microprogramming, 40 **(1994)**

[14] W. Garnett. Letter to the Editor. Nature **(1920)**

[15] B. Strachan. The Skirts of Alpha, APPENDIX I: SOME THOUGHTS ABOUT DIMENSIONS. http://panpsychic-philosophy.org.uk/index.php

[16] Lucid Software Inc. 2, UML–Timing Diagram Tutorial. **(2014)** https://www.lucidchart.com/pages/UML-timing-diagram-tutorial

[17] webmaster@uml-diagrams.org., Timing Diagrams, **(2014)** http://www.uml-diagrams.org/timing-diagrams.html

[18] Lucid Software Inc, Timing Diagram, **(2014)** https://www.lucidchart.com/pages/uml/timing-diagram

[19] H. Anderson. CS5270 Verification of Real Time Systems, 6.2.2 Coffee Machine Example in Uppaal. http://www.comp.nus.edu.sg/~cs5270/Notes/chapt6a.pdf

[20] H. Anderson. Verification of Real-Time Systems. **(2007)** http://www.comp.nus.edu.sg/~cs5270/2006-semesterII/foils11.colour.pdf

[21] K. G. Larsen. Quantitative Model Checking: Real-Time Systems, Exercises. http://people.cs.aau.dk/~kgl/QMC2010/exercises/#coffee

[22] H. S. Hong, J. H. Kim, S. D. Cha, and Y. R. Kwon. Static Semantics and Priority Schemes for Statecharts. Computer Software and Applications Conference (COMPSAC), **(1995)**

[23] S. Van Langenhove. Towards the Correctness of Software Behavior in UML: A Model Checking Approach based on Slicing, Ph.D. thesis, Faculteit Wetenschappen — Universiteit Gent, **(2006)** http://lib.ugent.be/fulltxt/RUG01/000/970/662/RUG01-000970662_2010_0001_AC.pdf