Developing a Physical and Digital Twin: An Example Process Model

Hao Feng
Cláudio Gomes
Michael Sandberg
Casper Thule
Kenneth Lausdahl
and Peter Gorm Larsen
DIGIT, Department of Electrical and Computer Engineering
Aarhus University
Åbogade 34, Aarhus N, DENMARK
Email: {haof,claudio.gomes,ms,casper.thule,lausdahl,pgl}@ece.au.dk

***Abstract*—As a result of the fast development of technologies and diversity of requirements, complexity exists not only in a system itself but also in the development process and environment. The complexity in the development process comes from specialization and concurrency, and a Digital Twin (DT) has the potential to mitigate these issues. We hope, during the process of building a DT, the accidental complexity can be minimized by application of model-based systems engineering and the Multi-Paradigm Modeling (MPM). In this paper, we give an example of the process of building a DT combined with a process model. Our long term goal is to model the development of various DTs to identify the commonalities and variations across multiple DTs.**

***Index Terms*—development process, digital twin, multi-paradigm modeling, process model, cyber-physical systems**

I. Introduction

It is not just that systems are complex [1]: their development process and environment are also complex. While systems comprised of many interacting, heterogeneous, components, are fundamental to our society [2], [3], we argue that the complexity in their development process should also be tackled. One can identify two main causes of complexity in the development process [4]: specialization and concurrency.

Specialization arises as our knowledge matures on each domain, opening new markets for supplier companies [5]. While there are clear benefits to specialization, the Original Equipment Manufacturer (OEM) has difficulties obtaining detailed descriptions of externally supplied components, due to Intellectual Property. In multi-paradigm modelling (MPM), every aspect of a problem is modelled explicitly, at the right level(s) of abstraction, using the most appropriate formalism(s) [6]. This paradigm, supported by model transformations [7], [8] and co-simulation [9], [10], embraces specialization and mitigates its effects.

We are grateful to the Poul Due Jensen Foundation, which has supported the establishment of a new Centre for Digital Twin Technology at Aarhus University. We are thankful for the discussions with Yuan Zhao, funded by Natural Science Foundation of Tianjin 20JCQNJC0039. Hao Feng also acknowledges support from China Scholarship Council.

The second cause of complexity in the development process, concurrency, arises out of the need to deliver products faster [11]. However, concurrent development processes require frequent communication among different teams, to avoid late integration problems [12], [13]. Here, process models play a fundamental role by facilitating communication, eliciting tool and tool integration requirements and risks, assessing maturity of companies, managing inconsistencies, and supporting automated workflows [14]–[16].

The concept of DT has emerged as a way to facilitate reasoning about a system's environment and to enable self-adaptation and optimization [17]. The sheer number of techniques to accomplish this goal makes the concept broad and interdisciplinary. As a result, there is no unifying and formal definition of what a DT is [18]–[20], but there are many commonalities among its features across a wide range of application domains. For example, the majority of DTs usually use one or more models, which is one of the common features of DTs. While a model plays an important role in a DT, it is not a DT. Usually, a DT is a system that integrates a model and different techniques increasing the value of the physical system, and a DT is synchronized with the physical system in real-time. These commonalities make it an ideal setting to focus on the modelling and engineering process of DTs.

Our vision is that the accidental complexity of building DTs should be minimized by application of model based systems engineering and the MPM paradigm. Process modelling is thus an important step in identifying the commonalities and variations across multiple DTs.

Contribution. This paper focuses on the development process followed in the development of an incubator system, while the components of a DT and detailed description can be found in the technical report [21]. We argue in [22] that such a system highlights the essential complexity in developing a DT, and that therefore the process exemplified here is also representative of a larger scale development. In addition, this paper does not try to solve the complexity problem but

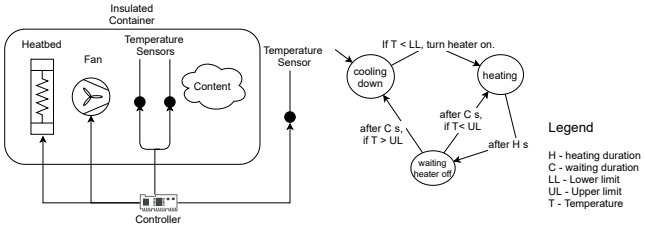

Fig. 1. Schematic overview of the incubator. The left is the incubator system. The right is the state-chart of the controller.

only an example process model which may help mitigate the complexity issue.

We recognize however some limitations to our contribution: we did not have experienced suppliers who could provide us with detailed specifications of the components. This makes it more difficult to develop first time right, because the initial models we built were not accurate (due to missing parameter values that had to be guessed from publicly available information). An experienced system integrator, together with suppliers, could produce accurate models to inform the design stage of the product. Instead, since cost was not a constraint, we assembled the prototype and then measured how accurate our initial models were.

The rest of the paper is organized as follows: in order to have an overview of the incubator system and its DT, section II introduces the end product of the incubator system and its DT. Next, section III presents the development process and the process model with details, including the comparison of different models. This is summarized in fig. 12, including the connections with each subsections. Finally, based on the incubator system, we give the discussion, challenges, and future work in section IV and we conclude in section V.

II. Incubator System and Its Digital Twin

In order to better understand the concept of DT, we will introduce the incubator Physical Twin (PT) first. Then we will present the incubator DT which comprises of multiple models of the PT and other auxiliary services.

A. Incubator Physical Twin

The incubator is a system with the ability to hold a desired temperature within an insulated container. The systematic diagram of the incubator is shown in Fig. 1.

As shown in Fig. 1, the incubator system consists of an insulated container, a heatbed, a fan, a controller, and three temperature sensors. The three temperature sensors are used for measuring the temperature inside the container and the room temperature. The fan is used to circulate the airflow inside the container to make the temperature uniformly distributed. The controller is similar to a bang-bang controller, reading the temperature and turning on/off the heatbed (a heating unit) and the fan. However, due to the delayed temperature propagation effect of the heatbed, the controller needs to wait after each actuation, to make sure the temperature does not rise too much. The state chart of the controller is in Fig. 1.

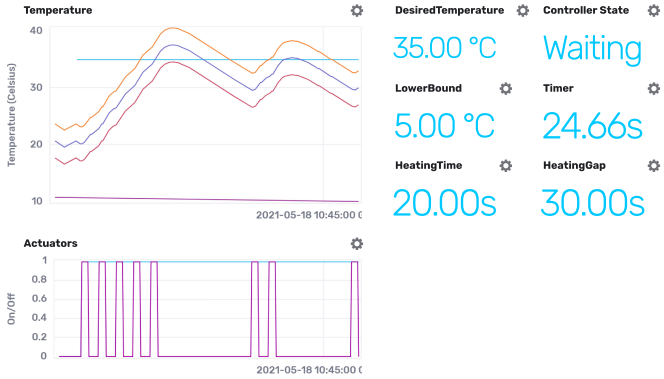

Fig. 2. Visualization interface of incubator PT. The horizontal axis represents time. The plots below shows the state the heatbed (purple line) and the fan (blue line). The plots above show the sensor temperatures and the desired temperature. The right section shows the parameters and states of the controller.

B. Incubator Digital Twin

Based on the incubator PT, we built multiple models of the incubator and services that increase the value of the incubator PT.

We consider a DT to be a set of services (that communicate with each other) and models (used by the services), whose goal is to provide feedback to the PT. In the following, we introduce some of these services and the models will be discussed in section III.

a) Data Storage: A crucial service in every DT is the collection and aggregation of data from several streams. This service is also responsible for allowing other services to access the data. Examples of possible tools are: InfluxDB and Kafka.

b) Data Visualization: This service helps an operator of a system to have a better understanding of the system because compared with the raw data, graph or other visual interface has the advantages in understanding information. In addition, this service becomes easier to be developed due to the recent advances in tools for creating visual interfaces such as Unity, Qt, Grafana, Dash, Gazebo, and so on. In the incubator, we used the InfluxDB because of its simplicity and powerfulness. The example of data visualization is shown in Fig. 2.

c) State Estimation: Most of the services in a DT rely on the states of the PT. However, it is not always feasible to obtain the states directly. Therefore, the need for state estimation in a DT arises. In the incubator, the air temperature can be measured directly from the temperature sensors while it is non-trivial to measure the temperature of the heatbed directly. In addition, the measurement data from the temperature sensors contain noise that covers the true state of the air temperature. Under these circumstances, it is necessary to estimate the states of the air temperature and the heatbed temperature. We used a Kalman Filter (KF) to do the state estimation. It combines the predicted behaviours produced by a dynamic model with the multiple sequential measurements from sensors, to form an estimate of the system's varying state, that is better than the estimate obtained by only using measurements.

d) Monitoring: Monitoring enables us to observe and evaluate the behaviours of the incubator as it operates. Due to the properties of the monitoring, we were able to develop the anomaly detection. Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behaviour [23]. Anomaly detection is not trivial due to noise, availability of labelled data, evolution of normal behaviours, and so on. We used a KF for basic anomaly detection by combining the measured behaviours from the PT and the predicted behaviours. While the incubator was in operation, we opened the lid as an anomaly, and the KF was successful in detecting the anomaly that is shown in Fig. 3.

e) What-If Simulation: What-if simulation is a data-intensive simulation whose goal is to inspect the behaviour of a physical system under some given hypotheses called scenarios [24]. This service would enable a human operator to try alternative interventions purely in a virtual setting to inspect what the consequences would be under some scenarios, *before* taking a final decision about what intervention would be best. For example, we used what-if simulation to determine the best parameters of *H* of the controller among four parameters set in the scenarios that we want the longest cooling time with a little tolerance of overheating.

f) Self-Adaptation: Self-adaptation is the ability of a computer system to change parts of its working algorithm over time [25], [26]. The goal of the self-adaptation is to make a DT be able to affect the PT automatically and step towards automation. We give, as an example, the following self-adaptation loop applied to the incubator. Those steps are finished automatically with the aid of the orchestration services inside a DT.

1) Starting when an anomaly is detected. This could for example be due to an object (e.g., a bucket of ice) being placed in the incubator. The KF and anomaly detector can be used to detect the changes in the system.
2) Schedule an experiment to gather relevant data. The nature of this experiment is application-specific, and design of experiments can be used for this [27].
3) Configure the controller to schedule the new experiment.
4) Gather experiment data, using the data recorder.
5) Run parameter estimation for new experiments.
6) Re-configure the KF with new parameters.
7) Run what-if simulations to optimize controller behaviours.
8) Re-configure the controller.

III. Process Model of Incubator and DT development

Section II has introduced the end product of the incubator and its DT. This section focuses on the development process, sketched in Fig. 12. In the following, we will focus on specific regions of the process model, and discuss them in more detail. Overall we applied the following formalism: algebraic equation, ordinary differential equations, and state machines.

A. Early 0D Models

In this stage of the development, we carried out the activities summarized in Fig. 4. We highlight the following:

0DModelling: We sketched the algebraic equations, with the goal of understanding the power requirements of the system. We collected some typical parameter values and wrote the equations that represent the energy increase of the system, assuming a fully insulated box.

CheckEnoughPower: Having the equation that relates the total energy in the box to the temperature, it is possible to calculate how much energy is needed to warm up the air from $25°\mathrm{C}$ to $35°\mathrm{C}$. The conclusion was that around $400\mathrm{J}$ were needed, and that a power supply of $100\mathrm{W}$ could provide that in about $4\mathrm{s}$.

B. Prototyping and Assumption Checking

As summarized in Fig. 5, in this stage, we assembled the system and ran some preliminary experiments with two goals:

Assumption Checking: To check how fast the obtained power supply could warm up the air inside the box, how uniform the temperature inside the box was (given the acquired fan), and how fast temperatures could be read from the temperature sensors.

Optimization: To optimize the placement of two of the three temperature sensors, in order to capture the average temperature in the box.

Fig. 6 shows the experimental setup and results, after finding a good enough position for the temperature sensors and the fan. As can be seen, the temperature is not uniform as the incubator warms up. The largest temperature difference is about $6°\mathrm{C}$. Moreover, it was concluded that 3 seconds was enough to read all three temperature sensors. This means that any incubator controller will have a sample time of at least 3 seconds. Finally, it can be seen that our initial estimate regarding the time it takes to increase the temperature by 10 degrees ($4\mathrm{s}$) was wildly incorrect. In practice, we observed that it took $100\mathrm{s}$ to warm up the air from $25°\mathrm{C}$ to $35°\mathrm{C}$. The results also show that one of the sensors (t1) could be removed, so that it can be re-used to measure the room temperature.

C. Plant & Controller Modeling

This stage, summarized in Fig. 7 consisted of mostly creating models at different levels of abstraction, and using them to fine tune the controller model. Aside from the algebraic equation models created in the early modelling stage, we created and calibrated a two-parameter model, a four-parameter model, a seven parameter model, and a Computational Fluid Dynamics CFD one. As shown later in section III-F, the two-parameter model is a very rough approximation of the system dynamics, but still useful for certain purposes, such as fine-tuning the controller parameters.

In addition, a hardware abstraction was created, so that the controller can be deployed and simulated in a real-time co-simulation together with an emulator of the plant, as described next.

D. Real-time Co-simulation

The real-time co-simulation was an important step in the development process of the incubator and its DT, because

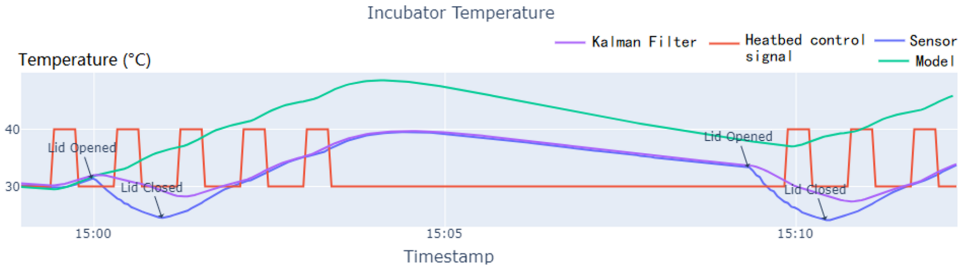

Fig. 3. Results of KF for anomaly detection.

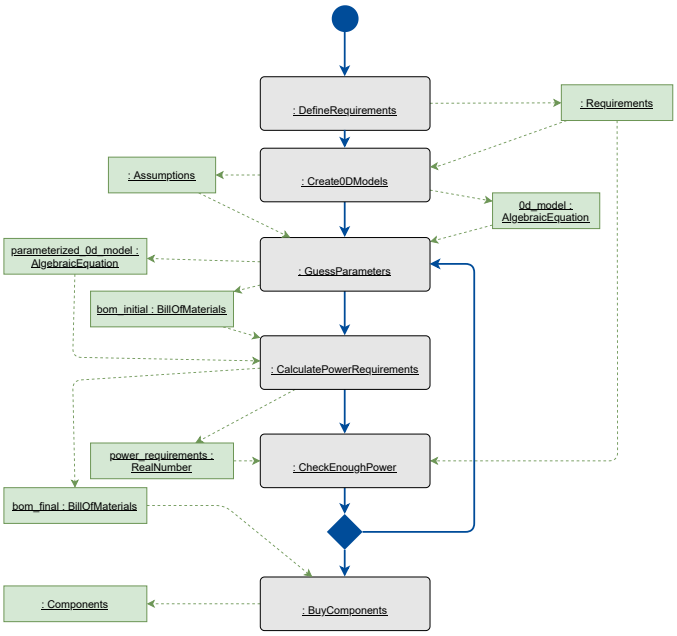

Fig. 4. Early Modelling Stage.

it is now used as "baseline" integration test, to try out new DT components, and ensure that existing ones are working correctly. The following steps were taken, summarized in Fig. 8:

Setup Timeseries Frameworks: To configure the data gathering with InfluxDB, and RabbitMQ, as the latter was the preferred method for communicating between DT components. This architecture was validated with the creation of a basic DT component: a KF.

Controller Implementation and Deployment: We created a plant emulator, which is simply a wrapper of a plant model, that is simulated in real-time and uses dependency injection to mimic the sensors and respond to actuation signals, so that the controller is tested in its production environment. Multiple failures modes for the controller were identified through the use of fault injection on the plant emulator. Counter measures were coded accordingly. For instance, if the temperature of the plant exceeds a maximum threshold, then the heatbed is turned off permanently, until the controller is restarted.

E. Digital Twin Component Development

The final stage of our development process, summarized in Fig. 9, focused on the iterative development of each of the DT Services. Thanks to the real-time co-simulation infrastructure developed earlier and thanks to the virtualization technology provided by Docker [28], we enabled continuous integration of DT services, supported by numerous tests and rapid development feedback. This also greatly enhanced safety, since there is less chance bugs can affect the real PT, and the fact that anyone can setup the development environment and run the tests means that remote collaboration is made possible.

The process diagram in Fig. 9 only shows the development of two DT services, for conciseness. The remaining services were developed in a similar iterative fashion.

F. Models, Complexity, and Their Relationship to the PT

This section gives the details of development of models.

While our early 0D models based on conservation of thermal energy in the system (section III) are a conceptually valid representation of the PT, a Digital Model needs to approximate state variables in time. This can be done with different levels of complexity.

1) 2-Parameter Model: To describe the time-dependent development in temperature of the incubator system, we again considered the conservation of thermal energy (section III). Now, however, we consider the rate of energy change to establish an Ordinary Differential Equation (ODE) that describes the development in temperature.

As noted, the temperature in the incubator system is not uniformly distributed. For this elementary simulation model, we assumed that the average of the signal from the two temperature sensors inside the incubator is representative as a temperature estimation of the full system, T_{system}. Accordingly, the thermal capacitance of the system was lumped into a single variable, C_{system}, that characterize the heater bed, the air inside the incubator, as well as the styrofoam box itself. As such, we assumed that the heat from the heatbed generated during warm-up would immediately be transmitted to the other parts of the incubator. The ODE of the 2-Parameter Model reads as follows:

$$\frac{dT_{system}}{dt} = \frac{1}{C_{system}}[VI - G_{box}(T_{system} - T_{room})]. \quad (1)$$

Similar to the early 0D models (section III), we modeled the energy that entered the system as the voltage times the

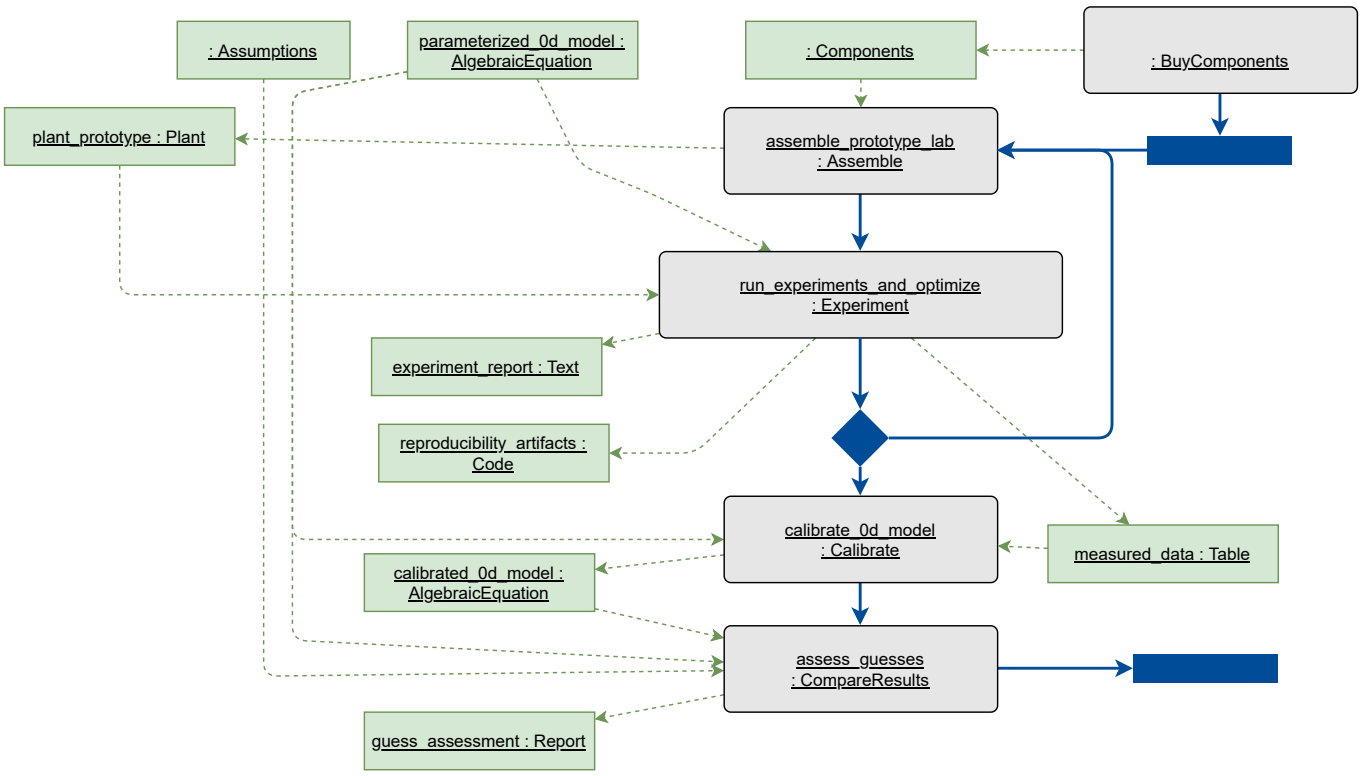

Fig. 5. Prototyping and Assumption Checking.

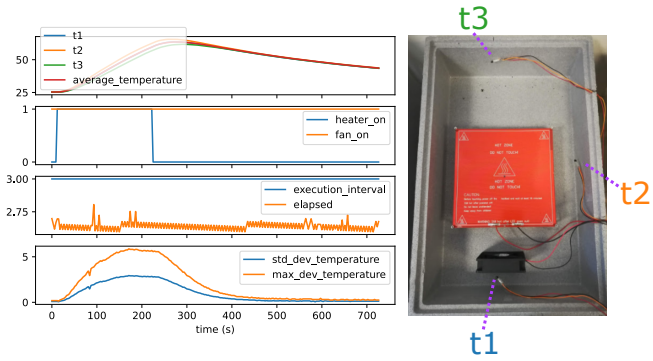

Fig. 6. Experimental setup and results.

ampere of the heatbed, but in the simulation model, this measure was now conditional based on the heatbed signal. In addition, the energy leaving the system was now considered. This was characterized as the temperature difference towards the ambient surroundings proportional to an effective heat transfer coefficient, G_{box}. Essentially, this approach lumped any convective cooling, conduction, and heat radiation towards the ambient surroundings into a single term. In conclusion, the 2-Parameter Model had two free parameters, C_{system} and G_{box}, that were calibrated with least-squares analysis using data from experiments.

The temperature response produced with the 2-Parameter Model is illustrated in Fig. 10 and compared to experimental data. As the figure shows, the development in temperature overall appeared to be well captured, but not without deficiencies. While the model predicted that the temperature of the system immediately dropped once the heatbed was turned off, the experimental measurements showed an intermediate, continued rise in temperature. This suggests that the approach of using a single temperature variable for the full system is insufficient since energy was obviously still transmitted from the heatbed after it was turned off.

2) 4-Parameter Model: To address the deficiency discussed of the 2-Parameter Model, the 4-Parameter Model was developed and it has the form of

$$\frac{dT_{heater}}{dt} = \frac{1}{C_{heater}}(VI - G_{heater}(T_{heater} - T_{bair})) \tag{2a}$$

$$\frac{dT_{bair}}{dt} = \frac{1}{C_{air}}[G_{heater}(T_{heater} - T_{bair}) - G_{box}(T_{bair} - T_{room})], \tag{2b}$$

where C converts the changing rate of temperature to energy and G determines the proportion of energy converted from the difference of temperatures. T_{heater} represents the temperature in the heatbed and T_{bair} the temperature of the air inside the incubator. The main difference with this model was that the temperature of the heatbed is now modeled explicitly, by adding an additional ODE to describe its temperature behavior. As (2) indicate, the thermal capacitance of the system was now split into the heatbed, C_{heater}, and the air inside the box, C_{air}. Essentially, the latter also includes the thermal capacitance of the styrofoam box itself. Finally, referring to (2), G_{heater} is an effective heat transfer coefficient that characterizes convective,

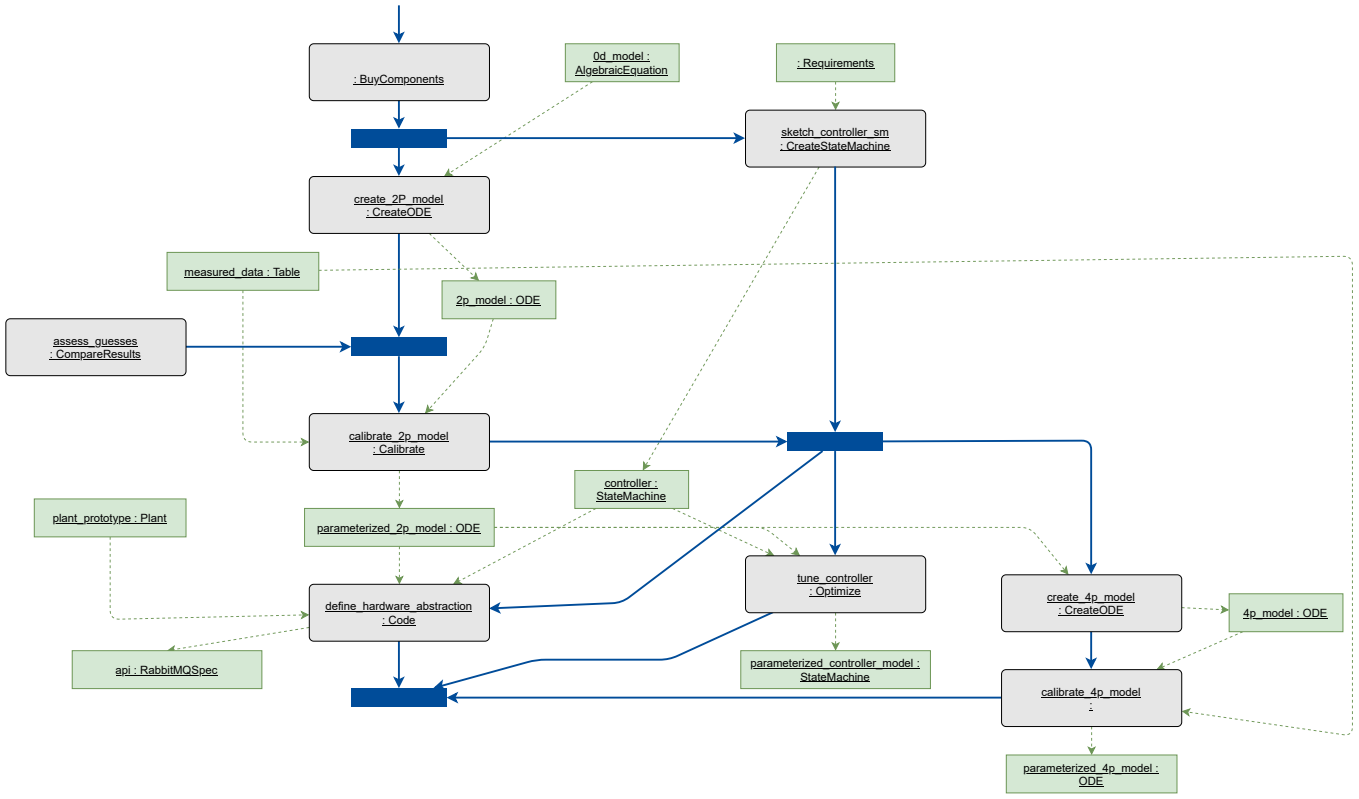

Fig. 7. Plant & Controller Modelling Process.

conductive, and radiative heat transfer from the heatbed to the rest of the system. Similar to the 2-Parameter Model, the four free parameters in the 4-Parameter Model were calibrated using least-squares.

As it can be seen in Fig. 10, the temperature response now appears better captured. After the heatbed was turned off, the model now predicted an intermediate, continued temperature rise, and overall, the simulated behavior followed the experiments closely. However, as the lid was opened later in the experiment (indicated in Fig. 10), the predicted temperature from the simulation model started to drift away from the experimental data.

3) 7-Parameter Model: If the lid of the incubator is opened, the resistance for the thermal energy to escape the system is significantly lowered. To include this behavior, we made the effective heat transfer coefficient of the incubator, G_{box}, conditional on whether the lid was open or not. In order to enable the "Self-Adaptation" service introduced in section II-B, we also modeled a foreign object being placed inside the incubator (e.g., an ice bucket, etc.). Similar to our approach for modeling the heatbed, we added an additional ODE for the temperature of the foreign object, T_{fo}, as well as an effective heat transfer coefficient, G_{fo}, describing the energy exchange between the foreign object and the air inside the incubator. In total, the 7-Parameter Model includes three ODEs with in total seven free parameters:

$$\frac{dT_{heater}}{dt} = \frac{1}{C_{heater}}(VI - G_{heater}(T_{heater} - T_{bair})) \quad (3a)$$

$$\frac{dT_{bair}}{dt} = \frac{1}{C_{air}}[G_{heater}(T_{heater} - T_{bair}) - G_{box}(T_{bair} - T_{room}) - G_{fo}(T_{bair} - T_{fo})] \quad (3b)$$

$$\frac{dT_{fo}}{dt} = \frac{1}{C_{fo}}G_{fo}(T_{fo} - T_{bair}) \quad (3c)$$

While we will not discuss temperature behavior related to placing foreign objects inside the incubator in this study, Fig. 10 shows that 7-Parameter Model is capable of capturing the temperature response related to opening the lid, provided that the timing of opening and closing the lid are known.

4) Computational Fluid Dynamics model: The final level of complexity that we considered in this study was a full CFD model of the system. In a CFD model, partial differential equations describing the behavior of the components in both time and space are solved, which means that this approach does not rely on model reduction techniques. As such, one can say that it gives the full picture instead of the temperature at a few selected points. The CFD model in this study was based on the solution of the Navier-Stokes equations assuming incompressible, non-isothermal flow and solved using the commercial finite element software COMSOL Multiphysics. A selected frame from the simulation can be seen in Fig. 11

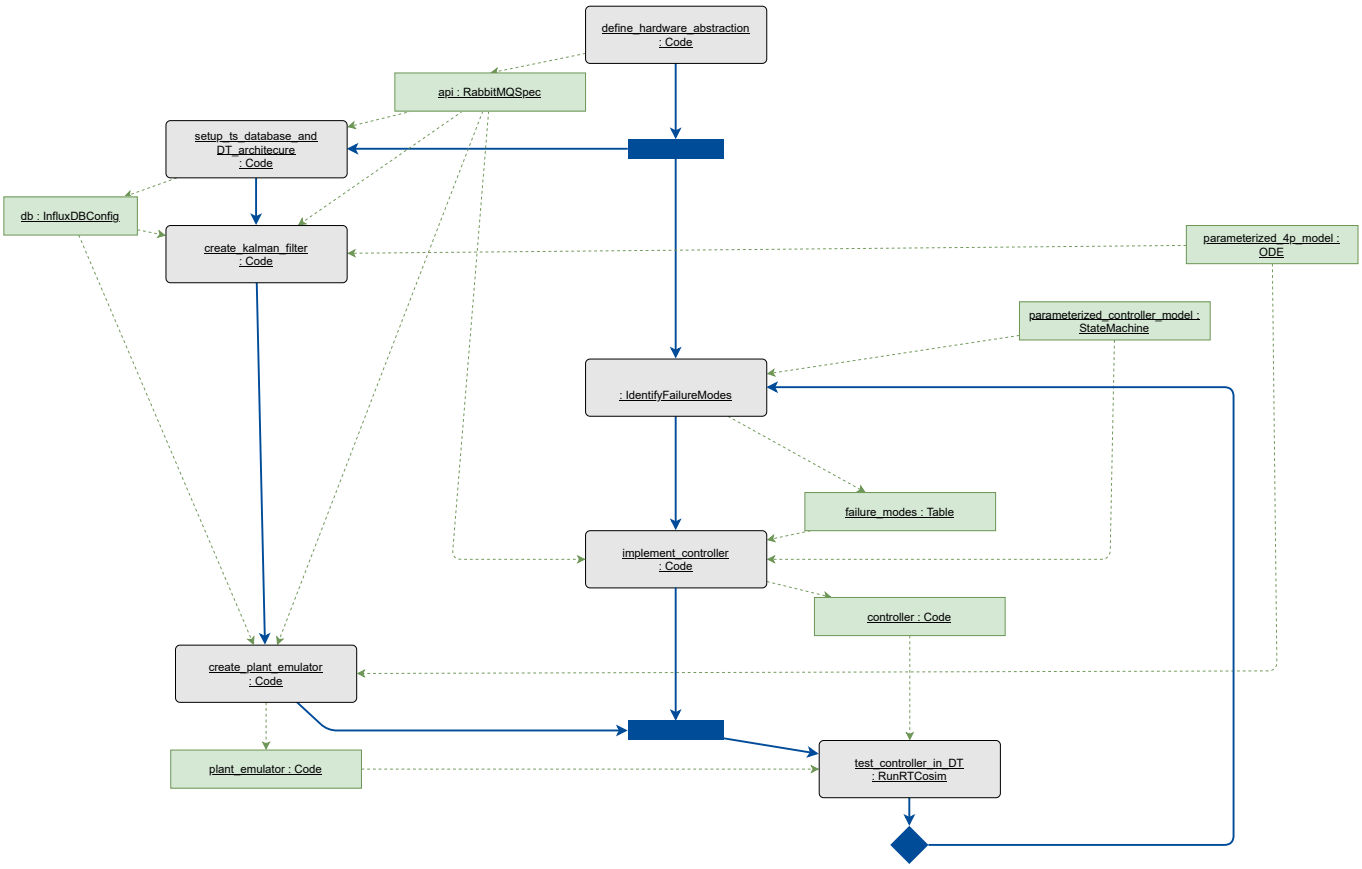

Fig. 8. Real-time Co-simulation Process.

(b). The frame shows that the temperature is, in fact, not evenly distributed. For example, since the airflow over the heater bed is uneven, the temperature is higher in some local areas, and as the air is circulated inside the incubator, it is gradually cooled down.

The higher fidelity of the CFD model comes at a cost. While the reduced 2,4, and 7-Parameters Models involved one to three state variables, the CFD model involved the solution of approximately 100 000 unknowns. This comes from the hexahedral mesh-discretisation (Fig. 11 (a)) that must be applied in order to obtain a numerical solution to the governing equations. Consequently, the computation time needed to advance the simulation model in time is roughly 100 000 times longer (compared to, e.g., the 2-Parameter Model), which makes a full-blown CFD approach unsuitable for anything close to real-time analyses.

IV. Discussion, Challenges, and Future Work

The development of the Incubator twin system has been carried out in a straightforward fashion in order to gain knowledge of the domain of DT engineering. This has led to a stage where it might be possible to see common traits that can be reused and assist in defining a methodology. One of the challenges is the process of verifying the fidelity of a model of a piece of hardware, i.e. the plant model of the incubator. Such an experiment can be carried out by profiling the hardware and subsequently issuing a calibration procedure in order to tune model parameters. If the process converges in the sense that the model has a sufficient fidelity, then it has finished and parameters of the model have been uncovered. However, if not, then the model potentially has to be transformed, as the case was with the 2, 4, and 7-parameter plant model of the incubator, repeating the whole process.

Regarding the first time right approach, due to the unreliable suppliers and components we worked it, our initial models were wildly inaccurate. However, after the first prototype and calibration, we were able to quickly tune the controller and reuse the calibrated models to implement a PT emulator, which relies on co-simulation [10] and speeds up the development of the DT. This emulator is important not just for training operators, but also for the development environment of the DT services: thanks to virtualization technology [29], it is possible to automatically setup the development environment and run integration tests. This leads to higher quality DT services and increased safety.

In the future, we plan to integrate units with co-simulation using the Functional Mockup Interface (FMI) standard [30], [31] with some of the services in the DT, and apply the lessons from the Incubator to other DT case studies, i.e. a Desktop Sized Robot [32] and extend on the gained knowledge

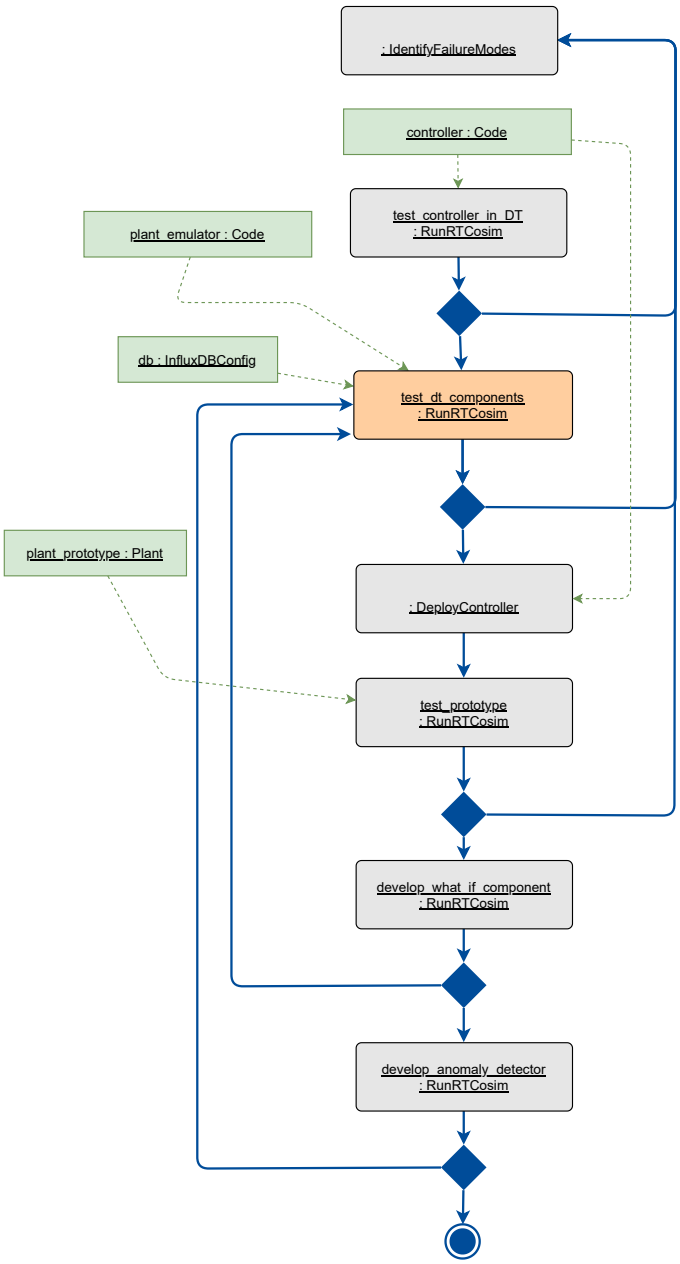

Fig. 9. DT Component Development Process.

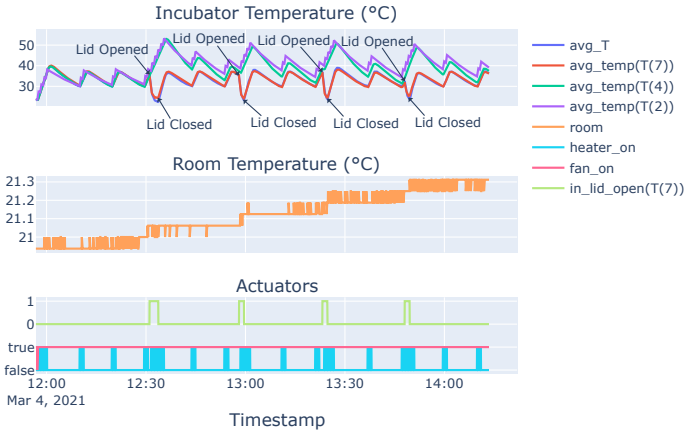

Fig. 10. The temperature response of the 2, 4, and 7-Parameter Models compared to the experimental measurements.

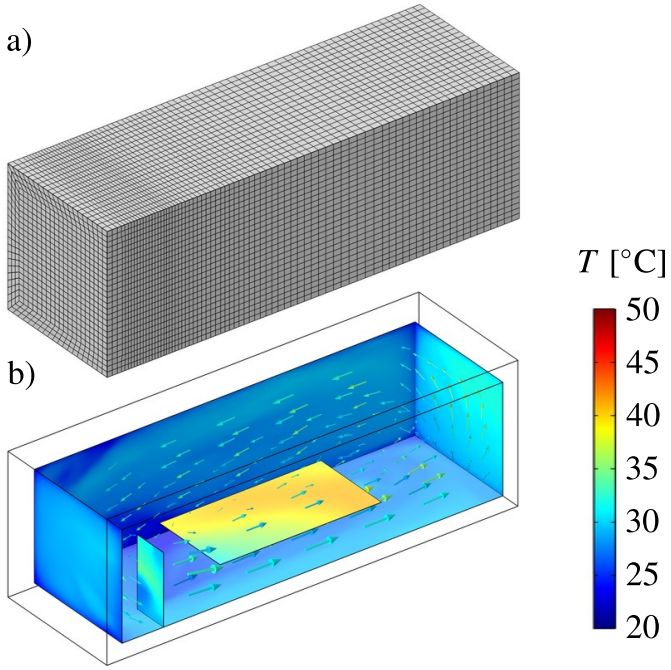

Fig. 11. (a) Mesh-discretisation of the CFD model. (b) Temperature distribution inside the incubator at a selected time instance approximated using CFD simulations.

for subsequent DT development. This will allow for quickly prototyping new DT services, that can be integrated as FMUs [33], using a contract-based approach to ensure correctness [34]–[36].

Much of this work is expected to be concerned with a tool-supported methodology for DT development. In this regard Formalism Transformation Graph + Process Model process modelling language plays a fundamental role.

V. Conclusion

In this paper, we first demonstrated the end product of the incubator system and its DT development, for an overview. Then we presented the development process model of the incubator and its DT, including the development of models, controller, co-simulation, and DT component development from its beginning stage. We report on the development of *one* DT. We, therefore, do not have yet the experience to discuss the impact of the process model on/to modelling steps or their complexity. During the development process, we realized that it might be possible to see common traits that can be reused and assist in defining a methodology. We hope that we can apply the lessons from the incubator system to other DTs development. Finally, we would like to mention that at a high level, the development process is consistent with the traditional iterative v-process: low fidelity modelling,

high fidelity modelling, simulation, component development, physical experiments, etc.

REFERENCES

[1] H. Vangheluwe, "Foundations of Modelling and Simulation of Complex Systems," *Electronic Communications of the EASST*, vol. 10, 2008.

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

[3] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions," *ACM Computing Surveys*, vol. 48, no. 2, pp. 18:1–18:41, 2015.

[4] T. Tomiyama, V. D'Amelio, J. Urbanic, and W. ElMaraghy, "Complexity of Multi-Disciplinary Design," *CIRP Annals - Manufacturing Technology*, vol. 56, no. 1, pp. 185–188, 2007.

[5] K. B. Clark, "Project scope and project performance: The effect of parts strategy and supplier involvement on product development," *Management science*, vol. 35, no. 10, pp. 1247–1263, 1989.

[6] H. Vangheluwe, J. De Lara, and P. J. Mosterman, "An introduction to multi-paradigm modelling and simulation," in *Proceedings of the AI, Simulation and Planning in High Autonomy Systems Conference*. Lisbon, Portugal: Society for Computer Simulation International, 2002, pp. 9–20.

[7] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Software*, vol. 20, no. 5, pp. 42–45, Sep. 2003.

[8] C. Gomes, B. Barroca, and V. Amaral, "Classification of Model Transformation Tools: Pattern Matching Techniques," in *Model-Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science, J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, Eds., vol. 8767. Springer International Publishing, 2014.

[9] R. Kübler and W. Schiehlen, "Two Methods of Simulator Coupling," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 6, no. 2, pp. 93–113, 2000.

[10] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A Survey," *ACM Computing Surveys*, vol. 51, no. 3, pp. 49:1–49:33, 2018.

[11] J. Buur *et al.*, "Mechatronics design in Japan," Ph.D. dissertation, Institute for Engineering Design, Technical University of Denmark (DTH), 1989.

[12] C. W. De Silva, *Mechatronics: An Integrated Approach*. CRC press, 2004.

[13] R. Plateaux, J. Choley, O. Penas, and A. Riviere, "Towards an integrated mechatronic design process," in *Proceedings of the 2009 IEEE International Conference on Mechatronics*, vol. 00. Malaga, Spain: IEEE, 2009, pp. 1–6.

[14] S. Mustafiz, J. Denil, L. Lúcio, and H. Vangheluwe, "The FTG+PM framework for multi-paradigm modelling," in *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling - MPM '12*. New York, New York, USA: ACM Press, 2012, pp. 13–18.

[15] M. Challenger, K. Vanherpen, J. Denil, and H. Vangheluwe, "FTG+ PM: Describing Engineering Processes in Multi-Paradigm Modelling," in *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Springer, Cham, 2020, pp. 259–271.

[16] I. Dávid, E. Syriani, C. Verbrugge, D. Buchs, D. Blouin, A. Cicchetti, and K. Vanherpen, "Towards inconsistency tolerance by quantification of semantic inconsistencies," in *1st International Workshop on Collaborative Modelling in MDE*, vol. 1717, 2016, pp. 35–44.

[17] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*. Cham: Springer International Publishing, 2017, pp. 85–113.

[18] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, Jan. 2021.

[19] A. Rasheed, O. San, and T. Kvamsdal, "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[20] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[21] H. Feng, C. Gomes, C. Thule, K. Lausdahl, M. Sandberg, and P. G. Larsen, "The Incubator Case Study for Digital Twin Engineering," *arXiv:2102.10390 [cs, eess]*, Feb. 2021.

[22] H. Feng, C. Gomes, C. Thule, K. Lausdahl, A. Iosifidis, and P. G. Larsen, "Introduction to Digital Twin Engineering," in *The Annual Modeling and Simulation Conference*, Virginia, USA, 2021, p. to appear.

[23] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[24] S. Rizzi, "What-if analysis." 2009.

[25] P. Zhou, D. Zuo, K. Hou, Z. Zhang, J. Dong, J. Li, and H. Zhou, "A Comprehensive Technological Survey on the Dependable Self-Management CPS: From Self-Adaptive Architecture to Self-Management Strategies," *Sensors*, vol. 19, no. 5, p. 1033, Feb. 2019.

[26] D. Weyns, "Software Engineering of Self-adaptive Systems," in *Handbook of Software Engineering*, S. Cha, R. N. Taylor, and K. Kang, Eds. Cham: Springer International Publishing, 2019, pp. 399–443.

[27] L. Pronzato and A. Pázman, *Design of Experiments in Nonlinear Models*, ser. Lecture Notes in Statistics. New York, NY: Springer New York, 2013, vol. 212.

[28] J. Turnbull, *The Docker Book: Containerization Is the New Virtualization*. James Turnbull, 2014.

[29] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[30] A. Junghanns, T. Blochwitz, C. Bertsch, T. Sommer, K. Wernersson, A. Pillekeit, I. Zacharias, M. Blesken, P. Mai, K. Schuch, C. Schulze, C. Gomes, and M. Najafi, "The Functional Mock-up Interface 3.0 - New Features Enabling New Applications," in *Proceedings of the 14th International Modelica Conference*. online: Linköping University Electronic Press, Linköpings Universitet, 2021, p. to be published.

[31] C. Gomes, M. Najafi, T. Sommer, M. Blesken, I. Zacharias, O. Kotte, P. Mai, K. Schuch, K. Wernersson, C. Bertsch, T. Blochwitz, and A. Junghanns, "The FMI 3.0 Standard Interface for Clocked and Scheduled Simulations," in *Proceedings of the 14th International Modelica Conference*. online: Linköping University Electronic Press, Linköpings Universitet, 2021, p. to be published.

[32] G. Lumer-Klabbers, J. O. Hausted, J. L. Kvistgaard, H. D. Macedo, M. Frasheri, and P. G. Larsen, "Towards a digital twin framework for autonomous robots," in *SESS: The 5th IEEE International Workshop on Software Engineering for Smart Systems*, COMPSAC 2021. IEEE, July 2021.

[33] C. M. Legaard, C. Gomes, P. G. Larsen, and F. F. Foldager, "Rapid Prototyping of Self-Adaptive-Systems using Python Functional Mockup Units," in *Proceedings of the 2020 Summer Simulation Conference*, ser. SummerSim '20. Virtual Event, Spain: Society for Computer Simulation Internationa, San Diego, CA, United States, 2020, pp. 1–12.

[34] S. T. Hansen, C. Gomes, P. Larsen, and J. van de Pol, "Synthesizing Co-Simulation Algorithms with Step Negotiation and Algebraic Loop Handling," in *The Annual Modeling and Simulation Conference*, Virginia, USA, 2021, p. to appear.

[35] E. O. Inci, C. Gomes, J. Croes, C. Thule, K. Lausdahl, W. Desmet, and P. G. Larsen, "The effect and selection of solution sequence in co-simulation," in *The Annual Modeling and Simulation Conference*, Virginia, USA, 2021, p. to appear.

[36] C. Thule, C. Gomes, and K. Lausdahl, "Formally Verified FMI Enabled Data Broker: RabbitMQ FMU," in *Proceedings of the 2020 Summer Simulation Conference*, ser. SummerSim '20. Virtual event: Society for Computer Simulation International, 2020, pp. Pages 1–12.

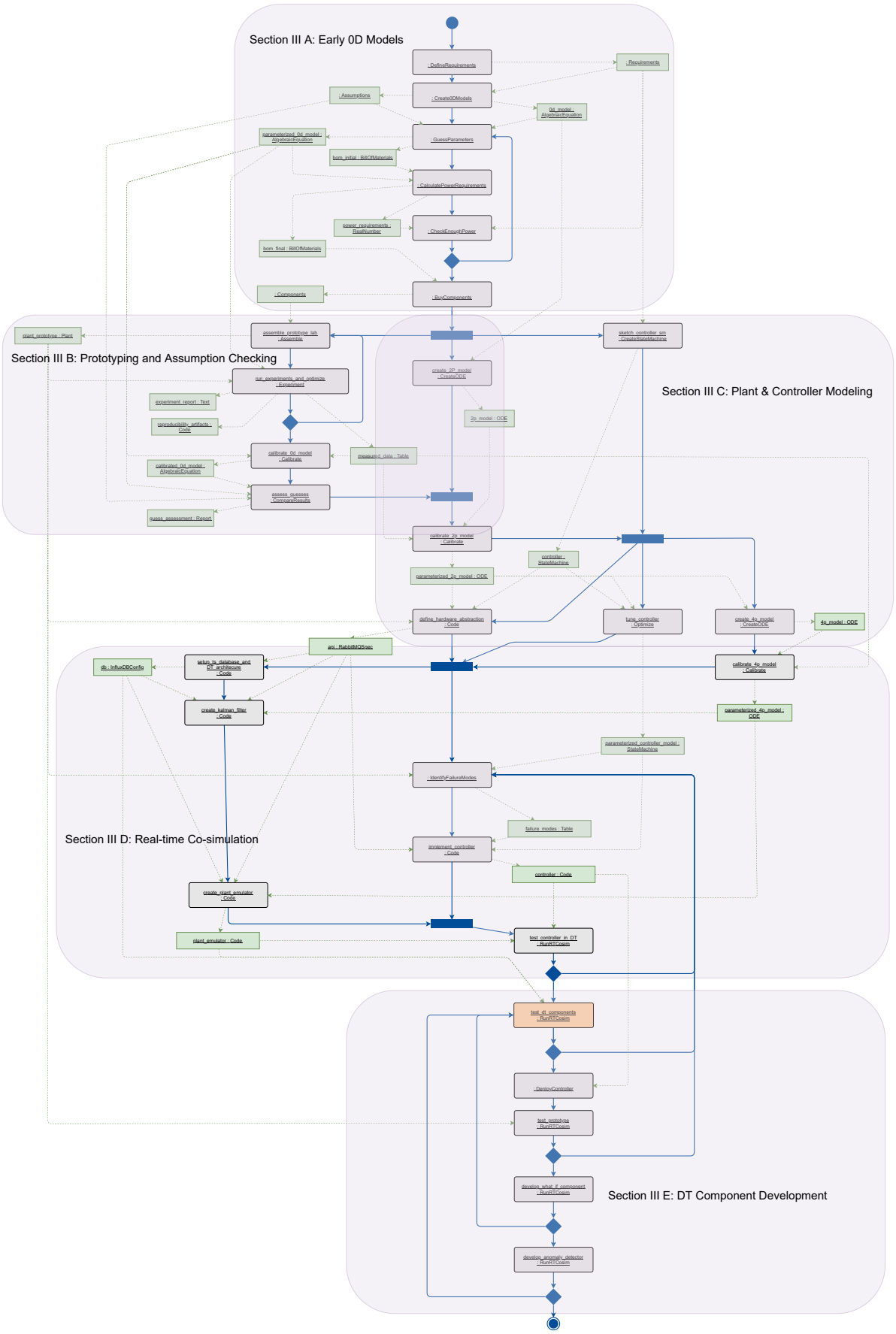

Fig. 12. Sketch of the incubator development process.