AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

Goal-driven Adaptive Monitoring of Dynamic Service Oriented Systems

---

Adaptacyjne monitorowanie dynamicznych systemów zorientowanych na usługi

Autoreferat rozprawy doktorskiej

Marek Psiuk

Promotor:
prof. dr hab. inż. Krzysztof Zieliński, Akademia Górniczo-Hutnicza

Recenzenci:
prof. dr hab. inż. Adam Grzech, Politechnika Wrocławska
dr hab. inż. Krzysztof Cetnarowicz, prof. AGH, Akademia Górniczo-Hutnicza

Kraków, 2014

1. Wprowadzenie

Rozpowszechnienie architektury zorientowanej na usługi – SOA (ang. *Service Oriented Architecture*) umożliwiło realizację coraz bardziej wymagających systemów informatycznych. W kolejnych realizacjach architektury SOA poziom skomplikowania spowodowany mnogością warstw abstrakcji, skala systemu oraz jego dynamika gwałtownie się zwiększały. Monitorowanie złożonych systemów SOA stało się wyzwaniem. Podejścia w których monitorowanie konfigurowane w sposób statyczny obejmuje wszystkie elementy systemu przez cały czas jego działania nie mogły być zastosowane w SOA, gdyż za bardzo wpływały na monitorowany system. Opisywana rozprawa doktorska proponuje innowacyjne podejście do monitorowania systemów SOA. Zaproponowane podejście umożliwia dynamiczne adaptowanie procesu monitorowania do zmian w monitorowanym systemie zgodnie z deklaratywnym opisem celu.

W rozprawie sformułowana została następująca teza:

> *Dynamiczne systemy zorientowane na usługi mogą być wzbogacone o adaptacyjne mechanizmy monitorowania, umożliwiające wykorzystanie deklaratywnego opisu celu do zarządzania procesem monitorowania, który uwzględnia aspekty dynamiczności poprzez dostosowanie poziomu selektywności oraz identyfikację pierwotnej przyczyny anomalii pojawiających się w systemie.*

Zgodnie ze sformułowaną tezą, zaproponowana w rozprawie adaptacja procesu monitorowania obejmuje dwa główne zagadnienia: zwiększenie poziomu selektywności oraz identyfikację źródła anomalii. Zagadnienia wyjaśnione są poniżej.

Zwiększenie poziomu selektywności – ograniczenie procesu monitorowania do tych fragmentów systemu, które są istotne biorąc pod uwagę zadeklarowany cel monitorowania oraz odkryty model systemu.

Identyfikacja źródła anomalii – zwiększenie zakresu monitorowania w sytuacji pojawienia się anomalii w celu zidentyfikowania jej pierwotnej przyczyny.

Podejście do monitorowania przedstawione w ramach rozprawy skupia się przede wszystkim na samych usługach, ale bierze również pod uwagę poziom warstwy pośredniczącej (ang. *middleware*). Nie jest natomiast rozważane monitorowanie systemu operacyjnego na którym warstwa pośrednicząca jest uruchomiona. U podstaw rozprawy leżą dwa istotne założenia. Po pierwsze, zakłada się, że monitorowanie systemów SOA może być zrealizowane za pomocą przechwytywania wywołań usług. Po drugie, przyjmuje się, że usługi zachowują te same niefunkcjonalne własności bez względu na kontekst wywołania.

Główne osiągnięcia rozprawy są następujące.

- Krytyczna analiza podejść do adaptacyjnego monitorowania przedstawianych w literaturze. Analiza identyfikuje szereg ograniczeń, które są istotne w kontekście dynamicznych systemów zorientowanych na usługi.

- Abstrakcyjna koncepcja adaptacyjnego monitorowania, która umożliwia zarządzanie procesem monitorowania za pomocą deklaratywnego opisu celu. Koncepcja następujące trzy warstwy: systemu SOA, pomiarów, sterowania. Każda warstwa jest opisywana przez stosowny meta-model, którego instancja jest wykorzystywana do opisu aktualnego stanu systemu, śledzenia jego zmian oraz dostosowywania zakresu monitorowania.

- Struktura strategii celu monitorowania, która umożliwia opisanie celu w deklaratywny sposób w kontekście wszystkich trzech warstw koncepcji adaptacyjnego monitorowania.

- Realizacja procesu monitorowania, którego działanie jest zarządzane przy użyciu modelu warstwy sterowania zbudowanego z wykorzystaniem teorii sieci Bayesowskich.

- Projekt systemu Dynamicznego Adaptacyjnego Monitorowania (ang. *Dynamic Adaptive Monitoring*), zwanego DAMON, który może być zainstalowany w istniejącym środowisku SOA w celu udostępnienia funkcjonalności kontrolowania i adaptowania monitoringu zgodnie z deklaratywnym opisem celu.

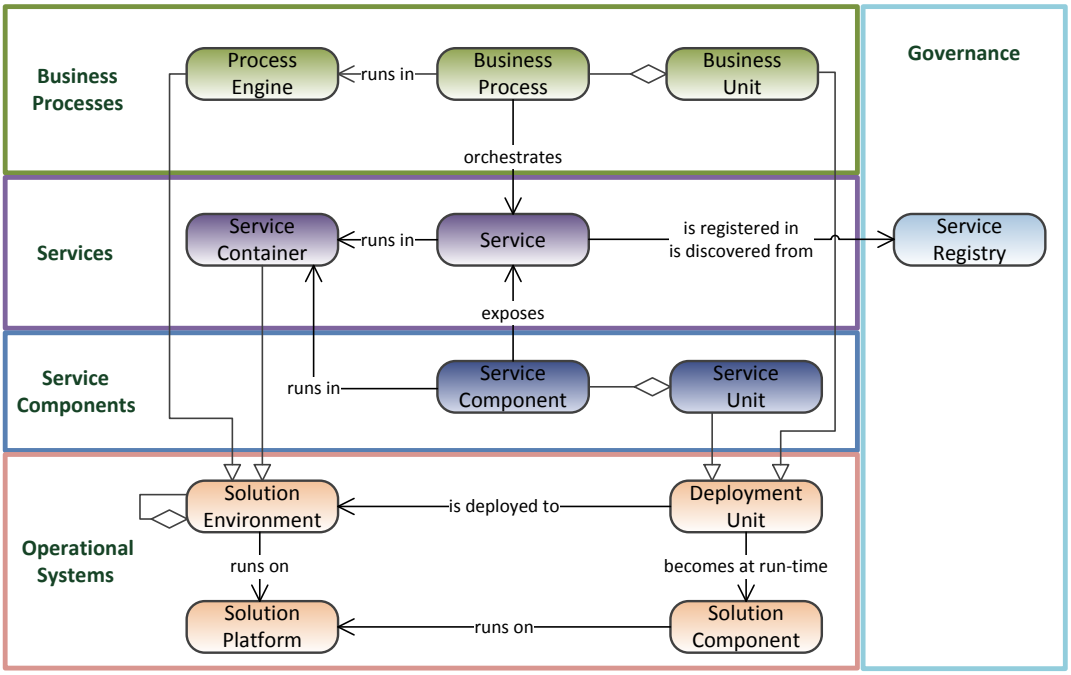
- Prototypowa implementacja systemu DAMON zrealizowana przy wsparciu technologii OSGi. Prototyp obejmuje zarówno mechanizmy monitorowania jak i zaawansowaną warstwę komunikacyjną umożliwiającą integrację mechanizmów z logiką zarządzającą całym procesem.

2. Prace związane z obszarem badań

Obszar badań rozprawy wymagał przeanalizowania szeregu ważnych aspektów, poruszanych w literaturze w mniej lub bardziej przekrojowy sposób. Po pierwsze niezbędne było wybranie odpowiedniego modelu, który wprowadzi właściwe definicje oraz umożliwi zamodelowanie systemu SOA w kontekście technologicznym. Następnie, konieczne było zidentyfikowanie wyzwań, które są istotne dla adaptacyjnego monitorowania. Potrzebny był również szczegółowy przegląd literaturowy, który umożliwił wskazanie niedoskonałości istniejących podejść ujawniających się w kontekście dynamicznych systemów SOA.

2.1. Model architektury zorientowanej na usługi

Opublikowano szereg zróżnicowanych podejść do standaryzacji architektury SOA. Spośród następujących pozycji: The OASIS Reference Model for SOA [1]; The Open Group SOA Ontology [2]; The OASIS Reference Architecture for SOA [3]; The Open Group SOA Governance Framework [4]; The Open Group Service Integration Maturity Model [5]; The OMG SOA Modeling Language [6]; The Open Group SOA Reference Architecture [7] wybrana została ostatnia nazywana skrótowo SOA-RA. Referencyjna architektura SOA-RA okazała się być najbardziej odpowiednia do obszaru badań rozprawy biorąc pod uwagę przyjęty poziom abstrakcji oraz kompletność poruszanych zagadnień.



Rysunek 1: Mapa pojęć przedstawiająca kluczowe elementy specyfikacji SOA-RA

Rysunek 1 przedstawia mapę pojęć [8] obejmującą elementy architektury SOA-RA, które są istotne w kontekście obszaru badań rozprawy. Elementy te obejmują cztery z pięciu warstw horyzontalnych oraz jedną z czterech warstw wertykalnych zdefiniowanych w ramach SOA-RA. Relacje pomiędzy poszczególnymi elementami rysunku są przedstawione za pomocą uproszczonej notacji Unified Modelling Language [9] uwzględniającej asocjację, agregację oraz generalizację. Kluczowym elementem przedstawionej mapy pojęć jest *usługa*, która oferuje pewną dobrze zdefiniowaną funkcjonalność potrzebną do osiągnięcia określonego celu biznesowego. Funkcjonalność usługi jest udostępniana za pośrednictwem *rejestru usług* (ang. *service registry*), w którym następuje rejestracja usługi. Usługa jest realizowana poprzez *komponent usługi* (ang. *service component*), który stanowi implementację jednej lub wielu usług. Rola *warstwy komponentów usług* (ang. *service components layer*) sprowadza się do odpowiedniego odwzorowania kontraktu usługi w implementację. Natomiast rola *warstwy usług* (ang. *services layer*) jest związana z udostępnianiem implementacji usług konsumentom. Rejestr usług jest odpowiedzialny za odkrywanie i ogłaszanie dostępnych usług oraz za umożliwianie wiązania (ang. *binding*) konsumentów do usług w trakcie działania systemu. Rejestr jest zaklasyfikowany do warstwy zarządzania (ang. *governance layer*), która odpowiada za zgodność systemu z SOA z pewnymi przyjętymi politykami, standardami i wytycznymi.

2.2. Wyzwania dla systemów monitorowania

Rozległy przegląd domeny systemów adaptowalnych pozwolił na zidentyfikowanie istotnych wyzwań, które pojawiają się w kontekście monitorowania. Wyzwania zostały zidentyfikowane na podstawie taksonomii adaptacji (ang. Taxonomy of Self-Adaptation) zaproponowanej przez autorów Salehie i Tahvildari [10]. Taksonomia wprowadza szereg różnych kategorii (ang. *facet*) i podkategorii (ang. *sub-facet*). Podkategorią istotną z punktu widzenia rozprawy jest *ciągłe/adaptacyjne monitorowanie* (ang. *continous/a-*

daptive monitoring), które zdefiniowane jest w następujący sposób: „Podkategoria ciągłego/adaptacyjnego monitorowania pozwala na sprecyzowanie tego czy proces monitorowania jest realizowany w sposób ciągły czy też zwyczajowo obejmuje jedynie podzbiór elementów systemu i zwiększa zakres monitorowania w momencie pojawienia się anomalii.”. Opis adaptacyjnego monitorowania w przytoczonym cytacie jest zbliżony do interpretacji, która została przyjęte w rozprawie.

Autorzy Salehie i Tahvildari [10] dokonali przeglądu szesnastu różnych projektów naukowych, które zostały ocenione w kategoriach i podkategoriach zdefiniowanych w ramach wspomnianej taksonomii adaptowalności. W podkategorii ciągłego/adaptacyjnego monitorowania, zwykłe, ciągłe monitorowania było obecne w czternastu projektach. Jedynie dwa projekty, Rainbow [11] oraz KX [12, 13], zawierały mechanizmy wspierające adaptacyjne monitorowania w pewnym ograniczonym zakresie. Taka sytuacja utwierdza w przekonaniu, że aktualny stan nauki w domenie systemów adaptowalnych wymaga zaproponowania pewnych nowych, innowacyjnych podejść. Sformułowany wniosek jest odzwierciedlony w wyzwaniach, które proponują wspomnieni autorzy [10]. Zaproponowane wyzwania zostały podzielone na następujące rodzaje: właściwości *self-** (ang. *self-* properties*), proces adaptacji, kwestie inżynierskie oraz interakcja. Kategoria procesu adaptacji zawiera wyzwania, które są szczególnie istotne dla obszaru badań rozprawy. Są to następujące wyzwania.

Wyzwanie monitorowania – Jednym z istotnych wyzwań związanych z monitorowaniem jest rozwiązanie problemu zbyt dużych narzutów pojawiających się podczas monitorowania rozległych dynamicznych systemów. W systemach monitorowania, często spotyka się sytuacje w których sensory zbierają informacje, które nie są potrzebne do realizacji procesu adaptacji. W takich sytuacjach monitorowanie powoduje niepotrzebne zużycie zasobów. Z kolei w sytuacjach w których zachowanie systemu zaczyna odbiegać od zachowania zdefiniowanego jako „normalne”, zwiększenie ilości zbieranych danych może okazać się nieodzowne. Oreizy wspólnie z innymi autorami [14] twierdzą, że poruszane kwestie są szczególnie istotne w systemach adaptowalnych. Faza planowania pętli sterowania powinna wybierać jedynie te sensory monitorowania, które są wartościowe dla aktualnego stanu procesu adaptacji.

Wyzwanie rozróżnialności stanów – Inne wyzwanie, ściśle związane z poprzednim, sprowadza się do zaproponowania podejścia, które pozwoliłoby na łatwe odróżnienie „poprawnego” (ang. *normal*) działania systemu od działania „niepoprawnego” (ang. *abnormal*). Istniejące podejścia, które wykorzystują zarówno statyczną jak i dynamiczną analizą, nie są w pełni kompletne i wystarczające. W związku z tym, że możliwość odróżniania działania poprawnego od niepoprawnego jest niezbędna do realizacji adaptacyjnego monitorowania, podjęcie wyzwania monitorowania wymaga również podjęcia wyzwania rozróżnialności stanów.

Rozprawa skupia się na zaproponowaniu oraz zweryfikowaniu podejścia, które podejmuje oba opisane wyzwania.

2.3. Ograniczenia istniejących podejść do adaptacyjnego monitorowania

Przegląd literaturowy istniejących podejść do wykrywania anomalii i diagnozowania jak również podejść proponujących kompletne rozwiązania adaptacyjnego monitorowania pozwolił na zidentyfikowanie szeregu istotnych ograniczeń. Podsumowanie tych ograniczeń przedstawione jest poniżej.

- Modele wykorzystywane do reprezentacji systemu nie biorą pod uwagę różnych wzorców komunikacyjnych, które mogą pojawiać się w systemach SOA. Luźne powiązanie (ang. *loose-coupling*), będące jedną z ważnych zasad zorientowania usługowego, jest często realizowane za pomocą komunikacji asynchronicznej. Model systemu SOA tworzony na potrzeby adaptacyjnego monitorowania powinien brać pod uwagą zarówno synchroniczną jak i asynchroniczną komunikację.

- Często, użycie danego modelu wymaga jego wcześniejszej kalibracji w zakresie pewnych istotnych parametrów (np. dostarczanie do sieci Bayesowskiej tabel prawdopodobieństwa warunkowego). Kalibracja musi być najczęściej realizowana ręcznie przez ekspertów domenowych i jest żmudnym, podatnym na błędy procesem. W dynamicznych systemach, w których zmiany wymagają częstej rekalibracji modelu, należy zastosować modele, które nie wymagają kalibracji.

- Większość podejść zaproponowanych w literaturze nie uwzględnia w swoich modelach dwóch ważnych cech systemu: informacji strukturalnej (zwanej topologią) oraz informacji na temat związków przyczynowo-skutkowych pomiędzy elementami systemu.

- Zaproponowane podejścia do sterowania monitorowaniem zgodnie z deklaratywnym opisem celu są albo mocno uproszczone albo sprowadzają się do zdefiniowania zakresu monitorowania bez możliwości wpływania na realizację pętli sterowania.

- Niektóre z proponowanych modeli są właściwe w rozważanym kontekście badawczym, ale nie pozwalają na reprezentowanie hierarchicznych zależności pomiędzy elementami systemu. Taki rodzaj zależności stosunkowo często pojawia się w systemach SOA.

- Funkcjonalność wnioskowania oferowana w kontekście adaptacyjnego monitorowania jest albo stosunkowo prosta albo zbyt mocno skupia się na wskazaniu najbardziej prawdopodobnej przyczyny anomalii wykrytej w systemie.

- Często faza wykonania pętli sterowania ogranicza się do kilku statycznie zdefiniowanych poziomów wśród których najwyższy poziom oznacza monitorowanie całego systemu. Takie podejście nie nadaje się do dynamicznych systemów, gdzie monitoring powinien uwzględniać pojawiające się zmiany bez powodowania niepotrzebnie dużych narzutów.

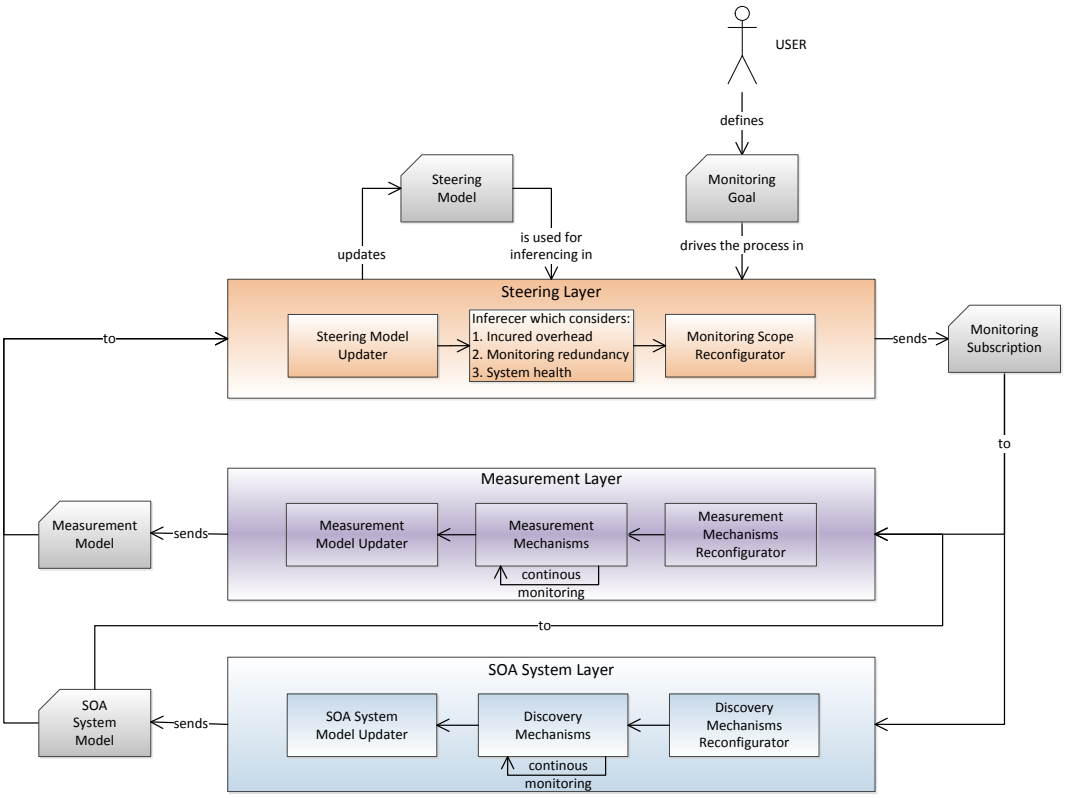
- Proponowane podejścia nie obejmują kompleksowej instrumentacji systemów SOA, która umożliwiłaby: uzyskiwanie danych niezbędnych do odkrywania topologii i re-

lacji przyczynowo-skutkowych; uzyskiwanie danych potrzebnych do realizacji adaptacyjnego procesu monitorowania.

Żadne z podejść znalezionych w literaturze nie było pozbawione wszystkich wskazanych ograniczeń. W związku z tym, rozprawa proponuje nowe podejście do adaptacyjnego monitorowania. Podejście nie posiada żadnego ze zidentyfikowanych ograniczeń i dzięki temu stanowi, w opinii autora, wartościowy wkład do domeny adaptowalnego monitorowania skupiającego się na systemach SOA.

3. Koncepcja adaptacyjnego monitorowania

Rysunek 2 przedstawia całościowe spojrzenie na wysokopoziomową koncepcję adaptacyjnego monitorowania. Koncepcja obejmuje trzy warstwy: *systemu SOA*, *pomiarów* oraz *sterowania*. Każda warstwa związana jest z odpowiadającym jej modelem. Proces monitorowania jest sterowany za pomocą *celu monitorowania* (ang. *monitoring goal*), który jest dostarczony do warstwy sterowania przez użytkownika. Cel monitorowania specyfikuje, co, gdzie oraz w jaki sposób powinno być monitorowane. Warstwy, zdefiniowane w ramach koncepcji, mają podobną konstrukcję. Każda z nich zawiera następujące elementy: *aktualizator modelu* (ang. *model updater*) odpowiadający za tworzenie i aktualizowanie modelu warstwy; mechanizmy, które dostarczają danych potrzebnych do tworzenia/aktualizacji modelu oraz *rekonfigurator* (ang. *reconfigurator*) pozwalający wpływać na wewnętrzną logikę danej warstwy. Mechanizmy warstwy systemu SOA oraz pomiarów nazywane są w rozprawie mechanizmami monitorowania.



Rysunek 2: Wysokopoziomowa koncepcja adaptacyjnego monitorowania

Poziom abstrakcji zaproponowanej koncepcji adaptacyjnego monitorowania pozwala na zrealizowanie każdej warstwy na wiele różnych sposobów. Struktury modeli, które w ramach rozprawy posłużyły do realizacji poszczególnych warstw zostały opisane za pomocą stosownych metamodeli. Metamodele uwzględniają następujące aspekty dynamiczności systemów SOA:

(i) dynamika rejestracji i odkrywania usług;

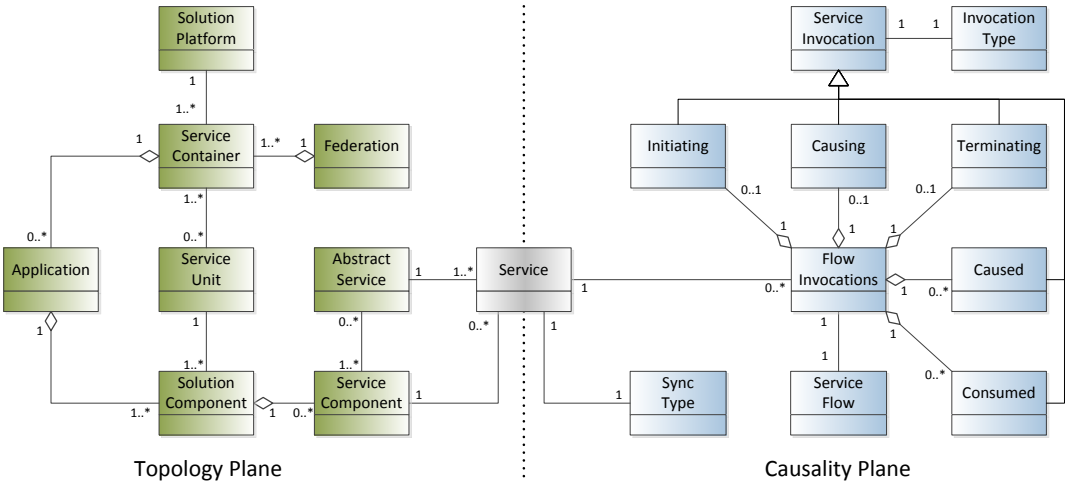
(ii) dynamika rozmieszczenia (ang. *deployment*) komponentów usług;

(iii) adaptowalność usług, komponentów usług oraz środowiska wykonawczego (ang. *operational systems*).

3.1. Metamodel systemu SOA

Rysunek 3 przedstawia metamodel systemu SOA wyrażony za pomocą diagramu klas notacji UML. W związku z tym, że system SOA jest związany z dwoma niezależnymi aspektami: *topologii* (struktura systemu) i *przyczynowości* (zależności pomiędzy elementami systemu związane z wywoływaniem usług), metamodel definiuje budowę tych aspektów za pomocą dwóch niezależnych planów. Instancje tych planów nazywane są odpowiednio modelem topologii oraz modelem przyczynowości. Ich połączenie stanowi instancję modelu systemu SOA. Usługa jest centralnym elementem metamodelu, który jest współdzielony przez oba plany.

Plan przyczynowości wprowadza pojęcie *przepływu usług* (ang. *service flow*), który może być postrzegany jako ślad wywołania (ang. *execution trace*) jakiegoś procesu biznesowego. Mając wiele historycznych śladów wywołania, można wyróżnić wśród nich grupy o izomorficznej strukturze. Każda taka struktura reprezentuje pojedynczy przepływ usług. Zatem dany przepływ usług obejmuje wszystkie usługi, które są pośrednio lub bezpośrednio powiązane zależnościami przyczynowo-skutkowymi. Np. jeśli usługa A wywołuje usługę B, która z kolei wywołuje usługi C oraz D, wówczas wszystkie cztery usługi wchodzą w skład jednego przepływu usług.



Rysunek 3: Metamodel systemu SOA

Celem planu przyczynowości jest opisanie przepływów usług z punktu widzenia pojedynczej usługi. Dzięki takiemu opisowi, wartość konkretnej metryki zmierzona w kontekście danej usługi może być przypisana do zbioru usług tzw. konsumowanych, które są wywoływane przez daną usługę. Takie przypisanie jest niezbędne do stworzenia instancji modelu pomiarów i w konsekwencji modelu sterowania. W ramach modelu przyczynowości, dana usługa posiada instancję klasy *wywołań przepływu* (ang. *flow invocations*) dla każdego przepływu, który obejmuje tę usługę. Klasa wywołań przepływu agreguje wszystkie wywołania związane z daną usługą i danym przepływem. *Wywołanie usługi* (ang. *service invocation*) jest związane z lokalną bądź zdalną komunikacją w wyniku której funkcjonalność usługi jest konsumowana przez inną usługę bądź system zewnętrzny. Wywołanie usługi może mieć jeden lub więcej z następujących typów: *inicjujące* (ang. *initiating*); *kończące* (ang. *terminating*); *konsumowane* (ang. *consumed*); *powodowane* (ang. *caused*); *powodujące* (ang. *causing*). Definicje każdego z typów w kontekście przykładowej usługi (zwanej usługą X) są następujące.

inicjujące – Wywołanie usługi X realizowane przez inną usługę. Wywołanie rozpoczyna działanie logiki usługi X.

kończące – Ostatnie asynchroniczne wywołanie realizowane przez usługę X (wywołania zakańcza działanie logiki usługi X) albo odpowiedź na synchroniczne wywołanie inicjujące.

konsumowane – Synchroniczne wywołanie innej usługi, np. usługi Y, zrealizowane przez usługę X lub asynchroniczne wywołanie usługi X zrealizowane przez inną usługę, np. usługę Z, pod warunkiem, że Z wywołuje X w ramach swojego kończącego wywołania. Jeśli dana usługa A konsumuje usługę B, wówczas relację pomiędzy usługami A i B nazywa się relacją konsumowania.

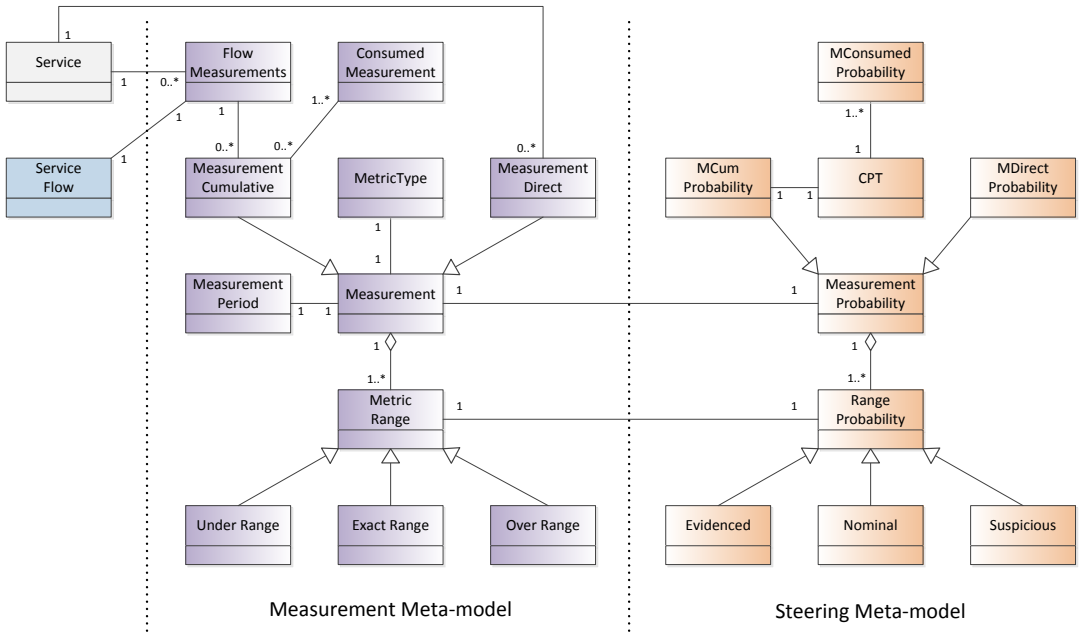
powodowane – Niekończące i niekonsumowane wywołanie realizowane przez usługę X (jedynie wywołania asynchroniczne mogą należeć do tego typu).

powodujące – Najnowsze wywołanie, które pośrednio lub bezpośrednio przyczyniło się do rozpoczęcia działania logiki usługi X i które zostało zrealizowane przez usługę, której logika nie uległa zakończeniu w momencie w którym miało miejsce wywołanie inicjujące X.

3.2. Metamodele pomiarów i sterowania

Metamodele przedstawiające aspekty pomiarów oraz sterowania zaprezentowane są na Rysunku 4. Podobnie do metamodelu systemu SOA, również tutaj zastosowany został diagram klas notacji UML.

Głównym elementem metamodelu pomiarów jest pojęcie *pomiaru*. Pomiar jest zawsze związany z daną usługą i konkretną metryką, która może mieć charakter systemowy (np. czas odpowiedzi usługi, wykorzystany czas procesora) bądź biznesowy (np. trend zmian kursu giełdowego, średnia kwota przelewu w systemie bankowym). Zakłada się, że wartość danej metryki jest niezależna od kontekstu wywołania (np. dana usługa konsumuje 25 MB pamięci bez względu na to przez jaką usługę jest wywoływana). Pomiar



Rysunek 4: Metamodele pomiarów i sterowania

może być w dwóch stanach: *aktywnym* i *nieaktywnym*. Gdy dany pomiar jest nieaktywny oznacza to, że proces monitorowania nie obejmuje pomiaru. Wówczas, pomiar reprezentuje jedynie potencjalne miejsce w systemie, które może być monitorowane. Gdy pomiar jest aktywowany, wartości jego metryki są zbierane przez proces monitorowania i pomiar zaczyna opisywać stan danej metryki usługi poprzez definiowanie zakresu (ang. *metric range*) do którego należą wartości metryki. Aktywowanie pomiaru rozpoczyna generowanie pewnego narzutu, który w mniejszym lub większym stopniu wpływa na monitorowany system. Aktywny pomiar ma znaczenie jedynie w określonym kontekście czasowym – element *okresu pomiaru* (ang. *measurement period*) – w którym to wszystkie zmierzone wartości metryki przynależą do jej zakresu (są większe niż dolna granica zakresu i mniejsze niż górna granica).

Formalna definicja aktywowanego pomiaru (symbol M) obejmuje cztery składowe:

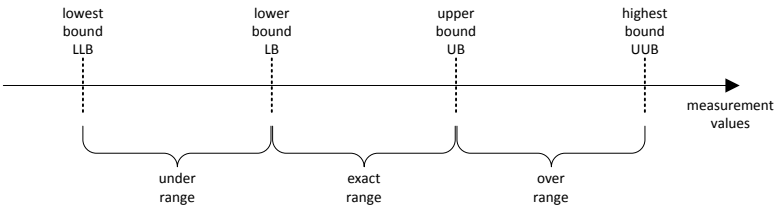
$$M = [LLB, LB, UB, UUB]$$

gdzie:

LLB	–	pewna przyjęta najniższa wartość metryki;
LB	–	dolna granica zakresu do którego należą wartości metryki;
UB	–	górna granica zakresu do którego należą wartości metryki;
UUB	–	pewna przyjęta najwyższa wartość metryki.

Objaśnione cztery składowe pomiaru umożliwiają zdefiniowane możliwych *zakresów metryki* (ang. *metric ranges*): *zakres dolny* (ang. *under range*); *zakres właściwy* (ang. *exact range*); *zakres górny* (ang. *over range*).

zakres dolny	=	$[LLB, LB]$;
zakres właściwy	=	$[LB, UB]$;
zakres górny	=	$[UB, UUB]$.



Rysunek 5: Zależność pomiędzy składowymi pomiaru a możliwymi zakresami metryki

Graficzna reprezentacja zależności pomiędzy składowymi pomiaru a zakresami metryki jest przedstawiona na rysunku 5. Zakres właściwy określa granice wartości metryki podczas gdy zakresy dolny i górny definiują przedziały do których wartości metryki zostaną zaklasyfikowane jeśli granice zakresu właściwego zostaną przekroczone.

Wprowadzono dwa rodzaje pomiarów: *bezpośrednie* (ang. *measurement direct*) oraz *kumulatywne* (ang. *measurement cumulative*) określane odpowiednio jako *mdirect* (symbol MD) oraz *mcum* (symbol MC). Pomiar bezpośredni opisuje wartość metryki danej usługi i nie jest powiązany z żadnym przepływem usług. Pomiar kumulatywny opisuje metrykę danej usługi w kontekście pewnego przepływu usług tzn. opisuje kumulatywną wartość metryki związaną z daną usługą oraz wszystkimi usługami przez nią konsumowanymi. Pomiary usług konsumowanych przez daną usługę nazywane są pomiarami konsumowanymi (ang. *consumed measurement*). Pomiar kumulatywny zawiera wyrażenie matematyczne (ang. *formula*), które na podstawie wartości metryk usług konsumowanych i usługi pomiaru pozwala wyliczyć wartość metryki kumulatywnej.

Kluczowym aspektem metamodelu pomiarów jest wykorzystanie relacji konsumowania wprowadzonej na poziomie modelu przyczynowości. Znajomość tej relacji jest wykorzystywana w konstrukcji pomiarów kumulatywnych. Na potrzeby realizacji modelu sterowania wykorzystana została teoria sieci Bayesowskich (zwanych również skrótowo BN). Sieć Bayesowska pozwala na bezpośrednie reprezentowanie relacji konsumowania. Jeśli dana usługa X konsumuje usługi A, B, C, wówczas w sieci Bayesowskiej węzeł X' może mieć rodziców A', B', C', którzy wpływają na stan węzła X'. Jensen wspólnie z innymi autorami [15] definiują sieć Bayesowską wymieniając jej składowe i ich cechy. Są one następujące.

(i) Zbiór zmiennych losowych oraz zbiór skierowanych krawędzi łączących zmienne.

(ii) Każda zmienna ma zbiór skończonych, wzajemnie się wykluczających stanów.

(iii) Zmienne wraz ze skierowanymi krawędziami tworzę skierowany acykliczny graf (ang. Directed Acyclic Graph – DAG).

(iv) Z każdą zmienną A posiadającą rodziców $B_1, \ldots, B_n$ związana jest tabela prawdopodobieństwa warunkowego (ang. conditional probability table – CPT): $P(A|B_1, \ldots, B_n)$.

Metamodel sterowania został stworzony w oparciu o metamodel pomiarów. Instancja metamodelu sterowania to sieć Bayesowska. Mechanizmy wnioskowania BN pozwalają na osiągnięcie następujących funkcjonalności.

- Identyfikacja pomiarów, które pośrednio dostarczają znaczącej ilości informacji o innych pomiarach. Ilość informacji jest mierzona za pomocą zmiany entropii (tzw. entropii Shannona) [16] w całej sieci w momencie w którym dany pomiar zaczyna być monitorowany.

- Identyfikacja pomiarów, które są anomalne w kontekście pewnych przyjętych kryteriów. Identyfikacja jest realizowana poprzez aktualizację sieci zgodnie z danymi z monitorowania i przeprowadzanie wnioskowania (ang. *inferencing*) w wyniku którego ustalane są prawdopodobieństwa stanów poszczególnych węzłów sieci.

Głównym elementem metamodelu sterowania jest *prawdopodobieństwo pomiaru* (ang. *measurement probability*), zwane również *mprob*. Każdy pomiar ma odpowiadający mu element mprob, który jest zmienną w sieci BN. Stany elementu mprob reprezentują zakresy metryk pomiaru. Każdy zakres jest opisywany przez *prawdopodobieństwo zakresu* (ang. *range probability*) reprezentujące szansę, że wartości metryki pomiaru przynależą do opisywanego zakresu. Element prawdopodobieństwa zakresu może mieć jeden z następujących typów: *znany* (ang. *evidenced*); *nominalny* (ang. *nominal*); *podejrzany* (ang. *suspicious*). Znaczenie każdego typu jest następujące.

Znany – Pomiar elementu mprob jest aktywny i dane z monitorowania potwierdzają, że wszystkie wartości metryki należą do znanego zakresu.

Nominalny – Pomiar był aktywny w przeszłości, ale aktualnie aktywny nie jest. Zakres nominalny to *zakres właściwy* zidentyfikowany w przeszłości, którego aktualne prawdopodobieństwo jest wyższe niż pewna określona wartość.

Podejrzany – Pomiar był aktywny w przeszłości, ale aktualnie aktywny nie jest. Zakres podejrzany to *zakres dolny* lub *górny* z przeszłości, którego aktualne prawdopodobieństwo jest wyższe niż pewna określona wartość.

Metamodel sterowania zawiera dwa rodzaje elementów mprob: *prawdopodobieństwo mdirect*, zwane *mdprob* (symbol MDP) oraz *prawdopodobieństwo mcum*, zwane *mcprob* (symbol MCP). Rodzaje elementów mprob są wprowadzone w celu odzwierciedlenia elementów mdirect i mcum obecnych w metamodelu pomiarów. W modelu sterowania, pomiary konsumowane przez dany pomiar są reprezentowane poprzez element *mconsumed probability*. Krawędzie sieci BN będącej instancją metamodelu pomiarów odpowiadają konstrukcji modelu pomiarów a dokładniej relacjom konsumowania pomiarów przez inne pomiary. Każdy taka relacja pomiędzy dwoma pomiarami odpowiada skierowanej krawędzi łączącej element *mconsumed probability* (rodzic) z jego elementem *mcprob* (dziecko). Elementy *mdprob* są zmiennymi sieci BN, które nie mają żadnych rodziców. W związku z tym nie mają również tabeli CPT. Z kolei elementy *mcprob* mają jednego lub kilku rodziców i w konsekwencji mają również tabelę CPT. Tabele CPT są obliczane za pomocą próbkowania wyrażenia matematycznego elementu *mcum*. Zbiór elementów *mprob* połączonych skierowanymi krawędziami tworzy graf DAG reprezentujący pojedynczą sieć BN. Każda metryka ma osobny zbiór pomiarów, a co za tym idzie, osobną instancję sieci BN.

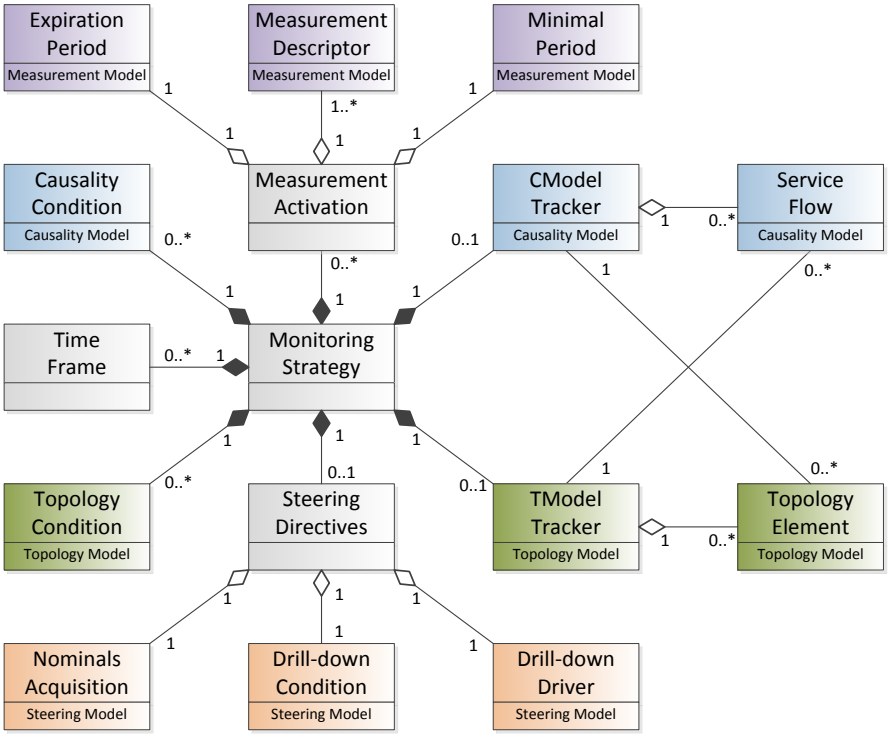
4. Proces monitorowania

Zaproponowana abstrakcyjna koncepcja adaptacyjnego monitorowania nie precyzuje sposobu realizacji procesu monitorowanie. Aby definicja realizacji procesu była kompletna musi obejmować dwa aspekty: opisywanie celu monitorowania oraz zarządzanie procesem adaptacji. Pierwszy aspekt jest uwzględniony w ramach wprowadzonej koncepcji strategii celu monitorowania, natomiast drugi aspekt uwzględniono w zdefiniowanej konstrukcji pętli sterowania.

4.1. Strategia celu monitorowania

Strategia celu monitorowania jest deklaratywnym opisem celu monitorowania. Jej struktura została przedstawiona na rysunku 6 za pomocą diagramu klas UML. Poszczególne elementy strategii nawiązują do modeli koncepcji adaptacyjnego monitorowania co umożliwia na odwoływanie się do następujących aspektów: *gdzie* proces monitorowania powinien być aktywowany – model systemu SOA; *co* powinno być monitorowane – model pomiarów; *jak* proces monitorowania powinien być adaptowany – model sterowania. Innym ważnym aspektem, który został uwzględniony jest kontekst czasowy. Kontekst czasowy precyzuje *kiedy* proces monitorowania powinien się rozpocząć, jak również kiedy powinien się zakończyć. Poszczególne aspekty procesu monitorowania – (i) gdzie, (ii) co, (iii) jak, (iv) kiedy – są opisywane przez następujące elementy strategii:

(i) zbieranie informacji o monitorowanym systemie SOA w zakresie topologi i przyczynowości jest opisywane przez elementy *tmodel tracker* and *cmodel tracker*;



Rysunek 6: Struktura strategii celu monitorowania

(ii) zarządzanie aktywowaniem pomiarów dostępnych w modelu pomiarów jest opisywane przez element *measurement activation*;

(iii) zarządzanie warstwą sterowania w kontekście selektywności i identyfikacji źródeł anomalii jest opisywane przez element *dyrektyw sterowania* (ang. *steering directives*);

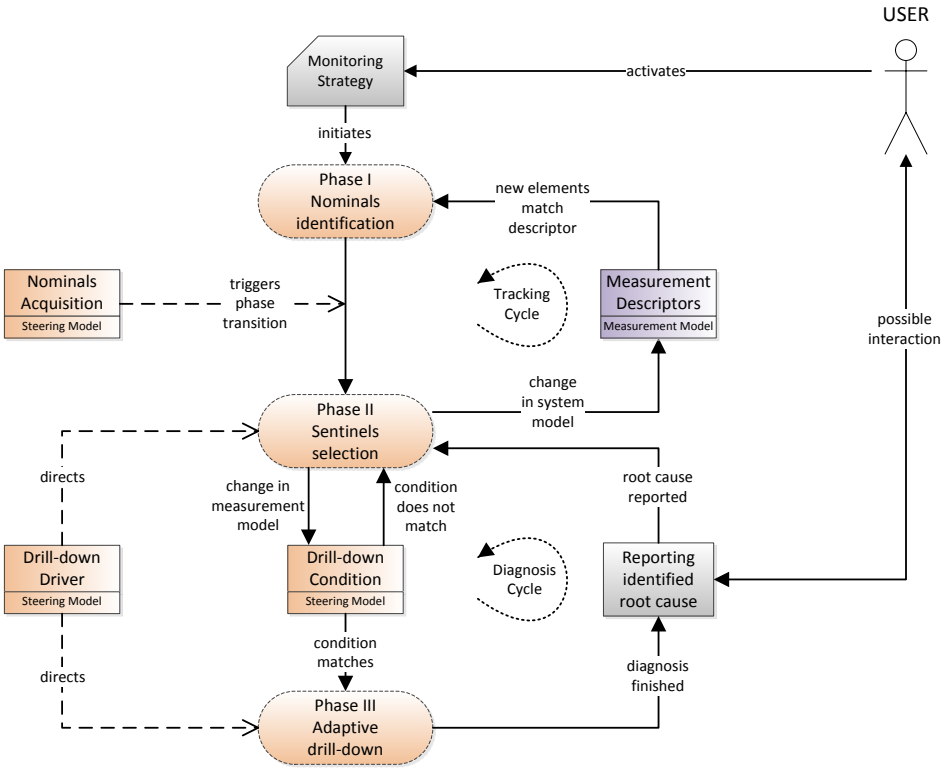
(iv) definiowanie ram czasowych procesu monitorowania opisywane jest przez elementy: *time frame*, *topology condition* oraz *causality condition*.

Strategia może mieć dwa stany: nieaktywny bądź aktywny. Dopiero w momencie gdy dana strategia zostanie aktywowana, system monitorowania uwzględnia ją w realizacji procesu monitorowania.

4.2. Pętla sterowania

Anatomia pętli sterowania jest przedstawiona na rysunku 7. Pętla została podzielona na trzy fazy: *identyfikację pomiarów nominalnych* (ang. *nominals selection*; *wybór strażników* (ang. *sentinels selection*) oraz *adaptacyjny proces wglądu w głąb* (ang. *adaptive drill-down*). Ponadto, w ramach pętli zdefiniowano dwa cykle: *cykl śledzenia* (ang. *tracking cycle* oraz *cykl diagnozowania* (ang *diagnosis cycle*).

Działanie pętli sterowania rozpoczyna się w momencie gdy użytkownik aktywuje strategię monitorowania zawierającą element *dyrektyw sterowania*. Na początku fazy pierwszej, model systemu SOA obejmujący plany topologii i przyczynowości jest odkrywany



Rysunek 7: Pętla sterowania przedstawiona w kontekście wybranych elementów strategii celu monitorowania

zgodnie z wytycznymi strategii. Następnie wszystkie pomiary związane ze strategią są aktywowane i rozpoczyna się identyfikowanie nominalnych zakresów metryk. Zakres metryki jest uznawany za nominalny gdy przez określony okres czasu, mierzone wartości metryki mieszczą się w zakresie. Instancja elementu *nominals acquisition* specyfikuje warunki, które muszą być spełnione aby identyfikacja zakresów nominalnych mogła zostać uznana za skończoną. Po zakończeniu identyfikacji pomiarów nominalnych, następuje utworzenie sieci BN. Pojedyncza sieć BN jest utworzona dla każdego zbioru pomiarów, które są pośrednio bądź bezpośrednio powiązane ze sobą relacją konsumowania. Dla każdej utworzonej sieci BN, logika fazy drugiej i, jeśli zajdzie konieczność, fazy trzeciej jest niezależnie realizowana.

W fazie drugiej, narzuty monitorowania są minimalizowane poprzez dezaktywowanie pomiarów, które nie dostarczają znaczącej ilości informacji o systemie. Jest to zrealizowane za pomocą wyboru zbioru usług tzw. strażniczych (ang. *sentinels*) i dezaktywowanie pomiarów wszystkich usług, które nie są usługami strażniczymi. Nazwa usług strażniczych wynika z tego, że usługi te strzegą nominalnego statu systemu. Po dezaktywowaniu pomiarów wszystkich niestrażniczych usług, system wchodzi o tzw. *okres niezmienności* (ang. *stale period*) i pozostaje w nim dopóki nie nastąpi istotna zmiana w modelu systemu SOA lub modelu pomiarów. W sytuacji zmiany w modelu systemu SOA następuje sprawdzenie czy zmiana ma wpływ na ostateczny zbiór pomiarów związanych ze strategią monitorowania. Jeśli tak, następuje powrót do fazy pierwszej, gdzie zakresy nominalne nowych lub zmienionych elementów systemu są identyfikowane. Po zakończeniu identyfikacji, realizacja fazy drugiej jest kontynuowana. Omówiony cykl, obejmujący fazę drugą i pierwszą, zwany jest cyklem śledzenia, gdyż jego istotą jest śledzenie zmian w monitorowanym systemie. W sytuacji w której zmianie ulegnie model pomiarów, *warunek wglądu w głąb* (ang. *drill-down condition*) zdefiniowany w strategii monitorowania jest sprawdzany. Jeśli w wyniku sprawdzenia okaże się, że różnica pomiędzy modelem pomiarów a sterowania jest większa niż dopuszczalna, następuje przejście do fazy trzeciej.

Celem fazy trzeciej jest zidentyfikowanie pierwotnej przyczyny różnicy pomiędzy modelem pomiarów a sterowania. Różnica jest spowodowana zmianą wartości metryki jakiegoś pomiaru (lub pomiarów). Wartości, które wcześniej nie wychodziły poza zakres nominalny, teraz mieszczą się w innym zakresie, który zwany jest anomalnym (ang. *anomalous range*). Gdyby wszystkie pomiary, których zakres stał się anomalny[1] były aktywne w fazie drugiej (usługi tych pomiarów musiałyby być usługami strażniczymi), identyfikacja przyczyny różnicy byłaby bardzo prosta. Niestety monitorowanie wszystkich usług w okresie niezmienności w których potencjalnie może pojawić się anomalia powodowałby zbyt duży narzut monitorowania. Aby umożliwić identyfikację przyczyny anomalii bez wprowadzania znaczącego narzutu, zaproponowany został proces adaptacyjnego wglądu w głąb. Koncepcja procesu została opracowana w oparciu o wcześniejszą pracę autora rozprawy [17].

Proces wglądu w głąb wykorzystuje wnioskowanie w sieciach Bayesowskich do podejmowania najbardziej właściwych kroków zmierzających do identyfikacji anomalii. Wybór kroków zależy od elementu *sterującego procesem wglądu w głąb* (ang. *drill-down dri-*

[1] Takie pomiary nazywane są pomiarami anomalnymi.

ver) strategii monitorowania a w szczególności od *funkcji fitness* (ang. *fitness function*), która umożliwia jakościową ocenę możliwych kroków. Ocena uwzględnia dwie kwestie: narzuty monitorowania oraz niepewność dotyczącą elementów systemu, które nie są monitorowane. Niepewność jest reprezentowana za pomocą entropii. Gdy identyfikacja przyczyny anomalii zostanie zakończona, diagnoza jest zwracana do użytkownika. Użytkownik może zdecydować czy strategia monitorowania powinna być nadal aktywna, czy też należy ją dezaktywować i rozwiązać problem zidentyfikowanej przyczyny anomalii. Jeśli użytkownik nie podejmie żadnej decyzji lub zdecyduje, że strategia powinna być nadal aktywna, następuje powrót do fazy drugiej (realizacja cyklu diagnozowania). Zidentyfikowane anomalne zakresy stają się nowymi zakresami nominalnymi i jeszcze raz wybierane są usługi strażnicze.

Przedstawiona koncepcja pętli sterowania jest opracowana w oparciu o pewne założenia dotyczące mechanizmów monitorowania. Założenia są następujące:

(a) Kumulatywne wartości metryki pomiaru są zawsze propagowane wraz z *wywołaniem kończącym* bez względu na to czy pomiar jest aktywny czy nie. Dzięki temu mechanizm monitorowania związany z daną usługą jest zawsze świadomy wartości metryk w konsumowanych usługach.

(b) Mierzenie wartości metryki i propagowanie jej wraz z *wywołaniem kończącym* nie powoduje istotnych narzutów monitorowania.

5. System DAMON

System Dynamicznego Adaptacyjnego Monitorowania (DAMON) został opracowany w oparciu o koncepcję adaptacyjnego monitorowania oraz zaproponowaną realizację procesu monitorowania. System DAMON może być zainstalowany w istniejącym systemie SOA celem udostępnienia funkcjonalności kontrolowania i adaptowania monitoringu zgodnie z deklaratywnym opisem celu. DAMON obejmuje wszystkie niezbędne elementy, począwszy od instrumentacji, poprzez mechanizmy monitorowania a skończywszy na całościowej architekturze. W celu przeprowadzenia badań eksperymentalnych zrealizowano prototypową implementację systemu DAMON. Prototyp wykonano przy wsparciu technologii OSGi [18].

5.1. Koncepcja gniazda interceptorów

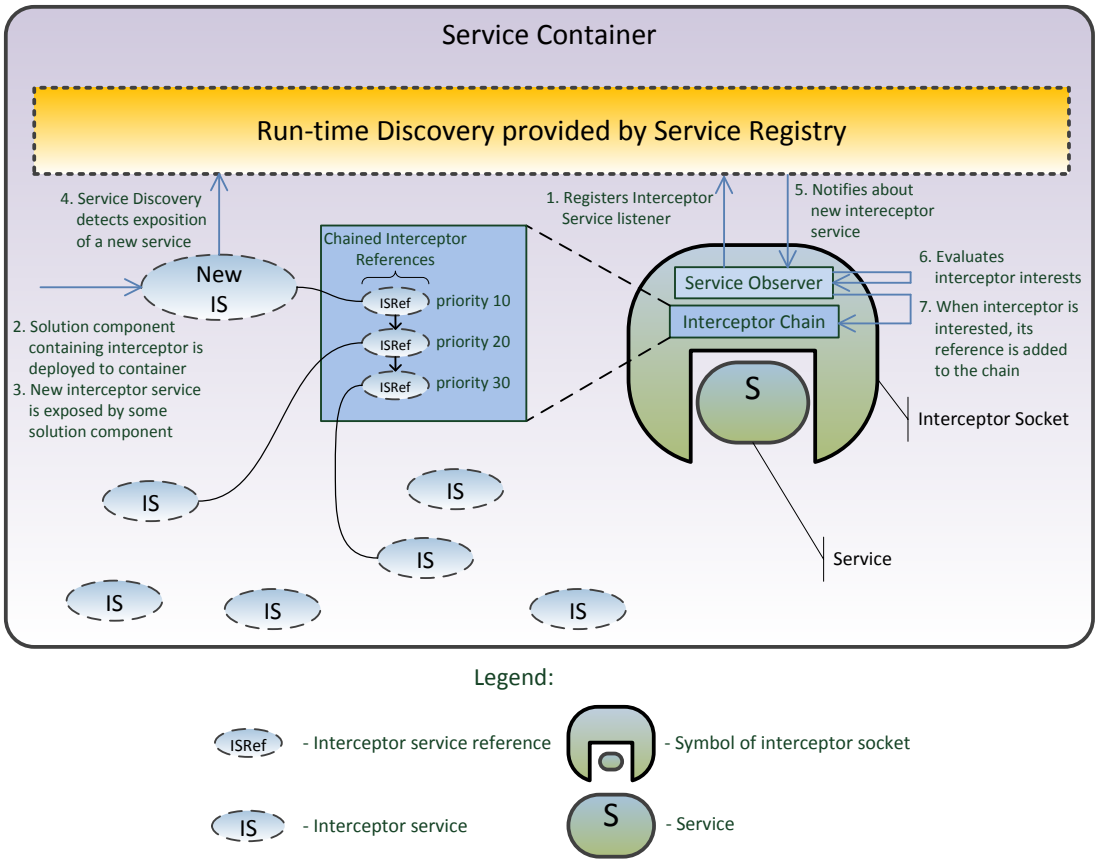
Gniazdo interceptorów (ang. *interceptor socket*) [19], zwane również *i-gniazdem* jest elementem, który opakowuje istniejącą usługę i przechwytuje wszystkich wywołania przychodzące do i wychodzące z usługi. Koncepcja gniazda interceptorów została opracowana na podstawie, znanego w literaturze, wzorca interceptora [20] w taki sposób aby uwzględnić specyfikę dynamicznych systemów SOA. Głównym założeniem tej koncepcji jest zrealizowanie samego elementu interceptora jako usługi. Usługa interceptora nazywana jest *i-usługą*. I-usługa może zostać dynamicznie podłączana do gniazda interceptorów dzięki czemu w łatwy sposób można dokonywać rekonfiguracji mechanizmów

monitorowania w trakcie działania systemu. Interakcja pomiędzy i-gniazdem a i-usługą obejmująca podłączenie i-usługi została przedstawiona na rysunku 8.

Gniazdo interceptorów posiada dwa wewnętrzne elementy: *obserwator usług* (ang. *service observer*) oraz *łańcuch interceptorów* (ang. *interceptor chain*). Zakłada się, że i-usługa udostępnia informacje, które specyfikują zainteresowanie w zakresie i-gniazd. Gdy w kontenerze pojawia się nowa i-usługa, obserwator usług i-gniazda jest powiadomiony o tym fakcie za pośrednictwem rejestru usług. Wówczas obserwator usług ewaluuje zainteresowania i-usługi, aby sprawdzić czy jest ona zainteresowana podłączeniem się do i-gniazda. Jeśli zainteresowanie zostaje potwierdzone, referencja do i-usługi zostaje dodana do łańcucha interceptorów. Każde wywołanie przechwycone przez i-gniazdo zostaje przekazane do łańcucha interceptorów, w którym każda i-usługa obsługuje wywołanie w niezależny sposób. Rozróżniane są dwa typy i-usług: *agnostyczne* dostarczające systemowych informacji oraz *biznesowe* dostarczające wysokopoziomowych informacji związanych z logiką biznesową.

Aby umożliwić przesyłanie danych wraz z przechwytywanym wywołaniem, i-gniazdo udostępnia funkcjonalność lokalnego i eksportowanego wzbogacenia (ang. *local enrichment*, *exported enrichment*). Wzbogacenie może być wykorzystywane zarówno przez i-gniazdo jak i przez i-usługi podłączone do gniazda. W przeciwieństwie do wzbogacenia eksportowanego, wzbogacenie lokalne nie jest propagowane poza obszar i-gniazda.

Zgodnie z podejściem opracowanym wcześniej przez autora rozprawy [21], mechanizm gniazda interceptorów jest dodawany do istniejącego kontenera usług poprzez instrumen-



Rysunek 8: Interakcja pomiędzy gniazdem interceptorów a usługami interceptorów

tację w momencie gdy kontener jest uruchamiany. Wymaga to zmodyfikowania skryptów startowych kontenera, ale pozwala uniknąć wszelkich innych ingerencji i umożliwia łatwe wyłączenie instrumentacji.

5.2. Mechanizmy monitorowania

Mechanizmy monitorowania związane są z warstwami systemu SOA oraz pomiarów zaproponowanej koncepcji adaptacyjnego monitorowania. W ramach warstwy systemu SOA wyróżnia się mechanizmy odkrywania topologii oraz identyfikacji przyczynowości, natomiast w ramach warstwy pomiarów jest tylko jeden rodzaj mechanizmów, których celem jest zbieranie danych potrzebnych do tworzenia i uaktualniania modelu pomiarów. Budowa mechanizmów zakłada, że wszystkie dane wysyłane są do tzw. centrum monitorowania (ang. *monitoring center*), w którym następuje odpowiednie ich przetworzenie i udostępnienie wyniku modułom systemu odpowiadającym za zarządzanie procesem monitorowania.

5.2.1. Odkrywanie topologii

Mechanizm odkrywania topologii obejmuje dwa niezależne komponenty: *pobierający* oraz *śledzący*. Celem komponentu *pobierającego* jest ekstrahowanie na żądanie aktualnego stanu topologii, podczas gdy komponent *śledzący* odpowiada za śledzenie topologicznych zmian. Pierwszy komponent jest zrealizowany za pomocą wzorca projektowego mediator [22]. Komponent ten realizuje mediację pomiędzy metamodelem planu topologii a natywnym formatem topologii udostępnianym przez monitorowany system. Konstrukcja komponentu *pobierającego* zakłada, że każdy system SOA ma pewne, co najmniej podstawowe, mechanizmy udostępniające podzbiór informacji topologicznych.

Komponent śledzący jest zrealizowany jako zbiór biznesowych i-usług, które są podłączane do i-gniazd usług związanych z zarządzaniem topologią w danym środowisku. Zakłada się, że każde środowisko SOA musi być wyposażone w usługi dostarczające następujących funkcjonalności: instalacja nowych usług, udostępnianie usług, zarządzanie cyklem życia kontenera oraz elementów w nim zainstalowanych. Każde wywołanie usługi zarządzającej topologią jest przechwytywane przez komponent śledzący i w razie potrzeby (jeśli wymaga tego jedna z aktywowanych strategii monitorowania) stosowne zdarzenie jest asynchronicznie wysyłane do centrum monitorowania.

Współdziałanie komponentów pobierającego i śledzącego umożliwia uzyskanie pełnej wiedzy na temat topologii. Po aktywowaniu strategii monitorowania komponent pobierający udostępnia aktualny stan topologii. Następnie, komponent śledzący rozpoczyna śledzenie wszystkich zmian topologicznych. Dzięki temu model topologii zawsze reprezentuje aktualny stan fragmentu systemu, którego dotyczy strategia monitorowania.

5.2.2. Identyfikacja przyczynowości

Mechanizm identyfikacji przyczynowości jest zrealizowany za pomocą agnostycznych i-usług, które są podłączane do i-gniazd wszystkich usług związanych z aktywowaną stra-

tegią monitorowania. Wykorzystywane i-usługi nazywane są snifferami przepływu (ang. *flow sniffers*). Celem snifferów przepływy jest wykrywanie wszystkich nowych (niewykrytych wcześniej) przepływów usług, identyfikowanie ich szczegółów przyczynowości i raportowanie uzyskanych informacji do centrum monitorowania. Część szczegółów przyczynowości jest identyfikowanych przez i-gniazdo. Są one następujące: typ usługi (synchroniczna/asynchroniczna), rodzaj wywołania (synchroniczne/asynchroniczne), informacje o konsumencie i dostawcy, dwa typy wywołania: inicjujące i kończące. Pozostałe szczegóły obejmujące typy wywołań: konsumowane, powodowane i powodujące są identyfikowane przez sniffer przepływu.

Gdy sniffer przepływu przechwytuje wywołanie na którymś z i-gniazd, przeprowadzany jest proces decyzyjny mający na celu ustalenie czy wywołanie należy do nowego przepływu czy też do jednego z wykrytych już wcześniej. Do przeprowadzenia procesu sniffer wykorzystuje dwa rodzaje informacji: (i) informacja na temat wcześniejszych wywołań, które miały miejsce w ramach przechwyconego przepływu usług; (ii) informacja na temat wywołań przechwyconych dotychczas na danym i-gnieździe. Aby dostarczyć obydwu rodzajów informacji w wiarygodny sposób, przepływ usług jest reprezentowanych poprzez skrót (ang. *hash*) o zdefiniowanym rozmiarze[2], który zapewnia unikalną identyfikację. Aby uzyskać informację (i), skrót jest obliczany przez każdy sniffer przepływu na bazie wcześniejszego skrótu i wykrytych wywołań. Następnie skrót jest przekazywany do następnego sniffera przepływu za pomocą wzbogacenia eksportowanego udostępnianego przez i-gniazdo. W celu zagwarantowania dostępu do informacji (ii) każdy sniffer przepływu jest wyposażony w tablicę mieszającą, w której zapisywane są wszystkie skróty przepływów zaraportowane dotychczas do centrum monitorowania.

5.2.3. Zbieranie danych do modelu pomiarów

Funkcjonalność mechanizmów pomiarów jest zrealizowana za pomocą *interceptora pomiarów* (ang. *measurement interceptor*), który nazywany jest *m-interceptorem*. M-interceptor jest i-usługą, która udostępnia następujące funkcjonalności: (i) zbieranie, przetwarzanie i propagowanie wartości metryk potrzebnych do obliczenia kumulatywnych wartości metryk; (ii) identyfikowanie zakresów właściwych pomiarów (mdirect oraz mcum) i narzutów monitorowania a następnie raportowanie tych informacji do centrum monitorowania. Aby uwzględnić fakt, że pomiar może być aktywny bądź nieaktywny, przyjęto, że m-interceptor również może mieć analogiczne dwa stany. Bezpośrednio po zainstalowaniu, m-interceptor jest w stanie nieaktywnym i wówczas realizuje tylko funkcjonalność (i). Po aktywowaniu m-interceptora rozpoczyna się realizowanie zarówno funkcjonalności (i) jak i (ii).

Zaproponowana realizacja procesu monitorowania formułuje dwa założenia związane z mechanizmami monitorowania. Zaprezentowany projekt wszystkich trzech rodzajów mechanizmów spełnia te założenia pod warunkiem, że aktywowanie pomiaru zostanie odpowiednio odwzorowane w aktywowanie m-interceptora. Przyjęto następujące odwzorowanie. Aktywowanie pomiaru związanego z metryką X oraz usługą Y jest odwzorowane w aktywowanie m-interceptora metryki X, który przechwytuje wywołania usługi Y. Na-

[2]Skrót może być obliczony przy pomocy takich funkcji skrótu jak SHA-1 czy też MD-5.

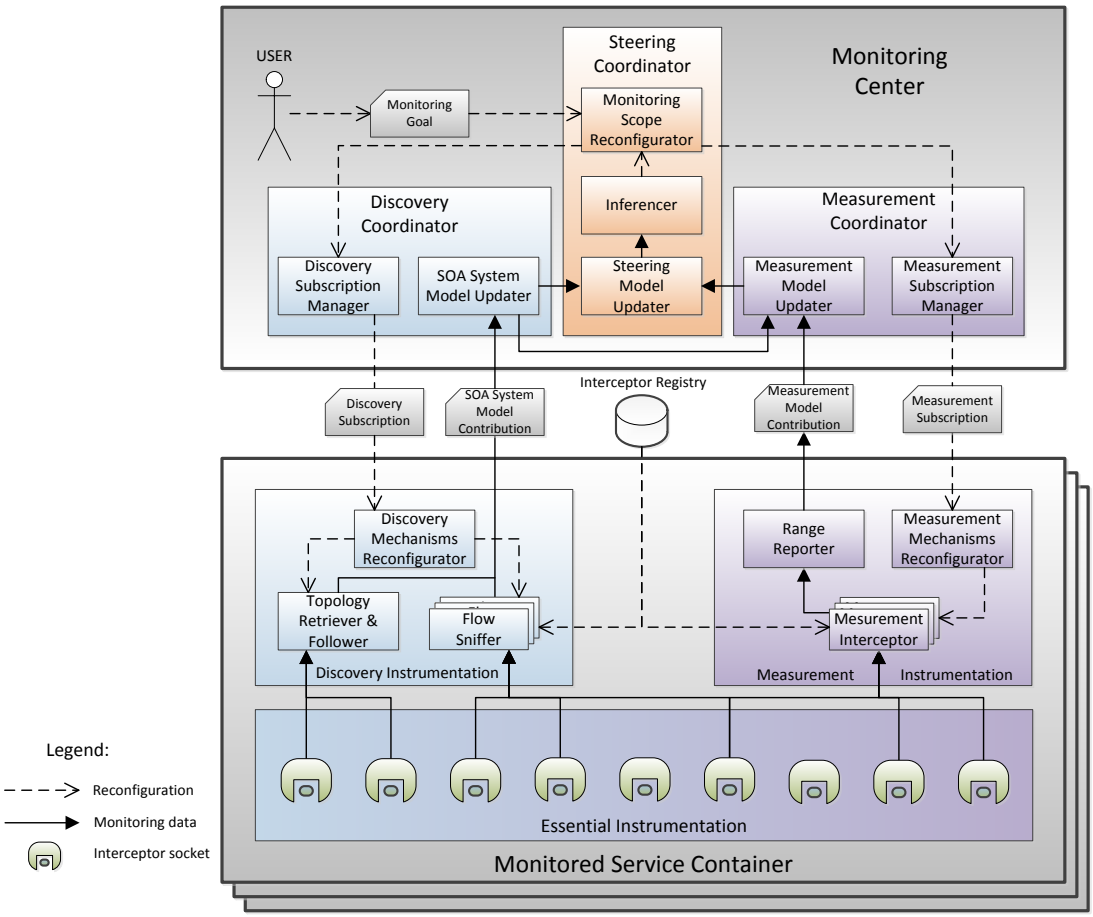
leży zwrócić uwagę, że taki m-interceptor monitoruje nie tylko dany aktywowany pomiar, ale również wszystkie inne pomiary (bezpośrednie, kumulatywne), które są związane z metryką X i usługą Y. Spełnienie pierwszego założenia, które mówi o propagowaniu wartości monitorowanej metryki wraz z *wywołaniem kończącym* jest zagwarantowane przez realizację funkcjonalności (i) m-interceptora wykorzystującej wzbogacenie eksportowane i-gniazda do przesyłania wartości kumulatywnych metryk. Spełnienie drugiego założenia, które mówi o unikaniu narzutów monitorowania jest znów związane z funkcjonalnością (i), której algorytmy oparto są na prostych operacjach (wyrażenia warunkowe, arytmetyczne, wykorzystywanie prostych struktur danych) niepowodujących znaczącego zużycia procesora, pasma sieciowego czy też pamięci (ang. *random-access memory – RAM*).

5.3. Architektura systemu

Architektura systemu DAMON jest przedstawiona na rysunku 9. Architektura łączy ze sobą wszystkie wcześniej wprowadzone elementy: koncepcję adaptacyjnego monitorowania, realizację procesu monitorowania, podejście do instrumentacji oraz mechanizmy monitorowania. Komponenty i grupy komponentów związane poszczególnymi warstwami koncepcji adaptacyjnego monitorowania są zaznaczone odpowiednimi kolorami (analogicznie do ich wcześniejszego użycia). Niektóre elementy architektury bezpośrednio odpowiadają elementom koncepcji adaptacyjnego monitorowania (np. *aktualizatory* modeli czy *rekonfiguratory*), podczas gdy pozostałe uszczegóławiają koncepcję i obejmuję mechanizmy monitorowania.

Zaproponowana architektura wprowadza dwa główne elementy: *centrum monitorowania* (ang. *monitoring center*) – symbol M_{HQ} oraz *monitorowany kontener* (ang. *monitored service container*) – symbol M_{SC}. M_{SC} jest kontenerem usług z metamodelu systemu SOA, do którego, za pomocą instrumentacji, dodany został mechanizm gniazda interceptorów. W kontenerze M_{SC} zainstalowano również mechanizmy odkrywania (warstwa systemu SOA) oraz pomiarów (warstwa pomiarów). M_{HQ} jest również kontenerem usług. Celem kontenera M_{HQ} jest dostarczenie środowiska dla komponentów związanych ze zbieraniem danych monitorowania i zarządzaniem procesem monitorowania. Zaproponowana architektura umożliwia monitorowanie całych federacji kontenerów usług. Aby to zrealizować, wszystkie kontenery federacji muszą zostać zinstrumentowane mechanizmem gniazda interceptorów. Ponadto jeden kontener, wybrany spośród istniejących lub zainstalowany dodatkowo, musi pełnić rolę centrum monitorowania.

Poza dwoma głównymi elementami systemu DAMON i ich komponentami, rysunek 9 przedstawia również przepływy informacji. Akwizycja danych monitorowania z M_{SC} i przesyłanie ich do M_{HQ} jest reprezentowany przez zwykłe strzałki, podczas gdy interakcje związane z rekonfiguracją inicjowaną przez M_{HQ} są reprezentowane przez strzałki o liniach przerywanych. Architektura systemu DAMON zakłada, że przesyłanie danych pomiędzy M_{SC} i M_{HQ} może być zrealizowane za pomocą mechanizmów komunikacji rozproszonej, które wykorzystywane są w danym środowisku SOA do komunikacji pomiędzy kontenerami.



Rysunek 9: Architektura systemu DAMON

5.4. Prototypowa implementacja

Prototypowa implementacja systemu DAMON, stworzona na potrzeby badań eksperymentalnych podejścia do adaptacyjnego monitorowania proponowanego w rozprawie, została zrealizowana przy użyciu technologii OSGi [18]. Prototyp jest zgodny z architekturą systemu co pozwala na realizację procesu monitorowania zgodnie z zaproponowaną pętlą sterownia.

Spośród wielu dostępnych środowisk SOA, wybrano technologię OSGi, która jest standaryzowana przez szereg specyfikacji. Wybrana technologia została uznana za najbardziej odpowiednią do realizacji prototypu, biorąc pod uwagę kryterium uniwersalności oraz wsparcie dla aspektów dynamiczności. Elementy technologii OSGi zostały odwzorowane w meta-model systemu SOA. Odwzorowanie jest dowodem na to, że zaproponowany meta-model nadaje się do modelowania rzeczywistych scenariuszy, mających miejsce w systemach SOA. Dzięki implementacji architektury systemu DAMON zgodnie ze stworzonym odwzorowaniem, prototyp umożliwia monitorowanie federacji rozproszonych kontenerów OSGi.

Szczegóły implementacyjne prototypu obejmują zarówno główne elementy strukturalne takie jak instrumentacja monitorowanego kontenera czy centrum monitorowania, jak i warstwę komunikacji, która łączy te elementy ze sobą. Warstwa komunikacji została stworzona w oparciu o wcześniejszą pracę autora rozprawy [23]. Warstwa umożliwia przesyłanie danych z monitorowania oraz pozwala na wysyłanie żądań zmiany zakresu

monitorowania, które powodują dynamiczną rekonfigurację instrumentacji monitorowanych kontenerów.

Wzbogacenie monitorowanego kontenera obejmuje aspekty instrumentacji oraz mechanizmy monitorowania. Instrumentacja została zrealizowana za pomocą programowania aspektowego (ang. *aspect oriented programming* – AOP) z wykorzystaniem biblioteki AspectJ [24]. Gniazdo interceptorów zostało zaimplementowane jako Java Proxy, które jest wstrzykiwane do wszystkich usług poprzez AspectJ. Implementacja mechanizmów odkrywania obejmuje moduły (ang. *bundles*) *agenta topologii* (ang. *topology agent*) oraz *sniffera osgi* (ang. *osgi-sniffer*). Agent topologii realizuje funkcjonalność komponentów *śledzącego* oraz *pobierającego* korzystając z i-gniazd oraz następujących elementów kontenera OSGi: *BundleContext*, *BundleListener* oraz *ServiceListener*. Implementacja mechanizmów pomiarów obejmuje moduły *agenta zarządzania* (ang. *management agent*) oraz *minterceptorów osgi* (ang. *osgi-minterceptors*). Agent zarządzania wykorzystuje mechanizm *ServiceTracker* kontenera OSGi, który umożliwia wykrywanie zainstalowania/odinstalowania usług subskrypcji monitorowania.

Funkcjonalność centrum monitorowania obejmująca trzech koordynatorów wprowadzonych na poziomie architektury (odkrywania, pomiarów i sterowania), została zaimplementowana w ramach jednego modułu nazwanego *damon-center*. Reprezentacja strategii celu monitorowania została zrealizowana za pomocą języka XML (Extensible Markup Language). Obsługa sieci Bayesowskich została zaimplementowana przy użyciu biblioteki UnBBayes [25] w ramach której wykorzystano jeden z najpopularniejszych algorytmów wnioskowania w sieciach BN – Junction Tree [15].

Zrealizowana implementacja systemu DAMON spełnia główny cel prototypu, jakim jest umożliwienie przeprowadzenia badań eksperymentalnych w zakresie zarówno funkcjonalnym, jak i niefunkcjonalnym.

6. Badania eksperymentalne

W celu zweryfikowania czy zaprojektowany system DAMON może być wykorzystany w rzeczywistych systemach SOA przy jednoczesnym spełnieniu wymagań stawianych w ramach rozprawy, przeprowadzono szereg badań eksperymentalnych. Badania zostały przeprowadzone w zwirtualizowanym środowisku, które symuluje infrastrukturę chmurową systemu bankowego, zrealizowaną przy wsparciu technologii OSGi. W środowisku testowym zainstalowano aplikację CCP (ang. Credit Card Process), która przetwarza zgłoszenia wydania kart kredytowych. Aplikacja jest wykorzystywana do przeprowadzenia szeregu eksperymentów.

6.1. Scenariusz aplikacyjny

Aplikacja CCP została przedstawiona na rysunku 10. Aplikację zrealizowano przy użyciu technologii OSGi. Jej celem jest zautomatyzowanie procesu wydawania kart kredytowych w banku. Aplikacja składa się z siedmiu abstrakcyjnych usług (ang. *abstract service*). Każda abstrakcyjna usługa ma jedną lub wiele instancji. Podczas poprawnego

funkcjonowania, aplikacja CCP obejmuje trzy różne przepływy usług nazwane czerwony, zielony i niebieski. Jak widać na rysunku 10, każdy przepływ obejmuje pewien podzbiór instancji usług aplikacji. Funkcjonalność poszczególnych usług jest następująca.

Credit Card Issuing (CCI) – główna usługa aplikacji, która otrzymuje wnioski o wydanie kart kredytowych z odrębnych modułów infrastruktury bankowej.

Offer Preparation (OP) – usługi odpowiedzialne za zarządzanie przygotowaniem oferty karty kredytowej.

Tax History Retrieval (TH) – usługi pobierające historię podatkową osoby składającej wniosek o kartę kredytową.

Credit History Retrieval (CH) – usługi pobierające historię kredytową osoby składającej wniosek o kartę kredytową.

Offer Calculation (OC) – usługi odpowiedzialne za obliczenie oferty kredytowej w oparciu o historię kredytową i podatkową.

Offer Verification (OVR) – usługa, która przeprowadza weryfikację ofert obliczonych przez usługi OC.

Persistence (P) – usługi odpowiedzialne za przechowywanie przygotowanych ofert w bazie danych.

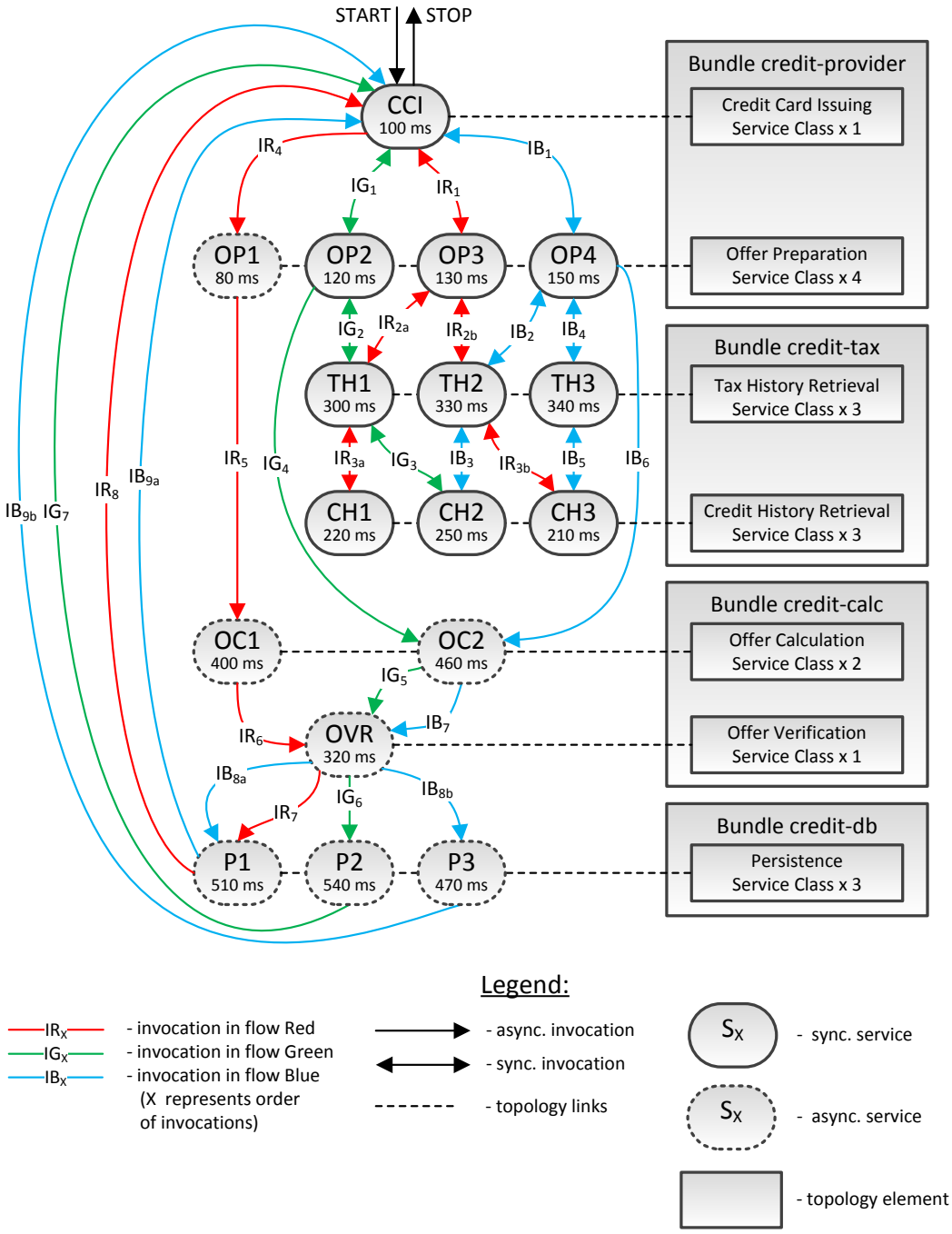
Wszystkie usługi aplikacji CCP mają swoje nominalne czasy odpowiedzi, co zostało ujęte na rysunku 10. W związku z tym, że aplikacja została zaimplementowana w technologii OSGi, każda abstrakcyjna usługa jest reprezentowana przez osobny interfejs a każda instancja usługi przez odpowiednią klasę, która implementuje interfejs usługi abstrakcyjnej. Klasy usług zostały umieszczone w czterech modułach (ang. *bundles*): *credit-provider*, *credit-tax*, *credit-calc* oraz *credit-db*. Aplikacja została zainstalowana w zwirtualizowanym środowisku obejmującym czterech maszynach wirtualnych. Rozmieszczenie poszczególnych modułów było następujące: VM1 – credit-provider; VM2 – credit-tax; VM3 – credit-calc; VM4 – credit-db.

6.2. Eksperymenty

Eksperymenty podzielone są na dwie grupy. Pierwsze trzy eksperymenty skupiają się na aspektach funkcjonalnych i weryfikują ogólną poprawność działania prototypu. Kolejne dwa eksperymenty ewaluują aspekty niefunkcjonalne takie jak narzuty monitorowania oraz aspekty skalowalności.

Eksperymenty funkcjonalne

Pierwszy eksperyment dowiódł, że istniejące środowisko SOA może zostać w łatwy sposób wzbogacone za pośrednictwem systemu DAMON. Eksperyment pokazał również, że odkrywanie topologii oraz śledzenie jej zmian jest zgodne z deklarowanym celem monitorowania. Kolejny eksperyment ujawnił potencjał zarządzania procesem monitorowania



Rysunek 10: Aplikacja CCP (ang. *Credit Card Process*) automatyzująca proces wydawania kart kredytowych w banku

za pomocą deklaratywnego celu. Potencjał został ujawniony dzięki zasymulowaniu dynamicznych zmian związanych z modelami topologii, przyczynowości oraz sterowania. W ramach reakcji na symulowane zmiany, system DAMON automatycznie dostosowywał proces monitorowania zgodnie z zadeklarowaną strategią monitorowania uwalniając administratora od konieczności ciągłego śledzenia rozwoju wypadków w systemie. Ostatni z eksperymentów funkcjonalnych obejmował proces adaptacyjnego wglądu w głąb, który został z powodzeniem zastosowany do wykrywania pojedynczej anomalii, jak również wielu anomalii występujących jednocześnie.

Eksperymenty niefunkcjonalne

Pierwszy eksperyment dotyczący aspektów niefunkcjonalnych pozwolił stwierdzić, że na-

rzuty monitorowania widoczne są głównie w zużyciu pasma sieciowego. Znaczące zwiększenie zużycia procesora czy czasu odpowiedzi monitorowanych usług nie zostało zaobserwowane. Ponadto eksperyment pokazuje, że konstrukcja pętli sterowania oraz jej odpowiednie algorytmy pozwalają znacząco zmniejszyć narzuty monitorowania. Ostatni eksperyment skupia się na aspektach skalowalności i analizuje szereg istotnych własności. Po pierwsze, pokazuje, że ilość usług strażniczych jest liniowo zależna od wielkości sieci Bayesowskiej. Po drugie, dowodzi, że zależność czasu wnioskowania w sieci Bayesowskiej od jej wielkości może być sprowadzona do zależności liniowej dzięki odpowiedniemu zarządzaniu celami monitorowania. Po trzecie, pozwala zaobserwować, że narzut monitorowania jest stosunkowo niewielki w porównaniu do narzutów generowanych przez nieadaptacyjny systemy monitorowania oraz, że wzrastająca dynamika systemu nie wpływa znacząco na zwiększenie narzutów.

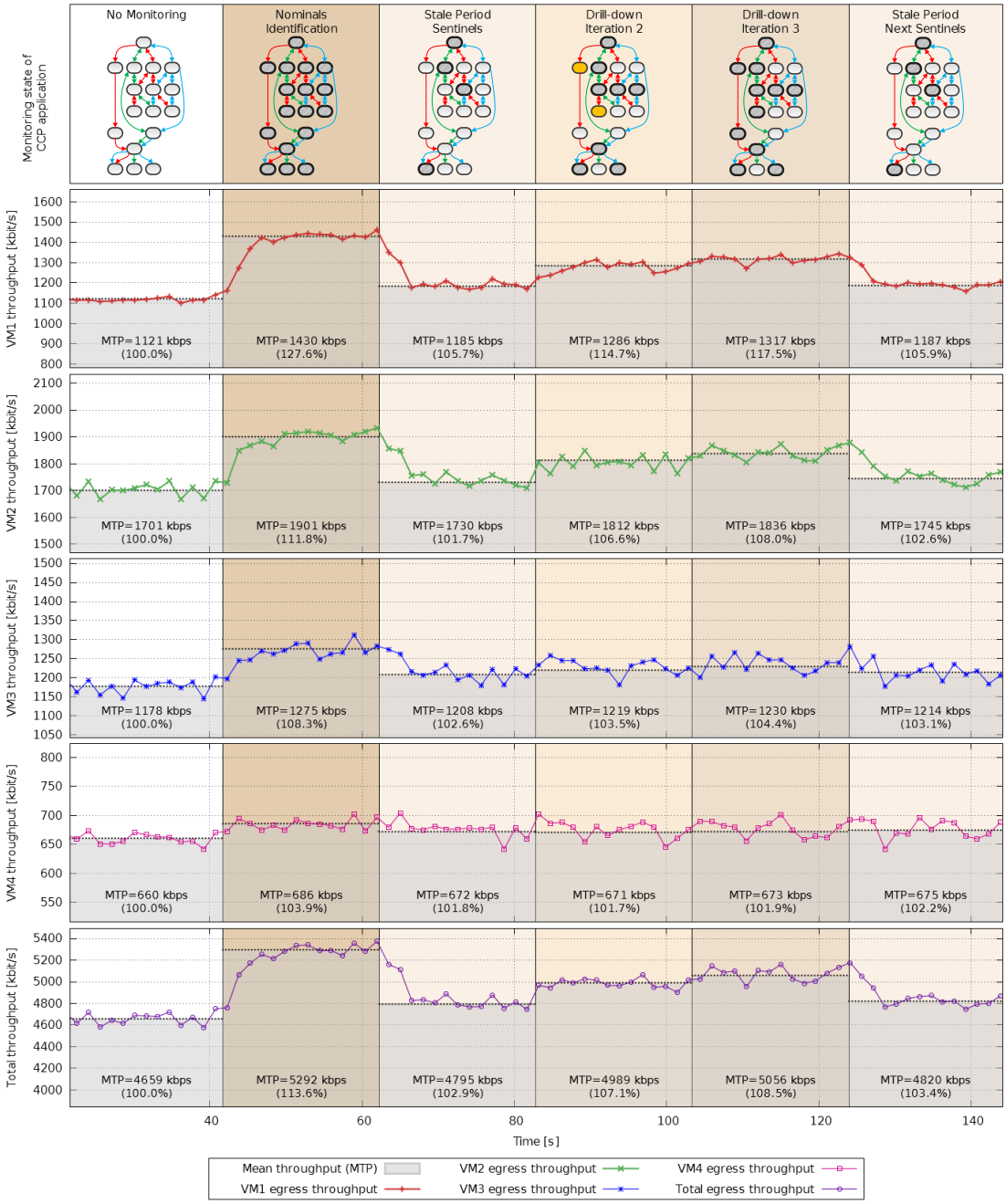
6.3. Wybrane wyniki eksperymentów

W niniejszej sekcji przedstawione są wybrane wyniki badań eksperymentalnych, które zostały uznane za szczególnie wartościowe.

Rysunek 11 przedstawia wykres wpływu realizacji procesu wglądu w głąb na pasmo sieciowe wykorzystywane w środowisku testowym przez poszczególne maszyny wirtualne. Na wykresie widocznych jest sześć różnych przedziałów czasowych, które różnią się konfiguracją mechanizmów monitorowania. Poza samym procesem wglądu w głąb (przedziały 4 i 5) wykres obejmuje również następujące przedziały: monitoring całkowicie wyłączone (przedział 1), identyfikacja pomiarów nominalnych (przedział 2) oraz okresy niezmienności (przedziały 3 i 6). Na górze wykresu, dla każdego z przedziałów czasowych przedstawiony jest schemat aplikacji CCP. Na schematach, kolorem ciemno szarym, zaznaczone są usługi, które były objęte monitorowaniem w danym przedziale czasowym.

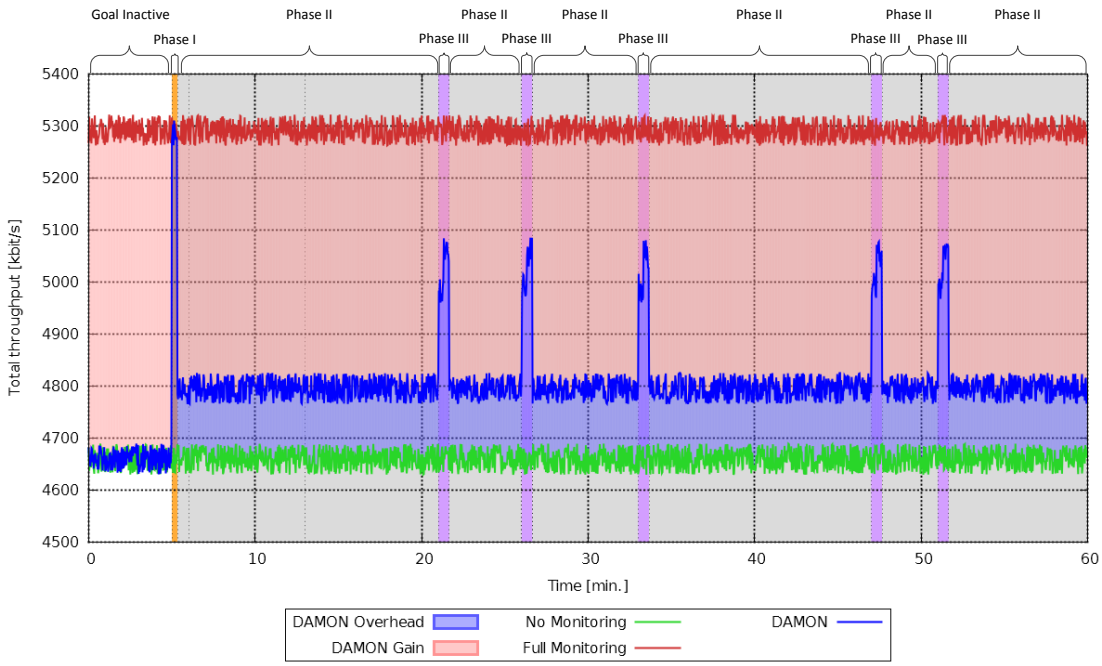
Różnica w zużytym paśmie pomiędzy pierwszymi dwoma przedziałami jest znacząca, co potwierdza założenie, że monitorowanie wszystkich usług przez cały czas powoduje zbyt duże narzuty monitorowania. Jednocześnie widoczne jest, że w okresie niezmienności całkowite zużywane pasmo jest jedynie niecałe 3% wyższe niż w przedziale pierwszym. Zatem ograniczenie monitorowania do usług strażniczych jest w stanie istotnie zmniejszyć narzuty monitorowania. W momencie gdy anomalie pojawiają się w usługach OP1, CH1 oraz TH3, rozpoczyna się działanie procesu wglądu w głąb. Wykres przedstawia dwie iteracje procesu. W każdej iteracji, monitorowanie jest aktywowane dla trzech nowych usług. Jak widać, już po dwóch iteracjach usługi anomalne są zidentyfikowane. Co więcej, narzuty monitorowania podczas procesu wglądu w głąb są mniejsze niż narzuty, które pojawiłyby się podczas monitorowania wszystkich usług (przedział 2). Gdy proces wglądu w głąb uległ zakończeniu, nastąpił powrót do okresu niezmienności, w którym to znów wybrany został ten sam zestaw usług strażniczych: CCI, OP2, TH2 oraz P1.

Wykres przedstawiony na rysunku 11 uwzględniał wpływ monitorowania na przepustowość. Natomiast, w kontekście rozpatrywanego scenariusza aplikacyjnego, właściwe narzuty monitorowania są tak naprawdę związane z dodatkową ilością danych przesłanych przez sieć. Zakłada się, że w ramach symulowanej infrastruktury chmurowej



Rysunek 11: Wpływ realizacji procesu wglądu w głąb na zużywane pasmo sieciowe. Proces jest uruchomiony w sytuacji pojawienia się anomalii w następujących usługach aplikacji CCP: OP1, CH1 oraz TH3

systemu bankowego, przesyłanie danych przez sieć wiąże się z odpowiednimi opłatami (liczonymi od ilości danych). Rysunek 12 przedstawia działanie systemu DAMON w ramach czasowych jednej godziny. Na rysunku zaprezentowano ilość przesyłanych danych jako iloczyn czasu oraz wykorzystanego pasma sieciowego. W uwzględnionych ramach czasowych wystąpiło pięć incydentów anomalii na które system reagował dokładnie tak samo jak w scenariuszu z rysunku 11. Narzuty monitorowania zostały przedstawione na rysunku 12 kolorem niebieskim. Jak widać większość narzutów jest związanych z okresami niezmienności. Narzuty związane z monitorowaniem wszystkich usług pojawiają się jedynie w krótkim przedziale czasowym bezpośrednio po aktywowaniu celu monitorowania. Narzuty związane z procesem wglądu w głąb występują również w relatywnie



Rysunek 12: Działanie systemu DAMON w ramach czasowych jednej godziny podczas której nastąpiło pięć incydentów wystąpienia anomalii

krótkich przedziałach wtedy, gdy anomalie pojawiają się w systemie. Poza narzutami monitorowania, rysunek 12 przedstawia również zysk monitorowania (ang. *monitoring gain*). Przedstawiony zysk jest różnicą pomiędzy ilością danych przesyłanych w sieci podczas monitorowania wszystkich usług przez cały czas a ilością występującą podczas monitorowania przy wsparciu systemu DAMON. Jak widać generowane narzuty monitorowania są relatywnie niewielkie w stosunku do osiąganego zysku.

7. Podsumowanie

Podejście do adaptacyjnego monitorowania dynamicznych systemów zorientowanych na usługi zaprezentowane w ramach rozprawy obejmuje szereg aspektów naukowych. Pierwszym aspektem jest reprezentacja wiedzy na temat struktury monitorowanego systemu oraz relacji pomiędzy jego elementami. Następną istotną kwestią jest sposób wyrażania wysokopoziomowego celu, który powinien zostać uwzględniony przez system monitorowania. Kolejnym aspektem jest schemat logiki odpowiadającej za dostosowywanie zakresu monitorowania. Ostatnim elementem jest odwzorowanie pomiędzy abstrakcyjnymi koncepcjami a konkretną architekturą, która pozwoli na monitorowanie rzeczywistych aplikacji SOA. Wszystkie wymienione aspekty naukowe są wzięte pod uwagę podczas analizy spełnienia wymagań sformułowanych w ramach tezy rozprawy jak również podczas wytyczania kierunków dalszych prac.

7.1. Weryfikacja tezy

Sformułowana teza oczekuje zaproponowania adaptacyjnych mechanizmów monitorowania, umożliwiających wykorzystanie deklaratywnego opisu celu do zarządzania procesem monitorowania, który uwzględnia aspekty dynamiczności poprzez dostosowanie poziomu selektywności oraz identyfikację źródła problemu w sytuacji anomalii.

Opisywana rozprawa przedstawia oryginalną koncepcję adaptacyjnego monitorowania uwzględniającą kwestie zbierania i reprezentowania wiedzy o systemie. W celu reprezentacji wiedzy, koncepcja wprowadza szereg modeli oraz definiuje następujące trzy warstwy: systemu SOA, pomiarów oraz sterowania. Struktura systemu jest przedstawiona za pomocą modelu topologii, natomiast dynamiczne relacje pomiędzy elementami systemu są wyrażone przy wsparciu modelów przyczynowości oraz pomiarów.

Zarządzanie procesem monitorowania jest wspierane przez nowatorską koncepcję wielowymiarowego celu monitorowania, którego instancja jest dostarczana do warstwy sterowania. Warstwa sterowania uwzględnia instancję celu odpowiednio dostosowując zakres działania w zgodzie z tzw. strategią monitorowania. Strategia celu monitorowania jest dokumentem, który specyfikuje co, gdzie i w jaki sposób powinno być monitorowane. Realizacja procesu monitorowania jest podzielona na trzy fazy. Na początku typowe (tzw. nominalne) zachowanie systemu jest odkrywane. Następnie, w fazie drugiej, monitorowanie jest ograniczane do niewielkiego zbioru elementów co pozwala na zmniejszenie narzutów monitorowania przy jednoczesnym ciągłym weryfikowaniu tego, czy zachowanie systemu nie zaczyna odbiegać od odkrytego zachowania nominalnego. Ograniczenie zakresu monitorowania jest utrzymywane do momentu wystąpienia anomalii. Wówczas następuje przejście do fazy trzeciej, która obejmuje realizację adaptacyjnego procesu wglądu w głąb polegającego na sukcesywnym zwiększaniu zakresu monitorowania aż do zidentyfikowania źródła anomalii.

Zaproponowane abstrakcyjne koncepcje mogą mieć wiele potencjalnych realizacji. Rozprawa wykorzystuje teorię sieci Bayesowskich do modelowania monitorowanego systemu i do wybierania zakresu monitorowania, który jest najbardziej właściwy dla danej fazy procesu monitorowania. Sposób wykorzystania sieci Bayesowskich jest związany z szeregiem wymagań w stosunku do mechanizmów monitorowania. Rozprawa prezentuje oryginalny system Dynamicznego Adaptacyjnego Monitorowania (DAMON), który obejmuje szczegółowy projekt mechanizmów monitorowania uwzględniający wszystkie postawione im wymagania. W ramach systemu DAMON zdefiniowana została kompleksowa architektura, która integruje mechanizmy monitorowania z komponentami warstwy sterowania.

Rozbudowane badania eksperymentalne przeprowadzone na prototypie systemu DAMON zrealizowanym przy wsparciu technologii OSGi pozwalają na sformułowanie szeregu wartościowych wniosków. Po pierwsze, DAMON poprawie identyfikuje wprowadzone modele i realizuje wszystkie fazy adaptacyjnego procesu monitorowania w zgodzie z oczekiwaniami. Wykonane eksperymenty ujawniają potencjał deklaratywnego celu monitorowania, którego użyciu umożliwia obsługiwanie dynamicznych zmian w systemach SOA związanych z topologią, przyczynowością i pomiarami. Obsługa jest możliwa dzięki dynamicznej instrumentacji, która jest rekonfigurowana w poszczególnych fazach

procesu monitorowania. Inną zaletę systemu DAMON jest istotne zmniejszanie narzutów monitorowania. Ponadto, generowane narzuty są relatywnie niskie w stosunku do zysku, czyli różnicy pomiędzy zwykłym nieadaptacyjnym monitorowaniem oraz podejściem proponowanym w ramach rozprawy. Eksperymenty dowiodły również, że skalowalność systemu DAMON jest wystarczająca do monitorowania rozległych systemów SOA. Wszystkie sformułowane wnioski pozwalają stwierdzić, że zaproponowane podejście nie tylko spełnia oczekiwania tezy rozprawy, ale również stanowi wkład naukowy do aktualnego stanu domeny adaptacyjnego monitorowania systemów SOA.

7.2. Kierunki dalszego rozwoju

Zaproponowane podejście do adaptacyjnego monitorowania sterowanego deklaratywnym opisem celu może być z powodzeniem użyte w dynamicznych systemach zorientowanych na usługi. Jednakże, jest szereg aspektów, które powinny zostać uwzględnione w ramach przyszłych prac, aby dodatkowo udoskonalić przedstawione podejście. Są one następujące.

- Model sterowania powinien zostać rozszerzony aby umożliwić wykorzystywanie sieci BN, których zmienne nie są dyskretne lecz ciągłe. W ramach proponowanego podejścia dyskretne stany zmiennych są reprezentowane jako zakresy wartości metryk, co wiąże się z utratą części informacji o zachowaniu usługi.
- Zarządzanie dekompozycją celu monitorowania, które jest niezbędne w rozległych systemach SOA powinno zostać zautomatyzowane. Aktualnie dekompozycja musi być przeprowadzana ręcznie.
- Sytuacje konfliktowe, które mogą się pojawić w momencie gdy sieci BN nachodzą na siebie nawzajem powinny być rozwiązywane w bardziej odpowiedni sposób. Rozprawa proponuje uproszczone podejście polegające na rozwiązaniu konfliktów poprzez przypisaniu priorytetów poszczególnym sieciom. Wartościowym usprawnieniem byłoby rozwiązywanie konfliktów poprzez łączenie różnych sieci ze sobą.
- Wnioskowanie oparte o model sterowania mogłoby zostać zrealizowane w oparciu o aproksymacyjną inferencję w sieciach BN (ang. *approximate BN inferencing*), która umożliwiłaby zmniejszenie złożoności obliczeniowej wykorzystywanych algorytmów.
- Algorytmy związane z realizacją pętli sterowania używają prostych heurystyk (zmodyfikowane podejście zachłanne), które mogłyby zostać udoskonalone w taki sposób, aby bezpośrednio uwzględniały modele pomiarów i przyczynowości jak również historyczne dane z monitorowania.

Każdy z wymienionych aspektów związanych z potencjalnymi przyszłymi udoskonaleniami skupiał się na wybranym elemencie zaproponowanego w rozprawie podejścia. Bardziej ogólne udoskonalenie mogłoby zostać osiągnięte poprzez imlementację systemu DAMON w technologii innej niż OSGi. Na przykład implementacja z wykorzystaniem usług webowych (ang. *web services*), bardzo popularnych w istniejących systemach SOA, pozwoliłaby na rozszerzenie zakresu badań eksperymentalnych.

Wybrane pozycje literatury

[1] C. M. MacKenzie, K. Laskey, F. G. McCabe, P. F. Brown, and R. Metz, *Reference Model for Service Oriented Architecture 1.0*, OASIS Standard, October 2006.

[2] The Open Group, "Service-Oriented Architecture Ontology," Tech. Rep., October 2010.

[3] J. A. Estefan, K. Laskey, F. G. McCabe, and D. Thornton, *Reference Architecture for Service Oriented Architecture 1.0*, OASIS Standard, April 2008.

[4] The Open Group, *SOA Governance Framework - Technical Standard*, Thames Tower, 37-45 Station Road, Reading, Berkshire, RG1 1LX, United Kingdom, August 2009, no. C093.

[5] The Open Group, *The Open Group Service Integration Maturity Model (OSIMM), Version 2*, November 2011, no. C117.

[6] Object Management Group, "Service oriented architecture Modeling Language (SoaML) Specification," Tech. Rep., May 2012, version 1.0.1.

[7] "SOA Reference Architecture," The Open Group, Tech. Rep., November 2011.

[8] J. Novak and D. Gowin, *Learning How to Learn.* Cambridge University Press, 1984.

[9] M. Fowler, *Uml Distilled: A Brief Guide To The Standard Object Modeling Language, 3/E.* Pearson Education, 2007.

[10] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, pp. 14:1–14:42, May 2009.

[11] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," *Computer*, vol. 37, pp. 46–54, 2004.

[12] G. E. Kaiser, J. J. Parekh, P. Gross, and G. Valetto, "Kinesthetics eXtreme: An External Infrastructure for Monitoring Distributed Legacy Systems." in *Active Middleware Services.* IEEE Computer Society, 2003, pp. 22–31.

[13] G. Valetto and G. E. Kaiser, "Using Process Technology to Control and Coordinate Software Adaptation." in *ICSE*, L. A. Clarke, L. Dillon, and W. F. Tichy, Eds. IEEE Computer Society, 2003, pp. 262–273.

[14] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An Architecture-Based Approach to Self-Adaptive Software," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 54–62, May 1999.

[15] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, ser. Information science and statistics. New York: Springer, 2007.

[16] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication.* Urbana and Chicago: University of Illinois Press, 1949.

[17] M. Psiuk, T. Bujok, and K. Zielinski, "Enterprise Service Bus Monitoring Framework for SOA Systems," *Services Computing, IEEE Transactions on*, vol. 5, no. 3, pp. 450–466, 2012.

[18] OSGi Alliance, "OSGi Core Release 5," March 2012.

[19] D. Zmuda, M. Psiuk, and K. Zielinski, "Dynamic monitoring framework for the SOA execution environment," *Procedia Computer Science*, vol. 1, no. 1, pp. 125 – 133, 2010, iCCS 2010.

[20] D. C. Schmidt, H. Rohnert, M. Stal, and D. Schultz, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[21] M. Psiuk, "AOP-Based Monitoring Instrumentation of JBI-Compliant ESB," in *Services - I, 2009 World Conference on*, 2009, pp. 570–577.

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[23] M. Psiuk, D. Zmuda, and K. Zielinski, "Distributed OSGi built over message-oriented middleware," *Software: Practice and Experience*, vol. 43, no. 1, pp. 1–31, 2013.

[24] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold, "An Overview of AspectJ," in *ECOOP 2001 - Object-Oriented Programming*, ser. Lecture Notes in Computer Science, J. Knudsen, Ed. Springer Berlin Heidelberg, 2001, vol. 2072, pp. 327–354.

[25] S. Matsumoto, R. N. Carvalho, M. Ladeira, P. Costa, L. Santos, D. Silva, M. Onishi, and E. Machado, "UnBBayes: a Java Framework for Probabilistic Models in AI," in *Java in Academia and Research.* iConcept Press, 2011.