Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

AN APPROACH TO INTEGRATE INTER-DEPENDENT SIMULATIONS USING HLA WITH APPLICATIONS TO SUSTAINABLE URBAN DEVELOPMENT

Ajitesh Jain
David Robinson
Bistra Dilkina
Richard Fujimoto

School of Computational Science & Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

ABSTRACT

Challenges such as understanding sustainable urban development require modeling interdependencies and interactions among systems. The High Level Architecture (HLA) provides an approach to studying these aspects by integrating separately developed simulations in a distributed computing environment. These applications require coupling interdependent simulations and sequencing their execution to ensure certain data dependence requirements are met. An approach to specifying the proper sequence of execution of interdependent simulations using SysML sequence diagrams is proposed. A means to implement these specifications by automatically generating code using HLA's time management services is described. This approach is demonstrated through the creation of a federated simulation to model interactions among land use, transportation, and transit in the San Diego area by integrating widely used simulators such as UrbanSim and MATSim.

1 INTRODUCTION

Many complex systems have multiple interdependent and interacting subcomponents, each of which has its own dynamics and often its own sophisticated simulation models. In order to facilitate understanding at the system level, one needs to be able to integrate separately developed simulation models into a unified model that allows for distributed simulations. Sustainable urban development is one such example of a complex system that requires understanding the interrelationships among multiple infrastructures such as water, energy and transportation.

Sustainability has become a key issue in the study of urban systems as more and more of the Earth's population is predicted to move to urban centers. According to the United Nations World Urbanization Prospects 2014 Report (United Nations 2015) the world's urban population is predicted to surpass 6 billion people by 2050, or 66% of the projected global population, compared to 54% today. Over $53 trillion in infrastructure (re)development is expected through 2030. Because of this, simulation tools that allow decision makers to assess the possible effects of development choices, especially those impacting multiple constituencies, will become more valuable. But as the complexity of urban systems grows, it is difficult to capture the variety of interactions that drive the overall development of a city using any single model.

The High Level Architecture (HLA), IEEE Standard 1516 (IEEE Std 1516-2010 2010, IEEE Std 1516.1-2010 2010, IEEE Std 1516.2-2010 2010) is the most well known approach to model such system of systems. It provides a common methodology and approach to integrate and reuse different simulations. However modifying existing simulations to make them HLA compliant can be a time consuming and error-prone task. In (Jain et al. 2015), we describe a methodology to assist the development of the Federation Objection Model (FOM) using SysML and to automatically generate code for the same. The FOM describes

the common structure and meaning of data shared across the federation. The two key main issues of FOM development which we addressed were: the disparity between measurement units and data types, and identification of semantically similar simulation entities. In order to tackle the first issue, the developer must explicitly document the data types and measurement units of entities while developing the conceptual model. To address the second issue of semantically similar entities, the developer must explicitly identify such entities and relate them in the conceptual model. This is important for the meaningful exchange of data among simulations.

When assembling a federation, it is often the case that federates have specific data dependence requirements among each other. Federate A is data dependent on federate B if A requires specific data from B in order to proceed with its execution. Data dependency is common especially in time-stepped models where one federate produces data that is required by another federate, all within a single time step. For example, a land use simulation may require knowledge of transportation delays before it can predict where new housing developments will occur in the current time step. Because these dependencies arise within a single time step, the HLA time management services, by themselves, do not directly support proper sequencing. The federation developer could determine dependencies and manipulate time management service calls such as *TimeAdvanceRequest* and *NextMessageRequest* with appropriate logical time values to ensure proper sequencing. However, this approach becomes increasingly complex if there are many data dependencies and a large number of federates.

To illustrate this problem, consider a federation consisting of three federates A, B and C. Federate A publishes AttributeA and subscribes to AttributeC. Federate B subscribes to AttributeA and publishes AttributeB, while federate C subscribes to AttributeB and publishes AttributeC. In one time step, federate A executes and then publishes AttributeA to the federation. It then must wait for AttributeC before completing the time step. Federate B waits for an updated value of AttributeA and can only start executing after it has received the update. It then publishes an update on AttributeB. Federate C waits for an update on AttributeB and then starts executing after which it publishes AttributeC to the federation. Thus we see that Federate B is data dependent on Federate A, Federate C is data dependent on Federate B and Federate A is data dependent on Federate C. The problem is to generate a simulation loop for each of the three federates. We describe an approach using SysML Sequence diagrams to depict the execution flow among the federates. This specification is input to a translator that automatically produces code for the simulation loop containing time management service calls with appropriate timestamp values. This helps the developer by simplifying the creation of a correctly synchronized federation. One must note that this approach is useful for integrating models with inter-dependencies and may not be applicable for integrating other types of models.

We exercise our method of automating the integration of federates into an HLA framework through a case study including three separate models: UrbanSim (Waddell 2002), MATSim (Balmer et al. 2008), and TransitSim (a new model created for this study), to form an Integrated Transportation Land Use Model (ITLUM). By combining these models we are able to capture more complex urban interactions, such as people deciding where to live based on traffic congestion, that otherwise would not be taken into account by any one model. We use this ITLUM to study the role of public transportation in urban development in the San Diego area.

2 RELATED WORK

In (Fakhimi et al. 2013), the authors give an overview of different approaches for modeling sustainable development. They recommend using hybrid approaches for modeling different aspects of the Triple Bottom Line (TBL) framework. TBL is a framework that accounts for economic, societal and environmental factors for achieving sustainable development. They argue that using hybrid approaches say Discrete Event Simulation (DES) for modeling queuing systems and System Dynamics (SD) for modeling interplay between different aspects of TBL could be more effective to model sustainable development in contrast to using these approaches in isolation.

Integration of models is typically a laborious, error-prone task. There have been various efforts in the past to address this problem. Uluat and Oğuztüzün (2011) describe a Model Driven Engineering (MDE) approach to integrate legacy simulations via HLA by manually creating mappings from entities of one simulation to another followed by automated code generation. In (Bocciarelli et al. 2012), the authors suggest a model-driven approach for generating HLA code from SysML diagrams. They also propose the use of sequence diagram for generating federate initialization code which generates calls to the RTI for a federate to join the federation and enable time regulation. Adak et al. (2010) suggest using Live Sequence Charts to describe a federate's behavior and then use model transformations to generate HLA code. The Live Sequence Charts contain all the necessary information such as timestamps, lookahead etc to generate HLA code. In contrast, our approach involves describing interactions with other federates using SysML sequence diagrams and the translator generates appropriate timestamp values instead of having the developer derive those values.

Urban growth/land use models and transportation models are particularly useful targets for integration because of the inherent feedback loop between those systems. Clearly decisions concerning where to live are influenced by transportation related factors, such as traffic congestion and the time required to get to work. Likewise, traffic congestion, and the average time taken to travel to work are affected by where individuals live. Newman and Kenworthy (1996) give a brief overview of various studies showing transit oriented development in different cities. It is often the case that infrastructure models require specific domain expertise and are developed in isolation, however they need to be integrated to gain a holistic view.

UrbanSim is a popular, open source land use model that has been developed and used for urban development simulation over the past 15 years (Waddell 2002). The original version of UrbanSim was written in Java in the late 1990's. The second iteration of UrbanSim (also known as OPUS) was implemented in Python and released in 2005. Finally, the latest implementation of UrbanSim, also called the Urban Data Science Toolkit (or UDST, see https://github.com/UDST/urbansim), is implemented in Python and uses more modern data science libraries such as Pandas.

Similarly, MATSim (Balmer et al. 2008) is an open source agent-based transportation simulator that has been developed over the last decade. It is implemented in Java in a modular manner and has been extended to address many research questions (for a list of extensions, see http://matsim.org/extensions). MATSim has been used to simulate traffic in, among other places, Singapore (Erath et al. 2012), Tel-Aviv (Bekhor et al. 2011), and Indonesia (Lämmel et al. 2009).

UrbanSim and MATSim were previously integrated by Thomas Nicolai (Nicolai 2013). However the MATSim extension that was developed involved a manual integration of the two models. MATSim4UrbanSim depends on custom code embedded in the UrbanSim model, and is not compatible with the newest release of UrbanSim. This highlights a common problem that custom integration efforts are fragile, and not easily adapted as new versions of the models become available. Coupling the models with standardized interfaces such as those provided by HLA combined with automated code generation simplifies code maintenance and allows for more models to be more easily integrated.

3 SYSTEM DESCRIPTION

3.1 Creating The Federation Object Model (FOM)

As described in (Jain et al. 2015), to generate a FOM the developer needs to complete the following steps:

1. Create a SysML description of the simulation models. SysML provides a platform independent method for specifying simulation models. This involves creating a Block Definition Diagram (BDD) containing simulation entities with their measurement units. The entities are annotated with HLA stereotypes such as HLAClass, HLAAttribute and Federate. The federates also specify the attributes to which they publish and subscribe.
2. Define relationship among similar entities. This is achieved by using an association name "similarEntities" between those entities. Similar entities that require some transformation, an association

block is used which specify transformation routines to convert between entities and indicate if the transformation is lossy or not.
3. After the above two steps, the diagram is exported as an XMI file and input to a translator. The translator routine develops an internal model represented as a directed weighted graph. It generates a FOM such that the loss of information for transforming from one entity type to another is minimal. The translator generates a fedfile and code for transformation routines.
4. The translator generates RTI routines such as the initialization code for publishing and subscribing to different attributes, their corresponding *UpdateAttributeValues* and *ReflectAttributeValues* methods.
5. In the last step, the developer manually inserts calls to the above generated routines at appropriate places in the simulation code. This includes invoking time management services to synchronize the federation.

The SysML description can be reused and adapted as the simulation is used in other federations for different purposes.

3.2 Generating The Simulation Loop

SysML provides Sequence Diagrams as a means to describe a system's dynamic behavior. They depict interactions between different entities in a system. In object oriented programming, sequence diagrams indicate the order of method call invocations by one object on another. We use these diagrams to show the relative order of how different federates exchange data, then input it to a translator that generates simulation loop code for each federate using time management service calls with appropriate timestamps values. The entities participating in the flow of execution are represented by a dashed vertical line called the lifeline. Time increases as one goes down the lifeline. The horizontal arrows represent the actual communication between entities. We use the Federated Simulations Development Kit (FDK) (Fujimoto et al. 2000), that implements a subset of the HLA Interface Specification for our work. This implementation of the RTI assumes that all federates begin at the same time, i.e. they all join the federation together at time $t = 0$, and that all attributes are delivered by the RTI in timestamp order. In order to parse these diagrams, we use the following vocabulary to describe the sequence flow. We indicate data flow from federate A to federate B with horizontal arrow annotated with label *Update(AttributeName)*. Here, *AttributeName* indicates the attribute being sent from A to B. It is assumed that federate A would have being described to publish and federate B to subscribe to that attribute or any attribute connected to it via "similarEntities" relation in the BDD. Federates also can send an *Execute()* message to themselves representing some internal processing by the federate. Also, the lifeline of each federate represents all interactions in one cycle of the simulation loop. Figure 1 shows an example of a sequence diagram depicting the interactions among federates for the federation described in the introduction section.

Once we have described the interactions among federates we export the diagram as XML Metadata Interchange (XMI) format and parse it using JAVA DOM parser and XPATH APIs. Each *Update(AttributeName)* gets translated to an *UpdateAttributeValues* call to the RTI. Let us assume that Δt is the length of each cycle, i.e., if the simulation loop begins at time t then the simulation time at the end of one iteration is $t + \Delta t$. We label all the *Update(AttributeName)* calls from 1 to N from top to bottom in the sequence diagram using a topological sort. This means that the first *Update(AttributeName)* by any federate is labeled 1 while the last such call by any federate is labeled N. Let $\varepsilon = \Delta t / N$. Then the timestamp of n^{th} update is given by $t + n\varepsilon$. We generate the simulation loop for a federate as follows:

1. Scan the lifeline of the federate to get all incoming and outing interactions along with *Execute()* self invocation messages from top to bottom (increasing time)
2. Initialize $time = t$. This keeps track of the simulation time of the federate
3. For each message m with label l:

(a) If m is an incoming *Update* message, check if $time < (t + l\varepsilon)$ is true. If this condition is true, output *NextEventRequest* with timestamp $(t + l\varepsilon)$ followed by a while loop invoking RTI's *tick()* method till the time is advanced. Set $time = (t + l\varepsilon)$

(b) If m is an outgoing *Update* message,
   i. Check if $time == (t + (l-1)\varepsilon)$ is true. If this condition is false, output *TimeAdvanceRequest* with timestamp $(t + (l-1)\varepsilon)$ followed by a while loop invoking RTI's *tick()* method till the time is advanced. Set $time = (t + (l-1)\varepsilon)$
   ii. Output *UpdateAttributeValues* function invocation for that attribute with timestamp $(t + l\varepsilon)$.

(c) If m is *Execute* message, find the immediate incoming or outgoing *Update* message p with label i after this message for this federate.
   i. If there are no messages, output *TimeAdvanceRequest* with timestamp $(t + \Delta t)$. Also output a call to *Execute()* followed by a while loop invoking RTI's *tick()* method till the time is advanced. Set $time = (t + \Delta t)$
   ii. If there is a message p with label i, find the difference between $time$ and $(t + (i-1)\varepsilon)$. If the difference is greater than 0, output *TimeAdvanceRequest* with timestamp $(t + (i-1)\varepsilon)$. Output a call to *Execute()*. If the difference is greater than 0, output while loop invoking RTI's *tick()* method till the time is advanced and set $time = (t + (i-1)\varepsilon)$

4. Check if $time < (t + \Delta t)$. If this condition is true, output *TimeAdvanceRequest* with timestamp $(t + \Delta t)$ followed by a while loop invoking RTI's *tick()* method till the time is advanced. Set $time = (t + \Delta t)$.

The code generated for federate A using the sequence diagram in Figure 1 is shown in Figure 2. It should be noted that since we are calling *UpdateAttributeValues* for timestamp $(t + l\varepsilon)$ that, when the federate is at $(t + (l-1)\varepsilon)$, a look-ahead value should be chosen so that it is less than ε. We need to call *TimeAdvanceRequest* before every *Execute()* call so that the execution of the federate doesn't block the execution for other federates. *Execute()* is a stub which is generated so as to produce an executable federation. This helps to first test the federation before modifying the existing simulations and uncover any susceptible bugs. The developer explicitly needs to remove calls to *Execute()* by invoking appropriate functions or embedding the above generated time APIs at appropriate places in the simulation code. The current implementation of the translator generates code for FDK in C. It can also generate the corresponding code in the programming language used to develop the simulation, for instance Python.

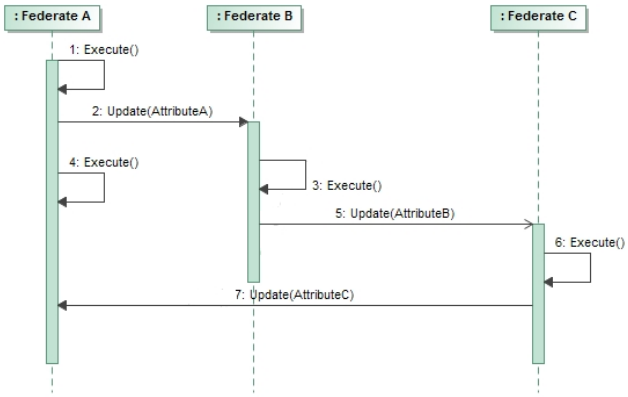


Figure 1: Example Sequence Diagram.

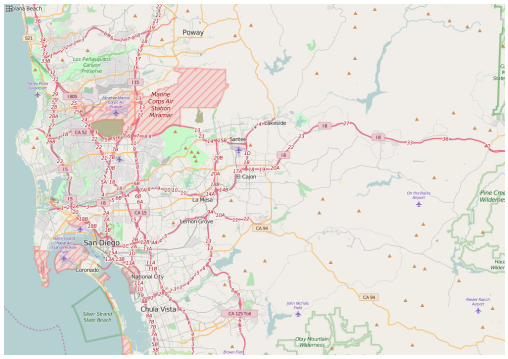
```
deltaT = 6.0; time = 0.0; epsilon = deltaT / 3; // Local Variables
waitingForTAG = FALSE; // Global variable modified in TimeAdvanceGrant callback
while (True) {
    Execute();
    UpdateAttributeValues(AttributeAObjectDesignator, attrA, time + epsilon);

    TimeAdvanceRequest(time + 2 * epsilon);
    waitingForTAG = TRUE;
    Execute();
    while(waitingForTAG == TRUE) // Wait for TimeAdvanceGrant
        RTI_Tick();

    NextEventRequest(time + 3 * epsilon);
    waitingForTAG = True;
    while(waitingForTAG == TRUE) // Wait for TimeAdvanceGrant
        RTI_Tick();
    time += deltaT;
}
```

Figure 2: Simulation loop for Federate A using sequence diagram shown in Figure 1.

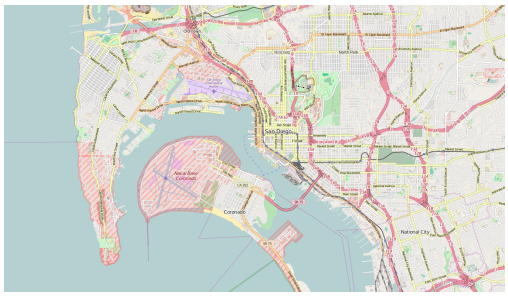
4 CASE STUDY

To illustrate the techniques described above, and to show how they can help facilitate sustainable development, we have performed a case study using the city of San Diego, California. Specifically, we have create a metamodel using the techniques from Section 3 in order to simulate urban development while taking into account feedback from the traffic and public transportation systems. The metamodel is made of three separate models, namely: UrbanSim (Waddell 2002), MATSim (Balmer et al. 2008), and a simple public transportation model we have developed for this case study, which we call TransitSim. These models are joined together as individual federates in a federation using the methodology described in the Section 3. The UrbanSim model is a parcel based model that completes urban development simulations using a wide variety of data sources. MATSim is an agent-based transportation model that runs traffic simulations on the road network to estimate the travel time between pairs of zones in the SUI. Lastly, TransitSim is a model that describes what portion of the population will use public transportation. When joined together, these models make up an Integrated Transportation and Land Use Model (ITLUM) for the San Diego area that allows us to study any number of urban development questions. The model covers a wide area around the city of San Diego, shown in Figure 3a, however for our case study we focus on results in the downtown San Diego area shown in Figure 3b. This downtown portion of the city will constitute our system under investigation (SUI).

According to the United States Census Bureau's 2007-2011 American Community Survey (United States Census Bureau 2011) only 4.1% of San Diego residents use public transportation to get to work, while 84.7% drive to work. Our case study examines how increasing the probability that workers will take public transportation to and from their jobs affects the congestion and development of the city over a 5 year time period. To do this, we run two different scenarios using our federation: a baseline scenario, and scenario in which an exaggerated percentage of the population uses public transportation. In the baseline scenario the probability that a person, whose home and work are within a mile of a bus stop, takes public transportation, is set to 4.1% to estimate the current conditions in San Diego. In the exaggerated experiment, this probability is set to 50% to see what effects a concentrated public transportation campaign might have on the development of the city as a whole. We simulate 5 years of development, starting from 2013, where each iteration of the entire federation represents a year, and examine the spatial distribution of the population over time, for both scenarios, to reason about the effects of each scenario.



(a) Wider simulation area surrounding San Diego.



(b) The area of downtown San Diego that we are using as our system under investigation.

Figure 3: System under investigation.

4.1 UrbanSim Model

In this study we use a heavily modified version of the example San Diego model and data provided with the new UrbanSim implementation, found at https://github.com/UDST/sandiego_urbansim. UrbanSim uses information such as land use, and census data, in order to simulate land use, including where people choose to live, in future years. As is, the UrbanSim model uses information about travel times between pairs of zones, but these values remain fixed over a simulation. In our integrated simulation, the travel times are updated by the output from the MATSim simulation after each simulation timestep, thereby affecting the development of the simulation in future timesteps.

Part of the UrbanSim data contains information about the population in San Diego at the household level (i.e. number of people who live in a house, number of workers, etc.). We have extended the model with a module that samples a percentage of the population from the households table to use in the subsequent TransitSim and MATSim simulation. The sampling process involves picking a random household, then for each worker in that household picking a random job location from the job table for that person. This temporary table contains the household and job locations of each sampled person and is written to file after each iteration of the simulation. The rest of the tables are written to file after each iteration using the "data_out" functionality of the orca.run method in the UrbanSim simulation pipeline.

4.2 TransitSim Model

We have developed a simple public transportation model called TransitSim to be used with UrbanSim and MATSim to provide a richer model, and another target for our HLA coupling. Given a list of people with home and work locations, a set of bus stops, and an underlying road network, this model decides which people will take public transportation to and from their job. This model is parameterized by: a.) how far a person is willing to travel to get to a bus stop (by walking or otherwise), called the *range*; and b.) the probability, p, that a person will take public transportation provided that it is within *range* of their home and work. Experiments on our SUI show that approximately 73% of the workers, from the baseline year data, are within a mile range of a bus stop. The details of the model are as follows:

1. Given a list of input individuals, for every input individual:
   (a) Find the distance from their home location to the nearest road, d_h, and distance from their job to the nearest road, d_j
   (b) Find the distance from those points to the nearest bus stops, d_{bh} and d_{bj} respectively
   (c) If $d_h + d_{bh} \leq range$ and $d_j + d_{bj} \leq range$

   i. Flip a coin with probability p, which if passes, means that the person will take public transportation, remove them from the output list
   ii. Else they drive to work, a process that is simulated by MATSim, they stay in the output list
2. Output the subset people on the input list that drive to work

We make some assumptions with this model. We assume that if someone has access to a single bus stop, they are able to travel to any other bus stop in the city. The transportation capacity of the bus system is not taken into account. The time needed to travel using a bus is not thresholded. Other modes of public transportation are not taken into account, and we assume the bus system does have a significant impact on traffic congestion in the city.

4.3 MATSim Model

MATSim (Balmer et al. 2008) is an open source agent based transportation simulator. The model takes several inputs: a table of persons with their home and job locations (the temporary table described in Section 4.1), and a road network that has the free speed, congested speed, number of lanes, and length information for each road segment. The output from the model is an origin-destination matrix that has the simulated travel times between each pair of zones in the SUI. This output is fed back into the UrbanSim model and used in the next iteration to model household and job location choices.

We run the MATSim model using the MATSim4Urbansim extension, developed by Thomas Nicolai (Nicolai 2013). To do this we have a Python script wrapper that generates a XML configuration file according to the MATSim4Urbansim XSD, found here http://matsim.org/files/dtd/matsim4urbansim_v3.xsd, then run MATSim via a command line call. In all of our case study experiments these parameters stay constant.

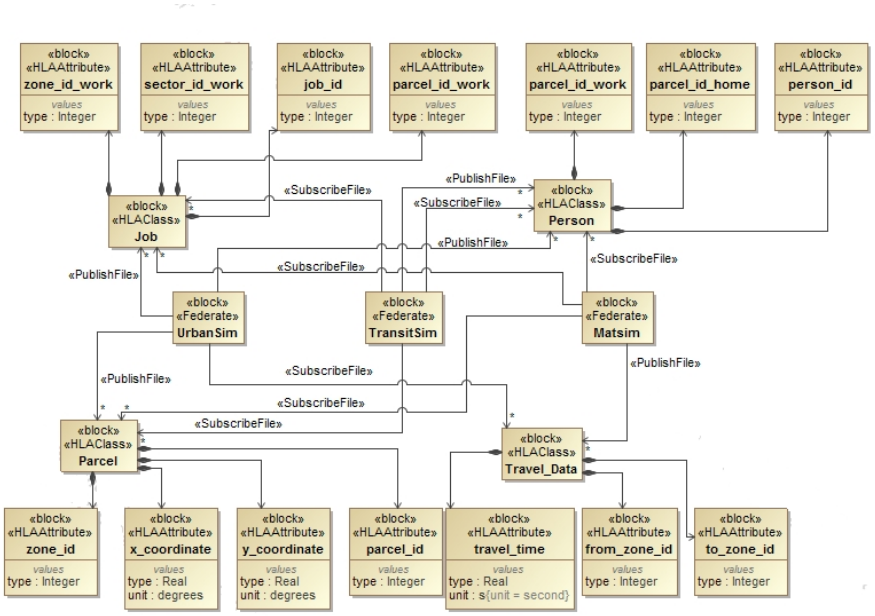
4.4 Integration Methodology

We integrate the three models first, by describing the BDD in a SysML editor. The entities involved in the federations consist of Parcel, Person, Travel data and Job. The entity classes and their attributes are shown in Figure 4a.

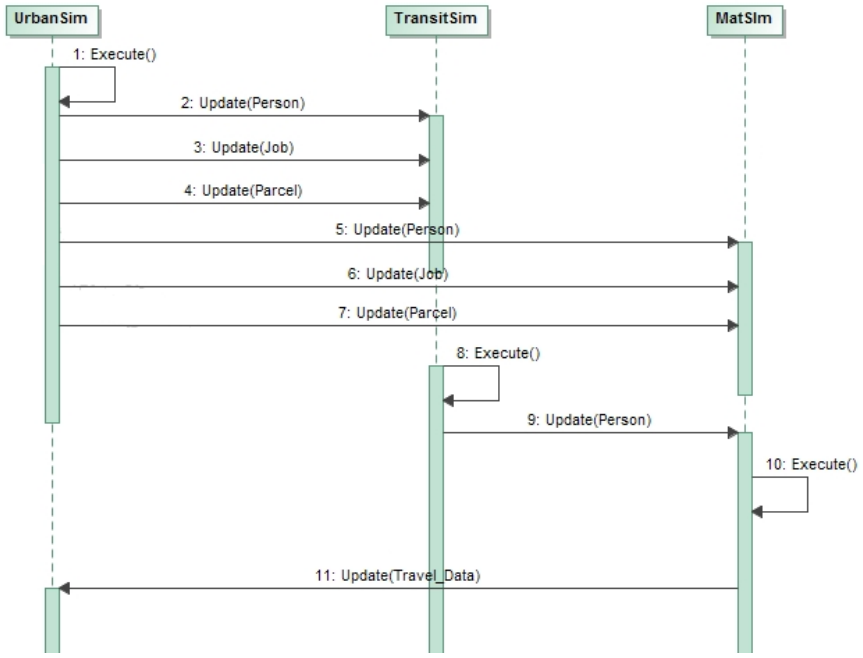
In order to generate the time management method calls, we used the sequence diagram to depict the flow of messages among the federates and input it to the translator. We then manually embedded the generated calls at appropriate places in the code. Figure 4b shows the sequence diagram for the federation. UrbanSim publishes person, parcel and job data and it subscribes to travel data. TransitSim publishes person data and subscribes to person, job and parcel data. MATSim publishes travel data and subscribes to person, job and parcel data. The execution begins with UrbanSim which generates the person, job and parcel data. The other two simulations wait for UrbanSim to publish this data. After UrbanSim publishes data, TransitSim executes and filters people based on the model described above. It publishes person data after execution. Once MATSim receives updates about person data published by TransitSim, it starts its execution and publishes travel data to the federates. We have introduced new stereotypes "publishFile" and "subscribeFile" in SysML to represent this where if an HLAClass is connected to an HLAFederate by an association annotated with any of these stereotypes, the UpdateAttribute and ReflectAttribute functions send and receive the file path where those files are written by the simulations.

4.5 Results and Discussion

Two experiments using the integrated model were developed, a baseline experiment and exaggerated public transportation experiment. In the baseline experiment we set the probability that a person, whose home and work locations area within 1 mile of a bus stop, takes public transportation (instead of driving to work) to 4.1%, while in the exaggerated experiment 50%. To see the effect that this change has on the development of San Diego we run our simulation for 5 years, from 2013 to 2018.



(a) BDD for the Urban Modeling federation.



(b) Sequence diagram of interaction among federates.

Figure 4: SysML Diagrams.

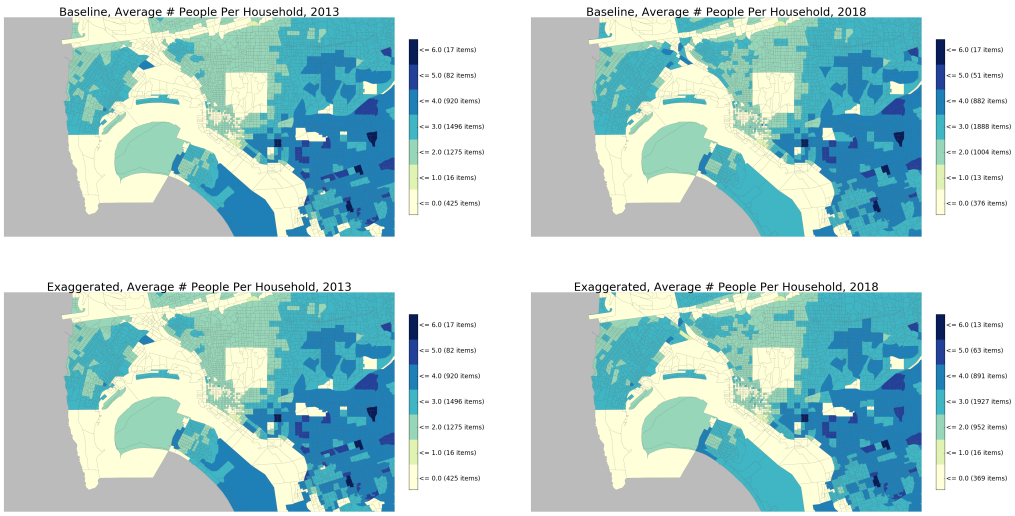


Figure 5: Average number of people per household per TAZ in downtown San Diego for both scenarios at the beginning and end of the simulation.

Our results consist of a set of maps that show the average number of people per household per TAZ, and the total population of each TAZ at the start of the simulation compared to that at the end. These results can be seen in Figures 5 and 6. The maps that show the total population per TAZ show that a smaller number of people live in the downtown area at the end of the exaggerated experiment than in the baseline experiment. A possible reason for this is that there will necessarily be less congestion downtown in the exaggerated experiment due to the larger number of people taking public transportation. Less congestion downtown means shorter commutes, and that people will have less incentive to live closer to their jobs. This effect will be realized in the MATSim traffic simulation where fewer agents will be used in the traffic simulation due to more people "deciding" to take public transportation in the TransitSim model. Another result that can be seen from the maps is that the average population per household seems to increase in the north west corner of the SUI in both scenarios, however the total population levels for the same areas do not change a substantial amount. Consistent behavior over different scenarios can be further explored by looking at more spatial variables from the UrbanSim output, but will be left to future work.

These initial results verify our automated modeling integration method and motivate future work in the field of computational sustainability (Gomes 2009). Using HLA to facilitate the communication between many models that previously would run independently will allow much richer analyses of urban systems, and consequently the effects of making certain developmental decisions.

5 CONCLUSION

In this paper, we describe a complete workflow for integrating simulations using HLA. This helps speed up the process of creating federations by automating code generation to address sequencing problems that arise when creating an executable federation of inter-dependent simulations. The developer only needs to spend time creating the SysML descriptions which can be reused in other federations and manually embedding the generated HLA code at appropriate places in the simulation code. We demonstrate the effectiveness of this methodology by integrating simulations for urban development.

Future work on this project involves expanding the case study to include models of the water and energy infrastructure systems in cities. Then, once a joint simulation environment for urban development is created, we can move on with optimizing sustainable urban growth under different metrics. We envision

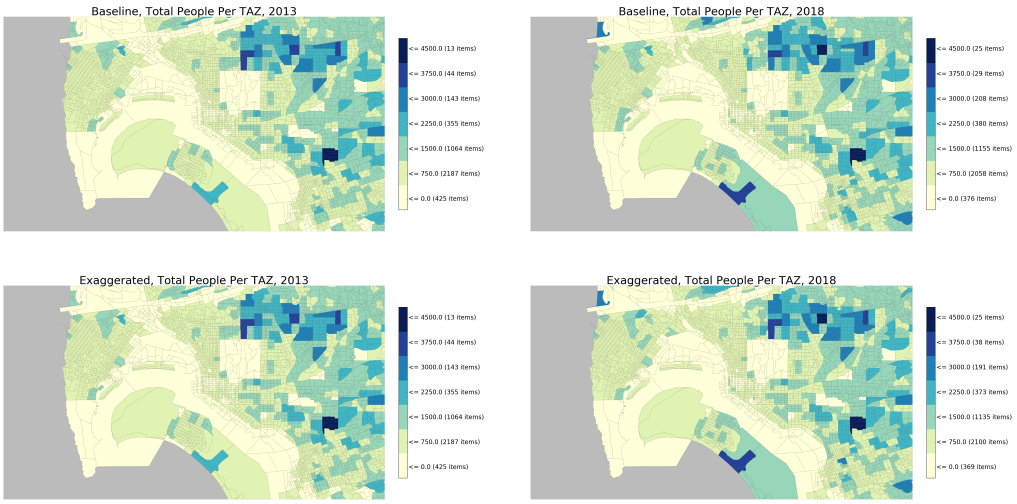


Figure 6: Total population per TAZ in downtown San Diego for both scenarios at the beginning and end of the simulation.

that these activities will help us to create an ontology for sustainable urban development in SysML, which can facilitate easier integration of further models in the future.

ACKNOWLEDGMENTS

This research was provided by National Science Foundation Grant 1441208 and CCF-1522054 (COMPUSTNET: Expanding Horizons of Computational Sustainability).

REFERENCES

Adak, M., O. Topçu, and H. Oguztüzün. 2010. "Model-based code generation for HLA federates". *Software: Practice and Experience* 40 (2): 149–175.

Balmer, M., K. Meister, M. Rieser, K. Nagel, K. W. Axhausen, K. W. Axhausen, and K. W. Axhausen. 2008. *Agent-based simulation of travel demand: Structure and computational performance of MATSim-T*. ETH, Eidgenössische Technische Hochschule Zürich, IVT Institut für Verkehrsplanung und Transportsysteme.

Bekhor, S., C. Dobler, and K. Axhausen. 2011. "Integration of activity-based and agent-based models: case of Tel Aviv, Israel". *Transportation Research Record: Journal of the Transportation Research Board* (2255): 38–47.

Bocciarelli, P., A. D'Ambrogio, and G. Fabiani. 2012. "A model-driven approach to build HLA-based distributed simulations from SysML models". In *SIMULTECH*, 49–60.

Delligatti, L. 2013. *SysML distilled: A brief guide to the systems modeling language*. Addison-Wesley.

Erath, A., P. Fourie, M. Van Eggermond, S. Ordóñez, A. Chakirov, and K. Axhausen. 2012. "Large-scale agent-based transport demand model for Singapore". In *13th International Conference on Travel Behaviour Research (IATBR). Toronto: International Association for Travel Behaviour Research*.

Fakhimi, M., N. Mustafee, L. Stergioulas, and T. Eldabi. 2013. "A review of literature in modeling approaches for sustainable development". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. Kuhl, 282–290. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fujimoto, R., T. McLean, K. Perumalla, and I. Tacic. 2000. "Design of high performance RTI software". In *Distributed Simulation and Real-Time Applications, 2000.(DS-RT 2000). Proceedings. Fourth IEEE International Workshop on*, 89–96. IEEE.

Gomes, C. P. 2009. "Computational sustainability: Computational methods for a sustainable environment, economy, and society". *The Bridge* 39 (4): 5–13.

IEEE 2010a. "IEEE 1516 - Standard for modeling and simulation high level architecture - framework and rules".

IEEE 2010b. "IEEE 1516.1 - Standard for modeling and simulation high level architecture - federate interface specification".

IEEE 2010c. "IEEE 1516.2 - Standard for modeling and simulation high level architecture - object model template (OMT) specification".

Jain, A., R. Fujimoto, J. Crittenden, M. Liu, J. Kim, and Z. Lu. 2015. "Towards automating the development of federated distributed simulations for modeling sustainable urban infrastructures". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 2668–2679. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kuhl, F., R. Weatherly, and J. Dahmann. 1999. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR.

Lämmel, G., H. Klüpfel, and K. Nagel. 2009. "The MATSim network flow model for traffic simulation adapted to large-scale emergency egress and an application to the evacuation of the Indonesian city of Padang in case of a tsunami warning". *Pedestrian behavior*:245–265.

Newman, P. W., and J. R. Kenworthy. 1996. "The land use-transport connection: An overview". *Land use policy* 13 (1): 1–22.

Nicolai, T. W. 2013. "MATSim for UrbanSim: Integrating an urban simulation model with a travel model".

Uluat, M. F., and H. Oğuztüzün. 2011. "Model based approach to the federation object model independence problem". In *Computer and Information Sciences II*, 451–459. Springer.

United Nations 2015. "World urbanization prospects: The 2014 revision".

United States Census Bureau 2011. "Means of transportation to work by selected characteristics: 2007-2011 American Community Survey 5-Year Estimates". http://factfinder2.census.gov.

Waddell, P. 2002. "UrbanSim: Modeling urban development for land use, transportation, and environmental planning". *Journal of the American Planning Association* 68 (3): 297–314.

AUTHOR BIOGRAPHIES

AJITESH JAIN has an MSCS degree from Georgia Institute of Technology. His research interests include modeling and simulation and artificial intelligence. His email address is ajiteshJain@gatech.edu.

DAVID ROBINSON is a Ph.D. student at Georgia Institute of Technology in the School of Computational Science and Engineering. His research interests include using modeling and simulation, and machine learning to solve problems in the field of Computational Sustainability. His email address is dcrobins@gatech.edu.

BISTRA DILKINA is an Assistant Professor in the School of Computational Science and Engineering at the Georgia Institute of Technology. She received a Ph.D. in Computer Science from Cornell University in January 2012. Her email address is bdilkina@cc.gatech.edu.

RICHARD FUJIMOTO is Regents' Professor in the School of Computational Science and Engineering at the Georgia Institute of Technology. He received a Ph.D. in Computer Science and Electrical Engineering from the University of California-Berkeley. His email address is fujimoto@cc.gatech.edu.