Streszczenie

Agentowy system zarządzania urządzeniami HVAC w budynku biurowym

???Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.???

Słowa kluczowe: Wieloagentowy, HVAC, aktorzy

Abstract

Agent system for HVAC management in office building

???Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.???

Keywords: Multi-agent, HVAC, actors

Jan Grzybowski
Nr albumu 245491

Warszawa, dnia .....

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem „Agentowy system zarządzania urządzeniami HVAC w budynku biurowym", której promotorem jest dr hab. Marcin Paprzycki wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Jan Grzybowski

Spis treści

Część I

Wstęp i analiza

1. Wprowadzenie

2. Analiza biznesowa

2.1. Omówienie zagadnienia

W większości budynków biurowych zamontowane są klimatyzatory, które zapewniają komfortowe warunki pracy osób przebywających w biurach. Ilość prądu potrzebnego do funkcjonowania sieci takich urządzeń jest sporą częścią miesięcznych kosztów dla właściciela budynku.

Przy ręcznym ustawianiu komfortowej temperatury przez człowieka zwykle odbywa się to na początku spotkania gdy uczestnicy uznają, że warunki w pomieszczeniu nie odpowiadają ich wymaganiom. Aby jak najszybciej pozbyć się tego uczucia włączają najmocniejszy, lecz nie koniecznie najbardziej oszczędny, tryb w klimatyzatorach.

Założeniem implementowanego w tej pracy systemu jest ustalanie temperatury komfortu przed spotkaniem i uruchomienie klimatyzatorów przed spotkaniem w najbardziej ekonomicznym wariancie, tak aby uczestnicy mieli komfortowe warunki od początku spotkania i utrzymanie ich przez całe spotkanie.

Głównym źródłem informacji w których pomieszczeniach i w jakich godzinach odbywają się spotkania są kalendarze firmowe. Można wyobrazić sobie inne źródła takie jak lokalizacja pracowników przypisanych do pomieszczenia, jednak implementowany system będzie skupiał się głównie na wydarzeniach firmowych, jako że są mniej dynamiczne i można za ich pomocą zamodelować również np. spóźniającego się pracownika przesuwając godzinę spotkania.

Przewidzieć ile czasu wcześniej należy uruchomić urządzenia i w jakim trybie można za pomocą matematycznych modeli. Przygotowanie dokładniejszych modeli nie mieści się w zakresie pracy, przyjęto zatem prosty model opierający się o moc i wydajność urządzeń.

2.2. Analiza wymagań systemu

Przy założeniu ewentualnej późniejszej rozbudowy systemu lub przeniesienia go na inny język programowania system musiał spełniać następujące wymagania:

Możliwość dodania nowych parametrów pomieszczenia

Dodanie nowych parametrów takich jak wilgotność czy nasłonecznienie pozwalałoby udoskonalić model temperatur i zbliżyć go do rzeczywistego modelu fizycznego.

Jednocześnie dodanie takich parametrów jak poziom tlenu w pomieszczeniu dawałby możliwość sprawdzenia czy osobom w pomieszczeniu nie jest zbyt duszno oraz przewietrzenia pomieszczenia za pomocą dostępnych urządzeń.

Oczywiście każde dodanie takiego parametru wiązałoby się z nowym typem sensora i w niektórych przypadkach aktuatora o ile badane zajwisko zmianiałoby się na tyle wolno, że moglibyśmy na nie oddziałowywać.

Parametryzacja i podmiana modeli

Możliwość parametryzacji model pozwoliłaby dostrajać modele i poprawić precyzję działania i sugesti systemu. Podmiana modeli umożliwiłaby aktualizację parametrów modelu lub podmianę na inaczej skonstruowany model (np. przewidujący temperatury z innych parametrów pomieszczenia) bez wyłączania systemu .

Możliwość podpięcia różnych źródeł danych

W różnych firmach używa się różnych systemów do ustalania terminów spotkań czy rezerwacji sal np. Microsoft Exchange Server czy IBM Domino. System powinien pozwalać na dopisanie adaptera zajmującego się przekazywaniem wydarzeń do systemu HVAC i uniezależnianiem go od źródła danych.

Możliwość zbadania stanu systemu

Dla celach diagnostycznych oraz aby dostroić modele potrzebne są dane o stanie urządzeń oraz dane z samego systemu HVAC. Musi zatem istnieć sposób, aby w łatwy sposób połączyć się z systemem HVAC i zerbać informacje o jego stanie.

2.3. Analiza wymagań aplikacji symulatora

Użycie rzeczywistych urządzeń HVAC w celu sprawdzenia poprawności działania systemu nie było możliwe w ramach tej pracy ze względu na kosztowność i czasochłonność tej metody.

Potrzebny był zatem symulator który zawierałby w sobie system HVAC oraz interfejs użytkownika umożliwiający interakcję z systemem. Taki symulator powinien też pozwalać na przyspieszenie czasu w symulowanym systemie, ograniczyć czas potrzebny na testowanie.

Aby móc wykonywać powtarzalne próby wydajnie potrzebne były:

Symulacja upływu czasu

Aby nie czekać na wynik działania systemu po np. dwóch godzinach, system powinien pozwalać na przeskalowanie czasu upływającego w systemie np. jedna sekunda czasu rzeczywistego to 5 minuty czasu symulatora.

Mechanizm ładowania schematu budynku

Aplikacja symulatora powinna pozwalać na zapisanie i ponowne załadowanie właściwości pomieszczeń wraz ze stanem urządzeń zanjdujących się w nim.

Silnik scenariuszy

Scenariusze czyli lista sygnałów które odbiera system o określonym czasie. Np. podniesienie się temperatury czy przesunięcie spotkania. Za pomocą odtwarzalnych scenariuszy możemy wielokrotnie testować zachowanie systemu w tych samych warunkach, ale np. z innymi parametrami w modelu temperatury.

2.4. Przegląd rozwiązań agentowych dla środowska .NET

2.4.1. Boris.NET

Boris jest biblioteką do tworzenia systemów agentowych stworzoną w takcie badania metod projektowania systemów agentowych w Teesside University, Wielka Brytania. Protokół Borisa pozwala na łączenie w jeden system agentów napisanych za pomocą różnych języków takich jak C++, Lisp czy Java.

Boris.NET jest biblioteką napisaną przez Aliego Bojarpour do obsługi biblioteki Boris za pomocą języków C#i F#.

Elastyczność Borisa pod względem ilości obsługiwanych jest cenną cechą, gdyż pozwalałaby na pisanie adapterów i dodatkowych funkcjonalności przez różne zespoły. Niestety ani Boris ani Boris.NET nie są projektami open-source co uniemożliwia analizę i poznanie mechanizmów

wewnętrznych biblioteki. Podobnie rzecz ma się ze wsparciem społeczności, która ogranicza sie do osób ze środowiska akademickiego.

Brak też obszernej dokumentacji, która jest potrzebna przy poznawaniu nowego modelu oprogramowania jakim jest system agentowy.

2.4.2. Orleans

2.4.3. Akka.net

2.4.4. Podobieństwa między Akka.net i Orelans

2.4.5. Różnice między Akką.net i Orelans

2.4.6. Uzasadnienie wyboru Akka.net

Część II

Dokumentacja rozwiązania

3. Architektura rozwiązania

3.1. Struktura systemu zarządzania urządzeniami

3.2. Struktura aplikacji symulatora

3.3. Sygnały z zewnątrz systemu

3.4. Sygnały na zewnątrz systemu

4. Aktorzy

4.1. Źródło czasu

4.2. Źródło modeli

4.3. Firma

4.4. Pomieszczenie

4.5. Symulator pomieszczenia

4.6. Komponenty

4.6.1. Komponent cykliczny

4.6.2. Komponent symulujący

4.7. Sensor

4.8. Symulator sensora

4.8.1. Symulator parametru pomieszczenia

4.9. Kontroler

5. Wzorce projektowe wykorzystane podczas realizacji projektu

5.1. Rekwizyty

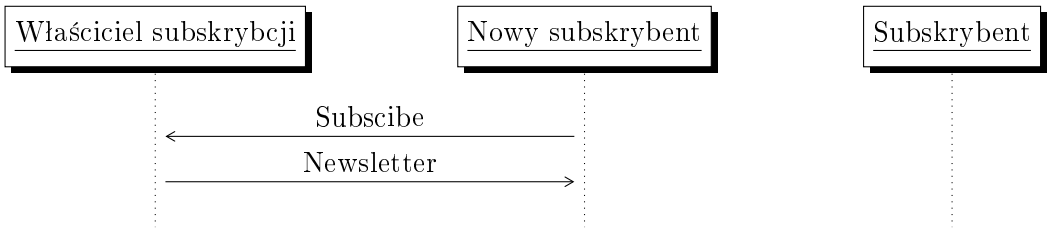
Rekwizyt (ang. props) to obiekt w akka.net pozwalający na przechowywanie sposobu tworzenia obiektu aktora. Są wykorzystywane przy zarówno tworzeniu jak i ponownym uruchamianiu aktorów po nastąpieniu nieobsłużonego wyjątku. Tylko one mają bezpośredni dostęp do kontruktora aktora. W momencie tworzenia aktora wykorzystywany jest obiekt rekwizytów i wywoływane jest ukryte wewnątrz wyrażenie z konstruktorem.

5.2. Model subskrypcyjny

Model subskrypcyjny (ang. subscribe/unsubscribe) to sposób komunikacji między agentami. Składa się z właściciela subskrypcji oraz subskrybentów.

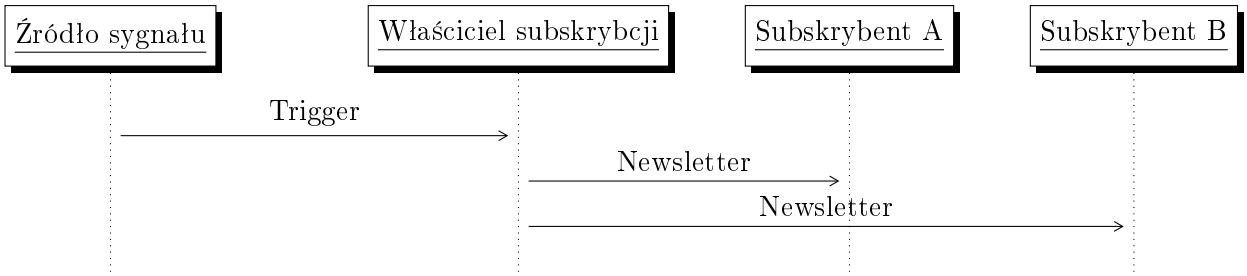
Definicja 5.1. Subskrypcja to wiadomość rozsyłana przez jej właściciela do wszystkich którzy się złosili.

Definicja 5.2. Subskrybentem nazywamy agenta, który zgłosił właścicielowi subskrypcji chęć otrzymywania subskrypcji.



Rysunek 5.1: Schemat dodawania subskrybenta

W realizowanym systemie właściciel subskrypcji wysyła ostatnią rozesłaną subskrypcję do agenta, który zapisał się na subskrypcję. Dzięki temu dopiero co zapisany agent nie musi czekać na kolejny syngał inicjalizujący rozesłanie subskrypcji.



Rysunek 5.2: Schemat rozsyłania subskrypcji

Właściciel subskrypcji może rozsyłać subskrypcję zarówno na sygnał czasowy jak i sygnał przychodzący z zewnątrz.

5.3. Udostępnianie informacji o stanie aktora

Aby móc w łatwy sposób odczytywać stan wewnętrzny systemu, większość jego aktorów dziedziczy po specjalnie napisanym typie aktora ???DebugableActor???. Ta klasa pozwala na przygotowanie informacji, która będzie rozsyłana za pomocą specjalnej subskrypcji diagnostycznej. Zadanie wybrania momentów w których rozsyłana jest subskrypcja należy do osoby rozszerzającej tą klasę.

Metody dostępne do rozszerzenia lub użycia w klasie ???DebuggableActor???

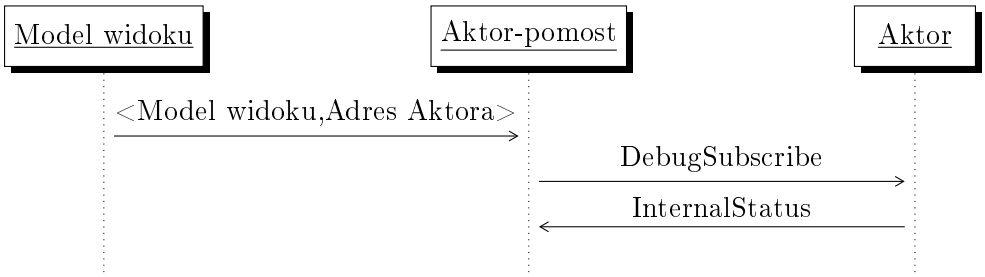
1. GenerateInternalState() - tworzy obiekt przedstawiający stan wewnętrzny aktora (metoda do rozszerzenia).
2. InformAboutInternalState() - rozsyła subskrypcję stanu wewnętrznego do wszystkich diagnostycznych subskrybentów.
3. InformDebugSubscribers(object x) - Pozwala rozesłać do subskrybentów diagnostyczych inną wiadomość niż tą generowaną za pomocą GenerateInternalState()
4. SetInternalStatus() - Metoda pozwalająca na nadgranie statusu wewnętrznego aktora np. w celu testowania. Domyślnie nie zmienia stanu aktora a jedynie wywołuje metodę InformAboutInternalState(). (metoda do rozszerzenia)

5.4. Aktor-pomost

Aktor-pomost (ang. BridgeActor) to aktor pośredniczący w komunikacji pomiędzy interfejsem użytkownika symulatora a właściwym agentem z właściwego systemu.

Tworzenie agenta-pomostu

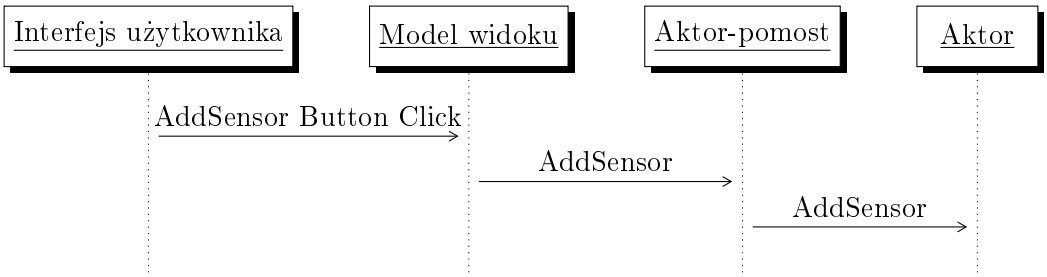
Powinien mieć przypisanego tylko jednego agenta z systemu i tylko jeden model widoku z interfejsu użytkownika. Przy tworzeniu pomost zapisuje się na subskrypcję diagnostyczną, ab odbierać informacje o wszelkich możliwych zmianach stanu aktora, do którego jest pośrednikiem.



Rysunek 5.3: Schemat tworzenia aktora-pomostu

Głównym zadaniem aktora-pomostu jest przesyłanie wiadomości do aktorów wewnątrz systemu zarządzającego urządzeniami oraz aktualizacja danych widocznych w aplikacji.

Przekazywanie wiadomości

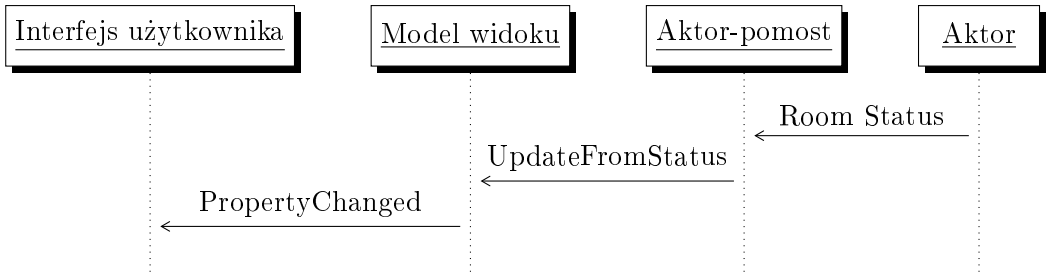


Rysunek 5.4: Schemat przekazywania wiadomości przez aktora-pomost

Po akcji użytkownika, model widoku przesyła wiadomość do aktora-pomostu, który przekazuje wiadomość dalej do przypisanego mu agenta. Czasami model widoku może oczekiwać odpowiedzi od systemu (tzw. Ask) jednak jest to widoczne tylko na poziomie widoku modelu i blokuje tylko wątek odpowiedzialny za obsługę danego kliknięcia/edycji.

Aktualizacja interfejsu użytkownika

Wiadomość od agenta z systemu przychodzi do agenta-pomostu. Ten, wywołuje przygotowaną w modelu widoku metodę do aktualizacji wartości wyświetlanych. Po wykonaniu metody uruchamia się zdarzenie aktualizujące wyświetlane wartości w interfejsie użytkownika.



Rysunek 5.5: Schemat aktualizowania interfejsu użytkownika przez aktora-pomost

5.5. Wiadomości typu Request/Update/Value

Do wygodnej obsługi odpytywania systemu oraz możliwości podmiany parametrów w symulatorze powstał schemat nazewnictwa wiadomości.

- Value - Wiadomość zawierająca wartość parametru.
- Request - Wezwanie do wysłania wartości danego parametru. Odbiorca powinien odesłać wiadomość typu Value.
- Update - Wiadmość zlecająca nadpisanie wartości danego parametru u odbiorcy na wartość podaną w wiadomości.

5.6. Obiekty typu Wymaganie/Praca/Zadanie

Obsługa oczekiwanych parametrów została podzielona na trzy etapy. Wymagania zostają rozbite na Zadania dla kontroleró poszczególnych parametrów, a następnie modele pozwalają wyliczyć najbardziej optymalną konfigurację urządzeń, zapisaną w obiekcie Pracy.

- Wymaganie - obiekt opisujący kiedy, który parametr i jaka jego wartość jest oczekiwana.
- Zadanie - obiekt opisujący ograniczenia w których musi być spełnione Wymaganie.

  Np. TemperatureTask to ilość czasu, obecna temperatura w pomieszczeniu i temperature jaką chcemy osiągnąć w przeciągu ww. czasu.
- Praca - obiekt określający najlepszą znalezioną konfigurację dla zadanego Zadania.

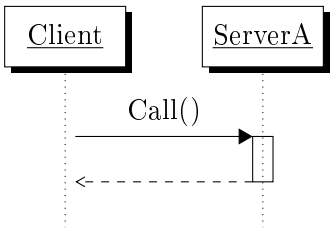
  Np. TemperatureJob składa się z informacji: w którym trybie, od kiedy do kiedy, i na jaką temperaturę nastawić klimatyzatory.

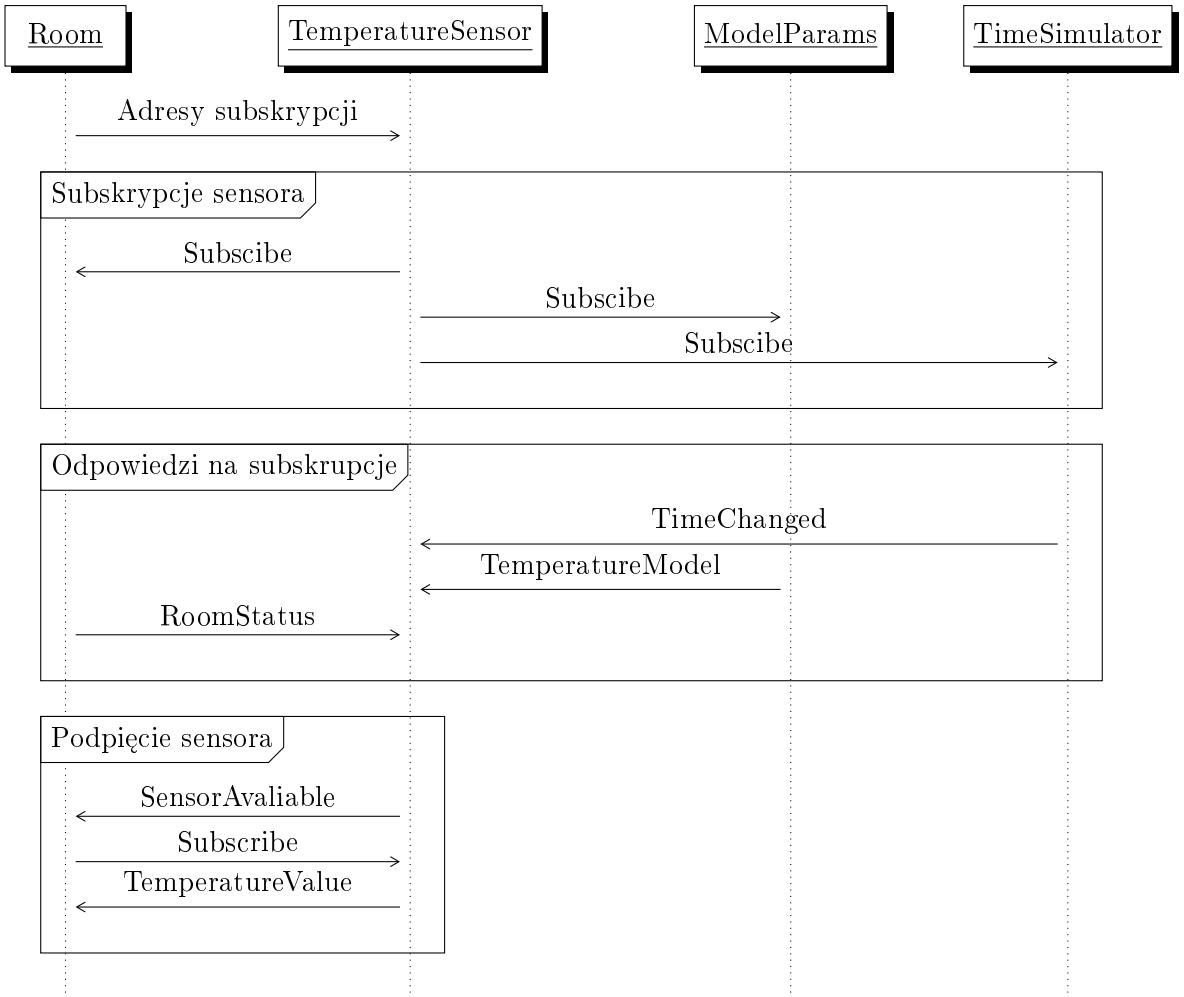
6. Interakcje między aktorami

6.1. Dodawanie/usuwanie pomieszczeń

Dodanie pomieszczenia

Linia tekstu opisująca diagram





Rysunek 6.1: Schemat dodawania symulatora sensora do pomieszczenia

Usunięcie pomieszczenia

6.2. Dodawanie/usuwanie sensorów i kontrolerów

Dodanie sensora

Usunięcie sensora

Dodanie kontrolera

Usunięcie kontrolera

6.3. Dodawanie/usuwanie urządzeń

Dodanie urządzenia

Usunięcie urządzenia

6.4. Upływ czasu

6.4.1. Sensor/symulator sensora

7. Testy

7.1. Testy w Akka.net

7.2. Pułapki związane z testowaniem aktorów

7.3. Scenariusze testowe

7.4. Testy symulacji upływu czasu

7.5. Testy przykładowego modelu temperatur

Część III

Podsumowania i wnioski

8. Wnioski i podsumowanie

8.1. Podsumowanie

8.2. Wnioski

9. Antywzorce w systemach wykorzystujących model aktorów

9.1. Wykorzystywanie eventów w akka.net

9.2. Nadmierne poleganie na kontenerach Dependency Injection

Bibliografia

[1] Dr R. Kuhn, *Orleans and Akka Actors: A Comparison* [dostęp: 07 XII 2017], https://github.com/akka/akka-meta/blob/master/ComparisonWithOrleans.md

[0] A. Aaaaa, *Tytuł*, Wydawnictwo, rok, strona-strona.

[0] J. Bobkowski, S. Dobkowski, *Blebleble*, Magazyn nr, rok, strony.

[0] C. Brink, *Power structures*, Algebra Universalis 30(2), 1993, 177-216.

[0] F. Burris, H. P. Sankappanavar, *A Course of Universal Algebra*, Springer-Verlag, New York, 1981.

Słownik

Definicja 9.1. Urządzenia HVAC

Definicja 9.2. Agent

Definicja 9.3. Aktor

Definicja 9.4. Sensor

Definicja 9.5. Aktuator

Definicja 9.6. Kontroler

Definicja 9.7. Komponent

Definicja 9.8. Aktor-pomost

Definicja 9.9. Symulator

Definicja 9.10. Aplikacja symulatora

Definicja 9.11. Scenariusz

Definicja 9.12. Wydarzenie

Definicja 9.13. Parametr

Definicja 9.14. Subskrypcja

Spis rysunków