Dynamische Co-Simulation von Automatisierungssystemen und ihren Komponenten im Internet der Dinge

Prozessorientierte Interaktion von IoT-Komponenten

Tobias Jung, Nasser Jazdi, Michael Weyrich
IAS - Institut für Automatisierungstechnik und Softwaresysteme
Universität Stuttgart
Pfaffenwaldring 47
70550 Stuttgart
Tel.: 0711 685 67301
Fax: 0711 685 67302
E-Mail: ias@ias.uni-stuttgart.de

Abstract: IoT-Systeme bringen neue Herausforderungen, wie den dynamischen Eintritt von heterogenen Komponenten zur Laufzeit mit sich, welche auch bei der Simulation von IoT-Systemen beachtet werden müssen. Ein möglicher Ansatz diesen Herausforderungen zu begegnen ist der Einsatz einer agentenbasierten Co-Simulation, welche jede IoT-Komponente eines IoT-Systems individuell in einer eigenen Simulation ausführt und über die Interaktion zwischen den zugrundeliegenden Modellen diese zu einer Gesamtsimulation zusammenfügt. Diese Interaktion unterteilt sich in Nachrichtenaustausch und prozessorientierte Interaktion zwischen den Komponenten. Der Fokus dieses Beitrags liegt auf der Simulation der prozessorientierten Interaktion zwischen Komponenten in einer Co-Simulation.

Stichworte: Co-Simulation, Agenten, Internet der Dinge, IoT-Komponente, IoT-System

1 Einleitung

Durch den Einzug des Internet der Dinge (IoT), u. a. in Produktion und Logistik, werden automatisierte Systeme immer dynamischer und heterogener. IoT-Komponenten können zur Laufzeit in ein IoT-System ein- und austreten, zudem findet eine Vernetzung über die bisherigen Domänengrenzen statt, wodurch die einzelnen Komponenten eines IoT-Systems immer unterschiedlicher werden. Um diesen Herausforderungen, welche in allen Phasen des Lebenszyklus auftreten, zu begegnen, werden neuartige Konzepte, wie beispielsweise „Plug-and-Produce“ während des Betriebs, vorgestellt. Auch in der Simulation solcher IoT-Systeme werden neue Lösungen benötigt, die dieser Dynamik und Heterogenität begegnen. Ein möglicher Ansatz hierzu ist „Plug-and-Simulate“.

In [JuJaWe2017] werden bestehende Ansätze und Konzepte zur Simulation von IoT-Systemen untersucht und auf ihre Tauglichkeit, insbesondere in Hinblick auf ihre „Plug-and-Simulate“-Fähigkeiten, untersucht und geschlussfolgert, dass ein neuartiges Co-Simulationskonzept benötigt wird, um den genannten Herausforderungen zu begegnen. In diesem Konzept werden einzelne IoT-Komponenten in individuellen Simulationstools modelliert und simuliert. Diese einzelnen Simulationen werden durch Softwareagenten repräsentiert und durch das so entstehende Agentensystem miteinander verknüpft. Durch die so entstandene Co-Simulation ist es möglich über das Agentensystem Nachrichten zwischen den einzelnen Teilsimulationen, welche die einzelnen Komponenten simulieren, zu versenden. Diese, zwischen den Agenten versendeten, Nachrichten simulieren die Kommunikation zwischen den Komponenten des realen IoT-Systems. Da die unterschiedlichen Simulationstools unterschiedliche Schnittstellen besitzen, werden zwischen die Agenten und die dazugehörigen Simulationen Übersetzer geschaltet, welche die von den Simulationstools versanden, Nachrichten in das von den Agenten genutzte Austauschformat übersetzen. Dieses Konzept ist in Abb. 1 dargestellt.

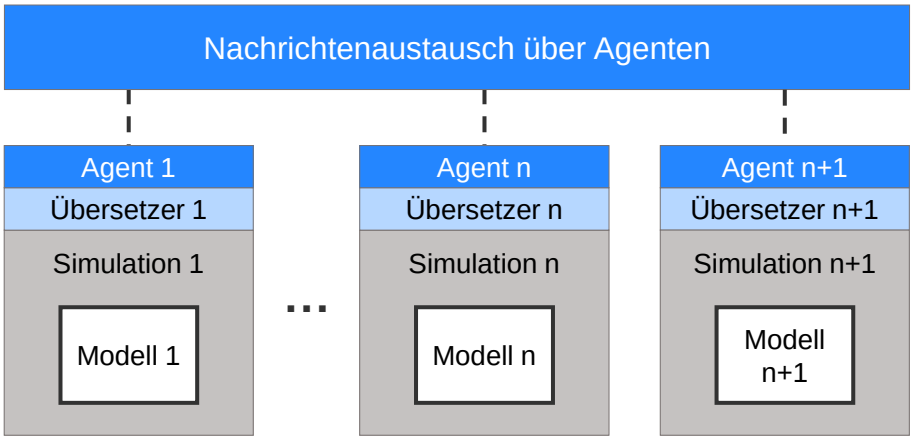

Abbildung 1: Dynamische, agentenbasierte Co-Simulation

Bei der Simulation von IoT-Systemen muss allerdings neben der Kommunikation zwischen den einzelnen Komponenten auch die prozessorientierte Interaktion zwischen den Komponenten betrachtet werden. Der Unterschied zwischen einer Interaktion per Nachrichtenaustausch und einer prozessorientierten Interaktion soll an folgendem Beispiel verdeutlicht werden: Fordert ein Werkstück durch eine Nachricht eine Bearbeitungsstation auf, es zu bearbeiten, findet eine Interaktion zwischen Werkstück und Bearbeitungsstation per Nachrichtenaustausch statt, welche den inneren Zustand beider beeinflusst. Die anschließende Bearbeitung des Werkstücks, beispielsweise Bohren, hingegen ist eine prozessorientierte Interaktion zwischen dem Bohrer und dem zu bearbeitenden Werkstück und beeinflusst ebenfalls den inneren Zustand von einer oder mehreren beteiligten Komponenten.

Da diese prozessorientiert Interaktion zwischen den IoT-Komponenten noch nicht in dem beschriebenen Konzept berücksichtigt wird, muss es um ein Konzept zur Simulation der

prozessorientierten Interaktion erweitert werden. Hierzu werden zuerst in Kapitel 2 bestehende Konzepte zum Datenaustausch in einer Co-Simulation hinsichtlich ihrer „Plug-and-Simulate“-Fähigkeiten untersucht und bewertet und anschließend werden in Kapitel 3 Eigenschaften des Datenaustauschs bei der Simulation von IoT-Systemen beschrieben sowie ein neues Datenaustauschkonzept für eine „Plug-and-SImulate“-fähige Co-Simulation vorgestellt. In Kapitel 4 wird eine prototypische Umsetzung des Konzepts präsentiert, anhand deren das Konzept evaluiert wird.

2 Standards und Ansätze zur Co-Simulation

Es existiert eine Vielzahl an Co-Simulationsstandards und –ansätzen, sowohl domänenspezifische, als auch domänenübergreifende. Domänenspezifische Standards und Ansätze wie CAPE-OPEN (Prozessindustrie) [Col2011] oder EPOCHS (Smart Grids) [HoWaGi2006] werden in diesem Beitrag nicht weiter berücksichtigt, da sie für eine Co-Simulation von IoT-Systemen nicht geeignet sind. Der Grund hierfür ist, dass IoT-Systeme durch die Vernetzung Komponenten aus verschiedenen Domänen enthalten können. Ein Beispiel dafür ist, wenn sich die Produktion mit der Logistik vernetzt.

Die existierenden, domänenübergreifende Standards wurden hauptsächlich hinsichtlich ihres Datenaustauschs untersucht. Hierbei wurde untersucht, ob der gewählte Ansatz zum Datenaustausch „Plug-and-Simulate“ ermöglicht und falls ja, ob dieser in das oben beschriebene Konzept integriert werden kann. Die untersuchten Co-Simulationskonzepte unterteilen die Interaktion zwischen den Simulationen nicht in kommunikations- und prozessorientiert, da mit den untersuchten Konzepten nicht nur IoT-Systeme simuliert werden. Daher wurde auch untersucht, ob die kommunikationsorientierte Interaktion ebenfalls durch die untersuchten Konzepte abgedeckt werden kann.

Functional Mock-Up Interface (FMI)

Laut [OpWoUr2014] bietet Functional Mock-Up Interface keine Möglichkeit zur Realisierung einer „Plug-and-Simulate“-fähigen Simulation, trotzdem wurde der gewählte Ansatz zum Datenaustausch hinsichtlich seiner „Plug-and-Simulate“-Eignung untersucht. Bei FMI werden die einzelnen Simulatoren in Functional Mock-Up Units (FMUs) gekapselt und in ein Simulationstool integriert, welches als Simulationsmaster dient. Die in den FMUs gekapselten Simulatoren nehmen als Simulationsslaves an der Co-Simulation teil. Hierbei wird der Datenaustausch durch den Simulationsmaster zentral koordiniert, wodurch kein direkter Datenaustausch zwischen zwei Simulationsslaves möglich ist. Der Datenaustausch selbst findet immer zu diskreten Zeitpunkten statt, indem der Simulationsmaster c-Funktionen der Simulationsslaves aufruft und ihnen so über Parameter die benötigten Daten liefert. Nach Beendigung des Simulationsschritts, übergeben die Simulationsslaves durch Rückgabewerte wieder Daten an den Simulationsmaster, welcher diese

vor dem nächsten Simulationsschritt wieder an die entsprechenden Simulationsslaves verteilt [Mod2013].

Prinzipiell bietet das Konzept des Datenaustauschs von FMI die Möglichkeit von „Plug-and-Simulate“, allerdings wird sie durch die zentrale Koordination des Datenaustausches durch den Simulationsmaster „Plug-and-Simulate“ erschwert, da das gesamte Wissen über die Co-Simulation an einem zentralen Ort aggregiert werden muss. Dies stellt eine Einschränkung bei der Modellierung und der Wahl der Simulationstools dar, da beim Eintritt einer neuen Simulation, diese dem Simulationsmaster mitteilen muss, welche Daten sie bereitstellen kann und welche sie benötigt. Um die Notwendigkeit der Wissensaggregation zu vermeiden ist eine dezentrale Koordination des Datenaustauschs wünschenswert.

High Level Architecture (HLA)

High Level Architecture bietet laut [OpWoUr2014] prinzipiell die Möglichkeit Modelle zur Laufzeit in die Co-Simulation zu integrieren. Der Datenaustausch wird auch bei HLA zentral koordiniert. Jede Co-Simulation bei HLA besitzt eine Runtime-Time-Infrastructure (RTI), welche viele Funktionen des Simulationsmaster in FMI erfüllt. Die RTI ist selbst allerdings keine Simulation sondern hauptsächlich nur für den Datenaustausch und die Synchronisierung der Co-Simulation zuständig. Der Datenaustausch selbst findet über den Austausch von Objekten statt und folgt dem Publish-Subscribe-Schema. Hierbei meldet jede Teilsimulation bei der RTI welche Daten sie bereitstellen kann und welche sie benötigt. Allerdings haben die Teilsimulationen keine Informationen über die anderen Teilsimulationen. Damit die RTI dennoch die Daten an die richtige Teilsimulation weiterleiten kann, existiert in jeder Co-Simulation mit HLA mindestens ein Federation Object Model, in welchem definiert ist, welche Daten in der Co-Simulation ausgetauscht werden können. Zusätzlich besitzt jede Teilsimulation ein Simulation Object Model, in dem spezifiziert ist, welche Daten sie bereitstellen kann und welche sie benötigt [Top2017].

Modelle, die schon vor Simulationsstart bekannt sind und berücksichtigt wurden, können zur Laufzeit der Simulation geladen werden. Zur Entwicklungszeit unbekannte Modelle können allerdings nicht zur Laufzeit geladen werden, ohne dass das Federation Object Model angepasst wird. Daher ermöglicht dieser Ansatz zum Datenaustausch in einer Co-Simulation nur sehr beschränkt „Plug-and-Simulate“.

Zusätzlich zu FMI und HLA wurden noch Co-Simulationsansätze basierend auf OSGi [OpDrLu2013] und OPC UA [HeGrUr2016] untersucht, welche beide prinzipiell „Plug-and-Simulate“-fähig sind. Zu diesen Ansätzen gab es allerdings keine Informationen bezüglich des Datenaustauschs.

Da keines der existierenden Konzepte zum Datenaustausch uneingeschränkt „Plug-and-Simulate“ ermöglicht, wird für das oben vorgestellte Co-Simulationskonzept ein neues Datenaustauschkonzept entwickelt.

3 Datenaustausch in einer agentenbasierte Co-Simulation von IoT-Systemen

Keines der untersuchten Konzepte ist uneingeschränkt „Plug-and-Simulate“-fähig, daher wird die Trennung in kommunikations- und prozessorientierte Interaktion beibehalten. Der Grund hierfür ist, dass, um einen Datenaustausch in einer Co-Simulation zu ermöglichen, eine Spezifikation dieses Datenaustauschs existieren muss, durch die Form und Inhalt der zwischen den Agenten versendeten Nachrichten definiert sind. Zur Simulation der kommunikationsorientierten Interaktion können die Spezifikationen der im IoT-System verwendeten Kommunikationstechnologien wie W-LAN oder Bluetooth verwendet werden. Durch die Verwendung dieser Spezifikationen des realen Systems kann die Kommunikation im IoT-System sehr realitätsnah simuliert werden. Für die prozessorientierte Interaktion existiert allerdings keine Spezifikation im realen System und zudem muss eine Abstraktion der prozessorientierten Interaktion gemacht werden, da komplexe, kontinuierliche Prozesse nicht in allen Einzelheiten simulierbar sind. In einem realen IoT-System, kann ein Temperatursensor selbst die Temperatur der Umgebung messen, in der Co-Simulation hingegen muss die Information der Temperatur von der Umgebungssimulation der Temperatursensorsimulation mitgeteilt werden. Genau für diese Mitteilung gibt es keine Spezifikation. Daher muss für die prozessorientierte Interaktion eine Spezifikation erstellt werden, welche die Realität abstrahiert. Durch die Trennung in kommunikationsorientierte und prozessorientierte Interaktionen müssen nur die prozessorientierten Interaktionen abstrahiert werden wohingegen die kommunikationsorientierten Interaktionen sehr realitätsnah simuliert werden können.

Prozessorientierte Interaktionen lassen sich in zwei Kategorien einteilen, eine direkte Interaktion zwischen zwei Komponenten und eine Interaktion einer Komponente mit der Umwelt. Ein Beispiel für die direkte Interaktion wäre eine Bearbeitung einer Komponente durch eine andere Komponente. Eine Interaktion mit der Umwelt hingegen ist beispielsweise die Erhitzung der Umgebung durch eine Komponente oder das Messen der Umgebungstemperatur.

Durch den Umstand, dass eine Interaktion mit der Umgebung stattfinden kann, reicht es nicht aus, die einzelnen Komponenten des IoT-Systems zu simulieren, sondern die Umgebung muss ebenfalls modelliert und simuliert werden. Somit wird das Agentensystem durch Simulationen der Umwelt erweitert. Hierbei kann jeder Aspekt der Umwelt durch ein eigenes Modell repräsentiert werden, beispielsweise können Umgebungstemperatur und Luftfeuchtigkeit jeweils in einer separaten Simulation simuliert werden.

Allerdings kann sich bei der direkten Interaktion das Problem ergeben, dass in dem Agentensystem und in den Modellen nicht zwingend bekannt ist, welche Komponenten direkt miteinander interagieren. Beispielsweise wenn ein Werkstück von einer Maschine bearbeitet wird, wird über das Agentensystem die Information über die Bearbeitung an die Werkstücke weitergeleitet, wenn in der Modellierung der Werkstücke allerdings keine Positionsbestimmung vorgesehen ist, so ist

es nicht möglich, das Modell des richtigen Werkstücks zu identifizieren. Daher muss eine Möglichkeit geschaffen werden, mit der die Position jeden Modells identifiziert werden kann. Dies kann ebenfalls über eine Modellierung der Umgebung geschehen, indem der Raum, in dem sich die IoT-Komponenten befinden modelliert wird und in diesem Umgebungsmodell die Positionsinformationen der IoT-Komponenten gespeichert werden.

Somit wird sowohl für die direkte als auch die indirekte prozessorientierte Interaktion eine Umgebungssimulation benötigt. In dieser Umgebungssimulation müssen allerdings alle Modelle, die über dieses Umgebungsmodell interagieren, und die diese Modelle repräsentierenden Agenten eindeutig identifizierbar sein, um zu gewährleisten, dass immer die richtigen Modelle miteinander interagieren.

Zusätzlich müssen die prozessorientierten Interaktionen abstrahiert und diskretisiert werden, da es nicht möglich ist, die prozessorientierten Interaktionen in allen Einzelheiten zu simulieren und ein Datenaustausch über die Agenten nicht kontinuierlich sein kann, da die Kommunikation zwischen den Agenten nachrichtenbasiert und somit diskret ist. Der Abstraktionsgrad und die Diskretisierung müssen allerdings einheitlich über alle Teilsimulationen, die miteinander interagieren, sein, um einen Datenaustausch zu ermöglichen. Daher muss eine Spezifikation der prozessorientierte Interaktion erstellt werden, in der der Abstraktionsgrad und die Diskretisierung definiert sind.

Da die prozessorientierte Interaktion zwischen zwei oder mehr Komponenten durch die Umgebungssimulation gekapselt ist, kann diese Spezifikation über die Umgebungssimulation realisiert werden. Hierzu gibt die Umgebungssimulation vor welche prozessorientierten Interaktionen über sie möglich sind und bietet diese Interaktionen in Form von Diensten an. Diese Dienste definieren die Art der Interaktion sowie die für diese Interaktion benötigten Informationen in Form von zu übergebenden Parametern. Wird ein Werkstück beispielsweise gebohrt, muss die Simulation der bohrenden Maschine den Dienst „Bohren“ der Umgebungssimulation aufrufen und die Parameter „Bohrdurchmesser“, „Bohrgeschwindigkeit“ und „Position des Bohrers“ übergeben. Anschließend fordert die Umgebungssimulation die Simulation des Werkstücks auf, den Dienst „Gebohrt werden“ aufzurufen und den Parameter „Materialeigenschaften“ zu übergeben.

Die IoT-Komponentensimulationen selbst bieten keine Dienste an um eine möglichst hohe Wiederverwendbarkeit der Modelle der IoT-Komponenten zu erreichen und „Plug-and-Simulate“ zu erleichtern. Jede IoT-Komponente muss nur einmal modelliert werden und anschließend kann sie ihr Modell mit sich tragen. Somit ist es möglich die Modelle der IoT-Komponenten für Simulationen von verschiedenen IoT-Systemen zu verwenden. Da aber die prozessorientierte Interaktionen in verschiedenen IoT-Systemen meistens auch unterschiedlich modelliert werden, sind Anpassungen an der Spezifikation der prozessorientierten Interaktionen und somit an den von der Umgebungssimulation angebotenen Diensten unerlässlich. Um die Modelle der IoT-

Komponenten ohne Anpassungen wiederverwenden zu können, müssen alle Dienste von der Umgebungssimulation angeboten werden. Somit muss bei unterschiedlichen Simulationen nur die Umgebungssimulation angepasst werden. Da bei vielen prozessorientierten Interaktionen die Simulationen der passiven IoT-Komponenten, wie das Werkstück im oben genannten Beispiel, aber nicht wissen, wann sie einen Dienst aufrufen müssen, muss ihnen die Umgebungssimulation mitteilen, wann ein Dienst aufgerufen werden muss. Diese Aufforderung geschieht über eine Nachricht, die zwischen den Agenten der Umgebungssimulation und der IoT-Komponente ausgetauscht wird.

Für den Nachrichtenaustausch reicht ein einfacher Übersetzer nicht mehr aus, sondern es wird eine zweite Schnittstelle zwischen Simulationstool und Agent benötigt, welche den Austausch der Nachrichten zur prozessorientierten Interaktion ermöglicht. Somit erweitert sich das bisherige Konzept zum einen um Umgebungsmodelle und zum anderen um eine weitere Schnittstelle zwischen Simulationstools und Agenten. Die Umgebungssimulation selbst benötigt keinen Übersetzer sondern nur eine Schnittstelle zur prozessorientierten Interaktion, da über sie keine kommunikationsorientierten Nachrichten ausgetauscht werden können. Dieses erweiterte Konzept ist in Abb. 2 dargestellt.

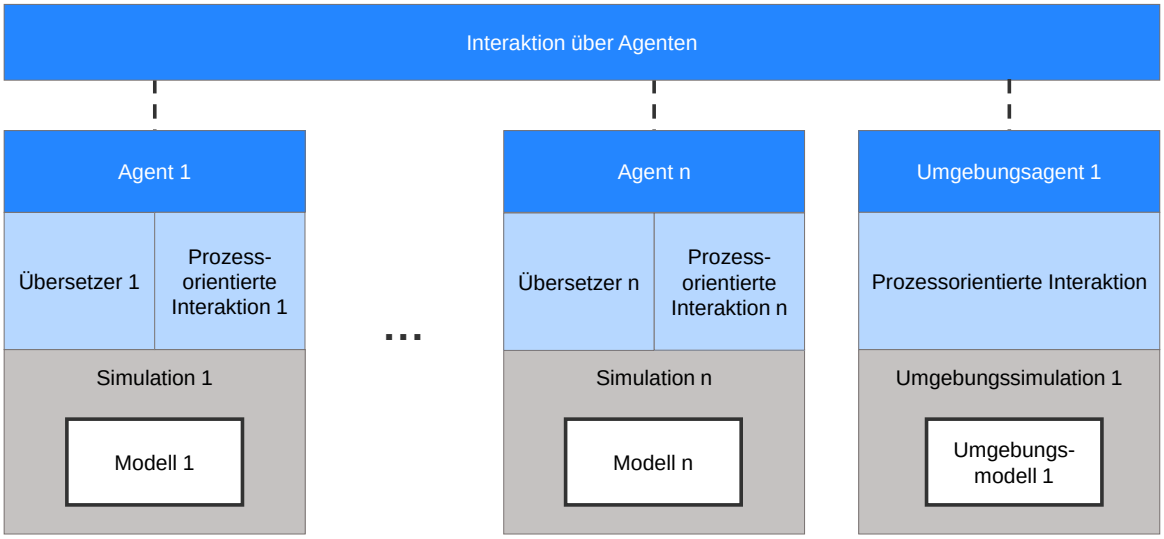

Abbildung 2: Erweiterte, dynamische, agentenbasierte Co-Simulation

Das Konzept zur Simulation der prozessorientierten Interaktion in IoT-Systemen besteht somit aus der einer Umgebungssimulation, welche über angebotene Dienste die prozessorientierte Interaktion diskretisiert und den Abstraktionsgrad der Simulation definiert. Zusätzlich wurde der Übersetzer zwischen den Agenten und den Simulationen um eine Schnittstelle für prozessorientierte Interaktionsnachrichten erweitert.

4 Simulation einer Klimasteuerung und Evaluierung des Konzepts

Um das oben beschriebene Konzept zu evaluieren, wurde ein System zur Klimasteuerung simuliert. Diese Simulation besteht aus einem Temperatursensor, einer Heizeinheit, einem Luftfeuchtigkeitssensor, einem Luftfeuchtigkeitsgenerator, einer Steuerung und einem Umgebungsmodell. Die Steuerung frägt regelmäßig die Temperatur- und Luftfeuchtigkeitswerte bei dem Temperatursensor und dem Luftfeuchtigkeitssensor ab und startet die Heizung bzw. den Luftfeuchtigkeitsgenerator, falls der entsprechende Wert unter einen gewissen Schwellwert sinkt. Die Umgebung ist so modelliert, dass sowohl die Temperatur, als auch die Luftfeuchtigkeit kontinuierlich sinkt und erst wieder steigt, wenn entweder die Heizung oder der Luftfeuchtigkeitsgenerator angeschaltet sind. Sobald diese angeschaltet sind, steigen die Temperatur bzw. die Luftfeuchtigkeit konstant an.

Die Modelle selbst wurden in Java erstellt. Somit wurden zwar nicht verschiedene Simulationstools verwendet, allerdings ist die gesamte Simulation trotzdem eine Co-Simulation, in die, durch die Repräsentation der Simulationen durch Agenten, die einzelnen Modelle in die Co-Simulation ein- und austreten können. Somit können die, für die prozessorientierte Interaktion relevanten Aspekte dennoch evaluiert werden. Das Agentensystem wurde mit Jadex implementiert, da in Jadex eine Serviceorientierung vorgesehen ist, mit der die Dienste des Umgebungsmodells realisiert werden können. In Abb. 3 ist das Ausgabefenster der Klimasimulation dargestellt, über welches auch die einzelnen Modelle zur Laufzeit der Simulation in die Simulation integriert bzw. aus der Simulation genommen werden können.

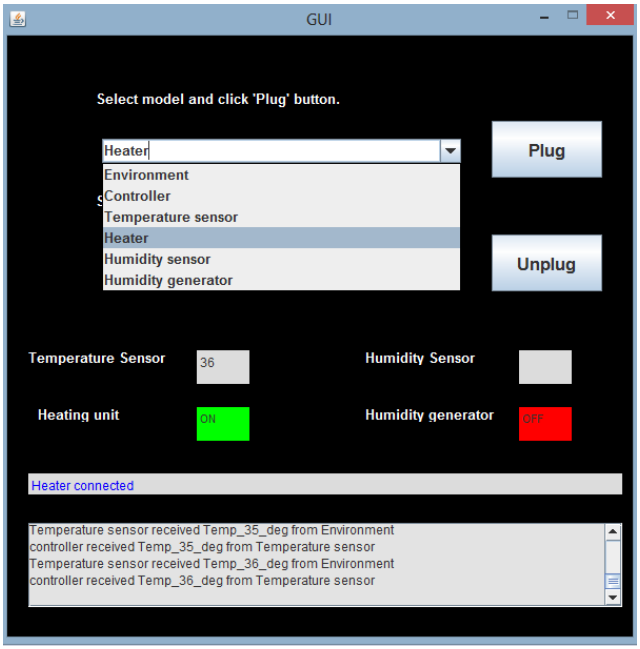

Abbildung 3: Simulation einer Klimasteuerung

In diesem Szenario bietet die Umgebungssimulation folgende Dienste an, welche von den anderen Modellen verwendet werden können:

- Heizung an / aus
- Luftfeuchtigkeitsgenerator an / aus
- Temperatur messen
- Luftfeuchtigkeit messen

Die kommunikationsorientierte Nachrichten werden direkt zwischen den Modellen ausgetauscht. Zwischen den Simulationen und den Agenten sind sowohl die Übersetzer als auch die Schnittstellen für die prozessorientierten Interaktion implementiert.

Durch die Umsetzung dieses Szenarios konnte nachgewiesen werden, dass es möglich ist mit dem beschriebenen Konzept ein IoT-System zu simulieren, in das zur Laufzeit neue Modelle eintreten können und welches sowohl die kommunikationsorientiert als auch die prozessorientierten Interaktion berücksichtigt.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein existierendes Konzept zur „Plug-and-Simulate“-fähigen Simulation von IoT-Systemen um die Simulation der prozessorientierten Interaktion zwischen den IoT-Komponenten erweitert. Hierzu wurden folgende Punkte gezeigt:

- Zuerst wurde der Bedarf der Simulation der prozessorientierten Interaktion in IoT-Systemen aufgezeigt.
- Es wurden bestehende CO-Simulationsansätze hinsichtlich ihrer Datenaustauschkonzepte untersucht und bewertet, insbesondere in Hinblick auf die „Plug-and-Simulate“-Möglichkeiten dieser Datenaustauschkonzepte.
- Anschließend wurde ein neuartiges Konzept zum Datenaustausch in einer agentenbasierten Co-Simulation vorgestellt, welches den Datenaustausch in kommunikationsorientierte und prozessorientierte Nachrichten trennt.
- Abschließend wurde eine prototypische Umsetzung dieses Konzepts anhand einer Klimasteuerung vorgestellt, mit der das beschriebene Konzept evaluiert wurde.

Im nächsten Schritt wird der Ansatz weiter modifiziert, mit dem die Dienste, die von der Umgebungssimulation angeboten werden, automatisiert erkannt und genutzt werden können. Momentan müssen diese Dienste noch manuell in die Modelle bzw. deren Agenten integriert werden. Diese manuelle Integration wiederspricht allerdings dem „Plug-and-Simulate“-Gedanken. Das automatisiert Erkennen und Nutzen der Dienst wird relevant, wenn die Modelle der IoT-Komponenten mit verschiedenen Umgebungssimulationen verschiedener Co-Simulationen interagieren, in denen ähnliche oder gleiche Dienst unterschiedlich benannt wurden. Eine

Möglichkeit die automatisiert Erkennung und Nutzung zu ermöglichen, ist eine Standardisierung der Dienste.

6 Literatur

[Col2011] CO-LaN consortium: CAPE-OPEN Thermodynamic and Physical Properties, Specification 2011, http://www.colan.org/wp-content/uploads/2015/05/CO_Thermo_1.1_Specification_311.pdf, 2011

[HeGrUr2016] Hensel, S., Graube, M., Urbas, L., Heinzerling, T., and Oppelt, M.: Jansen, G.: Co-simulation with OPC UA. 14th International Conference on Industrial Informatics (INDIN), 20–25, Poitiers, France, 2016

[HoWaGi2006] Hopkinson, K., Wang, X., Giovanini, R., Thorp, J., Birman, K., and Coury, D. 2006. EPOCHS. A Platform for Agent-Based Electric Power and Communication Simulation Built From Commercial Off-the-Shelf Components. IEEE Trans. Power Syst. 21, 2, 548–558.

[JuJaWe2017] Jung, T.; Jazdi, N.; Weyrich, M.: A Survey on Dynamic Simulation of Automation Systems and Components in the Internet of Things. 22nd IEEE International Conference on Emerging Technologies And Factory Automation, Limassol, Cyprus, 2017

[Mod2013] Modelica Association Project “FMI,” “Functional Mock-up Interface for Model Exchange and Co-Simulation,” no. 07006, pp. 1–120, 2013

[OpDrLu2013] Oppelt, M., Drumm, O., Lutz, B., Gerrit W.: Approach for integrated simulation based on plant engineering data. Proceedings of 2013 IEEE 18th International Conference on Emerging Techologies & Factory Automation, 1-4, Cagliari, Italy, 2013

[OpWoUr2014] M. Oppelt, G. Wolf, and L. Urbas, “Capability-analysis of co-simulation approaches for process industries,” in ETFA'2014: 19th IEEE International Conference on Emerging Techologies and Factory Automation : September 16-19, 2014 : Barcelona, Spain, Barcelona, Spain, 2014, pp. 1–4.

[Top2017] O. Topçu, Ed., Guide to distributed simulation with HLA. Cham, Switzerland: Springer, 2017