POLITECHNIKA POZNAŃSKA

Odkrywanie modeli procesów w reprezentacji sieci Petriego dla systemów rozproszonych typu CRS

STRESZCZENIE ROZPRAWY

mgr inż. Andrzej Stroiński

Promotor
Prof. dr hab. inż. Jerzy Brzeziński

Instytut Informtyki
Wydział Informatyki

Poznań 2018

Spis treści

1 Wprowadzenie

W dzisiejszych czasach systemy rozproszone obecne są w każdym aspekcie codziennego życia począwszy od darmowych usług takich jak wyszukiwarka gogle, edycja dokumentów w chmurze, aż po skomplikowane systemy takie jak: zarządzanie przedsiębiorstwem, inteligentny transport czy systemy bankowe. Implementacja skomplikowanych systemów rozproszonych, monitorowanie ich działania i zarządzenie nimi jest trudne i kosztowne. W celu usprawnienia procesu wytwarzania oraz później utrzymywania systemów rozproszonych stosuje się *Architekturę Zorientowaną na usługi* (1) (*ang. Service Oriented Architecture, SOA*). SOA to styl architektoniczny do budowania systemów rozproszonych, gdzie funkcjonalność systemu realizowana jest przez niezależne od siebie komponenty systemu zwane usługami (*ang. services*). Każda z usług realizuje jedynie fragment funkcjonalności systemu, który może zostać wykonany na żądanie klienta lub innej usługi. Usługi, w celu realizacji zaawansowanych funkcjonalności, są komponowane w procesy biznesowe (*ang. business processes*).

W praktyce, w budowie systemów SOA stosuje się dwa podejścia: proceduralne lub deklaratywne.

Podejście proceduralne charakteryzuje się wykorzystywaniem protokołu komunikacyjnego SOAP (*ang. Simple Object Access Protocol*), który pozwala na wywoływanie usług (np. zdalne wywoływanie metod) z wykorzystaniem wiadomości opakowywanych za pomocą odpowiednio przygotowanych komunikatów zapisanych w składni XML. Ponadto, usługi systemu opisywane są za pomocą specjalistycznych dokumentów WSDL, które zawierają informacje na temat funkcjonalności realizowanej przez usługę oraz opis jej interfejsu (API). Na bazie tego typu opisów przygotowuje się plan wykonania procesów biznesowych w systemie za pomocą dedykowanego języka np. języka WS-BPEL. Plan ten następnie jest wykonywany przez silniki procesów biznesowych. Historycznie podejście to, oferowało dużą łatwość implementacji, ale obszerny stos protokołów dodany w celu zapewnienia takich własności systemu takich jak bezpieczeństwo czy niezawodność komunikacji spowodował skomplikowanie zależności w systemie i powstanie bardzo dużych narzutów czasowych związanych z parsowaniem obszernych dokumentów XML.

Podejście deklaratywne, tzw. RESTful-WS bazujące na stylu architektonicznym REST (*ang. Representational State Transfer*) zaproponowanym przez R.T. Fieldinga w (2), wprowadza inne lżejsze podejście do implementacji SOA. Ponadto, w celu zapewnienia wydajnej i prostej w użyciu komunikacji opracowano Architekturę zorientowaną na zasoby (*ang. Resource Oriented Architecture*, ROA), która wraz z podejściem REST wykorzystuje protokół HTTP. W ROA funkcjonalność systemu jest zdekomponowana na kolekcję zasobów, które reprezentują istotne elementy systemu. Przykładem zasobu może być np. zmienna całkowitoliczbowa, kolekcja zmiennych będących numerami indeksów studentów, a także kod programu. W celu realizacji funkcjonalności systemu operuje się na zasobach za pomocą określonego zestawu podstawowych operacji obejmujących: tworzenie nowego zasobu, usuwanie zasobu, modyfikacja zasobu oraz pobieranie reprezentacji zasobu.

Uogólnieniem systemów bazujących na podejściu REST, są tzw. systemy komunikujących się zasobów (*ang. Communication Resource System*). Systemy te składają się z kolekcji komunikujących się i hierarchicznie uporządkowanych zasobów. W systemach CRS, każdy z zasobów systemu jest pasywny i podejmuje działanie tj. wykonuje lokalny proces zasobu wyłącznie w efekcie, wywołania przez inny zasób lub klienta systemu. Procesy biznesowe w systemach CRS są szeregiem wywołań odpowiednich zasobów wykonanych w ustalonej kolejności. Przykładami języków pozwalających na definiowanie procesów biznesowych w tym podejściu są: ROsWeL (3) i Joile (4).

Systemy CRS mają długi cykl życia. Z tego względu ich implementacja ma charakter iteracyjny i modularny. Ponadto, w systemach o długim cyklu życia wymagania stawiane systemowi ulegają zmianie w czasie, co prowadzi do modyfikacji działania poszczególnych zasobów. W konsekwencji może spowodować to wystąpienie niepożądanych sytuacji: pętle wywołań, wąskie gardła, zakleszczenia, braki zasobów lub naruszenia własności systemów CRS (REST, ROA), jak m.in. idempotencja metod, czy ich bezpieczeństwo.

Wystąpienie, któregoś z wyżej wymienionych problemów w działaniu systemów CRS może prowadzić do utraty jakości świadczonych usług (czyli obniżenia QoS, *ang. Quality of Service*), a w ekstremalnych przypadkach nawet do niepoprawnego działania systemu lub jego awarii. W ogólności w celu rozwiązania tego typu problemów stosuje się metody weryfikacji modelu (*ang. model checking*) lub badania zgodności (*ang. conformance checking*). Metody te, pozwalają zidentyfikować miejsce wystąpienia błędów w realizowanym procesie, a także zidentyfikować miejsca w których analizowane procesy działają niezgodnie ze swoja specyfikacją. Warunkiem zastosowania tych metod jest dostępność modelu procesu odpowiadającego rzeczywistemu działaniu systemu. Niestety, w praktyce jest to trudno osiągalne. Ze względu na wagę tego zagadnienia dla wydajnego zarządzania i utrzymywania oraz usprawniania systemów trwają intensywne poszukiwania metod pozyskiwania modeli procesów. Jednym z możliwych podejść pozwalających na uzyskanie modelu rzeczywistego procesu realizowanego przez system jest zastosowanie metod eksploracji procesów [5] (*ang. proces mining*) oferujących algorytmy odkrywania modeli procesów (*ang. proces discovery*).

Algorytmy te polegają na odkrywaniu modelu realizowanego procesu na podstawie analizy dziennika zdarzeń. Dziennik zdarzeń to zbiór zdarzeń odpowiadających aktywnościom realizowanym przez określony proces. Każde zdarzenie składa się z atrybutów pozwalających opisać aktywność związaną z danym zdarzeniem. W praktyce metody odkrywania modelu procesu analizują jedynie fragmenty dzienników zdarzeń związane z pojedynczym procesem. Taki fragment nazywany jest logiem procesu i składa się on ze zbioru instancji procesu, czyli sekwencji aktywności realizowanych w ramach pojedynczego wykonania procesu.

Obecnie dostępne są liczne algorytmy odkrywania modeli procesów działających w przedsiębiorstwach, na liniach produkcyjnych jak i w systemach komputerowych (5) (6) (14) (7) (8) (9). Niestety, ze względu na cechy systemów CRS, nie jest możliwe wykorzystanie tych algorytmów w celu odkrycia modelu procesu biznesowego realizowanego w systemach CRS.

Wynika to z faktu, że:

- dzienniki zdarzeń zbierane w systemach CRS nie uwzględniają informacji o realizowanym procesie biznesowym i są zbierane w izolacji od siebie,
- proces biznesowy składa się z wielu kooperujących procesów lokalnych realizowanych przez zasoby systemu, które komunikują się ze sobą za pomocą synchronicznego modelu komunikacji,
- zasoby mogą być od siebie hierarchicznie zależne.

Istnieje więc potrzeba znalezienia metod właściwych dla systemów CRS. W ten nurt prac badawczych wpisuje się niniejsza praca.

2 Cel i zakres pracy

Celem niniejszej pracy jest opracowanie nowych metod odkrywania modeli sieci Petriego dla procesów biznesowych wykonywanych w systemach CRS.

W niniejszej pracy założone zostało, że rzeczywisty model procesu biznesowego realizowanego w systemie CRS może być zamodelowany za pomocą specjalistycznych sieci Petriego: tzw. sieci komunikujących się zasobów i/lub zgodnej i ustrukturyzowanej sieci przepływu (*ang. sound and structured workflow nets*). Ponadto, lokalne modele procesów realizowanych przez pojedyncze zasoby systemu mogą być zamodelowane za pomocą zgodnych i ustrukturyzowanych sieci przepływu.

Dodatkowo założono, że zbierane dzienniki zdarzeń w systemie CRS są kompletnymi dziennikami zdarzeń, co oznacza, że wszystkie możliwe przeploty aktywności możliwe do wystąpienia podczas realizacji procesu biznesowego znajdują swoje odzwierciedlenie w tym dzienniku zdarzeń.

W tak zdefiniowanym środowisku problem badawczy rozprawy został sformułowany następująco:

Znaleźć algorytm (funkcję), oznaczoną jako DA(CRSGPL), która mapuje kompletny log procesu biznesowego (CRSGPL) realizowanego w systemie CRS na:

- zgodną i ustrukturyzowaną sieć przepływu,
- sieć komunikujących się zasobów (*ang. communication net)*, gdzie wszystkie lokalne procesy zasobów są reprezentowane przez zgodne i ustrukturyzowane sieci przepływu oraz
- sieć komunikujących się zasobów z hierarchią zasobów (*ang. communication net with resource hierarchy*), gdzie wszystkie lokalne procesy zasobów są reprezentowane przez zgodne i ustrukturyzowane sieci przepływu.

Ponadto, wynikowe sieci są równoważne rzeczywistym procesom biznesowym realizowanym w systemie CRS.

Powyższy problem badawczy został w niniejszej pracy zdekomponowany na następujące problemy szczegółowe:

- zbieranie dzienników zdarzeń w systemach CRS. Problem ten jest nietrywialny ze względu na charakterystykę systemów CRS, które są systemami składającymi się z dużej liczby asynchronicznych i rozproszonych zasobów, z których każdy niezależnie od pozostałych zbiera dzienniki zdarzeń, w związku z czym ustalenie kolejności wykonywanych aktywności z perspektywy globalnej na poszczególnych zasobach nie jest możliwe.
- rekonstrukcja procesu biznesowego, a dokładniej dziennika zdarzeń procesu biznesowego w CRS na podstawie dziennika zdarzeń możliwego do zebrania w systemie CRS, który składa się z niezależnie zbieranych dzienników poszczególnych zasobów systemu CRS.
- odkrywanie zgodnego i ustrukturyzowanego modelu procesu biznesowego realizowanego w systemie CRS, polegający na znalezieniu algorytmu mapującego kompletny dziennik zdarzeń procesu biznesowego na zgodny i ustrukturyzowany model procesu biznesowego który odpowiada modelowi rzeczywistemu tego procesu.
- odkrywanie perspektywy zasobowej w odkrywanym modelu procesu biznesowego realizowanego w systemie CRS, która polega na jawnym reprezentowaniu w modelu procesu: komunikacji pomiędzy poszczególnymi zasobami, kompozycji zasobów w proces biznesowy oraz hierarchii zasobów.
- rozproszenie obliczeń związanych z odkrywaniem modelu procesu biznesowego wykonywanego w systemach CRS – Problem polega na znalezieniu efektywnej metody rozproszenia obliczeń związanych z odkrywaniem modelu procesu biznesowego uruchamianego w systemie CRS, który umożliwi odkrywanie procesów składających się z wielu niezależnych lokalnych procesów realizowanych przez zasoby systemu.
- weryfikacja wydajnościowa zaproponowanych w pracy metod i algorytmów.

3 Modelowanie procesów biznesowych systemów CRS.

W rozprawie doktorskiej wprowadzono szereg definicji, które opisują najważniejsze terminy używane w pracy oraz własności występujące w systemach CRS. W streszczeniu przestawione zostaną jedynie najważniejsze z nich wraz ze skróconym opisem.

Dziennik zdarzeń – zbiór zdarzeń odpowiadający aktywnościom wykonywanym w ramach realizowanych procesów.

Zdarzenie – pojedynczy wpis w dzienniku zdarzeń, który składa się z listy wartości odpowiadających określonym atrybutom. Możliwymi atrybutami zdarzenia są: identyfikator procesu, identyfikator instancji procesu, nazwa wykonywanej aktywności, znacznik czasowy itd.

Zasób – pojedynczy komponent system CRS, który realizuje jeden proces zwany lokalnym procesem zasobu.

Proces biznesowy – proces realizowany przez wiele zasobów na zlecenie klienta (komponentu zewnętrznego względem systemu CRS).

Ślad procesu – sekwencja zdarzeń związana z pojedynczym wykonaniem (instancją) procesu odpowiadająca sekwencji wykonywanych aktywności w ramach pojedynczego wykonania procesu

Ślad aktywności procesu – sekwencja aktywności związana z pojedynczym wykonaniem (instancją) procesu. W niniejszym streszczeniu ślad aktywności procesu i ślad procesu będą tożsame

Ślad procesu biznesowego – to zbiór śladów procesów lokalnych zasobów uczestniczących po pojedynczym wykonaniu procesu biznesowego

Log procesu – zbiór śladów pojedynczego procesu.

Log procesu biznesowego – zbiór śladów procesu biznesowego dotyczących pojedynczego procesu biznesowego.

W ramach niniejszej pracy wprowadzono nowe typy sieci Petriego tzw. *sieci komunikujących się zasobów* (*ang. communication net*) będące rozszerzeniem istniejących sieci Petriego wykorzystywanych w eksploracji procesów zwanych *sieciami przepływu* (*ang. workflow net*). Sieci komunikujących się zasobów służą do modelowania procesów biznesowych w systemach CRS i charakteryzują się:

- hierarchiczną strukturą – składa się ona ze zbioru sieci przepływu odpowiadających procesom realizowanym przez poszczególne zasoby systemu.
- jawnym modelowaniem komunikacji – modelowaniem kanałów komunikacyjnych między poszczególnymi lokalnymi procesami zasobów

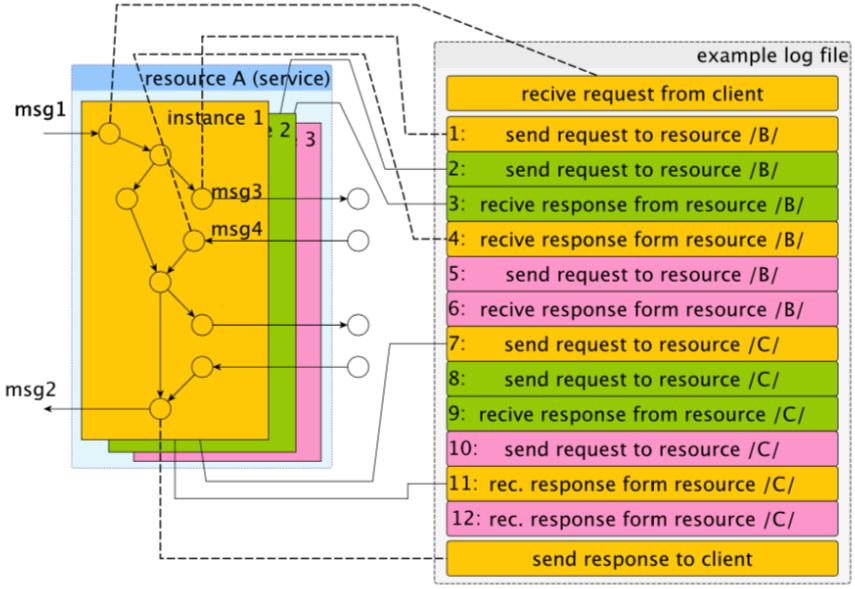
Rozszerzeniem, powyżej omówionego typu sieci Petriego jest sieć komunikujących się zasobów z hierarchią zasobów, która rozszerza definicję sieci komunikujących się zasobów o dodatkowe jawne modelowanie relacji hierarchii pomiędzy zasobami.

Ponadto, zaproponowano kolorowaną sieć komunikujących się zasobów, w celu minimalizacji modelu procesu wykorzystując ideę kolorów (10). Dzięki temu w odróżnieniu do zwyczajnej sieci komunikujących się zasobów modelowanie wielu jednocześnie wykonywanych instancji pojedynczego zasobu nie musi być realizowane przez tworzenie odrębnej sieci przepływu dla każdej z nich. Tutaj w celu odseparowania współbieżnego wykonania tego samego zasobu wykorzystuje się żetony w różnych kolorach.

4 Zbieranie dzienników zdarzeń w systemie CRS i rekonstrukcja procesu biznesowego

Dzienniki zdarzeń zbierane przez zasoby systemu CRS podczas wykonywania procesu biznesowego nie mogą być bezpośrednio wykorzystane do odkrywania modeli procesów biznesowych w CRS ze względu na:

- asynchronizm zasobów – oznacza on, że zasoby nie współdzielą zegara, zatem czas wykonywania określonych aktywności na poszczególnych zasobach może być różny. Zatem nie jest możliwe globalne uszeregowanie zadań wykonywanych na różnych zasobach korzystając z znaczników czasowych zapisanych w dzienniku zdarzeń poszczególnych zasobów. Należy również pamiętać, że czas wykonywania aktywności nie jest z góry znany i w ogólności jest nieprzewidywalny.
- niezależność zasobów - zasoby zbierają dzienniki zdarzeń niezależnie od siebie zatem nie można bezpośrednio z logu wnioskować globalnej relacji pomiędzy zdarzeniami na dwóch różnych zasobach. W związku z tym nie wiadomo, które aktywności realizowane przez które zasoby systemu CRS były wykonywane w ramach którego procesu biznesowego i której instancji procesu biznesowego.



Rysunek 1 Przykład przeplotu wpisów w dzienniku zdarzeń dla aktywności logowanych w ramach różnych instancji lokalnego procesu realizowanego przez zasób

- przeplot aktywności wynikający z współbieżnego logowania wielu instancji pojedynczego zasobu do jednego logu - zasoby często wywoływane są współbieżnie przez wiele procesów biznesowych systemu CRS uruchomionych jednocześnie. W praktyce dla każdego wywołania zasobu tworzona jest nowa instancja zasobu, która jest dedykowana do jego obsłużenia. Niestety, dzienniki zdarzeń są wspólne dla wszystkich instancji zasobu co prowadzi do przeplotu aktywności wykonywanych w ramach współbieżnie działających instancji zasobu. Przedstawia to Rysunek 1 gdzie kolorami zaznaczone są aktywności dotyczące innych instancji. Zatem, potrzebna jest dodatkowa informacja związana z identyfikatorem instancji procesu lokalnego realizowanego przez zasób pozwalająca skorelować aktywności wykonywane w celu obsłużenia pojedynczego żądania.

4.1 Logowanie kontekstowe

W ramach niniejszej pracy zaproponowano rozwiązane wyżej wymienionych problemów nazwane logowaniem kontekstowym, którego celem jest zbieranie dzienników zdarzeń zasobów systemu CRS w takiej postaci, która pozwoli na jego podstawie odtworzyć log procesu biznesowego, a następnie odkryć model tego procesu. Rysunek 2 przedstawia ideę logowania kontekstowego.

Kontekstowe logowanie polega na zdefiniowaniu oraz wymuszeniu logowania pewnego minimalnego zbioru atrybutów związanych z aktywnościami wykonywanymi w ramach lokalnego procesu realizowanego przez zasób, który umożliwi odtworzenie (zrekonstruowanie) logu procesu biznesowego. W tym celu zdefiniowano dwa rodzaje zdarzeń zapisywanych w logach (dziennikach zdarzeń):

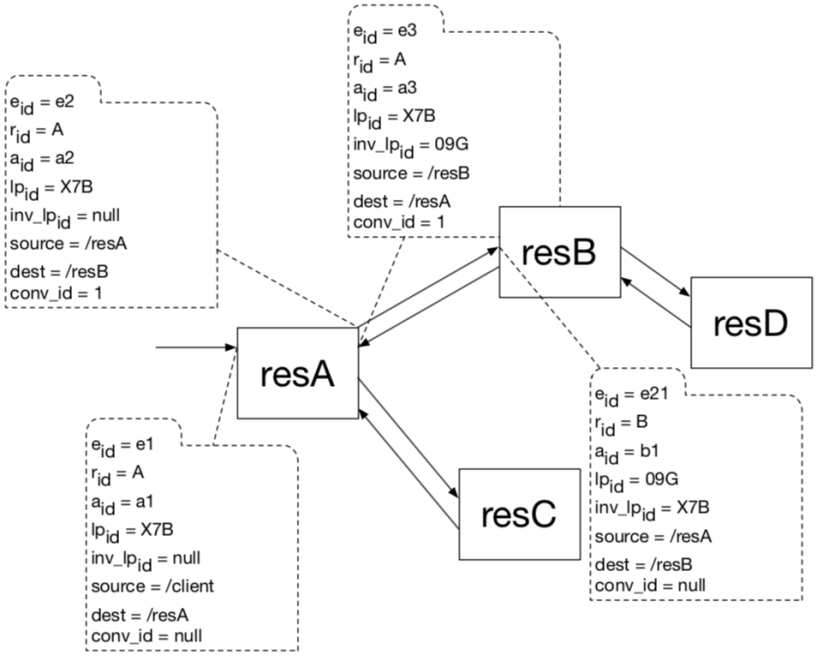
- zdarzenia lokalne, czyli zdarzenia odpowiadające aktywnościom lokalnym danego zasobu, zawierające atrybuty:
    - e_{id} – identyfikator zdarzenia (służy do identyfikacji wpisów w logu, zwykle kolejne wartości całkowitoliczbowe)
    - r_{id} – identyfikator zasobu
    - a_{id} – identyfikator aktywności wykonywanej przez proces
    - lp_{id} – identyfikator instancji lokalnego procesu realizowanego przez zasób, w ramach którego zarejestrowano wystąpienie aktywności a_{id}
- zdarzenia komunikacyjne, które odpowiadają aktywnościom związanym z odebraniem lub wysłaniem wiadomości, dodatkowo poza atrybutami obecnymi w zdarzeniach lokalnych zawierają:
    - inv_lp_{id} – identyfikator instancji lokalnego procesu, który wywołał tą (lp_{id}) instancję lokalnego procesu zasobu r_{id}
    - source – adres zasobu wysyłającego wiadomość
    - dest – adres zasobu będącego adresatem wysyłanej wiadomości
    - conv_id – identyfikator komunikacji, korelujący zdarzenie wysłania wiadomości i zdarzenie odebrania odpowiedzi na tę wiadomość

Najważniejszym elementem jest zapisywanie wraz z każdym wpisem w dzienniku zdarzeń zasobu tzw. kontekstu (unikalnego identyfikatora instancji procesu realizowanego przez zasób). W proponowanym rozwiązaniu kontekst ten jest zapisywany w polu lp_{id}. Atrybut ten umożliwia powiązanie ze sobą aktywności odebrania wiadomości będącą wywołaniem zasobu, z aktywnością odesłania odpowiedzi oraz wszystkich innych aktywności wywoływanych podczas tego wykonania lokalnego procesu jak np. wszystkich aktywności lokalnych lub wywoływania innych zasobów. Ponadto, atrybut ten pozwala szeregować aktywności zasobu w ramach pojedynczych instancji lokalnego procesu i tym samym rozwiązać problem przeplotu aktywności. Dzięki temu atrybutowi otrzymywany jest log lokalnego procesu związanego z zasobem, którego elementami są sekwencje zdarzeń związane z aktywnościami wykonywanymi podczas pojedynczej instancji procesu.

Drugim istotnym elementem kontekstowego logowania, jest przekazywanie lokalnego kontekstu zasobu do wszystkich sąsiadów, czyli zasobów, z którymi określony zasób się komunikuje. W tym celu zasób, kiedy wywołuje jakikolwiek inny zasób lub odpowiada na wywołanie innego zasobu do wysyłanej wiadomości dołącza swój lokalny kontekst w nagłówku wiadomości (np. w nagłówku HCTX wysyłanej wiadomości protokołu HTTP). Zasób, który odbiera wiadomość wraz z aktywnością odebrania tej wiadomości zapisuje wartość otrzymaną w nagłówku HCTX w atrybucie inv_lp_{id}. Dzięki temu w dzienniku zdarzeń zasobu odbierającego wiadomość znajdują się dwa identyfikatory lokalnego procesu: lokalny identyfikator aktualnego procesu oraz identyfikator lokalnego procesu zasobu, z którym nastąpiła komunikacja. Jeśli każdy zasób uczestniczący w realizacji procesu biznesowego będzie w ten sposób logował swój i sąsiadów kontekst, możliwe jest odtworzenie ciągu wywołań instancji procesów lokalnych zasobów. Ciąg wywołań instancji lokalnych procesów odpowiada ciągowi wywołań zasobów w ramach pojedynczej instancji procesu biznesowego. Tym samym pozwala na

odtworzenie pojedynczego wykonania jednego procesu biznesowego (odtworzenie śladu procesu biznesowego).

Na przykładzie, który przedstawia Rysunek 2 zasób „resA" jest wywołany przez klienta systemu i ten fakt zapisywany jest w dzienniku zdarzeń za pomocą listy atrybutów o następującej wartości: e_{id} = e1, r_{id} =A, a_{id} = a1, lp_{id} = X7B, inv_lp_{id}=null, source="/client" dest="/resA", conv_id=null. Wartość atrybutu inv_lp_{id} = null, oznacza, że zasób „resA" został wywołany przez zewnętrzny komponent systemu (wszystkie zasoby w ramach systemu CRS przesyłają swój lokalny kontekst w odpowiednim nagłówku HTTP) czyli przez klienta systemu zatem jest to początek procesu biznesowego. Każde wywołanie zasobu „resA" skutkuje stworzeniem nowej instancji tego zasobu. Instancja ta generuje lokalnie unikalny identyfikator lokalnego procesu zasobu i loguje wszystkie aktywności wykonywane podczas obsługiwania tego żądania z wygenerowanym identyfikatorem (na rysunku lp_{id} = „X7B"). Następnie w ramach wykonywania lokalnego procesu, wywoływany jest zasób „resB". Zatem odpowiednie zdarzenie wysłania wiadomości jest zapisywane w dzienniku zdarzeń. Tutaj również zapisywane są informacje, jaki zasób wywołuje jaki zasób oraz zapisywana jest informacja o lokalnym identyfikatorze instancji zasobu (lp_{id}=X7B). Po odebraniu odpowiedzi od zasobu „resB" co jest zalogowane w ramach zdarzenia e_{id} = e3 następuje analogicznie wywołanie do zasobu „resC". Warto zwrócić uwagę, że zasób „resA" nie jest świadomy tego, że w ramach realizowanego procesu biznesowego również wywoływany jest zasób „resD".



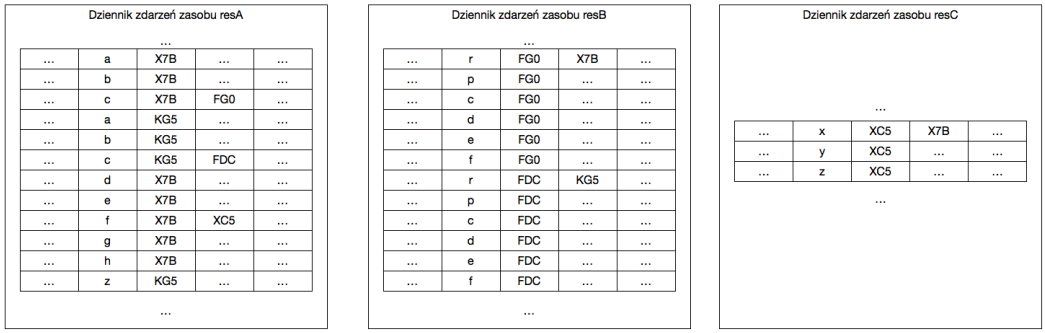
Rysunek 2 Idea logowania kontekstowego

Opisana powyżej idea kontekstowego logowania została prototypowo zaimplementowana w (11) z wykorzystaniem programowania aspektowego. Wykorzystanie aspektów pozwala stosować kontekstowy loger w sposób nieinwazyjny. Oznacza to, że nie trzeba na nowo implementować systemu, a jedynie dołączyć wspomniane aspekty do istniejącego systemu. Wymaganiem stawianym przed systemem jest by był on zaimplementowany zgodnie ze standardem JSR-311, który to jest referencyjnym standardem implementacji zasobów w języku JAVA.

4.2 Algorytm rekonstrukcji logu procesu biznesowego

W rezultacie wykorzystywania wspomnianego kontekstowego logera otrzymywany jest dziennik zdarzeń systemu CRS. Następnym krokiem przed zastosowaniem algorytmów odkrywania modelu procesu biznesowego jest odtworzenie logu procesu biznesowego na podstawie uzyskanego dziennika zdarzeń systemu CRS. W ramach niniejszej pracy zaproponowano algorytm rekonstrukcji logu procesu biznesowego, którego parametrami wejściowymi są:

- adres zasobu, który będzie traktowany jako początek procesu biznesowego (zwykle jest to zasób, który został wywołany przez klienta – inv_lp_{id} = null),
- dziennik zdarzeń systemu CRS, który posiada minimalny zdefiniowany w ramach logowania kontekstowego zestaw atrybutów.



Rysunek 3 Dzienniki zdarzeń zasobów w CRS, przed rekonstrukcją logu procesu biznesowego

Rysunek 3 przedstawia stan początkowy dziennika zdarzeń systemu CRS składającego się z dzienników zdarzeń zasobów systemu CRS. Kolejne kroki algorytmu są następujące:

1. Na początku algorytm przegląda dziennik zdarzeń zasobu, który jest początkowym zasobem procesu biznesowego. W przedstawionym przykładzie założono, że wybrany został zasób „resA".
   Każda z lokalnych instancji zasobu „resA" będzie odrębną instancją rekonstruowanego procesu biznesowego. Na przedstawionym przykładzie będzie to instancja „X7B" oraz „KG5".
2. Przeglądane są dzienniki zdarzeń każdego z zasobów i identyfikowane są zdarzenia komunikacyjne (tj. wywołania innych zasobów oraz przesyłane odpowiedzi na wywołania).
3. Analizowane są wpisy w polach lp_{id} oraz inv_lp_{id}. Dzięki temu tworzone są łańcuchy zależności kontekstów poszczególnych instancji procesów lokalnych zasobów. Dla omawianego przykładu są to:
   a. „X7B" -> „FG0", „X7B" -> „XC5"
   b. „KG5" -> „FDC"
   Znalezione łańcuchy zależności kontekstów odzwierciedlają informację, która instancja określonego zasobu wywołuje którą instancję innego zasobu.
4. Dla każdego łańcucha zależności (zbioru zależności instancji lokalnych zasobów), generowany jest unikalny identyfikator instancji globalnego procesu (identyfikator śladu procesu biznesowego) i dopisywany do każdego zdarzenia, w zasobach które należą do łańcucha. W analizowanym przykładzie (łańcuch „a.") wszystkie zdarzenia w zasobie „resA" należące do lokalnej instancji „X7B" oraz zdarzenia w zasobie „resB" należące do lokalnej instancji „FG0" oraz zdarzenia w zasobie „resC" należące do lokalnej instancji „XC5" otrzymają nowy atrybut będący identyfikatorem instancji rekonstruowanego globalnego procesu.
5. Rezultatem algorytmu jest zbiór śladów (odpowiadających instancjom) rekonstruowanego procesu biznesowego, składający się z kolekcji lokalnych śladów (instancji) procesów realizowanych przez zasoby wywoływane w ramach instancji procesu biznesowego. Zwracany zbiór śladów (odpowiadających instancjom) procesu biznesowego jest logiem procesu biznesowego.

5 Odkrywanie modelu procesu biznesowego w systemach CRS

W ramach niniejszej pracy zaproponowano trzy algorytmy, które na podstawie kompletnego logu procesu biznesowego realizowanego w systemach CRS umożliwiają odkrywanie modelu tego procesu. W tym celu wykorzystują one relacje określające zależności pomiędzy aktywnościami występującymi w ramach procesu na podstawie logu tego procesu. Relacje te uwzględniają lokalne zależności jakie występują w ramach lokalnych procesów realizowanych przez zasoby, zdalne zależności uwzględniające interakcje pomiędzy zasobami oraz globalne relacje z perspektywy teoretycznego obserwatora systemu postrzegającego cały proces biznesowy. Ich pełna definicja oraz dokładny opis zwarty został w pracy.

5.1 CRS miner – odkrywanie zgodnej i ustrukturyzowanej sieci przepływu

CRS miner to pierwszy z proponowanych algorytmów odkrywania modelu procesu biznesowego w systemach CRS (11). Algorytm ten zakłada następujące własności:

- Parametrem wejściowym jest kompletny log procesu biznesowego realizowanego w systemie CRS;
- Odkrywany proces biznesowy może być zamodelowany za pomocą zgodnej i ustrukturyzowanej sieci przepływu

Celem algorytmu CRS miner jest odkrycie modelu procesu biznesowego uruchamianego w CRS na podstawie kompletnego logu tego procesu. Niestety taki log nie zawsze jest dostępny. W związku z tym możliwym jest wykorzystanie odtworzonego logu uzyskanego jako rezultat działania algorytmu rekonstrukcji logu procesu biznesowego. Działanie algorytmu jest następujące:

1. Przejrzyj log procesu biznesowego i zmień nazwy aktywności na globalnie unikalne poprzez wykonanie konkatenacji unikalnej globalnie nazwy zasobu z nazwą aktywności;
2. Dla każdego zasobu przeanalizuj log procesu biznesowego z unikalnymi globalnie nazwami aktywności i wyznacz pary aktywności, które należą do kolejnych relacji lokalnych: $\succ^{G}_{RPL_j}, \overset{syn}{\longrightarrow}_{RPL_j}, \rightarrow^{G}_{RPL_j}, \#^{G}_{RPL_j}, ||^{G}_{RPL_j}$;
3. Dla każdej pary zasobów przeanalizuj log procesu biznesowego z unikalnymi globalnie nazwami aktywności i wyznacz pary aktywności, które należą do kolejnych relacji zdalnych: $\overset{ctx}{\longrightarrow}_{RPL_{j,k}}, ||^{G}_{RPL_{j,k}}$;
4. Na podstawie wyznaczonych relacji w kroku 3 i 4 wyznacz pary aktywności należące do relacji globalnych: $\rightarrow^{G}_{CRSGPL}, ||^{G}_{CRSGPL}, \#^{G}_{CRSGPL}$;
5. Wykonaj algorytm Alpha (9) dla logu procesu biznesowego z wykorzystaniem wyznaczonych relacji globalnych;

W pracy również przeprowadzono dowód potwierdzający, że algorytm przy zdefiniowanych założeniach zawsze odkryje zgodną i ustrukturyzowaną sieć przepływu, która będzie równoważna rzeczywistemu procesowi biznesowemu wykonywanemu w CRS. Rysunek 5 (prawy) przedstawia przykładowy wynik działania algorytmu.

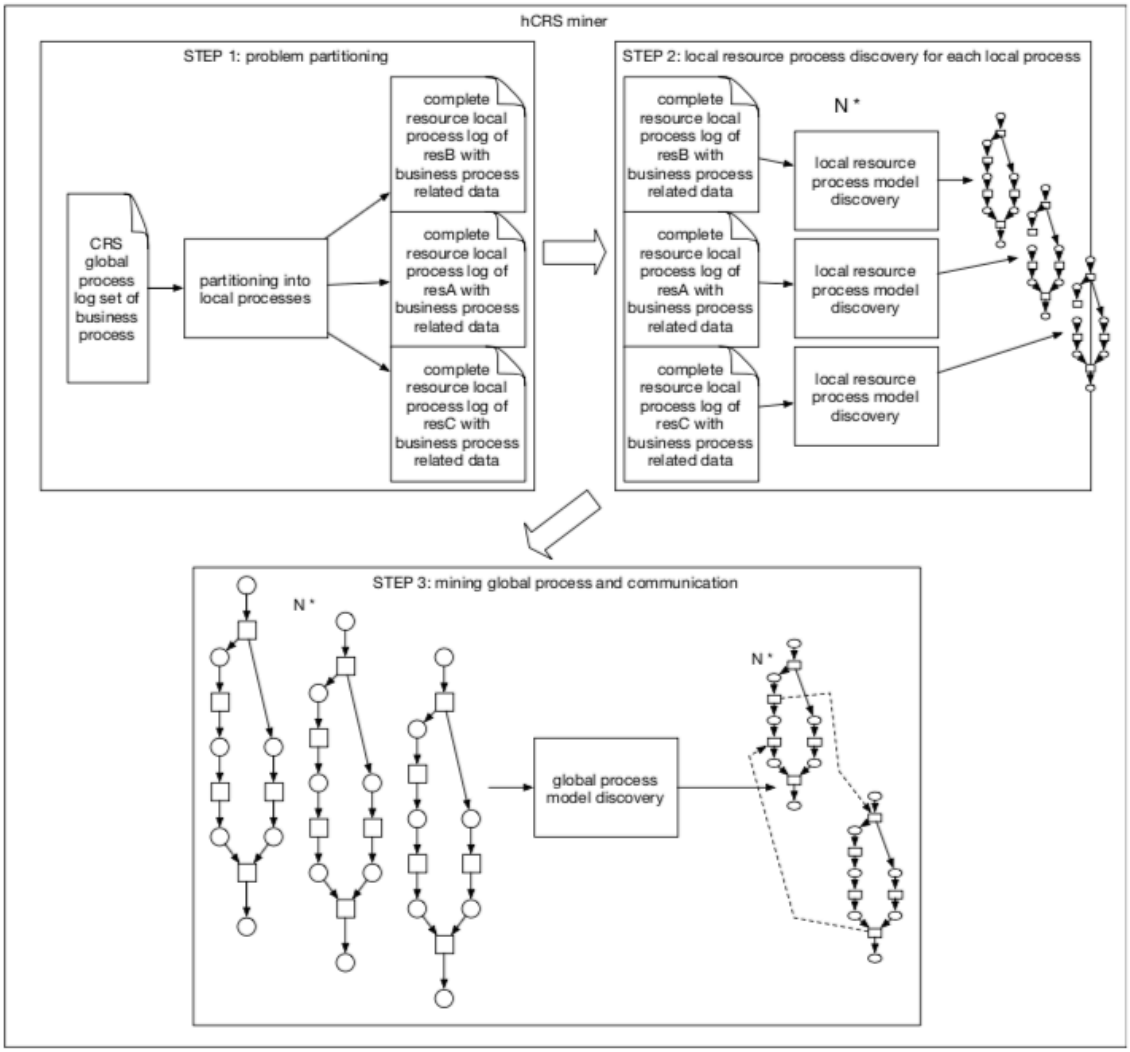
5.2 hCRS miner – odkrywanie sieci komunikujących się zasobów

Drugim algorytmem zaproponowanym w pracy jest hCRS miner (12) (13). Najważniejszymi udoskonaleniami względem poprzedniego algorytmu CRS miner są:

- Odkrywanie lokalnych procesów zasobów w sposób niezależny;
- Okrywanie procesu biznesowego jako kompozycji procesów lokalnych wykonywanych na poszczególnych zasobach systemu poprzez uwzględnienie komunikacji pomiędzy nimi;
- Dekompozycja problemu odkrywania na mniejsze pod problemy w celu zoptymalizowania czasu wykonania algorytmu.

Założenia algorytmu hCRS miner:

- Parametrem wejściowym jest kompletny log procesu biznesowego;
- Odkrywany proces biznesowy może być zamodelowany za pomocą sieci komunikujących się zasobów, a lokalne procesy zasobów mogą być zamodelowane za pomocą zgodnej i ustrukturyzowanej sieci przepływu.

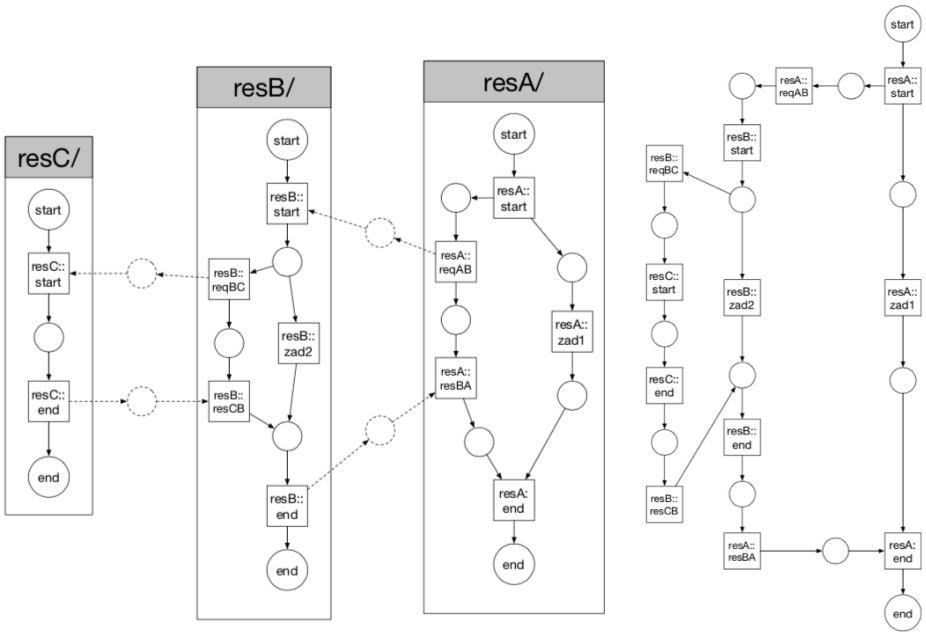


Rysunek 4 Zasada działania algorytmu hCRS miner

Ideę algorytmu przedstawia Rysunek 4 i polega ona na wykonaniu następujących kroków:

1. Kompletny log procesu biznesowego jest dekomponowany do kolekcji kompletnych logów lokalnych procesów zasobów, które uczestniczą w wykonaniu procesu biznesowego
2. Dla każdego logu procesu lokalnego zasobów odkrywany jest model procesu lokalnego (rozszerzony algorytm Alpha). Modyfikacje dotyczą:
    a. Wyznaczania aktywności odpowiadających za komunikację, w celu późniejszego ich połączenia kanałem komunikacyjnym z odpowiednimi zasobami
    b. Optymalizacji wyznaczenia możliwych stanów przetwarzania procesu lokalnego dzięki uwzględnieniu informacji o występowaniu komunikacji synchronicznej ($\xrightarrow{syn}_{RPL_j}$)
    c. Optymalizacji wyznaczania możliwych stanów przetwarzania procesu lokalnego wykorzystując fakt, że niektóre aktywności nie mogą być poprzednikami lub następnikami innych aktywności.
3. Odkrycie modelu procesu biznesowego jako kompozycja procesów lokalnych realizowanych przez zasoby poprzez połączenie kanałami komunikacyjnymi odpowiednich aktywności w procesach lokalnych ($\xrightarrow{ctx}_{RPL_{j,k}}$).

Wynikiem działania algorytmu jest model procesu biznesowego (Rysunek 5 - lewy) zapisany za pomocą sieci komunikujących się zasobów, gdzie każdy lokalny proces zasobu jest zgodną i ustrukturyzowaną siecią przepływu. Model ten charakteryzuje się przede wszystkim możliwością analizy każdego lokalnego procesu realizowanego przez zasoby niezależnie (można abstrahować od innych zasobów systemu) oraz możliwością analizy kolejności i miejsca wystąpienia komunikacji pomiędzy zasobami. Dowód twierdzenia, że odkryty model odpowiada rzeczywistemu procesowi biznesowemu zawarty został w pracy.



Rysunek 5 Lewy: Model procesu odkryty przez algorytm hCRS miner
Prawy: Model procesu odkryty przez algorytm CRS miner

5.3 dhCRS miner – rozproszone odkrywanie sieci komunikujących się zasobów

dhCRS miner (13) (12) to rozproszona wersja algorytmu hCRS miner, z dodatkowym krokiem pozwalającym na odkrywanie hierarchii zasobów. Głównymi celami modyfikacji są:

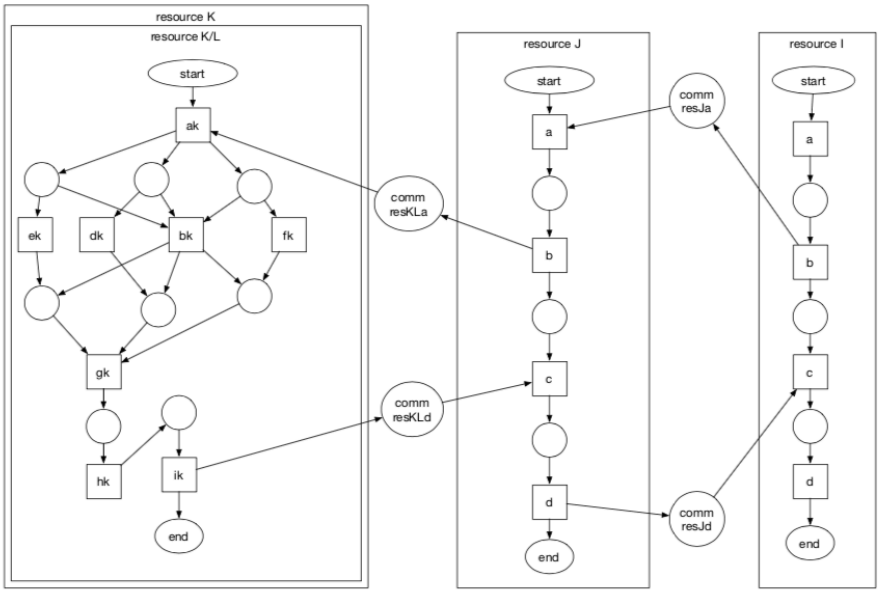
- Rozproszenie obliczeń związanych z odkrywaniem modelu procesu biznesowego w celu poprawy wydajności algorytmu.
- Odkrywanie hierarchii zasobów występującej pomiędzy zasobami kooperującymi w ramach odkrywanego procesu biznesowego.

Założenia dla tego algorytmu są takie same jak w przypadku hCRS miner. dhCRS miner w celu odkrywania modelu procesu biznesowego korzysta z klastra obliczeniowego, gdzie jeden węzeł nazywany jest masterem i zarządza obliczeniami częściowymi wykonywanymi na tzw. węzłach obliczeniowych.

Idea działania algorytmu jest następująca:

1. Na węźle master: Kompletny log procesu biznesowego jest dekomponowany do kolekcji kompletnych logów lokalnych procesów realizowanych przez zasoby, które uczestniczą w wykonaniu procesu biznesowego
2. Na węźle master: Każdy z logów lokalnych procesów jest przesyłany do węzła liczącego będącego w klastrze obliczeniowym. Zasada przydziału zadań do węzłów jest następująca: jeśli węzeł aktualnie nie odkrywa żadnego modelu lokalnego procesu, przydziel mu dostępny log.
    a. Na każdym węźle obliczeniowym:
    Dla każdego logu procesu lokalnego odkrywany jest model procesu lokalnego (rozszerzony algorytm Alpha). Modyfikacje dotyczą:
        i. Wyznaczania aktywności odpowiadających za komunikację, w celu późniejszego ich połączenia kanałem komunikacyjnym z odpowiednimi zasobami;
        ii. Optymalizacji wyznaczenia możliwych stanów przetwarzania procesu lokalnego dzięki uwzględnieniu informacji o występowaniu komunikacji synchronicznej ($\overset{syn}{\longrightarrow}_{RPL_j}$);
        iii. Optymalizacji wyznaczania możliwych stanów przetwarzania procesu lokalnego wykorzystując fakt, że niektóre aktywności nie mogą być poprzednikami lub następnikami innych aktywności.
3. Na węźle master: zbierane są częściowe wyniki (zgodne i ustrukturyzowane sieci przepływu) odkryte dla wszystkich zasobów biorących udział w procesie biznesowym przesłane od węzłów obliczeniowych
4. Odkrycie modelu procesu biznesowego jako kompozycji procesów lokalnych poprzez połączenie kanałami komunikacyjnymi odpowiednich aktywności w procesach lokalnych zasobów systemu ($\overset{ctx}{\longrightarrow}_{RPL_{j,k}}$).
5. Odkrycie hierarchii zasobów na podstawie informacji zapisanej w adresach np. URI zasobów.

Wynikiem działania algorytmu jest model procesu biznesowego zapisany za pomocą sieci komunikujących się zasobów, a lokalne procesy poszczególnych zasobów zamodelowane są za pomocą zgodnych i ustrukturyzowanych sieci przepływu (dowód jest tożsamy z dowodem dla algorytmu hCRS miner). Model charakteryzuje się tymi samymi własnościami co model odkryty prze hCRS miner z dodatkową informacją dotyczącą hierarchii zasobów. Ponadto, algorytm cechuje się znacznie poprawioną wydajnością czasową odkrywania modelu procesu względem CRS minera i hCRS minera. Rysunek 6 przedstawia przykładowy wynik działania algorytmu.

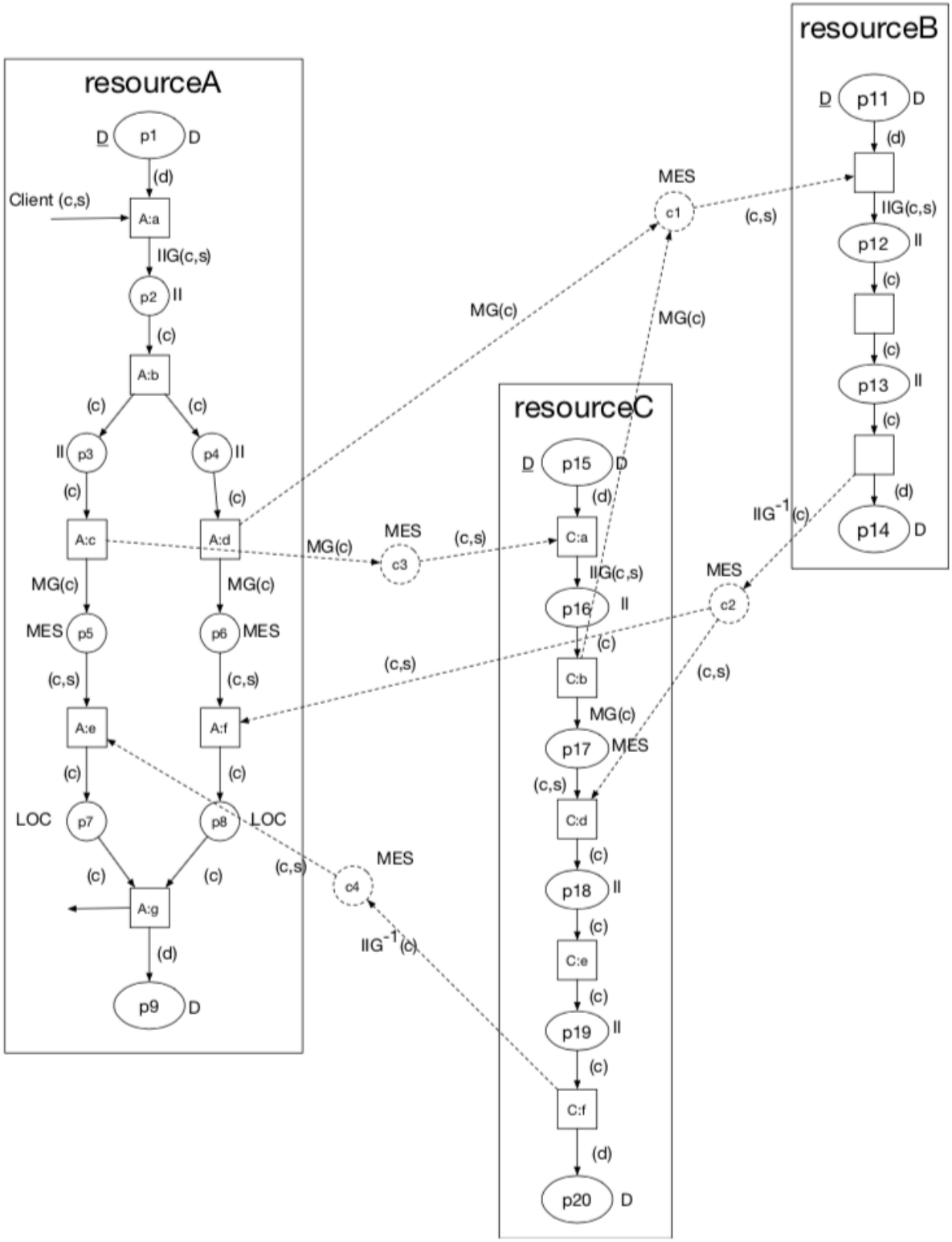


Rysunek 6 Model procesu biznesowego odkryty przez dhCRS miner

5.4 cdhCRS miner – odkrywanie kolorowanej sieci komunikujących się zasobów

Ostatnim z zaproponowanych algorytmów jest modyfikacja algorytmu dhCRS miner. Głównym celem tego algorytmu jest odkrywanie kolorowanej sieci komunikujących się zasobów, która dzięki wykorzystaniu idei kolorowanych żetonów umożliwia modelowanie procesu biznesowego w bardziej kompaktowej wersji. W przypadku sieci komunikujących się zasobów każda instancja poszczególnych zasobów systemu jest reprezentowana przez niezależną sieć (poszczególne instancje lokalnych procesów zasobów nie interferują ze sobą). W przypadku kolorowanych sieci reprezentujących lokalne procesy zasobów, izolacja gwarantowana jest poprzez kolory żetonów i odpowiednie wyrażenia na łukach.

Algorytm ten współdzieli z algorytmem dhCRS miner założenia. Ponadto, modyfikuje krok 4 algorytmu, poprzez generowanie zawsze dwóch kanałów komunikacyjnych dla każdego zasobu niezależnie, ile jego instancji bierze udział w przetwarzaniu (a nie jak przypadku algorytmu dhCRS miner gdzie każda instancja zasobu ma dwa kanały komunikacyjne). Dodatkowo rozszerzono dhCRS miner o dodatkowy krok kolorowania sieci oraz dodawania wyrażeń na łukach. Wynik działania tego algorytmu przedstawia Rysunek 7.



Rysunek 7 Kolorowana sieć komunikujących się zasobów

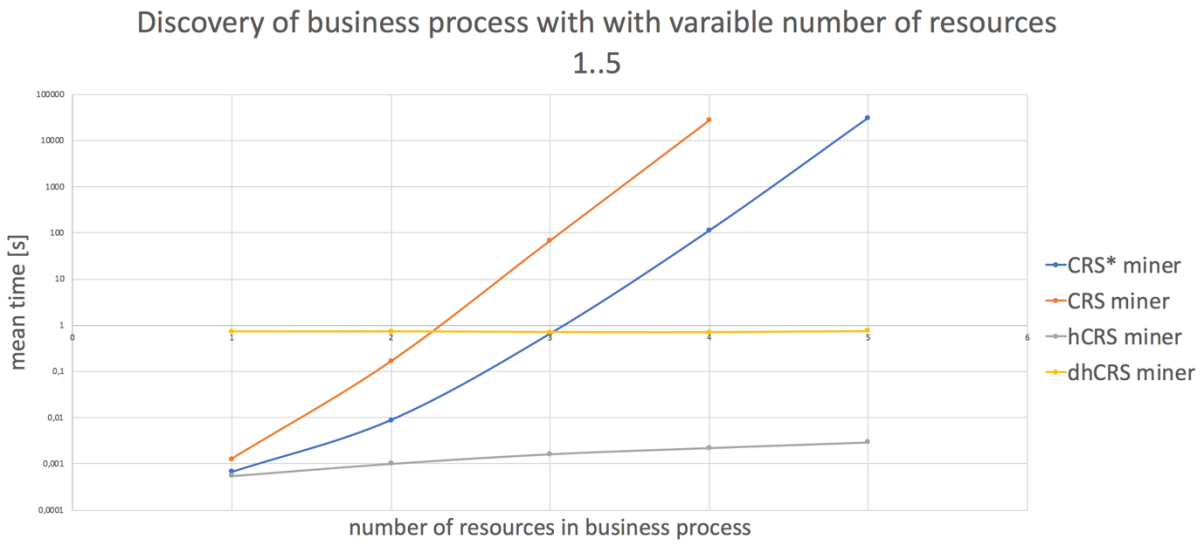
6 Porównanie zaproponowanych algorytmów

W pracy dokonano porównania zaproponowanych algorytmów.

Najważniejszymi punktami porównania jest analiza złożoności obliczeniowej oraz testy wydajnościowe.

W pracy wykazano, że Algorytm CRS miner jest wykładniczo zależny od liczby unikalnych globalnie aktywności wykonywanych w ramach odkrywanego procesu biznesowego. W związku z tym jest on przeznaczony dla stosunkowo niewielkich procesów biznesowych. hCRS miner oraz dhCRS miner charakteryzują się rozbiciem problemu na problemy szczegółowe, dzięki temu zależność wykładnicza dotyczy odkrywania tylko pojedynczego lokalnego procesu każdego z zasobów. Ponadto, krok łączenia lokalnych procesów w proces biznesowy charakteryzuje się liniową zależnością od liczby zasobów. W rezultacie hCRS miner i dhCRS miner mogą być stosowane w nawet bardzo złożonych procesach biznesowych.

W pracy przeprowadzono szereg testów wydajnościowych w celu praktycznej weryfikacji działania algorytmów (przykładowe wyniki zawarte w pracy prezentuje: Rysunek 8 i Rysunek 9).



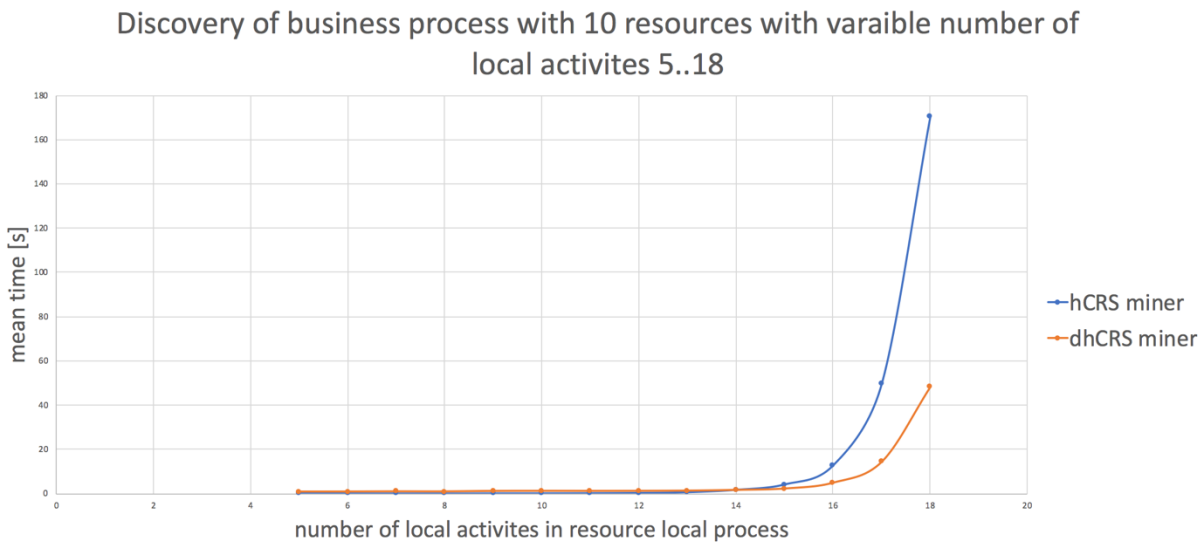
Rysunek 8 Odkrywanie modelu procesu biznesowego dla zmiennej liczby zasobów (1..5) składających się z 5 aktywności każdy.

Otrzymane rezultaty potwierdzają wykładniczą złożoność zaproponowanych metod w zależności od liczby lokalnych aktywności w lokalnych procesach oraz liniową zależność od liczby komunikujących się zasobów w przypadku hCRS oraz dhCRS.

Ponadto, wyniki potwierdzają, że podejście do problemu odkrywania modelu procesu biznesowego w CRS polegające na podzieleniu go na mniejsze problemy, a następnie łączenie ich ze sobą w kompletny model jest wydajną i efektywną metodą, skutkującą istotnym skróceniem czasu odkrywania modelu, nawet w sytuacji kiedy liczba unikalnych globalnie aktywności przekracza 100. Najwydajniejszym algorytmem dla skomplikowanych procesów jest dhCRS miner, który zauważalnie lepiej działa od innych proponowanych algorytmów przy procesach składających się z ponad 140 globalnie unikanych aktywności. Poniżej tej wartości hCRS miner może być lepszy (szczególnie gdy lokalne procesy zasobów cechują się niewielką liczbą aktywności lokalnych).

W przypadku CRS minera testy wykazują, że jego praktyczne zastosowanie jest ograniczone do dość niewielkich procesów. W ramach testów również zaimplementowano CRS* minera, czyli zoptymalizowaną wersję CRS minera wykorzystującą optymalizację odkrywania lokalnego procesu zasobu zaproponowaną dla hCRS minera. Optymalizacja ta już przy niewielkich procesach może skutkować ponad 100 krotnym przyśpieszeniem, natomiast algorytm jest dalej wykładniczo zależny od

liczby wszystkich aktywności występujących w procesie biznesowym, a zatem przy większych procesach hCRS miner i dhCRS miner będą wydajniejsze.



Rysunek 9 Odkrywanie modelu procesu biznesowego w zależności liczby aktywności realizowanych w procesach lokalnych (dla 10 zasobów).

7 Zakończenie

Praca dotyczy systemów komunikujących się zasobów CRS będących uogólnieniem systemów REST. W pracy osiągnięto wszystkie zdefiniowane cele badawcze oraz wykazano, że możliwe jest odkrywanie modelu procesu biznesowego realizowanego w systemie CRS, który jest strukturalnie równoważny rzeczywistemu procesowi wykonywanemu w systemie na podstawie jego kompletnego dziennika zdarzeń.

W tym celu opracowano koncepcję budowania dzienników zdarzeń systemów CRS nazwaną kontekstowym logowaniem, prototypową implementację kontekstowego logera oraz algorytm rekonstrukcji kompletnego logu procesu. Uzyskane w ten sposób kompletne logi procesów biznesowych stanowiły dane wejściowe dla zaproponowanych w ramach pracy algorytmów odkrywania modeli procesów biznesowych. Modele te budowane są z wykorzystaniem zaproponowanych w pracy sieci komunikujących się zasobów, bazujących na sieciach przepływu (*ang. workflow net*). Zaproponowane sieci umożliwiają modelowanie systemu CRS z uwzględnieniem jego szczególnych własności takich jak: kompozycja procesu biznesowego z wielu niezależnych procesów lokalnych, synchroniczny model komunikacji oraz hierarchia zasobów.

W ramach pracy zaproponowano trzy algorytmy odkrywania modelu procesu biznesowego w CRS oraz zaproponowano rozszerzenie jednego z nich:

- CRS miner, odkrywający model zgodnej i ustrukturyzowanej sieci przepływu,
- hCRS miner, odkrywający model sieci komunikujących się zasobów,
- dhCRS miner, rozproszona wersja hCRS minera umożliwiająca dodatkowo na odkrywanie hierarchii zasobów,
- modyfikacja dhCRS minera (cdhCRS miner) umożliwiającą odkrywanie kolorowanej sieci komunikujących się zasobów.

W pracy, udowodniono poprawność zaproponowanych algorytmów, podano ich złożoność obliczeniową oraz oceniono wydajność. Uzyskane wyniki eksperymentalne potwierdzają, że zaproponowane metody są wystarczająco wydajne by mogły być z powodzeniem stosowane w praktyce. Użycie opracowanego w ramach pracy podejścia do uzyskiwania modeli procesów biznesowych równoważnych procesom rzeczywistym może w istotny sposób wspomóc zarządzanie systemami CRS oraz umożliwić analizę odkrytych procesów, pozwalającą na identyfikację niepożądanych sytuacji takich jak: pętle wywołań, wąskie gardła, zakleszczenia, braki zasobów lub naruszenia własności systemów CRS (REST, ROA).

8 Bibliografia

1. **Eric A. Marks, Michael Bell.** *Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology.* 2006.

2.**Roy Thomas Fielding.** *Architectural Styles and the Design of Network-based Software Architectures .* 2000.

3. **Jerzy Brzeziński, Arkadiusz Danilecki, Jakub Flotyński, Anna Kobusińska, Andrzej Stroiński.** ROSWeL workflow language: A declarative, resource-oriented approach. *New Generation Computing.* 2012.

4. **Jolie homepage.** [Online] [Cited: 03 22, 2017.] http://www.jolie-lang.org.

5. Wil M. P. van der Aalst. *Process Mining Discovery, Conformance and Enhancement of Business Processes.* 2011.

6. Guido Schimm. Mining exact models of concurrent workflows. *Computers in Industry.* 2004.

7. Lijie Wen, Jianmin Wang, Wil MPvan der Aalst, Biqing Huang, and Jiaguang Sun,. *Process mining with the heuristics miner-algorithm.* 2010.

8. Schahram Dustdar, Robert Gombotz,. Discovering web service workflows using web services interaction mining. *International Journal of Business Process Integration and Management.* 2006.

9. W. M. P. van der Aalst, Ton Weijters, Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Knowledge and Data Eng.* 2004.

10. Kurt Jensen. An introduction to the theoretical aspects of Coloured Petri Nets. *A Decade of Concurrency Reflections and Perspectives.* 1994.

11. *Resource Mining: Applying Process Mining to Resource-Oriented Systems.* **Andrzej Stroiński, Dariusz Dwornikowski, Jerzy Brzeziński.** s.l. : Springer International Publishing, 2014. International Conference on Business Information Systems.

12. *RESTful Web Service Mining: simple algorithm supporting resource-oriented systems.* **Andrzej Stroinski, Dariusz Dwornikowski, Jerzy Brzezinski.** s.l. : IEEE, 2014. Web Services (ICWS), 2014 IEEE International Conference on.

13. Andrzej Stroinski, Dariusz Dwornikowski, Jerzy Brzezinski. A distributed discovery of communicating resource systems models. *IEEE Transactions on Services Computing.* 2016 (accepted for publication).

14. *Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics.* **Christian W. Günther, Wil M. P. van der. Aalst,** s.l. **:** Springer, 2007. International Conference on Business Process Management**.**