# Framework für Agentensysteme zur Parallelisierung von simulationsbasierten Entwicklungsaufgaben

Desirée Vögeli, Peter Göhner, Michael Weyrich
IAS - Institut für Automatisierungstechnik und Softwaresysteme
Universität Stuttgart
Pfaffenwaldring 47
70156 Stuttgart
Tel.: 0711 68567320
Fax: 0711 68567302
E-Mail: desiree.voegeli@ias.uni-stuttgart.de

**Abstract:** Um Ingenieure bei Simulationsaufgaben zu unterstützen, wird ein Framework vorgestellt, das Simulationen basierend auf strukturiertem Wissen parallel und dezentral berechnet. Hierbei können verschiedene Varianten automatisiert simuliert und miteinander verglichen werden. Zur flexiblen Koordination der dezentralen Simulationen werden Softwareagenten eingesetzt. Evaluiert wird das vorgestellte Konzept anhand der Anwendungsgebiete des Regelungsentwurfs und der multiphysikalischen Simulation.

**Stichworte:** Engineering, Softwareagenten, parallele Simulation, Framework

## 1 Motivation und Problemstellung

Die Automatisierung im Bereich des Designs und Entwurfs neuer Anlagen und Produkte ist im Vergleich zur automatisierten Fertigung noch wenig fortgeschritten. Durch die Forderung nach immer kürzeren Entwicklungszeiten und mehr Einsparpotential werden jedoch auch hierbei neue Konzepte erforderlich, um die Ingenieure bei ihren Tätigkeiten zu unterstützen. Hierzu gehört auch immer häufiger das Koordinieren und Durchführen von Simulationen. Von der Digitalisierung der Entwicklung hin zu Digitalen Zwillingen werden Simulationen in allen Phasen des Engineerings verwendet [ASW17]. In frühen Phasen des Engineerings dienen sie zur Untersuchung des Systemverhaltens [CMR+17]. Häufig ist dabei keine direkte Berechnung möglich, sondern das Verhalten muss iterativ simuliert werden. Um die erforderliche Rechenleistung für die immer aufwändiger werdenden Simulationen in kurzer Zeit zu erbringen, ohne auf teure Superrechner zurückzugreifen, werden häufig Ansätze zur Parallelisierung von Berechnungen verfolgt, um mit vorhandenen Rechenressourcen effektiv aufwändige Simulationen durchzuführen. Dabei wird meist ein System für genau eine Problemstellung bzw. ein Anwendungsgebiet erstellt, welches die verteilte Simulation dann automatisiert durchführt [VNG+17, PPS+13, ByGö16]. Neben diesen Lösungen für spezielle Anwendungsfälle, existieren zur Parallelisierung von Simulationen bereits Standards, wie z. B. [IEEE 1278, IEEE 1516]. Zur Realisierung ist jedoch viel Erfahrungswissen notwendig, sowohl im Anwendungsgebiet als auch bei den Standards. Zwar wird durch den Einsatz dieser Standards die Simulationsdurchführung automatisiert, nicht jedoch die Aufteilung in die einzelnen Simulationen. Auch eine Untersuchung alternativer Lösungsmöglichkeiten wird

hier nicht unterstützt. Eine andere Möglichkeit zur automatisierten dezentralen Simulation bietet das Functional Mock-Up Interface (FMI). Dieses ermöglicht den Austausch von Simulationsergebnissen zwischen unterschiedlichen Simulationswerkzeugen. Jedoch sind nicht für alle Simulationstools sowohl Export- als auch Import-Funktionen gegeben und für manche (z. B. COMSOL Multiphysics) gibt es keine Schnittstelle [FMI18]. Auch ist der Austausch der Daten hierbei nur zwischen Master und Slave möglich. Insgesamt mangelt es an einer flexibleren Assistenzlösung, die zur Integration eines neuen Anwendungsgebiets kein Spezialwissen über parallele Programmierung benötigt. Durch die Unterstützung bei Simulationen im Engineering soll die Qualität der Ergebnisse verbessert bzw. die Ergebnisse schneller geliefert werden, ohne dass mehr Personal benötigt wird. Zudem sollen phasenweise ungenutzte PCs für die Simulation verwendet werden können, um vorhandene Rechenressourcen effizient zu nutzen.

In diesem Paper wird nun ein flexiblerer Ansatz zur automatisierten, parallelen Simulation im Engineering untersucht. Hierbei werden keine gekoppelten Einzelsimulationen wie bei [JJW18] betrachtet, sondern die effiziente Verhaltenssimulation für die Entwicklung eines Systems steht im Fokus. Dabei werden automatisiert alternative Wege und Lösungen untersucht, um auf diese Weise der eingeschränkten, subjektiven Sicht eines Ingenieurs entgegenzuwirken und gleichzeitig die Variantenvielfalt beherrschbar zu machen. Das Paper beschreibt hierzu zunächst das verwendete Konzept. Danach werden die Realisierung und zwei Anwendungsbeispiele beschrieben, bevor abschließend eine Zusammenfassung und ein Ausblick folgen.

# 2 Konzept eines Frameworks zur Parallelisierung von Simulationsprozessen

Zur Unterstützung von Ingenieuren muss das entwickelte Assistenzsystem in der Lage sein, Simulationsaufgaben zu zerlegen und die Teilsimulationen parallel zu verarbeiten. Zwischen diesen Teilsimulationen muss ein kooperativer Lösungsaustausch erfolgen, um Abhängigkeiten – z. B. Randbedingungen – aufzulösen und Entscheidungen über den besten Lösungsweg zu treffen. Das Assistenzsystem soll zudem verschiedene vorhandene Softwaretools und Rechenressourcen zur Simulation einbinden können und keine neue Simulationsumgebung darstellen. Um eine Einsetzbarkeit in vielen unterschiedlichen Anwendungsgebieten beim Engineering zu gewährleisten, muss eine leichte Übertragbarkeit des Wissens vorhanden sein. Daher wird in diesem Kapitel zunächst auf die Aufgaben bei Simulationen eingegangen, bevor die Wissensintegration, die Toolanbindung und die Systemarchitektur besprochen werden.

## 2.1 Ablauf und Aufgaben während einer Simulation

Für dezentrale Simulationen sind die gleichen Phasen erforderlich wie beim dezentralen Problemlösen allgemein (siehe Abbildung 1). Dieser Ablauf ist unabhängig vom Anwendungsgebiet der Simulation.


*Abbildung 1: Phasen des dezentralen Problemlösens nach [Durf01]*

Bei der Problemzerlegung müssen hierbei Teilsimulationen identifiziert und mögliche Lösungsansätze, die entsprechend simuliert werden sollen, gefunden werden. Diese Ansätze müssen auf die zur Verfügung stehenden Ressourcen verteilt werden, damit sie dort anschließend bearbeitet werden können. Während dieser Bearbeitungsphase ist es notwendig, dass Informationen ausgetauscht werden können. Auch eine Integration der verschiedenen Lösungen bzw. eine Auswahl des besten Ansatzes muss erfolgen. Zudem ist es für ein Assistenzsystem wichtig, dass mit dem Benutzer interagiert wird, um die Problemstellung zu erfragen und die Ergebnisse zurückzugeben. Für viele dieser Aufgaben muss kein anwendungsgebietsspezifisches Wissen vorhanden sein. Und auch bei spezifischen Aufgaben findet man in den unterschiedlichen Anwendungen Ähnlichkeiten in der Struktur. Dadurch ist es möglich, ein Framework zur parallelen Simulation (ParSimIAS) bereitzustellen, das diese allgemein benötigten Funktionen bereitstellt und durch spezifische ergänzt werden kann. Zu diesen allgemeinen Aufgaben gehört unter anderem die Projektabfrage, welcher aus der Entgegennahme der zu simulierenden Modelle sowie Anforderungen besteht, der Information des Benutzers über den aktuellen Zustand des Systems und den in Abbildung 1 gezeigten grundsätzlichen Phasen. Auch die Protokolle zur Kooperation sind gleich. So können Ergebnisse oder Teilergebnisse ausgetauscht werden, es kann nach einer spezifischen Information gefragt werden und die Reaktion auf eingegangene Informationen kann durch Ignorieren, direkt Verwenden oder im nächsten Schritt Verwenden beschrieben sein. Auch die Verteilung der Teilsimulationen auf die unterschiedlichen Rechenressourcen kann durch denselben Allokationsalgorithmus erfolgen. Auch ein Benutzungsinterface braucht nicht jedes Mal neu entwickelt zu werden, sondern es reichen kleine anwendungsgebietsspezifische Anpassungen. Für die konkrete Zerlegung eines übergebenen Projekts, die Entscheidung für eine Lösungsmethode, die Bewertung der Ergebnisse und ähnliches ist dann jedoch spezifisches Wissen erforderlich. Auf die Integration dieses Wissens wird im nachfolgenden Unterkapitel eingegangen.

Um die dezentralen Teilsimulationen zu koordinieren sowie mit dem Benutzer zu interagieren, eignen sich Softwareagenten. Diese in ihren Verhaltensspielräumen autonomen Softwareeinheiten sind in der Lage, untereinander und mit ihrer Umgebung zu kooperieren [VDI/VDE 2653]. Für Simulationen werden Agenten bereits erfolgreich eingesetzt [ByGö17, JJW18]. So kann bei einer verteilten Simulation jeder Agent eines Agentensystems ein Teilproblem vertreten.

## 2.2 Integration anwendungsspezifischen Wissens in das Agentensystem

Ziel ist es, dass für jedes Anwendungsgebiet das benötigte Wissen einfach in das Agentensystem integriert und ausgetauscht werden kann. Zudem soll es möglich sein, bereits formalisiert festgehaltenes Wissen in externen Quellen in das System einzubinden. Da Experten in den Anwendungsgebieten meist wenig bis keine Erfahrung mit der Programmierung von Parallelisierungssystemen haben, soll das Wissen durch eine übersichtlich gestalteten Schnittstelle in das System eingespeist werden, so dass der Systemaufbau und -ablauf hierfür nicht relevant ist Um diese Anforderungen zu erfüllen, soll das Wissen durch Wissensagenten in das System eingebunden werden. Hierbei wird auf das Konzept von [Pech13] zurückgegriffen, mit dem es Agenten möglich ist, Wissen aus externen Datenquellen zu extrahieren und für den jeweiligen

Anwendungsfall aufzuarbeiten. Um das Wissen leicht bearbeitbar zu gestalten, wird hierbei für jeden Wissensbereich ein separater Agent vorgesehen (siehe Abbildung 2).

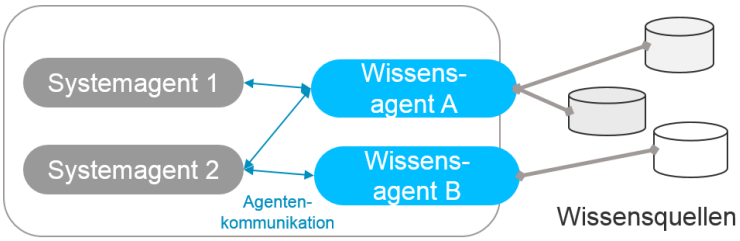


*Abbildung 2: Wissensagenten zur Einbindung externer Wissensquellen*

Die Aufteilung in die verschiedenen Wissensbereiche, durch welche das Wissen strukturiert und dadurch einfacher zu erstellen bzw. bearbeiten wird, ist in Tabelle 1 gegeben.

*Tabelle 1: Für das Framework benötigte Wissensbereiche*

| **Bereich** | **Beinhaltete Informationen** |
|---|---|
| Aufgabenstellung | • Simulationsdateiformat<br>• Anforderungen |
| Zerlegung | • Zerlegungsverfahren mit Ablauf<br>• Zerlegungsregeln<br>• Analyse für Zerlegung |
| Methodenfindung | • Methoden mit Ablauf<br>• Analyse für Methodenfindung<br>• Regeln für Methodenfindung<br>• Austauschformen |
| Bewertung | • Ergebnisbewertung<br>• Ergebnisintegrationsregeln und -abläufe |
| Darstellung | • Ergebnisdarstellungsform<br>• Zusätzliche Darstellungselemente |

Die Wissensagenten bieten ihren Dienst den anderen Agenten an und verteilen so das anwendungsgebietsspezifische Wissen im Agentensystem. Neben der Verwendung bereits vorhandener Wissensquellen, kann auch ein Experte das benötigte Wissen den Wissensagenten bereitstellen.

### 2.3 Einbindung vorhandener Softwarewerkzeuge zur Simulation

Neben allgemeinen Abläufen und dem anwendungsgebietsspezifischen Wissen, wird auch Wissen zur Anbindung von Softwaretools zur Simulation benötigt. Da diese Softwaretools häufig sehr mächtige Simulationswerkzeuge sind, die sich für mehr als ein Anwendungsgebiet verwenden lassen, wird hierfür eine Bibliothek an Toolagenten aufgebaut, um bereits vorhandene Agenten für ein Softwaretool wiederverwenden zu können. Diese Toolagenten haben die Aufgabe, ihr jeweiliges Softwaretool über die meist bereits vorhandene Schnittstelle (API) zu steuern. Hierzu können sie bei Bedarf erzeugt werden und erhalten dann Anweisungen von einem Agenten. Diese Anweisungen führen sie dann mit ihrem Softwaretool aus und liefern die Ergebnisse an den

auftraggebenden Agenten zurück. Eine Liste der abstrahierten Agentennachrichten mit den Anweisungen kann Tabelle 2 entnommen werden. Der genaue Inhalt der entsprechenden Nachrichten ist anwendungsgebietsspezifisch und die zu versendenden Objekte werden bei Bedarf dynamisch erzeugt.

*Tabelle 2: Die wichtigsten Auftrags-Nachrichten an einen Toolagenten*

| **Nachrichten ID** | **Beschreibung** |
|---|---|
| LoadModel | Übergibt das Modell, mit dem gearbeitet werden soll |
| AnalyzeModel | Analysiert das Modell, um über Zerlegung und Methoden zu entscheiden |
| ChangeModel | Verändert das Modell indem z. B. Elemente hinzugefügt oder entfernt werden, bzw. die Parametrisierung angepasst wird |
| SimulateModel | Übergibt die zur Simulation zu verwendenden Konfigurationen |
| ChangeSimulation | Anhalten, Rekonfigurieren, etc. während einer Simulation |

Da eine Simulation in der Durchführung einige Zeit dauern kann und häufig mehrere nacheinander zu erfolgende Schritte abgearbeitet werden müssen, stehen Toolagenten immer nur für den Agenten zur Verfügung, für den sie erzeugt wurden. Sobald sie nicht mehr benötigt werden, beenden sie sich, um die Ressource wieder freizugeben. Unterstützt ein Werkzeug die Nutzung mehrerer Instanzen auf einer Rechenressource, so kann bei Bedarf ein weiterer Toolagent auf dieser Ressource erzeugt werden.

## 2.4 Systemarchitektur des agentenbasierten Assistenzsystem zur parallelen Simulation im Engineering

Eine Übersicht über die im Framework enthaltenen Agententypen ist in Abbildung 3 dargestellt. Zusätzlich zu den dargestellten Agententypen werden das Agentenmanagement und der Directory Faciliator, welche die javabasierte Agentenentwicklungsumgebung JADE [BCG07] bereitstellt, verwendet.

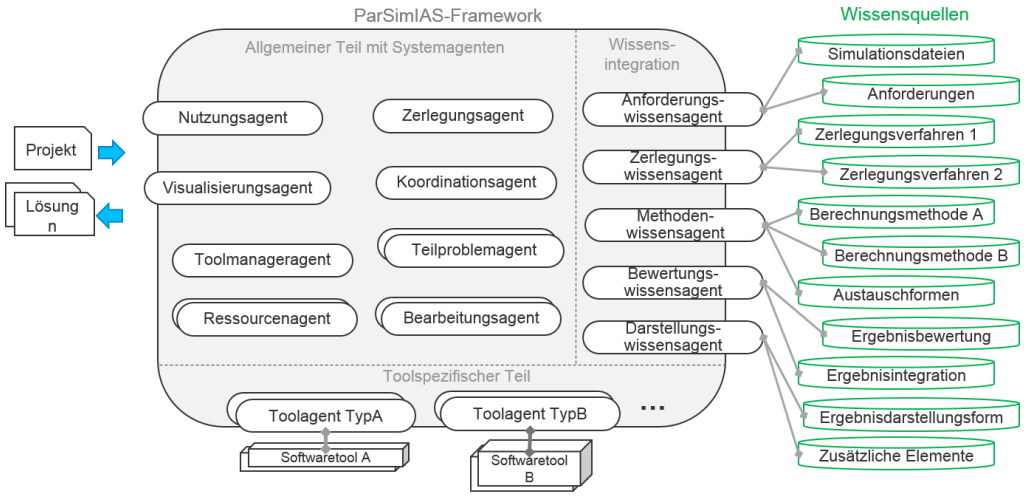


*Abbildung 3: Struktur des Agentensystemframeworks ParSimIAS mit Beispiel-Wissensquellen*

Zur Interaktion mit dem Benutzer stehen der Nutzungsagent und der Visualisierungsagent zur Verfügung. Der Nutzungsagent bietet einen Überblick darüber, was im Agentensystem passiert,

um die internen Vorgänge nachvollziehbar anzuzeigen und dadurch die Akzeptanz zu erhöhen. Mit dem Visualisierungsagenten und seiner GUI können Projekte übergeben werden sowie die Lösungen ausgegeben werden. Ein Koordinationsagent ist für das Gesamtprojekt zuständig und koordiniert den Ablauf. Mithilfe eines Zerlegungsagenten wird das Projekt in Teilprobleme zerlegt. Hierfür werden Regeln und Methoden bei einem Wissensagenten angefragt. Jedes Teilproblem wird von einem Teilproblemagent verwaltet, welcher mithilfe des erfragten Wissens nach Lösungsmöglichkeiten sucht. Für jeden gefundenen Ansatz, der simuliert werden soll, wird ein Bearbeitungsagent erzeugt. Dieser erhält vom entsprechenden Wissensagenten die Informationen über notwendige Schritte, Kommunikationsvorgänge und Integrationsverfahren von externen Informationen. Die Rechenressourcen werden durch jeweils einen Rechenagenten verwaltet. Dieser schreibt die bei ihm verfügbaren Dienste zur Simulation aus. Soll ein Softwarewerkzeug genutzt werden, so erzeugt der Ressourcenagent für den anfragenden Agenten einen entsprechenden Toolagenten. Dieser kann die Aufträge mit dem Softwaretool ausführen und die Ergebnisse zurückliefern. Um die Allokation der verschiedenen Simulationsaufgaben auf den Rechenressourcen zu koordinieren, gibt es einen Toolmanageragent. Dieser unterstützt die anderen Systemagenten bei der Suche nach einer Ressource und vermittelt die Aufgaben entsprechend ihren Anforderungen und ihrer Priorität. Dabei wird die Priorität entsprechend den Erfolgsaussichten und der Redundanz des Teilproblems vergeben.

# 3 Anwendungsbeispiel des Frameworks ParSimIAS

Um die Abläufe im Agentensystem besser nachvollziehen zu können, werden in diesem Kapitel nach der Beschreibung der Realisierung Anwendungsbeispiele betrachtet. Hierbei wird sowohl auf die notwendige Vorbereitung für die Nutzung in einem Anwendungsgebiet eingegangen, als auch auf die Bearbeitung eines konkreten Projekts.

## 3.1 Realisierung des ParSimIAS-Frameworks

Das Agentenframework ist mithilfe von JADE [BCG07] in Java implementiert. Für den Regelungsentwurf sind Toolagenten für OMEdit, ein auf Modelica basierendes Simulationswerkzeug, und MATLAB/Simulink entwickelt worden. Für das Anwendungsgebiet der multiphysikalischen Simulation wurden Toolagenten für COMSOL Multiphysics entwickelt. Auch das Simulationswerkzeug FEKO kann von Softwareagenten verwendet werden [GJV+17]. Die Kommunikationsschnittstelle dieser Agenten zum restlichen System, die Mailbox, ist für alle Softwaretools gleich strukturiert, so dass nur das Interface zum Softwaretool für einen neuen Toolagenten zu entwerfen ist. Die Nachrichten rufen dann, entsprechend ihrer ID, in der Mailbox die entsprechenden Schnittstellenfunktionen des Softwaretools auf. Der Austausch zwischen den Wissensagenten und den Systemagenten erfolgt durch XML-basierte Nachrichten. Systemagenten können dabei das benötigte Wissen von den Wissensagenten per Request erfragen.

## 3.2 Einsatz von ParSimIAS für den Regelungsentwurf

Die Verwendung von ParSimIAS wird zunächst an einem einfachen Beispiel aus dem Regelungsentwurf gezeigt. Dabei werden dezentrale Regelungen mit schwacher Kopplung

betrachtet. Hierfür muss das Framework mit dem für den Regelungsentwurf benötigten Wissen befüllt werden (siehe Tabelle 3).

*Tabelle 3: Auszug aus dem Anwendungsgebietsspezifischen Wissen für den Regelungsentwurf*

| | | |
|---|---|---|
| Aufgaben-stellung | Simulationsdateiformat | • .slx (MATLAB/Simulink)<br>• .mo (OMEdit) |
| | Anforderungen | • Führungsgröße<br>• Überschwingzeit<br>• Beruhigungszeit<br>• Anstiegszeit … |
| Zerlegung | Zerlegungsverfahren mit Ablauf | • Ein- und Ausgangsgrößenpaare |
| | Analyse für Zerlegung | • Modell nach Ein- und Ausgängen untersuchen |
| | Zerlegungsregeln | • Für schwache Kopplung immer Ein- und Ausgangsgrößenpaarzerlegung |
| Methoden-findung | Methoden mit Ablauf | • Unterschiedliche Regler (PID, Fuzzy, …)<br>• Unterschiedliche Parametrisierungsverfahren |
| | Analyse für Methodenfindung | • Sprungantwort<br>• Relay-Feedback |
| | Regeln für Methodenfindung | • WENN *selbsteinstellend* DANN *FOPDT*<br>• WENN oszilierend DANN … |
| | Austauschformen | • Reglerbewertung nach jeder Simulation |
| Bewertung | Ergebnisbewertung | • Endbewertung $= [\sum_{i=1}^{6} Erfüllungsgrad_i * Gewichtung_i] \div \sum_{i=1}^{6} Gewichtung_i$ |
| | Ergebnisintegrationsregeln und -abläufe | • Teilregler in Gesamtsystem integrieren<br>• Wenig vielversprechende Ansätze abbrechen |
| Darstellung | Zusätzliche Darstellungselemente | • - (keine besonderen Darstellungen notwendig) |
| | Ergebnisdarstellungsform | • Bewertung als Float<br>• Bild des Einschwingverhaltens<br>• Link zu Ordner mit entworfenen Reglern |

Nachdem das anwendungsspezifische Wissen durch die Wissensagenten integriert ist, können das Agentensystem gestartet werden und Projekte bearbeitet werden. Als Beispiel dient hier eine schwach gekoppelte Übertragungsfunktion mit 2 Ein- und Ausgangsgrößenpaaren. Der Benutzer übergibt dabei über die links in Abbildung 4 dargestellte Eingabemaske das Projekt an den Visualisierungsagenten. Der übersichtlichkeitshalber wird auf die notwendige Kommunikation mit den Wissensagenten im Folgenden nicht weiter eingegangen.

Der Visualisierungsagent erstellt daraus ein neues Projekt und übergibt dies an den Koordinationsagenten. Dieser beauftragt den Zerlegungsagenten mit der Zerlegung. Dazu wird eine freie Ressource gesucht und mindestens ein Toolagent gestartet, um eine Analyse der Simulationsdatei durchzuführen. Anhand der Ergebnisse und der hinterlegten Regel erfolgt eine Aufteilung in Teilprobleme. Der Koordinationsagent erstellt für jedes Teilproblem einen

Teilproblemagent, der dieses koordiniert. Bei diesem Beispiel wird je ein Teilproblemagent für jedes Ein- und Ausgangsgrößenpaar erstellt, da nur eine schwache Kopplung vorliegt. Diese Teilproblemagenten erstellen für jede Lösungsalternative, die sie simulieren wollen, einen Bearbeitungsagenten, der für sie zuständig ist. Die Bearbeitungsagenten verhandeln mit den Ressourcenagenten, um einen geeigneten Toolagenten zu finden. Mit diesem führen sie ihren Simulationsauftrag aus. Dazu werden zuerst die Komponenten des Teilreglers sowie eine Anregung zur Regelstrecke hinzugefügt. Anschließend werden die Einstellparameter in einem iterativen Prozess optimiert. Die Teilproblemagenten sammeln die Ergebnisse und teilen den anderen Teilproblemagenten die beste Variante mit. Der Koordinationsagent fügt am Ende mithilfe des Zerlegungsagenten die besten Teillösungen zu einer Gesamtlösung zusammen. Während der ganzen Zeit können über den Nutzungsagenten die Vorgänge im System verfolgt werden und über den Visualisierungsagenten das Projekt aktualisiert bzw. die Ergebnisse angezeigt werden (siehe Abbildung 4, rechts).

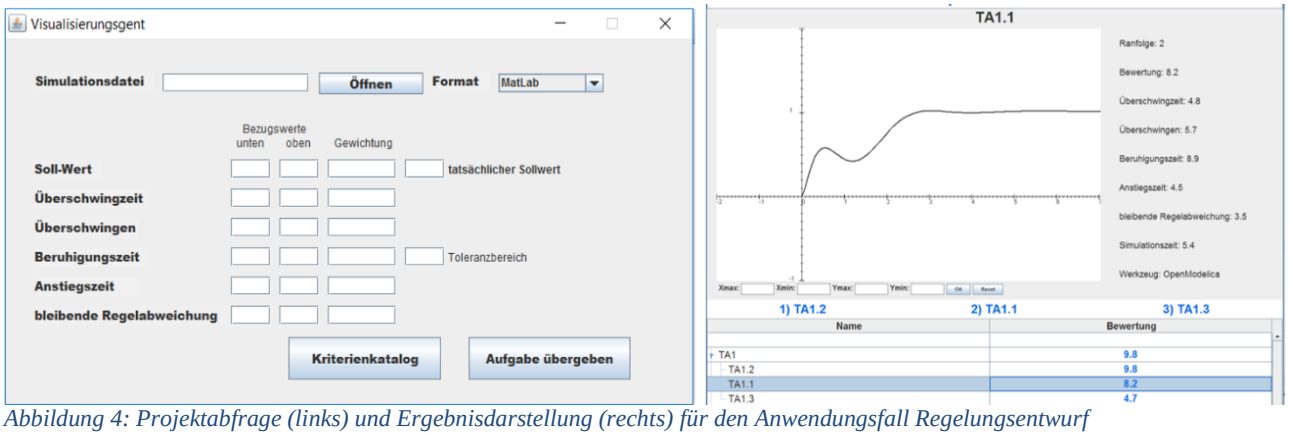


*Abbildung 4: Projektabfrage (links) und Ergebnisdarstellung (rechts) für den Anwendungsfall Regelungsentwurf*

## 3.3 Einsatz von ParSimIAS für die multiphysikalische Simulation

Als weiteres Anwendungsbeispiel soll eine multiphysikalische Simulation verwendet werden. Hierbei geht es beispielsweise darum, Wellenausbreitung, Temperatur- und Spannungsverläufe von Bauteilen und Schaltungen zu untersuchen. Das notwendige anwendungsspezifische Wissen ist Tabelle 4 zu entnehmen. Als Projekt wurde das Modell eines Hohlleiters übergeben und als Anforderungen die Temperatursimulation und die Wellenausbreitung eingegeben. Die Ergebnisse, die das Agentensystem nach mehreren Iterationen liefert, können Abbildung 5 entnommen werden. Der Ablauf der Bearbeitung entspricht dabei dem des in Kapitel 3.2 beschriebenen.

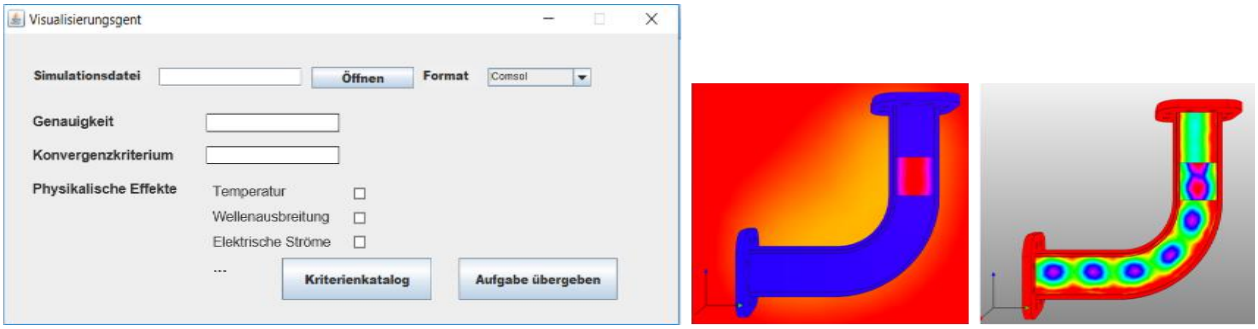


*Abbildung 5: Projektabfrage (links) und Ergebnisse für eine Hohlleitersimulation mit Dielektrikum (mittig: Temperatur, rechts: Wellenausbreitung)*

*Tabelle 4: Auszug aus dem anwendungsgebietsspezifischen Wissen für die multiphysikalische Simulation*

| | | |
|---|---|---|
| Aufgaben-stellung | Simulationsdateiformat | • .mph (COMSOL Multiphysics) |
| | Anforderungen | • Genauigkeit<br>• Physikalische Effekte<br>• Konvergenzkriterium |
| Zerlegung | Zerlegungsverfahren mit Ablauf | • Physikalische Effekte<br>• Räumliche Zerlegung |
| | Analyse für Zerlegung | • Modelleigenschaften und zugrunde liegende Gleichungssysteme |
| | Zerlegungsregeln | • Ähnliche Gebiete zusammenfassen<br>• Physikalische Effekte separieren |
| Methoden-findung | Methoden mit Ablauf | • Finite-Element-Methode<br>• Boundary-Element-Methode<br>• Unterschiedliche Solver |
| | Analyse für Methodenfindung | • Modellparameter wie z. B. *shapeOrder*, *maxConductionFactor*, … |
| | Regeln für Methodenfindung | • Vorschlag vom Softwaretool verwenden<br>• WENN *Parameter A > X* DANN *Solver y* |
| | Austauschformen | • Ergebnisse nach jedem Iterationsschritt publizieren |
| Bewertung | Ergebnisbewertung | • Zeit und Qualität (Diskretisierung) |
| | Ergebnisintegrationsregeln und –abläufe | • Physikalische Randbedingungen korrigieren<br>• Zu langsame Vorgänge abbrechen |
| Darstellung | Zusätzliche Darstellungselemente | • Extrafenster mit 3D-Darstellung |
| | Ergebnisdarstellungsform | • Jede Physik extra<br>• Link zu Ordner mit Simulationsergebnissen |

# 4 Zusammenfassung und Ausblick

Durch die Auslagerung des anwendungsspezifischen Wissens aus dem Agentensystem zur parallelen Simulation, ist es Experten eines Anwendungsgebiets möglich, das notwendige Wissen bereitzustellen oder auszutauschen. Ein mit dem Framework generiertes Agentensystem kann durch anwendungsspezifisches Wissen beispielsweise ein regelungstechnisches oder ein multiphysikalisches Modell auswerten, verschiedene vielversprechende Regelungsansätze bzw. Simulationsansätze parallel verfolgen und dem Benutzer die bewerteten Ergebnisse zur Auswahl präsentieren. Durch die Flexibilität des Agentensystems kann jede Teilsimulation mit dem am besten geeigneten Simulationswerkzeug durchgeführt werden. Auch werden ungenutzte Rechenressourcen für alternative Berechnungen und Konfigurationen genutzt, um so in gleicher Zeit mehr Varianten zu untersuchen und die Qualität der Ergebnisse zu verbessern. Das bestehende allgemeine Framework kann zukünftig um lernfähige Agenten erweitert werden, wie es für den konkreten Fall der multiphysikalische Simulationen bereits erfolgte [VGJ+18]. Hierbei kann aus bereits gelösten Projekten für zukünftige gelernt werden.

## 5 Danksagung

Die Autoren bedanken sich bei der Deutschen Forschungsgemeinschaft für die Unterstützung bei dem Projekt GekoProAg (RU 720/11-2 & WE 5312/8-2).

## 6 Literatur

[ASW17] Ashtari Talkhestani, B.; Schlögl, W.; Weyrich, M.: Synchronisierung von digitalen Modellen - Anwendung einer Ankerpunktmethode für Fertigungszellen. In atp edition - Automatisierungstechnische Praxis, vol. 59, no. 07–08, pp. 62–69, 2017.

[BCG07] Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. Wiley, 2007.

[ByGö16] Beyer, T.; Göhner,P.: Agentenbasiertes Assistenzsystem zur Entwicklung und Adaption von automatisierten Systemen am Beispiel von Aufzugsystemen. In Automation 2016 07.-08.06.2016 Baden-Baden, 2016.

[CMR+17] Clement, S. J., McKee, D. W., Romano, R., Xu, J., Lopez, J. M., Battersby, D.: The Internet of Simulation: Enabling agile model based systems engineering for cyber-physical systems. In: System of Systems Engineering Conference, 1-6, 2017

[Durf01] Durfee, E. H.: Distributed Problem Solving and Planning. In: Luck M., Mařík V., Štěpánková O., Trappl R. (eds) Multi-Agent Systems and Applications. ACAI 2001. Lecture Notes in Computer Science, vol 2086. Springer, Berlin, Heidelberg

[FMI18] Functional Mockup Interface: http://fmi-standard.org/tools/ Stand: 03.04.18

[GJV+17] Grabmaier, S.; Jüttner, M.; Vögeli, D.; Rucker, W. M.; Göhner, P.: Numerical framework for the simulation of dielectric heating using finite and boundary element method. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 2017.

[IEEE 1278] IEEE Standard for Distributed Interactive Simulation. 2012

[IEEE 1516] IEEE Standard for Modeling and Simulation High Level Architecture. 2010

[JJW18] Jung, T., Jazdi, N., Weyrich, M.: Dynamische Co-Simulation von Automatisierungssystem und ihren Komponenten im Internet der Dinge – Prozessorientierte Interaktion von IoT-Komponenten. EKA, 2018.

[Pech13] Pech, S.: Agentenbasierte Informationsgewinnung für automatisierte Systeme. IAS-Forschungsberichte Band 1/2014.

[PPS+13] Pronk, S., Páll, S., Schulz, R., Larsson, P., Bjelkmar, P., Apostolov, R., ... & Hess, B.: GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit. In Bioinformatics, 29(7), 845-854, 2013.

[VDI/VDE 2653] Agentensysteme in der Automatisierungstechnik - Grundlagen. 2017

[VGJ+18] Vögeli, D., Grabmaier, S., Jüttner, M., Weyrich, M., Göhner, P., Rucker, W. M.: Intelligent and Distributed Solving of Multiphysics Problems Coordinated by Software Agents. International Conference on Agents and Artificial Intelligence, Madeira, Portugal, 2018.

[VNG+17] Vögeli, D., Jazdi, N., Grabmaier, S., Jüttner, M., Weyrich, M., Göhner, P., Rucker, W. M.: Softwareagenten zur zuverlässigen Durchführung dezentraler multiphysikalischer Simulationen. In at-Automatisierungstechnik, 65(11), 793-803, 2017.