A Simulation-Based Architecture for Smart Cyber-Physical Systems

Thomas Gabor
LMU Munich

Lenz Belzner
LMU Munich

Marie Kiermeier
LMU Munich

Michael Till Beck
LMU Munich

Alexander Neitz
LMU Munich

***Abstract*—In order to accurately predict future states of a smart cyber-physical system, which can change its behavior to a large degree in response to environmental influences, the existence of precise models of the system and its surroundings is demandable. In machine engineering, ultra-high fidelity simulations have been developed to better understand both constraints in system design and possible consequences of external influences during the system's operation. These *digital twins* enable further applications in software design for complex cyber-physical systems as online planning methods can utilize good simulations to continuously optimize the system behavior, yielding a software architecture framework based on the information flow between the cyber-physical system, its physical environment and the digital twin model.**

I. Introduction

The advancing integration of software into complex machinery has been a major source for technical improvements recently. Many complex applications like smart energy grids or autonomous factories require a substantial amount of computational power to work as expected. The mutual dependency of hardware and software design is represented in the area of cyber-physical systems, whose capabilities to aid humans at a variety of tasks are still rapidly increasing [1], [2]. One driving force of such improvements is the advent of smart cyber-physical systems that are able to reflect on and improve their own behavior. Essentially, not only complex physical labor is being performed better by advanced machines, but also some mental labor can better be dealt with by computers on site. Tackling problems previously reserved for human minds has led programmers of smart cyber-physical systems into a territory where software needs to deal with the many restrictions of the physical world compared to the virtual one. For example, software controlling a cyber-physical system needs to have a basic understanding of the laws of physics impeding its movements or hindering its communication. But as software design has approached traditional design of machines for the physical world, so is the design process for physical machines closing up on the traditional process of software system design: As computers have been powerful enough to meaningfully understand the physical world, they have also become powerful enough to mimic and predict it or (more precisely) to simulate it.

II. Simulations in Engineering for smart cyber-physical systems

Simulations naturally play a huge role in modern engineering: They allow engineers to test designs and prototypes without spending excessive temporal and monetary resources on construction and manufacturing [3]. Instead, a computer simulation is relatively quick and cheap to deploy. The cost of failure is minimal, encouraging experimentation and creative thinking. Until recently, however, simulations were feasible only for certain parts of a complex system, for which analytic models existed, like, e.g., the air flow or the structural integrity. Newer increases in the computational power that is easily and widely available have enabled more complex simulations that integrate previously separate models of various aspects of structural design in order to accurately simulate the behavior of the system as whole. These ultra-high fidelity simulations are commonly called a *digital twin* with respect to the system they model [4], [5], [6]. Digital twins are characterized by their ability to accurately simulate events on different scales of space and time. In order to do so, they are not only based on expert knowledge, like for example an advanced physics simulation, but can also collect data from all deployed systems of their type and thus aggregate the experience gained in the field. One of the main challenges is, of course, to tweak the simulation software performance so that events in the digital twin can be simulated much faster than in the real world. Then, digital twins can not only be used during system design but also during run time in order to predict system behavior online. This can be helpful for fault prediction, but it is also a major asset to be exploited by simulation-based planning algorithms [7]. Using a digital twin, these can optimize system behavior based on its observed effects in the future. In this case, the availability of a digital twin mitigates the need for additional models of a "good outcome" of the system's actions: Instead, we can apply the system's general reward function to the configuration found in the simulation, which should be accurate enough to provide all parameters necessary to compute a reward estimation.

III. A Software Architecture for a digital twin system

It is straightforward to use a digital twin inside a software project just like one would use any other model: For example, a simulation is typically used in the early stages of development to argue the viability of the system to build by running it successfully in a virtual environment. Furthermore, an accurate simulation can be used to generate test cases for the system in order to determine when the deployed system behaves differently from the previously designed model, thus

978-1-5090-1654-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICAC.2016.29

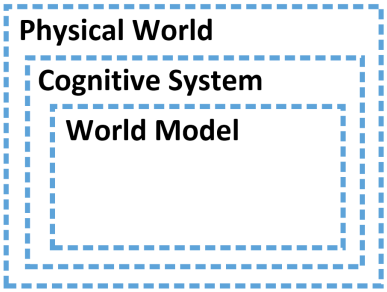

Fig. 1. Classical view of a simulation architecture: The world model used for the simulation is embedded inside the software system and controlled by a planning agent, for example.

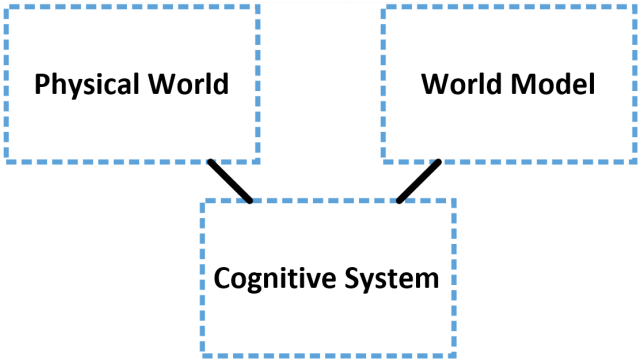

Fig. 2. The digital twin view of a simulation architecture: The cyber-physical system communicates with both its real world hardware and its simulation model using the same interface.

finding errors either in the simulation or the deployed system. It is clear that disposing of a high-fidelity simulation like a digital twin is quite beneficiary for these cases as well [4]. However, the digital twin not only brings a quantitative advantage in the form of improved test accuracy or fault detection, but it also features a unique qualitative advantage with regard to optimization and planning algorithms: Since the digital twin closely mimicks the physical world down to the smallest scales, it is also able to produce the same class of sensory input as the real world would produce and execute the same actions as the cyber-physical system's real actuators would execute in the physical world. This allows for a paradigm shift when it comes to integrating the logical model with the software architecture, as shown in Figures 1 and 2: Because the digital twin is able to implement the same interface as the sensors and actuators (or motors) used to interact with the physical world, it is possible to use the exact same action abstractions both for planning in the virtual space and for the execution of plans in the real space. Thus, from a software point of view, the sensorimotor controller connecting the cyber-physical system to its hardware components and the respective interface of the digital twin can be exchanged at will. Essentially, the difference between the physical world and the virtual world becomes transparent to the programmer of a smart cyber-physical system. The effort required in order to employ a digital twin is relatively high from a software engineer's perspective: High-fidelity simulations require both a huge amount of computational power to be evaluated during run time but usually also a huge amount of human labor to be developed. However, as previously discussed, such models are currently being developed by engineers in other fields in order to support design studies and machine maintenance. And the cost of computational power is still decreasing with technological advancements in hardware. On the other hand, having a simulation that can be interfaced with like the real world not only makes for a more elegant programming architecture, but mainly allows to test said simulation against real observations (and thus improve it) and allows for a general definition on how to integrate multiple world models that is then compatible with any version of a digital twin which may be plugged into the system.

A. *Integrating and Using a Digital Twin*

Figure 3 shows an instance model of a cyber-physical system. The Controller is the component that represents the interface to the physical world and thus the system's environment. It understands a predefined set of actions that can be executed on the physical actuators with the *sendAction* method. Event objects are generated by the Controller whenever it observes an update to its state, usually initiated by the system's sensors but also by the system's clock or other internal parts. Other components can subscribe to these events and are subsequently notified through the Controller's *publishTo* method. Commonly, the events observed directly through changes in sensor input are too subtle to be immediately useful to adjust the system's behavior. Thus, an *Aggregator* can be used to watch all the subtle events and abstract from the loads of data observed. In turn, an Aggregator can also publish Event objects to its subscribed observers but will usually be expected to do so less frequently.

The Cognitive System consists of components that basically react on Event objects by sending Action objects to the Controller. This may happen almost immediately, in which case we call the component an "intuitive" reactor meaning that the component is quick to compute a reaction to incoming events. However, the component may also first trigger a planning phase in order to compute a suitable reaction. The latter case is more interesting as it allows the system to behave more intelligently and is ultimately the main reason for variance and flexibility in system behavior in our smart cyber-physical system. There are multiple ways to implement planning into a cyber-physical system, but using the existing digital twin, it is natural to resort to *simulation-based planning*, i.e. planning methods relying on a model describing the whole environment from which test runs can be sampled.

One of the defining aspects of using simulation-based planning is that the simulation at work, in this case the digital twin, supports the same kinds of observations, i.e. events, and actions as the interface to the real world (represented by the Controller). In Figure 3, this shows as the digital twin component supports the *publishTo* and *sendAction* methods.

However, there is one main difference between the interfaces of the Controller and the digital twin: The flow of time can be controlled in the digital twin. This includes the fact that simulations in the digital twin can be started and halted by another software component, mainly the Planner, which is ultimately what makes the simulation useful for planning.[1] Furthermore, it is possible and often also quite advisable for a cyber-physical system to feature multiple digital twins. These may focus on different aspects of the environment and thus be used in different situations or may correspond to various internal components of the cyber-physical system. One may also instantiate multiple digital twins dynamically to account for uncertainty of the system about its own state: For example, when it is unclear which ones of several kind of motors are actually used in a factory robot, it may be the easiest way to simulate all possible kinds in order to avoid any risks. As discussed previously, the cyber-physical system may learn which motor it possesses by comparing the simulations from several digital twins to the observations made in the physical world over a certain period of time.

B. Defining an Architectural Framework for Digital Twins

Note that the instance diagram in Figure 3 is only an example of one possible configuration. In different cyber-physical systems, we would expect a different range and quantity of components. Typically, many different levels of intuitive reactors and planning components are combined to cover a wide range of possible situations and events. However, it is possible to generalize the architecture displayed in Figure 3; the resulting class diagram is shown in Figure 4. Both the Controller and the Digital Twin class inherit from the World superclass. This is what defines their ability to receive actions and produce events. On a more general level, every World can be regarded as an Event Source, to which any Event Observer can subscribe, thus forming a classic observer pattern [8]. It should be noted that Aggregators function as both Event Sources and Event Observers. Thus, multiple Aggregators can be constructed so that they form a hierarchy of information abstraction. For example, the Controller interacting with the physical world may publish events for every time its temperature sensor returns a reading of the current temperature. Due to random external circumstances (air flow, position relative to the sun, etc.), these readings may vary by some degrees even in a relatively short amount of time. In some scenarios, it may only be critical for the cyber-physical system to notice if the temperature consistently increases by several degrees over a longer period of time. An aptly configured Aggregator might thus observe every little event thrown by the temperature sensor (as part of the Controller) but only generate an event of its own when it notices the described pattern in temperature data. Other components of the system may then only subscribe to the Aggregator instead of the temperature sensor to hide superfluous information from their event interface. Lastly, Cognitive System is the broad term for any component that can observe events and produce actions, i.e. react to some kind of impulse. As previously mentioned, these components may come in different degrees of complexity. The simpler ones are called Intuitive Reactor while a Planner is able to produce more flexible behavior by optimizing its actions according to certain goals defined by the programmer of the system. To do so, the Planner is supposed to use the digital twin to predict the success of its plans before choosing a single plan to execute.

[1]Note that while this may seem a trivial property to achieve when considering simpler models, for sufficiently complex world models, there often is an upper limit as to what speed-up of time with respect to the physical world can be achieved while keeping the level of detail constant. Effectively, we may not have arbitrary control over time in the digital twin because we have limited computational resources.

Tier	Description
0	**physical necessity**: laws of nature
1	**machine-environment interface**: circuit-level control, sensor/actuator hardware, computational capabilties
2	**immediate reaction**: watchdogs, fixed behavioral rulesets, expert systems
3	**planned reaction**: reward functions for online planners and self-awareness

Fig. 5. Overview over the different levels of control present in the information flow of a cyber-physical system.

IV. Safety Engineering using a Simulation-Based Architecture

From an engineering point of view, the architectural framework presented in Figure 4 is on the more general side. It is supposed to be abstract enough so that it can be applied to a wide range of challenges for which smart cyber-physical systems are being developed. The central focus point of the proposed architecture is information flow: All behavior exerted by the cyber-physical system is triggered by some kind of event. Essentially, behavior is nothing more than a translation from observations to actions.

When dealing with complex smart cyber-physical systems, it is often difficult to predict in advance how exactly the system will behave. However, for basically all practical applications there need to be certain guarantees on the behavior of the system as it would be unsafe to deploy otherwise. Focusing on information flow helps us identify different levels of control present in the simulation-based architectural model. We can thus group different methods of exerting control on the behavior of a smart cyber-physical system into different tiers. These can be seen as different classes of constraints we can have our smart cyber-physical system comply to. For a quick overview over the classes we are about to discuss, refer to the table in Figure 5.

A. Physical Necessity (Tier 0)

Starting with tier 0, we are facing the odd candidate of these levels of control: In the most basic sense, the laws of the physical world the system operates in dictate a certain level of control. Our smart cyber-physical system is never free

to decide on actions that would not comply with the laws of physics, for example. Obviously, this seems like a rather trivial rule to model in our classification of control mechanisms, but it is not be neglected in the simulation part of the system. Since the digital twin is supposed to closely model reality, we have to explicitly model tier 0 control mechanisms for our simulation, even if they do not seem to directly temper with our planning process; we would not have to do this when using more abstract system models for planning. Thus, the need to accurately describe tier 0 constraints on system behavior is a unique consequence of using a high-fidelity simulation. Furthermore, there is no way to alter these control mechanisms when designing a system so the system developers need to understand and cope with what is preset by nature. Luckily, the phenomona of the physical world are usually well understood on the scales contemporary machines are dealing with and have been subject to simulation numerous times before.

B. Machine-Environment Interface (Tier 1)

Tier 1 summarizes control mechanisms placed at the system's interface with the environment. They are not enforced by anything but the hardware itself, but they can be regarded as part of the system's physical "body" to some extent. In contrast to tier 0 control, these mechanisms are usually created by the system developers deliberately. A typical example for a tier 1 control mechanism would be a circuit preventing a motor part from overheating by shutting down the engine. Many of these feedback loops are usually directly built into hardware and not exposed to software at all. This means that they cannot be altered online at run time, but are usually fixed once the respective parts are built. A rather delicate variant of this kind of control may happen with sensors and actuators: When a sensor can, for example, only provide a certain quality of data or only measure certain inputs, that can result in a severe constraint on the behavior of the cyber-physical system as it can no longer discern all different states of the physical world and is thus forced to treat situations similarily when they yield the same sensor data. The same effect can occur when regarding the precision with which actuators can comply to an action specification sent to them. In either case, it is clearly important to consider endowing the system with the necessary hardware so that it can reach solutions to all problems desired to be solved by the system. From a software architect's point of view, tier 1 control mechanisms mainly play a role when programming the Controller operating all the sensors and actuators.[2] However, since software has no means of altering tier 1 control dynamically, it must be informed about the tier 1 mechanisms by having them included in the digital twin's simulations. Thus, tier 1 control needs to be placed both at the interface with the real world, but also at the interface with the virtual world.

[2]While we have previously discussed issues on the hardware side, the Controller needs to accurately expose the level of precision in sensor and actor data that it can actually deliver and is thus usually dependend on the hardware design process as well. Also consider a system in which sensors produce loads of highly accurate data but the deployed software system is unable to process it because the amounts of computational power required would exceed the capacity the system's CPU, or worse, any CPU on the planet.

C. Immediate Reaction (Tier 2)

Tier 2 corresponds closely to the Intuitive Reactor in our system architecture. A classic example of tier 2 control would be to equip a self-driving factory robot with a software component that constantly checks the sensor events with respect to whether there is a human worker walking in the close vicinity. In that case, the Intuitive Reactor would cause the robot to stop by issuing the respective action. We expect the robot to do so immediately regardless of what it currently tries to achieve as once the Intuitive Reactor recognizes one of the situations it is looking out for, it is expected to know how to react correctly without weighing different options too much (hence "*intuitive* reactor"). Software components like these are also called *watchdogs* and we expect any sufficiently complex cyber-physical system to feature many of those. If situations may occur in which multiple Intuitive Reactors may trigger, it is up to the system designers to specify a reasonable order of precedence amongst them. In contrast to the lower tiers of control, tier 2 control mechanisms are part of the cyber-physical system's main information flow cycle. Thus, in order for them to work they depend on an at least rudimentally healthy system, which is also an argument why certain crucial control mechanisms should be placed at tier 1 when possible. However, tier 2 can also use the full set of features available to the system, usually including a variety of sensors and abstraction layers on observed data, i.e. Aggregators, which should aid them to make their decisions more precise. As Intuitive Reactors are completely deployed as software components, they are susceptible to online updates and may be adjusted during run time. For the system developer, it also follows that the system is not stuck with all the watchdogs it is deployed with and watchdogs can be removed if they no longer fit a new task the system is expected to fulfill or new ones can be added if a new tendency for dangerous behavior has been discovered, for example because lower or higher tier control mechanisms failed to work as expected in practical situations.

D. Planned Reaction (Tier 3)

The highest tier of control in a smart cyber-physical system results in a *planned* or *deliberate reaction* as opposed to an intuitive one. Tier 3 control mechanisms are carried out by the Planner or multiple Planners active in the cyber-physical system. Typically, tier 3 control is implemented by defining a reward function which the Planners try to optimize. When the Planners work successfully, this results in the system showing a tendency to behave in a way that maximizes the value of the reward function. Both hard and soft constraints can be implemented by defining the reward function accordingly. However, planners for sufficiently complex problems usually function non-deterministically and thus tier 3 control is always "a bit fuzzy" in nature, implying that it is mostly uncertain to which extent a set constraint will be fulfilled at a specific point in time. Nonetheless, in a smart system, the reward function

for the planner can be seen as the central definition of what the system is trying to achieve and thus tier 3 control mechanisms tend to have a big and broad impact on system behavior. For example, we may specify that a factory robot ought to maximize its throughput, causing the system to refrain from any self-damaging behavior whenever other options exist as this would be detrimental to its throughput. However, one can see why this mechanism is not suitable for safety concerns like never harming human collaborators, as it is not guaranteed that the planner recognizes other options. Obviously, tier 3 control will closely resemble the results of requirements engineering when building the system, however, it is relatively easy to change the reward function along the way and (in contrast to tier 2 control) often with little change to the code base. Thus, we can expect frequent, although oftentimes subtle, updates of tier 3 control mechanisms over the life time of a smart cyber-physical system.

V. Conclusion and Outlook

The concept of the digital twin has originally been developed to aid at the more precise design of machinery with respect to higher requirements of functionality and flexibility. Simulation supports rapid testing phases and the growing availability of computational power has made computer simulation more accurate and less expensive. In this paper, we motivated the use of the digital twin for online planning and presented an architectural framework centered around the information flow inside a cyber-physical system that incorporates the digital twin in a general and expandable way. We applied this analysis on information flow between components of a smart cyber-physical system to the engineering process by defining a classification of different methods of controlling system behavior with respect to the placement of said mechanisms inside the information flow.

Among the issues not yet covered by the presented architecture is the continuous improvement of the digital twin. As run time data is gathered by the operating cyber-physical systems as well as many similar cyber-physical systems, this data should be used to further improve simulation quality and adapt every single digital twin to possible changes occurring in its environment or its own main system. One example would be to account for material exhaustion by not only including a predictive model for that phenomenon in the digital twin, but also by checking how much decay has actually happened on the hardware side and restricting the digital twin to only simulate the probable consequences of the measured decay instead of calculating a model for all possible states of material degradation. In order to meaningfully incorporate knowledge about the system and its environment gained during the system's run time, it seems imperative to employ a capable model learning algorithm as an additional component into the system architecture. However, the analysis of the interplay between a learner for simulations and other parts of the system architecture is yet to be performed.

In line with the issue of adapting digital twins comes the problem of unifying multiple digital twins. As simulations may be altered during run time, the cyber-physical system may end up with multiple slightly different instances of digital twins, which is no concern as long as they agree on their prediction of future events. However, when they yield different simulation results, there must exist a method of reconciliation between multiple digital twins. For this purpose, a level of trust may be assigned to each digital twin and subsequently computed based on the factual prediction quality of the simulation derived from a comparison with the observed events in the environment. By doing so, inapt digital twins may be sorted out, at least eventually. Treating simulations as just another form of knowledge sources, this approach mimicks the teacher/student pattern described in [9].

Finally, it is still to be researched how the digital twin can best be integrated into the development process and the overall system life cycle respectively. As the digital twin needs to accurately simulate every part of system hardware and system hardware may only be determined after testing multiple prototypes we are left with a classic chicken-and-egg problem. Intuitive processes that are able to develop a digital twin alongside a system specification while still disposing of a consistent system at least at most times during the development processes are most definitely sought for.

References

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*. IEEE, 2008, pp. 363–369.

[2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.

[3] J. Morris, S. Zemerick, M. Grubb, J. Lucas, M. Jaridi, J. N. Gross, N. Ohi, J. A. Christian, D. Vassiliadis, A. Kadiyala *et al.*, "Simulation-to-flight (stf-1): A mission to enable cubesat software-based validation and verification," 2016.

[4] E. J. Tuegel, A. R. Ingraffea, T. G. Eason, and S. M. Spottswood, "Reengineering aircraft structural life prediction using a digital twin," *International Journal of Aerospace Engineering*, vol. 2011, 2011.

[5] E. H. Glaessgen and D. Stargel, "The digital twin paradigm for future nasa and us air force vehicles," in *53rd Struct. Dyn. Mater. Conf. Special Session: Digital Twin, Honolulu, HI, US*, 2012, pp. 1–14.

[6] A. Cerrone, J. Hochhalter, G. Heber, and A. Ingraffea, "On the effects of modeling as-manufactured geometry: Toward digital twin," *International Journal of Aerospace Engineering*, vol. 2014, 2014.

[7] L. Belzner, R. Hennicker, and M. Wirsing, "Onplan: A framework for simulation-based online planning," in *Formal Aspects of Component Software*. Springer, 2015, pp. 1–30.

[8] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, vol. 49, no. 120, p. 11, 1995.

[9] M. Hölzl and T. Gabor, "Continuous collaboration: a case study on the development of an adaptive cyber-physical system," in *Software Engineering for Smart Cyber-Physical Systems (SEsCPS), 2015 IEEE/ACM 1st International Workshop on*. IEEE, 2015, pp. 19–25.

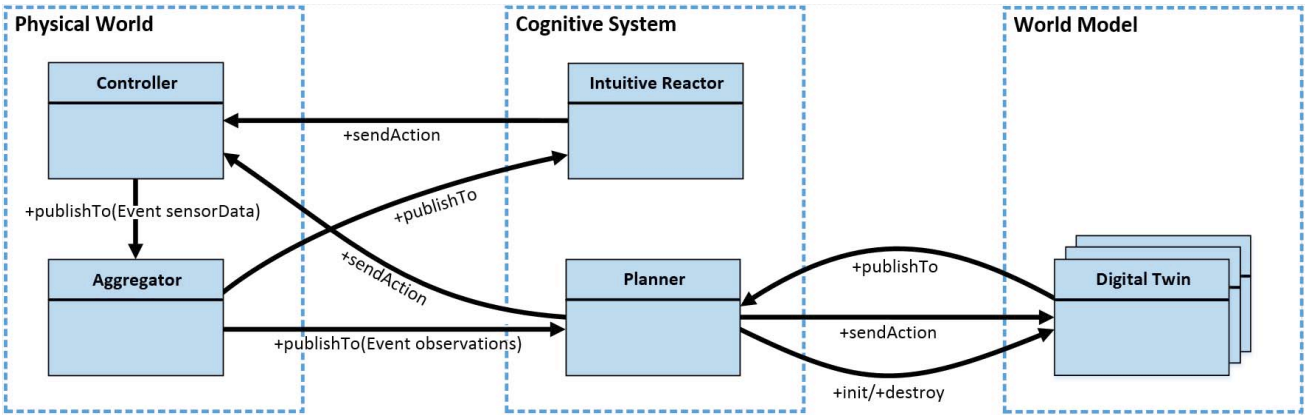

Fig. 3. Instance model of a software system using a digital twin. The dashed boxes mirror the setup shown in Figure 2.

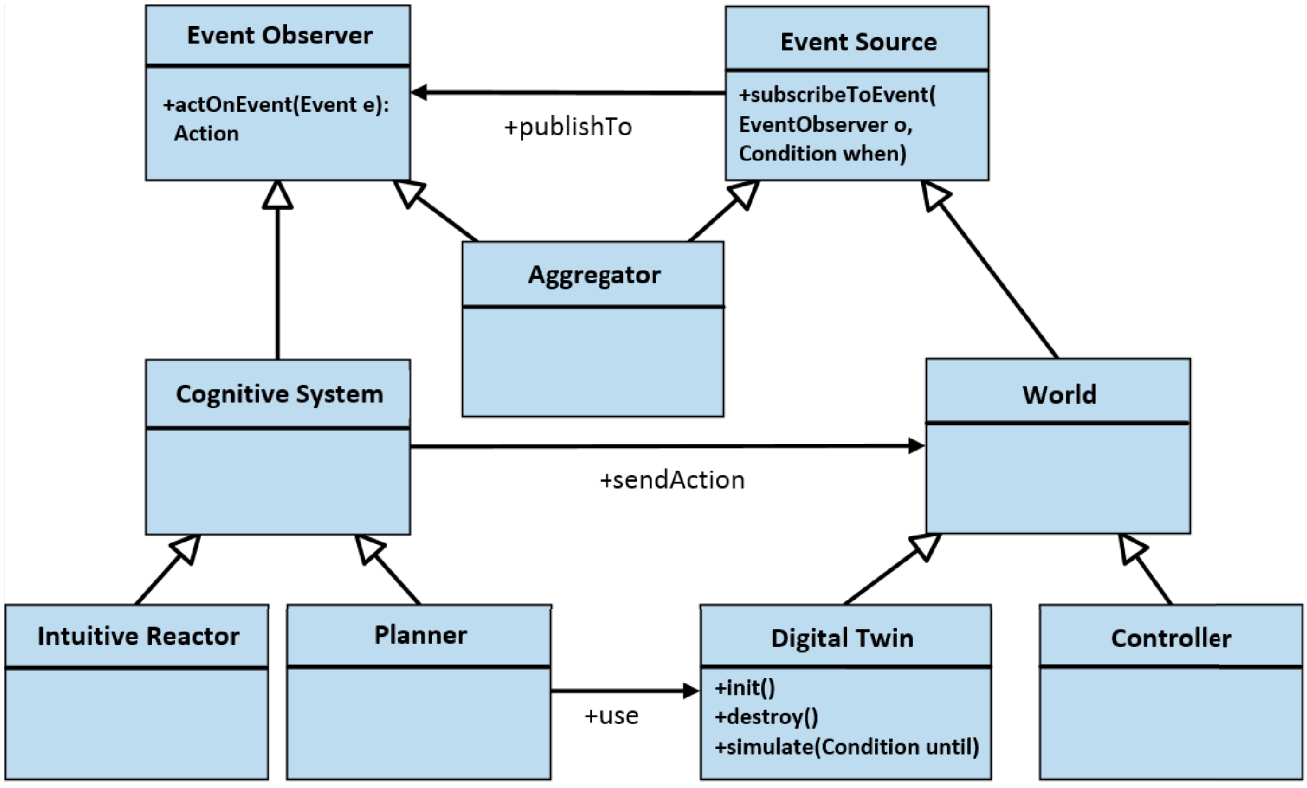

Fig. 4. Class diagram of a simulation-based architecture using a digital twin.