T U M

I N S T I T U T F Ü R I N F O R M A T I K

Grid-Workflow-Management-Systeme für die Ausführung wissenschaftlicher Prozessabläufe

Ekaterina Elts, Hans-Joachim Bungartz

TUM-I1004
Februar 10

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-02-I1004-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Grid-Workflow-Management-Systeme für die Ausführung wissenschaftlicher Prozessabläufe

Ekaterina Elts, Hans-Joachim Bungartz

Zusammenfassung

Für die Ausführung komplexer wissenschaftlicher Prozessabläufe (Workflows) in einer verteilten und heterogenen Rechner- und Softwareumgebung braucht man speziell darauf ausgerichtete Workflow-Management-Systeme. In diesem Bericht wurden einige international anerkannte Workflow-Management-Systeme untersucht und verglichen. Dabei wurden die besonderen Anforderungen an die wissenschaftlichen Workflows (im Gegensatz zu Geschäftsprozessen) beachtet und die jeweiligen Besonderheiten der betrachteten Systeme bezüglich der Anwendung im Bereich der Wissenschaft herausgearbeitet.

Keywords: wissenschaftliche Workflows, Workflow-Management-System, Grid.

Inhaltsverzeichnis

1 Einführung

Mit der Entwicklung der Grid-Technologie erhöht sich ständig die Komplexität der Systeme, die die Wissenschaftler, Ingenieure, und Praktiker verstehen und modellieren wollen. Im Fokus zahlreicher wissenschaftlicher Projekte stehen immer häufiger komplizierte Workflows, die zur Lösung unterschiedlicher wissenschaftlicher Fragestellungen (z.B. in der Bioinformatik, Hochenergiephysik, Medizin usw.) eingesetzt werden.

In der letzten Zeit wurden und werden viele Projekte gefördert, die die Entwicklung von geeigneten Systemen für Modellierung, Verwaltung und Ausführung der Workflows in Grid-Umgebungen als Thema haben. Hierbei wird häufig das Ziel verfolgt, den Fachanwendern einfache Methoden zur Verfügung zu stellen, mit denen komplexe Anwendungen erstellt und transparent auf den verfügbaren Ressourcen ausgeführt werden können. Zu solchen Systemen gehören z.B. Taverna, Triana, Kepler, Askalon und andere wissenschaftliche Workflow-Management-Systeme (WMS).

Trotz der Attraktivität dieser Systeme werden diese leider kaum im Bereich des CSE[1] genutzt. Nur einige Gründe dafür sind:

- Das System ist nicht Open Source.
- Das System ist zu kompliziert für Nicht-IT-Spezialisten.
- Die GUI Funktionalität ist zu beschränkt.
- Grid-Kenntnisse sind für die Workflowausführung in Grid-Umgebungen notwendig.
- Das System erfüllt die Benutzeranforderungen nicht (Der Anwendungsbereich ist sehr spezifisch, es gibt keine Parameterstudienunterstützung, es gibt Probleme mit der Ausführung des Legacy Codes, usw.).
- Fehlende Portalzugangsmöglichkeit: häufig passiert es, dass Wissenschaftler ein Workflow-System nicht herunterladen und installieren wollen, auch wenn es frei verfügbar ist.

Es ist wohlbekannt [1, 2], dass viele Wissenschaftler und Ingenieure der Einführung von Grid-Technologie oft widerstreben wegen des damit verbundenen Benutzungsoverheads. Das GridSFEA-System (Grid-Based Simulation Framework for Engineering Applications) [3, 1] wurde entwickelt, um die Lücke zwischen CSE-Anwendungen und dem Grid zu überbrücken und einen einfachen Benutzungsmechanismus anzubieten. Das Framework ermöglicht die Ausführung von verschiedenen Anwendungsszenarien auf dem Grid sowie anwendungsunabhängige Parameterstudien und Checkpoint-basierte Migration für lange Simulationsläufe im Grid. Es gibt auch eine Portalzugangsmöglichkeit: Das GridSFEA-Portal basiert auf dem GridSphere-Portal und erlaubt den Benutzern, die Dateien in das Grid zu laden, Simulationsaufgaben zu bestimmen sowie den Status zu überwachen. Das GridSFEA-System wird bereits für die Ausführung vieler Szenarien (z.B. Molekulardynamik, Astrophysik, Fluiddynamik) auf dem Grid erfolgreich benutzt [3, 1]. Das Einzige, was dem GridSFEA-System noch fehlt,

[1] Computational Science Engineering

ist die Möglichkeit, einen Workflow zu erstellen und auszuführen. Deshalb sollte das GridSFEA-Framework um ein Workflow-Management-System erweitert werden. Solche Erweiterung würde dem Benutzer von GridSFEA sowie auch der CSE-Gemeinde ermöglichen, von den Vorteilen der beiden Systemen zu profitieren. Neue Anwendungsszenarien, wie Grid-Workflows mit Parameterstudien- und Migrationsunterstützung würden dann sich herausbilden. In diesem Bericht untersuchen wir einige anerkannte WMS und versuchen ein passendes WMS für die Integration mit GridSFEA zu ermitteln.

2 Wissenschaftliche Workflows und Workflow-Management-Systeme

Unter dem Begriff "*Workflow*" versteht man eine Folge von Einzeltätigkeiten (Aktivitäten, Modulen), die schrittweise ausgeführt werden, um ein Ziel zu erreichen. Die einzelnen Aktivitäten stehen in einem Workflow in Abhängigkeit zueinander, die folgende Aktivität ist normalerweise durch den Ausgang der jeweils vorangehenden determiniert.

Zuerst wurden solche Workflows für die Beschreibung von Geschäftsprozessen eingeführt [4], derzeit aber werden Workflows immer häufiger im Bereich der Wissenschaft verwendet, wie z.B. bei der Simulation, der Analyse oder der Ausführung von Experimenten. Wissenschaftliche Workflows sind ein Verfahren, komplexe wissenschaftliche Arbeitsabläufe zu beschreiben. Dabei wird Wissenschaftlern die Definition von Programmen von der Analyse, über die Durchführung von Berechnungen bis zur Zusammenfassung der Ergebnisse gegenüber dem traditionellen Programmieren, wesentlich vereinfacht. Außerdem können wissenschaftliche Workflows jederzeit und beliebig oft wiederholt werden, was wesentlich zur Reproduzierbarkeit der wissenschaftlichen Ergebnisse beiträgt.

Für die Modellierung, die Verwaltung und die Ausführung der wissenschaftlichen Workflows braucht man speziell darauf ausgerichtete *Workflow-Management-Systeme*. Da meist große Datenmengen anfallen und hohe Rechenkapazitäten erforderlich sind, nutzen viele WMS eine Grid-Infrastruktur und unterstützen durch entsprechende Programme die Ausführung von Workflows in verteilten und heterogenen Umgebungen. Ein WMS besteht aus mehreren Komponenten und bietet mindestens an:

- Ein Workflow-Beschreibungsmodell definiert die Struktur, in der die Aktivitäten in einem Workflow miteinander verbunden sind. Vor allem beschreibt es, in welcher Reihenfolge die Aktivitäten (Module) erledigt werden müssen, sowie die Abhängigkeit der Module zueinander.

- Eine Workflow-Engine (Workflow-Manager) ist dafür verantwortlich, einen Workflow durchzuführen, in dem sie externe Anwendungen startet, um einzelne Aktivitäten des Workflows durchzuführen. Die Engine bildet den Workflow auf die unterlegene Grid-Ressourcen ab, und manipuliert dessen Laufzeitverhalten entsprechend der Workflowbeschreibung.

- Benutzerunterstützung auf verschiedenen Ebenen für Workflowerstellung, Ausführung und Kontrolle.

In diesem Bericht werden einige WMS untersucht. Dabei werden die besonderen Anforderungen an wissenschaftlichen Workflows (im Gegensatz zu Geschäftsprozessen) beachtet und die jeweiligen Besonderheiten der betrachteten Systeme bezüglich der Anwendung im Bereich der Wissenschaft herausgearbeitet. Die folgenden international anerkannten WMS werden betrachtet:

- Taverna
- Triana
- PGrade/ WS-PGrade
- WS-VLAM
- Kepler
- ASKALON
- ICENI
- Pegasus
- K-Wf Grid GWES
- Karajan
- gEclipse
- UNICORE

Die WMS unterscheiden sich in verschiedenen Aspekten. Sie benutzen verschiedene Workflow-Engines, verschiedene Workflow-Beschreibungssprachen, verschiedene Workflow-Formalismen und verschiedene Grid-Middleware.

3 Bewertungskriterien für Workflow-Management-Systeme

Die wichtigsten Kriterien, nach denen man verschiedene WMS für die Ausführung von wissenschaftlichen Workflows bewerten könnte, sind unserer Meinung nach aufgrund vieler Artikel und Berichte [4, 5, 2, 6, 7, 8] die folgenden:

1. Offene Lösung der Software (Open Source) – damit die notwendigen Modifikationen und Anpassungen gemacht werden können.
2. Universalität (vielseitige Anwendbarkeit) der Software – damit die Workflows für Prozessbeschreibungen aus verschiedenen wissenschaftlichen Bereichen erstellt werden können.
3. Unterstützung der Standartmiddleware, Web- / Grid-Services.
4. Vorhandensein eines intuitiv bedienbaren Editors (Workflowkompositionstool / GUI) – damit Workflowerstellung einfach und ohne spezielles Informatik-Fachwissen gemacht werden kann.

5. Unterstützung des Legacy Codes – damit der Benutzer sein Programm als eine Komponente des Workflows ausführen kann, ohne das Programm zu ändern.
6. Umfangreiche Bibliothek von Standard-Workflowkomponenten.
7. Unterstützung der Entwicklung der neuen Workflowkomponenten.
8. Unterstützung von Parameterstudien – da die meisten Gridanwendungen mit teilbaren Daten arbeiten, ist ein Workflowwerkzeug ohne Parameterstudiumunterstützung nicht praktisch einsetzbar [7].
9. Unterstützung der großen (langlebigen, datenintensiven) Workflows:
   (a) da die Workflowausführung häufig lange dauert, soll die Software mindestens die Trennung zwischen Editor und Engine (dem Workflow-Manager) ermöglichen [4], damit man Workflow im Batch-Mode ausführen könnte;
   (b) Unterstützung der hierarchischen/eingebetteten Workflowerstellung – damit man die einfachen Workflows als Komponente eines größeren, komplizierteren Workflow wiederbenutzen kann, für den einfachen Aufbau beliebig komplexer Prozessdefinitionen;
   (c) Unterstützung von parallelen Prozessen/Jobs.
10. Fehlertolerante Behandlung von Workflows – aufgrund der heterogenen Natur des Grids und der sehr hohen Anzahl an Ressourcen ist die Wahrscheinlichkeit von Ausfällen hoch. Aus diesem Grund ist es wichtig, dass im Fall von Berechnungsabbrüchen die Workflowausführung dennoch fortgesetzt werden kann, ohne dass ein inkonsistenter Zustand eintritt.
11. Überwachung von laufenden Workflows – damit man den Status der Workflowausführung beobachten kann. In idealer Weise soll dies auch schrittweise Ausführung ermöglichen, was für das Workflowdebugging besser wäre.
12. Dokumentation (für die Benutzern und Entwickler).
13. Interoperabilität mit verschiedenen WMS – damit die Ergebnisse eines Projekts für die anderen interessierten Projekte zugänglich gemacht werden können [5].

Bezüglich der Integration von WMS und GridSFEA ist es wichtig, dass WMS mit GT4 (Globus Toolkit 4) funktioniert, damit man GridSFEA-Services, die GT4-Services sind [1], benutzen kann. Die Workflow-Engine sollte unbedingt Web-Services starten und überwachen können, für GridSFEA-Architektur wären zusätzlich WSRF-Services wünschenswert.

4 Analyse bestehender Workflow-Management-Systeme

In diesem Abschnitt wird eine kurze Beschreibung von WMS bezüglich der obengennanten Kriterien gegeben, dabei betonen wir die Vor- sowie auch die Nachteile, die diese Systeme haben. Alle untersuchten Systeme sind international

anerkannt, für die Ausführung der wissenschaftlichen Workflows geeignet und bereits erfolgreich in diesem Gebiet eingesetzt.

4.1 Taverna[2]

Taverna ist ein Open Source WMS (Lesser General Public Lizenz), das sich vor allem auf Bioinformatik-Anwendungen und -Services konzentriert. In Taverna kann man Workflows mit der Hilfe einer GUI durch menübasierte Zusammenstellung aus Workflowkomponenten per Drag und Drop konstruieren. Der Workflow wird dann in einer eigenen XML-basierten Sprache beschrieben, die sich SCUFL (Simple Conceptual Unified Flow) nennt. Der Workflow kann direkt in Taverna ausgeführt und überwacht werden. Die Inkraftsetzung des Workflows wird durch die Workflow-Engine Freefluo realisiert, die im Hintergrund läuft. Die Engine kann alternativ auch auf einem speziellen Rechner als Service laufen. Um inhaltsreiche Datensätze zu speichern, bietet Taverna das myGrid Information Repository (MIR) – wobei es sich um einen Web-Service mit Datenbank Anbindung handelt –, das man auch durch einen speziellen Browser nach Daten absuchen kann. Fault Handling kann explizit spezifiziert werden. Dabei unterstützt Taverna den dynamischen Dienstaustausch und Jobwiederholung [9], das heißt, Fehlertoleranz wird auf Task-Ebene unterstützt. Globus Toolkit wird noch in keiner Version unterstützt [6], obwohl man aktuell versucht, Taverna mit den Grid-Systemen wie EGEE und Globus zu verbinden [7]. Unterstützung von Legacy Codes ist nicht direkt, sondern mit der Hilfe von SOAPLAB [10], einem Tool, das von Taverna unterstützt wird und Kommandozeilen-Anwendungen in Web-Services verwandeln lässt.

Hierarchische Workflows werden unterstützt, die Workflows kann man im Batch-Mode mittels Taverna Remote Execution Service (TRES) ausführen, die Workflowüberwachung wird durch Workflow-Monitor (WM) schrittweise gemacht, dafür müssen diese zwei Plug-in (TRES und WM) installiert werden. Bei den Parameterstudien bietet Taverna nur eine Standard-Möglichkeit – Liste, und dann wird auch das Kreuzprodukt oder das Skalarprodukt von mehreren Listen angeboten. Die Dokumentation für die Benutzer ist ziemlich gut, für die Entwickler scheint sie aber nicht ausreichend zu sein. Interoperabilität mit den anderen WMS wird zurzeit in Taverna nicht angeboten.

Taverna Vorteile:

- Unabhängig von Sprache, Plattform und Bereich (obwohl vor allem für Bioinformatik Anwendungen geeignet).
- Die Dienste sind als entfernte bzw. lokale Komponenten verfügbar.
- Visuelles Interface.
- Workflow Provenienz (Das System sammelt automatisch die Workflowmetadaten).

[2] Zusammenarbeit von mehreren Europaeischen Instituten und Firmen, http://taverna.sourceforge.net, aktuelle Version 1.7.1

Taverna Nachteile:

- Keine visuelle Ausgabe aus den Workflowkomponenten.
- Kein Process-/ Workflow-Checkpointing.
- Das System ist zu groß und hat viele Abhängigkeiten von diversen Bibliotheken.
- Sicherheitsfragen werden wenig betrachtet.
- Derzeit können die Taverna-Workflows keine Grids zu Ausführung der rechenintensiven Modulen verwenden [7].

4.2 Triana[3]

Triana ist ein Open Source (Apache Software Lizenz) WMS, das ursprünglich für Astrophysik-Anwendungen entwickelt wurde. Es ist in Java geschrieben, und deshalb ist es plattformunabhängig. Das semantische Modell basiert auf Datenfluss und DAG(Directed Acyclic Graph), es beinhaltet auch die grundlegenden Programmierkonstrukte wie Schleifen (do, while) und Logik (if then), die man für die Kontrolle des Datenflusses verwenden kann. Es gibt aber keine explizite Unterstützung für die Programmierkonstrukte – Schleifen und Logik werden durch Triana Komponente (Triana Units) realisiert. Die Bedienung des Workflow-Editors ist sehr intuitiv per Drag&Drop, Komponenten werden durch Datenflusskabel verbunden. Inkompatibilität von Datentypen wird noch bei der Komposition des Workflows abgefangen. Eine Komponententoolbox wird zur Verfügung gestellt, die die Komponente für die Signalverarbeitung, Bildverarbeitung, Mathematische und Audiofunktionen beinhaltet. Es gibt Unterstützung des Legacy Codes – es kann als Standard-Workflowkomponente repräsentiert werden; die Erstellung der neuen Workflowkomponenten wird durch Bibliotheken und API unterstützt. Das Triana System interpretiert BPEL (Business Process Execution Language) und seine eigene Sprache. Die Erstellung von hierarchischen Workflows wird unterstützt, große Workflows (Elternprozesse) kann man aus kleineren Workflows (Kindprozesse) zusammensetzen. Die Ausführung und Überwachung der Workflows kann vom Editor aus gesteuert werden. Der Benutzer kann ins TCS (Triana Control Service) einloggen, einen Triana-Workflow entfernt zusammenstellen und ausführen, dann während der Workflowausführung sich abmelden und periodisch wieder einloggen, um den Ausführungsstatus zu überprüfen. GT(4) Unterstützung wurde durch JavaGAT (Grid Application Toolkit) gemacht. Im Gegensatz zum Taverna werden keine Datenbanken benutzt. Triana unterstützt Service-orientierte Module. Teile der Triana-Workflows können auch als Dienste (Services) eingesetzt werden. So kann die Interoperabilität mit den anderen WMS unterstützt werden. Außerdem kann man mit der Triana GUI schon die Pegasus/Condor Eingangsdateien (DAX Format) generieren. Die Interoperabilität mit Taverna und Kepler wird auch geplant. Der Fehlertoleranz-Mechanismus basiert auf GAT Manager [11]. Ein einfaches Parameterstudium (Liste) wird unterstützt.

[3] Cardiff University, UK, http://www.trianacode.org

Triana Vorteile:

- Plattformunabhängig.
- Daten und Control Fluss.
- Sehr intuitive Bedienung der GUI.
- Umfangreiche Bibliothek der Komponenten.

Triana Nachteile:

- Die Fehlertoleranzfrage ist wenig betrachtet.
- Es fehlen informative Fehlermeldungen.
- Kein Checkpointing und Zurückrollen der Workflowausführung.

4.3 P-Grade/WS-PGrade[4]

Das P-Grade Portal [12] ist eine graphische Umgebung zum Erstellen und zur Ausführung und Überwachung von Workflows in verschiedenen Grids. Das P-Grade Portal basiert auf GridSphere-Portal [13] und hat die gleiche Funktionalität, aber mittels Portlets bzw. Applets. Das gesamte Portal ist seit Januar 2008 als Open Source (General Public Lizenz) verfügbar, leider gibt es noch keine Dokumentation für die Entwickler, nur für die Benutzer. Die älteren Versionen des P-GRADE haben Condor DAGMan-Sprache sowie Condor DAGMan-Engine benutzt, die neue Version, die sich WS-PGrade [7] nennt, benutzt ihre eigene Sprache, die auch DAG-basiert ist, aber mit der Rekursion und Schachtelung erweitert wurde, sowie auch ihre eigene Engine Xen hat. Im Gegensatz zum P-Grade Portal ist WS-PGrade nicht Open Source, es steht unter Akademiker Lizenz, den Code kann man nicht ändern.

Die folgende Middleware wird unterstützt: GT2, GT4, gLite, LCG-2.

Der Workflow-Editor kann als lokales Applet über Java WebStart einfach und komfortabel heruntergeladen werden, die Bedienung ist intuitiv. Der Workflow-Graph ist ein einfacher DAG, bei dem Graphknoten entweder Jobs (sequenzielle, MPI oder PVM) oder Legacy-Code-Services (wenn GEMLCA Legacy Code Architektur [14] in Portal eingebaut ist) sind. Jobs müssen als Executable vorliegen und werden zum Portal und von dort aus zum gewünschten Grid-Rechner hochgeladen. In P-Grade gibt es keine Möglichkeit, Web-Services auszuführen und zu überwachen, aber in WS-PGrade gibt es solche Möglichkeiten. Nach Erstellung kann der Workflow dem Portal Server übergeben werden. Alle Workflows liegen auf dem Portal Server und können von dort aus gestartet, überwacht und auch zur weiteren Modifikation im Applet heruntergeladen werden.

Parameter Studium wird gut unterstützt (Liste, Bereich, Random, aus Eingangsdatei), und das Kreuzprodukt oder das Skalarprodukt zwischen den Eingabeports des Workflowmoduls kann definiert werden. Hierarchische Workflowerstellung wird nur in WS-PGrade Portal unterstützt. Kepler-, Taverna-, und Triana-Workflows können als eingebettete Subworkfows im Portal ausgeführt

[4] Laboratory of Parallel and Distributed Systems at MTA-SZTAKI, Hungary, http://www.lpds.sztaki.hu/pgportal/v23, aktuelle Version 2.8 von 29.05.2009 (P-Grade) oder http://portal.p-grade.hu/v.23, Aktuelle Version 3.1 von April 2009 (WS-PGrade)

werden. Die Lösung basiert auf dem Grid Anwendungsrepository und Antragsdienst Service GEMLCA, der die Ausführung der fremden Workflows im P-Grade ermöglicht.

Fehlertoleranz wird auf Task-Ebene unterstützt. Wenn ein Fehler passiert, kann der Benutzer den Job neu einreichen, und der Workflow wird von dem letzten Checkpoint weitergeführt. Über Workflowausführung im Batch-Mode gibt es leider keine Information in der Dokumentation.

P-Grade/ WS-PGrade Vorteile:

- Es kann ortsunabhängig und einfach zugegriffen werden, weil es als Web-Portal eingesetzt wird.
- Beliebige Parameterstudien für Workflows.
- Interoperabilität mit Taverna, Kepler, Triana.

P-Grade/ WS-PGrade Nachteile:

- Code ist nicht Open Source (WS-PGrade).
- Keine Dokumentation für die Entwickler (P-Grade, WS-PGrade).
- Keine Möglichkeit Web-Services auszuführen und zu überwachen (P-Grade).

4.4 WS-VLAM[5]

WS-VLAM [15, 16, 17] ist ein Open Source (Apache Lizenz) GT4 basiertes, universelles, "Data driven" WMS. Das WS-VLAM wird schon in einigen Bereichen erfolgreich verwendet, wie z. B. in Bioinformatik [18], Medizin [19] usw. Das WMS besteht aus der GUI (Composer) und der Workflow-Engine. Die GUI basiert auf einer Open Source JGraph Bibliothek (http://www.jgraph.org) und erlaubt die graphische Workflowerstellung mit der Drag und Drop Technik, generiert die XML Beschreibung des Workflows automatisch und überträgt die Workflowbeschreibung mittels SOAP Protokoll an die Workflow-Engine. Inkompatibilität von Datentypen wird noch bei der Komposition des Workflows abgefangen. Es gibt einige Standard-Workflowkomponenten, die zur Verfügung gestellt werden, Legacy Code wird auch unterstützt – dafür gibt es einen Wrapper, und man braucht nur die entsprechenden Konfigurationen anzupassen, um sein Programm aufrufen zu lassen. Die Erstellung von neuen Workflowkomponenten auf Java, Python oder C++ mittels VLPort Bibliothek ist auch problemlos. Das Repository für Workflowkomponenten wurde als WSRF-Service implementiert.

Die Workflow-Engine ist als ein GT4 WSRF kompatibler Dienst implementiert. Damit ist sie eine der ersten Workflow-Engines, die dem OGF-Standard für die Execution-Management-Services folgt. Die GUI erlaubt Verbindung mit und Trennung von Workflow-Engine bei langlebigen Workflows und beobachtet Workflowausführung mittels WSRF-Nachrichten. GUI und Engine kommunizieren über WSRF. Die Erstellung von hierarchischen Workflows, sowie parallele Jobs werden unterstützt. Bei den Parameterstudien bietet WS-VLAM zwei

[5] Virtual Laboratory AMsterdam, University of Amsterdam, Projekt "Virtual Laboratory for e-Science", http://staff.science.uva.nl/~gvlam/wsvlam/

Standard-Möglichkeiten: Liste und Bereich. Das heißt, auch wenn der Workflow in WS-VLAM ein azyklischer Graph bleibt, ist es möglich die wiederholten Aktionen mit Parameterstudium oder hierarchischem Aufbau jedes Knotens zu organisieren. Auch die graphische Ausgabe (X11) aus Workflowkomponente ist möglich, GSI fähiges, privates VNC wird dafür benutzt. Außerdem wurden einige spezielle Algorithmen für die Lastverteilung und Ressourcenverwaltung auf den heterogenen Ressourcen entwickelt.

Interoperabilität mit den anderen Workflow-Management-Systemen wird unterstützt, ein in WS-VLAM definierter Workflow kann in Kepler/Taverna als einzelner Schritt ausgeführt werden, damit ist es nicht nötig, den gesamten Workflow graphisch neu zu definieren.

WS-VLAM Vorteile:

- Intuitive GUI.
- Das Tool ist relativ klein und kompakt.
- Unterstützung von großen verteilten Workflows ist stark (Batch-Mode, parallele Jobs und Parameter Studium, sowie Algorithmen für Lastverteilung und Ressourcenverwaltung auf der heterogenen Ressourcen).
- Interaktion mit dem Workflow zu Laufzeit.
- Semantische Anmerkung und Suche nach Komponenten mittels der Ontology.

WS-VLAM Nachteile:

- Die Fehlertoleranzfrage ist wenig betrachtet, kein Checkpointing und Zurückrollen der Workflowausführung.

4.5 Kepler[6]

Kepler ist ein Open Source (BSD Lizenz) Java-basiertes wissenschaftliches WMS, das auf Ptolemy-II System (ein visuelles Modelling Tool in Java, [20]) gebaut wurde. Das System ist relativ groß, hat viele Bibliotheken. Eine GUI (Workflow-Editor) wird zur Verfügung gestellt, mit der Drag und Drop Technik kann man Workflows aus Komponenten erstellen. Es gibt mehr als 350 betriebsfertige Workflowkomponenten ("Actors"). Die Actors werden in einem Workflow durch Data Links verbunden. Kepler hat mehrere Actors, die unterschiedliche Dienste in einem Workflow aufrufen können, von Web-Services bis zu GriddLeS. GriddLeS bietet die Unterstützung für Legacy Software, die nicht für die verteilte Ausführung entwickelt ist; das ist ein Tool für Erstellung von Grid-Workflows, die Legacy Software benutzen. Es gibt auch einen “External Execution” Actor, mit dem man in einem Workflow Kommandozeilen-Anwendung ausführen kann. Die Actors für Grid-Services und Globus-Grid-Jobs (GRAM, GT2) stehen auch zur Verfügung, aber man braucht Grid-Kenntnisse dafür, und zwar RSL-Sprache für GT2 Job Submission. Das Repository für Workflowkomponenten bietet einen

[6] A cross-project collaboration, https://www.kepler-project.org, aktuelle Version 1.0.0 von May 2008

zentralisierten Server an, wobei die Komponenten (Actors) und Workflows abgelegt werden können, um Herunterladen, Suche und Verteilung zu ermöglichen. Vom Workflow-Editor wird auch die Ausführung des Workflows gesteuert. Verschiedene Scheduler ("Directors") sind verfügbar. Der Workflow wird in einer eigenen Sprache – einem proprietären XML-Format – beschrieben, die sich MoML (Modeling Markup Language) nennt. Zusammengesetzte (hierarchische) Actors sind erlaubt, mehrstufige Hierarchien sind möglich. Es gibt keine eigene Unterstützung für das Parameter Studium, dafür wird Nimrod [21] verwendet. Kepler-Workflow kann im Batch-Mode ausgeführt werden – mittels "Ausführung im Hintergrund" Feature von Ptolemy. Fehlertoleranz wird auf Task- und Workflow-Ebene unterstützt. Auf der Task-Ebene wird der Dienstaustausch im Falle der Fehler durchgeführt. Auch wenn der gesamte Workflow fehlgeschlagen hat, kann Kepler partielle Ergebnisse produzieren. Komplexere Fehlerbehandlung wird auch durch die Erweiterungen an den ausnahmefähigen Aktoren möglich. Die Workflowüberwachung ist zurzeit begrenzt und befindet sich in weiterer Entwicklung. Die Erstellung der neuen Actors auf Java wird mittels API gut unterstützt. Interoperabilität mit dem Pegasus WMS wird geplant, und zwar soll die Kepler GUI ermöglichen, die Pegasus/Condor Eingangsdateien (DAX Format) zu generieren, aber es steht noch nicht zur Verfügung.

Kepler Vorteile:

- Die GUI ist sehr intuitiv.
- Viele Actors (Workflowkomponenten) werden zur Verfügung gestellt.
- Multidisziplinäre wissenschaftliche Bereiche.
- Es gibt gute Anleitungen mit vielen Beispielen von Workflows.
- Semantische Anmerkung und Suche nach Komponenten mittels der Ontology.

Kepler Nachteile:

- Benutzer braucht Grid-Kenntnisse (RSL-Sprache für GT2 Job Submission). Man muss besondere Grid-Actors verwenden, um seinen Workflow auf dem Grid laufen zu lassen, Standard-Workflows laufen nicht auf dem Grid.

4.6 ASKALON[7]

ASKALON ist eine Entwicklungs- und Laufzeitumgebung für verteilte Anwendungen (Parameterstudien und Workflows). ASKALON kann nicht nur für wissenschaftliche Anwendungen, sondern auch für Industrie- und Wirtschaftszwecke verwendet werden. Bisher wurde ASKALON in den Bereichen Online Spiele, E-Learning, Umweltkatastrophenverwaltung, Simulation von alpinen Gewässern, Astrophysik, Materialwissenschaften sowie für die Wettervorhersage eingesetzt.

ASKALON beruht auf einer Service-orientierten Architektur mit einem Scheduler, der Anwendungen auf beliebig vielen Rechnern eines Grids verteilt

[7] University of Innsbruck, http://www.askalon.org/

ausführt. Für die Optimierung werden generische Dienstgüteparameter (Laufzeit, Kosten und Zuverlässigkeit) berücksichtigt, die vom Benutzer in Form von Bedingungen vorgegeben werden können. Die Optimierungen, die vom Scheduler und vom Laufzeitsystem eingesetzt werden, beruhen auf diversen Algorithmen (z.B. genetische Algorithmen und dynamisches Programmieren), die für das Grid speziell angepasst werden.

Für die Workflowbeschreibung nutzt ASKALON seine eigene Workflow-Sprache AGWL (Abstract Grid Workflow Language). Eine GUI, die sich Teuta nennt, steht zur Verfügung. Sie unterstützt die graphische Spezifikation der Grid-Workflows, die auf einem UML Aktivitätsdiagramm basiert und ist ein graphisches Interface zum AGWL. Für die Workflowausführung wird die Globus-Middleware (GT2, GT4) benutzt. Legacy Anwendungen müssen als Services adaptiert werden, dann kann man sie in ASKALON verwenden. Hierarchische Workflows werden unterstützt.

ASKALON beinhaltet vier vollständige kompatible Tools: SCALEA für die Instrumentation und Performanzanalyse, ZENTURIO für automatische Experiment-Management, AKSUM für die automatische Engpassanalyse, PerformanceProphet für Modelling und Voraussage der Programmperformanz.

ZEN ist eine direktivbasierte Sprache für automatische Performanz und Parameterstudien. Mit dem Einfügen der ZEN-Direktiven in die arbiträren Dateien (Programmcode, Eingabedateien, Makefiles für das Programm) kann der Benutzer die Parameter für die erweiterte Programmausführung, wie Parameterstudien, Performanzanalyse oder Tuning spezifizieren.

Die meisten ASKALON Tools bieten derzeit die Unterstützung nur für die Fortran 90 (inklusive MPI, OpenMP, HPF, gemischte OpenMP/MPI, und gemischte HPF/OpenMP) parallele und verteilte Programme an. In der näheren Zukunft wird aber auch die Unterstützung für die moderne JavaSymphony (ein performanzorientiertes, paralleles und verteiltes Java-basiertes Framework) angeboten.

ASKALON Vorteile:

- Fehlertoleranzmanagement ist sehr stark und unterstützt Task- sowie Workflow-Fehlerbeseitigung.
- Checkpointing und Zurückrollen der Workflowausführung sind erlaubt.
- Performanzanalyse.

ASKALON Nachteile:

- Code ist nicht Open Source.
- Der Anwendungsentwickler muss jeden Teil des Graphes, der in die parallelen Arbeiten aufgeteilt und auf den Multiprozessorsystemen gleichzeitig ausgeführt wird, manuell definieren. Das System kann diese Information nicht aus der AWGL Datei herauslesen [7].

4.7 ICENI[8]

ICENI ist ein Open Source (ICENI Open Source Code Lizenz) WMS, es wurde auf Java geschrieben und läuft auf allen Plattformen, die Java unterstützt, inklusive Windows, Linux, und Web-Browsern. ICENI ist eine Service-orientierte Architektur, alle Workflowkomponenten werden als Service abrufbar gemacht. ICENI benutzt seine eigenen XML-basierte Workflow-Sprachen (CDL (Component Definition Language), BDL (Behaviour Definition Language), IDL (Implemetation Definition Language)) sowie seine eigene Grid-Middleware. Die Workflow-Sprache beinhaltet alle Standard-Workflow-Strukturelemente wie die Sequenz, den Parallelismus, Logik und die Iterationsschleife (deswegen nicht DAG). Ein Workflow kann mittels einer einfachen graph-basierten GUI erstellt werden oder direkt auf der Workflow-Sprache beschrieben werden. Alle Middleware, die OGSA entspricht, kann verwendet werden, Condor, GT2, SGE wurden ausprobiert. Anwendungen sind in Bereiche von Physik, Geophysik, Chemie, Biochemie, Biologie, Medizin usw. Parallele Jobs und Parameterstudien [22] werden unterstützt. Für Batch-Mode gibt es leider keine direkte Unterstützung von der GUI-Seite. Für die Legacy Codes gibt es einen binären Wrapper, die Erstellung von neuen Modulen wird auch unterstützt. Der Benutzer kann die Parameter des Moduls während der Workflowausführung ändern. Der Scheduler ist gut für die Verteilung der Workflows in den heterogenen Umgebungen geeignet und beschließt, wo ein Modul ausgeführt werden soll, auf Grund der historischen Daten über die Performanz. Die Fehlertoleranz basiert auf ICENI-Middleware. Es gibt keine Interaktion mit dem Workflow-Controller während der Laufzeit, nur mit einzelnen Diensten/Services, es beinhaltet allerdings keine Start/ Stop Befehle.

ICENI Vorteile:

- Das System ist sehr gut geeignet für die Verteilung der Workflows in den heterogenen Umgebungen.

ICENI Nachteile:

- Die Benutzerunterstützung für die Workflowerstellung ist minimal.
- Keine Unterstützung für hierarchische Workflows.

4.8 Pegasus[9]

Pegasus ist ein Open Source WMS (Globus Toolkit Public License), das für die datenintensive Anwendungen (Hochenergiephysik, Gravitationswellen-Physik und Astronomie) entwickelt wurde. Das System besteht aus Pegasus Workflow-Mapper und Condor DAGMan Workflow-Engine. Der Hauptteil des Systems wurde auf Java entwickelt, einige Tools wurden auf C geschrieben. Die Implementierung benutzt Globus und Condor/CondorG als die grundlegende Infrastruktur für die Ressourcenverwaltung und Job Submission. Das System ist

[8] Imperial College e-Science Network Infrastructure, London e-Science Center, http://www.lesc.ic.ac.uk/iceni/

[9] Planning for Execution in Grids, University of Southern California, http://pegasus.isi.edu/wms/

von Condor und Globus abhängig. Der abstrakte Workflow kann mittels Anfragen des Chimera Tool erstellt werden. Das Chimera Virtual Data System (VDS) bietet einen Katalog an, der benutzt werden kann, um eine Gruppe der Anwendungsprogramme ("transformations") zu beschreiben, anschließend ermöglicht es, alle von den o.g. Programmen erzeugten Dateien ("derivations") zu verfolgen. Dann wird der Workflow mittels VDL-Sprache (Virtual Data Language) beschrieben. Zurzeit gibt es auch eine andere Möglichkeit, einen Workflow zu erstellen. Seit 10.03.2009 gibt es eine GUI (Version 1.0.0) für die Workflowerstellung und das Monitoring. In diesem Fall werden die Workflows in einem proprietären XML Format beschrieben, das sich DAX ("DAG in XML") nennt. Hierarchische Workflows werden auch unterstützt. Die abstrakten Workflows werden dann automatisch von Pegasus auf dem Grid abgebildet. Vor der Abbildung versucht Pegasus den abstrakten Workflow zu reduzieren – es erkennt, ob die Zwischendaten bereits vorhanden sind, und entscheidet, ob es effizienter ist, die vorhandenen Daten wiederzuverwenden und nicht neu zu berechnen. Dann bildet Pegasus den reduzierten abstrakten Workflow auf die Menge der verfügbaren Grid-Ressoursen ab und generiert einen ausführbaren Workflow. Bei Pegasus gibt es zwei Methoden für die Ressourcenauswahl, eine Randomverteilung und eine auf der Performanzvorhersagebasierende.

Die Workflows können in Batch-Mode ausgeführt werden. Während der Laufzeit wird keine Interaktion mit dem Workflow erlaubt. Für die Parameterstudien und Task-Farming sowie für Legacy Codes bietet Pegasus nur eine beschränkte Unterstützung an.

Der Fehlertoleranzmechanismus basiert auf Condor DAGMan, das heißt, die Fehlertoleranz wird auf Task- und Workflow-Ebene unterstützt. Für die Condor Standard-Jobs ist das Checkpointing Feature eingebaut, aber für die normalen Jobs, die Checkpoint nicht unterstützen, kann Condor dann kein Checkpointing machen. Auf der Task-Ebene wird auch die Jobmigration und Jobwiederholung im Fall der Fehler ausgeführt. Bezüglich der Workflow-Ebene-Fehlertoleranz laüft der Rest des Workflow weiter bis kein weiterer Fortschritt gemacht werden kann. Der Rettungs-DAG zeigt die unfertigen Teile der DAG samt Fehlermeldungen an. Der Benutzer kann dann die Fehler korrigieren und den Rettungs-DAG wieder ausführen lassen.

Pegasus Vorteile:

- Das System ist sehr gut für datenintensive Anwendungen geeignet.
- Intelligenter Planungsmechanismus zur Generierung der Workflows.
- Effizienz.

Pegasus Nachteile:

- GUI hat nur beschränkte Möglichkeiten.
- Wenig Dokumentation.
- Die Parameterstudiummöglichkeiten sind beschränkt.

4.9 K-Wf Grid GWES[10]

Das von Fraunhofer FIRST entwickelte Grid-Workflow-Management-System GWES (Grid Workflow Execution Service) ermöglicht das Management und die Automatisierung von komplexen Prozessabläufen in Grid-Umgebungen. Das System wird im KWFGrid, InstantGrid und MediGrid [23] eingesetzt, vormals im Fraunhofer Ressource Grid.

Das GWES Sysem besteht aus einer Workflow-Engine (GWES, Grid Workflow Execution Service) und einer GUI (GWUI, Grid Workflow User Interface) [24]. Das GWES System ist ein Open Source System mit der freien Lizenz nur für nichtkommerziellen Einsatz, der Worflow Editor des WMS ist aber noch nicht Open Source.

Das System basiert auf Petrinetzen, bei denen Aktivitäten mit Transitionen und Daten als Token in Stellen modelliert werden. Das WMS unterstützt reine Web-Services sowie Globus Toolkit 4, es kann einfach auch für die weiteren Plattformen, wie UNICORE, Condor, GRIA, und SUN Grid Engine erweitert werden. Dabei können die existierenden Workflows auf verschiedenen Grid-Middleware-Systemen wiederverwendet werden.

Es gibt eine klare Trennung zwischen der Workflow-Engine (GWES) und dem AJAX-basierten graphischen Client (GWUI). Die Engine läuft als Web-Service, der Workflows in der Open Source XML-Datenbank eXist speichert. Diese führt auch die automatische Abbildung von abstrakten Workflows auf jeweils geeignete und verfügbare Ressourcen aus. Das GWUI ist ein Applet, das im Browser (direkt oder als GridSphere-Portlet) läuft. Es ist eine interaktive graphische Benutzeroberfläche für den GWES. Es bietet die Funktionen für Starten, Monitoring und Manipulieren der Workflows, sowie die Mechanismen für die Benutzerinteraktion wie Spezifizieren der Eingangsdaten oder Entscheidungen an. Die Workflowerstellung ist mit der GUI aber leider nicht möglich, man muss dann selbst seinen Workflow zuerst beschreiben. Für die Workflowbeschreibung wird eine proprietäre XML Repräsentation für Petrinetze benutzt, die sich GWorkflowDL-Sprache (Grid Workflow Description Language) nennt. Derzeit wird auch ein Workflow-Editor für die automatische Workflowerstellung entwickelt, er hat schon die meisten Funktionen, aber er ist leider instabil und generell nicht verfügbar. Es gibt neben dem GWUI auch einen Kommandozeilen-Client. Hierarchische Workflowerstellung wird noch nicht unterstützt, für die Parameterstudien gibt es zurzeit nur eingeschränkte Unterstützung (Liste). Das Kreuzprodukt oder das Skalarprodukt von mehreren Listen wird noch nicht angeboten. Bezüglich der Fehlertoleranz kann man die folgende Methode für seinen Workflow auswählen: AbortOnActivityTerminated, ContinueOnActivityTerminated und SuspendOnActivityTerminated. Ein Checkpoint für das Workflow kann erstellt werden.

GWES Vorteile:

- Das WMS ist kompatibel zu unterschiedlichen Arten von Grid-Middleware (z.B. Globus Toolkit, Condor, PBS, Web-Services) und Plattformen (Windows, Linux, Unix).
- Workflow-Checkpointing ist möglich.

[10] Grid Workflow Execution Service, D-Grid-Projekt, Laufzeit 10/2005 – 05/2009, http://www.gridworkflow.org/kwfgrid/gwes/docs/

- Web-Services, sowie WS-GRAM Jobs werden unterstützt.

GWES Nachteile:

- Workflow-Editor ist nicht Open Source und generell noch nicht verfügbar.
- Hierarchische Workflows werden noch nicht unterstützt.
- Für die Parameterstudien gibt es zurzeit nur eine sehr eingeschränkte Unterstützung.

4.10 Karajan[11]

Java CoG Karajan [25] ist ein Open Source integriertes Workflow-Tool. Eine Graphische Komposition des Workflows ist nicht möglich – die Autoren bevorzugen das direkte Editieren von XML Code, mit dem man auch komplizierte Workflows mit den Programmierkonstrukten wie Schleifen (do, while) und Logik (if then) beschreiben kann. Das heißt, die Karajan GUI erlaubt nicht die Erstellung, sondern nur das Herunterladen eines existierenden Workflows, die Visualisierung, Starten und Überwachung eines Workflows werden aber in der GUI gemacht. Die GUI unterstützt nicht die Workflowausführung in Batch-Mode, aber es ist aus der Kommandozeile möglich. Als Teil des Java CoG Kits wird Globus unterstützt. Workflow-Engine basiert auf Globus Toolkit. Das Abstraktion Interface JobSpecification geht von Executables und nicht Web-Services aus. Ein einfaches Checkpointing wurde auf Workflow-Ebene implementiert. Ein Workflow-Checkpoint kann erstellt werden, soweit alle Elemente sich in einem konsistenten Zustand befinden. Karajan bietet auch die erweiterten Datenstrukturen wie Listen, Bereiche, Hashmaps für die Parameterstudien an. Hierarchische Workflowerstellung wird unterstützt.

Karajan Vorteile:

- Das Tool ist relativ klein und kompakt.

Karajan Nachteile:

- Keine GUI für die Workflowerstellung.
- WS werden nicht unterstützt.

4.11 g-Eclipse[12]

g-Eclipse [26] ist ein Open-Source-Framework und ein integriertes Werkzeug, welches Anwender, Administratoren und Entwickler bei der Arbeit mit Computing Grids unterstützt. Es basiert auf der offenen Eclipse-Plattform und hat auch eine Workflowkomponente, mit der man einfache Workflows erstellen, editieren und für die Ausführung einreichen kann. Diese Workflowkomponente ist zunächst abstrakt und besteht aus einem Workflow-Editor mit Jobs, In- und Output Ports und Links, der eine Workflowbeschreibung im XMI-Format erstellt. Man muss nur vordefinierte JSDL-Dateien (Job Submission Description

[11] Argonne National Laboratory, http://wiki.cogkit.org
[12] g-Eclipse Consortium, http://www.geclipse.org, aktuelle Version 1.0.0, von 21.01.2009

Language, OGF Standard für Jobbeschreibung) in den Workflow-Editor per Drag&Drop reinziehen und die Ein- und Ausgabeports verbinden.

Eigene Workflow-Engine ist nicht vorgesehen, stattdessen wurde die Unterstützung von mehreren Workflow-Sprachen (d.h. Übersetzung des g-Eclipse spezifischen Format XMI auf eine oder andere Workflow-Beschreibungssprache) und folglich Workflow-Engines mittels der "Export to" Funktionalität geplant. Die Ausführung der Workflows wird dann mittels eines anderen externen Workflow-Management-Systems gemacht. Das heißt, die Abbildung des Workflows auf eine konkrete Grid-Umgebung geschieht dann über einen entsprechenden Konverter. Im Augenblick sind diese für gLite gemacht, später GRIA und Globus. (In gLite gibt es ein Grid-weites Workload-Management-System, an das DAG-basierte in JDL beschriebene Jobs übermittelt werden können; einzelne Workflow-Engines a la Taverna sind nicht vorgesehen). Die GRIA-Middleware unterstützt die Ausführung der in XSCUFL beschriebenen Workflows, deshalb wurde Programmierung der Konversion in Taverna-Sprache XSCUFL auch schon gemacht. Programmierung der Konversion in andere konkrete Beschreibungssprachen muss noch vorgenommen werden, mitsamt der Auswahl einer konkreten Engine.

g-Eclipse Vorteile:

- Lässt sich mit weiterer Funktionalität ergänzen.
- Die populären Workflow-Sprachen und Systeme kann man als g-Eclipse Plug-ins einbinden.

g-Eclipse Nachteile:

- Momentan noch in der Entwicklung, daher instabil.

4.12 UNICORE[13]

Die Version 6.1 von UNICORE beinhaltet die erste Version der Workflow-Engine. Das UNICORE Workflow-System ist ein Open Source System (BSD Lizense), welches das Erstellen und die Ausführung von komplexen Workflows auf unterschiedlichen Systemen ermöglicht [27]. Es ist in Java geschrieben und deshalb portabel. Das UNICORE Workflow-System modelliert nicht abstrakte sondern konkrete Workflows und besteht aus einer graphischen, eclipse-basierten GUI, die sich UNICORE Rich Client nennt, einer Workflow-Engine, einem Service-Orchestrator und einem Tracing-Service.

Mit der GUI erzeugt man DAG-basierte Workflows aus den verschiedenen Diensten, die normalerweise unterschiedliche Anwendungen inklusive auch notwendiger Datenerzeugungselementen aufrufen. Programmierkonstrukte wie Schleifen und Logik (if-then-else) werden dabei unterstützt. Die Workflow Engine basiert auf Shark Open-Source XPDL Engine. Die Workflow Engine nimmt DAG-basierte Workflowbeschreibungen auf, und erstellt daraus die entsprechenden Schritte. Anderseits übernimmt der Service-Orchestrator die Job Submission und Beobachtung der einzelnen Schritte des Workflows. Auch wenn der Ausführungsort manuell ausgewählt werden kann, unterstützt der Orchestrator

[13] UNiform Interface to COmputer REsurces, Collaboration between German research institutions and industies, http://www.unicore.org

eine automatische Vermittlung, die auf austauschbaren Strategien basiert. Dazu unterstützt der Orchestrator auch die Callbacks an die Workflow-Engine. Das Tracing-Service erlaubt, jeden Workflowausführungsschritt zu beobachten, was man leicht im Client verfolgen kann.

Workflowausführung im Batch-Mode wird unterstützt, außerdem ist die Workflowsubmission aus der Kommandozeile auch möglich. Hierarchische Workflows, sowie Interoperabilität mit den anderen Workflow-Management-Systemen werden nicht unterstützt. Mit dem UNICORE WMS kann man auch die einfache Parameterstudien ausführen, aber das automatische Kreuzprodukt oder das Skalarprodukt von mehreren Listen ist noch nicht möglich. Der Fehlertoleranzmechanismus basiert auf UNICORE-Middleware.

UNICORE Vorteile:

- Job- und Workflowerstellung mit Hilfe der GUI.
- Erweiterbarkeit des Clients durch anwendungsspezifische Plugins.

UNICORE Nachteile:

- Hierarchische Workflows werden nicht unterstützt.
- Interoperabilität mit den anderen WMS wird nicht unterstützt.

5 Fazit der Evaluation

Die Tabellen 1 und 2 zeigen eine kurze Übersicht von Workflow-Management-Systemen entsprechend der im Abschnitt 3 gennanten Kriterien.

Den Tabellen lässt sich entnehmen, dass es kein vollendetes Workflow-Management-System gibt, das alle obengenannten Kriterien erfüllen würde. Einige Systeme sind nicht Open Source (WS-PGrade, ASKALON), einige sind zu kompliziert (ASKALON) und von vielen verschiedenen Bibliotheken abhängig (Taverna, Triana). In den anderen Systemen ist die Workflowausführung auf dem Grid noch nicht transparent, und der Benutzer braucht dafür Grid-Kenntnisse (Kepler), einige Systeme haben noch verschiedene Probleme mit der GUI (ICENI, Pegasus, Karajan) oder unterstützen GT4 nicht (gEclipse).

Von den Merkmalen her, die wir für das GridSFEA-Framework benötigen, sind WS-VLAM und GWES am ehesten geeignet. Hauptvorteil bei WS-VLAM ist, dass die Workflow-Engine als ein GT4 WSRF kompatibler Dienst implementiert wurde, genauso wie die GridSFEA-Dienste implementiert wurden, während die GWES-Engine als Web-Service realisiert wurde. Vorteile des GWES sind die Integration in das GridSphere-Portal (GridSFEA-Portal basiert übrigens auch auf GridSphere-Portal) samt graphischem Client, die schon weit vorangekommen ist, und die konzeptionelle Einfacheit des Formalismus (Petrinetze) sowie die daraus resultierende intuitive Bedienbarkeit des Clients. Für beide Systeme gilt, dass sie relativ klein und kompakt sind, es gibt eine Trennung zwischen Engine und GUI, und Web-Services werden unterstützt. Dabei unterstützt WS-VLAM auch die hierarchischen Workflows, was das GWES System noch nicht

	Open Source	Universalität	Middleware	GUI	Legacy Code	Komponenten-bibliothek	Komponenten-entwicklung	Parameter Studium
Taverna	ja	nein	-	gut	mit SOAPLAB	umfangreich	N/A	Liste
Triana	ja	ja	GT4 durch Java-GAT	gut	ja	umfangreich	Java	Liste
WS-PGrade	nein	ja	GT2/4, gLite LCG2	gut	mit GEMLCA	-	-	Liste,Bereich, Random, aus Eingangs-datei
WS-VLAM	ja	ja	GT2, GT4	gut	ja	umfangreich	Java, Python C++	Liste, Bereich
Kepler	ja	ja	GT2	gut	ja	sehr umfangreich	Java	via Nimrod
ASKALON	nein	ja	GT2, GT4	gut	muss als Service adaptiert wer-den	-	-	beliebig kom-pliziertes, mittels ZEN-Direktiven
ICENI	ja	ja	Condor, GT2, SGE	beschränkt	ja	nicht umfangreich	Java	Liste
Pegasus	ja	ja	Condor, GT2/3/4	beschränkt	beschränkt	nicht umfangreich	N/A	beschränkt
GWES	ja,aber nicht GUI	ja	GT4, pure Web-Services	beschränkt	ja	kein	kein	beschränkt
Karajan	ja	ja	GT2, GT4	beschränkt	ja	kein	kein	Liste, Bereich, Hashmap
g-Eclipse	ja	ja	gLite, GRIA, GT2	gut	ja	JSDL-Dateien	JSDL-Dateien	nein
UNICORE	ja	ja	UNICORE	gut	ja	JSDL-Dateien	JSDL-Dateien	Liste, beschränkt

Tabelle 1: WMS Übersicht

	Unterstützung der großen Workflows Batch-Mode/hierarchische W./parallele Jobs			Fehlertoleranz	Workflow-überwachung	Dokumentation Benutz./Entw.	Interoperabilität
Taverna	Plug-in TRES	ja	ja	Task-Ebene	schrittweise (Plug-in WS)	gut/nicht genügend	N/A
Triana	ja	ja	ja	basiert auf GAT-Manager	schrittweise	gut/nicht genügend	Pegasus
WS-PGrade	N/A	ja	ja	Task-Ebene	schrittweise	gut/kein	Kepler Taverna Triana
WS-VLAM	ja	ja	ja	in Entwicklung	schrittweise	gut/ gut	Kepler Taverna
Kepler	ja	ja	ja	Task- und Workflow-Ebene	beschränkt, unter Entwicklung	gut/ nicht genügend	Pegasus geplannt
ASKALON	N/A	ja	ja	Task- und Workflow-Ebene	schrittweise	ausreichend/-	N/A
ICENI	nein	nein	ja	basiert auf Middleware	beschränkt	ausreichend/ausreichend	N/A
Pegasus	ja	ja	ja	Task- und Workflow-Ebene	schrittweise	ausreichend/nicht genügend	Kepler Triana geplannt
GWES	ja, aber nicht vom GUI	nein	ja	Workflow-Ebene	schrittweise?	ausreichend/ gut	Taverna
Karajan	ja, aber nicht vom GUI	ja	ja	Task- und Workflow-Ebene	schrittweise	gut / nicht genügend	N/A
gEclipse	N/A	nein	ja	N/A	schrittweise	ausreichend/ausreichend	Taverna
UNICORE	ja	nein	ja	basiert auf Middleware	schrittweise (Tracing-Service)	gut/ gut	nein

Tabelle 2: WMS Übersicht

kann, außerdem ist die Parameterstudiumunterstützung in WS-VLAM deutlich besser. Was dem WS-VLAM System noch fehlt und was im Gegensatz im GWES System sehr gut entwickelt wurde, ist das Process-/ Workflow-Checkpointing. Aber dieses Problem könnte durch die Integrierung von GridSFEA Checkpoint-Migration-Tool und WS-VLAM System gelöst werden. Deshalb wurde WS-VLAM System für die Integration mit dem GridSFEA-System gewählt.

Danksagung

Wir danken der Bayerischen Forschungsstiftung für die finanzielle Unterstützung dieser Arbeit.

Literatur

[1] Muntean, I.L. ; Elts, E. ; Buchholz, M. ; Bungartz, H.-J.: Grid-supported simulation of vapour-liquid equilibria with GridSFEA. In: *Proceedings of 8th International Conference Computational Science - ICCS 2008* Bd. Part I. Krakow, Poland : Springer-Verlag, June 23-25 2008, S. 45–55

[2] Zhao, Yong ; Raicu, I. ; Foster, I.: Scientific Workflow Systems for 21st Century, New Bottle or New Wine? In: *Proc. IEEE Congress on Services - Part I SERVICES '08*, 2008, S. 467–471

[3] Muntean, I. L.: *Efficient Distributed Numerical Simulation on the Grid*, Institut für Informatik, Technische Universität München, Verlag Dr. Hut, München, ISBN 978-3-89963-870-7, Diss., October 2008

[4] Barker, Adam ; Hemert, Jano van: Scientific Workflow: A Survey and Research Directions. In: *Lecture Notes in Computer Science* 4967 (2008), S. 746–753. – Springer

[5] Gudenkauf, Stefan: *Bericht zum Grid Workflow Workshop - Oldenburg 2008.* OFFIS - Institut für Informatik. `http://pi.informatik.uni-siegen.de/stt/28_2/08_Sonstiges/2008_GWW_report.pdf`

[6] Aschenbrenner, Andreas ; Gietz, Peter ; Haase, Martin ; Knoll, Frank ; Küster, Marc W. ; Ludwig, Christoph: *Bericht über eine Evaluation von Grid Middleware Standards und von Grid Software Paketen.* Version: August 2006. `http://www.textgrid.de/fileadmin/TextGrid/reports/TextGrid_Report_3_1.pdf` TextGrid Report 3.1

[7] Kacsuk, P. ; Karoczkai, K. ; Hermann, G. ; Sipos, G. ; Kovacs, J.: WS-PGRADE: Supporting parameter sweep applications in workflows. In: *Proc. Third Workshop on Workflows in Support of Large-Scale Science WORKS 2008*, 2008, S. 1–10

[8] Gil, Y. ; Deelman, E. ; Ellisman, M. ; Fahringer, T. ; Fox, G. ; Gannon, D. ; Goble, C. ; Livny, M. ; Moreau, L. ; Myers, J.: Examining the Challenges of Scientific Workflows. In: *IEEE Computer* 40 (2007),

Dec., Nr. 12, S. 24–32. http://dx.doi.org/10.1109/MC.2007.421. – DOI 10.1109/MC.2007.421

[9] Oinn, T. ; Greenwood, M. ; Addis, M. J. ; Alpdemir, M. N. ; Ferris, J. ; Glover, K. ; Goble, C. ; Goderis, A. ; Hull, D. ; Marvin, D. J. ; Li, P. ; Lord, P. ; Pocock, M. R. ; Senger, M. ; Stevens, R. ; Wipat, A. ; Wroe, C.: Taverna: lessons in creating a workflow environment for the life sciences: Research Articles. In: *Concurrency and Computation: Practice & Experience* 18 (2006), August, Nr. Issue 10, S. 1067 – 1100. – Workflow in Grid Systems, ISSN:1532-0626

[10] Senger, Martin ; Rice, Peter ; Oinn, Tom: SOAPLAB - a unified Sesame door to analysis tools. In: Cox, Simon J. (Hrsg.): *UK e-Science All Hands Meeting*, 2003, S. 509–513. – ISBN - 1-904425-11-9

[11] Yu, J. ; Buyya, R.: A taxonomy of scientific workflow systems for grid computing. In: *Journal of Grid Computing* 3(3) (2005), S. 171–200

[12] Kacsuk, Peter ; Farkas, Zoltan ; Hermann, Gabor: Workflow-Level Parameter Study Support for Production Grids. In: *Computational Science and Its Applications - ICCSA 2007* Bd. 4707/2007. Kuala Lumpur, Malaysia : Springer Berlin / Heidelberg, August 26-29 2007, S. 872–885

[13] http://www.gridsphere.org

[14] Delaitre, Thierry ; Kiss, Tamas ; Goyeneche, Ariel ; Terstyanszky, Gabor ; Winter, Stephen ; Kacsuk, Peter K.: GEMLCA: running legacy code applications as grid services. In: *Journal of Grid Computing* 3 (1-2) (2005), S. 75–90

[15] Korkhov, Vladimir ; Vasyunin, Dmitry ; Wibisono, Adianto ; Belloum, Adam S. ; Inda, Márcia A. ; Roos, Marco ; Breit, Timo M. ; Hertzberger, L.O.: VLAM-G: Interactive data driven workflow engine for Grid-enabled resources. In: *Scientific Programming* 15 (2007), Nr. 3, S. 173–188. – IOS Press - publisher ISSN 1058-9244 (Print) 1875-919X (Online)

[16] Wibisono, Adianto ; Vasyunin, Dmitry ; Korkhov, Vladimir ; Zhao, Zhiming ; Belloum, Adam ; Laat, Cees de ; Adriaans, Pieter ; Hertzberger, Bob: WS-VLAM: A GT4 Based Workflow Management System. In: *Computational Science - ICCS 2007* Bd. 4489. Beijing, China : Springer Berlin / Heidelberg, May 27 - 30 2007, S. 191–198. – ISBN978-3-540-72587-9

[17] Korkhov, V. ; Vasunin, D. ; Wibisono, A. ; Guevara-Masis, V. ; A.Belloum ; Laat, C. de ; Adriaans, P. ; L.O.Hertzberger: WS-VLAM: Towards a scalable workflow system on the Grid. In: *16th IEEE International Symposium on High Performance Distributed Computing*. Monterey Bay, California, USA, June 27-29 2007, S. 63–68

[18] Inda, M. A. ; Belloum, A. S. Z. ; Roos, M. ; Vasunin, D. ; Laat, C. de ; Hertzberger, L. O. ; Breit, T. M.: Interactive Workflows in a Virtual Laboratory for e-Bioscience: The SigWin-Detector Tool for Gene

Expression Analysis. In: *Proc. Second IEEE International Conference on e-Science and Grid Computing e-Science '06*, 2006, S. 19–26

[19] Zudilova-Seinstra, Elena V. ; Yang, Ning ; Axner, Lilit ; Wibisono, Adianto ; Vasunin, Dmitry: Service-oriented visualization applied to medical data analysis. In: *Service Oriented Computing and Applications* 2 (2008), Nr. 4, S. 187–201

[20] `http://ptolemy.eecs.berkeley.edu/ptolemyII`

[21] `http://messagelab.monash.edu.au/Nimrod`

[22] Gulamali, M. ; McGough, S. ; Newhouse, S. ; Darlington, J.: Using ICENI to run Parameter sweep Applications across Multiple Grid Resources. In: *Global Grid Forum 10, Case Studies on Grid Applications Workshop*. Berlin, Germany, March 2004

[23] Krefting, Dagmar ; Vossberg, Michal ; Tolxdorff, Thomas: Simplified Grid Implementation of Medical Image Processing Algorithms using a Workflow Management System. In: Olabarriaga (Hrsg.) ; Lingrand (Hrsg.) ; Montagnat (Hrsg.): *MICCAI-Grid Workshop*. New York, September 6 2008

[24] Hoheisel, Andreas: Grid Workflow Execution Service - Dynamic and Interactive Execution and Visualiyation of Distributed Workflows. In: *K-WfGrid - The Knowledge-based Workflow System for Grid Applications, Proceedings of the Cracow Grid Workshop* Bd. II, 2006, S. 13–24. – ISBN: 978-83-915141-8-4

[25] Laszewski, Gregor ; Hategan, Mike: Workflow Concepts of the Java CoG Kit. In: *Journal of Grid Computing* 3 (2005), Nr. 3-4, S. 239–258

[26] *g-Eclipse Technical Documentation for the final g-Eclipse release.* `www.geclipse.org/fileadmin/Documents/Deliverables/D2.9_D3.8_D4.8-TechnicalDocumentation.pdf`. Version: 2009

[27] Becker, Daniel ; Riedel, Morris ; Streit, Achim ; Wolf, Felix: Grid-Based Workflow Management for Automatic Performance Analysis of Massively Parallel Applications. In: N. Meyer, R. Y. D. Talia T. D. Talia (Hrsg.): *Grid and Services Evolution, Proceedings of the 3rd CoreGRID Workshop on Grid Middleware*, Springer, 2009, S. 103 – 118

Glossar

Batch-Mode	eine sequentielle, nicht-interaktive Bearbeitung von Aufgaben, Stapelverarbeitung
BPEL	Business Process Execution Language
CoG	Commodity Grid
CSE	Computational Science Engineering
DAG	Directed Acyclic Graph

DAX	DAG in XML
EGEE	Enabling Grids for E-sciencE
GAT	Grid Application Toolkit
GEMLCA	Grid Execution Management for Legacy Code Applications
GRAM	Grid Resource Allocation and Management
Grid	eine Infrastruktur zur Integration bestehender, verteilter, möglicherweise heterogener Rechner-Ressourcen mit dem Ziel, eine möglichst verlässliche, konsistente, kostengünstige, einfache und transparente Nutzung von diesen Ressourcen zu ermöglichen.
GT4(2)	Globus Toolkit 4(2)
GUI	Grafical User Interface, engl. für Graphishe Benutzeroberfläche/Schnittstelle
JSDL	Job Submission Description Language
MIR	MyGrid Information Repository
OGF	Open Grid Forum
OGSA	Open Grid Service Architecture
PBS	Portable Batch System
RSL	Resource Specification Language
SCUFL	Simplified Conceptual Unified Flow
SGE	Sun Grid Engine
SOA	Service-orientierte Architektur
SOAP	Service-orientierte Architektur Protokoll
TRES	Taverna Remote Execution Service
VDL	Virtual Data Language
VDS	Virtual Data System
WMS	Workflow Management System
Workflow	eine Folge von Einzeltätigkeiten (Aktivitäten, Modulen), die schrittweise ausgeführt werden, um ein Ziel zu erreichen.
WS	Web Service
WSRF	Web Service Resource Framework
XML	Extensible Markup Language, engl. für erweiterbare Auszeichnungssprache