Thinging the Robotic Architectural Structure

Sabah Al-Fedaghi and Manar AlSaraf
Computer Engineering Department, Kuwait University, Kuwait
Email: sabah.alfedaghi@ku.edu.kw, manar.j.alsaraf@grad.ku.edu.kw

***Abstract*—Robot architecture refers to the architectural structure of a system and its subsystems and how those subsystems interact. From the modeling point of view, the architecture is the conceptual model that defines the structure, behavior, and other aspects of a system. This paper proposes the adaptation of a recent modeling technique, thinging machine (TM), to develop a high-level neutral (no hardware or software) description of robot architecture. Using the TM, several robot projects from the literature are remodeled and an actual robot case study is redesigned. The results point to the viability of applying the TM in robotics.**

***Index Terms*—robotic architectural model, conceptual model, navigating robot, SLAM, thinging machine**

I. Introduction

Many projects in robotics are developed from scratch with an absence of a systematic approach that specifies robotic architectures, which may cause ad hoc designs that are not flexible and reusable [1]. According to Ramaswamy [1], "In the last two decades, the robotics research community has seen a large number of middlewares, code libraries, and component frameworks developed by different research laboratories and universities… It is high time that roboticists transform themselves as system thinkers in addition to being domain experts."

According to Kortenkamp and Simmons [2], robot architecture refers to two related, but distinct, concepts.

(i) *Architectural structure* describes how a system is divided into subsystems and how those subsystems interact. From a robot architecture perspective, it facilitates the decomposition of systems into simpler, largely independent modules. This robot system structure is often represented informally using traditional "boxes and arrows" diagrams or more formally using techniques such as UML. Marinho da Silva, Rodrigues de Souza, Simões and Campos [3] recently presented a framework based on Petri Net, which is utilized in modeling multi-robot systems due to the high interactions between them.

(ii) On the other hand, *architectural style* refers to the computational concepts that underlie a given system. In the robotics community, for example, one robot system might use a message passing style of communication, while another may use a synchronous client-server approach.

Many existing robot systems have unclear architectures because the implementations have no defined subsystem boundaries [2]. A well-defined architecture has significant advantages in the specification, execution, and validation of robot systems. Robot architecture facilitates discipline in the design and implementation of robotic systems. For example, according to Kortenkamp and Simmons [2], "Separating behaviors into modular units helps to increase understandability and reusability, and can facilitate unit testing and validation."

In the modeling community, according to Jaakkola and Thalheim [4], the architecture is *the conceptual model that defines the structure, behavior, and other aspects of a system*. We will adopt this definition.

Architecture in this context refers to the integration of various functionalities to determine the overall behavior of a system. It includes control architecture that integrates various kinds of hardware and software modules [5]. According to Ramaswamy [1], an architecture model captures multiple viewpoints that satisfy the requirements of various stakeholders. A hardware engineer sees the parts that need particular processors, whereas a system architect is interested in component topology. A good architecture model acts as a mediator between requirements and implementation. The architecture model plays a critical role in the development life cycle.

This paper proposes a systematic methodological approach that helps in specifying various aspects of robots at the architectural level. Section 2 provides a brief review of related research with samples of current diagramming methods. Section 3 introduces a brief description of the TM model. Several robot projects from the literature are remodeled using the thinging machine (TM) concept in Sections 4 and 5. In Section 6, an actual robot case study is redesigned using the TM.

II. Related Works

This section summarizes an extensive review of robot architecture. Robot architecture and programming began in the late 1960s using stepping motors, cameras, range-finder sensors, touch sensors (to identify collisions), and computer connections via radio links [6]. A major development occurred in the 1980s with the so-called subsumption architecture built from layers of interacting finite state machines that each connect sensors to actuators [7]. Several behavioral architectures arose in addition to subsumption with schemes for combining the outputs of behaviors [8]. Furthermore, Firby [9]

Manuscript received October 23, 2019; revised July 13, 2020.

doi: 10.18178/ijmerr.9.8.1110-1121

developed the Reactive Action Package system that aimed at the integration of reactivity and deliberation in three-layer architecture. Bonasso [10] devised the so-called Rex machines with an architecture that guarantees consistent semantics between a robot's internal and external states. Moving to the 21st century, the syndicate architecture [11] extended the three tiers of interacting control processes to multi-robot coordination.

In subsumption architecture, behaviors run concurrently with a hierarchical control scheme, whereas in autonomous robot architecture, [12] behaviors are combined using potential functions. Other architectures [8] use explicit arbitration mechanisms to choose among potentially conflicting behaviors.

Kortenkamp and Simmons [2] give an example of a delivery robot that operates in an office building to illustrate various architectural approaches that handle robot behavior. The behavioral control layer contains the functions to move and carry out delivery tasks, assuming an a priori map. Some possible behaviors for this robot include:

1. Move to location while avoiding obstacles
2. Move down hallway while avoiding obstacles

…

8. Determine location
9. Find the destination office
10. Announce delivery.

This paper proposes the adaptation of the recently developed TM modeling technique to create a high-level neutral (no hardware or software) description of robot architecture. The service robot example above gives an opportunity to examine the form of the description produced by the TM.

Before giving our approach to modeling this problem, we describe service robot architecture.

Kim [5] developed software for a service robot's autonomous navigation system using UML. According to Kim, the functional requirements define what the system should do. Kim used a use case model that is specified as follows:

1. The commander enters a destination.
2. The system calculates the optimal path to the destination.
3. The system commands the actuator to start moving to the destination.
4. The actuator notifies the system that it has started moving.
5. The system periodically reads sensor data and calculates the current position.
6. The system determines that it has arrived at the destination and stops.
7. The wheel actuator notifies the system that it has stopped moving [5].

Figs. 1–5 show parts of the system's different diagrams. Static modeling is achieved using a class diagram. Kim also used a state diagram, as well as collaboration, sequence, and other diagrams. Additionally, to integrate these diagrams, Kim [5] applied the concurrent object modeling and architectural design method [13] to analyze and design subsystem structures and interfaces, including the synchronization and communication between them.

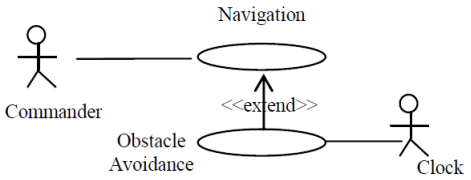


Figure 1. Use case diagram for navigation (partially redrawn from [5]).

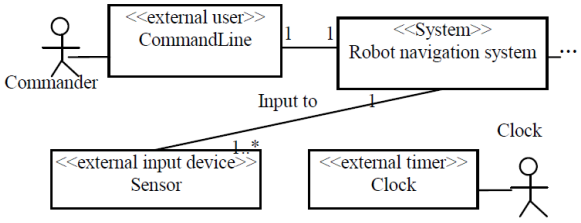


Figure 2. Robot navigation system context class diagram (partially redrawn from [5]).

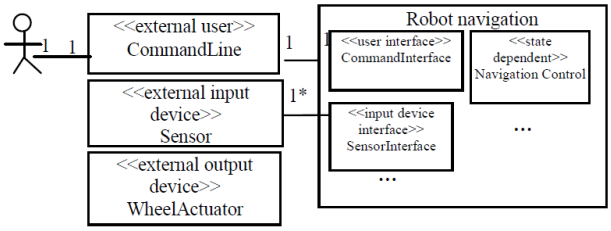


Figure 3. Object structuring class diagram for navigation system (partially redrawn from [5]).

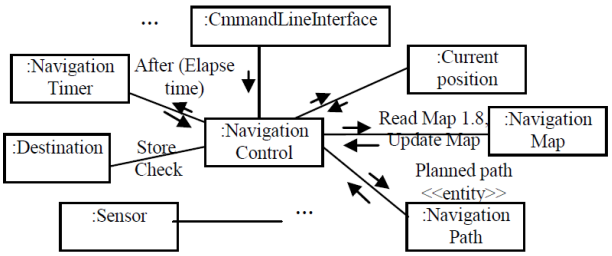


Figure 4. Collaboration diagram for navigation use case (partially redrawn from [5]).

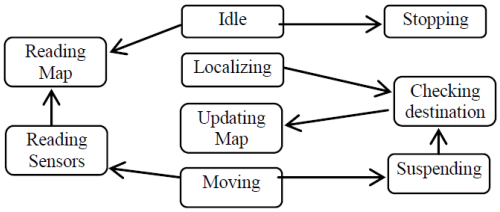


Figure 5. State chart for navigation control (partially redrawn from [5]).

III. About the TM

The TM is centered on *things* and (abstract) *machines* in a system. According to Malafouris [14], people are

creative "thingers" in the sense that "we make new things that scaffold the ecology of our minds, shape the boundaries of our thinking and form new ways to engage and make sense of the world." The TM is based on Heidegger's philosophical concept of *thinging* [15]. According to Riemer et al. [16], Heidegger's philosophy gives an alternative analysis of "(1) eliciting knowledge of routine activities, (2) capturing knowledge from domain experts and (3) representing organizational reality in authentic ways." More information about this topic can be found in [17]-[19].

The TM is the simplest type of the thing/machine and a generalization of the known input-process-output model. The *flow of things* in a TM refers to the exclusive conceptual movement among five operations (stages) as shown in Fig. 6. A thing is *what is created, processed, released, transferred, and/or received in a machine*.

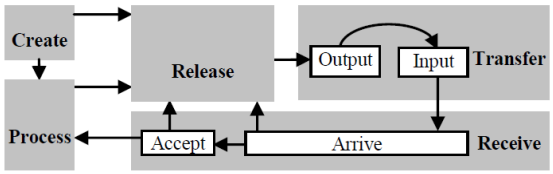


Figure 6. Thinging machine.

Accordingly, the TM stages can be described as operations that transform, modify, etc. things either in the abstract sense or in the "concrete" sense. They are briefly described as follows.

Arrive: A thing flows to a new machine (e.g., packets arrive at a port in a router).
Accept: A thing enters a TM after arrival (we will assume that all arriving things are accepted). Hence, we can combine arrive and accept as the **receiving** stage.
Release: A thing is marked as ready to be transferred outside the machine (e.g., in an airport, passengers wait to board after passport clearance).
Process: A thing is changed in description rather than producing a new thing.
Create: A new thing is born in the machine (e.g., a forward packet is generated in a machine).
Transfer: A thing is input or output in/out of the machine.

The TM includes one additional notation—*triggering* (denoted by dashed arrows)—that initiates a new flow (e.g., a flow of electricity triggers a flow of air). TM modeling has been utilized in many applications (e.g., Al-Fedaghi [20]-[42]).

IV. Applying the Proposed Architectural Structure

We simplify Kortenkamp and Simmons' [2] example by ignoring minor details (e.g., find doorknob). The TM uses only five verbs: create, process (change), release, transfer, and receive. It uses two types of arrows: a solid arrow denotes flow, and a dashed arrow signifies triggering. Fig. 7 shows the TM model of this example. The figure includes the starting point (circle 1), the robot inside the starting point area (2), other areas in the way to the destination (3), and the destination (4).

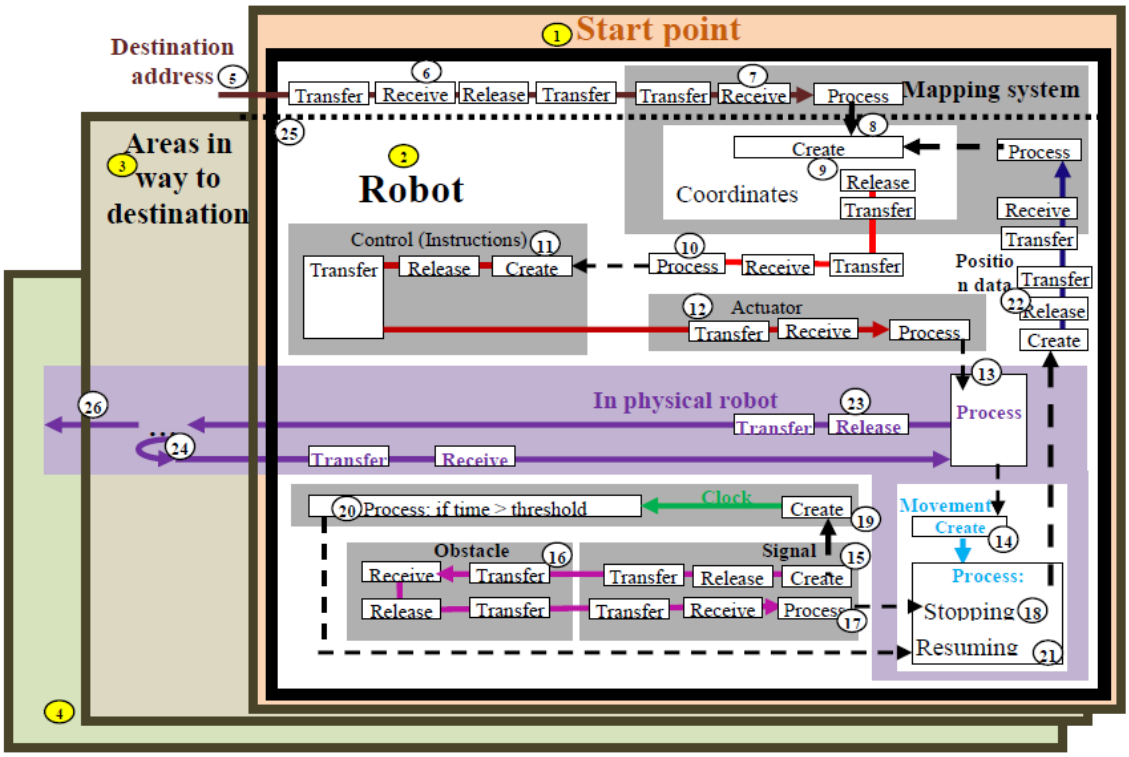


Figure 7. The static TM model of Kortenkamp and Simmons' [2] simplified example.

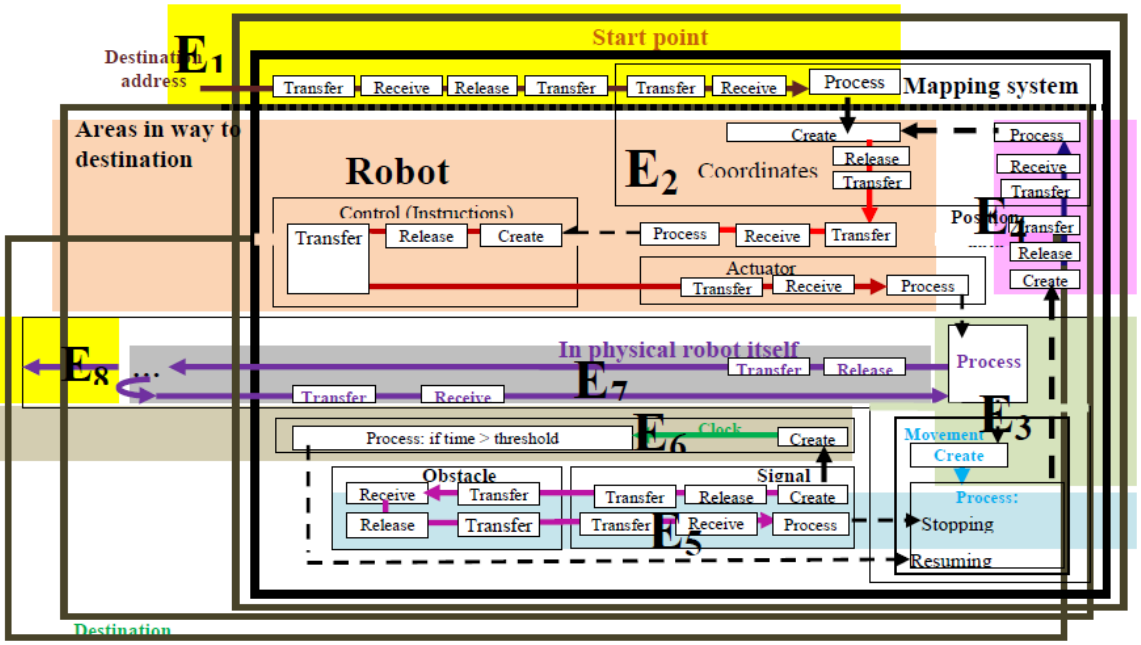


Figure 8. The static TM model of Kortenkamp and Simmons' [2] simplified example.

a. A destination address is input to the robot (5; upper left corner of the figure).
b. The robot receives (6) the address and sends it to its mapping system.
c. In the mapping system, the address is received (7) and processed (8), triggering the robot to create the necessary coordinates (9) for the movement to the destination.
d. The coordinates flow to be processed (10), which triggers the control unit to create (11) instructions that flow to the actuator (12).
e. The actuator processes the instructions and triggers the processing (13) of the physical robot, which involves creating (14) movement. Note that the movement is a component of the physical robot (white box inside the L-shaped purple figure).
f. A signal is continuously (to be discussed in the dynamic TM model) generated (15) that flows (16) to any obstacle (if any exist) and reflected to be processed (17) which stops (18) the robot's movement.
g. Creating the signal initializes the clock time (20). If the time reaches a certain threshold (21), then it triggers the robot to resume movement (22).
h. The robot's movement triggers the creation of current position data (23) that flow to the mapping system, creating new coordinates that cause the creation of new instructions.
i. The robot eventually moves (24) from one area to another (25) on its way to the destination. In all areas the robot's description is identical to the robot at the start, except for receiving the first destination input (separated by a dotted line).
j. At last, the robot arrives (26) at the destination.

To build the dynamic model we use the notion of events. We select the following events (see Fig. 8):

Event 1 (E_1): The destination address is received by the robot and processed.

Event 2 (E_2): Coordinates are generated and flows to the control unit to issue instructions to the actuator.

Event 3 (E_3): The robot moves.

Event 4 (E_4): Position data are generated and sent to the mapping system.

Event 5 (E_5): An obstacle causes the robot to stop.

Event 6 (E_6): The robot avoids the obstacle and movement is resumed.

Event 7 (E_7): The robot moves to another area on its way.

Event 8 (E_8): The robot reaches its destination.

Fig. 9 shows the robot system's behavior according to the chronology of events.

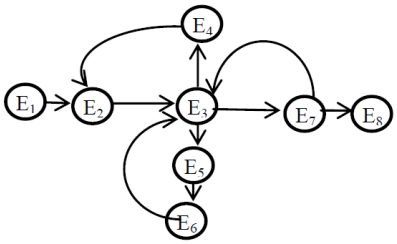


Figure 9. Chronology of events in Kortenkamp and Simmons' [2] simplified example.

V. TM vs. UML Activity Diagram

Matsas et al. [43] described a human-robot collaboration project that involves hand lay-up process of pre-impregnated carbon fabric in an industrial work cell.

In the project scenario, a robotic manipulator is assigned the task of picking patches and transferring them to a user. The user takes a patch from the robot and places it in the correct position inside a metallic die, and the robot proceeds to feed the next patch. The process is

repeated until the user has placed all patches properly in the die, hereby represented by the avatar's hands. During execution, the user is asked to wear a head-mounted display. Calibration of the user against the avatar skeleton is achieved by the user raising his or her hands with both elbows, thereby standing in a "Y" posture. After calibrating, users are allowed some time to familiarize themselves with the system. When the user moves in real space, the avatar's body and the virtual viewpoint change accordingly. When the user walks around in real space, the avatar follows the same path in the virtual environment. When the user turns his or her head, the avatar's head and the first-person camera attached to it respond accordingly. The user can also collide or interact with rigid bodies in the virtual scene and bend his or her body in every direction.

Fig. 10 is a diagrammatic representation of the system activity workflow using a UML activity diagram. This gives us opportunity to contrast the TM (Fig. 11) model with the UML representation.

In Fig. 11, the model starts with the user wearing the head-mounted display (circle 1) and entering the system area (2) to be processed (3), which involves the Kinect tracker detecting the user (4). The user then stands in the "Y" posture (5) for the Kinect tracker to detect (6). The user creates an avatar (7), then presses the start button via the avatar (8), which activates the robot (9 and 10). The robot holds the first patch (11), and then the user approaches (12) the robot (13).

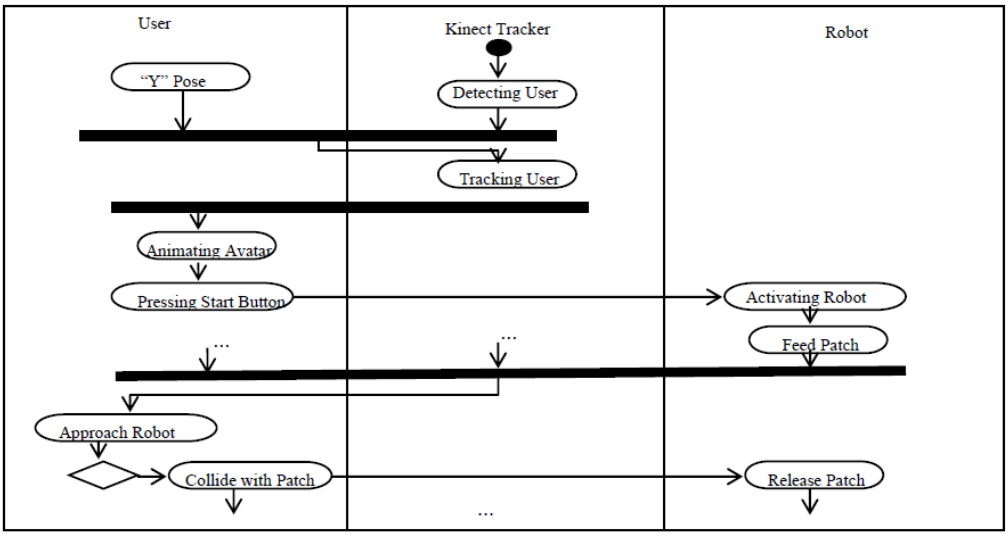


Figure 10. UML activity diagram for the system workflow (partially redrawn from [43]).

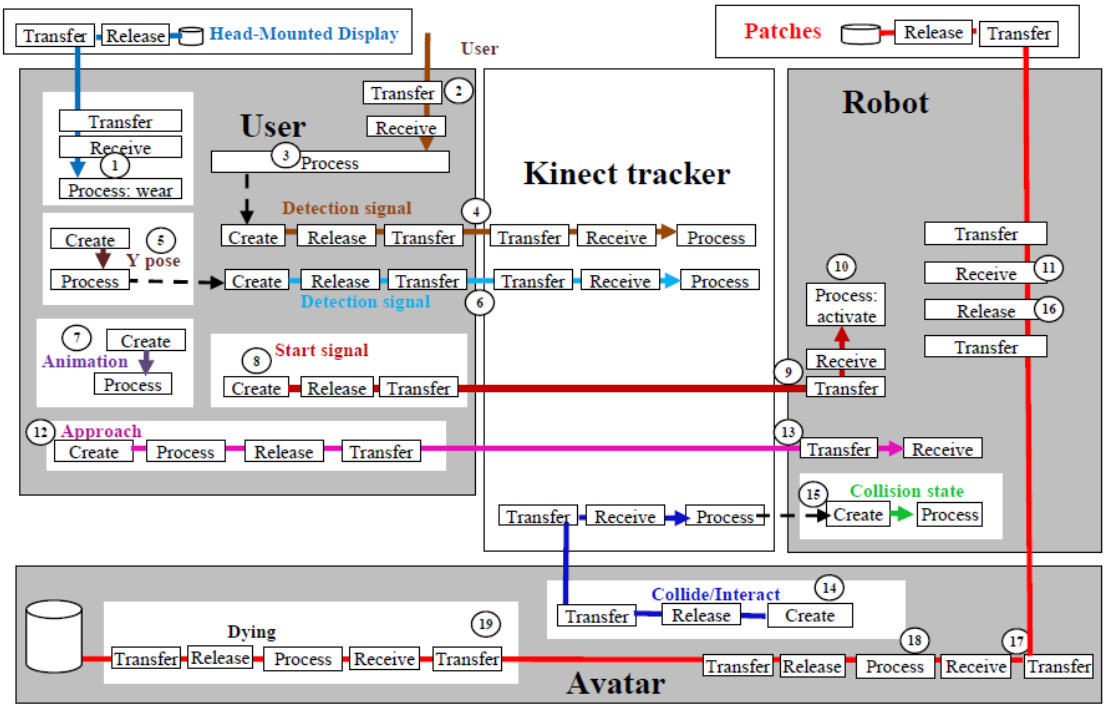


Figure 11. The TM model that corresponds to the activity diagram of Fig. 9.

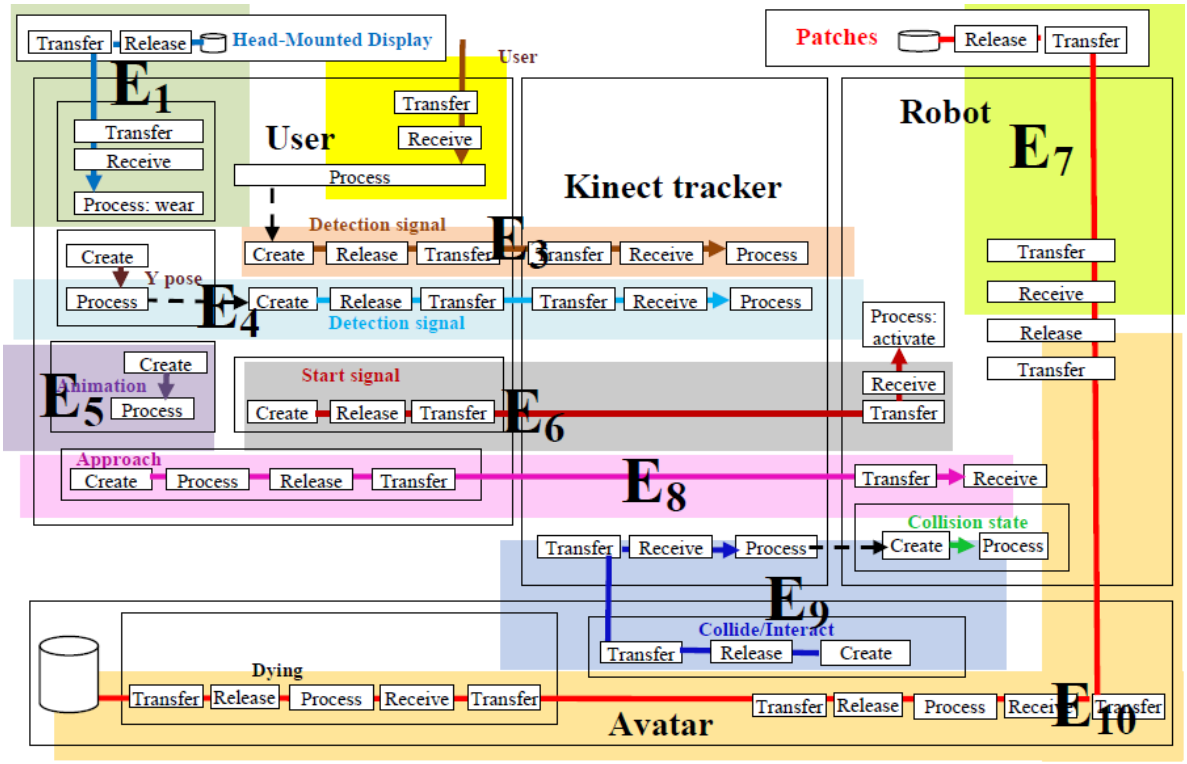


Figure 12. The events in the human-robot collaboration system.

The avatar then interacts with the robot (14) to create a collision state (15) in which the robot releases the patches (16). The avatar then receives (17) the patches for processing (18) and dying (19).

To construct the dynamic model, we identify the following events (see Fig. 12).

Event 1 (E_1): The user wears the head-mounted display.

Event 2 (E_2): The user moves in the system area.

Event 3 (E_3): The user is tracked by the Kinect tracker.

Event 4 (E_4): The user takes the "Y" posture, which is detected by the Kinect tracker.

Event 5 (E_5): The user performs animation of the avatar.

Event 6 (E_6): The user presses the start button, which activates the robot.

Event 7 (E_7): The robot holds a patch.

Event 8 (E_8): The user approaches the robot.

Event 9 (E_9): The avatar collides with the robot to create a collision state.

Event 10 (E_{10}): The robot releases the patch, which the avatar receives for processing and dying.

Fig. 13 shows the chronology of events.

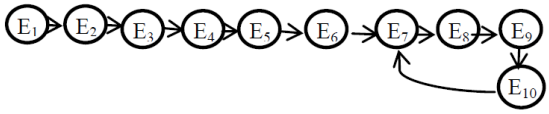


Figure 13. The chronology of events for the human-robot collaboration system.

VI. Case Study: The SLAM Robot Project

In this section, we introduce an actual university project that involves the well-known simultaneous localization and mapping (SLAM) algorithm [44]. The project aims to develop a robot that traverses an environment and provides a map of the environment.

A. Project Description

According to the researchers' final report, almost all current technology that allows a robot to traverse an area uses Global Positioning System (GPS) data." However, the university researchers want the robot to navigate inside buildings or dangerous locations when GPS cannot be used.

In the review of SLAM implementations in their final report, the researchers discussed an overall flow chart of the robot navigation developed by French company NAO [45] (see Fig. 14). We show such a diagram to illustrate the description of the interest level in the project.

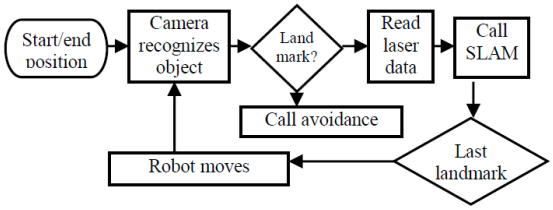


Figure 14. The NAO overall navigation flow chart (from [45]).

The authors in [44] created a new robot named The Exploring Master (TEM) that outperforms similar projects. Many hardware and software tools are compared/contrasted, including platforms, motors, sensors, and software tools (e.g., robot control center, phidgets, stepper drives, interface kits, adapters, and sonar range finders) to decide which hardware best suits the project. This indicates that implementation details are considered very early in the robot development cycle.

Then came a system architecture as a model that defines the structure and behavior of a system. "Basically, it is a representation of the system that contains the system's components and the relationships between them including the direction of data transfer" [44]. Architecture diagrams mostly involve hardware devices, such as a PC, monitor, phidgets, battery, and stepper motor. A use-case system is shown in Fig. 15.

The next section in the university researchers' report discusses the hardware and software components. Fig. 16 shows the flowchart of the movement subsystem.

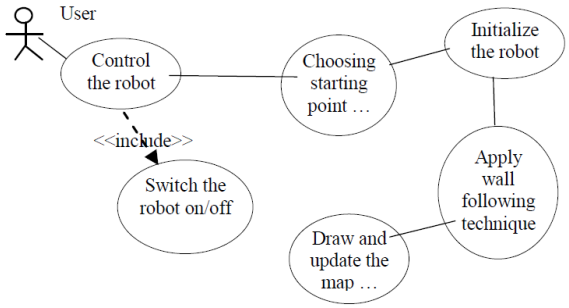


Figure 15. TEM use-case diagram (partially redrawn from [44]).

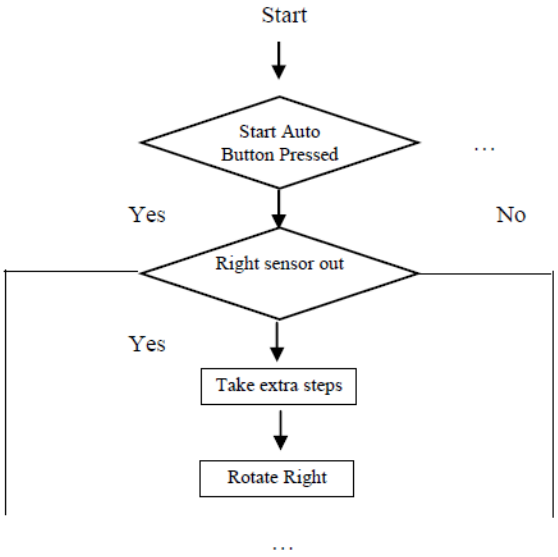


Figure 16. Movement subsystem flowchart (partially redrawn from [44]).

In addition, the researchers discuss the obstacle detection subsystem, including the problem faced and how to overcome it. Furthermore, to reduce measurement errors, the researchers used a compass to orient the angles of the robot's movement. Finally, they introduce the testing results.

B. Observation

The robot was built without substantial reliance on requirement analysis and logical design. For example, the researchers never mentioned the use-case model in implementing the project. The project is based on hardware-oriented construction and lacks a conceptual architecture model that integrates various functionalities to determine the overall behavior of the system. As mentioned previously in the introduction, a good architecture model acts as a mediator between requirements and implementation, and it plays a critical role in the development life cycle. We believe that the TM can play a role that enhances the analysis and design phases in the robotics development cycle.

C. General Problem

We claim that constructing robots based mainly on hardware with little attention to conceptual modeling in requirement analysis and design is a common problem in robotics, especially in university environments. We give another example of a University of Auckland project, as described in Looker et al. [46].

Looker et al. [46] state that the SLAM problem is difficult because sensors do not receive enough information about the robot's surroundings. They identified critical tasks (e.g., motor control and movement, sensor management and scanning, and map generation and presentation) and studied motor control, including the motor speeds and their signals. They conducted a series of tests (with equations and graphs) using a tachometer, and they analyzed trajectory control, wall follow control, yaw control, and position control, which they integrated in a unified control system shown in Fig. 17.

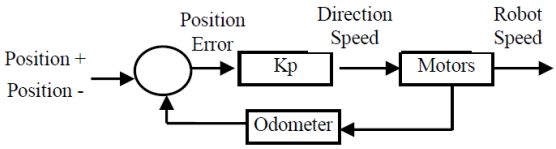


Figure 17. Position control system diagram.

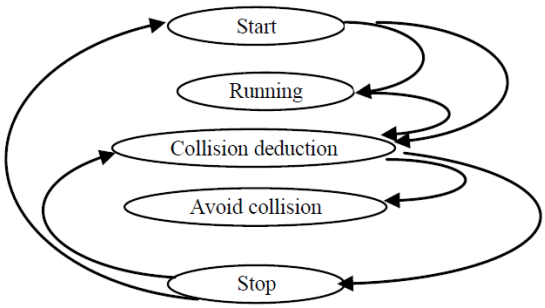


Figure 18. Finite state diagram.

Looker et al. [46] discussed blockage identification using an in-depth level of description. Furthermore, they used mathematical equations and diagrams to discuss sensing, scanning, and geometry calculations.

They used a finite state machine (Fig. 18) to describe the robot's highest level of intelligence. The description level of the robot's dynamic behavior is specified as follows: "When the robot first starts up there are 3 basic steps it completes. Throughout all these steps, obstacle detection and avoidance is automatically running. During the start state, when a blockage is identified as an obstacle, the robot rotates 20˚ and rescans, this is done continuously until an orientation is found in which there are no obstacles blocking the wall."

They described the code structure in plain English and made a class diagram showing the direction controller, sensors, map maker, collision detection, motor, and motor controller. In addition, they included hardware descriptions of various types of sensors.

Looker et al. [46] concluded that the project was very successful: "Unfortunately on the day of assessment there were issues with the sensor and some minor code fixes. These problems were fixed later." The author believed that the verification of functioning hardware could have been improved. One other difficulty faced while working in a group is understanding each other's code and ensuring various modules match.

As we can observe, Looker et al. [46] progressed the project from the hardware and mathematics stages but without an architectural view that ties the project together. They developed a state diagram but it hardly expressed the project.

VII. TM Model of a Simple Navigating Robot

We have decided to redesign the navigating robot described in section 6 (A) to demonstrate the proposed TM-based architecture. It is important to note that one of the authors of this paper was part of the team that built that robot. For practical reasons, the navigating robot will be simplified as follows.

The simplified problem involves a robot that navigates to measure the length of a wall, as shown in Fig. 19. The robot's first task is to reach one end of the wall, then it goes to the other end to measure the distance between the two ends (i.e., the wall's distance).

Fig. 20 shows the static TM representation of the robot system that includes a user interface, a processor, and the robot itself.

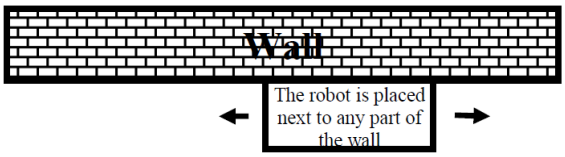


Figure 19. The problem of measuring the length of a wall.

The robot consists of two parts: the internal controller and the physical robot. The user enters a one-time threshold (1), and a signal is generated in the user interface (2) that flows to the processor (3) then to the robot controller (4). In the robot controller, the signal is processed (5) to trigger the state of the physical robot to be ON (6), which triggers (7) the sensor to be ON (8). Accordingly, the robot works as follows.

The sensor generates signals (9) that flow to the robot controller (10) to move the robot toward the wall (11) where we have three situations.

i. **Location beside wall (not corner):** The signal is reflected (12) and arrives to the robot controller where it is processed (13) and triggers a one-step movement of the robot (14). This updates the counter (15) and triggers (16) the sensor to generate another signal (9).

Location at the corner: Upon receiving the signal (17), the clock time is triggered to be ON (18) and is triggered every incoming sensor signal in which it takes its course (i.e., processed (19)). In addition, the threshold (2) is frequently compared to the clock time, in other words, processed (20). If the clock time is greater than the threshold, then the following occurs:

ii. **First corner**: Upon the robot's arrival at the first corner, a first-time flag is set (21). In addition, the robot is triggered to rotate 180 ° (22), followed by initializing the counter (23) and generating a new signal (24).

iii. **Second corner**: The signal triggers the counter (25) to be transferred (26) to the processor to generate (27) the display of the distance on the user interface (28).

To build the dynamic model, we identify the following events (see Fig. 21)**.**

Event 1 (E_1): The user enters a one-time threshold.

Event 2 (E_2): The user turns the robot on.

Event 3 (E_3): The processor sends the TURN ON signal to the robot.

Event 4 (E_4): The first-time flag is turned ON, and the distance field is created.

Event 5 (E_5): The robot is switched ON.

Event 6 (E_6): The sensor is switched ON.

Event 7 (E_7): The sensor sends a signal to the controller.

Event 8 (E_8): The controller sends a signal to the wall.

Event 9 (E_9): The signal is reflected by the controller.

Event 10 (E_{10}): The robot takes one step forward and updates the counter.

Event 11 (E_{11}): The time at which the sensor sent the signal is registered

Event 12 (E_{12}): The robot reaches the first or second corner.

Event 13 (E_{13}): The flag is updated and the robot rotates 180 °(corner 1).

Event 14 (E_{14}): The counter is reset (corner 2).

Event 15 (E_{15}): The distance is retrieved and displayed.

Fig. 22 shows the behavior of the robot system according to the chronology of events.

VIII. Conclusion

It has been proven that robot architectures play an important role in fully understanding any system. In addition, they allow systems to control actuators, monitor execution, and be flexible with any changes. Researchers have come up with several architectures that can be applied in various projects. It is up to the researcher to choose among the available architectures which architecture is best depends upon the requirements. This paper includes a new model (i.e., TM) that may be used in systems with conceptual movement between real and virtual objects. This type of model contains five operations with transition flow, providing a more detailed, expressive methodology than common models used today. Therefore, we used it to remodel existing robot projects for better understanding and visualization of the robots' systems.

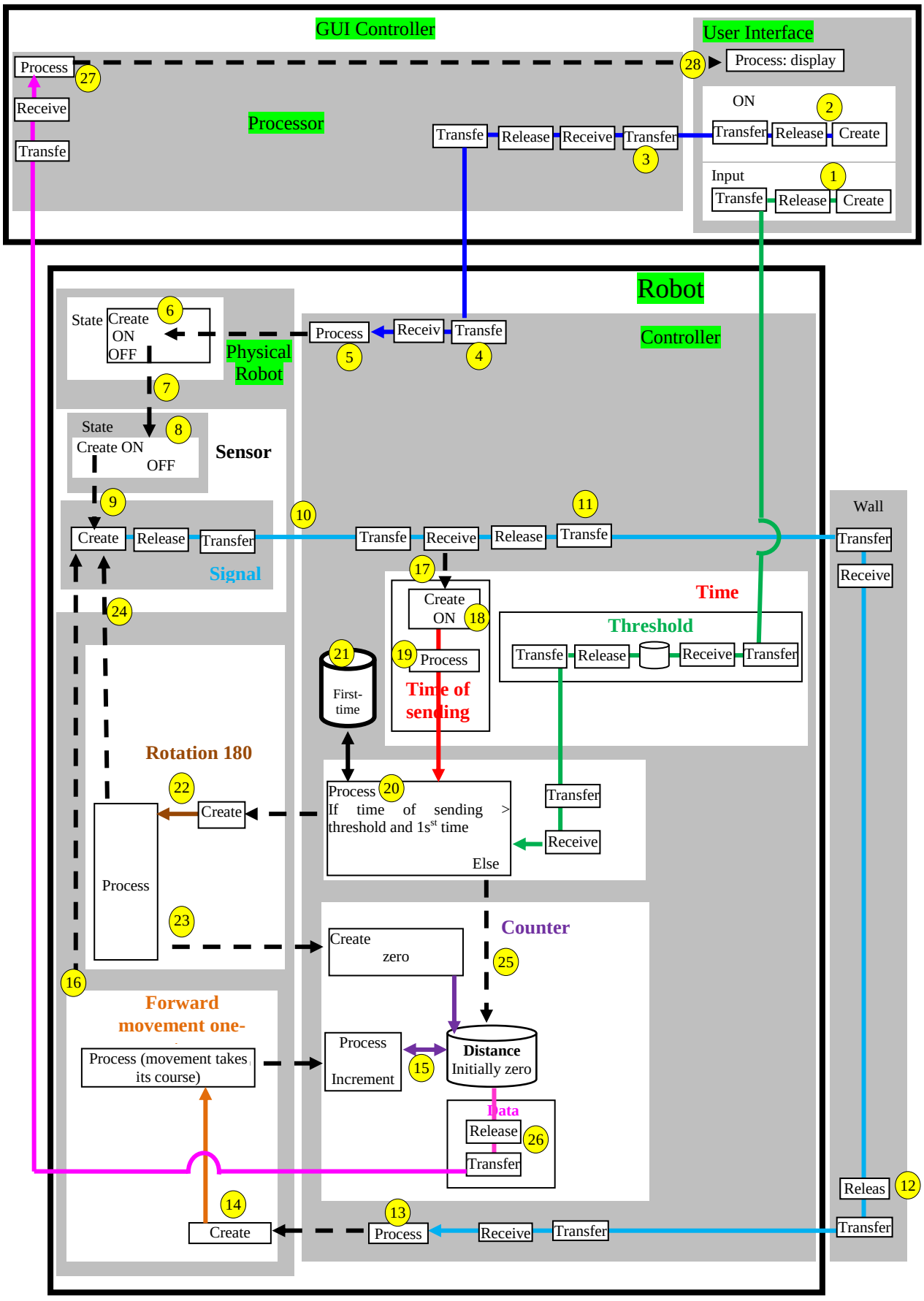


Figure 20. The static TM model of the measuring the wall problem.

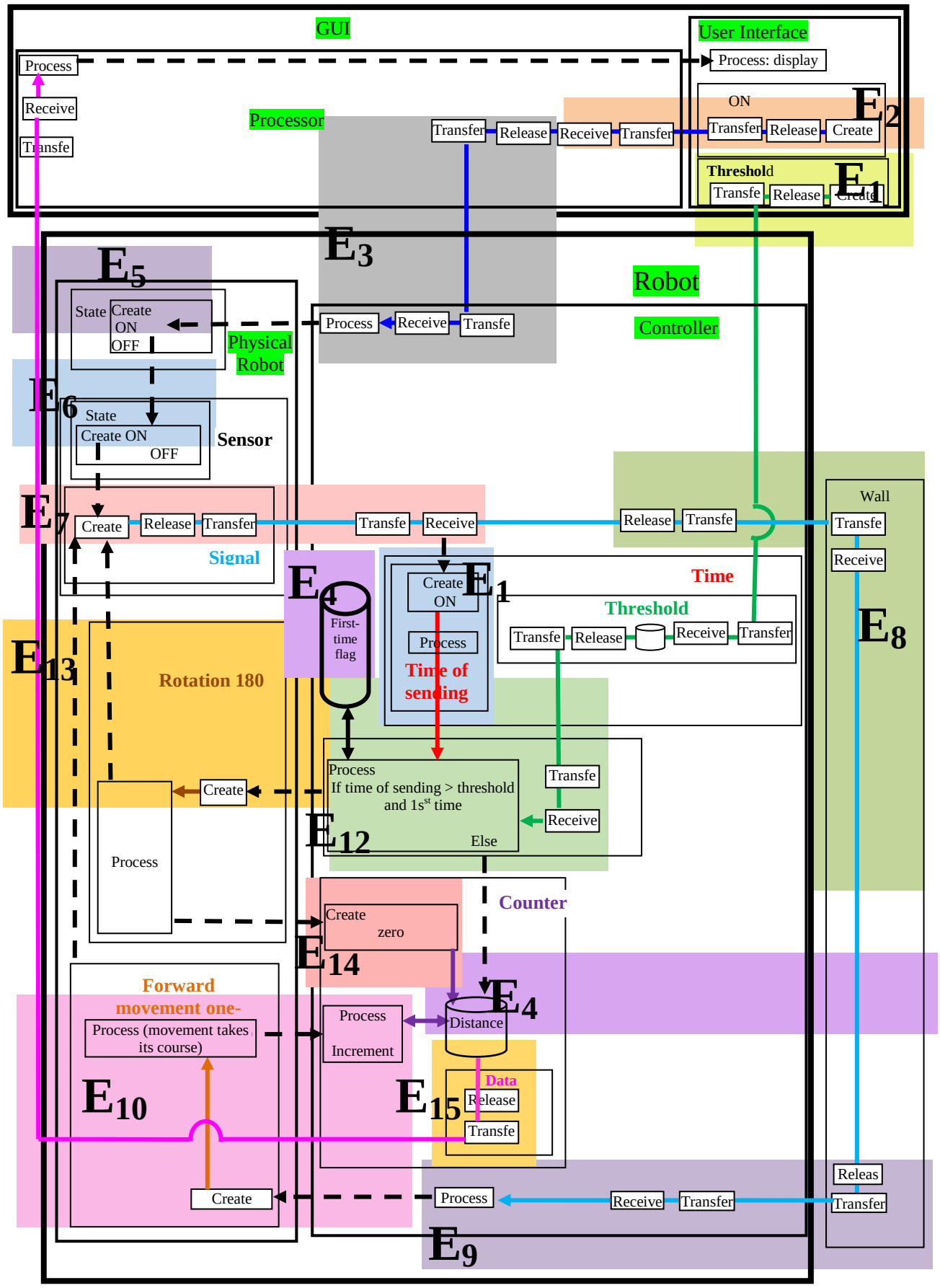


Figure 21. The dynamic model of measuring the length of a wall.

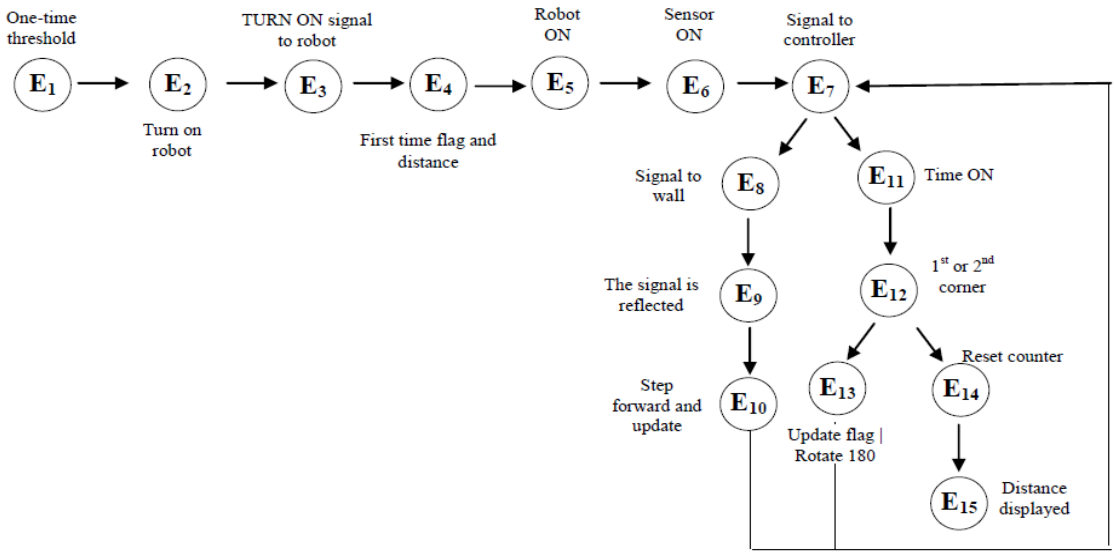


Figure 22. The chronology of events in the model of measuring the length of a wall.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The first author is the main developer of the TM model as described in Section III. The second author is one of the builders of the navigating robot described in Section VI. Both authors developed the TM model in Section VI.

References

[1] A. Ramaswamy, "A model-driven framework development methodology for robotic systems," Ph.D. thesis, Dept. Comput. Sci. and Syst. Eng., Univ. Paris-Saclay, France, September, 2017.

[2] D. Kortenkamp and R. Simmons, "Chapter 8: Robotic systems architectures and programming," in *Handbook of Robotics*, Berlin: Springer-Verlag, 2008, pp. 187-206.

[3] R. Marinho da Silva, J. Rodrigues de Souza; M. A. C. Simões and J. A. P. Campos, "Framework for modeling autonomous multi-robots systems," *15th Latin American Robotics Symposium (LARS) Joao Pessoa*, Brazil. November 6-10, 2018.

[4] H. Jaakkola and B. Thalheim, "Architecture-driven modelling methodologies," *Frontiers in Artificial Intelligence and Applications*, vol. 225, pp. 97-116, 2010.

[5] M. Kim, S. Kim, S. Park, M. Choi, M. Kim and H. Gomaa, "UML-based service robot software development: A case study," in *28th International Conference on Software Engineering* (ICSE 2006), Shanghai, 2006, pp. 534-543.

[6] N.J. Nilsson, "A mobile automaton: An application of AI techniques," in *Proc. of the First International Joint Conference on Artificial Intelligence*, San Francisco, 1969, pp. 509-520.

[7] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, 1986, pp. 14-23.

[8] J. K. Rosenblatt, "DAMN: A distributed architecture for mobile robot navigation," PhD dissertation, Carnegie Mellon University, United States, 1997.

[9] R. J. Firby, "An investigation into reactive planning in complex domains," in *Proc. the Fifth National Conference on Artificial Intelligence*, 1987.

[10] R. P. Bonasso, "Integrating reaction plans and layered competences through synchronous control," in *Proc. International Joint Conferences on Artificial Intelligence*, 1991.

[11] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh. "Coordinated multiagent teams and sliding autonomy for largescale assembly," in *Proc. the IEEE, Special Issue on Multi-Agent Systems*, vol. 94, no. 7, July 2006.

[12] R. C. Arkin and T. Balch, "AuRA: Principles and practice in review," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2/3, pp. 175-188, April 1997.

[13] H. Gomaa, "Designing concurrent, distributed, and real-time applications with UML," in *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Boston, USA: Addison-Wesley Longman Publishing Co., Inc. 2000.

[14] L. Malafouris, "The feeling of and for clay," *Pragmatics & Cognition*, vol. 22, no. 1, pp. 140–158, January 2014.

[15] M. Heidegger, "The thing," in *Poetry, Language, Thought*, A. Hofstadter, Trans. New York, NY: Harper & Row, 1975, pp. 161-184.

[16] K. Riemer, R. B. Johnston, D. Hovorka, and M. Indulska, "Challenging the philosophical foundations of modeling organizational reality: The case of process modeling," International Conf. on Information Systems, Milan, Italy, 2013. Available: http://aisel.aisnet.org/icis2013/proceedings/BreakthroughIdeas/4/

[17] S. Al-Fedaghi, "Five generic processes for behavior description in software engineering," *Int. J. Comput. Sci. Inf. Secur.*, vol. 17, no. 7, pp. 120-131, July 2019.

[18] S. Al-Fedaghi, "Thing/Machine-s (Thimacs) applied to structural description in software engineering," *Int. J. Comput. Sci. Inf. Secur.*, vol. 17, no. 8, August 2019.

[19] S. Al-Fedaghi, "Toward maximum grip process modeling in software engineering," *Int. J. Comput. Sci. Inf. Secur.*, vol. 17, no. 6, pp. 8-18, June 2019.

[20] S. Al-Fedaghi and A. AlQallaf, "Modeling and control of engineering plant processes," *Int. J. Applied Syst. Studies*, vol. 8, no. 3, pp. 255-277, 2018.

[21] S. Al-Fedaghi and N. Al-Huwais, "Conceptual modeling of inventory management processes as a thinging machine," *Int. J. Advanced Comput. Sci. and Applications*, vol. 9, no. 11, November 2018.

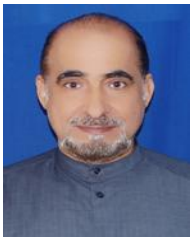
[22] S. Al-Fedaghi and M. Al-Otaibi, "Conceptual modeling of a procurement process: Case study of RFP for public key infrastructure," *Int. J. Advanced Comput. Sci. and Applications*, vol. 9, no. 1, January 2018.

[23] S. Al-Fedaghi, "Privacy things: Systematic approach to privacy and personal identifiable information," *Int. J. Comput. Sci. and Inf. Sec.*, vol. 16, no. 2, February 2018.

[24] S. Al-Fedaghi and J. Al-Fadhli, "Modeling an unmanned aerial vehicle as a thinging machine," in *Proc. 5th International Conference on Control, Automation and Robotics* (ICCAR 2019), Beijing, 2019.

[25] S. Al-Fedaghi and G. Aldamkhi, "Conceptual modeling of an IP phone communication system: A case study," in *Proc. 18th*

Annual Wireless Telecommunications Symposium (WTS 2019), New York, 2019.
[26] S. Al-Fedaghi and E. Haidar, "Programming is diagramming is programming," in *Proc. 3rd International Conference on Computer, Software and Modeling*, Barcelona, 2019.
[27] S. Al-Fedaghi and M. Al-Otaibi, "Service-oriented systems as a thinging machine: A case study of customer relationship management," *IEEE International Conference on Information and Computer Technologies (ICICT)*, Kahului, HA, 2019.
[28] S. Al-Fedaghi and Y. Atiyah, "Modeling with thinging for intelligent monitoring system," *IEEE 89th Vehicular Technology Conference: VTC2019-Spring*, Kuala Lumpur, Malaysia, 2019.
[29] S. Al-Fedaghi and A. Hassouneh, "Modeling the engineering process as a thinging machine: A case study of chip manufacturing," in *Proc. 8th Computer Science On-line Conference* (CSOC 2019). Springer Advances in Intelligent Systems and Computing, in press.
[30] S. Al-Fedaghi and H. Alnasser, "Network architecture as a thinging machine," *Symposium on Mobile Computing, Wireless Networks, & Security (CSCI-ISMC)*, Las Vegas, Nevada, 2018.
[31] S. Al-Fedaghi and M. Alsulaimi, "Privacy thinging applied to the processing cycle of bank cheques," in *Proc. 3rd International Conference on System Reliability and Safety* (ICSRS 2018), Barcelona, Spain, 2018.
[32] S. Al-Fedaghi and H. Almutairi, "Diagramming language for process documentation," in *Proc. 15th International Conference on Applied Computing* (AC 2018), Budapest, Hungary, 2018.
[33] S. Al-Fedaghi and H. Aljenfawi, "A small company as a thinging machine," in *Proc. 10th International Conference on Information Management and Engineering* (ICIME 2018), Manchester, UK, 2018.
[34] S. Al-Fedaghi and M. Alsharah, "Security processes as machines: A case study," in *Proc. Eighth international conference on Innovative Computing Technology* (INTECH 2018), London, 2018.
[35] S. Al-Fedaghi and R. Al-Azmi, "Control of waste water treatment as a flow machine: A case study," in *Proc. 24th IEEE International Conference on Automation and Computing* (ICAC'18), Newcastle upon Tyne, UK, 2018.
[36] S. Al-Fedaghi and M. Allah Bayoumi, "Computer attacks as machines of things that flow," *International Conference on Security and Management* (SAM'18), Las Vegas, 2018.
[37] S. Al-Fedaghi and N. Al-Huwais, "Toward modeling information in asset management: Case study using Maximo," 4th International Conference on Information Management (ICIM2018), Oxford, UK, 2018.
[38] S. Al-Fedaghi and N. Warsame, "Provenance as a machine," *International Conference on Information Society* (i-Society), Dublin, Ireland, 2018.
[39] S. Al-Fedaghi and M. Alsharah, "Modeling IT processes: A case study using Microsoft Orchestrator," in *Proc. 4th IEEE International Conference on Advances in Computing and Communication Engineering*, Paris, 2018.
[40] S. Al-Fedaghi, "User interface as a machine of things that flow," *The 2nd SERSC International Conference on Multimedia Technology and Human-Computer, Interaction 2018* (MTHCI 2018), Bangkok, 2018.
[41] S. Al-Fedaghi and M. Alsulaimi, "Re-conceptualization of IT services in banking industry architecture network," in *Proc. 7th IEEE International Conference on Industrial Technology and Management* (ICITM 2018), Oxford, UK, 2018.
[42] S. Al-Fedaghi and M. BehBehani, "Modeling banking processes," in *Proc. International Conference on Information and Computer Technologies* (ICICT 2018), DeKalb, IL, 2018.
[43] E. Matsas and G. C. Vosniakos, "Design of a virtual reality training system for human-robot collaboration in manufacturing tasks," *Int. J. Interact. Des. Manuf*., vol. 11, no. 2, pp. 1-15, 2017.
[44] N. A. Al-Hirz, M. J. Al-Sarraf and M. J Hussain, "A SLAM Project," Ph.D. dissertation, Compute. and Elect. Eng. Dept., American University of Kuwait, May 2015.
[45] S. Wen, K. M. Othman, A. B. Rad and Y. Zhang, "Indoor SLAM using laser and camera with closed-loop controller for NAO humanoid," *Abstract and Applied Analysis*, vol. 1, no. 1-8, July 2014.
[46] T. Looker, K. Brown, K. Turner and M. Walbran, "SLAM Robot Project," University of Auckland (no date). Available: http://homepages.engineering.auckland.ac.nz/~pxu012/mechatronics2015/group8.pdf



Sabah S. Al-Fedaghi is an associate professor in the Department of Computer Engineering at Kuwait University. He holds an MS and a PhD from the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, and a BS in Engineering Sciences (computer) from Arizona State University. He has published more than 320 journal articles and papers in conferences on software engineering, database systems, information ethics, privacy, and security. He previously worked as a programmer at the Kuwait Oil Company, where he headed the Electrical and Computer Engineering Department (1991–1994) and the Computer Engineering Department (2000–2007).

Manar J. AlSaraf is a research assistant in the Kuwait Institute of Scientific Research (2017 onwards). She holds a BS in Computer Engineering from The American University of Kuwait, Kuwait, and is currently a Computer Engineering MS student in Kuwait University. She previously worked as a Computer Engineer specialist for 6 months in Abu Dhabi (Abu Dhabi's Louvre museum project).