On Early Statistical Requirements Validation of Cyber-Physical Space Systems

Christos Tsigkanos
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

Nianyu Li
Key Laboratory of High Confidence Software Technology, Peking University, China

Zhi Jin
Key Laboratory of High Confidence Software Technology, Peking University, China

Zhenjiang Hu
National Institute of Informatics, Graduate University for Advanced Studies, Japan

Carlo Ghezzi
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

ABSTRACT

Cyber-physical space systems are becoming increasingly important. Such systems have to satisfy requirements that are heavily affected by the physical space they operate in and by the active entities inhabiting the space, whose dynamic behaviors generate continuous topological changes. Reasoning about requirements in the early design phases is extremely challenging. High-level design can be facilitated by systematic application of separation of concerns throughout modeling, analysis, and early requirements validation. We outline an approach that identifies key recurrent concerns arising in cyber-physical space systems, supports systematic and semi-automatic modeling of separate concerns, and a formally defined composition of the separately developed models. Early requirements validation is then supported by leveraging statistical model checking techniques. We illustrate our approach through an example disaster scenario in a smart city.

ACM Reference Format:
Christos Tsigkanos, Nianyu Li, Zhi Jin, Zhenjiang Hu, and Carlo Ghezzi. 2018. On Early Statistical Requirements Validation of Cyber-Physical Space Systems. In *SEsCPS'18: SEsCPS'18:IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3196478.3196485

1 INTRODUCTION

Reasoning about requirements validation is a critical software engineering activity. Complex ubiquitous cyber-physical systems often have to satisfy requirements that are heavily affected by the physical space they operate in, in addition to other system concerns; this has been identified as a future challenge in the domain of smart cyber-physical systems [7]. Examples of this vision are often referred to with the evocative terms *smart building* or *smart city*. In such systems, which we collectively call *cyber-physical space systems–CPSSs*, computational and physical elements function and interact in a manner contingent on physical laws and the active entities inhabiting them and performing actions exhibiting complex dynamic behaviors that are both space- and time-dependent.

SEsCPS'18, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5728-9/18/05…$15.00
https://doi.org/10.1145/3196478.3196485

Software engineering techniques for validation of complex systems can range from plain testing to formal verification, each obtaining different levels of assurance. The type and complexity of a system as well as the form of requirements that are sought to be validated are important concerns that dictate different techniques. Our approach concerns high-level reasoning within the critical system requirements phase of the design of CPSSs, where a way to validate system behaviors is highly sought. We focus on early requirements validation, before implementing and deploying the actual system. We position our approach within engineering of dependable systems operating in a discrete space arising from topological relations in the spatial environment, where the information abstraction of physical *location* or position of entities is inherently important [19, 24]. Novel software engineering research problems arise to support high-level design and early requirements validation of complex CPSSs. Because of their scale and complexity, separation of concerns has to be applied systematically in design and validation.

In this paper, we outline a systematic approach to high-level modeling of CPSSs via separation of key recurrent concerns and a formally defined composition of models that capture them. Our contribution consists of a framework integrating existing model-based analysis techniques for the engineering of CPSSs through (partial) automation of model construction and composition. This enables early requirements validation by reasoning at the model level. Existing analysis techniques mostly focus on formal verification [25]. In several practical cases, however, such techniques do not scale to a required level needed for large spatial systems inhabited by many active components. We argue instead that statistical model checking methods [18] can provide the required analysis support for those systems. Such methods may not offer assurances like formal verification, but can provide valuable insights early in the design process.

We treat physical activities, interaction and requirement objectives of entities operating in a cyber-physical space system as separate design concerns. Regarding physical activities, we assume that a model of the physical space exists in some accessible digital form, e.g. BIM [12] or CityGML [15] models for a building or a city; this can be used as a source model for a model-transformation step that automatically develops a graph-based formally analyzable model enjoying well-defined semantics [24, 26]. Besides static topology, the process takes into account dynamics generated by the active entities that operate in the space [25], yielding representations of entity physical behaviors. Requirements lead to the identification of certain states as success or failure states. A success state indicates that a certain objective is achieved by an entity while a failure state its violation. Entity interaction models provide a high-level description of the distributed controller which coordinates entities.

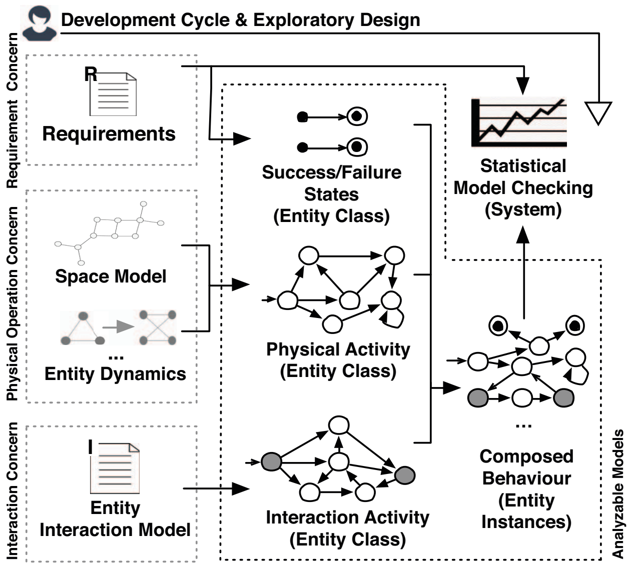
Models capturing system concerns are semi-automatically formally composed into composed behaviors, i.e., automata equipped with transition guards, invariants and probabilistic features. Subsequently, we propose the application of state-of-the-art statistical model checking techniques [6] for validation. In essence, activity models are used to generate execution traces upon which statistical model checking is used to produce *statistical evidence* about the system's satisfaction or violation of a requirements specification. We showcase our modeling and analysis framework with a disaster scenario situated in a smart city.

The rest of the paper is structured as follows. Section 2 presents an overview of our approach to early statistical requirements validation of cyber-physical space systems. Section 3 pertains modeling of system concerns, yielding composite automata capturing system entities, while Section 4 illustrates early requirements validation through statistical model checking. Related work is considered in Section 5 and Section 6 concludes the paper.

2 DESIGN PROCESS: EARLY VALIDATION

Design concerns common to most CPSSs include (1) modeling the active behaviors of entities operating in the physical space, (2) modeling how they interact, and (3) specifying requirements of the overall system. This set of concerns is similar across applications of this kind; it is desirable to model these concerns separately and then be allowed to compose them by following a well-defined rigorous method. This is crucial within the development cycle, as the system designer can change each criterion independently and evaluate its effect on system goals early in the process. As illustrated in Fig. 1, our approach supports separation of key recurrent concerns and also composition of models that capture them, leading to behavioral models that describe entities operating within a cyber-physical space system. Early requirements validation is then enabled through statistical model checking.

Running Example. As a motivating example showcasing our approach, consider a disaster recovery scenario in a smart city where emergency response is provided by autonomous Unmanned Aerial Vehicles (UAVs) [28], which are used to locate and rescue victims. The city is comprised of connected buildings, roads, squares, etc, which are considered as discrete *locations* where active entities at any time may reside. The time spent by an entity at any location is assumed to be sourced from the domain and adheres to some probability distribution. UAVs move from location to location, looking for victims, while (for example) it is known beforehand that victims themselves may move between specific locations due to local conditions. If UAVs are found in the same location at the same time, they are said to *collide*. Collision should of course be avoided, and can be mitigated by minimal communication infrastructure on board the UAVs, which attempts to communicate the position of a UAV to others. If communication is successful, neighboring UAVs avoid entering the corresponding location until the UAV has left it. Successful communication is however governed by probabilistic constraints imposed by antenna features. A goal concerns the composite cyber-physical space system that the UAVs and victims induce, and entails the UAVs *rescuing victims while avoiding collisions*. Typical design questions in such a scenario can be the following.



Development Cycle & Exploratory Design
Requirement Concern
Physical Operation Concern
Interaction Concern
R
Requirements
Space Model
...
Entity Dynamics
I
Entity Interaction Model
Success/Failure States (Entity Class)
Physical Activity (Entity Class)
Interaction Activity (Entity Class)
Statistical Model Checking (System)
...
Composed Behaviour (Entity Instances)
Analyzable Models

Figure 1: Requirements Validation Overview.

(DQ1) Two deployment options for initially positioning UAVs exist, given that victims are stranded in a certain city. Which deployment option should be chosen to better satisfy the system goal?

(DQ2) Two kinds of antennas are available to equip the UAVs with collision-avoidance capabilities, providing low (20%) and high (80%) guarantees of successful communication. Given certain UAV initial positions, what is the impact of adoption of each choice on the system goal?

Challenges inherent in the design questions considered, include evaluation of different deployments, since they can naturally have significant impact on the system goal – deploying UAVs close to positions of victims will enable more probable discovery with perhaps less collisions. Thus, each deployment yields different (statistical) system goal satisfaction. Equipping UAVs with probabilistic communication – abstracted away as two-way position updates – is a typical design concern in cyber-physical systems and related to sensing constraints. A high-performing antenna may also have increased cost, so the effect of its choice should be evaluated.

3 MODELLING SYSTEM CONCERNS

In the following, we describe a systematic approach to model the active entities of a system, how their change in spatial location can be represented along with their interaction mode, and how requirements can be modeled. Finally, we illustrate how such models can be composed, enabling analysis. Formally, our models are stochastic timed automata (STA [21]), timed automata [2] extended with a finite set of real-valued clocks and stochastic features. The stochastic interpretation enables probabilistic choices between enabled transitions and probability distributions of time-bounded delays at states. Additionally, transition guards may control triggering of transitions from state to state during an execution. The stochastic timed automata we consider are input-enabled, deterministic, and non-zeno, as supported by *uppaal-smc* [6]. Effective tool support exists for analyses based on such automata [11, 17] and the formalism is generic enough for modeling concerns of space- and time-dependent systems as discussed in this paper.

3.1 Entity Physical Activity

A *physical activity* models change of discrete position of (a class of) entities. In our example, there are two classes of entities – victim and UAV. Each class is modeled by a STA, and instances of this class represent specific entities. The development of this model follows a systematic sequence of steps:

(1) Space models given in a domain-specific notation, such as BIM [12] or CityGML [15], are automatically abstracted into a formally defined static and discrete topological model [26].
(2) Dynamic topological changes induced by actions performed by active entities are modeled by transformation rules [24].
(3) STAs modeling entity's behavior in space are automatically derived from the previous two models [25].

STA states are labeled. A label records position of an entity in some location through a proposition, while transitions capture possible change from a location to another. For example, if a victim is in a building named 'B4', and adjacent to it is another building named 'B6', the STA has a transition from a state labeled 'B4' to one labeled 'B6' and back. STAs, along with appropriate propositions in states encoding location of entities, are produced automatically from a physical domain model [25].

Regarding initial states of the physical activities, entity classes may differ. Initial states can be chosen probabilistically by allowing a distribution over some defined set of initial states. For our disaster scenario, this can be useful if e.g., exact initial positions of victims are not known, but domain knowledge can estimate a part of the city where victims may be located. For our example, the STA in the lower part of Fig. 2 captures the change of location for a single victim entity. States B4 and B6 represent the position of the victim in those locations through propositions. The initial position may be B4 or B6, with probabilities 0.3 and 0.7 respectively. From there, and depending on delay distributions of states, a change of position may occur according to the transitions of the automaton. Delay distributions of states are omitted from Fig. 2 for simplicity. In constrast, for UAVs, initial positions are given as part of the deployment strategy of DQ1.

3.2 Entity Interaction Activity

In a CPSS, entities do not operate in isolation; they also interact. Through interaction, they may also coordinate behaviors. The interaction concern between entities may differ in different scenarios;

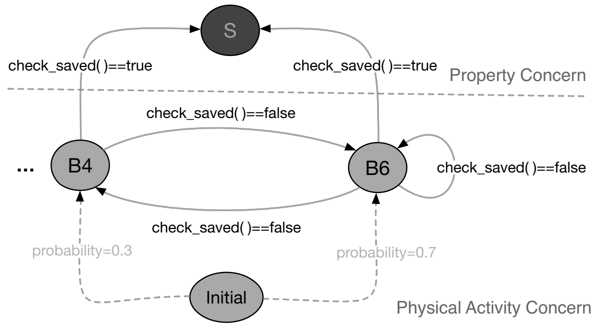


Figure 2: Fragment of the generated Victim model.

faithful to the separation of concerns principle, we describe the interaction model via a separate STA. Communication between entities operating in a cyber-physical space system for instance, is a typical form of interaction, and a model of the communication protocol describes how interaction between entity instances takes place. Typically, the STA describing the interaction model may be specified by a domain expert; e.g., an expert in communication protocols. The STA has entry and exit points (states) that indicate the start and end of the interaction logic. Given transfer of control to the entry state, progression to the exit state has to always eventually occur.

A set of shared variables common to the various entities are in place, and can be used in conjunction with transition guards to encode interaction logic by the system designer; transition guards operate on shared variables, enabling or disabling transitions. In the lower part of Fig. 3, a STA modeling interaction between UAVs is illustrated. Starting from some state (in this case, B4) where interaction is enabled, the STA probabilistically decides to communicate the position of the UAV (in this case, B4) to other UAVs.

3.3 Entity Success-Failure States

In general, system requirements may refer to cross-cutting system concerns, predicate about one or multiple entities in the cyber-physical space system, and may refer to either quantitative or qualitative objectives of a set of entities. For example, a system requirement may state "probability of UAV collision should be at most 20% in the case of less than half of the victims being rescued". Inherent in this formulation are elementary predicates about single entities which are composed in a logical manner that introduces quantitative or logic constraints. For instance, the fact that a UAV collided with another, or that a victim was rescued are predicates that can be either true or false for an entity. Thus, to enable reasoning we encode such success-failure predicates in states; if a STA is found in such a state, the predicate is considered satisfied (or violated) for that entity. This designates *reachability properties* as the fundamental requirement building block, as success-failure states reflect reachability. An entity may be at any point found in a situation that fulfills an elementary predicate; in other words, the STA capturing the entity's behavior may enter a state which encodes such a predicate.

Function prototypes are attached to transition guards of the corresponding success-failure states and their implementation is delegated to the system designer who can utilize domain knowledge to encode success-failure of predicates inherent in the system requirements programatically. This renders our approach semi-automatic. In our example, a success state (representing victims rescued), and a failure state (representing drone collision) are introduced:

- For the victim entity modeled, a “S” state in the upper part of Fig. 2 reflects the success of the *safe* elementary predicate for a single victim entity; if that state is entered by the automaton, the victim is considered safe.
- For the UAV entity modeled, a “C” state reflects the UAV’s status as *collided* (Fig. 3). The transition guard encodes the case where a UAV is found in the same location as another; if that state is entered by the automaton, the UAV is considered as collided.

By delegating encoding of transition guards to the designer, a variety of domain-specific predicates can be modeled. Since in our approach entities operate in a discrete space arising from topological relations in the spatial environment, information abstractions of physical location or position of entities are available for use, as well as current entity configurations. In the following section, the STA transition guard mechanism will be explained further.

3.4 Composing Entity Behavior

While the aforementioned models capture separate concerns, a unified model capturing physical and interaction activities and success-failure states of entities is needed. This occurs in three distinct steps.

(1) Automata corresponding to physical and interaction activities are composed; from every state of the physical activity STA, a transition leads to the state of the interaction one that signifies an interaction entry point; a transition from the interaction exit state hands over control back to the originating state. If interaction is triggered, execution will jump to the interaction automaton and continue in the physical activity STA when interaction logic ends.
(2) Success-failure states are added to the STA. Transitions lead from every state of the physical activity to every predicate state; such states enjoy an *absorbing* markovian property.
(3) Transition guards ensuring its structural validity are added in every transition of the STA. Further functions encoding application-specific logic can be added by the designer.

Crucial to the compositional process is that certain automatically generated transition guards encode high-level entity semantics through structure in the STA. The high-level control flow of the composed STA ensures that predicate state guards are always checked first, interaction is subsequently attempted and finally physical movement occurs, in order to ensure that structural transitioning between activity automata is valid. Transition guards that ensure structural validity are automatically generated, encoding valid behavior of entity automata by ensuring that high-level control flow is maintained. Specifically, that i) the execution of transitions across similar entities is valid and ii) real-time does not pass when transitioning between automata that capture different concerns. In addition, function prototypes corresponding to application-specific logic are attached to relevant transitions as additional guards; their implementation is left to the system designer, who can utilize domain knowledge to encode concerns that span

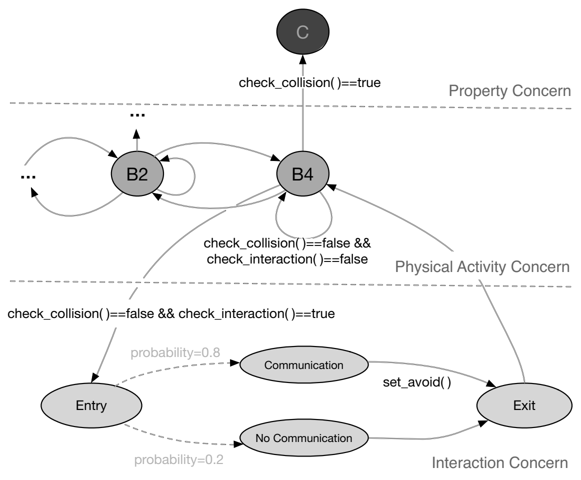


Figure 3: Fragment of the generated UAV model.

different (classes of) entities. STAs may communicate via broadcast channels and shared variables; a STA may perform a transition separately or synchronize with others. This renders broadcast synchronization and shared variables available to transition guards to implement entity interaction logic. The designer may encode as desired more complex operations through additional functions bound to transitions.

For the example presented, a fragment of the UAV model generated is illustrated in Fig. 3. In the middle part, the physical activity captures position of a UAV in locations B4 and B2. From location B4, the STA may enter either the failure state (if *check_collision*() yields true), or the interaction STA (lower part), where the UAV will probabilistically decide to communicate. If that is the case, it will notify other UAVs of the relevant position (by executing a *set_avoid*() function). Control subsequently returns to the originating state B4. Note that since a UAV may decide to communicate at any physical location, composition of the interaction with the physical activity is performed for every of its states, encoding appropriate guards; guard implementation is omitted for clarity in Fig. 3, but can be found along with complete models in accompanying material [1].

4 EARLY REQUIREMENTS VALIDATION

Fundamentally, our approach entails model-based requirements analysis of cyber-physical space systems. The systems we consider are characterized by discrete change of locations in space by the inhabiting system entities which additionally interact among themselves. As is common in cyber-physical systems, the behavior of entities is governed by stochastic features. To this end, the underlying technology that we utilize for analysis is founded on statistical model-based techniques.

4.1 Reasoning on Networks of Entity Instances

The fundamental idea behind statistical model checking [20, 29], is that from sample executions of a stochastic system drawn according to the distribution defined by the system model, quantitative estimates of the probability measure of executions can be calculated. To apply such techniques, what is essentially required is i) a formal model describing a system able to generate sets of executions serving the purpose of observations, ii) a monitoring procedure to decide whether a finite execution satisfies the property under consideration and iii) a statistical algorithm yielding overall results. In essence, we use a formal model of a system to generate execution traces upon which statistical methods produce statistical evidence about the system's satisfaction or violation of a specification. Subsequently, the designer obtains results useful to early requirements validation of the system [5].

Recall that the composite behavior models outlined in the previous section represent classes of entities operating in a space, whose behavior is captured through STAs. We instantiate as many instances of these classes depending on the deployment configuration. The result is a network of STAs. Initial states of entity instances depend on the deployment evaluated by the system designer. In some execution of a network of STAs, each automaton repeatedly races against others. This mechanism entails each STA deciding independently and stochastically how much time to delay before transitioning, where the transition by a STA that chose the minimum delay makes it in the execution trace [11]. Delay distributions on states are normal. Essentially, system components race against each other with respect to the corresponding distributions. Models generated are compatible to *uppaal-smc* [6].

Figure 4 illustrates from a high level perspective how statistical requirements analysis is performed. Sets of executions are obtained from the entity models, upon which a monitoring procedure decides satisfaction or violation of properties [10] sourced from the requirements. Statistical analysis of executions, along with system requirements and the deployment setup configured, yield analysis results for the overall system.

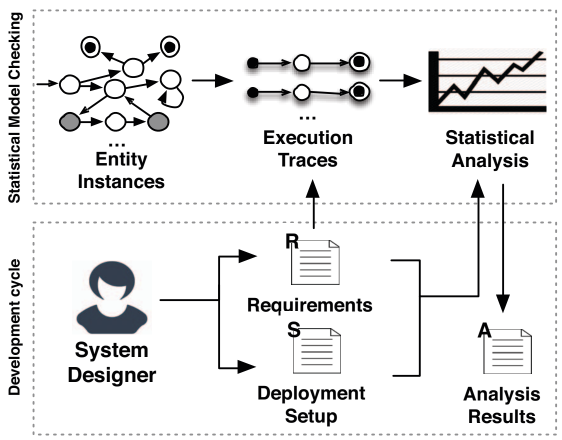


Figure 4: Statistical Requirements Analysis Overview.

4.2 Validation in Practice

To utilize our approach in practice, (i) physical activity models are sourced from domain models yielding physical activity STAs [25] and (ii) integration of interaction STAs, success-failure states, generated transition guards and function prototypes yield composite behavior STAs for single entities automatically. Then (iii) the designer encodes domain specific logic by implementing functions and (iv) a network of STAs is deployed depending on a chosen test configuration. The configuration is (v) loaded in an off-the-shelf statistical model checking tool [6], and results are obtained.

We succinctly indicate results of analyses with respect to the design questions posed in Section 2; complete models and details about the experimental setup can be found in accompanying material [1]. We assume the scenario is within a randomly generated city [4] comprising of 164 distinct locations where 4 UAVs operate and 10 victims are allocated probabilistically in 8 of them. Recall that the system goal entails *rescuing victims while avoiding collisions*.

(DA1) Initial configurations of UAVs can have significant effect on outcomes; optimal and suboptimal initial positions may account for a 31% variance in the average satisfaction probability of the system goal.

(DA2) Given certain fixed initial positions of UAVs and an allocation of victims, selecting the antenna guaranteeing 80% successful UAV communication over one of 20%, leads to a 4.3% increase in the average goal satisfaction probability.

For DQ1, evaluation of different deployments is aided by our approach, as we enable quantitative analysis of specific initial positions that may be selected at will by the designer. Since initial positions can greatly affect the system goal (e.g. deploying UAVs close to each other renders collision more probable), a significant variance in goal satisfaction is observed. By utilizing our approach, the system designer can evaluate different initial positions perhaps taking into account other domain requirements. Regarding DQ2, the selection of the efficient antenna has a moderate effect on goal satisfaction, a result that is not intuitive – increasing the successful communication rate improves non-collision minimally. Such a quantitative result can aid early design-time decisions on the overall system. Analysis times of the scenario presented are in the order of 3.7 to 5.9 minutes.

5 RELATED WORK

Statistical model checking has emerged as a compromise between verification and testing, where statistical techniques produce an estimate for a system's satisfaction of a property [18] while being competitive with state of the art numerical solution methods for probabilistic model checking [29]. The modeling discipline utilizing timed automata with different types of relationships among clocks of a distributed system [21] inspire the modeling of different activities in our approach. Subsequently, statistical methods [29] (e.g. hypothesis testing or Monte Carlo simulation) upon execution traces, produce statistical quantitative evidence about the system's satisfaction or violation of a requirements specification [23]. Similarly to our approach, high-level constructs for communication and data propagation, explicitly taking into account stochastics are developed for validation of cyber-physical multi-agent systems [16]. Statistical model checking has been used to support design of systems-of-systems, such as smart cities and internet-of-things, i.e. for quantifying mission achievement [3] or reasoning in smart grids [30]. Previous work [25] has targeted automatically obtaining automata structures for cyber-physical spaces, which can bootstrap a core aspect of the present work – behaviour STAs from space descriptions. Moreover, physical models may be automatically obtained [24, 26]. Grounding our approach on MDE principles and standards facilitates development of integrated and open tool environments for systematic model-based engineering [14] on top of physical space domain models.

Notable recent approaches have focused on applications of the statistical model checking paradigm in diverse domains within cyber-physical space systems. Statistical model checking and spatio-temporal logics are combined to tackle problems in collective and smart transportation systems, allowing estimation of likelihood of specific spatio-temporal predicates such as clusters in space [9]. Our approach utilizes automata obtained from space models given in a domain-specific notation, abstracted into a formally defined static and discrete topological model, upon which dynamic topological changes are modeled by transformation rules [25, 27]; spatial behavioral models are then automatically derived. Advanced spatial predication could be added to our model support [8], something we identify as future work. In [22], a framework utilizing statistical model checking for dependability within railway systems is developed. Also within software engineering, the ActivFORMS [13] framework exploits statistical model checking at runtime to select configurations that comply with self-adaptation goals over an internet-of-things network topology. In contrast, our approach targets early requirements validation through separation of system design concerns.

6 CONCLUSIONS

Within the context of complex cyber-physical space systems, support for early requirements validation is crucial to the design process. To this end, we outlined a systematic approach to high-level reasoning through separation of key recurrent system concerns and formally defined composition of models that capture them. Our contribution consists of a framework integrating existing techniques for the engineering of cyber-physical space systems through automation of model construction and composition, enabling design-time validation through statistical model checking. The proposed approach has been applied to preliminary studies that confirm that design can be generally decomposed through modeling concerns corresponding to physical and interaction activities and requirements. While we defer a thorough discussion of assumptions and limitations to future work, we plan to integrate reasoning related to variable time constraints of system execution, investigate system requirements expressibility, provide formal support to model construction and guard code generation. Moreover, further support for evaluation of initial entity states as well as their automatic generation can be integrated in the design process. Additionally, we plan to consider cyber-physical domain-related particulars like sensing and actuation.

REFERENCES

[1] 2018. Accompanied specifications and models. 178.62.206.217/sescps18. (2018).

[2] Rajeev Alur and David L Dill. 1994. A theory of timed automata. *Theoretical computer science* 126, 2 (1994), 183–235.

[3] Alexandre Arnold, Massimo Baleani, Alberto Ferrari, Marco Marazza, Valerio Senni, Axel Legay, Jean Quilbeuf, and Christoph Etzien. 2016. An Application of SMC to continuous validation of heterogeneous systems. In *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques.* 76–85.

[4] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. 2016. Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* (2016), 51–59.

[5] Dimitri Bohlender, Harold Bruintjes, Sebastian Junges, Jens Katelaan, Viet Yen Nguyen, and Thomas Noll. 2014. A review of statistical model checking pitfalls on real-time stochastic models. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation.* Springer, 177–192.

[6] Peter Bulychev, Alexandre David, Kim Gulstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. 2012. UPPAAL-SMC: Statistical model checking for priced timed automata. *arXiv:1207.1272* (2012).

[7] Tomás Bures, Danny Weyns, Bradley R. Schmerl, Eduardo Tovar, Eric Boden, Thomas Gabor, Ilias Gerostathopoulos, Pragya Gupta, Eunsuk Kang, Alessia Knauss, Pankesh Patel, Awais Rashid, Ivan Ruchkin, Roykrong Sukkerd, and Christos Tsigkanos. 2017. Software Engineering for Smart Cyber-Physical Systems: Challenges and Promising Solutions. *ACM SIGSOFT Software Engineering Notes* 42, 2 (2017), 19–24.

[8] Vincenzo Ciancia, Stephen Gilmore, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. 2015. Spatio-temporal model checking of vehicular movement in public transport systems. *International Journal on Software Tools for Technology Transfer* (2015), 1–23.

[9] Vincenzo Ciancia, Diego Latella, Mieke Massink, Rytis Paškauskas, and Andrea Vandin. 2016. A tool-chain for statistical spatio-temporal model checking of bike sharing systems. In *International Symposium on Leveraging Applications of Formal Methods.* Springer, 657–673.

[10] Edmund M Clarke, Orna Grumberg, and Doron A Peled. 1999. *Model Checking.* MIT press.

[11] Alexandre David, Kim Larsen, Axel Legay, Marius Mikučionis, Danny Poulsen, Jonas Van Vliet, and Zheng Wang. 2011. Statistical model checking for networks of priced timed automata. *Formal Modeling and Analysis of Timed Systems* (2011).

[12] Chuck Eastman, Charles M Eastman, Paul Teicholz, and Rafael Sacks. 2011. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors.* J.W & S.

[13] Muhammad Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansée, Danny Weyns, and Danny Hughes. 2017. DeltaIoT: A Self-Adaptive Internet of Things Exemplar. In *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017.* 76–82. https://doi.org/10.1109/SEAMS.2017.21

[14] Timo Kehrer, Christos Tsigkanos, and Carlo Ghezzi. 2016. An EMOF-Compliant Abstract Syntax for Bigraphs. In *Graphs as Models at ETAPS16.*

[15] Thomas Kolbe, Gerhard Gröger, and Lutz Plümer. 2005. CityGML: Interoperable access to 3D city models. In *Geo-information for disaster management.* Springer.

[16] Christian Kroiß and Tomáš Bureš. 2016. Logic-based modeling of information transfer in cyber–physical multi-agent systems. *Future Generation Computer Systems* 56 (2016), 124–139.

[17] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS)*, Vol. 6806. Springer, 585–591.

[18] Kim G Larsen and Axel Legay. 2014. Statistical model checking past, present, and future. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation.* Springer, 135–142.

[19] Edward A. Lee. 2008. *Cyber Physical Systems: Design Challenges.* Technical Report. EECS Department, University of California, Berkeley.

[20] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical Model Checking: An Overview. *RV* 10 (2010), 122–135.

[21] Guillermo Rodriguez-Navas and Julián Proenza. 2013. Using timed automata for modeling distributed systems with clocks: Challenges and solutions. *IEEE Transactions on Software Engineering* 39, 6 (2013), 857–868.

[22] Enno Ruijters and Mariëlle Stoelinga. 2016. Better railway engineering through statistical model checking. In *International Symposium on Leveraging Applications of Formal Methods.* Springer, 151–165.

[23] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *CAV*, Vol. 3114. Springer, 202–215.

[24] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. 2016. Architecting dynamic cyber-physical spaces. *Computing* 98, 10 (2016), 1011–1040.

[25] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. 2017. Modeling and verification of evolving cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, September 4-8, Paderborn, Germany, 2017.* 38–48.

[26] Christos Tsigkanos, Timo Kehrer, Carlo Ghezzi, Liliana Pasquale, and Bashar Nuseibeh. 2016. Adding static and dynamic semantics to building information models. In *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems.* ACM, 1–7.

[27] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. 2016. On the Interplay Between Cyber and Physical Spaces for Adaptive Security. *IEEE Transactions on Dependable and Secure Computing* PP, 99 (2016), 1–1.

[28] Junfei Xie, Firas AI-Emrani, Yixin Gu, Yan Wan, and Shengli Fu. 2016. UAV-Carried Long Distance Wi-Fi Communication Infrastructure. In *AIAA Infotech@ Aerospace.*

[29] Hakan L Younes. 2005. *Verification and planning for stochastic processes with asynchronous events.* Technical Report. Carnegie-Mellon University - Pittsburgh PA School of Computer Science.

[30] Ender Yüksel, Huibiao Zhu, Hanne Riis Nielson, Heqing Huang, and Flemming Nielson. 2012. Modelling and analysis of smart grid: A stochastic model checking case study. In *Theoretical Aspects of Software Engineering (TASE), 2012 Sixth International Symposium on.* IEEE, 25–32.