AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I ELEKTRONIKI
KATEDRA INFORMATYKI

Bartosz Baliś

Monitoring of Grid Scientific Workflows
Monitorowanie gridowych aplikacji naukowych sterowanych przepływem pracy

Autoreferat rozprawy doktorskiej

Promotor: Prof. dr hab. inż. Jacek Kitowski

Kraków 2009

1. Wstęp

Obliczenia wykonywane przy pomocy komputerów są obecnie niezbędnym elementem umożliwiającym przełomowe odkrycia w wielu dziedzinach nauki. Często badania wymagają nie tylko ogromnej mocy obliczeniowej, ale także połączenia wyników obliczeń z wielu różnych dziedzin, dostępu do danych zgromadzonych w różnych ośrodkach badawczych i współpracy naukowców z wielu instytucji i z różnych dziedzin nauki, stąd mówi się o *e-Science* lub *system-level science* [Foster2006]. Jako przykłady tego typu badań można wymienić przetwarzanie danych z Wielkiego Zderzacza Hadronów, wykrywanie nowych leków lub przewidywanie zmian klimatu. Aby takie badania były możliwe, konieczna jest *cyberinfrastruktura*, która zapewnia dostęp do zasobów obliczeniowych, zbiorów danych oraz dostarcza środowisk, w których naukowcy mogą wykonywać eksperymenty *in silico*. W ostatnich latach prowadzi się wiele projektów, których celem jest utworzenie takich infrastruktur na skalę narodową, regionalną i światową. W ogólnej wizji takiej cyberinfrastruktury, naukowiec mógłby w łatwy sposób definiować eksperyment, w którym specyfikowałby dane naukowe i transformacje na nich, które prowadziłyby do końcowego rezultatu eksperymentu. Zadaniem cyberinfrastruktury byłoby wyszukanie źródeł potrzebnych danych, być może integracja tych danych, oraz wyszukanie zasobów i uruchomienie usług, które wykonałyby potrzebne transformacje w sposób transparentny dla naukowca, który otrzymałby końcowe rezultaty obliczeń [Stein2008].

W tej wizji kilka technologii odgrywa szczególnie ważną rolę:

1. **Aplikacje naukowe typu workflow** (*scientific workflows*) [Gil2007]. Przy ich pomocy naukowiec definiuje eksperyment. Specyfikacja eksperymentu jako workflow jest zarazem używana przez usługi infrastruktury do wykonania go jako aplikacji naukowej na dostępnych zasobach.
2. **Technologie Gridowe** [Foster2003]. Dostarczają one ujednoliconej warstwy dostępu do rozproszonych zasobów obliczeniowych i źródeł danych.
3. **Technologie semantycznej sieci Web i semantycznego Gridu** (*Semantic Web* i *Semantic Grid*) [Hendler2004]. Technologie te wzbogacają istniejące usługi i dane o znaczenie semantyczne, które jest niezbędne, np. w celu osiągnięcia integracji heterogenicznych danych lub interoperabilności usług.

Wykorzystanie workflow do opisu eksperymentów *in silico* oraz ich uruchamiania daje wiele korzyści, wśród których najważniejsze to [Ludaescher2006] [Ludaescher2005]:

- Automatyzacja procesu analizy danych naukowych.
- Łatwość użycia, istotna dla naukowca niebędącego ekspertem w dziedzinie technologii informacyjnych.
- Możliwość zapisania i powtórzenia eksperymentu.
- Możliwość opublikowania przebiegu eksperymentu w publicznie dostępnej bazie, a tym samym publikowanie wiedzy i doświadczenia dostępnego dla środowiska naukowców.

Monitorowanie aplikacji polega na zbieraniu danych opisujących przebieg wykonania tych aplikacji. Zbieranie tych danych opiera się o *instrumentację* aplikacji, tj. dodanie dodatkowych instrukcji do kodu aplikacji, które powodują generowanie *zdarzeń* przenoszących podstawowe jednostki informacji. Monitorowanie aplikacji ma wiele zastosowań, np. pozwala ocenić i poprawić wydajność aplikacji, umożliwia wykrywanie błędów w trakcie działania aplikacji, dynamiczną rekonfigurację procesów aplikacji lub używanych przez nie zasobów, lub też

zapisywanie historii wykonania po to, by na jej podstawie przewidzieć lub poprawić wydajność następnych uruchomień. W przypadku Gridowych aplikacji naukowych typu workflow różnorodność scenariuszy, w których monitorowanie jest istotne, jest szczególnie duża. Obok typowych scenariuszy, takich jak pomiar wydajności czy wykrywanie błędów, można wymienić kilka scenariuszy swoistych:

- Środowiska Gridowe charakteryzują się zmienną dostępnością zasobów, dlatego optymalne ich wykorzystanie może wymagać dynamicznej rekonfiguracji. W tym celu koniecznie jest monitorowanie aplikacji i wykorzystywanych przez nią zasobów, aby podjąć ewentualne działanie korygujące.
- W wielu wypadkach przydatna jest historia wykonań eksperymentów, która może być wykorzystana do poprawienia wydajności następnych uruchomień.
- Naukowcy są zainteresowani nie tylko rezultatem eksperymentu, ale także jego przebiegiem. Tzw. dane o pochodzeniu (*provenance*) wyniku eksperymentu, które opisują proces, który prowadził do uzyskania wyniku, są niezwykle istotne dla badacza, gdyż pozwalają ocenić wiarygodność wyników. Aby dane o pochodzeniu wyników eksperymentu były dostępne dla badacza, konieczne jest monitorowanie aplikacji workflow, przy pomocy której eksperyment był realizowany.

Analizując te scenariusze, łatwo dostrzec ich różnorodność w kilku aspektach:

- Niektóre scenariusze można zrealizować w trybie monitorowania *off-line* (dane zbierane w trakcie monitorowania są analizowane po zakończeniu działania aplikacji), inne zaś wymagają monitorowania w trybie *on-line* (dane zbierane w trakcie monitorowania muszą być dostarczone do analizy w trakcie działania aplikacji).
- Konsumentami danych pochodzących z monitorowania są zarówno procesy, jak i ludzie.
- Pewne scenariusze wymagają historycznych zapisów eksperymentów, by na ich podstawie przeprowadzić swoistą ekstrakcję wiedzy, która umożliwi podjęcie decyzji.

Z punktu widzenia monitorowania ważna jest charakterystyka aplikacji typu workflow. Do niedawna obliczenia naukowe wykonywane były głównie jako ściśle powiązane (*tightly coupled*), homogeniczne aplikacje równoległe uruchamiane na klastrach. Aplikacje naukowe typu workflow jednakże posiadają całkiem odmienną charakterystykę – są one luźno powiązane (*loosely coupled*), heterogeniczne i rozproszone, a w dodatku najczęściej wykonywane w środowiskach Gridowych. Co więcej, aplikacje typu workflow często wykorzystują istniejące aplikacje „zastane" (*legacy applications*) do wykonania obliczeń pojedynczego zadania workflow. Jest tak dlatego, że zastane aplikacje naukowe (np. aplikacje równoległe napisane w MPI w języku Fotran lub C) są sprawdzone, a koszt ich ponownego napisania w nowoczesnych językach programowania i technologiach jest zbyt duży. W typowym przypadku, aplikacja zastana jest wykonywana w tle (*backend*) zadania workflow, w sposób przezroczysty dla użytkownika.

Podczas gdy ogólne problemy związane z monitorowaniem równoległych i rozproszonych aplikacji dużej skali są dość dobrze rozpoznane i rozwiązane, np. redukcja danych w celu obsługi ogromnej ilości generowanych zdarzeń (można wymienić chociażby GraDS / Autopilot [Vetter2007]), żadne z istniejących rozwiązań znanych autorowi nie podejmuje **problemów swoistych dla monitorowania Gridowych aplikacji naukowych typu workflow**. Te problemy wypływają z wszystkich trzech aspektów: aspektu aplikacji „Gridowej", aspektu aplikacji „naukowej", oraz aspektu aplikacji „typu workflow".

Po pierwsze, duża skala Gridu może się wiązać z dużym rozproszeniem obliczeń w ramach jednej aplikacji, a charakter zasobów Gridowych i sposobu dostępu do nich powoduje, że cykl życia aplikacji typu workflow jest bardzo złożony. Aplikacje typu workflow są zwykle specyfikowane w sposób abstrakcyjny, bez przyporządkowania zadaniom workflow konkretnych zasobów obliczeniowych. To przyporządkowanie, zwłaszcza w środowiskach Gridowych, odbywa się w sposób dynamiczny, przed wykonaniem, albo nawet w trakcie wykonania. W tym procesie uczestniczy wiele Gridowych usług *middleware*, np. do rozdzielania zasobów (*resource brokers*) lub szeregowania zadań (*schedulers*), które to usługi mogą być zdecentralizowane [Yu2007]. Konsekwencją tego jest to, że szczególnym wyzwaniem staje się *wykrywanie zasobów*, tj. wykrywanie wszystkich źródeł danych związanych z monitorowaniem pojedynczego workflow. Problem ten utrudnia zwłaszcza zapewnienie monitorowania w trybie *on-line*.

Po drugie, aplikacje naukowe typu workflow są heterogeniczne pod względem języków programowania użytych do implementacji poszczególnych zadań workflow, a także platform i zasobów, na których te zadania się wykonują. W konsekwencji może zaistnieć konieczność zastosowania wielu heterogenicznych narzędzi do instrumentacji i monitorowania pojedynczego workflow. Aby temu zaradzić, infrastruktura monitorowania musi zapewnić interoperabilność tych narzędzi. W przeciwnym wypadku, jednolite monitorowanie aplikacji typu workflow w ramach jednej infrastruktury nie będzie możliwe.

Po trzecie, zapis pochodzący z monitorowania naukowych aplikacji typu workflow jest istotny dla końcowego użytkownika, tj. naukowca, np. ze względu na dane o pochodzeniu. W konsekwencji, zwykłe zbieranie śladów zdarzeń, które jest wystarczające np. dla zastosowań związanych z wydajnością, nie wystarcza w przypadku naukowych workflow. Konieczny jest *model informacji*, która pozwoli w sposób ustrukturyzowany przechowywać zapis wykonania workflow. Co więcej, semantyka dziedzinowa aplikacji jest istotnym elementem takiego zapisu, gdyż pozwala na korzystanie z zapisanych rekordów posługując się językiem z dziedziny naukowca.

2. Cele, teza, wkład naukowy pracy i metody weryfikacji badań

W poprzednim rozdziale pokazano, że swoiste cechy Gridowych aplikacji naukowych typu workflow, oraz różnorodność scenariuszy, w których ich monitorowanie jest istotne, powodują, że monitorowanie Gridowych aplikacji naukowych typu workflow wiąże się z kilkoma swoistymi wyzwaniami, które muszą być rozwiązane. **Głównym celem badawczym niniejszej pracy jest identyfikacja i analiza tych wyzwań, zaproponowanie rozwiązań dla nich, oraz weryfikacja tych rozwiązań.**

W trakcie badań zidentyfikowano następujące główne problemy swoiste dla monitorowania Gridowych aplikacji naukowych typu workflow:

- **Budowa infrastruktury monitorowania** dla Gridowych aplikacji naukowych typu workflow.
- Wsparcie dla **monitorowania Gridowych aplikacji typu workflow w trybie *on-line***.
- **Monitorowanie aplikacji zastanych** wykonujących się w ramach workflow.
- **Utworzenie modelu informacji do zapisywania przebiegu eksperymentów** wykonywanych jako aplikacje workflow.

Główna teza pracy jest następująca:

> **Monitorowanie gridowych aplikacji naukowych typu workflow wymaga użycia odpowiedniej infrastruktury monitorującej i może być źródłem wartościowej informacji przydatnej do podjęcia celowego działania, jeśli dane opisujące przebieg działania aplikacji, pochodzące z jej monitorowania, są odpowiednio zorganizowane i wzbogacone o semantykę specyficzną dla dziedziny aplikacji.**

Główne osiągnięcie badawcze pracy to zrealizowanie postawionego celu – identyfikacji i analizy problemów związanych z monitorowaniem Gridowych aplikacji naukowych typu workflow, które są zasadnicze i swoiste dla tego typu aplikacji, zaproponowanie rozwiązań dla tych problemów, oraz weryfikacja tych rozwiązań. Ten ogólnie sformułowany wkład pracy może być szczegółowo opisany w następujących punktach:

1. Przeprowadzono szczegółową analizę wymagań dla infrastruktury monitorowania odpowiedniej dla Gridowych aplikacji naukowych typu workflow. Dzięki tej analizie, oraz studium istniejących pokrewnych rozwiązań, zidentyfikowane zostały braki w istniejących rozwiązaniach i zaproponowana została infrastruktura monitorowania dla Gridowych aplikacji workflow. Projekt tej infrastruktury realizuje następujące szczegółowe osiągnięcia:
   - Zaproponowany został projekt infrastruktury jako ramy programowej (*framework*), co pozwala ukryć złożoność i heterogeniczność aplikacji typu workflow.
   - Zaproponowano model zdarzeń do monitorowania wykonania aplikacji typu workflow.
   - Zaproponowano technologie *Complex Event Processing* (CEP) do zarządzania subskrypcjami w ramach infrastruktury monitorowania.
   - Zaadaptowano istniejące specyfikacje dla celów instrumentacji: *Workflow Instrumentation Request Language* (WIRL) jako baza interfejsu do instrumentacji, oraz *Standardized Intermediate Representation* (SIR) do reprezentacji regionów kodu workflow.
2. Zaproponowano rozwiązanie zapewniające wydajne i skalowalne wsparcie dla monitorowania aplikacji workflow w trybie on-line. Rozwiązanie to oparte jest o automatyczne wykrywanie zasobów, realizowane przy pomocy infrastruktury *Distributed Hash Table* (DHT) stowarzyszonej z usługami monitorowania. Przeprowadzono badanie wydajności i skalowalności tego rozwiązania.
3. Problem monitorowania aplikacji zastanych wykonujących się w ramach workflow został rozwiązany przez adaptację istniejącego podejścia do monitorowania OMIS.
4. Zaproponowano model informacji do zapisywania przebiegu wykonania aplikacji naukowych typu workflow. Wykorzystano technologie semantycznej sieci (*Semantic Web*) do opisu modelu i reprezentacji rekordów. Dowiedziono, że zaproponowany model informacji wzbogacony o semantyczną informację o aplikacji umożliwia końcowym użytkownikom (naukowcom) ekstrakcję wartościowej informacji (wiedzy).

Zaproponowane w pracy rozwiązania zweryfikowano przy użyciu różnych metod:

1. Stworzono prototyp zaproponowanej infrastruktury do monitorowania aplikacji Gridowych typu workflow „GEMINI”.
2. Zbudowano model systemu wykorzystując formalizm Sieci Kolejkowych i przy jego użyciu przeprowadzono analizę wydajności zaproponowanego rozwiązania do wydajnego i skalowalnego monitorowania w trybie on-line.
3. Stworzono prototyp systemu do monitorowania zastanych (*legacy*) aplikacji Gridowych – OCM-G i zintegrowano go z globalną infrastrukturą do monitorowania aplikacji typu

workflow. Zademonstrowano monitorowanie aplikacji MPI realizującej obliczenia w ramach workflow.

4. W celu weryfikacji koncepcji semantycznej informacji o eksperymencie pod kierunkiem autora stworzono dodatkowe narzędzie do wizualnego formułowania zapytań do ontologicznych baz danych – QUaTRO. Dowiedziono, że zaproponowany model informacji ułatwia formułowanie zapytań w sposób przyjazny dla końcowego użytkownika, umożliwia formułowanie interesujących zapytań, a także semantyczną integrację danych naukowych oraz metadanych, które opisują proces, w którym używano tych danych.
5. Stworzone prototypy systemów – GEMINI, OCM-G oraz QUaTRO – wdrożono do infrastruktury projektów EU-IST – K-Wf Grid i ViroLab, w których wykorzystano je do monitorowania rzeczywistych aplikacji typu workflow – do zarządzania ruchem w mieście (*Coordinated Traffic Management*) i badania oporności wirusa HIV na leki (*Drug Resistance*).

3. Infrastruktura do monitorowania aplikacji typu workflow

3.1. Wymagania

Infrastruktura monitorowania odpowiedzialna jest za kilka podstawowych zadań:

- Model i reprezentację danych pochodzących z monitorowania (zdarzeń).
- Udostępnianie usług do subskrypcji na zdarzenia i instrumentację.
- Zarządzanie subskrypcjami.
- Zarządzanie lokalnymi monitorami – sensorami (zbieranie zdarzeń) i mutatorami (manipulacja monitorowanymi zasobami).
- Zbieranie danych od producentów zdarzeń.
- Dostarczanie danych do konsumentów zdarzeń.

3.2. Architektura infrastruktury monitorowania

Rysunek 1 przedstawia logiczną architekturę zaproponowanej infrastruktury monitorowania o nazwie GEMINI (GEneric Monitoring INfrastructure). Podstawowe elementy tej architektury to: *Monitory* (rysunek przedstawia również wewnętrzną strukturę Monitora), *Sensory*, *Mutatory*, *Usługa Wykrywania Zasobów* (*Discovery Service*) oraz *Klienci*.

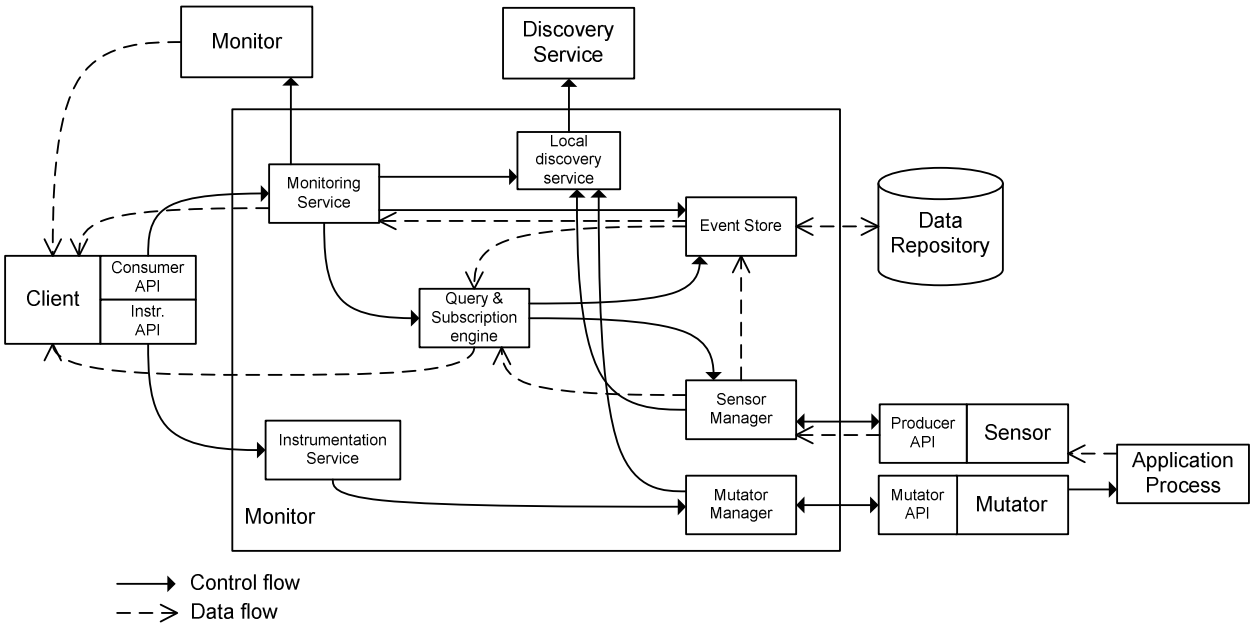
Klient używa usług udostępnianych przez Monitor – usługi do monitorowania (*Monitoring Service*) oraz usługi do instrumentacji (*Instrumentation Service*). Usługa do monitorowania umożliwia żądanie zdarzeń w jednym z dwóch trybów:

- Zapytanie (*query*) – zdarzenia pasujące do zapytania zwracane są w pojedynczej odpowiedzi.
- Subskrypcja (*subscribe*) – zdarzenia pasujące do zapytania zwracane są w serii odpowiedzi, które napływają asynchronicznie w nieznanej z góry liczbie i częstotliwości. Subskrypcja kończy się po zadanym z góry czasie lub przez jawne jej odwołanie.

Żądania zapytań lub subskrypcji są przekazywane do Silnika Zapytań i Subskrypcji (*Query and Subscription Engine*). W przypadku subskrypcji, w silniku tworzony jest nowy kontekst subskrypcji. W przypadku zapytania, zdarzenia mogą zostać pobrane z Magazynu Zdarzeń (*Event Store*) lub bezpośrednio poprzez zapytanie do Sensora. W ostatnim przypadku, silnik korzysta z Zarządcy Sensorów (*Sensor Manager*) aby sterować lub wysyłać żądania do sensora. Magazyn Zdarzeń pełni rolę pamięci podręcznej zdarzeń i przechowuje zdarzenia według pewnej

konfigurowalnej strategii (np. przez pewien czas, 100 ostatnich zdarzeń, itp.). Persystencję zdarzeń zapewnia Repozytorium Danych (*Data Repository*).

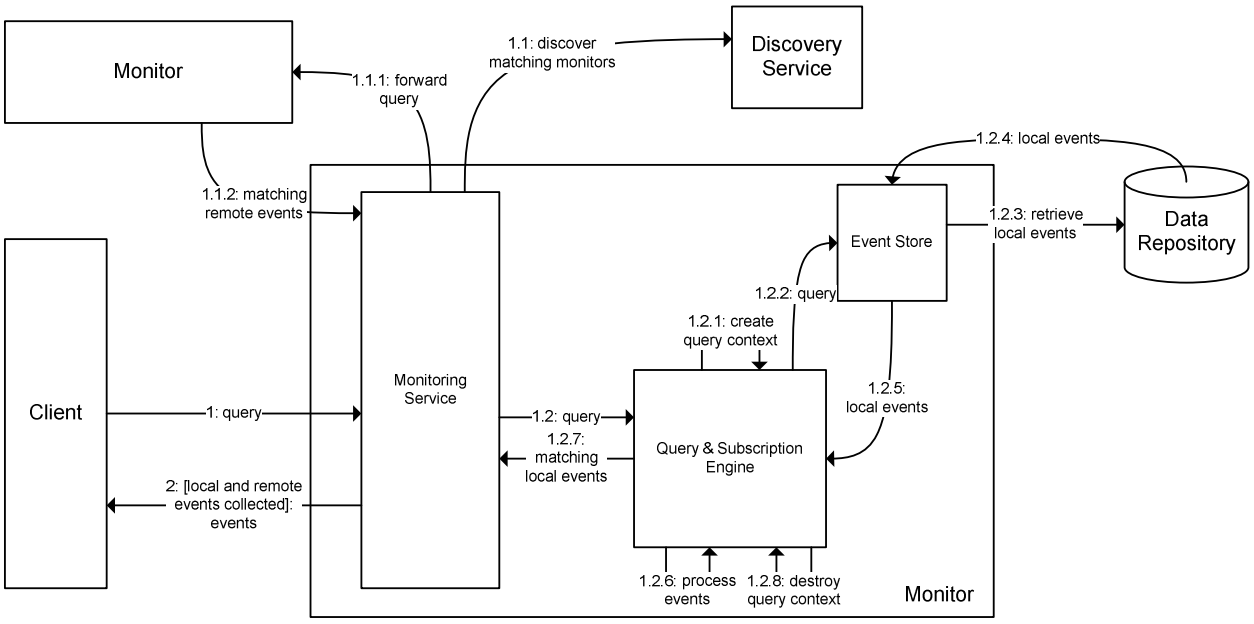
Usługa Wykrywania Zasobów (*Discovery Service*) jest zewnętrzną usługą, w której Monitory publikują informacje o monitorowanych przez siebie zasobach i dostarczanych typach zdarzeń. Inne usługi korzystają z usługi wykrywania by odszukać te Monitory, które dostarczają danych na temat zasobów będących przedmiotem zainteresowania. Z Usługi Wykrywania Zasobów korzysta m.in. Silnik Zapytań i Subskrypcji, który po otrzymaniu zapytania lub subskrypcji może odszukać odległe Monitory, które pasują do tych żądań.



Rysunek 1: Logiczna architektura infrastruktury do monitorowania GEMINI. Linie ciągłe symbolizują przepływ kontroli, linie przerywane – przepływ danych.

Rysunek 2 przedstawia scenariusz zapytania (*query*). Główne przepływy kontroli i danych w tym scenariuszu są następujące:

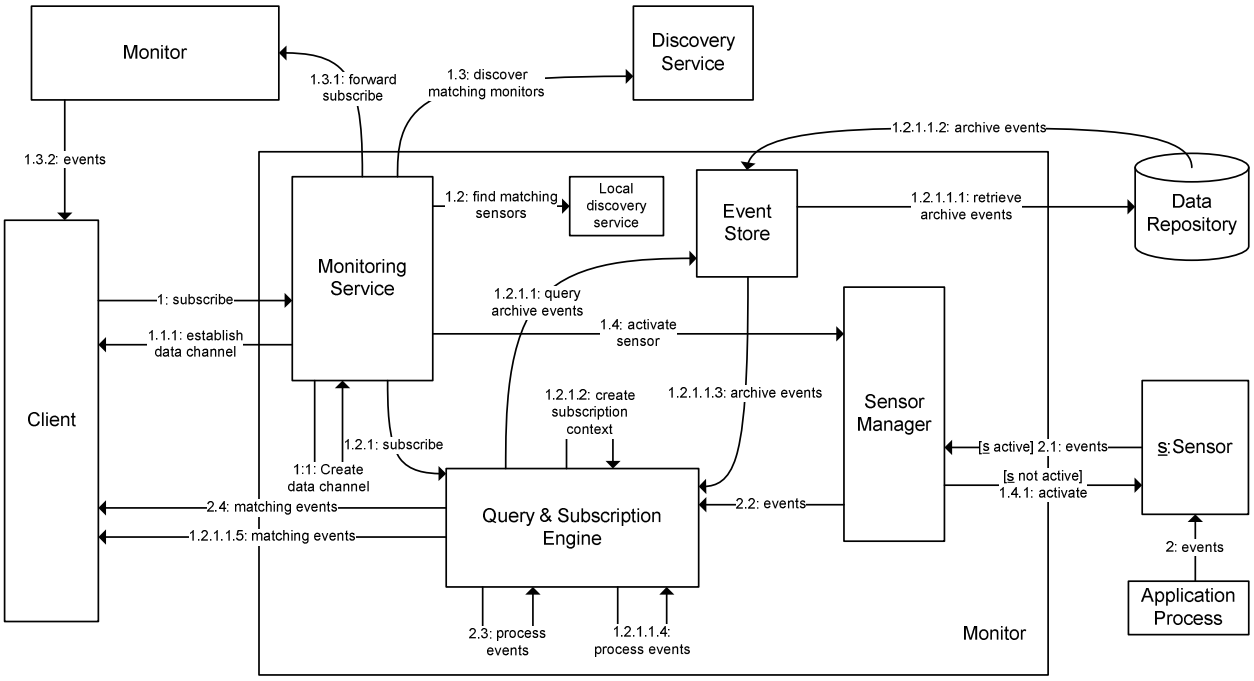
- **1.1.***: wykrycie odległych Monitorów, które pasują do zapytania, przekazanie im żądania i odbiór pasujących zdarzeń od odległych Monitorów.
- **1.2.***: odpytanie lokalnego magazynu zdarzeń o zdarzenia pasujące do zapytania.
- **2**: dostarczenie wszystkich pasujących do zapytania zdarzeń w pojedynczej odpowiedzi.



Rysunek 2: Scenariusz zapytania (query).

Rysunek 3 przedstawia scenariusz subskrypcji. Główne przepływy kontroli i danych w tym scenariuszu są następujące:

- **1.***: żądanie subskrypcji.
- **2.***: przepływ zdarzeń od zainstrumentowanej aplikacji do klienta.
- **1.1.***: ustanowienie kanału komunikacji pomiędzy Monitorem i klientem.
- **1.2.1.1.***: zapytanie o pasujące do żądania subskrypcji archiwalne zdarzenia, które zostały wygenerowane przed rozpoczęciem subskrypcji i przechowane w magazynie zdarzeń.
- **1.3.***: wykrywanie odległych monitorów, które pasują od żądania subskrypcji, przekazanie im tego żądania i transfer zdarzeń z odległych Monitorów do klienta.



Rysunek 3: Scenariusz subskrypcji.

3.3. Model zdarzeń

Standardowy model zdarzeń dla monitorowania umożliwia interoperabilność pomiędzy producentami i konsumentami informacji. Stanowi on wspólny *model danych wymienianych* (*data exchange model*) pomiędzy uczestnikami komunikacji. Nie narzuca on natomiast *modelu systemu* przyjętego wewnętrznie przez każdego uczestnika komunikacji. Zapewnia również luźne powiązanie uczestników komunikacji. Alternatywne rozwiązanie polegające na dostarczeniu złożonego interfejsu programistycznego (API) do publikowania i konsumowania zdarzeń, ukrywającego reprezentację przesyłanych danych – stosowane przez wiele infrastruktur monitorowania (np. R-GMA [Cooke2003] i Mercury [Podhorszki2004]) – nie zapewnia równie luźnego powiązania.

Istnieją nieliczne próby standaryzacji zdarzeń dla monitorowania w Gridzie, ale skupiają się one na monitorowaniu zasobów. Przykładem jest hierarchia zdarzeń „najwyższego poziomu" („Top N") zdefiniowana przez grupę roboczą Global Grid Forum DAMED [Gunter2003]. Widocznym brakiem tego rozwiązania jest brak zdarzeń dla *monitorowania wykonania aplikacji*.

W niniejszej pracy zaproponowano rozszerzenie taksonomii DAMED „Top N" o zdarzenia dla monitorowania wykonania aplikacji. Każde zdarzenie jest charakteryzowane przez:

- **Typ zdarzenia**: określa pomiar zawarty w zdarzeniu, np. 'delay.TCP'
- **Obiekt** (*target*): monitorowany zasób, którego dotyczy pomiar zawarty w zdarzeniu.
- **Czas**: znacznik czasowy oznaczający czas wystąpienia zdarzenia.
- **Dane zdarzenia**: wartość pomiaru zawartego w zdarzeniu. Dane zdarzenia określone są przez odpowiedni **typ danych**.

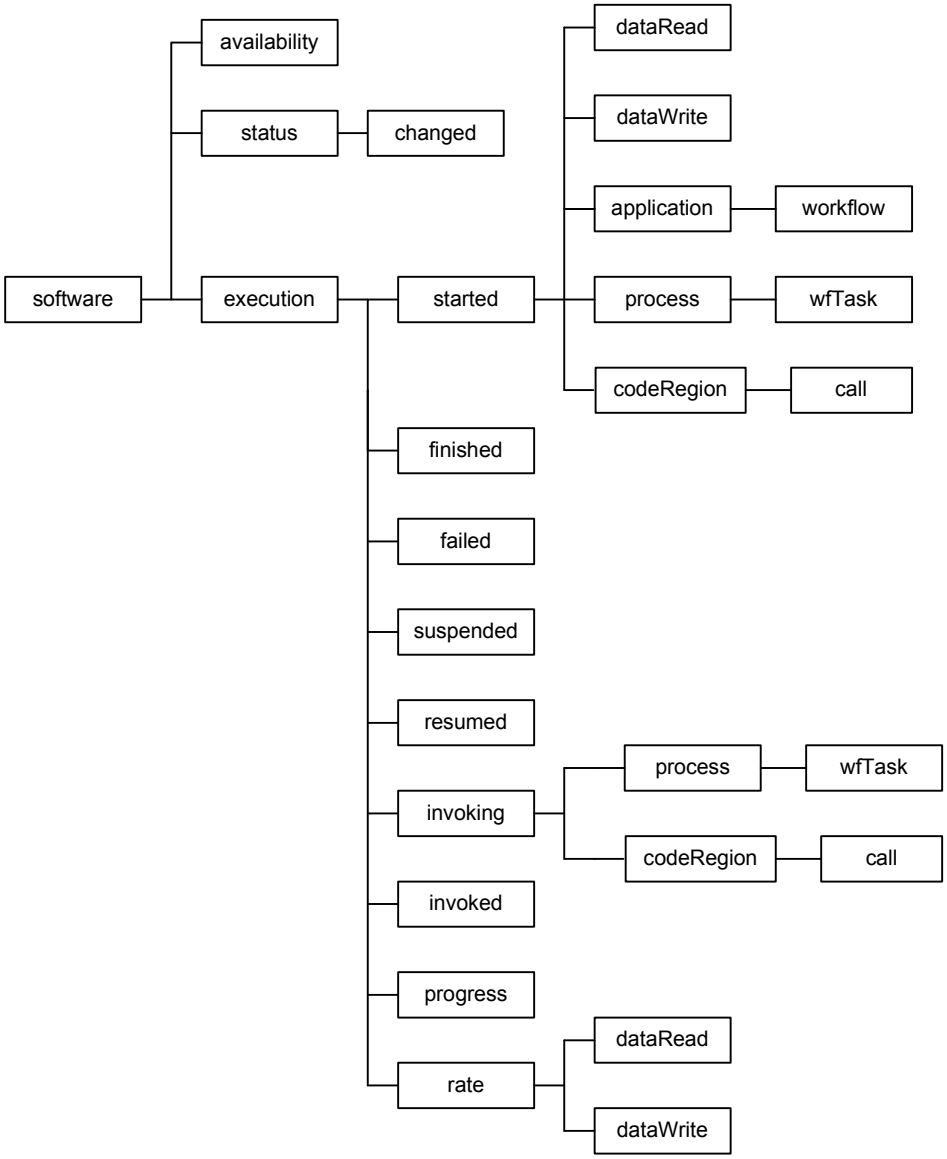
Rysunek 4 przedstawia bazową taksonomię zdarzeń dla monitorowania wykonania aplikacji typu workflow. Taksonomia ta ma budowę drzewa typów zdarzeń, gdzie bazowym zdarzeniem jest

‘software.execution’. Hierarchia została tak skonstruowana, że zdarzenie na każdym jej poziomie ma sens, a zdarzenia na wyższych poziomach są ogólniejsze od tych na niższych poziomach.

Następujące zdarzenia zostały zdefiniowane (w niektórych przypadkach pominięto pełne prefiksy nazw typów zdarzeń):

1. software.execution – wykonanie instrukcji lub regionu kodu w procesie.
2. software.execution.started – początek wykonania pewnej jednostki wykonawczej.
3. software.execution.finished – koniec wykonania pewnej jednostki wykonawczej.
4. execution.invoking – wywołanie pewnej jednostki wykonawczej.
5. execution.invoked – koniec wywołania pewnej jednostki wykonawczej.
6. execution.failure – błąd wykonania pewnej jednostki wykonawczej.
7. execution.progress – postęp wykonania pewnej jednostki wykonawczej (np. w procentach).
8. execution.rate – tempo wykonania pewnej jednostki wykonawczej (np. wykonań na jednostkę czasu, iteracji na sekundę, itp.)
9. execution.suspended – zawieszenie wykonania pewnej jednostki wykonawczej.
10. execution.resumed – wznowienie wykonania pewnej jednostki wykonawczej.
11. started.application – początek wykonania aplikacji.
12. invoking.process – inicjacja wykonania procesu (zdarzenie generowane przez inicjatora).
13. started.process – początek wykonania procesu (zdarzenie generowane przez sam proces).
14. started.application.workflow – początek wykonania aplikacji typu workflow.
15. started.codeRegion – początek wykonania region kodu w obrębie procesu.
16. started.codeRegion.call – początek wywołania procedury.
17. started.dataRead – początek operacji odczytu danych.
18. started.dataWrite – początek operacji zapisu danych.
19. rate.dataRead – przepustowość z jaką czytane są dane.
20. rate.dataWrite – przepustowość z jaką zapisywane są dane.

Lista ta nie jest wyczerpująca. Dla wielu zdefiniowanych zdarzeń ‘started’ konieczne są odpowiedniki ‘finished’ (np. wtedy, gdy pewne informacje dostępne są tylko po zakończeniu wykonania). Ponadto taksonomia ta jest z założenia rozszerzalna i można ją rozbudowywać o kolejne zdarzenia, np. ogólne ‘start aplikacji MPI’, ‘start pętli’, albo takie, które określają znaczenie pewnych operacji (specyficzne dla aplikacji).



Rysunek 4: Bazowa taksonomia zdarzeń dla monitorowania wykonania aplikacji typu workflow.

3.4. Instrumentacja

W przypadku aplikacji wielkiej skali, takich jak naukowe aplikacje Gridowe typu workflow, kontrola narzutu spowodowanego przez monitorowanie jest kwestią zasadniczą. Narzut związany z monitorowaniem wynika z generacji i transmisji zdarzeń i dotyczy wielu zasobów – procesora, pamięci i sieci. Podstawowym mechanizmem kontroli narzutu jest kontrolowana instrumentacja. Mechanizm instrumentacji aplikacji w Gridzie powinien posiadać trzy cechy: być dynamiczny, selektywny i cechować się regulowaną ziarnistością.

- **Dynamiczna** instrumentacja oznacza możliwość jej włączania i wyłączania w trakcie działania aplikacji.
- **Selektywna** instrumentacja oznacza, że można wybrać regiony kodu, które mają być zainstrumentowane – np. można zażądać instrumentacji konkretnego wywołania funkcji, a nie tylko wszystkich wywołań tej funkcji.

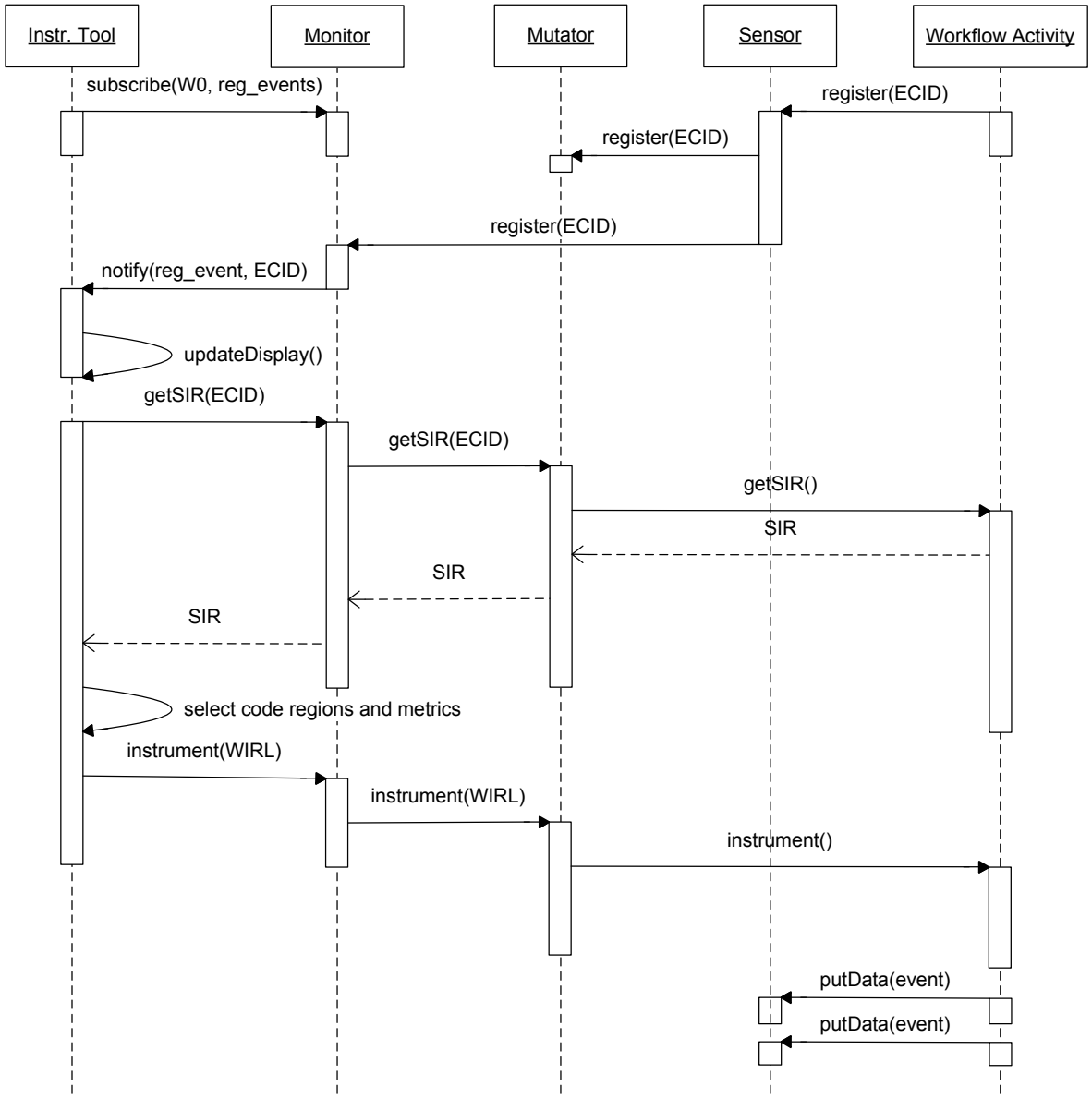
- **Regulowana ziarnistość** oznacza, że powinna istnieć możliwość włączania instrumentacji na różnych poziomach kodu – od poziomu workflow jako całości po poziom poszczególnych regionów kodu (wywołań funkcji, pętli, itp.).

Ponadto usługa instrumentacji w systemie monitorowania powinna umożliwić integrację heterogenicznych narzędzi i technologii, które mogą być użyte do instrumentacji różnych części pojedynczej aplikacji typu workflow. Po przeanalizowaniu istniejących rozwiązań w zakresie instrumentacji, podjęto następujące decyzje dotyczące instrumentacji w systemie GEMINI:

- Interfejs usługi instrumentacji został oparty o Workflow Instrumentation Request Language (WIRL) [Balis2006], język oparty o XML.
- Koncepcja standardowej pośredniej reprezentacji (Standardized Intermediate Representation, SIR) [Seragiotto2004] została zaadoptowana do reprezentacji regionów kodu, które mogą być instrumentowane. SIR jest językiem opartym o XML, który umożliwia abstrakcyjną, drobnoziarnistą i niezależną od języka programowania specyfikację struktury kodu aplikacji.
- Poszczególne technologie instrumentacji są adaptowane przy pomocy dedykowanych Mutatorów, które implementują standardowy interfejs instrumentacji oparty o WIRL.

Rysunek 5 przedstawia podstawowy scenariusz instrumentacji:

1. Narzędzie, które wykonuje instrumentację (*Instr.Tool*) subskrybuje się w monitorze na zdarzenia rejestracji nowych zadań workflow W0.
2. Równolegle tworzone jest nowe zadanie workflow W0 (*WorkflowActivity*), które rejestruje się w lokalnym Sensorze jako nowy zasób i przekazuje swój identyfikator (*Experiment Correlation Identifier*, ECID).
3. Sensor przekazuje zdarzenie rejestracji do Mutatora i Monitora, tak aby te komponenty były powiadomione o nowym zasobie.
4. Ponieważ Monitor posiada aktywną subskrypcję na właśnie otrzymane zdarzenie rejestracji, przekazuje to zdarzenie do narzędzia.
5. Narzędzie odświeża wygląd swojego interfejsu graficznego, tak aby odzwierciedlić informację o nowym zadaniu workflow W0. Następnie, narzędzie wysyła żądanie pobrania SIR (standardowej pośredniej reprezentacji) nowego zadania.
6. Żądanie pobrania SIR jest przekazywane do Mutatora (i być może do samego zadania, lecz jest to zależne od implementacji). W rezultacie narzędzie dostaje SIR jako odpowiedź na swoje żądanie.
7. SIR jest wyświetlany w narzędziu, co umożliwia selektywne wybranie regionów kodu, które mają być zainstrumentowane, a także określenie metryk, które mają być obliczane (tj. zdarzeń, które mają być tworzone) w wybranych miejscach. Po wybraniu regionów i metryk, generowane jest żądanie WIRL, które zawiera pełną specyfikację żądanej instrumentacji.
8. Żądanie WIRL jest przekazywane do Mutatora, który aktywuje instrumentację w *WorkflowActivity* (sposób, w jaki się to odbywa, jest zależny od implementacji).
9. Gdy wykona się instrukcja instrumentacji, generowane są zdarzenia i przekazywane do lokalnego Sensora.



Rysunek 5: Scenariusz instrumentacji.

4. Wsparcie dla monitorowania aplikacji workflow w trybie on-line

4.1. Określenie problemu

Gridowe aplikacje typu workflow są rozproszone, dynamiczne i charakteryzują się złożonym cyklem wykonania. Wykrywanie zasobów stanowi zasadnicze wyzwanie dla monitorowania aplikacji typu workflow, ponieważ poszczególne zadania workflow są tworzone dynamicznie na dynamicznie przydzielanych zasobach. Aby zapewnić wsparcie dla monitorowania w trybie *on-line*, tj. takie, w którym zdarzenia pochodzące z monitorowania aplikacji są przekazywane do subskrybentów w trakcie jej działania, konieczny jest mechanizm, który w niniejszej pracy nazwano *automatycznym wykrywaniem zasobów*. Automatyczne wykrywanie zasobów zapewnia, że konsumenci są automatycznie powiadamiani o nowo pojawiających się producentach, którzy

dostarczają danych pasujących do aktywnych subskrypcji konsumentów. Przy założeniu zdecentralizowanej architektury usług Gridowych, taki mechanizm musi się opierać o lokalne wykrywanie zasobów, następnie ich globalne rozgłaszanie, a w końcu globalne wykrywanie (*local discovery – global advertisement – global discovery*). Istniejące obecnie systemy Gridowych usług informacyjnych (*Grid information systems*), które w Gridzie używane są do wykrywania zasobów, takie jak R-GMA [Cooke2003], BDII [Astalos2008], lub MDS [Schopf2006], nie nadają się jednak do scenariuszy opartych o szybką notyfikację o zmianach stanu zasobów. Te systemy są zorientowane na wydajność zapytań i skalowalność do dużej liczby współbieżnych klientów. Istniejące badania wydajności Gridowych systemów informacyjnych pokazują, że zachodzą w nich duże opóźnienia między czasem wystąpienia zdarzenia, a notyfikacją o tym wystąpieniu [Decandia2007] [Berger2008] [Huang2007]. Potrzebny jest zatem inny mechanizm, który zapewni usługom monitorowania szybką, a zarazem skalowalną notyfikację o pojawieniu się nowych zadań aplikacji workflow.

4.2. Rozwiązanie oparte o infrastrukturę Distributed Hash Table

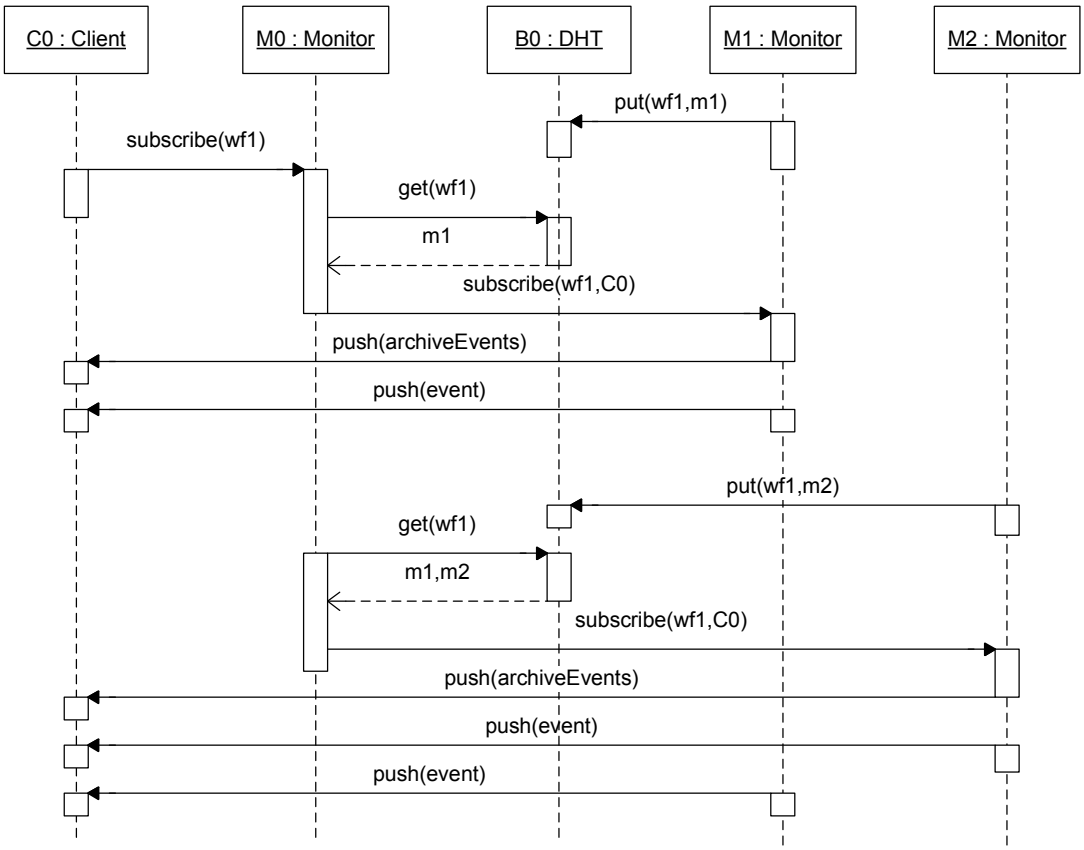
W niniejszej pracy zaproponowano rozwiązanie oparte o infrastrukturę *Distributed Hash Table* (DHT) stowarzyszoną z infrastrukturą monitorowania. Sieć typu DHT dostarcza infrastruktury do przechowywania danych typu klucz – wartość. Typowa infrastruktura DHT zorganizowana jest w nakładkową sieć *peer-to-peer* (*overaly P2P network*), która umożliwia praktycznie liniową skalowalność dla bardzo dużej liczby rekordów i współbieżnych zgłoszeń, a jednocześnie gwarantuje dostęp do każdego klucza w czasie logarytmicznym względem liczby węzłów w sieci P2P. Sieci typu DHT wykorzystuje się z dużym sukcesem do przechowywania dzielonego stanu usług w produkcyjnych systemach dużej skali, czego przykładem mogą być usługi Amazon.com korzystające z systemu Dynamo [Decandia2007].

Idea rozwiązania opiera się o rejestrowanie rekordów „klucz – wartość" w sieci DHT, gdzie kluczem jest unikalny identyfikator aplikacji workflow, zaś wartościami są rekordy opisujące producentów zdarzeń dla tego workflow. Scenariusz monitorowania w trybie *on-line* z automatycznym wykrywaniem zasobów przy użyciu sieci DHT jest przedstawiony na Rysunku Rysunek 6. Trzy Monitory uczestniczą w tym scenariuszu: **M0** otrzymuje żądanie subskrypcji od klienta **C0**, zaś **M1** i **M2** są producentami zdarzeń dla monitorowanego workflow identyfikowanego jako **wf1**. Ponadto w scenariuszu uczestniczy sieć DHT **B0**. Przebieg scenariusza jest następujący:

1. Pierwszy producent danych dla wf1 – monitor M1 – rejestruje się w sieci DHT przed zgłoszeniem jakiejkolwiek subskrypcji.
2. Następnie klient C0 zgłasza subskrypcję do Monitora M0.
3. Monitor M0 pobiera z sieci DHT listę producentów dla workflow wf1. Otrzymuje adres monitora M1.
4. M0 przesyła do M1 żądanie subskrypcji otrzymane od C0. W rezultacie M1 rozpoczyna transfer zdarzeń do C0.
5. Po pewnym czasie drugi producent – M2 – pojawia się i rejestruje w DHT.
6. Monitor M0 periodycznie sprawdza w DHT rekord o kluczu wf1. W tym wypadku okazuje się, że pojawiła się nowa wartość – producent M2.
7. Monitor M0 przekazuje żądanie subskrypcji do Monitora M2, który rozpoczyna przesyłanie zdarzeń do C0.

Przedstawione rozwiązanie charakteryzuje się następującymi cechami:

- Klient może zgłosić żądanie subskrypcji w dowolnym Monitorze. Korzystanie z zewnętrznej usługi wykrywania zasobów nie jest konieczne.
- Skalowalność i wysoka dostępność mechanizmu automatycznego wykrywania zasobów jest zapewniona dzięki temu, że tymi cechami charakteryzuje się wykorzystywana sieć DHT.



Rysunek 6: Monitorowanie w trybie on-line w oparciu o automatyczne wykrywanie zasobów przy pomocy sieci DHT.

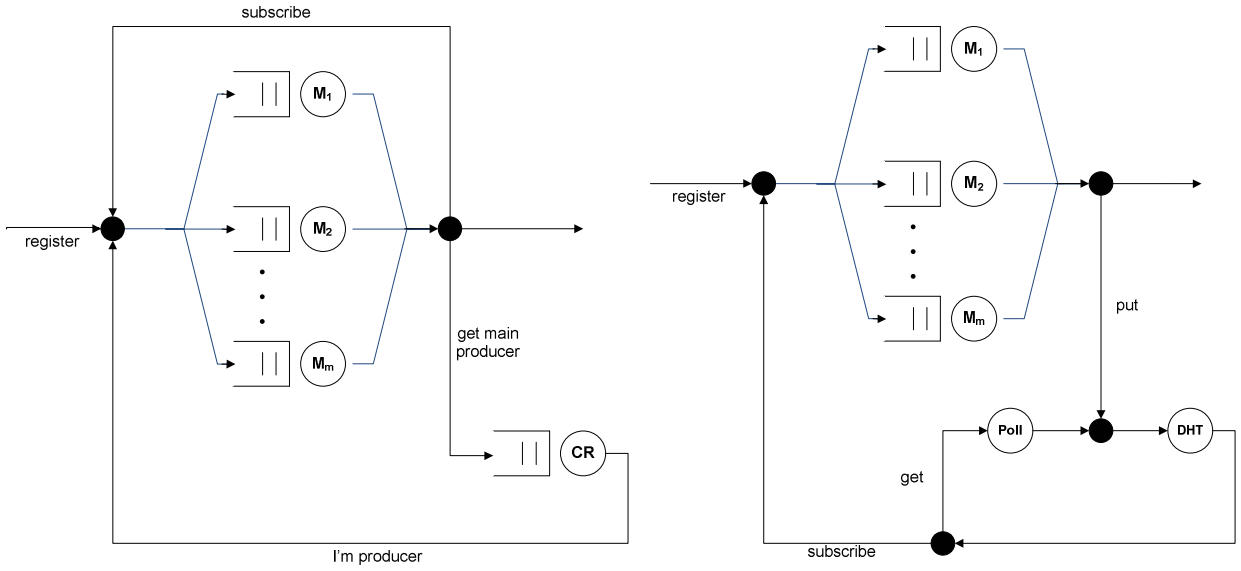
4.3. Pomiar wydajności i skalowalności rozwiązania

W celu oceny wydajności i skalowalności zaproponowanego rozwiązania, zastosowano podejście oparte o budowę modelu systemu i analizę wskaźników wydajności tego modelu. Dla porównania zbudowano również model rozwiązania scentralizowanego, w którym rozproszoną sieć DHT zastępuje centralny koordynator. Do utworzenia i analizy modelu posłużono się dwiema modelem sieci kolejkowych (*queuing networks*) [Menasce2004]. System reprezentowany jest jako sieć kolejek, zaś jego wskaźniki obliczane są analitycznie.

Modele sieci kolejkowych dla obu wariantów przedstawione są na RysunkuRysunek 7. Aby wyniki uzyskane z obliczeń tych modeli były wiarygodne, modele muszą być sparametryzowane wielkościami pochodzącymi z rzeczywistych systemów. Dwa główne parametry modelu sieci kolejkowej to *średnia intensywność zgłoszeń* do systemu (*workload intensity*) oraz *średnie czasy przebywania zgłoszeń* (*service demand*) w poszczególnych centrach obsługi (*service center*).

Intensywność zgłoszeń w tym przypadku to częstotliwość z jaką do systemu zgłaszają się nowe zadania workflow. Z kolei następujące centra obsługi występują w modelach:

- Monitory.
- Sieć DHT.
- Koordynator.



(a) **(b)**

Rysunek 7: Modele sieci kolejkowych dla automatycznego wykrywania zasobów z uzyciem (a) centralnego koordynator, (b) sieci P2P Distributed Hash Table.

Analiza systemów doprowadziła do wniosku, że właściwe jest zastosowanie modelu sieci kolejkowych z wieloma klasami klientów [Menasce2004, s. 40-41]. Parametry modeli określono w sposób następujący:

- Zakres badanych intensywności zgłoszeń przyjęto w granicach od 1/3 do 10 zadań na sekundę. Liczby te wzorowane są na produkcyjnej infrastrukturze Gridowej dużej skali projektu EGEE, w którym średnia dzienna liczba zadań wynosi 30.000, zaś maksymalna sięga 120.000.
- Czasy przebywania zgłoszeń dla Monitora i koordynatora określono przez budowę prototypów tych komponentów i ich testowanie pod kątem wydajności.
- Średnie czasy przebywania zgłoszeń dla sieci DHT przyjęto na podstawie istniejącej literatury dotyczącej pomiaru wydajności dwóch produkcyjnych sieci DHT dużej skali – OpenDHT [Rhea2005] i Amazon Dynamo [Decandia2007].
- Dodatkowo przyjęto, że Monitory, które posiadają aktywną subskrypcję raz na sekundę odpytują sieć DHT o pojawienie się nowych producentów pasujących do tej subskrypcji.

Otrzymane średnie czasy zgłoszeń przedstawione są w Tabeli 1 (model z siecią DHT) i Tabeli 2 (model z centralnym koordynatorem). Wartość 0 oznacza, że dana klasa klientów w ogóle nie korzysta z określonego centrum obsługi.

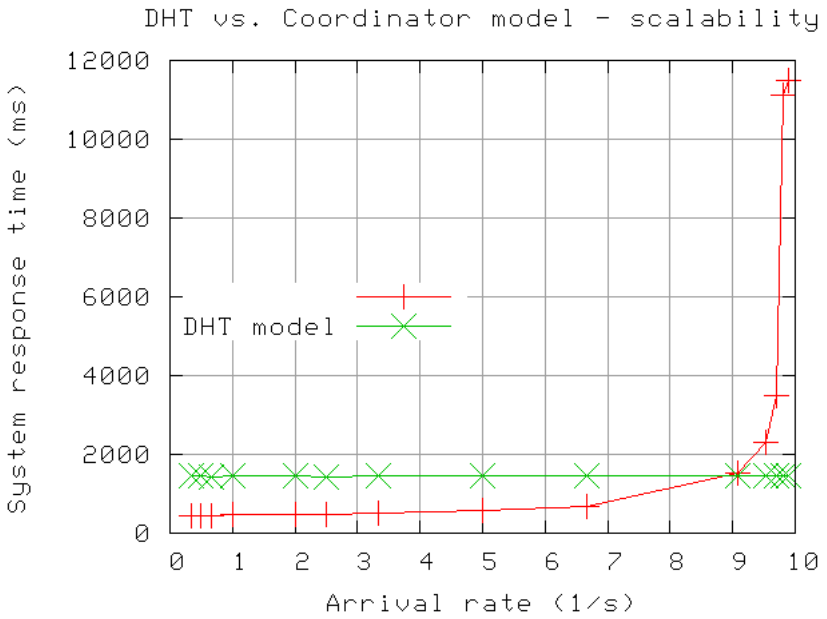
Tabela 1: Średnie czasy zgłoszeń (model z siecią DHT)

	Średni czas zgłoszeń (ms)		
Klasa	Monitor	DHT	Poll
Register	50	0	0
Subscribe	150	0	1000
Put	0	200	0
Get	0	100	0

Tabela 2: Średnie czasy zgłoszeń (model z centralnym koordynatorem)

	Średni czas zgłoszeń (ms)	
Klasa	Monitor	Koordynator
Register	50	0
I'm Producer	150	0
Subscribe	150	0
Get Main producer	0	100

Rysunek 8 przedstawia wyniki ewaluacji – porównanie czasów odpowiedzi systemu (tj. czasu pomiędzy zgłoszeniem nowego zadania i rozpoczęciem transferu zdarzeń pochodzących z monitorowania tego zadania) dla dwóch zaproponowanych rozwiązań – opartego o DHT i o centralny koordynator. Z analizy rysunku można wywnioskować, że o ile dla mniejszej intensywności zgłoszeń model oparty o koordynator jest wydajniejszy, model ten nie jest skalowalny. Rozwiązanie oparte o sieć DHT zachowuje się stabilnie dla wszystkich badanych wielkości intensywności zgłoszeń.



Rysunek 8: Porównanie skalowalności monitorowania on-line w oparciu o DHT i centralny koordynator.

5. Monitorowanie aplikacji zastanych w ramach workflow

5.1. Wprowadzenie

Aplikacje zastane (*legacy applications*) są to takie aplikacje, które zostały utworzone przy pomocy języka programowania lub technologii nie wspieranych w pełni przez nowoczesną infrastrukturę Gridową. Na przykład usługi sieciowe (*Web services*) typowo realizowane są przy pomocy technologii opartych o język Java, podczas gdy wiele użytecznych aplikacji naukowych to aplikacje równoległe napisane w Fortranie lub C. Takie aplikacje są często wykonywane jako zadanie workflow, np. poprzez opakowanie (*wrapping*) ich w fasadowy interfejs usługi sieciowej. Problem monitorowania aplikacji zastanych wykonujących się jako część workflow w środowisku Gridowym jest osobno rozważany w tej pracy z następujących powodów:

- Problem monitorowania aplikacji zastanych w środowisku Gridowym sam w sobie wiąże się z istotnymi wyzwaniami.
- Rozwiązanie tego problemu jest jednocześnie weryfikacją przydatności globalnej infrastruktury monitorowania aplikacji typu worfklow do integracji i umożliwienia interoperabilności zewnętrznych systemów monitorowania.

Jako złożony i jednocześnie typowy przykład naukowej aplikacji zastanej wybrano aplikacje równoległe oparte o *Message Passing Interface* (MPI). Ponieważ istnieje kilka systemów do monitorowania takich aplikacji, najlepszym rozwiązaniem jest integracja jednego z takich systemów do globalnej infrastruktury monitorowania GEMINI. Zadanie to wiąże się z następującymi wyzwaniami:

- *Rozmieszczenie komponentów systemu do monitorowania aplikacji MPI w Gridzie*. Tradycyjne systemy do monitorowania aplikacji MPI są dostosowane do środowisk klastrowych. W takim środowisku systemy do monitorowania są zwykle uruchamiane ręcznie. Jest to jednak możliwe z dwóch powodów: (1) węzły, na których aplikacja zostanie wykonana są znane z góry; (2) użytkownik wykonujący aplikację zwykle posiada konto użytkownika na węzłach klastra. Żadna z tych okoliczności nie występuje w Gridzie. Dlatego konieczne jest opracowanie koncepcji automatycznego rozmieszczania komponentów istniejącego systemu monitorowania w Gridzie.
- *Adaptacja reprezentacji danych i interfejsów*. Integracja istniejącego systemu monitorowania do aplikacji MPI z globalną infrastrukturą dla monitorowania aplikacji workflow wymaga adaptacji interfejsów i reprezentacji danych, które są specyficzne dla adaptowanego systemu monitorowania.

W tym rozdziale wybrano podejście OMIS (*On-line Monitoring Interface Specification*), oraz oparty o to podejście system monitorowania aplikacji równoległych w klastrach, OCM (*OMIS-Compliant Monitor*). Dokonano adaptacji OMIS/OCM do środowisk Gridowych oraz do infrastruktury monitorowania GEMINI. Adaptację tę można podzielić na dwa etapy:

- Rozszerzenie paradygmatu OMIS do środowisk Gridowych i budowa systemu OCM-G, przystosowanego do monitorowania aplikacji równoległych w Gridzie.
- Integracja OCM-G z infrastrukturą GEMINI.

On-line Monitoring Interface Specification (OMIS) definiuje uniwersalny interfejs dla systemu monitorowania. Interfejs ten oparty jest o *usługi*, które dzielą się na trzy kategorie:

- Usługi *informacyjne*, które służą do pobierania informacji o monitorowanych obiektach (np. informacji o stanie procesu).
- Usługi *manipulacyjne*, przy pomocy których zmienia się stan monitorowanych obiektów (np. wstrzymanie procesu).

- Usługi *zdarzeniowe*, przy użyciu których specyfikuje się zdarzenia, które mają być monitorowane.

Usługi informacyjne i manipulacyjne są również nazywane *akcjami*.

Monitorowany system jest przez OMIS postrzegany jako zbiór obiektów, które mogą być monitorowane. OMIS definiuje kilka kategorii obiektów, z których główne to węzeł, proces i wątek. Każdy obiekt identyfikowany jest unikalnym *tokenem*.

Żądanie monitorowania w OMIS polega na specyfikacji jednej lub więcej usług oraz parametrów tych usług, w tym obiektów, które mają być monitorowane. Rozróżnia się dwa typy żądań:

- Żądanie *bezwarunkowe*. Specyfikuje jedną lub więcej akcji, które mają być natychmiast wykonane (np. wstrzymaj proces, potem zwróć informację o jego stanie). W składni OMIS (p – token procesu): thread_stop([p]).
- Żądanie *warunkowe*. Specyfikuje zdarzenie oraz jedną lub więcej akcji, które mają być wykonane w momencie wystąpienia tego zdarzenia (np. gdy utworzony zostanie nowy proces, zwróć informację o węźle, na którym się on wykonuje oraz PID tego procesu). W składni OMIS: thread_creates_proc([p]): proc_get_info([$newproc], 0x1).

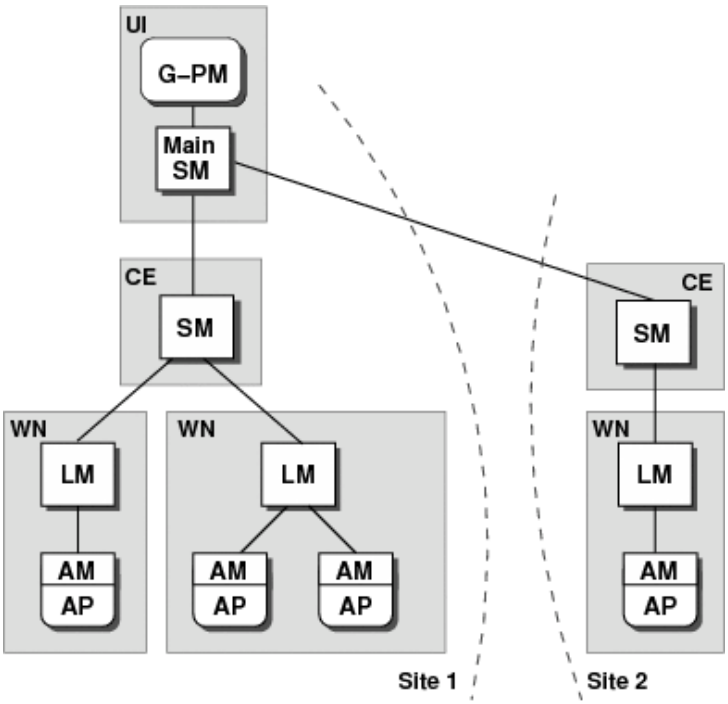
5.2. Rozszerzenie paradygmatu OMIS

Istniejąca specyfikacja OMIS nie była w pełni przystosowana do wsparcia środowisk Gridowych. Z tego powodu opracowano jej rozszerzenie, które wprowadza dwa nowe typy obiektów monitorowanych: *aplikacje* oraz *ośrodki* (*site*). Dzięki tym obiektom z jednej strony lepiej odzwierciedlona jest architektura Gridu (zbiór ośrodków, w ramach których rozmieszczone są węzły, na których wykonują się procesy), z drugiej strony uwzględniony jest fakt, że kolekcja procesów może logicznie stanowić jedną aplikację. Zdefiniowano również nowe usługi dla wprowadzonych typów obiektów.

5.3. System OCM-G

Kolejnym etapem jest budowa systemu OCM-G w oparciu o istniejący system OCM. Do systemu OCM-G wprowadzono następujące innowacje w stosunku do OCM:

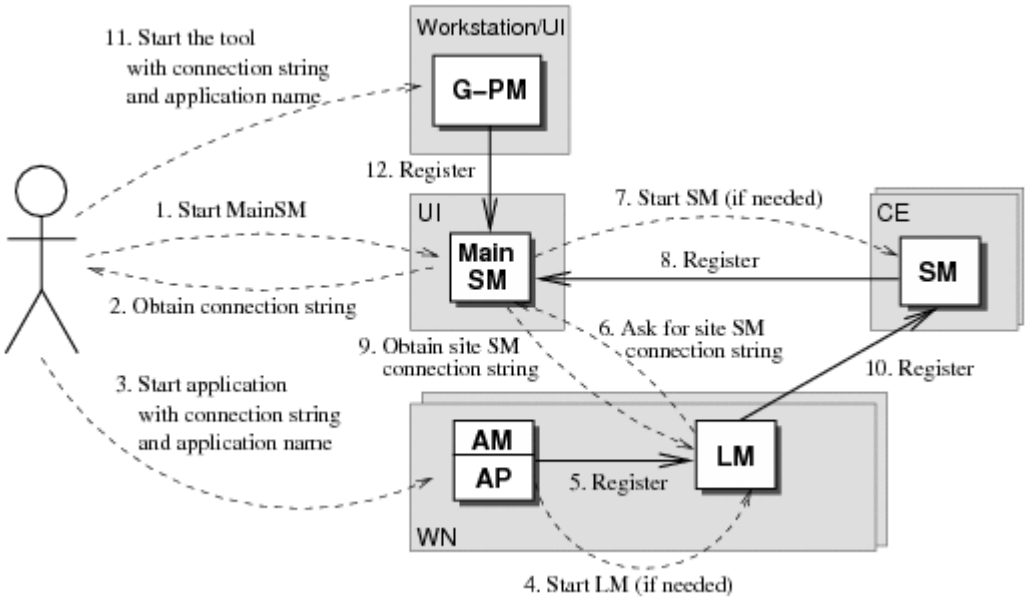
- Architektura systemu została rozszerzona, tak aby mógł on być rozmieszczony w wielu ośrodkach, a nie tylko na pojedynczym klastrze.
- Nowe typy obiektów dodane do OMIS (aplikacje i ośrodki) zostały zaimplementowane w OCM-G wraz z nowymi usługami zdefiniowanymi dla tych obiektów.
- Procedura startu systemu OCM-G została na nowo zaprojektowana, tak aby umożliwić automatyczne rozmieszczanie komponentów systemu monitorowania w Gridzie.
- Koncepcja standardowej pośredniej reprezentacji (SIR) została wprowadzona do OMIS oraz OCM-G, tak aby umożliwić dynamiczną, drobnoziarnistą i selektywną instrumentację aplikacji zastanych.



Rysunek 9: Architektura OCM-G i rozmieszczenie jego komponentów w Gridzie.

Rysunek 9 przedstawia architekturę OCM-G i jednocześnie rozmieszczenie jego komponentów w środowisku Gridowym. System OCM-G składa się z następujących komponentów:

- Lokalne Monitory (LM) odpowiedzialne są za bezpośrednie wykonywanie żądań monitorowania obiektów i zlokalizowane są na tych samych węzłach roboczych (WN), na których znajdują się monitorowane obiekty.
- Zarządcy Usług (SM) zlokalizowani są na wybranym węźle ośrodka (oznaczonym jako *Computing Element*, CE) i odpowiedzialne są za obsługę żądań dotyczących obiektów znajdujących się w tym ośrodku. Przekazują one żądania do lokalnych monitorów i odbierają od nich odpowiedzi.
- Główny Zarządca Usług (Main SM) jest odpowiedzialny za monitorowanie aplikacji konkretnego użytkownika, od którego pobiera żądania (np. za pośrednictwem narzędzia G-PM do pomiaru wydajności) i przekazuje je do stosownych zarządców usług. Main SM może być rozmieszczony na dowolnym węźle, np. na lokalnej stacji roboczej, na której wykonywany jest interfejs użytkownika i narzędzie (UI).
- Dodatkowo, do procesów aplikacji (AP) dołączony jest moduł monitorowania (AM), który odpowiedzialny jest za wykonywanie pewnych akcji bezpośredni w kontekście procesu aplikacji.



Rysunek 10: Procedura startu OCM-G w środowisku Gridowym.

Rysunek 10 prezentuje procedurę automatycznego startu OCM-G w środowisku Gridowym, która umożliwia automatyczne rozmieszczanie komponentów OCM-G. Procedura ta składa się z następujących kroków:

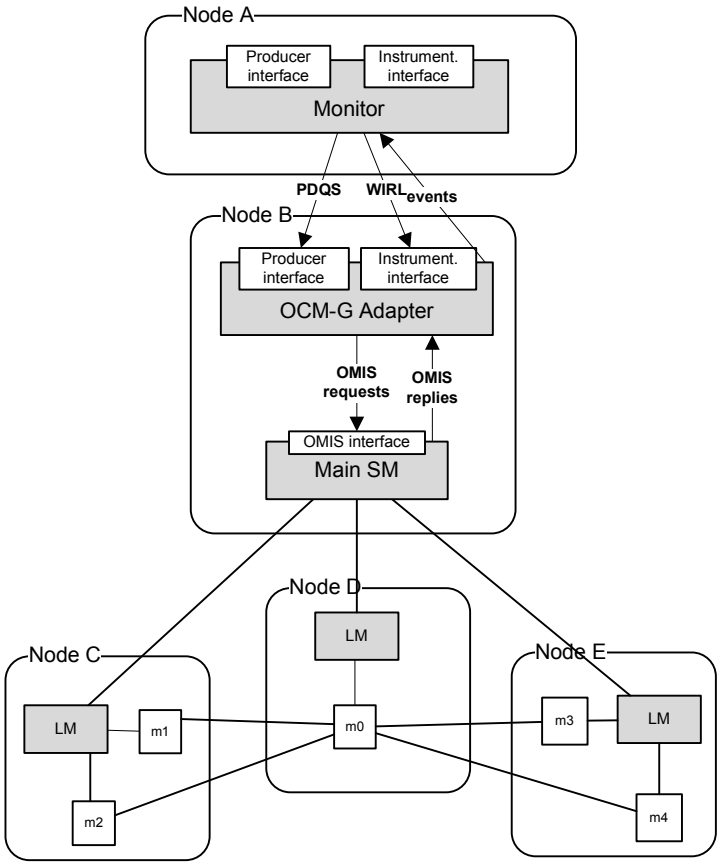
1. MainSM musi być uruchomiony przed wysłaniem aplikacji jako zadania. MainSM może np. uruchamiać się automatycznie wraz z narzędziem (krok 1). MainSM otwiera kanał, na którym oczekuje na zgłoszenia i zwraca adres tego kanału (*connection string*) (krok 2).
2. Przy wysyłaniu aplikacji jako zadanie Gridowe, muszą być przekazane dwa dodatkowe parametry: adres kanału zwróconego przez MainSM oraz nazwa aplikacji wybrana przez użytkownika (krok 3).
3. Gdy procesy aplikacji są startowane na węzłach roboczych (WN), próbują one odszukać instancji lokalnego monitora na tym samym węźle. Gdy to się nie powiedzie, same startują lokalne monitory (krok 4).
4. Każdy process aplikacji rejestruje się w swoim lokalnym monitorze, przesyłając m.in. adres kanału MainSM. (krok 5).
5. Lokalny monitor, po wystartowaniu, kontaktuje się najpierw bezpośrednio z MainSM, któremu przekazuje identyfikator ośrodka, a w odpowiedzi dostaje adres zarządcy usług, do którego ma się podłączyć (krok 6).
6. Gdy MainSM dostaje informację o nowym ośrodku, startuje na nim zarządcę usług, posługując się mechanizmami zlecania zadań (krok 7).
7. Nowo wystartowany menadżer zadań nawiązuje połączenie z MainSM i rejestruje się w nim (krok 8).
8. W tym momencie MainSM może przekazać wszystkim stosownym lokalnym monitorom adres kanału nowego zarządcy usług (krok 9). Następnie połączenie pomiędzy lokalnym monitorem a MainSM zostaje zamknięte.
9. Lokalne monitory nawiązują połączenie ze swoimi zarządcami usług (krok 10).
10. Gdy użytkownik uruchomi narzędzie (np. do pomiaru wydajności G-PM), połączy się ono z OCM-G poprzez MainSM aby wysyłać żądania monitorowania (kroki 11 i 12).

5.4. Integracja OCM-G z GEMINI

Integracja OCM-G z GEMINI wiązała się z kilkoma problemami: (1) konwersją reprezentacji danych oraz interfejsów, (2) wsparciem mechanizmu SIR. Następujące okoliczności utrudniały te zadania:

- Język formułowania żądań monitorowania w OCM-G oparty o OMIS jest raczej imperatywny (specyfikuje usługi, które mają być wykonane, ich parametry i kolejność), podczas gdy interfejs GEMINI oparty o PDQS (*Performance Data Query and Subscribe*) jest deklaratywny (specyfikuje metryki, które należy obliczać oraz miejsca aplikacji workflow, w których należy to robić).
- Żądania instrumentacji w OMIS są niejawne – odbywają się poprzez specyfikację usług zdarzeniowych, które określają, w których miejscach aplikacji należy uaktywnić instrumentację. W GEMINI żądania instrumentacji są jawne, oparte o język WIRL (*Workflow Instrumentation Request Language*), i określają miejsca w aplikacji, w których należy włączyć lub wyłączyć instrumentację.
- Reprezentacja danych w OMIS oparta jest o struktury danych zdefiniowane przez OMIS, które przesyłane są binarnie. W GEMINI przesył danych oparty jest o standardowe zdarzenia, które są reprezentowane w XML.
- W OCM-G wspierana jest selektywna, dynamicznie aktywowana instrumentacja, która jednak nie jest drobnoziarnista. Np. można włączyć instrumentację danej funkcji, ale nie konkretnego *wywołania* tej funkcji. GEMINI wymaga instrumentacji również drobnoziarnistej, specyfikowanej przy pomocy SIR.

Koncepcję integracji GEMINI i OCM-G prezentuje Rysunek 11. Integracja oparta jest o sensor GEMINI, który jednocześnie jest adapterem dla OCM-G. Konwertuje on żądania WIRL i PDQS na żądania OMIS. Jednocześnie konwertuje struktury OMIS na zdarzenia GEMINI.



Rysunek 11: Integracja OCM-G z infrastrukturą GEMINI.

Stosunkowo najtrudniejszym wyzwaniem było wsparcie SIR w OCM-G. Zostało ono rozwiązane dzięki mechanizmowi *sond* (*probes*) w OCM-G. Sonda jest wywołaniem funkcji umieszczonym w dowolnym miejscu kodu aplikacji. To wywołanie może być zainstrumentowane, tak aby wygenerować zdarzenie. Dzięki temu możliwe jest wysyłanie warunkowych żądań OMIS, które odwołują się do konkretnej sondy. W celu wsparcia SIR w OCM-G stworzono narzędzie, które analizuje kod źródłowy aplikacji, identyfikuje regiony kodu, automatycznie wprowadza sondy OCM-G przed i po każdym regionie kodu, a następnie generuje SIR, który jest dołączany do aplikacji podczas konsolidacji [Balis2006]. W trakcie wykonania SIR może zostać pobrany przez nowo dodane żądanie OMIS. Wadą tego rozwiązania jest konieczność dostępu do kodów źródłowych aplikacji, tak aby możliwa była ich analiza i budowa pliku wykonywalnego aplikacji zawierającego SIR. Jest to jednak wada implementacji, nie koncepcji. Istnieją technologie umożliwiające analizę pliku binarnego i umieszczanie instrumentacji w tym pliku, a nawet bezpośrednio w pamięci, po załadowaniu do niej kodu aplikacji.

6. Model informacji do zapisu eksperymentów

6.1. Wymagania i projekt modelu

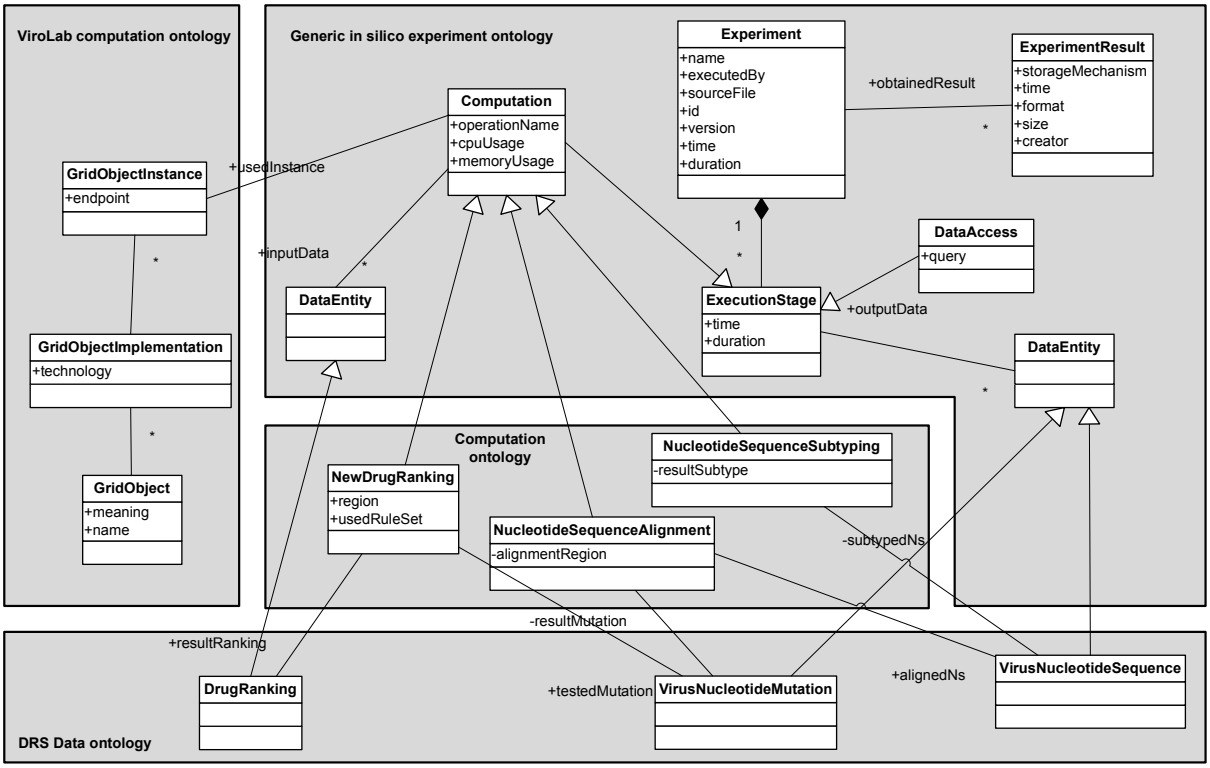
W wielu przypadkach konsumenci danych pochodzących z monitorowania aplikacji zainteresowani są pełnymi zapisami historycznych wykonań tych aplikacji. Dla aplikacji typu workflow przypadki te mają szczególne znaczenie. Należy tu zaliczyć scenariusze istotne dla naukowca – pytanie o pochodzenie (*provenance*) wyników eksperymentu, wyszukiwanie eksperymentów o interesujących atrybutach lub powtarzanie eksperymentów w części lub

w całości; a także scenariusze istotne dla usług middleware w Gridzie, które mogą używać historii wykonań aplikacji do optymalizacji następnych uruchomień.

Analiza tych scenariuszy doprowadziła do sformułowania koncepcji modelu Informacji o Eksperymencie (*Experiment Information*), wymagania wobec którego można scharakteryzować przy pomocy następujących cech:

- *Przydatność do przetwarzania automatycznego i czytelność dla człowieka*. Wymaganie to odzwierciedla fakt, że konsumentami zapisów eksperymentów są zarówno procesy jak i ludzie.
- *Wysoki poziom strukturyzacji*. Proste zapisywanie śladu zdarzeń generowanych w trakcie monitorowanie wykonania aplikacji workflow nie jest wystarczające, gdyż ślad zdarzeń ma zbyt prostą strukturę aby wspierać złożone zapytania. Konieczny jest model wysoce ustrukturyzowany, który odzwierciedlałby strukturę eksperymentu, tj. wchodzące w jego skład elementy (etapy, dane, zasoby, użytkownicy, itp.) oraz powiązania między nimi.
- *Użyteczność*. Model powinien odzwierciedlać różne aspekty informacji opisującej wykonanie się eksperymentu, z precyzją odpowiadającą potrzebom różnorodnych konsumentów. Na przykład zapisy eksperymentów powinny umożliwić odtwarzanie pochodzenia rezultatów, odtwarzanie eksperymentu, czy też pobieranie informacji o wykorzystaniu zasobów w trakcie wykonania.
- *Rozszerzalność*. Model powinien być rozszerzalny o nowe aspekty informacji wynikające z nowo pojawiających się wymagań, w taki sposób aby zachować kompatybilność wstecz względem konsumentów i producentów korzystających z poprzednich wersji modelu.
- *Wzbogacenie o semantykę dziedzinową*. Zapisy eksperymentów powinny zawierać informację o semantyce specyficznej dla dziedziny, której dotyczy eksperyment. Umożliwi to wyszukiwanie eksperymentów, które posiadają pewne cechy istotne dla naukowca.
- *Elastyczność*. Ponieważ niektóre aspekty informacji mogą nie zawsze być dostępne, model powinien umożliwiać tworzenie poprawnego i użytecznego zapisu na podstawie częściowych informacji (np. bez informacji o semantyce).

Po przeanalizowaniu wyżej wymienionych wymagań, wybrano *Web Ontology Language* (OWL) jako język opisu modelu i reprezentacji danych zgodnych z tym modelem (reprezentacja XML OWL/RDF). Ontologie opisane w OWL w sposób naturalny posiadają większość wymienionych wyżej cech, np. rozszerzalność, elastyczność oraz przydatność do opisu semantyki. Ontologie OWL są również dostosowane do przetwarzania maszynowego, a jednocześnie są uważane za czytelne dla człowieka.



Rysunek 12: Model Informacji o Eksperymencie. Pokazano przykładowe ontologie domenowe z dziedziny aplikacji do badania oporności wirusa HIV na leki (Drug Resistance).

- *Ontologia eksperymentu in silico* jest centralną ontologią modelu, która określa strukturę eksperymentu – jego etapy (obliczenia lub dostęp do danych), dane wejściowe, rezultaty, informacje o wykorzystanych zasobach, itp.
- *Ontologia zasobów obliczeniowych ViroLab* opisuje model zasobów obliczeniowych wirtualnego laboratorium ViroLab. W tym modelu zasoby obliczeniowe reprezentowane są jako tzw. Obiekty Gridowe (GridObjects), które udostępniają interfejs programistyczny.
- *Domenowa ontologia danych* opisuje semantykę danych używanych w eksperymencie. Pojęcia odpowiedzialne za semantyczny opis danych są specjalizacją ogólnego pojęcia *DataItem*, które należy do ogólnej ontologii eksperymentu.
- *Domenowa ontologia obliczeń* opisuje semantykę obliczeń wykonywanych w trakcie eksperymentu. W podobny sposób jak w przypadku ontologii danych, pojęcia tej ontologii są specjalizacjami ogólnego pojęcia *Computation* wchodzącego w skład ogólnej ontologii eksperymentu.

6.2. Własności modelu

Opisany model posiada szereg interesujących własności i umożliwia formułowanie użytecznych zapytań. Należy tu wymienić:

1. *Semantyczne formułowanie zapytań*. Dzięki temu, że model zawiera informacje o semantyce danych i obliczeń możliwe jest zadawanie zapytań formułowanych w języku dziedziny nauki, której dotyczy eksperyment. Przykładem takiego zapytania może być ‘wybierz wszystkie eksperymenty, w których danymi wejściowymi były *sekwencje nukleotydów*’. W tym wypadku typ danych wejściowych do eksperymentu mógł być zwykłym łańcuchem znaków. Informacja o tym, że łańcuch ten opisuje *sekwencję*

nukleotydów jest dostępna tylko dzięki temu, że informacja ta została zapisana w trakcie monitorowania eksperymentu.

2. *Semantyczną integrację danych i meta danych*. Pojęcia reprezentujące dane wykorzystywane w trakcie eksperymentu mogą być *odwzorowane* na fizyczne zbiory danych, które te dane przechowują [Wache2001]. Umożliwia to semantyczną integrację tych danych oraz *meta danych*, które opisują *proces, w którym używane były dane*. Dzięki tej semantycznej integracji możliwe jest formułowanie takich zapytań, które *jednocześnie nakładają ograniczenia na strukturę meta danych oraz danych*. Przykładem takiego zapytania może być 'wybierz wszystkie eksperymenty, w których danymi wejściowymi były sekwencje nukleotydów, *zawierające mutacje X*'. Dane wejściowe do eksperymentu mogły być pobrane z bazy danych. W konsekwencji ich struktura jest zapisana w bazie danych, nie w ontologii danych. Odwzorowanie pojęć ontologii na tabele relacyjne pozwala logicznie połączyć rekordy ontologiczne z rekordami relacyjnymi. W tym wypadku informacja o mutacjach jest zapisana w bazie danych. Wykonanie tego zapytania wymaga zatem odwołanie się zarówno do bazy ontologicznej jak i bazy relacyjnej.
3. *Pytanie o pochodzenie (provenance) wyników eksperymentu*. Pochodzenie wyników eksperymentu może mieć dwojaką postać: grafu zależności zadań odzwierciedlającego proces, który doprowadził do powstania tego wyniku lub grafu zależności danych, który pokazuje jakie dane przyczyniły się do powstania końcowego rezultatu. Zaproponowany model umożliwia obliczenie obydwu postaci pochodzenia wyniku dzięki zastosowaniu globalnie unikalnych identyfikatorów danych wejściowych i wyjściowych. Zakłada się zależność pomiędzy daną wyjściową i wejściową jeśli posiadają one ten sam identyfikator.
4. *Wnioskowanie w oparciu o relację generyczności – specyficzności.* Pojęcia ogólne w modelu są powiązane z pojęciami szczegółowymi za pomocą relacji uogólnienia (pojęcia ogólniejsze są rodzajem, pojęcia bardziej szczegółowe – gatunkiem). Umożliwia to zadawanie zapytań na różnych poziomach szczegółowości. Można zatem zapytać o wszystkie obliczenia o określonych cechach, lub tylko obliczenia dopasowywania sekwencji nukleotydów (*sequence alignment*) o tych cechach. Zapytanie ogólniejsze – dzięki relacji uogólnienia – zwróci również rezultaty drugiego zapytania. Ponadto, w wypadku rozszerzenia modelu zapytania ogólne będą automatycznie obejmować nowo dodane pojęcia szczegółowe.

6.3. Budowa zapisu eksperymentu

Aby utworzyć zapis eksperymentu zgodny z wprowadzonym modelem informacji, konieczna jest agregacja niskopoziomowych zdarzeń pochodzących z monitorowania eksperymentu i translacja ich do rekordów ontologicznych. Standaryzacja zarówno modelu zdarzeń jak i modelu informacji umożliwiła *zaproponowanie standardowego procesu agregacji*. Proces ten opisywany jest przy pomocy zestawu reguł, które określają:

- Kiedy należy utworzyć instancje indywiduów ontologicznych.
- W jaki sposób obliczyć atrybuty tych indywiduów.
- Jakie powiązania powinny być ustanowione pomiędzy indywiduami.

Najprostsza reguła określa w jaki sposób stworzyć indywiduum *Eksperymentu*, który reprezentuje unikalny eksperyment. Do tego celu potrzebne jest otrzymanie dwóch zdarzeń:

- *software.execution.started.application.workflow*

- *software.execution.finished.application.workflow*

Zdarzenia te powinny mieć taki sam identyfikator korelacji (ECID) na pierwszym poziomie hierarchii (tj. identyfikatory eksperymentu powinny być identyczne).

W przypadku indywiduum reprezentującego wykonanie operacji w ramach eksperymentu (*ExperimentStage*), konieczne są zdarzenia *started.process.wfActivity* oraz *finished.process.wfActivity*, jednakże ich identyfikatory korelacji powinny być identyczne na pierwszym i drugim poziomie hierarchii (tj. identyfikatory eksperymentu i operacji powinny być takie same).

Inne reguły określają jak obliczać atrybuty utworzonych indywiduów. Np. aby obliczyć czas trwania eksperymentu – atrybut *Experiment/duration* – konieczne są atrybuty dwóch zdarzeń:

1. *started.process.workflow/time*
2. *finished.process.workflow/time*

Z regułą można stowarzyszyć dowolny kod, który ma być wykonany w celu obliczenia wartości atrybutu.

W ramach pracy stworzono prototyp *semantycznego agregatora* – który konsumuje zdarzenia pochodzące z monitorowania eksperymentu i tworzy zapis eksperymentu zgodny z zaproponowanym modelem informacji. Semantyczny agregator korzysta ze zbioru reguł, które określają zasady agregacji.

6.4. Przykład wykorzystania modelu

Model Informacji o Eksperymencie został zweryfikowany w oparciu o monitorowanie aplikacji do badania oporności wirusa HIV na mieszanki leków (*Drug Resistance*). W tej aplikacji, wirus HIV izolowany jest z próbki krwi pacjenta, następnie identyfikowana jest sekwencja nukleotydów RNA tego wirusa, po czym przeprowadza się dopasowywanie (*alignment*) tej sekwencji w celu identyfikacji genów i białek, za które są one odpowiedzialne. Kolejnym etapem jest wykrywanie mutacji w genach. Na końcu obliczana jest najlepsza mieszanka leków dla wykrytych mutacji. W tym celu wybiera się jeden z kilku istniejących zestawów reguł (*rule set*), które określają działanie kilkunastu dostępnych leków na wirusa z określonymi mutacjami. Eksperyment składa się zatem z następujących etapów:

- Sekwencjonowanie wirusa HIV (identyfikacja sekwencji nukleotydów RNA).
- Identyfikowanie genów przez dopasowanie sekwencji (*sequence alignment*).
- Wykrywanie mutacji w genach.
- Obliczanie najlepszej mieszanki leków (*drug ranking*).

W trakcie wykonywania wielu takich eksperymentów zbierano zdarzenia pochodzące z ich monitorowania, agregowano i konwertowano na zapisy eksperymentu zgodne z modelem Informacji o Eksperymencie. W ten sposób powstało repozytorium zawierające zapisy wielu eksperymentów. Repozytorium to wykorzystano by zademonstrować:

- Przydatność modelu do budowy użytecznych zapytań.
- Zorientowane na końcowego użytkownika, wizualne formułowanie zapytań w oparciu o ontologie
- Zademonstrowanie semantycznej integracji danych i meta danych.

6.4.1. Przykłady zapytań

Aby pokazać przydatność modelu informacji do ekstrakcji użytecznej informacji, sformułowano kilka złożonych zapytań opartych o aplikację oporności wirusa HIV na leki. Zapytania te wzorowano na zapytaniach sformułowanych w ramach *Provenance Challenge* [Moreau2008]. Dzięki temu można mieć pewność, że są to zapytania użyteczne i o stosownym stopniu komplikacji. Następnie zapytania te zaimplementowano w języku XQuery, który operuje bezpośrednio na modelu informacji zaimplementowanym w XML/OWL. Poniżej przedstawiono te zapytania, oraz ich implementacje w XQuery.

1. *Zwróć proces, który doprowadził do powstania konkretnej oceny mieszanki leków*. Jest to typowe pytanie o pochodzenie rezultatu eksperymentu.

```
declare function local:data-provenance1($dataId as xs:string)
as list {
  for $op in //ExecutionStage
  where $op//outputData contains $dataId
  return
    {
      for $input in $op//inputData
      return
        $input
        local:data-provenance1($input)
    }
};

declare function local:data-provenance($dataId as xs:string)
as element {
  for $op in //ExecutionStage
  where $op//outputData contains $dataId
  return
    <dataProvenance>
      $op//outputData
      local:data-provenance1($op//dataId)
    </dataProvenance>
};

<provenance>
  local:data-provenance({dataId as passed})
</provenance>
```

2. *Znajdź wszystkie operacje wykonane po etapie dopasowywania sekwencji, które doprowadziły do powstania konkretnej oceny mieszanki leków*.

```
<provenance>
  for $op in //ExecutionStage
    $exp in //Experiment
    where
    $exp/outputData@[name()='rdf:resource'
    and . eq {drug ranking id}]
    and $exp/stageContext/@rdf:resource = $op/@rdf:Resource
    and $op/stageNumber > 1
  return $op
</provenance>
```

3. *Znajdź wszystkie operacje, które doprowadziły do powstania konkretnej oceny mieszanki leków i były wykonywane jako trzecia, czwarta i piąta w kolejności.*

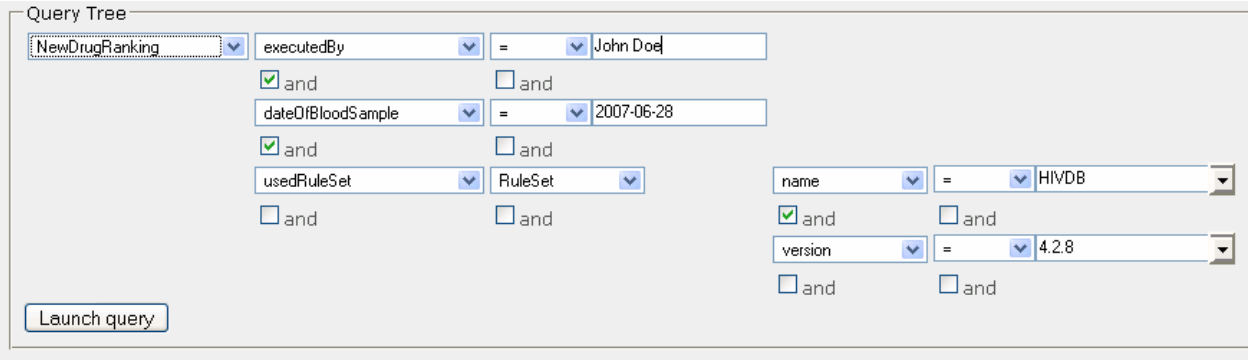
```
<provenance>
 for $op in //ExecutionStage
   $exp in //Experiment
   where
     $exp/outputData@[name() = 'rdf:resource'
     and . eq {drug ranking id}]
     and $exp/stageContext/@rdf:resource = $op/@rdf:Resource
     and $op/stageNumber in {3, 4, 5}
   return $op
</provenance>
```

4. *Znajdź wszystkie operacje dopasowywania sekwencji wykonane 10.10.2007, w których użyto konkretnej sekwencji nukleotydów*.

```
<provenance>
 for $op in //ExecutionStage
   $exp in //Experiment
   where
     $exp/date eq '10.10.2007'
     and $op/inData@[name() = 'vl-data-protos:dasId'
     and . eq {nucleotide sequence id}]
     and $exp/stageContext/@rdf:resource = $op/@rdf:Resource
     and $op/stageNumber = 1
 return $op
</provenance>
```

6.4.2. Wizualne formułowanie zapytań

Końcowymi użytkownikami zapisów eksperymentów są naukowcy, którzy często nie są ekspertami w dziedzinie technologii informacyjnych, np. nie posiadają znajomości języków zapytań do baz danych takich jak SQL lub SPARQL. Aby umożliwić budowanie złożonych zapytań takim użytkownikom, konieczne są wizualne narzędzia, które to ułatwiają. Zastosowanie ontologii do opisu modelu Informacji o Eksperymencie szczególnie sprzyja wizualnemu formułowaniu zapytań, gdyż ontologie domenowe konceptualizują dziedzinę nauki posługując się terminami zrozumiałymi dla naukowca. W ramach pracy stworzono narzędzie do wizualnego formułowania zapytań do repozytorium zawierającego zapisy eksperymentów [Balis2007]. Rysunek 13 przedstawia przykładowe zapytanie stworzone w graficznym interfejsie użytkownika tego narzędzia.



Rysunek 13: Wizualna budowa zapytania w oparciu o ontologie powiązane w modelu Informacji o Eksperymencie.

Wizualna konstrukcja sprowadza się do utworzenia *drzewa zapytania* i rozpoczyna się od wyboru jednego z listy dostępnych pojęć (klas ontologicznych), w tym wypadku *NewDrugRanking*. Następnie formularz jest dynamicznie rozbudowywany – wyświetlana jest lista wyboru zawierająca atrybuty wybranej klasy. Wśród atrybutów mogą znajdować się:

- *Literały*, których posiadają wartość określonego typu.
- *Własności obiektowe*, które są powiązaniami z innymi klasami.

Gdy wybraną własnością jest literał, wyświetlane jest pole, do którego użytkownik może wpisać jego wartość. Gdy jednak wybrana została własność obiektowa, drzewo zapytania jest rozbudowywane poprzez dodanie klasy skojarzonej z wybraną własnością, dla której znowu można wybierać atrybuty.

W podanym przykładzie utworzono zapytanie „*Wybierz wszystkie obliczenia najlepszej mieszanki leków wykonane przez «Johna Doe», w których próbkę krwi pobrano 28-06-2007, i w których użyto zbioru reguł o nazwie «HIVDB» w wersji 4.2.8*”.

W powyższym zapytaniu:

- Atrybuty *executedBy* i *dateOfBloodSample* są literałami i użytkownik wpisuje pożądane dla nich wartości.
- Atrybut *usedRuleSet* prowadzi do powiązanej klasy *RuleSet*, która reprezentuje zestaw reguł – jedną z danych wejściowych dla obliczenia.
- Klasa *RuleSet* jest *odwzorowana na rekordy relacyjnej bazy danych* przechowującej informacje o zbiorach reguł. Klasa ta nie zawiera atrybutów, a jedynie informację o tym, w której bazie danych i której tabeli przechowywane są rekordy reprezentujące zbiory reguł. W tym wypadku atrybuty *name* i *version* **nie są atrybutami klasy** *RuleSet*, lecz **nazwami kolumn tabeli**. Nazwy te zostały **dynamicznie pobrane** z serwera bazy danych.

Po wypełnieniu formularza, utworzone drzewo zapytania jest przetwarzane i translowane na języki zapytań właściwe dla repozytoriów, w których zgromadzone są dane. W powyższym przykładzie w grę wchodzą dwa repozytoria: w pierwszym zgromadzone są zapisy eksperymentów w OWL (język *XQuery*), w drugim dane o zbiorach reguł (język SQL).

Przykład ten demonstruje kilka interesujących możliwości zaproponowanego modelu:

- Zapytanie tworzone jest w **języku zrozumiałym dla końcowego użytkownika**. Ontologiczna struktura modelu służy zarówno jako *język zapytań* dla końcowego

użytkownika, jak i model *reprezentacji zapytania*, który jest tłumaczony na języki repozytoriów danych.

- Powyższe zapytanie jest **przykładem semantycznej integracji danych i meta danych**. Chociaż zapytanie zasadniczo dotyczy meta danych opisujących eksperyment, pozwala też nałożyć ograniczenia na dane używane w tym eksperymencie, w tym wypadku zbiór reguł. Semantyczna integracja możliwa jest dzięki semantycznemu odwzorowaniu klas ontologicznych na zbiory danych.
- Przykład ten pokazuje również możliwość tworzenia zapytania, które odwołuje się do **wielu heterogenicznych zbiorów danych**, jednakże heterogeniczność ta jest ukryta za warstwą ontologicznej reprezentacji. W tym wypadku część zapytania odnosi się do repozytorium Informacji o Eksperymencie, inna część zaś do relacyjnej bazy danych przechowującej dane o zbiorach reguł. Końcowy rezultat zapytania jest wynikiem scalenia rezultatów tych części.

7. Podsumowanie

Zaawansowana cyberinfrastruktura dla e-Nauki pozwala na wykonywanie eksperymentów *in silico*, co umożliwia przełomowe odkrycia naukowe i usprawnia codzienną pracę naukowca. Kluczowymi elementami takiej infrastruktury są technologie Gridowe, technologie sieci semantycznej (*Semantic Web*) oraz *aplikacje naukowe typu workflow*. Aplikacje te pełnią istotną rolę z punktu widzenia naukowca, któremu umożliwiają specyfikowanie eksperymentów *in silico*, oraz infrastruktury, która używa tej specyfikacji do uruchomienia eksperymentu na dostępnych zasobach.

Monitorowanie aplikacji naukowych jest istotne w wielu zastosowaniach. Monitorowanie Gridowych aplikacji naukowych typu workflow niesie ze sobą szereg specyficznych problemów. Specyfika ta wynika z jednej strony z natury infrastruktury Gridowej, która charakteryzuje się dynamiką, zmienną dostępnością zasobów, co w konsekwencji powoduje, że cykl wykonania aplikacji typu workflow jest w Gridzie bardzo złożony. Z drugiej strony zasadniczą rolę odgrywa bogactwo samych scenariuszy, w których monitorowanie aplikacji typu workflow jest istotne. Dane pochodzące z monitorowania są konsumowane zarówno w trybie *on-line*, jak i *off-line*. Konsumentami są zarówno procesy jak i ludzie. Istotne są aktualne informacje dotyczące eksperymentu wykonującego się na bieżąco, a także historyczne zapisy eksperymentów.

Głównym wkładem naukowym tej pracy było zidentyfikowanie problemów specyficznych dla monitorowania Gridowych aplikacji naukowych typu workflow, zaproponowanie rozwiązań tych problemów oraz ich weryfikacja. Te problemy to:

- Budowa infrastruktury odpowiedniej do monitorowania heterogenicznych, rozproszonych i luźno powiązanych Gridowych aplikacji naukowych typu workflow.
- Zapewnienie monitorowania tych aplikacji w trybie on-line.
- Wsparcie dla monitorowania aplikacji zastanych wykonujących się w ramach workflow.
- Opracowanie modelu informacji do reprezentacji zapisu eksperymentów.

7.1.1. Infrastruktura do monitorowania Gridowych aplikacji naukowych typu workflow

Pierwsze wyzwanie dotyczy podstawowych usług do monitorowania, które powinny być zrealizowane w taki sposób, aby ukrywać heterogeniczność aplikacji workflow pod względem

języków programowania i platform, na których wykonują się zadania workflow. Ze względu na tę heterogeniczność, monitorowania pojedynczego workflow może wymagać współdziałania wielu heterogenicznych narzędzi i technologii do instrumentacji i monitorowania. Konieczne jest zapewnienie interoperabilności tych narzędzi przy jednoczesnym dostarczeniu jednolitego interfejsu do specyfikowania żądań monitorowania i instrumentacji, oraz wspólnej reprezentacji danych. Aby to zapewnić, zaproponowano model standardowej hierarchii typów zdarzeń reprezentujących podstawowe pomiary pochodzące z monitorowania. Taka hierarchia umożliwia interoperabilność opartą o wspólny model danych wymienianych, a nie wspólny narzucony model systemu, który musi być wewnętrznie implementowany przez każde narzędzie. W konsekwencji możliwa jest interoperabilność poprzez luźną kooperację, a nie ścisłą integrację. Drugim ważnym elementem jest zapewnienie możliwości faktyczne współdziałania różnych narzędzi w ramach jednej infrastruktury. Aby to umożliwić, infrastruktura monitorowania została zaprojektowana jako rama programowa (*framework*), do której można zaadoptować narzędzi do monitorowania i instrumentacji. Zaadoptowano również standardowe interfejsy do specyfikowania żądań monitorowania i instrumentacji. Zastosowano technologie *Complex Event Processing*, które okazały się dobrze spełniać wymagania w zakresie ekspresywności języka formułowania żądań monitorowania, oraz efektywności silnika obsługi subskrypcji. Koncepcja *standardowej pośredniej reprezentacji* została zaadoptowana aby umożliwić abstrakcyjne specyfikowanie miejsc w aplikacji, które mają podlegać instrumentacji. Problem korelacji został rozpoznany i zaproponowano hierarchiczne identyfikatory korelacji (ECID), aby umożliwić precyzyjną identyfikację pochodzenia zdarzenia w sensie miejsca w aplikacji workflow. Na bazie tych elementów zaprojektowano i stworzono infrastrukturę do monitorowania Gridowych aplikacji typu workflow – GEMINI.

7.1.2. Wsparcie dla monitorowania w trybie on-line

Drugie wyzwanie jest związane z potrzebą monitorowania Gridowych aplikacji typu workflow w trybie on-line. Z powodu skomplikowanego cyklu życia takich aplikacji, a także dynamicznej natury samej infrastruktury Gridowej, okazało się, że największym wyzwaniem jest w tym wypadku wykrywanie zasobów. Zdefiniowano automatyczne wykrywanie zasobów jako sposób rozwiązania problemu. W automatycznym wykrywaniu zasobów nowo pojawiający się producenci danych pasujący do istniejącej subskrypcji zarejestrowanej w systemie monitorowania są automatycznie wykrywani, a następnie przekazywane jest im żądanie subskrypcji. Analiza istniejących Gridowych systemów informacyjnych doprowadziła do wniosku, że nie nadają się one do wsparcia takiego scenariusza w sposób wystarczająco efektywny, aby możliwe było monitorowania on-line. Zaproponowano zatem alternatywne rozwiązanie oparte o infrastrukture *Distributed Hash Table* (DHT) stowarzyszoną z infrastrukturą monitorowania. Celem tego rozwiązania nie było zastąpienie Gridowych usług informacyjnych, ale raczej dostarczenie efektywnej, niezawodnej i skalowalnej infrastruktury do utrzymywania współdzielonego stanu rozproszonych usług monitorowania. Istniejące sieci DHT dobrze spełniają te wymagania. Aby zweryfikować skalowalność zaproponowanego algorytmu zbierania zdarzeń w trybie on-line, opartego o DHT, stworzono model systemu i zbadano jego wydajność dla różnych intensywności zgłoszeń. Dla porównania zbudowano również i zbadano model rozwiązania scentralizowanego. Wyniki badań potwierdziły dużą efektywność i skalowalność zaproponowanego rozwiązania. Jednocześnie stwierdzono, że proste scentralizowane rozwiązanie w pewnych wypadkach równie dobrze zdaje egzamin.

7.1.3. Monitorowania aplikacji zastanych w ramach workflow

Trzecie wyzwanie związane jest ze specyfiką aplikacji naukowych typu workflow, w których zasadnicze obliczenia często wykonywane są przez aplikacje zastane, które łatwiej jest zaadoptować do nowoczesnej infrastruktury Gridowej, niż napisać na nowo. Jednakże obecność takich aplikacji komplikuje monitorowanie na tyle, że problem ten zasługuje na osobną uwagę. Wzięto pod uwagę równoległe aplikacje MPI jako typowy i najbardziej złożony przykład aplikacji zastanych. Zgodnie z założonym paradygmatem, w którym infrastruktura GEMINI służy jako rama programowa dla istniejących systemów monitorowania, zdecydowano się na adaptację istniejącego podejścia OMIS i opartego na nim systemu OCM do monitorowania aplikacji równoległych. Zidentyfikowano dwa aspekty problemu monitorowania aplikacji zastanych wykonujących się w ramach workflow: dostosowanie istniejących systemów monitorowania takich aplikacji do środowiska gridowego, oraz integrację tych systemów z infrastrukturą GEMINI. Stworzono oparty o OCM system OCM-G, o zmienionej architekturze, dostosowanej do Gridu. Rozwiązano problem automatycznego rozmieszczania komponentów istniejącego systemu monitorowania w Gridzie. Rozszerzono specyfikację OMIS, tak aby odzwierciedlała budowę Gridu. Rozszerzono system OCM-G o wsparcie dla koncepcji standardowej pośredniej reprezentacji. Dostosowanie systemu OCM-G do GEMINI posłużyło jednocześnie jako studium przypadku adaptacji istniejącego systemu monitorowania do infrastruktury GEMINI i było weryfikacją projektu tej infrastruktury.

7.1.4. Model informacji do zapisu przebiegu wykonania aplikacji typu workflow

Czwarte wyzwanie wiąże się z wykorzystaniem zapisów poprzednich uruchomień aplikacji naukowych typu workflow (eksperymentów). Analiza scenariuszy, w których takie zapisy są istotne, doprowadziła do koncepcji modelu informacji, który opisywałby przebieg wykonania aplikacji naukowej typu workflow w sposób ustrukturyzowany. Stwierdzono, że istniejące modele informacji pochodzącej z monitorowania dotyczą aktualnego stanu zasobów Gridowych, nie istnieją natomiast modele informacji opisujące przebieg wykonania aplikacji. Wprowadzenie wspólnego modelu informacji umożliwia interoperabilność istniejących i przyszłych narzędzi w oparciu o ten model. Ze względu na zidentyfikowane wymagania względem modelu informacji, użyte zostały technologie semantycznej sieci (*Semantic Web*) do opisania modelu, oraz do reprezentacji faktycznych danych zgodnych z tym modelem. Stworzony został model *informacji o eksperymencie*, który za pomocą ontologii opisuje eksperyment, jego etapy (w tym obliczenia i operacje dostępu do danych), wykorzystane zasoby, itp. Stwierdzono również, że niezwykle istotne jest, aby zapis eksperymentu był wzbogacony o semantykę z dziedziny, której dotyczył eksperyment. W tym celu model informacji o eksperymencie powiązano z domenowymi ontologiami, które modelują znaczenie obliczeń i danych wykorzystywanych w eksperymencie. Zaproponowano ogólny mechanizm agregacji niskopoziomowych zdarzeń pochodzących z monitorowania do ustrukturyzowanej informacji o eksperymencie. Zbadano możliwości wprowadzonego modelu względem wsparcia dla złożonych zapytań. Pokazano, że dzięki zastosowaniu ontologii, model umożliwia semantyczną integrację danych i meta danych, gdzie meta dane opisują proces, w którym używane były dane. Pokazano również w jaki sposób model umożliwia wizualne formułowanie zapytań w sposób zorientowany na końcowego użytkownika, nie będącego ekspertem w dziedzinie technologii informacyjnych.

7.1.5. Wnioski

Wszystkie wyżej opisane rozwiązania zostały zweryfikowane przy użyciu różnorodnych metod. Stworzony został prototyp zaprojektowanej infrastruktury GEMINI, a także systemu OCM-G. Dodatkowe wspomagające narzędzia zostały utworzone pod kierownictwem autora. Prototypy te zostały rozmieszczone na infrastrukturach dwóch projektów finansowanych przez Unię Europejską – K-Wf Grid [Bubak2007] oraz ViroLab [Bubak2008]. W projektach tych monitorowano aplikacje naukowe typu workflow. Zebrane w ten sposób dane, w tym zapisy zgodne z modelem informacji o eksperymencie, posłużyły do przeprowadzenia eksperymentów potwierdzających hipotezy badawcze i weryfikujących cele naukowe sformułowane w tej pracy. Skalowalność zaproponowanego rozwiązania do monitorowania w trybie on-line została zweryfikowana przy pomocy podejścia opartego o analizę modelu systemu. Zastosowano dwa podejścia do budowy modelu: formalizm sieci kolejkowych oraz symulację. Parametry dla modeli zostały zebrane przez badanie wydajności stworzonych prototypów oraz na podstawie dostępnej literatury.

7.1.6. Perspektywy dalszego rozwoju badań

Szereg problemów przedstawionych w ramach pracy lub ściśle związanych z jej tematyką może być przedmiotem dalszych badań.

- Przedstawione modele zdarzeń oraz informacji dla danych pochodzących z monitorowania mogą być uzupełnione o dalsze typy pomiarów lub szczegółowe atrybuty, a także rozszerzone na aplikacje z innych dziedzin nauki.
- Model sieci kolejkowych systemu monitorowania został w pracy zweryfikowany przez porównanie go do modelu symulacyjnego. Równie pożądane jest przeprowadzenie eksperymentów w rzeczywistym środowisku dużej skali. Ponadto, wyniki pochodzące z modelu mogą być dokładniejsze poprzez opracowanie bardziej szczegółowej charakterystyki zachowania się rzeczywistych aplikacji typu workflow w środowiskach produkcyjnych. Wymaga to jednak dostępu do dużej ilości danych historycznych.
- W pracy przedstawiono model informacji opisujący przebieg wykonania historycznych aplikacji workflow. Również użyteczny mógłby być model informacji opisujący *aktualny stan* wykonania takiej aplikacji, na wzór informacji o aktualnym stanie zasobów. Obecne usługi informacyjne w Gridzie nie udostępniają takich informacji.
- Istotnym wymogiem dla instrumentacji aplikacji dużej skali jest – obok wymagań wymienionych w pracy – jej *automatyczność*. Użytkownik nie jest bowiem w stanie ręcznie określać instrumentacji dla setek zadań aplikacji. Konieczna jest analiza przyjętego w pracy podejścia do instrumentacji pod kątem jego przydatności do instrumentacji automatycznej.

Bibliografia

[Astalos2008] J. Astalos, L. Flis, M. Radecki, and W. Ziajka. Performance Improvements to BDII – Grid Information Service in EGEE. In M. Bubak, M. Turala, and K. Wiatr, editors, Proc. Cracow Grid Workshop ’07, pages 398–404, Krakow, Poland, 2008. ACC CYFRONET AGH.

[Balis2006] B. Balis, M. Bubak, and K. Guzy. Fine-Grained Instrumentation and Monitoring of Legacy Application in a Service-Oriented Environment. In V. N. Alexandrow, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, editors, Computational Science – ICCS 2006. 6th International Conference, Reading, UK, May 28-31, 2006. Proceedings, Part II, volume 3992 of Lecture Notes in Computer Science, pages 542–548, Reading, UK, 2006. Springer.

[Balis2007] B. Balis, M. Bubak, and J. Wach. User-Oriented Querying over Repositories of Data and Provenance. In G. Fox, K. Chiu, and R. Buyya, editors, Third IEEE International Conference on e-Science and Grid Computing, e-Science 2007, Bangalore, India, 10-13 December 2007, pages 77–84. IEEE Computer Society, 2007.

[Berger2008] M. Berger, T. Zangerl, and T. Fahringer. Analysis of Overhead and Waiting Times in the EGEE Production Grid. In Cracow Grid Workshop ’08, Krakow, Poland, 2008.

[Bubak2008] M. Bubak, T. Gubala, M. Malawski, B. Balis, W. Funika, T. Bartynski, E. Ciepiela, D. Harezlak, M. Kasztelnik, J. Kocot, D. Król, P. Nowakowski, M. Pelczar, J. Wach, M. Assel, and A. Tirado-Ramos. Virtual Laboratory for Development and Execution of Biomedical Collaborative Applications. In Proceedings of the Twenty-First IEEE International Symposium on Computer-Based Medical Systems, June 17-19, 2008, Jyväskylä, Finland, pages 373–378. IEEE Computer Society, 2008.

[Bubak2007] M. Bubak, P. Nowakowski, and S. Unger. K-Wf Grid: Knowledge-Based Workflow System for Grid Applications. In M. Bubak and S. Unger, editors, K-Wf Grid. The Knowledge-based Workflow System for Grid Applications, pages 1–12. ACC CYFRONET AGH, Krakow, Poland, 2007.

[Cooke2003] A. Cooke, A. Gray, et al. R-GMA: An Information Integration System for Grid Monitoring. In Proc. Tenth International Conference on Cooperative Information Systems, volume 2888 of Lecture Notes in Computer Science, pages 462–481. Springer, 2003.

[Decandia2007] G. Decandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon’s Highly Available Key-value Store. In SOSP ’07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, pages 205–220, New York, NY, USA, 2007. ACM Press.

[Foster2003] I. Foster and C. Kesselman (eds). The Grid 2: Blueprint for a New Computing Infrastructure, pages 319–351. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[Foster2006] I. Foster and C. Kesselman. Scaling System-Level Science: Scientific Exploration and IT Implications. Computer, 39(11):31–39, 2006.

[Gil2007] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the Challenges of Scientific Workflows. Computer, 40(12):24–32, 2007.

[Gunter2003] D. Gunter and J. Magowan. An Analysis of “Top N” Event Descriptions. Technical Report GGFDAMED-01-I, Global Grid Forum, April 2003.

[Hendler2004] J. Hendler and D. De Roure. E-Science: the Grid and the SemanticWeb. IEEE Intelligent Systems, 19(1):65–71, January 2004.

[Ludaescher2006] B. Ludäscher. Scientific Workflows: Towards a New Synthesis for Information Integration. In Workshop on Information Integration (II-WS'06), University of Pennsylvania, 2006. http://db.cis.upenn.edu/iiworkshop/postworkshop/positionPapers/130.pdf.

[Ludaescher2005] B. Ludäscher and C. A. Goble. Guest Editors' Introduction to the Special Section on Scientific Workflows. SIGMOD Record, 34(3):3–4, 2005.

[Rhea2005] S. Rhea, B.-G. Chun, J. Kubiatowicz, and S. Shenker. Fixing the embarrassing slowness of OpenDHT on PlanetLab. In WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems, pages 5–5, Berkeley, CA, USA, 2005. USENIX Association.

[Schopf2006] J. M. Schopf, I. Raicu, L. Pearlman, et al. Monitoring and discovery in a web services framework: Functionality and performance of Globus Toolkit MDS4. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.

[Seragiotto2004] C. Seragiotto, H.-L. Truong, T. Fahringer, B. Mohr, M. Gerndt, and T. Li. Standardized Intermediate Representation for Fortran, Java, C and C++ Programs. Technical report, Institute for Software Science, University of Vienna, October 2004.

[Vetter2007] J. S. Vetter and D. A. Reed. Real-Time Performance Monitoring, Adaptive Control, and Interactive Steering of Computational Grids. Int. J. High Perform. Comput. Appl., 14(4):357–366, 2000.

[Yu2007] J. Yu and R. Buyya. A Taxonomy of ScientificWorkflow Systems for Grid Computing. In I. Taylor, E. Deelman, D. Gannon, and M. Shields, editors, Workflows for e-Science, pages 44–49. Springer, New York, Secaucus, NJ, USA, 2007.

[Huang2007] Chenxi Huang, P. R. Hobson, G. A. Taylor, and P. Kyberd. A Study of Publish/Subscribe Systems for Real-Time Grid Monitoring. In 21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA, pages 1–8. IEEE, 2007.

[Menasce2004] D. A. Menasce, L.W. Dowdy, and V. A. F. Almeida. Performance by Design: Computer Capacity Planning By Example. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[Moreau2008] L. Moreau et al. The First Provenance Challenge. Concurrency and Computation: Practice and Experience, 20, 5 (Apr. 2008), pp. 409-418, Wiley InterScience, 2008.

[Podhorszki2004] N. Podhorszki, Z. Balaton, and G. Gombas. Monitoring Message-Passing Parallel Applications in the Grid with GRM and Mercury Monitor. In F. F. Rivera, M. Bubak, A. Gomez Tato, and R. Doallo, editors, Proc. First European Across Grids Conference, pages 179–181. Springer, 2004.

[Stein2008] L. D. Stein. Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges. Nature Review Genetics, 9(9):678–688, 2008.

[Wache2001] H. Wache, T. Voegele, T. Visser, H. Stuckenschmidt, H. Schuster, G. Neumann, and S. Huebner. Ontology-based integration of information – a survey of existing approaches. In H. Stuckenschmidt, editor, IJCAI-01 Workshop: Ontologies and Information, pages 108–117, 2001.

Dodatek: Najważniejsze publikacje autora dotyczące tematyki doktoratu (dla każdej pozycji zaznaczono procentowy udział autora w przygotowaniu publikacji).

1. B. Baliś (70%) and M. Bubak. An Ontology Model for Execution Records of Grid Scientific Applications. In The 3rd International Conference on Semantics, Knowledge and Grid, 2008. Accepted.
2. B. Baliś (70%) and M. Bubak. Monitoring Infrastructure for Grid Scientific Workflows. In E. Deelman and I. Taylor, editors, The 3rd Workshop on Workflows in Support of Large-Scale Science, in conjunction with SC 2008, Austin, TX, November 17, 2008, 2008. Accepted.
3. B. Baliś (60%), M. Bubak, and B. Labno. Monitoring of Grid scientific workflows. Scientific Programming, 16(2-3):205–216, 2008.
4. B. Baliś (60%), M. Bubak, and B. Labno. GEMINI: Generic Monitoring Infrastructure for Grid Resources and Applications. In M. Bubak and S. Unger, editors, K-Wf Grid. The Knowledge-based Workflow System for Grid Applications, pages 60–73. ACC CYFRONET AGH, Krakow, Poland, 2007.
5. B. Baliś (60%), M. Bubak, and M. Pelczar. From Monitoring Data to Experiment Information –Monitoring of Grid Scientific Workflows. In G. Fox, K. Chiu, and R. Buyya, editors, Third IEEE International Conference on e-Science and Grid Computing, e-Science 2007, Bangalore, India, 10-13 December 2007, pages 187–194. IEEE Computer Society, 2007.
6. B. Baliś (50%), M. Bubak, and J. Wach. User-Oriented Querying over Repositories of Data and Provenance. In G. Fox, K. Chiu, and R. Buyya, editors, Third IEEE International Conference on e-Science and Grid Computing, e-Science 2007, Bangalore, India, 10-13 December 2007, pages 77–84. IEEE Computer Society, 2007.
7. B. Baliś (60%), M. Bubak, and K. Guzy. Fine-Grained Instrumentation and Monitoring of Legacy Application in a Service-Oriented Environment. In V. N. Alexandrow, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, editors, Computational Science – ICCS 2006. 6th International Conference, Reading, UK, May 28-31, 2006. Proceedings, Part II, volume 3992 of Lecture Notes in Computer Science, pages 542–548, Reading, UK, 2006. Springer.
8. B. Baliś (50%), H.-L. Truong, M. Bubak, T. Fahringer, K. Guzy, and K. Rozkwitalski. An Instrumentation Infrastructure for Grid Workflow Applications. In R. Meersman and Z. Tari et al., editors, On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part II, volume 4276 of Lecture Notes in Computer Science, pages 1305–1314, Montpellier, France, 2006. Springer.
9. B. Baliś (50%), M. Bubak, W. Funika, T. Szepieniec, R. Wismüller, and M. Radecki. Monitoring Grid Applications with Grid-enabled OMIS Monitor. In F. F. Rivera, M. Bubak, A. Gomez Tato, and R. Doallo, editors, Proc. First European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2003. Revised Papers, volume 2970 of Lecture Notes in Computer Science, pages 230–239. Springer, 2004.