POLITECHNIKA GDAŃSKA
Wydział Elektroniki, Telekomunikacji i Informatyki

Sławomir Nasiadka

Wpływ kontekstu na efektywność wykonania interaktywnych aplikacji iteracyjnych w dedykowanej przestrzeni usług

Rozprawa doktorska

Promotor:

prof. dr hab. inż. Henryk Krawczyk
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Gdańsk, 2012

Spis treści

SŁOWNIK SKRÓTÓW

4W1H - What, When, Where, Who, How,

AK – Analiza Kontekstu,

APA - Abstract Parallel Automata,

API – Application Programming Interface,

CBSE – Component Based Software Engineering,

CDM – Context-Driven Model,

CoBrA – Context Broker Architecture,

CORBA - Common Object Request Broker Architecture,

CSM - Concurrent State Machine,

CRCW - Concurrent Read, Concurrent Write,

CREW - Concurrent Read, Exclusive Write,

DA – Data Automata,

DFA - Deterministic Finite Automata,

DIND – Distributed Intelligent Network Device,

EREW - Exclusive Read, Exclusive Write,

GPS – Global Positioning System,

HMI - Human-Machine Interaction,

HTTP - Hypertext Transfer Protocol,

FA – Finite Automata,

IAI – Interaktywna Aplikacja Iteracyjna,

IAK – Interaktywna Architektura Komponentowa,

IP – Internet Protocol,

ISO - International Organization for Standardization,

ITS – Intelligent Transportation System,

LK – model, w którym logika analizy kontekstu znajduje się w PIAI,

LS – model, w którym logika analizy kontekstu znajduje się po stronie sensorów,

MIMD - Multiple Instruction - Multiple Data,

MISD - Multiple Instruction – Single Data,

MVC – Model – View – Controller,

MVP – Model – View – Presenter,

NFA – Nondeterministic Finite Automata,

OSI – Open System Interconnection,

OWL – Web Ontology Language,

P2P – Peer to Peer,

PAC – Presentation – Abstraction – Control,

PC – Personal Computer,

PDA - Personal Digital Assistant,

PDA – Push-Down Automata,

PIAI – Parallel IAI,

PRAM – Parallel Random Access Machine,

PSS – Personal Smart Space,

RA – Register Automata,

RAM - Random Access Machine,

RFID - Radio-frequency identification,

RMI – Remote Method Invocation,

RTS – Real Time System,

SIMD - Single Instruction – Multiple Data

SISD - Single Instruction – Single Data,

SN – Sensor Network,

SOA – Service Oriented Architecture,

SOAF - Service-Oriented Adaptative Framework,

SOAP - Simple Object Access Protocol,

SOC – Service Oriented Computing,

SOCAM - Service-Oriented Context-Aware Middleware,

SOM – Self Organizing Maps,

SOM – Service-Oriented Model,

SPARQL - Simple Protocol And RDF Query Language,

SSB – Soft System Bus,

TA – Timed Automata,

TCSM - Timed Concurrent State Machine,

ι_{ZD} – instrukcja zbierania danych,

ι_{TDK} – instrukcja transformacji danych kontekstowych,

ι_{AK} – instrukcja analizy kontekstu,

ι_{DA} – instrukcja doboru akcji,

ι_{TDA} – instrukcja transformacja danych dla akcji,

ι_{UA} – instrukcja aruchomienia akcji,

WSCDL – Web Service Choreography Description Language,

WSN - Wireless Sensor Networks,

WSAN - Wireless sensor and actuator networks,

WWW – World Wide Web,

XML – Extensible Markup Language.

SŁOWNIK OZNACZEŃ

2^X - zbiór wszystkich podzbiorów zbioru X, gdzie $X = \{P, PK, Z, SK, OS\}$,

$\mathbb{R}$ - zbiór liczb rzeczywistych,

i, j, l, m - indeksy,

π - płaszczyzna,

Algorytm iteracyjny

$AI = \langle X, V, v_S, v_E, Q \rangle$ – klasyczny algorytm iteracyjny,

V - skończony zbiór etykiet (symboli) algorytmu AI,

v_S – etykieta początkowa algorytmu AI,

v_E – etykieta końcowa algorytmu AI,

v_P – etykieta początkowa instrukcji w algorytmie AI,

v_K – etykieta końcowa instrukcji w algorytmie AI,

$\Theta \subset X \times X$ - relacja binarna, reprezentująca część operacyjną instrukcji algorytmu AI,

X - zbiór obiektów algorytmu AI,

Q – zbiór instrukcji algorytmu AI,

Przestrzeń inteligentna

$PI = \langle O, P, \alpha, U \rangle$ - przestrzeń inteligentna,

$O = \{o_1, o_2, \dots, o_{mO}\}$ – zbiór obiektów przestrzeni, w której działa aplikacja,

mO – liczba obiektów w zbiorze O,

$P = \{p_1, p_2, \dots, p_{mP}\}$ – parametry obiektów O,

mP - liczba obiektów w zbiorze P,

$U = \{u_1, u_2, \dots, u_{mU}\}$ – zbiór usług w przestrzeni,

mU – liczba obiektów w zbiorze U,

$o_i \alpha P_i$ – relacja przyporządkowania podzbioru parametrów P_i do obiektu o_i,

Stan przestrzeni inteligentnej

$SP = \{sp_1, sp_2, \dots, sp_{mSP}\}$ - zbiór stanów przestrzeni PI,

mSP - liczba elementów zbioru SP,

$sp, sp_l = \langle PI, W_l, \beta \rangle$ - stan przestrzeni PI,

$p_{i_j} \beta w_{l_{i_j}}$ – relacja przyporządkowania wartości $w_{l_{i_j}}$ do parametrów p_{i_j},

$W = \{w_1, w_2, \dots\}$ – zbiór wartości jakie mogą przyjmować parametry P przypisane do

obiektów przestrzeni (zbiór nieskończony),

Kontekst

$K = \{k_1, k_2, \dots, k_{mK}\}$ – zbiór kontekstów,

mK - liczba obiektów w zbiorze K,

$k, k_l = \langle OK_l, PK_l, \alpha_K, WK_l, \beta_K \rangle$ - kontekst,

$OK_l \subseteq O$ – obiekty należące do kontekstu,

$PK_l \subseteq P$ – parametry opisujące obiekty należące do kontekstu,

$ok_{l_i} \alpha_K PK_{l_i}$ – relacja przyporządkowania podzbioru parametrów PK_{l_i} do obiektu ok_{l_i},

$WK_l \subseteq W$ – zbiór wartości parametrów należących do kontekstu,

$pk_{l_{i_j}} \beta_K wk_{l_{i_j}}$ - relacja przyporządkowania wartości $wk_{l_{i_j}}$ do parametru $pk_{l_{i_j}}$,

K_{IAI} – zbiór kontekstów aplikacji IAI,

K_{WA} – zbiór kontekstów wywołania akcji,

K_A – zbiór kontekstów wykonania akcji,

Zdarzenia

$Z = Z_A \cup Z_Z$ – zbiór zdarzeń wewnętrznych i zewnętrznych,

$Z_A = \{za_1, za_2, \dots, za_{mZA}\}$ - zbiór zdarzeń wewnętrznych będących wynikiem wykonania akcji aplikacji IAI,

$Z_Z = \{zz_0, zz_1, \dots, zz_{mZZ}\}$ - zbiór zdarzeń zewnętrznych,

mZA - liczba elementów zbioru Z_A,

mZZ - liczba elementów zbioru Z_Z,

c - liczba jednocześnie pojawiających się w przestrzeni PI zdarzeń zewnętrznych,

d - liczba jednoczesnych zdarzeń wewnętrznych (będących wynikiem wykonania akcji aplikacji IAI),

$\zeta: (SP, 2^Z) \rightarrow SP$ - funkcja zmiany stanu przestrzeni działania aplikacji,

Aplikacja IAI

$IAI = \langle OS, A, \gamma \rangle$ – definicja aplikacji IAI,

$OS \supset (SK_{WA} \cup \emptyset)$ – zbiór oczekiwanych kontekstów wywołania akcji,

os_l - oczekiwany kontekst wywołania akcji,

$\gamma: OS \rightarrow A$ – funkcja przyporządkowania akcji do oczekiwanego kontekstu wywołania akcji,

$A = \{a_1, a_2, \dots, a_{mA}\}$ - zbiór akcji aplikacji IAI,

mA - liczba elementów zbioru A,

$WR = \{wr_1, wr_{2,} \dots, wr_{mWR}\}$ - zbiór warunków,

mWR – liczba elementów w zbiorze WR,

$OZ = \{oz_1, oz_2, \dots, oz_{mOZ}\}$ – zbiór oczekiwanych warunków wywołania akcji,

mOZ - liczba elementów zbioru OZ,

$oz_l = \langle OK_l, PK_l, \alpha_K, WR_l, \varrho \rangle$ - oczekiwane warunki wywołania akcji,

$\left(ok_{l_i} \times pk_{l_j}\right) \varrho wr_{l_m}$ - relacja przypisania warunku do parametru obiektu kontekstu,

$OS_l \lambda oz_l$ – relacja grupowania oczekiwanych kontekstów wywołania akcji,

$\sigma: OZ \rightarrow A$ - funkcja przyporządkowująca akcje do oczekiwanych warunków wywołania akcji,

ψ - wielkość kontekstu,

$\mu = |OZ|$ - miara określająca stopień interaktywności aplikacji IAI,

Automat opisujący aplikację IAI

Σ – alfabet wejściowy automatu FA bądź NFA,

ε – symbol reprezentujący przejście spontaniczne w automacie,

ω-automaty – automaty NFA z możliwością rozpoznawania nieskończonych słów,

$\delta: OZ \rightarrow \Sigma^*$ – funkcja zmiany języka wejściowego automatu dla aplikacji typu IAI,

T_O – przedział czasu reprezentujący ograniczenie czasowe na wyszukanie i rozpoczęcie wykonywania akcji,

c_{IAI} – zegar w automacie dla aplikacji typu IAI,

$\Sigma^* = \{oz_b, oz_f\}$ – alfabet wejściowy dla automatu bazującego na TA, opisującego aplikację typu IAI,

Maszyna PIAI

$(+)$ - operacja dodawania kontekstów,

r – stopień rozproszenia logiki analizy kontekstu,

ψ_{PIAI} - wielkość kontekstu, który jest analizowany w PIAI,

T_{IAI} – przedział czasu reakcji aplikacji typu IAI,

T_{PIAI} – przedział czasu reakcji maszyny PIAI,

T_Z – rzeczywisty czas (przedział) reakcji aplikacji IAI,

T_{AK} – przedział czasu analizy kontekstu w PIAI dla pojedynczej danej kontekstowej,

T_{DA} – przedział czasu doboru (wyszukania) akcji w PIAI,

T_S – przedział czasu analizy pojedynczej danej z sensora w PIAI,

T_{TDA} – przedział czasu transformacji danych w PIAI dla wywołania akcji,

T_{TDK} – przedział czasu transformacji pojedynczej danej kontekstowej przekazanej od sensora do PIAI,

T_{UA} – przedział czasu uruchomienia akcji w PIAI,

T_{ZD} – przedział czasu zbierania pojedynczej danej kontekstowej przez PIAI,

$\tau_{UA}(a)$ - funkcja przypisująca akcji całkowity przedział czasu jej uruchamiania (tzn. przedział czasu od początku do końca uruchamiania akcji),

t - czas,

$t^{'}$ - moment czasu zakończenia wykonywania akcji,

$t_Z^{'}$ - moment czasu pojawienia się w przestrzeni inteligentnej zdarzenia, które powoduje zaistnienie kontekstu spełniającego oczekiwane warunki wywołania akcji,

$t_{ZD}^{'}$ - moment czasu rozpoczęcia czynności zbierania danych w ramach maszyny PIAI dla zdarzenia, które powoduje zaistnienie kontekstu spełniającego oczekiwane warunki wywołania akcji

$t_{AK}(z)$ – funkcja przypisująca moment czasu rozpoczęcia analizy kontekstu (AK) dla danych pochodzących ze zdarzenia z do tego zdarzenia,

$t_{DA}(oz)$ - funkcja przypisująca moment czasu rozpoczęcia wykonywania instrukcji doboru akcji (ι_{DA}) w maszynie PIAI dla rozpoznanych oczekiwanych warunków wywołania akcji oz do tych warunków,

$t_{UA}(oz)$ - funkcja przypisująca moment czasu rozpoczęcia wykonywania instrukcji uruchomienia akcji (ι_{UA}) w PIAI dla rozpoznanych oczekiwanych warunków wywołania akcji oz do tych warunków,

$t_Z(z)$ – funkcja przypisująca moment czasu pojawienia się zdarzenia z do tego zdarzenia,

$t_{ZD}(z)$ - funkcja przypisująca moment czasu rozpoczęcia wykonywania instrukcji ι_{ZD} dla danych pochodzących ze zdarzenia z do tego zdarzenia,

s - liczba danych kontekstowych analizowanych w PIAI.

I WPROWADZENIE

Zakres tematyki poruszonej w rozprawie dotyczy aplikacji kontekstowych wykonywanych w środowisku czasu rzeczywistego typu *pervasive computing*, zwanym przestrzenią inteligentną. Można je zamodelować (w odniesieniu do analizy kontekstu, reprezentującego sytuację w przestrzeni, a także wykonywania akcji dostosowujących działanie do określonych sytuacji) jako Interaktywną Aplikację Iteracyjną (IAI). Model takiej aplikacji zakłada, iż kontekst jest analizowany w sposób ciągły a w przypadku, gdy spełnia określone warunki następuje wykonanie wskazanej w definicji aplikacji akcji. Na efektywność środowiska odpowiedzialnego za wykonywanie aplikacji IAI, ma wpływ wiele czynników, w tym efektywność analizy kontekstu oraz efektywność wykonywania związanej z rozpoznaną sytuacją akcji. Stanowi to punkt wyjścia dla analizy efektywności zaimplementowanego i analizowanego w rozprawie środowiska wykonywania zbudowanego zgodnie z opracowaną architekturą takiego środowiska.

I.1. Tematyka rozprawy

Przestrzenie inteligentne są zagadnieniem szeroko opisywanym w literaturze (np. [11][15][19][24][50][51][77][78][82]). Ich architektura jak i sposoby realizacji są wciąż badane, rozwijane i dokumentowane. Liczba aplikacji, które w ramach nich działają (tzw. aplikacje przetwarzania wszechobecnego) wciąż rośnie tak, iż obejmują one zasięgiem swojego działania prawie wszystkie sfery życia człowieka, który jest jednym z użytkowników przestrzeni. Jednakże zasady, według których te aplikacje funkcjonują, pozostają niezmienne. Przede wszystkim muszą być interaktywne, czyli powinny wchodzić w interakcję z otaczającymi je innymi użytkownikami przestrzeni (aplikacjami, ludźmi, robotami). Z drugiej strony wyraźnie zarysowuje się iteracyjny charakter ich działania, oznaczający ciągłe analizowanie sytuacji mających miejsce w przestrzeni (kontekstu) i odpowiednie na nie reagowanie. Zatem aplikacje działające w przestrzeniach inteligentnych są aplikacjami kontekstowymi, tzn. dostosowującymi swoje działania do sytuacji mających miejsce w ich otoczeniu. Dodatkowo w przypadku aplikacji, funkcjonujących w przestrzeniach inteligentnych (nazywanych także przestrzeniami typu *pervasive*), istotny jest aspekt działania w czasie rzeczywistym. Odnosi się on zarówno do fazy sensorycznej (zbierania danych kontekstowych), fazy analizy jak i fazy wykonywania działań. Zarysowuje się zatem potrzeba utworzenia nowego modelu aplikacji dla przestrzeni inteligentnych (posiadającej wymienione cechy) IAI - Interaktywnej Aplikacji Iteracyjnej, a wraz z nim modelu architektury środowiska do jej wykonywania. Mając takie formalne modele, można przeprowadzić ich jakościową analizę pod kątem wpływu kontekstu na wykonanie aplikacji. Analiza ta może zostać wykonana jedynie w zakresie

badania samych modeli, ponieważ każda z aplikacji, działających w przestrzeniach inteligentnych jak i same przestrzenie, jest inna (zarówno w zakresie rozpoznawania kontekstu jak i sposobu reakcji na zmiany w otoczeniu).

Podstawowymi problemami poruszonymi w rozprawie jest zidentyfikowanie własności aplikacji IAI, zdefiniowanie ich modelu oraz opracowanie modelu maszyny (PIAI - Parallel IAI), która mógłby posłużyć do zbudowania środowiska wykonywania aplikacji, w taki sposób aby wspomóc je w procesie analizy kontekstu i wykonywania działań adaptacyjnych (do nowej sytuacji w przestrzeni). Na bazie modeli, w rozprawie została zaproponowana architektura takiego środowiska (IAK - Interaktywna Architektura Kontekstowa) oraz przedstawiona przykładowa jego implementacja (SWIAI - Środowisko Wykonywania Interaktywnych Aplikacji Iteracyjnych). Ostatnim zagadnieniem, na którym skupia się rozprawa, jest zidentyfikowanie parametrów, od których w największym stopniu zależy efektywność (rozumiana jako czas reakcji) aplikacji IAI. W celu ujednolicenia rozważań opracowano zarówno słownik pojęć, zawierający podstawowe definicje, na bazie których prowadzone są rozważania, jak też zbudowano uniwersalną przestrzeń inteligentną do realizacji określonych badań eksperymentalnych.

I.2. Tezy rozprawy

Najważniejszymi wyzwaniami, jakie pojawiają się podczas wytwarzania aplikacji IAI, są przygotowanie opisu tego rodzaju aplikacji, utworzenie środowiska ich wykonania oraz zapewnienie odpowiedniego poziomu ich jakości. Pierwsze z nich wiąże się z opracowaniem definicji aplikacji działającej w przestrzeni inteligentnej uwzględniającej aspekty interaktywności (w odniesieniu do współpracy z przestrzenią i jej użytkownikami), kontekstowości i iteracyjności. W tym celu została opracowana ogólna definicja kontekstu, wykorzystana następnie do opisu sytuacji mających miejsce w przestrzeni, na których pojawienie się aplikacja ma reagować uruchomieniem akcji (tzw. akcji adaptacyjnej). Akcje są skojarzone z określonym (tzw. oczekiwanym) kontekstem i są zamodelowane jako pojedyncza usługa SOA, mogąca wykonywać działania w ramach przestrzeni (np. powodując nową sytuację, bądź wchodząc w interakcję z jej użytkownikami). W konsekwencji aplikacje typu IAI definiowane są jako zestaw par: oczekiwany kontekst wywołania akcji – akcja (usługa), co powoduje, iż ich wykonanie ma charakter iteracyjny. Oznacza to, że musi istnieć odpowiednie środowisko wykonywania takich aplikacji realizujące funkcjonalność rozpoznawania kontekstu (analizy danych docierających z sensorów) oraz uruchamiania odpowiedniej akcji.

W odniesieniu do aplikacji IAI, jakość można zdefiniować na dwóch płaszczyznach: statycznej i dynamicznej. Jakość w ujęciu statycznym odnosi się do sposobu opisu

aplikacji (zasad jej działania – np. kodu źródłowego), struktury analizowanych danych (np. sposobu definiowania kontekstu oraz warunków określających kiedy wykonać akcję adaptacyjną), a także jej zrozumiałości. Płaszczyzna dynamiczna związana jest z działaniem aplikacji IAI, w odpowiednio skonstruowanym środowisku (przestrzeni inteligentnej dotyczącej określonej dziedziny zastosowań, w ramach której działają określone usługi i silnik umożliwiający wykonywanie aplikacji IAI – tzw. dedykowanej przestrzeni usług) i na niej skupia się niniejsza rozprawa. Jakość tego działania można zdefiniować między innymi jako efektywność wykonania aplikacji IAI (co bezpośrednio przekłada się na efektywność działania środowiska wykonywania), określoną przez czas reakcji aplikacji, na zachodzące w przestrzeni zdarzenia (powodujące zaistnienie określonej sytuacji). Czas reakcji aplikacji może uwzględniać efektywność i skuteczność wykonywanych usług dostępnych w przestrzeni wykonania tej aplikacji, jednakże w rozprawie nie badano tego aspektu ze względu na to, że środowisko wykonywania aplikacji IAI nie ma wpływu na własności tych usług.

Celem rozprawy jest zaprojektowanie, zamodelowanie, zaimplementowanie oraz jakościowa analiza środowiska, umożliwiającego wykonywanie Interaktywnych Aplikacji Iteracyjnych w dedykowanej przestrzeni usług. Z celem tym związane są następujące tezy:

1. Aplikację IAI można opisać jako maszynę równoległą (PIAI), zbudowaną na bazie maszyny PRAM.
2. Maszyna PIAI stanowi bazę do budowy przestrzeni inteligentnej zawierającej środowisko wykonywania aplikacji IAI (SWIAI) z wydzielonym zbiorem instrukcji i usług wspomagających działalność tej przestrzeni.
3. Efektywność działania (czas reakcji) SWIAI zależy od wielkości kontekstu, rozproszenia jego analizy pomiędzy maszynę i usługi znajdujące się przestrzeni inteligentnej, a także od liczby zdarzeń jednocześnie zachodzących w przestrzeni.

W celu wykazania powyższych tezy należy określić własności aplikacji IAI oraz zamodelować ją jak i środowisko jej wykonania. Dlatego też, główne zadania realizowane w ramach pracy doktorskiej polegają na zdefiniowaniu pojęcia kontekstu i zbudowaniu na jego podstawie modelu aplikacji IAI oraz modelu maszyny PIAI, opisującej jej wykonanie. Na bazie tej maszyny opracowano architekturę IAK (Interaktywna Architektura Komponentowa), zgodnie z którą mogą być budowane środowiska wykonywania aplikacji IAI. Określono również reprezentatywne usługi w ramach IAK (odpowiadające jej komponentom) niezbędne do realizacji takiej aplikacji, a także usługi jakie muszą istnieć w przestrzeni inteligentnej. Przeprowadzono badania empiryczne ukierunkowane na pomiary szybkości reakcji środowiska (odzwierciedlającej efektywność wykonania

aplikacji IAI) na zdarzenia powodujące zaistnienie określonej sytuacji w zależności od wielkości kontekstu, liczby zdarzeń zewnętrznych, jakie zachodzą w przestrzeni podczas wykonywania aplikacji IAI oraz stopnia rozproszenia logiki analizy kontekstu pomiędzy SWIAI a usługi znajdujące się w przestrzeni inteligentnej. Badania odnosiły się do poszczególnych instrukcji maszyny PIAI wykonywanych podczas analizy danych kontekstowych. Na tej podstawie wykazano, że analiza danych kontekstowych (obejmująca analizę i przetwarzanie kontekstu tworzonego z danych kontekstowych) stanowi główny czynnik, obok efektywności wykonania akcji, istotnie wpływający na szybkość wykonania aplikacji IAI w ramach IAK (jej implementacji). Rozprawa ma zatem charakter teoretyczno-doświadczalny.

Niniejsza rozprawa doktorska w drugim rozdziale opisuje koncepcję przestrzeni inteligentnych oraz kierunki ich rozwoju. Rozdział 3 zawiera informacje na temat istotnych własności aplikacji IAI, której nieformalny opis i formalna definicja są następnie wprowadzone w rozdziale 4 (teza 1 rozprawy). Rozdział 5 opisuje maszynę PIAI będącą podstawą do opracowania architektury IAK dla środowiska wykonywania aplikacji IAI. Jego implementacja (SWIAI), która może zostać umieszczona w przestrzeni inteligentnej, została opisana w rozdziale 6 (teza 2 rozprawy). Na bazie tej implementacji wykonano badania eksperymentalne (opisane w rozdziale 7), służące do potwierdzenia tezy 3 rozprawy doktorskiej. W rozdziale 8 zawarto uwagi końcowe oraz przedstawiono problemy otwarte wymagające dalszych badań.

II KIERUNKI ROZWOJU PRZESTRZENI INTELIGENTNYCH

Projektowanie aplikacji, wykorzystujących idę przetwarzania wszechobecnego, stawia przed ich twórcami zupełnie nowe wyzwania, jakie nie były wcześniej spotykane w informatyce. Przede wszystkim aplikacje wszechobecne muszą być świadome swojego otoczenia (przestrzeni w jakiej działają). Oznacza to konieczność adaptacji do zmieniających się warunków i sytuacji - aplikacje nie mogą ograniczać się jedynie do analizy danych wejściowych, które są przekazywane bezpośrednio przez użytkownika, ale też reagować na zmieniający się stan środowiska, w którym ten użytkownik się znajduje. Świadomość stanu otoczenia musi także przekładać się na czynne (inicjowane przez aplikacje) działania co oznacza, że muszą one mieć możliwość przewidywania zachowań człowieka i reagować na nie z wyprzedzeniem. Środowiska funkcjonowania takich aplikacji określane są mianem przestrzeni inteligentnych, którym poświęcony jest ten rozdział pracy doktorskiej.

II.1. Współczesne kategorie przetwarzania komputerowego

Przestrzenie inteligentne są środowiskami, w ramach których wykonywane jest różnego rodzaju przetwarzanie danych dostępnych dla aplikacji w nich działających. Ich (przestrzeni inteligentnych) ideę sformułował już w 1991 roku M. Weiser w pracy [139] – poprzez sprecyzowanie wizji przetwarzania wszechobecnego (*ubiquitous computing*). Według niego przyszłością systemów komputerowych miała być transparentna integracja z otoczeniem człowieka, tak aby systemy te były „zanurzone" w środowisku, w jakim się on porusza. Dzięki temu przetwarzanie danych przez komputery miało stać się tak powszechne jak elektryczność [62]. Jak się później okazało, jego wizja stała się nowym paradygmatem w zakresie budowania aplikacji nowoczesnych [117] a koncepcja przetwarzania wszechobecnego została określona mianem trzeciej fali technologii komputerowej, przy czym:

- pierwszą falę stanowią komputery obliczeniowe typu *mainframe* (z jednego komputera korzysta poprzez terminale wiele osób),
- druga fala związana jest z komputerami osobistymi (PC). W tym przypadku jeden komputer używany jest przez jedną osobę i wymaga ciągłej (i świadomej) interakcji z jej strony,

- trzecia fala dotyczy przetwarzania wszechobecnego (*ubiquitous*). Tworzą ją systemy rozproszone, gdzie z człowiekiem współpracuje wiele komputerów, umieszczonych w sposób niezauważalny w jego środowisku.

Zgodnie z wizją M. Weiser'a człowiek nie powinien być ograniczony do korzystania z aplikacji komputerowych w jednym określonym miejscu. Nawet urządzenia typu laptop nie zapewniają swobodnego dostępu do funkcjonalności przez nie oferowanych każdym miejscu i w każdym czasie. Co więcej komputery i działające w nich aplikacje wymagają nieustannej interakcji ze strony człowieka. Według M. Weiser'a powinny być one obecne wszędzie tam gdzie człowiek wykonuje swoje codzienne czynności (tzn. wszechobecne) i aktywnie go w nich wspomagać, jednakże nie mogą przy tym wymagać od niego ciągłego nadzoru. Na przykład każda kartka papieru, czy każda notatka może mieć formę takiej interaktywnej aplikacji, dzięki której łatwiej będzie człowiekowi znaleźć zapisane wcześniej informacje.

Koncepcją będącą pochodną wizji przetwarzania wszechobecnego jest *pervasive computing* [72]. Oznacza ona dostęp do funkcjonalności i informacji (udostępnianych przez aplikacje) przy wykorzystaniu urządzeń mobilnych oraz mechanizmów wspierania grup użytkowników i przetwarzania kontekstu. Z tego punktu widzenia istotny jest sposób, w jaki użytkownicy wykorzystują urządzenia do interakcji ze środowiskiem. Z jednej strony ważne jest by człowiek nie musiał podejmować dodatkowego wysiłku (związanego z ich wykorzystaniem), a z drugiej miał możliwość skorzystania z oferowanych przez nie funkcjonalności w każdym miejscu i w każdym czasie. Istnieją trzy główne pola, na których koncentrują się systemy typu *pervasive* [62]:

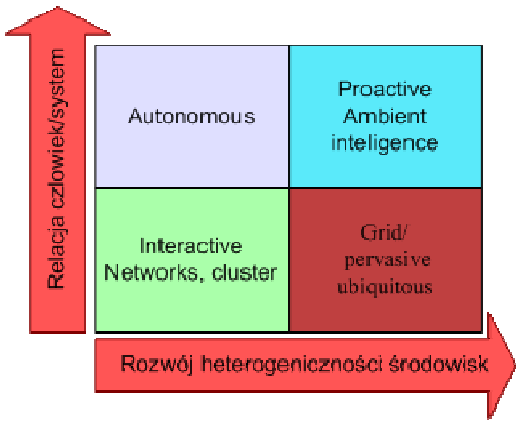
- sposób wykorzystania urządzeń mobilnych przez użytkowników do realizacji swoich zadań (z uwzględnieniem sytuacji w środowisku, w jakim się znajdują),
- sposób wytwarzania aplikacji umożliwiających realizację tych zadań,
- sposób, w jaki środowisko jest wzbogacone i rozwijane poprzez udostępnianie nowych informacji i nowych funkcjonalności.

Najważniejszą różnicą tego rodzaju systemów w stosunku do *ubiquitous computing* jest ich koncentracja na urządzeniach mobilnych (i sposobie ich wykorzystania) tak, aby wykonać zadania istotne z punktu widzenia użytkownika.

Trzecim z często wymienianych pojęć w kontekście zarówno *ubiquitous computing* jak i *pervasive computing* jest *ambient intelligence* [121]. Skupia się ono na wykorzystaniu technologii komunikacyjnych, aby uczynić życie prostszym [62]. *The Information Society Technologies Advisory Group* w 6 programie ramowym stwierdza, iż *ubiquitous*

computing jest jedną z kluczowych technologii potrzebną do realizacji koncepcji *ambient intelligence* i że realizacja zawartej w niej wizji jest możliwa jedynie wtedy, gdy istnieją sieci typu *pervasive networks* [113]. Wszystkie te trzy pojęcia są do siebie bardzo zbliżone, do tego stopnia, iż w literaturze można zauważyć stopniowe zacieranie się różnic między nimi. Jednakże nie oznaczają dokładnie tego samego – każde z nich skupia się na pewnej wizji przetwarzania wszechobecnego, jednakże z trochę innej perspektywy.

Przetwarzanie wszechobecne (w rozumieniu opisanych w poprzednich akapitach terminów) należy zatem rozpatrywać w szerokim kontekście. Jego podstawą są systemy rozproszone, których koncepcja opiera się na wykorzystaniu mocy obliczeniowej wielu autonomicznych maszyn (połączonych siecią komputerową) w celu zwiększenia szybkości lub wiarygodności wykonania aplikacji. Jest ono przeciwieństwem podejścia scentralizowanego, gdzie wszystkie operacje wykonywane w ramach aplikacji były realizowane fizycznie w obrębie pojedynczej maszyny. Tendencję rozwojową systemów rozproszonych przedstawiono na rysunku 1.



Rysunek 1 Tendencja rozwojowa systemów rozproszonych

Zbudowane na nich środowiska wykonywania nowoczesnych aplikacji odeszły od zamkniętych środowisk obliczeniowych (klastry) i niezależnego przetwarzania w wyizolowanych jednostkach do środowisk typu *grid* ze współdzieleniem (między nimi) zasobów. Ponadto, uwzględniając szybki rozwój różnego typu systemów wbudowanych oraz mobilności różnego typu urządzeń dedykowanych, wyłania się nowy typ środowiska, w którym sensory, komputery i urządzenia są wszechobecne w otaczającym środowisku, choć pozostają niewidoczne dla człowieka. Cechą istotną tych ostatnich jest to, że aplikacje współdziałają ze środowiskiem bardziej naturalnie i wykorzystują zarówno możliwości techniczne (wnoszone przez urządzenia) jak i socjalne (wnoszone przez ludzi). Zatem kluczowe dla realizacji wizji przetwarzania wszechobecnego są zarówno rozproszenie jak i mobilność [84][33], z tym że w jej przypadku podstawowy cel aplikacji

ją wykorzystujących – dostarczanie funkcjonalności „gdziekolwiek w każdym czasie" (*anytime anywhere*), reprezentujący podejście bierne, zmienił się na cel „w każdym miejscu cały czas" (*all the time everywhere*), odpowiadający podejściu czynnemu. Środowisko i przetwarzanie *ambient intelligence* obejmują właśnie wykorzystanie wszechobecnych i przenośnych urządzeń wraz z inteligentnymi algorytmami przetwarzania zdarzeń oraz metodami uczenia się, dzięki którym takie (czynne) podejście może zostać zastosowane. Tak więc, rozwój autonomiczności aplikacji i wyposażenie środowiska w różnego typu inteligentne urządzenia wbudowane kreuje nowy model obliczeń, tzw. *proactive computing* [138]. Środowisko fizycznie przejmuje na siebie rolę aktywnego analizowania i przetwarzania pojawiających się w nim informacji. W ten sposób rodzi się potrzeba autonomicznych obliczeń [106], co nie wyklucza istotnej roli człowieka, która jest niezbędna do określania scenariuszy dla nieznanych i nowych sytuacji. Takie środowisko, wzbogacone o mechanizmy przetwarzania wszechobecnego (w znaczeniu *ambient intelligence*), nazywane jest właśnie przestrzenią inteligentną i stanowi główny nurt rozważań rozprawy doktorskiej.

II.2. Koncepcja przestrzeni inteligentnych

Rozwój przestrzeni inteligentnych (*inteligent space, smart space, pervasive space*) trwa nieprzerwanie od parunastu lat. Jedną z ich pierwszych realizacji było laboratorium *Hashimoto Lab*, na Uniwersytecie w Tokyo w Japonii (nazwane *iSpace*) [77]. Od tamtego czasu zmieniło się podejście do realizacji samej przestrzeni jak i zostały zdefiniowane jej nowe funkcje oraz sposoby wykorzystania [79]. Inteligentne przestrzenie zaczęły być wdrażane w postaci inteligentnych pokojów (np. *iRoom* [130]), budynków [89] (dostosowujących się do zachodzących wokół nich zmian i potrafiących sprostać potrzebom znajdujących się w nich ludzi [24][110]) i innych przestrzeni życiowych [55][151]. Pierwotnie działania te ukierunkowane były na fabryki i zakłady pracy, gdzie stosunkowo łatwo można wydzielić funkcje, jakie przestrzeń ma realizować. Przykład takiego wdrożenia opisany jest pracy [75], gdzie zaprezentowano inteligentne pojazdy, które potrafią przemieszczać się bezkolizyjnie w środowisku zakładu przemysłowego. W miarę rozwoju technologii rozpoczęto realizować bardziej śmiałe wizje - na przykład w zakresie systemów transportowych (tzw. ITS – *Intelligent Transportation System*) [135][136][154], w wyniku których zarówno pojazdy poruszające się po drogach publicznych (np. autostradach) jak i same drogi (będące dla nich środowiskiem działania) stają się coraz bardziej inteligentne. Samochody wyposażone są w różnego rodzaju czujniki (np. GPS, parkowania) przez co mogą wchodzić w interakcję z otaczającą je przestrzenią i korzystać z informacji w niej zawartych. Powyższe przykłady pokazują, iż w przestrzeniach inteligentnych ludzie, maszyny i usługi współistnieją obok siebie.

II.2.1. Definicja przestrzeni inteligentnej

Przestrzeń inteligentna jest to przestrzeń funkcjonowania człowieka i aplikacji informatycznych [79] (np. pomieszczenie, budynek, ulica), stanowiąca realizację koncepcji przetwarzania wszechobecnego *ubiquitous/pervasive computing*. Jej głównym celem jest dostarczenie człowiekowi środowiska, w którym będzie mógł swobodnie działać, a jego działania będą efektywnie wspomagane [80] (inaczej niż w klasycznych systemach typu *computer – centered*, przestrzeń jest ukierunkowana na człowieka tzn. *human – centered* [154][160]). Aplikacje, działające w takiej przestrzeni, są definiowane bardzo ogólnie jako określona funkcjonalność, zatem mogą być traktowane jako usługi w znaczeniu SOA [8] (z punktu widzenia przestrzeni nie jest istotna ich wewnętrzna architektura, lecz jedynie dostarczane funkcje oraz sposób współpracy z innymi elementami przestrzeni). Przykłady sposobów konstrukcji tego rodzaju aplikacji zostały zawarte w pracach [97] i [98]. Ponieważ działają one w przestrzeni, stają się jednocześnie jej elementami i mogą być przez nią wykorzystywane do wspierania jej użytkowników. Użytkownicy przestrzeni to ludzie, roboty oraz aplikacje (które mogą być także zrealizowane przy wykorzystaniu technik agentowych [58][85]). W dalszej części rozprawy roboty będą zaliczane do aplikacji (tzn. użytkownikami przestrzeni są ludzie oraz aplikacje). Przestrzeń inteligentna jest to zatem przestrzeń, oferująca usługi funkcjonującym w niej użytkownikom. Według pracy [131] można ją scharakteryzować następująco:

- przestrzeń inteligentna jest stworzona dla człowieka, który może w jej ramach wykonywać zwykłe, codzienne czynności,
- przestrzeń inteligentna powinna automatycznie rozpoznawać i być świadoma zachodzących w niej zdarzeń i aktywności jej użytkowników,
- przestrzeń inteligentna powinna reagować na określone zdarzenia,
- przestrzeń inteligentna powinna być skalowalna i posiadać możliwość adaptowania się do dynamicznie zachodzących zmian.

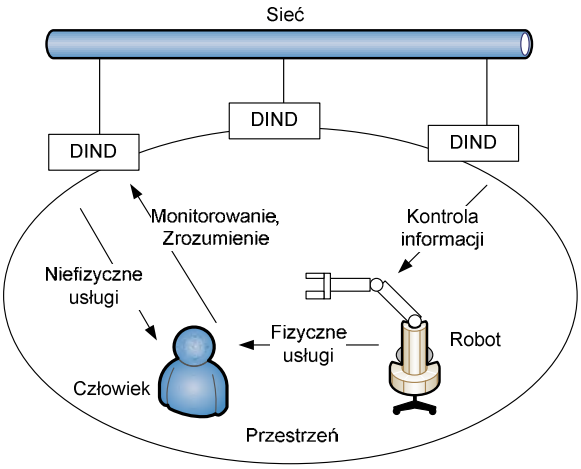
Rozszerzenia względem przedstawionej definicji przestrzeni wprowadza pojęcie *smart space* [44]:

- przestrzeń inteligentna (w znaczeniu *smart space, pervasive space*) jest to fizyczne środowisko, w którym znajdują się urządzenia ogólnego zastosowania (które nie muszą wchodzić bezpośrednio w interakcję z użytkownikami przestrzeni),
- przetwarzanie danych pochodzących z przestrzeni odbywa się w sposób rozproszony,
- funkcjonujące w ramach przestrzeni aplikacje działają w sposób rozproszony,

- przestrzeń inteligentna to środowisko dynamiczne – na przykład istniejące w niej urządzenia, aplikacje oraz ludzie zmieniają swoje funkcje czy położenie bez określonych wzorców zachowań. Dodatkowo w takiej przestrzeni mogą pojawiać się nowe urządzenia, a inne być z niej usuwane,
- wykorzystywane w ramach przestrzeni inteligentnej technologie i urządzenia są heterogeniczne,
- w przestrzeni może istnieć wiele osób, z których każda ma inne (w stosunku do niej) oczekiwania (np. niektóre osoby mogą oczekiwać szybkiej reakcji przestrzeni na swoje zachowania, podczas gdy dla innych najważniejsze jest śledzenie ich położenia), zatem przestrzeń powinna dostosowywać się do każdego swojego użytkownika.

Koncepcja *smart space* kładzie szczególny nacisk na aspekt przetwarzania rozproszonego i heterogeniczności funkcjonujących w przestrzeni technologii i aplikacji. Oba pojęcia (przestrzeń inteligentna i *smart space*) są do siebie bardzo zbliżone co powoduje coraz częstsze ich utożsamianie. Mimo, że dzisiaj terminy te są praktycznie nierozróżnialne, to w literaturze dalej da się zauważyć, większe zorientowanie przestrzeni inteligentnych na bezpośrednie wsparcie człowieka (poprzez na przykład roboty), z kolei przestrzeń *smart space* (*pervasive space*) znacznie częściej ogniskuje się na tematach związanych z architekturą, automatycznym wykrywaniem nowych usług i ich rozproszeniem, świadomością stanu, heterogenicznością itp. W dalszej części rozprawy terminy te będą wykorzystywane zamiennie.

Aktywne wsparcie użytkownika, będące podstawowym celem przestrzeni inteligentnych, jest określeniem bardzo ogólnym. W odniesieniu do przytoczonej wcześniej (za publikacją [131]) ich charakterystyki cel ten można uszczegółowić poprzez zdefiniowanie w ramach przestrzeni inteligentnej trzech funkcji: obserwacji, zrozumienia oraz działania [77]. Idea takiej przestrzeni została przedstawiona na rysunku 2. Człowiek (bądź aplikacja), znajdujący się w przestrzeni, otoczony jest siecią sensorów (*Sensors* - odpowiedzialnych za zbieranie informacji o zdarzeniach zachodzących w przestrzeni [11] – funkcja obserwacji) oraz aktuatorów (*Actuators* – mogących wykonywać działania [78] – funkcja działania). Aktuatory wspierają użytkowników fizycznie (wykonują za nich czynności), bądź też informacyjnie (dostarczają określone informacje) [11]. Wspólnie z sensorami są one określane jako DIND (*Distributed Intelligent Network Device*) [51] i pierwotnie były realizowane w formie urządzeń. Obecnie DIND można traktować jako



Rysunek 2 Idea przestrzeni inteligentnych i rozproszone inteligentne urządzenia sieciowe (DIND)

pewien obiekt fizyczny (urządzenia) bądź logiczny (aplikacje lub usługi typu SOA), który pośrednio albo bezpośrednio wchodzi w interakcję z użytkownikiem, analizuje oraz przetwarza dane z przestrzeni i może podejmować decyzje dotyczące sposobu działania (funkcja zrozumienia) [50]. Realizowana przez niego interakcja pośrednia polega na monitorowaniu zachowania użytkownika oraz dostarczaniu mu tzw. niefizycznych usług (np. informacji), natomiast interakcja bezpośrednia odbywa się poprzez urządzenia (roboty) – np. przemieszczanie ciężkich przedmiotów. W niniejszej rozprawie przyjęto założenie, iż obiekty DIND (pełniące zarówno funkcję sensorów jak i aktuatorów) są opakowane w usługi (w znaczeniu SOA). Założenie takie można przyjąć ponieważ obecnie jest to coraz częściej spotykana forma realizacji DIND. Posiadają one nowoczesne interfejsy komunikacyjne (takie jak TCP/IP, HTTP, Web Service [145]/SOAP [144]) oraz logikę aplikacyjną, które umożliwiają ich wykorzystanie zgodnie z ideą SOA. Funkcje DIND (których obecność jest warunkiem koniecznym istnienia przestrzeni inteligentnych [76]) są zatem zbieżne z funkcjami samej przestrzeni, które są opisane szerzej w następnym rozdziale.

Formalny model przestrzeni typu *pervasive*, abstrahujący od jej technicznych aspektów został zaproponowany w publikacjach [55] i [59]. Środowisko rzeczywiste zdefiniowano tam jako zbiór domen. Ma ono pewien stan odczytywany przez sensory (zawarte w obiektach DIND) i jest opisane przy pomocy ontologii. W środowisku tym może poruszać się człowiek, który wchodzi z nim w interakcję, która z kolei zmienia jego (środowiska) stan. Na bazie modelu przedstawionego w przytoczonych publikacjach przygotowano i rozszerzono podstawowe definicje wykorzystywane w rozprawie.

II.2.2. Funkcje przestrzeni inteligentnej

Dzięki sensorom (i opakowującym je obiektom DIND) przestrzeń zbiera dane dotyczące obiektów w niej funkcjonujących (ludziach, obiektach fizycznych, itp.). W pracy [103] tę funkcję przestrzeni określono jako najważniejszą, ponieważ rozpoczyna cały proces wspomagania użytkowników. W celu jej realizacji może być stosowanych wiele podejść i technologii. Najprostsze rozwiązania obejmują usługę lokalizacyjną, w której pozycję użytkownika dostarcza zainstalowany czujnik, bądź czytnik (na przykład czytnik RFID [39]). Bardziej skomplikowane metody opierają się na analizie obrazu (wykorzystywane np. w inteligentnych budynkach [89]). Algorytmy tej analizy znacząco rozwinęły się w ciągu ostatnich dziesięciu lat, dostarczając rozwiązań, umożliwiających identyfikację osób (i obiektów) w czasie rzeczywistym. Niektóre z nich zostały opisane w pracy [63], gdzie zaproponowano sposób odczytywania własności obiektów, poprzez analizę położenia ręki człowieka, wchodzącego z nimi w interakcję. Praca ta opisuje sposób realizacji procesu pozyskiwania stanu przestrzeni (wartości parametrów opisujących obiekty w niej istniejące), tak aby pozwalał na szybką reakcję na pojawiające się w niej w dowolnym momencie czasu (tzw. ciągłość działania) nowe sytuacje (którymi są stany przestrzeni). Własność ciągłości działania została również poruszona w pracy [23], gdzie dokonano porównania podejścia *persistent computing* i *Soft System Bus* (SSB) jako podstaw do konstrukcji przestrzeni inteligentnej. Jednakże samo zbieranie bieżących danych jest niewystarczające do zbudowania w pełni funkcjonalnej przestrzeni inteligentnej. Potrzebne jest także ich zapamiętywanie w taki sposób, aby zarówno procesy analityczne zachodzące w przestrzeni jak i człowiek w niej funkcjonujący mogli mieć do nich dostęp. Opisana w pracy [81] pamięć przestrzenna – *spatial memory* [99][103] realizuje funkcję pozwalającą spełnić to wymaganie. Dzięki niej przestrzeń potrafi zapamiętać różnego rodzaju informacje związane z zaistniałymi sytuacjami (np. rozmowa) i następnie zaprezentować je użytkownikowi na żądanie. Istotną zaletą tego rodzaju pamięci jest możliwość przechowywania informacji o emocjach, jakie przeżywały osoby znajdujące się w danym miejscu przestrzeni.

Zbieranie i przechowywanie danych jest jedną z funkcji przestrzeni inteligentnych lecz dopiero sposób ich przetwarzania oraz analizy decyduje o tym czy (i jak bardzo) przestrzeń jest inteligentna. Nierozłącznym elementem analizowania danych o stanie przestrzeni jest ich interpretacja w sposób rozmyty [44] (analogicznie do podejścia stosowanego w logice rozmytej [115]). Jest to spowodowane tym, iż odczyty dokonywane przez sensory są najczęściej nieprecyzyjne. Dzięki zastosowaniu tego podejścia można wykorzystywać algorytmy, dostarczające bardziej abstrakcyjne informacje o użytkownikach (na przykład przeżywane przez nich emocje), przez co przestrzeń może w dokładniejszy

i wieloaspektowy sposób prowadzić analizę ich (a także jej samej) zachowań oraz rozumieć zachodzące w niej zdarzenia i zmiany [95][96]. Może ona budować modele tych zachowań, na bazie których będzie w stanie tworzyć ich strategie, przewidywać dalsze zachowania oraz odpowiednio się zmieniać (przystosować do nowych warunków, zmieniających się na przykład na skutek działań człowieka). Przykładem analizowania zachowań człowieka jest obserwowanie jego interakcji z obiektami. Takie podejście przedstawiono w publikacjach [101][102], gdzie wykorzystano ideę 4W1H (*What,When,Where,Who,How*). Podejście to opisane jest również w pracy [103] – gdzie wykorzystano SOM - *Self Organizing Maps* [66] do określenia składowej *How*. Jednocześnie istotne jest to, iż użytkowników w przestrzeni inteligentnej może być wielu i do każdego z nich przestrzeń musi osobno dostosowywać swoje działanie (praca [3] proponuje tzw. PSS - *Personal Smart Space* jako rozwiązanie tego problemu). W konsekwencji przestrzeń inteligentna coraz bardziej zbliża się do idei przedstawionej w pracy [139] - jako przestrzeni, rozumiejącej człowieka w niej działającego i poprzez to efektywnie go wspierającej. Świadomość stanu idzie w parze z faktem, iż przestrzeń inteligentna musi być zarówno reaktywna jak i pro-aktywna. Nie wystarczy, aby akcje dostosowujące zachowanie przestrzeni do nowej sytuacji (stanu przestrzeni) podejmowane były na skutek bezpośrednich działań człowieka. W przypadku, gdy przestrzeń sama stwierdzi, że zaistniała pewna sytuacja (na przykład wystąpienie zagrożenia), musi umieć odpowiednio zareagować [117].

Przestrzeń inteligentna jest aktywną przestrzenią informacyjną, co oznacza, iż jej użytkownicy mogą uzyskiwać od niej potrzebne informacje. Funkcja ta opiera się na opisanej wcześniej analizie i przetwarzaniu danych z niej zebranych. Jeden z rodzajów informacji udostępniany przez przestrzeń dotyczy afordancji [79], oznaczającej oferowanie przez przestrzeń możliwości wykonania przez użytkownika określonych czynności. W pracy [74] jest to druga (obok lokalizacji obiektów i użytkownika) informacja, pozwalająca zrozumieć to co się dzieje w przestrzeni inteligentnej. W publikacji tej wyróżniono dwa rodzaje przestrzeni w zależności od stopnia afordancji:

- przestrzeń potencjalnie informacyjna (*Potential Information Space*) – przestrzeń ta zawiera pewne informacje, jednakże aby je uzyskać użytkownik (człowiek bądź aplikacja) musi posiadać większą inteligencję. Środowisko naturalne jest przykładem tego rodzaju przestrzeni,
- przestrzeń pasywna informacyjnie (*Passive Information Space*) – ten rodzaj przestrzeni wzbogacony jest o pewne znaki, pozwalające użytkownikom łatwiej uzyskać zawartą w niej informację. Dzięki temu sami użytkownicy mogą mieć mniejszą inteligencję. Przykładem takiej przestrzeni jest droga ze znakami.

Zwiększenie poziomu afordancji pozwala na zmniejszenie wymaganego poziomu inteligencji, działających w przestrzeni aplikacji. Mogą one być odciążone w określonych aspektach (na przykład w zakresie analizowania informacji na temat stanu przestrzeni), a człowiek może w większym stopniu skupić się na zadaniu jakie ma do wykonania (ponieważ jest efektywniej wspierany). Roboty z kolei mogą mieć mniejsze zapotrzebowanie na energię, oraz wykonywać więcej czynności lepiej dopasowanych do zachodzących wokół nich zmian.

II.2.3. Wymagania niefunkcjonalne i aspekty jakościowe przestrzeni inteligentnej

W pracach [79] i [117] można znaleźć podstawowe sprzętowe i programowe wymagania (i zarazem wyzwania) stawiane przestrzeniom inteligentnym:

- dynamizm – heterogeniczne usługi (w znaczeniu SOA) mogą być często dodawane i usuwane z przestrzeni inteligentnej, zatem musi być ona rekonfigurowalna w czasie działania (*run-time*). Z punktu widzenia człowieka w niej funkcjonującego procesy te nie powinny być odczuwalne, co dla twórców oprogramowania stanowi duże wyzwanie (aktualnie istnieją tylko cząstkowe rozwiązania – np. w pracy [12] zaproponowano sposób rozwiązania problemu zaniku połączenia urządzeń mobilnych użytkownika z siecią komunikacyjną),
- skalowalność – architektura przestrzeni powinna wspierać zarówno implementację w pojedynczym pomieszczeniu jak i np. w obrębie całego miasta,
- integracja – w ramach przestrzeni powinna istnieć możliwość integracji pomiędzy już funkcjonującymi usługami,
- realizowalność – aby przestrzeń inteligentna była praktyczna, musi być zbudowana na bazie istniejących standardów, technologii i elementów, które są łatwo dostępne i mają akceptowalny koszt,
- konfigurowalność i utrzymanie – budowanie i utrzymywanie przestrzeni inteligentnej powinno wymagać minimalnego wysiłku. Przestrzeń musi sama się uczyć i dostosowywać do bieżących warunków (np. *auto-calibration*),
- przezroczystość dla użytkownika (*invisibility*) – przestrzeń inteligentna typu *pervasive* powinna w minimalnym stopniu angażować człowieka (zmuszać go do wykonywania działań) – jedynie wtedy, gdy jej działania są niezgodne z zamierzeniem człowieka lub grożą niebezpiecznymi konsekwencjami,
- percepcja i mądrość (*smartness*) - świadomość własnego stanu oraz właściwe nim zarządzanie.

Spełnienie powyższych wymagań umożliwia prawidłowe funkcjonowanie przestrzeni inteligentnej a sposób ich realizacji decyduje o jego jakości. Bez uwzględnienia dynamizmu, percepcji i mądrości nie można mówić o adaptacji do nowych sytuacji, a z kolei zapewnienie realizowalności, skalowalności, integracji i łatwości konfiguracji oraz właściwego utrzymania są gwarantami udanego wdrożenia przestrzeni w rzeczywistym świecie.

Aspekty jakościowe przestrzeni inteligentnych dotyczą łatwości współpracy z użytkownikiem, wydajności realizacji dostępnych w niej funkcji, bezpieczeństwa dostępu i wiarygodności analizy oraz przetwarzania danych z niej odczytanych. W celu ich realizacji niezbędny jest odpowiedni interfejs komunikacyjny pomiędzy przestrzenią i jej użytkownikami. Badania opisane w pracy [43] pokazują, że dla człowieka najważniejsze w zakresie produktywności jest dobre samopoczucie w środowisku, w którym się znajduje. Z kolei możliwość zmiany środowiska, odbywająca się właśnie poprzez interfejs komunikacyjny (którego jakość jest uzależniona w dużej mierze od jego prostoty i intuicyjności), może wpływać na to samopoczucie. Przykłady takich interfejsów, zbudowanych przy wykorzystaniu gestów i mowy (poprzez które człowiek wydaje polecenia przestrzeni) opisano w pracach [19][87][132][134][162]. Można także zaliczyć do nich opisywaną wcześniej pamięć przestrzenną. Dzięki wizualizacjom umożliwia ona wykorzystanie dużej liczby zmysłów do lepszego zapamiętywania danych (dane związane są z konkretnym fizycznym miejscem). Sposoby współpracy użytkownika z przestrzenią inteligentną można najogólniej podzielić na dwie kategorie (podział autorski):

- bezpośrednia, tam gdzie człowiek musi posiadać pewne urządzenie, poprzez które komunikuje się z przestrzenią (podejście wywodzące się z trendu *pervasive*),
- pośrednia, w której nie ma konieczności posiadania przez człowieka dedykowanych urządzeń (podejście bardziej zbliżone do koncepcji *ubiquitous*).

Komunikacja przy wykorzystaniu urządzeń jest szeroko rozpowszechniona w przestrzeniach inteligentnych [102], ponieważ może zapewniać wiarygodną transmisję danych. Współpraca pośrednia nabrała znaczenia w ciągu ostatnich lat, co jest spowodowane rozwojem samych algorytmów analizy sytuacji [134], a także postępem technologicznym (np. w zakresie analizy głosu [54]). Oprócz komunikacji z człowiekiem przestrzeń musi komunikować się także z aplikacjami i urządzeniami w ramach niej działającymi. Musi ona zatem posiadać odpowiednie interfejsy, umożliwiające także tę komunikację.

Ponieważ użytkownik oczekuje bardzo często natychmiastowej reakcji na wykonywane przez siebie czynności, zatem procesy akwizycji, analizy i przetwarzania

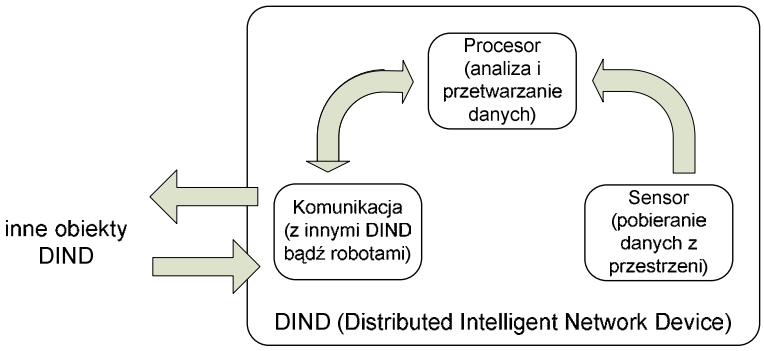
danych oraz działania w przestrzeni muszą spełniać wymogi działania w czasie rzeczywistym [82] (w pracy [14] zaproponowano wykorzystanie procesorów GPU do tego celu). Publikacja [82] zwraca uwagę na jeszcze inny aspekt, jaki pojawił się wraz z wprowadzeniem inteligentnych przestrzeni wirtualnych [133] - mianowicie na czas komunikacji pomiędzy ich rozproszonymi geograficznie komponentami (na przykład pomiędzy obiektami DIND a procesami analizującymi i przetwarzającymi dane o stanie przestrzeni). Gdy jest on zbyt długi, wtedy sumaryczny czas reakcji przestrzeni na działania użytkownika może być dla niego nieakceptowalny.

Przestrzeń inteligentna może być współdzielona pomiędzy wiele osób, dlatego też istotne jest zapewnienie w jej ramach odpowiedniego poziomu bezpieczeństwa w zakresie zarówno dynamicznego budowania aplikacji [122] jak i dostępu do informacji [100] (np. w przypadku wykorzystania pamięci przestrzennej). Innym aspektem, jaki jest istotny podczas wykorzystania w ramach przestrzeni inteligentnej heterogenicznych usług (tzn. pochodzących z różnych źródeł), jest zaufanie (*trust*). Korzystające z nich aplikacje mogą mieć dostęp do wrażliwych z punktu widzenia użytkownika danych i informacji (na przykład wzorca linii papilarnych). Dodatkowo, ze względu na istnienie w obrębie przestrzeni człowieka, w czasie jej budowania należy być szczególne ostrożnym, kładąc duży nacisk na aspekty bezpieczeństwa oraz prywatności [152]. Muszą istnieć mechanizmy zapewnienia zarówno bezpiecznego wykonania usług, jak i ochrony informacji oraz danych przez nie analizowanych oraz przetwarzanych. Zagadnienie to poruszone zostało w pracach [20][137][149][156].

II.2.4. Sposoby realizacji przestrzeni inteligentnych

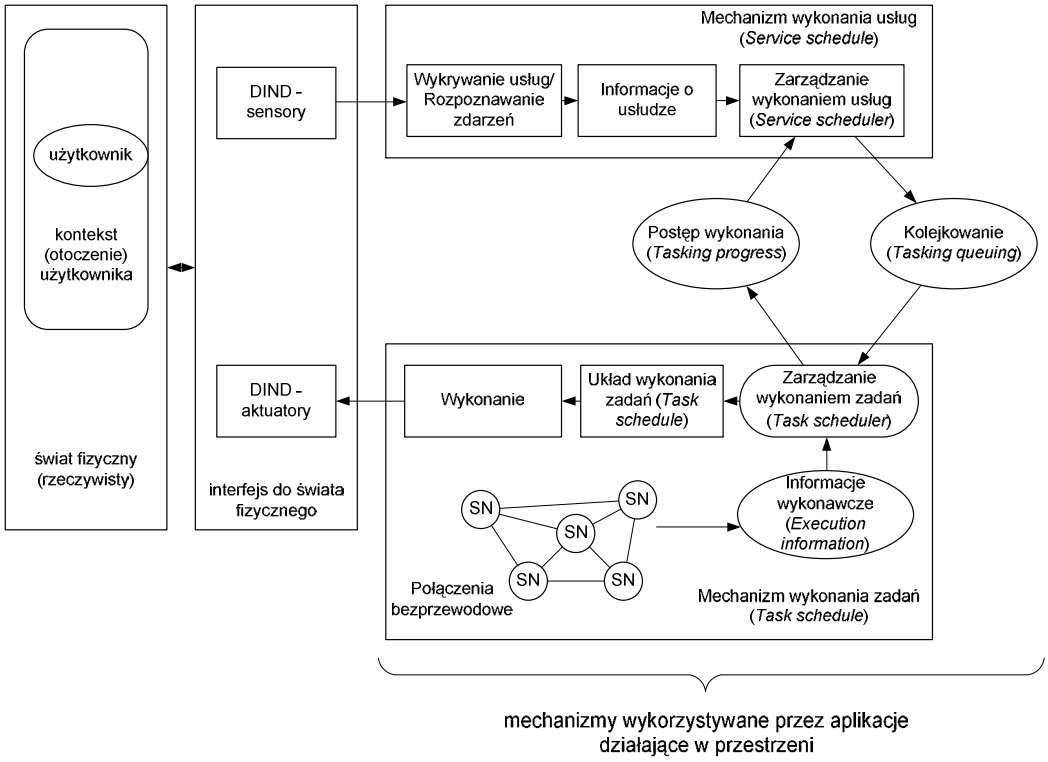
W kontekście realizacji przestrzeni inteligentnych należy rozpatrywać dwa aspekty. Pierwszy to budowa samej przestrzeni, w znaczeniu wymaganych komponentów (realizujących funkcje przestrzeni - np. obiektów DIND), powiązań między nimi oraz funkcji jakie pełnią. Oprócz tego w przestrzeni działają różnego rodzaju aplikacje, które nie tylko z niej korzystają (np. odczytują jej stan), ale mogą również stanowić jej istotny fragment (tzn. mogą być wykorzystane do wspomagania innych jej użytkowników). Powoduje to silne powiązanie i przenikanie konstrukcji przestrzeni oraz aplikacji. Jednakże wciąż nie można utożsamiać tych pojęć. Przestrzeń inteligentna powinna być odpowiedzialna, za dostarczenie odpowiednich funkcji (opisanych w rozdziale II.2.2), natomiast aplikacje powinny decydować o sposobie ich wykorzystania.

Najbardziej podstawowym elementem przestrzeni inteligentnych są wprowadzone w rozdziale II.2.1 obiekty DIND [73], których strukturę przedstawiono na rysunku 3.



Rysunek 3 Podstawowa struktura obiektu DIND

Obiekty DIND znajdują się bezpośrednio w przestrzeni i komunikują się między sobą, realizując ideę przetwarzania rozproszonego. Przy ich wykorzystaniu można zbudować dowolną przestrzeń inteligentną. Przykład architektury takiej przestrzeni został przestawiony w pracy [118], której uogólnienie można znaleźć w publikacji [84] (rysunek 4).



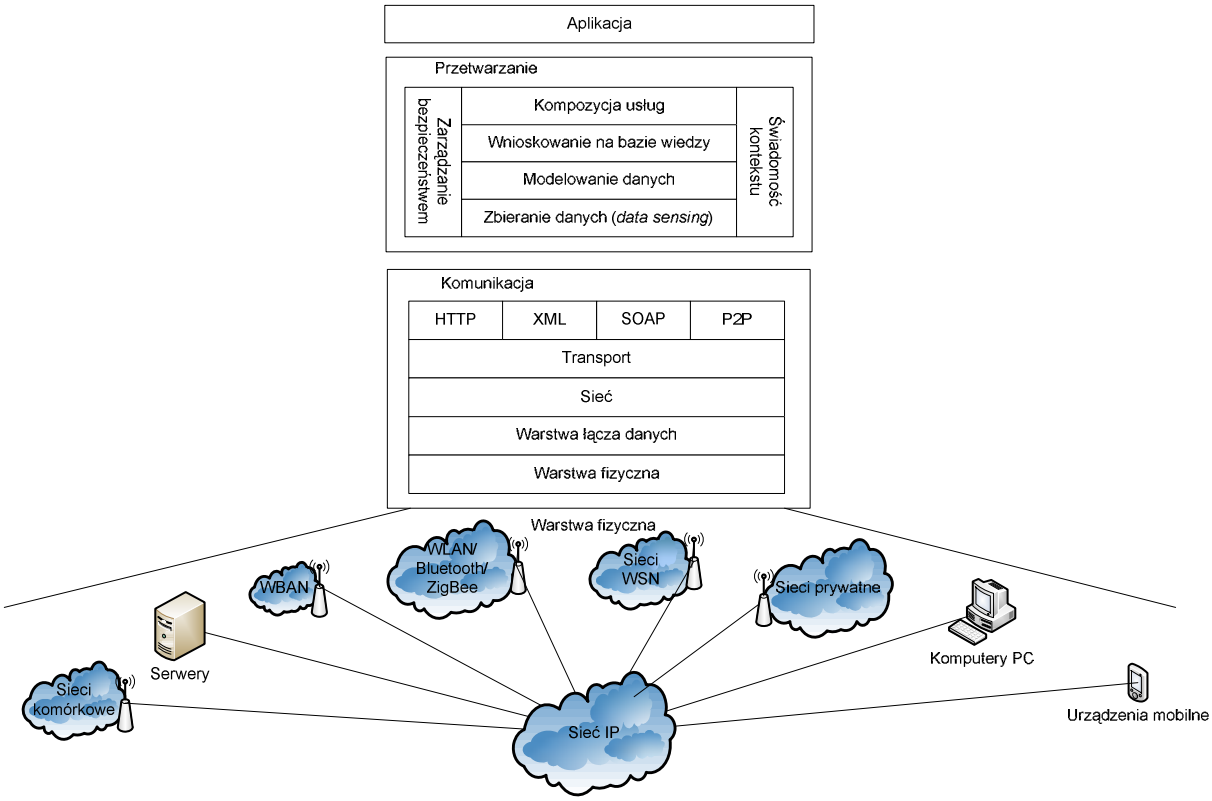
Rysunek 4 Ogólna architektura przestrzeni inteligentnych

W przestrzeni inteligentnej świat fizyczny (rzeczywisty) postrzegany jest przez pryzmat sensorów (zawartych w usługach opakowujących obiekty DIND), które dostarczają informacje na temat jego stanu. Stan ten jest następnie analizowany przez aplikacje

utworzone przy wykorzystaniu zawartych w przestrzeni mechanizmów (opartych na usługach w znaczeniu SOA i mechanizmów realizacji zadań) – dostępnych np. w postaci warstwy pośredniczącej *middleware* (jest to opisane w dalszej części rozdziału). Analiza i przetwarzanie stanu przestrzeni może odbywać się w sposób scentralizowany bądź rozproszony. Oba te podejścia i ich wpływ na podejmowanie działań przez aplikacje zostały omówione w pracy [129], gdzie autor zwraca także uwagę na związane z nimi aspekty wydajnościowe. Na podstawie tej analizy aktuatory mogą następnie wykonać określone (przez aplikacje) działania. Według tej architektury wymagane jest istnienie w przestrzeni następujących rozwiązań:

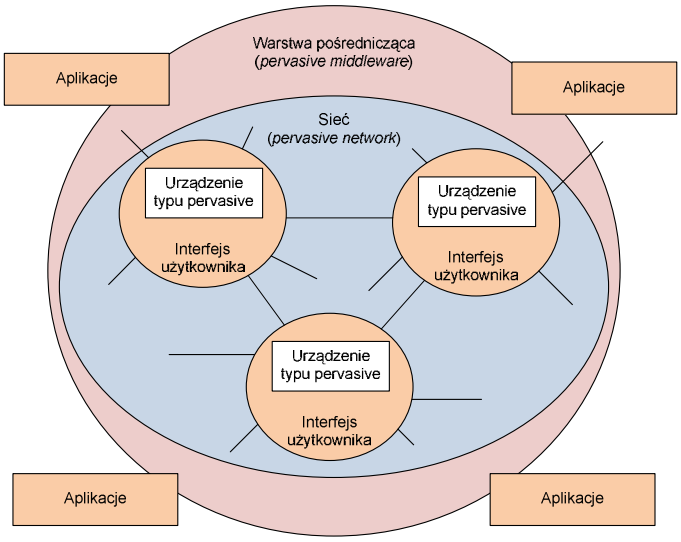
- powiązania ze światem rzeczywistym – realizowane przez sensory (znajdujące się w obiektach DIND),
- wszechobecna komunikacja – realizowana przez zastosowanie *Wireless Sensor Networks* (bezprzewodowo połączone sieci sensorów – SN), dzięki której urządzenia wszechobecne komunikują się ze sobą, propagując wiedzę o stanie przestrzeni,
- zrozumienie informacji odczytywanych z przestrzeni (jej stanu) – na przykład poprzez zastosowanie logiki rozmytej,
- podejmowanie decyzji i wykonywanie działań – na przykład poprzez wykorzystanie *Wireless Sensor and Actuator Networks* (WSANs) - rozproszonych systemów wyposażonych w sensory i aktuatory, które analizują i przetwarzają informacje dotyczące stanu przestrzeni i komunikują się między sobą w sposób bezprzewodowy.

Przykład warstwowego modelu organizacji przestrzeni inteligentnych, ukierunkowanego na automatyczne budowanie aplikacji z dostępnych w nich usług (w znaczeniu SOA), został przedstawiony w pracy [163] (rysunek 5). Model ten składa się z trzech głównych warstw. Warstwa fizyczna dotyczy konstrukcji sieci służącej do wymiany informacji pomiędzy elementami (w tym także użytkownikami) przestrzeni (np. DIND, ludźmi, robotami). Warstwa komunikacji zapewnia (poprzez wykorzystanie różnych technologii) łączność pomiędzy urządzeniami i usługami działającymi w przestrzeni. Warstwa przetwarzania odpowiedzialna jest za odbieranie danych z sensorów znajdujących się w obiektach DIND (*data sensing*) i wstępne ich przetwarzanie (*data modeling*). Na bazie przetworzonych danych dokonywane jest wnioskowanie (*knowledge reasoning*) co z kolei pozwala na kompozycję usług [164] (*service composition*) – realizującą określoną funkcjonalność (tzn. aplikację). Wszystkie te działania wykorzystują informacje o kontekście (*context awareness*) oraz wymagają określonego poziomu bezpieczeństwa (*security management*).



Rysunek 5 Warstwowy model architektury przestrzeni inteligentnej

Rozwój przestrzeni typu *smart space* znacząco uwypuklił aspekt heterogeniczności istniejących w nich usług SOA, aplikacji i urządzeń. Jak to zostało omówione w rozdziale II.2.1 obiekty DIND (sensory i aktuatory) można traktować jako usługi a także aplikacje dostarczają dodatkowych usług, z których może korzystać człowiek (i inni użytkownicy), poruszający się w takiej przestrzeni. Niesie to ze sobą konieczność ich dynamicznego wykrywania oraz integracji. Przedstawiona na rysunku 4 architektura przestrzeni oraz model warstwowy (patrz rysunek 5) jeszcze bardziej obrazują potrzebę wyodrębnienia tych mechanizmów w taki sposób, aby mogły być następnie wykorzystywane przez użytkowników przestrzeni, w szczególności przez aplikacje. Powstała zatem koncepcja budowania przestrzeni inteligentnych w oparciu o podejście *middleware* [9][117] (zakładające istnienie warstwy pośredniczącej), które zapewnia aplikacjom w nich działającym dostęp do mechanizmów przez nie dostarczanych (np. zarządzanie dostępem do usług opakowujących obiekty DIND, zarządzanie stanem przestrzeni, itp.). Stanowi to przeciwieństwo względem podejścia zamkniętego, w którym wszystkie mechanizmy potrzebne do działania aplikacji były bezpośrednio w nich zawarte. Aspekty takie jak wykrywanie pojawiających się usług i ich integracja, a nawet analiza stanu przestrzeni mogą być realizowane przez warstwę pośredniczącą (rysunek 6).



Rysunek 6 Warstwa pośrednicząca (middleware) dla przestrzeni inteligentnych

Rysunek 6 przedstawia cztery obszary (wraz z zależnościami między nimi), które muszą być technologicznie przygotowane do wytwarzania aplikacji wszechobecnych dla przestrzeni inteligentnych. Obszary te to obiekty DIND (oznaczone jako urządzenia wszechobecne *pervasive*), łącząca je sieć, warstwa pośrednicząca oraz aplikacje użytkowe.

Uzupełnieniem przedstawionych architektur jest wykorzystanie podczas projektowania przestrzeni inteligentnych podejść SOA/SOC [105] (*Service Oriented Architecture/Service Oriented Computing*) – omówionych na przykład w pracach [56][61]. Jest to spowodowane opisywaną wcześniej heterogenicznością aplikacji, urządzeń i usług (w znaczeniu SOA) w niej działających. Naturalne dla koncepcji SOA jest luźne powiązanie usług, jakie składają się na wynikową aplikację, co bardzo dobrze wpasowuje się w wymagania na przestrzeń typu *smart space*. Przykładem realizacji takiej architektury jest [12] – *Service-Oriented Adaptative Framework* (SOAF).

Osobnym problemem, obok organizacji architektury przestrzeni inteligentnych, jest budowa aplikacji w nich działających. Od momentu powstania paradygmatu przetwarzania wszechobecnego brakowało warstwowego modelu takich aplikacji (podobnego do modelu OSI dla sieci komputerowych [142]), który umożliwiałby logiczne wydzielenie ich fragmentów, skuteczniejsze eliminowanie błędów, a także usystematyzowane projektowanie oraz implementację. Luka ta została wypełniona w pracy [22]. W publikacji zaproponowano model warstwowy zarówno dla części aplikacyjnej jak i również dla działań samego człowieka. Model ten jako jeden z pierwszych zakłada, iż człowiek staje się integralnym fragmentem całej aplikacji wykonywanej w przestrzeni typu *smart space*. Jego działania są niczym instrukcje takiej aplikacji, co oznacza iż są one (aplikacje)

z natury rzeczy interaktywne. Inny model warstwowy dla oprogramowania typu *pervasive*, który w szczególności zwraca uwagę na centralne położenie człowieka w tego rodzaju aplikacjach, został zaproponowany w pracy [160]. Zdefiniowano w nim następujące warstwy (analogią jest model OSI dla sieci komputerowych):

- warstwa sieci (*pervasive network layer*) – może być również określona jako uniwersalna sieć, bądź środowisko przetwarzania nieustającego (*pervasive computing environment*). Warstwa ta zawiera wszystkie obecnie istniejące sieci, jakie mogą być podłączone do Internetu (bezpośrednio bądź pośrednio) i zajmuje się przydzielaniem zasobów, organizacją warstwy pośredniczącej (*pervasive middleware*) itp. Do jej opisu można zastosować siedmio warstwowy model referencyjny OSI,
- warstwa dostępu (*pervasive access layer*) – jest to warstwa zapewniająca połączenia pomiędzy innymi fragmentami infrastruktury (min. wykrywanie usług, zarządzanie usługami, bezpieczeństwo, prywatność, paradygmaty takie jak Web Services i wykorzystanie agentów, świadomość stanu, protokoły sieciowe),
- warstwa urządzeń (*pervasive device layer*) – zawiera wszystkie urządzenia, jakie są wykorzystywane przez człowieka (bezpośrednio i pośrednio), niezależnie od tego czy jest on świadomy ich istnienia. Aspekty poruszane w ramach tej warstwy to zarządzanie przez urządzenia danymi o stanie przestrzeni, organizacja obiektów DIND (pełniących zarówno funkcję sensorów jak i aktuatorów), urządzenia inteligentne, struktura i projekt oprogramowania urządzeń, systemy wbudowane. Do jej analizy można również zastosować siedmio warstwowy model referencyjny OSI,
- warstwa współpracy człowieka i urządzeń (HMI - *Human-Machine Interaction*) – warstwa opisująca współpracę człowieka z komputerem, ale także z wszystkimi innymi urządzeniami (takimi jak np. PDA). Aspekty poruszane w ramach tej warstwy to min. świadomość stanu przestrzeni z punktu widzenia człowieka, paradygmaty komputerowe dla koncepcji HMI, pozycjonowanie i śledzenie obiektów w przestrzeni inteligentnej, interfejsy użytkownika,
- warstwa użytkownika (*human core layer*) – odzwierciedla ona charakter przetwarzania wszechobecnego, w którego centrum jest człowiek otoczony urządzeniami i siecią realizującą to przetwarzanie. Warstwa ta koncentruje się na wymaganiach użytkownika oraz systemów bazujących na aplikacjach (rozumianych jako np. opieka zdrowotna, automatyzacja, inteligentne budynki, środowisko pracy człowieka, aplikacje biznesowe).

Poszczególne warstwy zdefiniowane w przedstawionym modelu mogą być realizowane przez przestrzeń inteligentną w formie warstwy pośredniczącej *middleware*.

Zagadnienie sposobu budowy aplikacji dla przestrzeni inteligentnych można także rozważać z punktu widzenia możliwości zmian takiej aplikacji (analizującej i przetwarzającej dane o stanie przestrzeni). Moim zdaniem na tym polu można wyróżnić 2 zasadnicze podejścia:

- niedefiniowalne - analizowanie i przetwarzanie wbudowane na stałe w aplikację a każda zmiana w zakresie tych mechanizmów oraz sposobie reagowania aplikacji wymaga pracy programistycznej,
- definiowalne, umożliwiające zmianę przez użytkownika zachowania aplikacji bez konieczności pracy programistycznej.

Podejście niedefiniowalne opiera się na założeniu wykorzystania określonych obiektów DIND (pełniących funkcję sensorów i aktuatorów) w procesie analizy i przetwarzania danych pochodzących z przestrzeni inteligentnej. Dzięki temu pozostają one pod kontrolą aplikacji (są „na sztywno" w niej osadzone). Sposób reakcji aplikacji na zachodzące w przestrzeni zmiany jest ustalany podczas jej tworzenia i pozostaje niezmienny z punktu widzenia użytkownika (jego zmiana wymaga pracy programistycznej). Podejście definiowalne, pozwala na określenie przez użytkownika przestrzeni (który nie musi mieć wiedzy na temat programowania) zasad, według których aplikacja ma reagować na zachodzące w niej sytuacje. Przykłady tego typu rozwiązań zostały opisane w publikacji [21].

Przestrzeń inteligentna wspiera człowieka w niej funkcjonującego poprzez obiekty DIND (pełniące funkcję sensorów bądź aktuatorów), które mogą być fizyczne albo logiczne. Na ich działanie mają wpływ aplikacje istniejące w ramach przestrzeni inteligentnej. Wykorzystują one mechanizmy w niej zawarte oraz dostarczają określonych funkcjonalności i usług (dla innych użytkowników przestrzeni). Można zatem stwierdzić, że decydują one o stopniu jej inteligencji. Stanowi to aktualny kierunek rozwoju tego rodzaju przestrzeni (zarówno w odniesieniu do terminu *ubiquitous* jak i *pervasive*). W zależności od stopnia zaawansowania aplikacji w nich działających będą stawały się one w mniejszym lub większym stopniu inteligentne, jednakże niewątpliwie będą realizowały coraz więcej i to coraz bardziej skomplikowanych funkcji. Różnica pomiędzy przestrzenią inteligentną oraz aplikacją ulega poprzez to zatarciu (także w odniesieniu do architektury) – w przypadku, gdy człowiek współpracuje z pewną aplikacją interaktywną można również mówić o jego współpracy z przestrzenią, w której się ona znajduje.

W dalszej części rozprawy przestrzeń inteligentna będzie rozumiana jako środowisko działania (człowieka bądź aplikacji) realizujące określone funkcje (w znaczeniu dostarczania mechanizmów pozwalających na wykonywanie aplikacji), natomiast aplikacja działająca w przestrzeni inteligentnej (w szczególności aplikacja IAI, której własności opisane są w następnym rozdziale) będzie rozumiana jako pewien zbiór funkcjonalności realizowany zgodnie z założonym scenariuszem wykonywany bezpośrednio przez przestrzeń (przy wykorzystaniu jej mechanizmów), wspomagający działania człowieka.

III WŁASNOŚCI INTERAKTYWNEJ APLIKACJI ITERACYJNEJ

Interaktywna Aplikacja Iteracyjna (IAI) jest to aplikacja wszechobecna, działająca w przestrzeni inteligentnej [69]. Podczas działania wchodzi ona w interakcję z innymi aplikacjami oraz z ludźmi. W rozdziale przedstawiono własności jakie posiada aplikacja IAI, takie jak iteracyjność, działanie w czasie rzeczywistym i równoległość obliczeń oraz interaktywność. Poza tym, dla tego rodzaju aplikacji wymagane jest zapewnienie efektywnej komunikacji z użytkownikami i otoczeniem zewnętrznym reprezentowanym przez jej kontekst, który musi być rozpoznawany i na bieżąco uwzględniany przy podejmowaniu kolejnych działań. Ponieważ przestrzeń inteligentna staje się środowiskiem wykonywania aplikacji w niej działających (jak to zostało zaznaczone w rozdziale II) niniejszy rozdział konkretyzuje również sposób osadzenia aplikacji IAI w takiej przestrzeni.

III.1. Aspekty iteracyjności

Algorytmy iteracyjne weszły do kanonu wiedzy informatycznej i znajdują zastosowanie w praktycznie każdej aplikacji [92]. Można na nie spojrzeć z dwóch punktów widzenia, jako realizacja idei powtarzalności operacji oraz szerzej, jako matematyczny model programu. Pierwszy polega na zaprojektowaniu algorytmu, jako pętli w ramach której wykonują się określone instrukcje (w odróżnieniu od rekurencji [123]). Jednokrotny przebieg takiej pętli stanowi pojedynczy krok algorytmu [25] i realizuje określony zestaw obliczeń, mających doprowadzić do poszukiwanego rozwiązania. Podejście to spotyka się najczęściej wtedy, gdy algorytm wykonuje typowo matematyczne obliczenia, których wynikiem ma być odpowiednio dokładne rozwiązanie (np. programowanie liniowe [48]). Jednakże nie zawsze celem aplikacji, implementującej dany algorytm, jest znalezienie rozwiązania podanego problemu matematycznego. Często musi ona wykonać ten sam ciąg operacji dla różnego zestawu parametrów (danych). Wówczas liczba iteracji (tzn. wykonań pętli) jest z góry określona i nie zależy od dokładności obliczeń (na przykład podczas przetwarzania obrazu).

Szersze spojrzenie na algorytmy iteracyjne opisane zostało w pracy [92], gdzie rozpatrywane są one jako matematyczne modele programów. Każdy program (aplikacja) może zostać przedstawiony jako zbiór etykiet i instrukcji. Dane reprezentowane są jako obiekty (względem których nie wprowadza się dodatkowych założeń), instrukcje opisane są przez relacje binarne w zbiorze obiektów i etykiet a program stanowi zbiór instrukcji.

Dzięki abstrahowaniu od konkretnych własności wymienionych elementów można uzyskać z jednej strony bardzo ogólny model programu, a z drugiej zaś określić działanie i logikę jego struktury. Wówczas algorytm iteracyjny definiuje się jako piątkę:

$$AI = \langle X, V, v_S, v_E, Q \rangle$$

gdzie X to zbiór obiektów algorytmu AI, V to skończony zbiór etykiet (tzw. symboli) algorytmu AI, v_S jest to pewien wyróżniony element V różny od v_E, zwany etykietą początkową algorytmu AI, v_E jest wyróżnionym elementem zbioru V (różnym od v_S), zwanym etykietą końcową algorytmu AI, natomiast Q jest to zbiór instrukcji (tzw. obliczeń) algorytmu AI definiowany jako skończony podzbiór następującego zbioru:

$$(V - \{v_E\}) \times 2^{X \times X} \times (V - \{v_S\})$$

gdzie $2^{X \times X}$ oznacza zbiór wszystkich podzbiorów zbioru $X \times X$. Instrukcje algorytmu są trójkami (v_P, Θ, v_K), w których $v_P \in V, v_K \in V$ są etykietami (odpowiednio wejściową i wyjściową instrukcji) natomiast $\Theta \subset X \times X$ jest relacją binarną (tzw. częścią operacyjną instrukcji). Jeżeli w wyniku wykonania instrukcji algorytm przechodzi co najwyżej do jednej etykiety (zmieniając przy tym wartości obiektów) nazywany jest algorytmem deterministycznym. Tak opisany algorytm będzie określany w dalszej części rozprawy jako klasyczny algorytm iteracyjny (za pracą [92]). Reprezentuje on podstawową konstrukcję aplikacji (także działającej w przestrzeni inteligentnej), która może być następnie przystosowana do wykonywania w sposób równoległy i uwzględniać określone ograniczenia czasowe.

III.2. Aspekty pracy w czasie rzeczywistym i zapewnienia równoległości

Podczas interakcji z innymi systemami, aplikacje często muszą zwracać odpowiedź w określonym czasie. Jest to szczególnie ważne np. w sterownikach, gdzie każde opóźnienie może powodować nieprawidłowe działanie układu (np. silnika). Aspekt ten nabiera jeszcze większego znaczenia, gdy dotyczy sterowania urządzeniami, od których bezpośrednio zależy życie człowieka. Podobnie aplikacje działające w przestrzeniach inteligentnych, które współpracują z innymi użytkownikami tych przestrzeni mają bardzo często narzucone czasy reakcji. Jest to związane z tym, iż opóźnienie w wykonywaniu działania przez jedną z takich aplikacji, automatycznie powoduje opóźnienie w innej (na przykład w przypadku kontroli dostępu dłuższy czas rozpoznawania osoby znajdującej się przed drzwiami skutkuje późniejszym ich odblokowaniem). Systemy, które muszą wykonać operacje z uwzględnieniem ograniczeń czasowych są określane jako systemy

czasu rzeczywistego (*Real Time System* – RTS) [64]. Nie zawsze oznacza to, iż operacje muszą być wykonane szybko, lecz muszą być wykonane w akceptowalnym czasie [88], nadążając za zmianami w przestrzeni, w której pracują. Cecha ta pozwala również na budowanie zaawansowanych scenariuszy interakcji pomiędzy systemami RTS, które posiadają gwarantowany czas wykonania. Dwoma z możliwych sposobów modelowania tego rodzaju systemów są teoria automatów oraz sieci Petrie'go [107].

Klasyczne automaty DFA i NFA [53] nie są wystarczające do modelowania aplikacji czasu rzeczywistego. Jest to spowodowane tym, iż nie można przy ich pomocy wyrazić ograniczeń czasowych, występujących w systemach RTS. W tym celu został opracowany tzw. automat czasowy (TA - *timed automata*) [4], w którym wartość czasu reprezentowana jest jako nieujemne liczby rzeczywiste (w odróżnieniu do dyskretnych modeli czasu). Automat akceptuje czasowe słowa (*timed words*), będące ciągami symboli, w których każdy z symboli (pochodzący z wejściowego alfabetu Σ) związany jest z czasem jego pojawienia się. Na bazie słów czasowych, można zdefiniować czasowy język (*timed language*) nad alfabetem Σ, jako zbiór słów czasowych nad tym alfabetem. W definicji automatu określa się także tzw. zegary, które mogą być niezależnie zerowane podczas przechodzenia między stanami i odliczają czas od ostatniego zerowania. Zmiany stanów automatu uwzględniają ograniczenia dotyczące zegarów – innymi słowy przejście automatu do innego stanu może być wykonane tylko wtedy, gdy wartości zegarów spełniają określone (w funkcji przejścia) warunki. Dla systemów działających w sposób ciągły (np. aplikacji przetwarzania wszechobecnego) ograniczenie słów, które podawane są do automatu, do skończonych ciągów symboli z alfabetu wejściowego może okazać się niewystarczające. W tym celu powstały tzw. ω-automaty, posiadające możliwość rozpoznawania nieskończonych słów [4].

Pełne zdefiniowanie konkretnego automatu FA polega na określeniu wszystkich jego zbiorów (alfabetu i stanów) oraz relacji przejścia. Jednakże nie dla wszystkich aplikacji taki opis można podać, gdyż albo operują one na nieskończonym alfabecie wejściowym, albo posiadają nieskończoną liczbę stanów [46]. Właśnie taka sytuacja występuje przy aplikacjach przetwarzania wszechobecnego, ponieważ zmiana ich stanu może nastąpić za każdym razem, gdy zmienia się stan przestrzeni w jakiej działają. Innymi słowy, słownik wejściowy dla automatów, modelujących aplikacje typu *pervasive/ubiquitous*, jest nieskończony (praktycznie każdy stan przestrzeni, jest elementem tego słownika). Istnieje szereg automatów, pozwalających modelować systemy z nieskończonym słownikiem wejściowym (np. *Push-Down Automata* – PDA [5], *Register Automata* – RA [46], *Data Automata* – DA [10]), które można także zastosować do modelowania systemów z nieskończoną liczbą stanów. Większość z nich (w szczególności DA) wykorzystuje

podejście polegające na skojarzeniu z symbolami należącymi do nieskończonego alfabetu wejściowego, symboli z (innego) alfabetu skończonego (funkcja transduktora) i przy jego wykorzystaniu prowadzenia dalszej analizy systemu.

Aplikacje, działające w przestrzeni inteligentnej, są skonstruowane również jako aplikacje równoległe (tzn. zbudowane przy wykorzystaniu algorytmów równoległych) w zakresie reagowania na zmiany zachodzące w otoczeniu. Jest to konieczne ponieważ zmiany te często następują jednocześnie i analizowanie informacji o jednej z nich nie może wstrzymywać analizowania informacji o kolejnej. Zrównoleglenie najczęściej wykorzystywane jest w zakresie realizacji interfejsów komunikacyjnych (do wielu systemów, w tym człowieka), a także jednoczesnego wykonywania różnego typu działań adaptacyjnych (np. do nowych sytuacji w przestrzeni). Aplikacje równoległe, podobnie jak aplikacje czasu rzeczywistego mogą być modelowane przy pomocy np. automatu PFA (*Parallel Finite Automata*) [127], bądź maszyny CSM (*Concurrent State Machine*) [27]. To drugie podejście wprowadza pojęcie sygnałów, które mogą pojawiać się jednocześnie (jako element wszystkich podzbiorów alfabetu wejściowego) oraz umożliwia przedstawienie każdego zrównoleglonego fragmentu aplikacji jako osobnego automatu.

Algorytmy równoległe są projektowane przy wykorzystaniu podejścia SIMD bądź MIMD [40]. Podczas ich tworzenia istotny jest dostęp do pamięci (przechowującej dane i wyniki operacji), która może być globalna bądź rozproszona. Pamięć globalna jest wspólna dla wszystkich obliczeń (każda z aplikacji wykonujących obliczenia może tak samo odczytywać bądź zapisywać do niej dane). Abstrakcyjnym modelem aplikacji z tego typu pamięcią jest maszyna PRAM (*Parallel Random Access Machine*) [41], która składa się z wielu równolegle wykonywanych maszyn RAM, komunikujących się ze sobą. Do komunikacji między nimi służy pamięć współdzielona, a każda z maszyn ma dodatkowo własną pamięć prywatną oraz jest wyposażona (w taki sam dla wszystkich maszyn) procesor do wykonywania obliczeń. Każdy z procesorów ma własny wskaźnik instrukcji. Model ten wykorzystywany jest do opisu algorytmów równoległych oraz szacowania ich złożoności obliczeniowej. Jednakże jednoczesny dostęp do współdzielonej pamięci może oznaczać występowanie konfliktów. Są one rozwiązywane według jednej z poniższych zasad [125]:

- CRCW (*Concurrent Read, Concurrent Write*) – jednoczesny zapis i odczyt do/z tej samej komórki pamięci jest dozwolony,
- CREW (*Concurrent Read, Exclusive Write*) – jednoczesne odczyty z pamięci są dozwolone, natomiast każdy procesor może zapisać daną w komórce pamięci tylko wtedy, gdy ma do niej wyłączny dostęp,

- EREW (*Exclusive Read, Exclusive Write*) – jednoczesny odczyt i jednoczesny zapis do tej samej komórki pamięci jest zabroniony.

Jednoczesne odczyty (gdy są dozwolone) nie powodują konfliktów, natomiast konflikty występujące podczas jednoczesnych zapisów mogą być rozwiązywane w jeden z następujących sposobów:

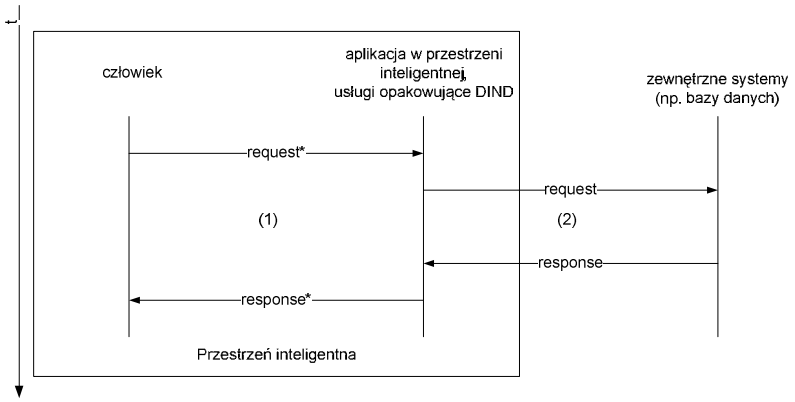
- *Priority* – każdy procesor ma przypisany priorytet. Dokonać zapisu do komórki pamięci może procesor z najwyższym priorytetem,
- *Arbitrary* – jeden losowo wybrany procesor może dokonać zapisu do komórki pamięci,
- *Common* – wszystkie procesory mogą dokonać zapisu, jednakże tylko wtedy, gdy wartość zapisu jest dla wszystkich procesorów taka sama.

Aplikacje równoległe, które muszą spełniać określone wymagania czasowe, mogą być opisane za pomocą modelu będącego rozszerzeniem podstawowego automatu FA. Przykładem takiego modelu jest TCSM (*Timed Concurrent State Machine* [27]), w którym zostało zaproponowane rozszerzenie modelu CSM o aspekty czasu rzeczywistego. W podejściu tym każdy z równolegle wykonywanych komponentów aplikacji rozpatrywany jest jako osobny automat. W pracy [86] zastosowano podobne podejście, w którym równolegle wykonywane komponenty komunikują się poprzez sygnały czasowe (*timed signals*). Ciekawym automatem jest APA (*Abstract Parallel Automata*) [94], gdzie potraktowano rozszerzenia wprowadzane przez TA oraz wszelkie inne parametry zmiany stanów jako zmienne, których wartości mogą być porównywane z innymi zmiennymi jak i ze stałymi. Zmienne określają również stan, w jakim znajduje się cały automat. Jest to zatem bardzo ogólna definicja automatu, przy pomocy którego można opisywać aplikacje jako jeden rozbudowany automat, bądź podzielić go na mniejsze automaty, reprezentujące równoległe bądź rozproszone komponenty aplikacji. Niezależnie od typu automatu, aplikacje przez niego opisane mogą odpowiednio szybko zwracać odpowiedź ich użytkownikom. Stało się to jedną z motywacji rozwoju sposobów współpracy pomiędzy aplikacjami oraz aplikacjami i człowiekiem.

Funkcjonujące w przestrzeni inteligentnej aplikacje IAI posiadają zarówno cechy równoległego wykonywania operacji jak i wykonania ich w czasie rzeczywistym. Dlatego też, w dalszej części rozprawy do opisu tych aspektów ich działania zostanie wykorzystany automat czasowy (TA) rozszerzony o możliwość równoległego przechodzenia do wielu stanów (tak jak ma to miejsce w automacie PFA). Ze względu na konieczność reagowania takiej aplikacji na dowolne sytuacje mające miejsce w przestrzeni zastosowany zostanie także transduktor dokonujący zmiany alfabetu wejściowego automatu TA.

III.3. Aspekty interaktywności

Przestrzeń inteligentna (poprzez obiekty DIND), jak i działające w niej aplikacje, wchodzi w interakcję z użytkownikami w niej funkcjonującymi (ludźmi bądź aplikacjami). Jednym ze sposobów jej przedstawienia są diagramy interakcji, do których należą diagramy sekwencji [114]. Przykład takiego diagramu, obrazującego typowe interakcje (oparte o wykorzystanie przykładowej komunikacji zorganizowanej na zasadzie wymiany komunikatów *request/response*) występujące w przestrzeni inteligentnej, został przedstawiony na rysunku 7.

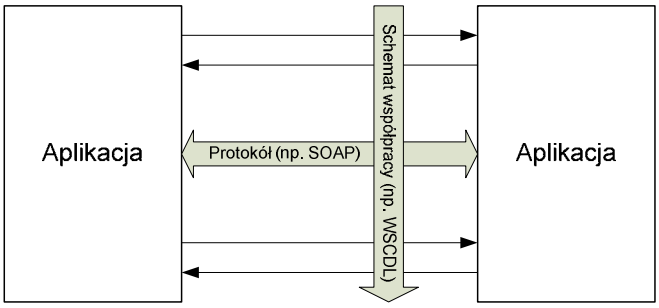


Rysunek 7 Diagram sekwencji dla przykładowej aplikacji interaktywnej w przestrzeni inteligentnej

Na rysunku 7 zostały zaznaczone funkcjonujące w przestrzeni obiekty DIND (opakowane w usługi) oraz dwie kategorie jej użytkowników: człowieka i aplikację. Współpracują oni na zasadzie *request*/response** (pierwszy rodzaj interakcji), która może przyjmować różną formę. Na przykład rozpoznanie osoby stojącej przed drzwiami poprzez sensor RFID nie wymaga wygenerowania przez tę osobę żadnego fizycznego komunikatu. Samo jej podejście do drzwi jest wyrażeniem prośby (*request**) o otwarcie drzwi. Następnie system kontroli dostępu, musi daną osobę zidentyfikować i sprawdzić czy może ona uzyskać dostęp do pomieszczenia. Jest to drugi rodzaj interakcji zachodzący w przestrzeni, w ramach którego wszyscy jej użytkownicy jak i sama przestrzeń (poprzez obiekty DIND) może wchodzić w interakcję z innymi (zewnętrznymi) systemami (np. bazami danych) – często zorganizowanymi w postaci usług (w znaczeniu SOA). Komunikacja z nimi (np. odpytanie bazy danych systemu kontroli dostępu) realizowana jest przy pomocy typowych komunikatów *request/response*. Istotne jest to, iż w ramach przestrzeni wszyscy jej użytkownicy mogą być autonomiczni. Jest to szczególnie ważne w odniesieniu do aplikacji ponieważ oznacza to, że mogą rozpoczynać wykonywanie pewnych procesów bez

ingerencji człowieka. W zależności od organizacji pracy użytkowników przestrzeni jak i możliwości innych systemów komunikacja może być zorganizowana w różny sposób (np. zamiast *request/response* może to być *publish/subscribe*), jednakże zasady jej funkcjonowania pozostają takie same.

Aplikacje interaktywne muszą ze sobą współpracować, zgodnie z przyjętymi zasadami (patrz rysunek 8). Interakcja ta jest przede wszystkim uzależniona od kompatybilności dostarczanych interfejsów oraz wspólnych zasad wymiany danych. Określa to odpowiedni protokół (np. SOAP dla usług sieciowych Web Service) reprezentowany przez oś poziomą przedstawioną na rys. 8. Dotyczy to każdego poziomu współpracy, zatem uwzględniając wszystkie poziomy interakcji otrzymujemy oś pionową współpracy określoną przez choreografię bądź orkiestrację [146].

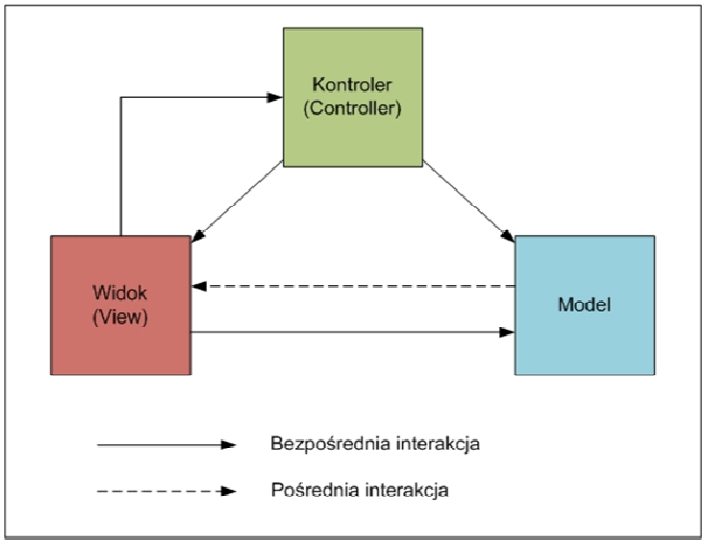


Rysunek 8 Współpraca między aplikacjami interaktywnymi

Jeden z coraz częściej stosowanych protokołów komunikacji dotyczy standardu usług sieciowych (*Web Service*). Umożliwia on udostępnienie funkcjonalności oprogramowania dla zewnętrznych aplikacji. Dzięki wykorzystaniu języka opisu danych XML usługi sieciowe są niezależne od platformy uruchomieniowej i pozwalają na wymianę informacji nawet między niezależnymi, heterogenicznymi środowiskami. To prowadzi do architektur SOA obecnie szeroko stosowanych przy konstruowaniu systemów. Ich rosnąca popularność doprowadziła do utworzenia specjalnego języka WSCDL (*Web Service Choreography Description Languge*) [146], opisującego zasady współpracy aplikacji na poziomie usług sieciowych. Stanowi on kolejny etap tworzenia nowych środowisk dla luźno powiązanych aplikacji.

Istnieje szereg wzorców według których tworzy się aplikacje interaktywne. Najpopularniejszymi z nich są MVC (*Model – View – Controller*) [141] wraz z późniejszymi MVP (*Model – View – Presenter*) [108] i PAC (*Presentation – Abstraction – Control*) [26]. Określają one, w jaki sposób należy rozdzielić funkcjonalność prezentacji,

interakcji z użytkownikiem oraz logiki aplikacyjnej pomiędzy poszczególne komponenty aplikacji. PAC (utworzony przez Joëlle Coutaz w 1987 roku) jest wzorcem hierarchicznym, którego głównym założeniem jest to, iż człowiek odbiera świat jako sieć powiązanych ze sobą, abstrakcyjnych idei. Z kolei wzorzec MVC (podobnie jak MVP) reprezentuje podejście oparte na warstwach. Został on utworzony przez Trygve Reenskaug w latach 1978 – 1979 [141]. Typowa struktura modelu MVC została przedstawiona na rysunku 9.



Rysunek 9 Struktura aplikacji interaktywnej dla modelu MVC

Zawiera on następujące role jakie mogą być przypisane do obiektów aplikacji [38]:

- Model – odpowiada za zarządzanie danymi oraz wykonywanie operacji związanych z logiką aplikacyjną (umożliwiającą realizację wymagań funkcjonalnych i niefunkcjonalnych aplikacji). Model może komunikować się z Widokiem w sposób pośredni (poprzez zdarzenia generowane w momencie, gdy pewna wartość danej w Modelu ulegnie zmianie). W ten sposób Widok może od razu uaktualniać prezentowane wartości. Element ten, w momencie gdy wykonuje działania, można traktować jako aktuator. Jest to jedna z funkcjonalności obiektów DIND.
- Widok – stanowi wizualną reprezentację modelu (zawartych w nim danych) i składa się z ekranów, na których umiejscowione są kontrolki (*widgets*). Widok posiada bezpośrednie odwołanie (referencję) do Modelu, którą wykorzystuje do pobrania danych jakie ma wyświetlić. Element ten realizuje sensoryczną funkcjonalność obiektów DIND (pobiera informacje od użytkownika).

- Kontroler – jest to komponent odpowiedzialny za analizowanie i przetwarzanie informacji przekazanych od użytkownika (takich jak polecenia wprowadzane z klawiatury). Poprzez Kontroler użytkownik może wpływać na zachowanie aplikacji i modyfikować dane zawarte w Modelu. Interpretowanie danych dostarczanych przez użytkownika (w znaczeniu stanu przestrzeni) jest również funkcją obiektów DIND.

Obiekty pełniące rolę Modelu nie posiadają bezpośredniego dostępu do obiektów pełniących rolę Widoku bądź Kontrolera i jedynie odzwierciedlają strukturę i stan danych, na których operuje aplikacja. Mogą natomiast być modyfikowane zarówno przez obiekty w roli Widoku jak i Kontrolera. Gdy zachodzi konieczność powiadomienia przez obiekty Modelu obiektów posiadających rolę Widoku lub Kontrolera o tym, iż zaszło jakieś zdarzenie (zmiana dotycząca danych), wykorzystuje się wzorzec obserwatora (*Observer Pattern*). Obiekty w rolach Widok i Kontroler współpracują, umożliwiając użytkownikowi wykonywanie operacji na danych (tzn. na Modelu). Każdy obiekt pełniący rolę Widoku jest skojarzony z pojedynczym obiektem w roli Kontrolera i odwrotnie – każdy obiekt w roli Kontrolera jest skojarzony z pojedynczym obiektem pełniącym rolę Widoku przy czym oba posiadają bezpośrednie powiązania z obiektami Modelu. Głównym zadaniem obiektów w roli Widoku jest odpowiednia reprezentacja wyjścia (*output*s) aplikacji, natomiast w przypadku obiektów Kontrolera jest to analizowanie i przetwarzanie wejścia (*inputs*) aplikacji (tzn. danych przekazywanych przez użytkownika). Niektóre akcje podejmowane przez użytkownika bezpośrednio oddziałują na obiekty pełniące rolę Modelu, natomiast niektóre bezpośrednio oddziałują na obiekty w roli Widoku.

W dalszej części rozprawy doktorskiej za reprezentatywny dla aplikacji interaktywnych zostanie wybrany właśnie wzorzec MVC ze względu na jego szerokie zastosowanie w wielu aplikacjach. Posłuży on do opisu interaktywności aplikacji IAI jaka wynika z ich definicji (wprowadzonej w rozdziale IV). Podobne podejście zastosowano także w pracach [111] i [155]. W pierwszej zaproponowano rozwiązanie, w ramach którego kontekst decyduje o tym, jaki kontroler jest uruchamiany w czasie interakcji z człowiekiem. Twórca aplikacji musi skupić się jedynie na przygotowaniu odpowiednich kontrolerów, dla poszczególnych sytuacji, jakie mogą wystąpić w przestrzeni. Druga publikacja proponuje wykonywanie różnych fragmentów kodu (tzw. izotopów), w zależności od sytuacji w przestrzeni.

III.4. Aspekty kontekstowości

Ważną własnością przestrzeni inteligentnych jest opisana w rozdziale II.2.2 świadomość własnego stanu (bieżącego bądź historycznego), wyrażającego sytuacje jakie

mają lub miały miejsce w przestrzeni. Stan fragmentu tej przestrzeni, w jakim działa aplikacja, określany jest jako kontekst [131][155] (w znaczeniu wartości przypisanych do parametrów obiektów znajdujących się w przestrzeni - np. wartość temperatury w sali seminaryjnej). Wszystkie aplikacje, które są czułe na dane wejściowe oraz zmiany zachodzące w przestrzeni, w jakiej są wykonywane, określane są jako aplikacje kontekstowe. Świadomość kontekstu (*context awareness*) umożliwia im lepiej służyć innym użytkownikom przestrzeni, poprzez automatyczne (transparentne dla użytkownika) dostosowanie się do zmieniających się warunków pracy (tzn. stanu przestrzeni) [49][126]. Jednakże każda aplikacja posiada własny kontekst do którego mogą należeć różne elementy przestrzeni, w jakiej działa. Dlatego też literatura obfituje w różne definicje kontekstu (np. prace [28][42][112])[57]. Jedną z najbardziej znanych i szeroko zaakceptowanych przez świat nauki jest ta zaproponowana przez A. K. Dey'a [29]. Według niego kontekst jest to każda informacja (np. charakterystyka, wymagania, możliwości, stan emocjonalny, odczucia itp.), pozwalająca scharakteryzować obiekt (aplikację, użytkownika, urządzenie, aktywność, proces itp.) [30][124]. Wcześniejsze definicje ograniczały kontekst do pewnego, z góry określonego zbioru obiektów i ich parametrów (np. miejsce, czas, aktywność) [116]. Jednakże wraz z postępem badań okazało się, iż takie jego pojmowanie jest niewystarczające. Dzieje się tak dlatego, iż ta sama informacja (ten sam fragment przestrzeni) może być fragmentem kontekstu w jednej aplikacji natomiast w innych już nie. Co więcej ta sama informacja może mieć różne znaczenie, które wraz z jej ważnością dla działania aplikacji się zmienia (na przykład w czasie). Powoduje to, iż praktycznie nie da się podać bardziej szczegółowej definicji kontekstu. Publikacja [29] zwraca również uwagę na to, iż nie jest możliwe określenie wszystkich ważnych aspektów sytuacji, w jakiej może znaleźć się aplikacja *a priori* – czyli sztywnego określenia kontekstu w momencie tworzenia aplikacji. Praca [140] konkluduje, iż o tym czy określona informacja jest kontekstem decyduje sposób jej interpretacji przez aplikację, a nie zestaw parametrów jakimi jest ona opisywana. Takie podejście ma duże konsekwencje dla sposobu działania przestrzeni inteligentnych. Kontekst, z definicji, staje się tworem dynamicznym [45], który podlega ewolucji [35]. Istotne jest zatem, aby aplikacje przetwarzania wszechobecnego nie tyle wspierały konkretny kontekst, lecz oferowały wsparcie dla procesu jego definiowania, negocjacji i współdzielenia – co stanowi dużo większe wyzwanie dla ich twórców, ale także pozwala na dużo większą elastyczność (pod względem adaptacji do zmian zachodzących w przestrzeni). Ogólna definicja kontekstu w praktyce oznacza przyjęcie postrzegania przestrzeni jako konstrukcji wielowymiarowej. Prace [35][52] zwracają uwagę na powiązania między różnymi wymiarami kontekstu – na przykład przebywanie w domu bądź w pracy jest bezpośrednio związane z czasem [126]. Relacje te są bardzo istotne w odniesieniu do reprezentacji

i analizowania kontekstu. Ta sama publikacja zwraca uwagę na to, iż analizowanie danych kontekstowych jeszcze bardziej komplikuje problem niejednoznaczności i niepewności dotyczący odczytywanych z przestrzeni danych (aspekt ten został także poruszony w rozdziale II.2.2). Nie mniej ważne od podjęcia odpowiedniej dla zaistniałego kontekstu akcji jest określenie czy rzeczywiście dany kontekst zaistniał (biorąc pod uwagę błędy odczytu sensorów zawartych w obiektach DIND, niekompletność informacji itp.). Do rozwiązania tego rodzaju problemów wykorzystuje się techniki z np. zakresu sztucznej inteligencji. Problem niespójności oraz metod jej poprawy został poruszony w pracach [32][37][91].

Zdefiniowany na potrzeby aplikacji kontekst można skategoryzować na wiele sposobów. Publikacja [36] proponuje jego kategoryzację (ze względu na źródło pochodzenia) na uzyskany z niskopoziomowych informacji (odczytanych bezpośrednio przez sensory) oraz wysokopoziomowych informacji, określonych na podstawie informacji niskopoziomowych. Z punktu widzenia zbierania danych o stanie przestrzeni kontekst można podzielić na bezpośredni (taki do zebrania którego nie trzeba dodatkowego przetwarzania) oraz pośredni (określony na bazie bezpośredniego) [158]. Innym aspektem kontekstu jest jego zmienność w czasie, zatem kolejnym przekrojem, jaki można zastosować jest kontekst statyczny i kontekst dynamiczny [126]. Ta sama publikacja proponuje warstwową kategoryzację kontekstu, zorientowaną na aktorów [114] aplikacji przetwarzania wszechobecnego:

- kontekst użytkownika (zewnętrzny – na przykład miejsce i wewnętrzny – na przykład uczucia),
- kontekst urządzeń (zainstalowane oprogramowanie i stan sprzętu),
- kontekst aplikacji (np. użytkownicy, urządzenia, czas pracy, itp.),
- kontekst informacji (np. rodzaj informacji – niskopoziomowe czy abstrakcyjne),
- kontekst środowiska (fizyczny – np. obiekty środowiska, cyfrowy – np. własności sieci transmisyjnej),
- kontekst czasu (czas zaistnienia zdarzenia, strefa czasowa),
- kontekst historyczny (sytuacje, jakie wydarzyły się wcześniej),
- kontekst relacyjny (określający relacje między różnymi wymiarami kontekstu).

Sposoby reprezentacji kontekstu (niezależnie od jego kategorii) można podzielić, w zależności od warunków działania aplikacji oraz rodzaju analizowanych i przetwarzanych przez nią danych, na dwie zasadnicze kategorie: jednoznaczne oraz synonimiczne. Pierwsza z nich zakłada sztywne określenie, iż kontekst zawiera pewien niezmienny zbiór elementów, które jednoznacznie opisują, interesujące z punktu widzenia

aplikacji aspekty jej działania. Przykładami może być publikacja [119], gdzie autorzy opisują kontekst jako miejsce, otaczające obiekty (w tym ludzie) i zmiany zachodzące w stanie tych obiektów oraz publikacja [28], w której do kontekstu zalicza się stan emocjonalny użytkownika, cel skupienia uwagi, miejsce oraz czas, obiekty i ludzi. Podejście oparte na synonimach zostało zaprezentowane np. w publikacji [57] gdzie kontekst oznacza aspekty trwającej sytuacji. Tego rodzaju reprezentacje są jednakże przeważnie zbyt ogólne i przez to nie mogą być zaimplementowane w postaci zrozumiałej dla komputerów. Pewnym rozwiązaniem tego problemu może być wykorzystanie ontologii, która zawierałaby wiedzę na temat przestrzeni działania aplikacji (w postaci pojęć i relacji między nimi) i jednocześnie umożliwiała analizowanie i przetwarzanie tej wiedzy przez komputery. Zasadnicze cele, jakie należy osiągnąć podczas tworzenia ontologii opisującej kontekst (tzn. fragment przestrzeni) wymienione zostały w pracy [68]. Są to:

- prostota opisu,
- elastyczność i rozszerzalność opisu o nowe elementy,
- możliwość wspierania różnych typów kontekstów,
- umiejętność wyrażania możliwie największej liczby kontekstów.

Przykładem środowiska, działającego w oparciu o ontologię kontekstu jest CoBrA, której opis działania można znaleźć w opracowaniach [6] i [16]. Wykorzystuje ona ontologię SOUPA [17][18][47], zawierającą większość podstawowych pojęć (takich jak usługa, osoba, czas itp.), jakie są wykorzystywane do definiowania danych kontekstowych w istniejących aplikacjach. Ontologie umożliwiają przedstawienie kontekstu w najbardziej ogólny sposób [16]. Dodatkowo, dzięki swoim własnościom pozwalają na wnioskowanie i łatwą rozbudowę o nowe informacje oraz fakty. Do ich zapisu i przechowywania wykorzystywane są języki opisu ontologii takie jak OWL [143].

Wykorzystanie kontekstu w aplikacjach powoduje, iż muszą one korzystać z szeregu mechanizmów umożliwiających zarządzanie, analizowanie i przetwarzanie danych kontekstowych. Mechanizmy te mogą być dostarczone przez przestrzeń inteligentną bądź być wbudowane w aplikację (patrz rozdział II.2.4). Publikacja [49] proponuje następującą ich kategoryzację:

- filtrowanie informacji oraz rekomendowanie informacji i usług (na przykład odnajdowanie najbliżej położonej drukarki, dostęp do historii obiektów znajdujących się w najbliższym otoczeniu),
- prezentacja i dostęp do usług (na przykład interpretowanie głosu w momencie, gdy

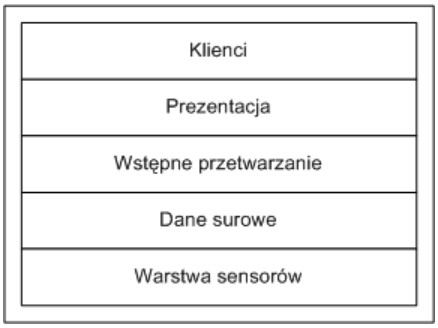
klawiatura jest niedostępna),

- wyszukiwanie usług i informacji (budowanie zapytań do systemów informacyjnych uwzględniające preferencje użytkownika, aktualne położenie itp.),
- układanie zadań (*task sequencing*) – budowa scenariusza zachowania się aplikacji,
- modyfikacje i konfiguracje usług oraz aplikacji,
- realizacja akcji – w pracy [93] wyróżniono trzy poziomy zależnych od kontekstu akcji: manualne, półautomatyczne i automatyczne. Publikacja [67] zauważa jednak że w pełni automatyczne akcje bazujące na kontekście są rzadko użyteczne, a akcje nieprawidłowe (będące wynikiem błędnej interpretacji kontekstu) mogą być dla użytkownika frustrujące,
- przydzielanie zasobów fizycznych do realizacji konkretnych zadań czy usług.

Zakładając, że aplikacja stanowi złożenie funkcjonalności dostarczanych przez różne heterogeniczne usługi [44] (w znaczeniu SOA), wpływ kontekstu może oznaczać na przykład inną ich kompozycję w zależności od zmieniających się warunków pracy [161]. Z punktu widzenia analizowania i przetwarzania kontekstu cykl życia aplikacji można podzielić na następujące fazy [7]: projektowanie, uruchamianie, działanie. Podczas etapu projektowania zaleca, się aby aplikacje były opisywane bez odniesień do konkretnych urządzeń. Dodatkowo aplikacje nie powinny czynić założeń o dostępności określonych usług, co powoduje konieczność abstrakcyjnego modelowania interfejsu użytkownika oraz opisów usług wymaganych do działania. Sama struktura aplikacji powinna być opisana przy wykorzystaniu zadań (*task*) i podzadań (*sub-task*). Z punktu widzenia etapu uruchamiania aplikacja musi być zdefiniowana przy pomocy wymagań funkcjonalnych i niefunkcjonalnych oraz możliwości jej działania, z kolei podczas wykonania musi monitorować zasoby i odpowiednio się dostosowywać do możliwości ich wykorzystania. Takie sformułowanie cyklu życia aplikacji oznacza dynamizm w zakresie jej wykonania. Proces ten składa się z szeregu kroków [103], z których pierwszy to zdefiniowanie wizji aplikacji. Następnie należy wybrać odpowiednie usługi do jej wykonania [128][159] (musi zatem istnieć mechanizm wykrywania [1], opisu [83] i budowania rejestru dla usług istniejących w przestrzeni), zbudować z nich docelową aplikację (w pracy [157] zaproponowano sposób zapewnienia spójności takiej aplikacji) i na końcu ją uruchomić. Już w trakcie działania aplikacja może ewoluować (tzn. być ciągle udoskonalana). Jest to ułatwione dzięki wykorzystaniu semantycznego podejścia do zagadnienia opisu usług (zaprezentowane w publikacji [150]), dzięki któremu aplikacja łatwo może dostosowywać się do nowych warunków pracy [79]. Zagadnienie to nabiera szczególnego znaczenia w momencie zastosowania koncepcji SOC, która zakłada, iż interfejsy poszczególnych usług mogą być różne [157]. W publikacji [65], opisano architekturę ukierunkowaną na

sposób doboru usług w heterogenicznych środowiskach (poprzez koncepcję ochotników / woluntariuszy). Z kolei praca [147] prezentuje podejście oparte o ideę komponentów do budowania aplikacji *pervasive*. Z podejścia komponentowego korzysta także *Component Based Software Engineering* (CBSE) [120]. Biorąc pod uwagę fakt, iż duża część istniejących aplikacji działa w formie nieprzystosowanej do przestrzeni inteligentnych, pojawiają się rozwiązania, które pozwalają je do tych przestrzeni przystosować bez zmian w ich kodzie [90] (np. dostosowują ich konfigurację do bieżącego stanu przestrzeni). Można zatem przyjąć, iż aspekt kontekstowości silnie wpływa na każdą fazę cyklu życia aplikacji.

Wymienione aspekty dotyczące uwzględnienia kontekstu podczas wytwarzania i wykonywania aplikacji można zorganizować w postaci warstwowego modelu aplikacji kontekstowej (ukierunkowanego na analizę i przetwarzanie danych kontekstowych). Najbardziej ogólna jego postać została przedstawiona w pracach [2][31] (rysunek 10).



Rysunek 10 Warstwowy model aplikacji z punktu widzenia analizy i przetwarzania kontekstu

Warstwa sensorów (*sensors*) odpowiada za zbieranie danych ze świata rzeczywistego (np. z czujników badających właściwości fizyczne, z innych aplikacji poprzez interfejs RMI - *Remote Method Invocation* czy też usługi sieciowe, itp.) i przekazywanie ich do warstwy przetwarzania danych surowych (*raw data retrieval*). Ta z kolei przetwarza je przy pomocy sterowników (w przypadku źródeł danych mierzących fizyczne własności - czujników) bądź odpowiedniego API (w przypadku komunikacji z innych systemami). Odpowiednio zinterpretowane i zorganizowane (na przykład w postaci dokumentów XML) dane są następnie przedstawiane przy pomocy ujednoliconego interfejsu do warstwy wstępnego przetwarzania (*preprocessing*), której zadaniem jest agregowanie, formatowanie i takie przekształcenie informacji, aby były one użyteczne dla wyżej położonych w hierarchii abstrakcji komponentów systemu (np. konwersja temperatury na binarną wartość mówiącą czy pojawił się ogień). Warstwa prezentacji (*presentation/storage/management*) odpowiada za dostarczanie przetworzonych danych do

ostatniej warstwy aplikacji - klientów (*application*), gdzie dochodzi do interpretacji informacji oraz wykonywania właściwej logiki aplikacyjnej (na przykład interakcji z użytkownikiem). Model ten jest zbliżony do modelu aplikacji dla przestrzeni inteligentnej, a także niektóre jego warstwy mogą być zrealizowane w samej przestrzeni (patrz rozdział II.2.4).

Organizacja struktury aplikacji zależy głównie od dostępności sensorów zawartych w obiektach DIND, liczby użytkowników systemu i innych zasobów. Biorąc pod uwagę te elementy można wyodrębnić trzy główne sposoby realizacji architektury aplikacji kontekstowych [15] (podobnie do sposobu realizacji przestrzeni inteligentnych uwzględniającego podejście zamknięte i oparte o warstwę pośredniczącą):

- bezpośredni dostęp do czujników - podejście najczęściej spotykane w sytuacjach, gdy czujniki są bezpośrednio wykorzystywane przez aplikacje. Wadą tego podejścia jest utrudniona skalowalność, możliwe ograniczenie przyszłego rozwoju aplikacji oraz trudności w ich wykorzystaniu w środowisku rozproszonym,
- dostęp do czujników przy wykorzystaniu warstwy pośredniczącej - podejście to opiera się na koncepcji rozdzielenia dostępu do sensorów od danych przez nie dostarczanych. Do realizacji tego celu służy warstwa pośrednicząca, która ukrywa szczegóły techniczne wymagane do komunikacji z urządzeniem końcowym (DIND) i przekazuje jedynie wartości przez nie zwracane,
- architektura oparta na serwerze kontekstu – umożliwiająca jednoczesne korzystanie przez wiele aplikacji z danych kontekstowych gromadzonych na serwerze kontekstu. Zaletą tego rozwiązania jest odciążenie urządzeń końcowych (które często mają niewielką moc obliczeniową) od skomplikowanego przetwarzania danych, na których muszą działać i przeniesienia tych operacji na serwer.

Obecnie zdecydowana większość architektur dla aplikacji kontekstowych zakłada istnienie wyspecjalizowanego modułu do analizy kontekstu [13][148][155]. Przykład zasady działania takiego modułu został przedstawiony w pracy [70], gdzie do analizy danych kontekstowych zastosowano zmodyfikowaną sieć Petrie'go z adnotacjami. Moduły te często wyposażone są w zaawansowaną logikę wnioskowania i przetwarzania danych oraz posiadają możliwość generowania bardziej abstrakcyjnych kontekstów na bazie surowych danych dostarczanych przez sensory zawarte w obiektach DIND [60]. W dynamicznym środowisku, jakim jest przestrzeń inteligentna, ich zastosowanie umożliwia dostosowywanie analizy kontekstu do bieżących jego możliwości (np. innych aplikacji udostępniających usługi w znaczeniu SOA). Przykład mechanizmu umożliwiającego takie dostosowanie (tzn. korzystanie w czasie analizy kontekstu z usług znajdujących się

w przestrzeni inteligentnej) został przedstawiony w pracy [71]. Mechanizm ten umożliwia automatyczne wyszukiwanie i wywoływanie usług działających w oparciu o standard Web Service, które odpowiadają semantycznie warunkowi jaki musi spełniać dana kontekstowa (kreująca kontekst) oraz posiadają odpowiedni interfejs, do tego aby mogły taką daną przeanalizować. Rozwiązanie przedstawione w tej publikacji korzysta z bazy wiedzy opisanej przy pomocy ontologii, w której jest zawarta wiedza na temat kontekstu działania aplikacji oraz obecnych w przestrzeni usług (wraz z informacjami na temat sposobu wywołania usługi i koniecznymi transformacjami danych kontekstowych do postaci zrozumiałej przez tę usługę).

Aspekt kontekstowości, w odniesieniu do aplikacji wszechobecnych [34][56][109], powoduje że mogą być one w różny sposób wytwarzane. W publikacji [153] podjęto dyskusję na temat wad i zalet modeli programowania opartych o kontekst (*context-driven model* - CDM) i opartych o usługi (*services-oriented model* - SOM). Zalety modelu CDM to przede wszystkim działanie w oparciu o wykrywanie stanów otoczenia. W tym podejściu kontekst jest wyrażony w sposób bezpośredni (precyzyjnie), dzięki czemu łatwo można wykryć sprzeczne jego określenia i niespójności oraz łatwiej można „uczyć" aplikację (w zakresie sposobu zmiany sposób zachowania w zależności od uprzednio uzyskanych wyników działania). Aplikacje zbudowane zgodnie z tym modelem są łatwo rozszerzalne (należy jedynie określić nowy kontekst) co bezpośrednio przekłada się na skalowalność (istnieje jeden logiczny obiekt w systemie odpowiadający za interpretację kontekstu). W przypadku modelu SOM aplikacja ma większą kontrolę nad przebiegiem działań (usługi cały czas obserwują stan otoczenia, dzięki czemu mogą przeprowadzać bardziej skomplikowane analizy i podejmować bardziej zaawansowane działania). Aby osiągnąć ten sam poziom kontroli w przypadku modelu opartego o kontekst należałoby zdefiniować bardzo dużą liczbę kontekstów (bądź ich opis byłby skomplikowany) – co z implementacyjnego i utrzymaniowego punktu widzenia może być trudne. Podejście oparte o wykorzystanie kontekstu (*context-driven*) z natury analizuje bieżący stan środowiska. Oznacza to, że przy jego zastosowaniu trudniej analizować dane kontekstowe, jakie zostały zgromadzone wcześniej. SOM umożliwia wykorzystanie takich danych (historycznych) i podejmowanie decyzji opartych o wnioskowanie. Ostatnią podstawową różnicą, przemawiającą na korzyść modelu SOM, jest jego częstsze wykorzystywanie przez twórców oprogramowania. Stosowanie modelu CDM oznacza całkowitą zmianę sposobu myślenia podczas wytwarzania aplikacji. Przykładem wyzwań związanym z modelem CDM jest dodanie funkcjonalności w aplikacji, które w praktyce może oznaczać konieczność wprowadzenia zmian w dużej liczbie już zdefiniowanych kontekstów. Powoduje to znaczą komplikację procesu wytwarzania i wdrażania aplikacji. Praca [153] proponuje połączenie obu tych modeli, którego wynikiem jest rozszerzone

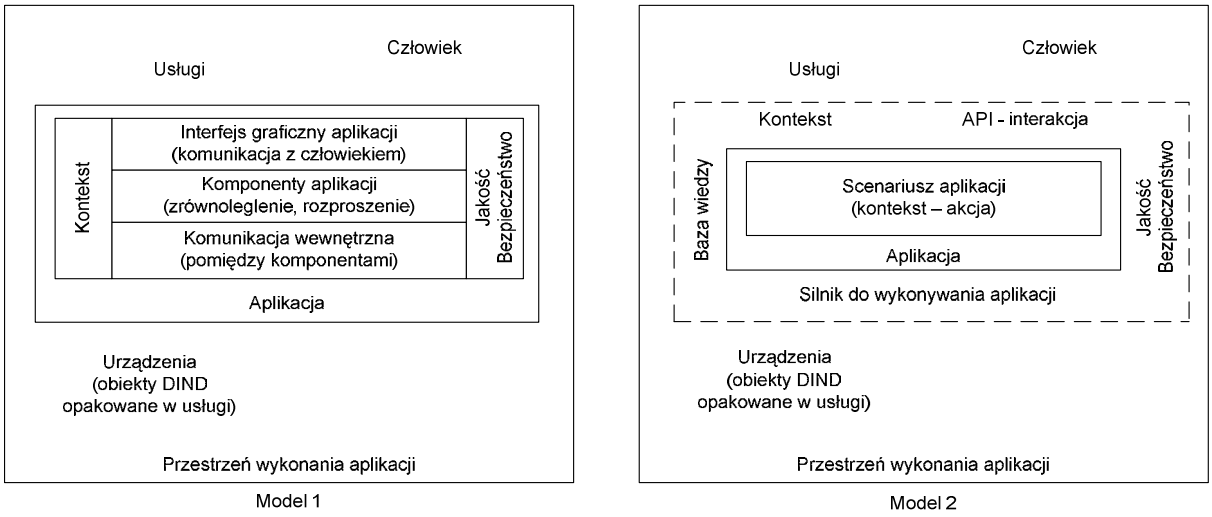
podejście oparte o kontekst (*enhanced context-driven approach*). Większość tak zintegrowanego modelu pozostaje taka sama jak w modelach poprzednich, ale proponuje się wykorzystanie usług obejmujących (opakowujących) działanie obiektów DIND.

Aplikacje IAI, których definicja została podana w dalszej części rozprawy, zostaną sformułowane przy wykorzystaniu przytoczonej ogólnej definicji kontekstu podanej przez A.K. Dey'a. Na jej podstawie kontekst został określony jako fragment przestrzeni, w jakim działa aplikacja i w zależności od wymagań stawianych tej aplikacji możliwa jest jego kategoryzacja zgodnie z podziałem przedstawionym w niniejszym rozdziale. Jako środek opisu kontekstu wybrano ontologię, ze względu na prostotę i elastyczność w odniesieniu do opisu przy jej pomocy przygotowanego (w zakresie łatwości dodawania nowych pojęć i utrzymania spójności). Idea świadomości kontekstu (*context awareness*) była punktem wyjścia do przygotowania definicji aplikacji IAI (która umożliwia tworzenie aplikacji przy wykorzystaniu podejścia *enhanced context-driven approach*), natomiast ogólny model warstwowy aplikacji kontekstowych został wykorzystany do rozdzielenia funkcji koniecznych do istnienia aplikacji w przestrzeni pomiędzy IAI oraz środowisko jej wykonywania (zbudowane na bazie maszyny wprowadzonej w rozdziale V).

III.5. Umiejscowienie aplikacji w przestrzeni inteligentnej

Jak zostało to przedstawione w rozdziale II przestrzeń inteligentna i aplikacje w niej działające coraz silniej się przenikają. Aplikacje zaczynają określać stopień inteligencji przestrzeni, podczas gdy same przestrzenie, będąc środowiskami ich wykonywania, dostarczają do realizacji tego procesu (wykonania) niezbędnych mechanizmów. Oznacza to, nowe podejście do opisu zależności między przestrzenią inteligentną a aplikacją kontekstową w niej działającą. Zostało one przedstawione na rysunku 11. Pierwszy z przedstawionych modeli (Model 1) opisuje możliwą konstrukcję aplikacji dla podejścia zamkniętego gdzie wszystkie mechanizmy potrzebne do działania aplikacji są wewnątrz niej zaszyte (patrz rozdział II.2.4). Oznacza to, że aplikacja składa się z określonych komponentów, między którymi zachodzi komunikacja wewnętrzna. W celu udostępnienia oferowanej przez nią funkcjonalności człowiekowi posiada ona także interfejs graficzny. We wszystkich aspektach jej działania konieczne jest zapewnienie wymaganego poziomu jakości oraz bezpieczeństwa, a także wykonywana jest analiza kontekstu aby dostosowywać działanie aplikacji do zmian zachodzących w przestrzeni. Znajdują się w niej także usługi (w znaczeniu SOA), ludzie oraz urządzenia, z którymi aplikacja może współpracować.

Model 2 odpowiada podejściu opartemu o warstwę pośredniczącą (patrz rozdział II.2.4). Oznacza ono wydzielenie w ramach przestrzeni inteligentnej warstwy, która



Rysunek 11 Ogólne modele aplikacji kontekstowej dla przestrzeni inteligentnej

realizuje określoną funkcjonalność konieczną dla działania aplikacji (np. dostarcza mechanizmy zapewniające bezpieczeństwo, albo interfejs API do innych aplikacji). Dzięki istnieniu takiej warstwy skraca się czas tworzenia aplikacji (ponieważ wiele aplikacji może korzystać z oferowanych przez przestrzeń funkcjonalności), a także systematyzuje się sposób ich projektowania. Przy wykorzystaniu tego podejścia możliwe staje się budowanie aplikacji definiowalnych (patrz rozdział II.2.4), które wymagają od użytkownika jedynie określenia scenariusza ich działania (tzn. na jaki kontekst i w jaki sposób reagować), natomiast warstwa pośrednicząca przyjmuje postać silnika do wykonywania takich scenariuszy, który dostarcza wszystkich niezbędnych do tego celu mechanizmów. Aplikacje IAI są osadzone w przestrzeni inteligentnej zgodnie z Modelem 2.

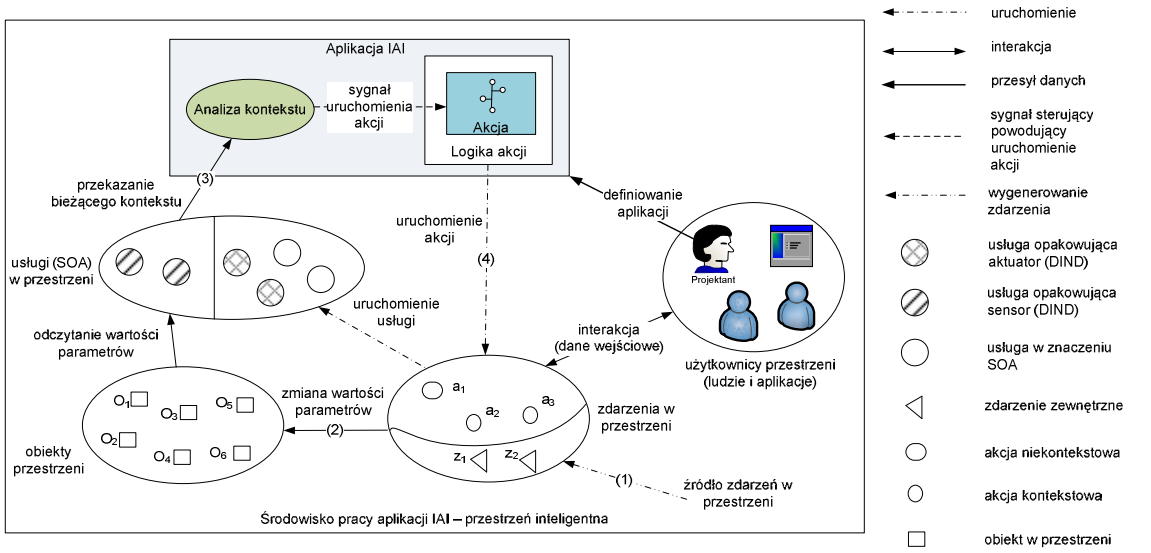
W niniejszym rozdziale zostały opisane własności jakie posiada aplikacja IAI. Przeanalizowane zostały poszczególne aspekty jej działania i dla każdego z nich przedstawiono wpływ na rozważania prowadzane w dalszej części rozprawy. Wyjaśniono termin iteracyjność w odniesieniu do tych aplikacji oraz podano narzędzie opisu jej interaktywności i działania w sposób równoległy oraz w czasie rzeczywistym. Sprecyzowane zostało znaczenie słowa kontekst jako fragment przestrzeni inteligentnej, w ramach którego działa aplikacja oraz z jakim wchodzi w interakcję, a także sposoby organizacji procesu jego pozyskiwania. Przedstawiono także sposób umiejscowienia oraz współpracy aplikacji IAI w przestrzeni inteligentnej. W następnym rozdziale został wprowadzony opis funkcjonalny tego rodzaju aplikacji, uwzględniający opisane własności.

IV OPIS FUNKCJONALNY IAI

W rozdziale podano nieformalny opis aplikacji IAI (Interaktywnej Aplikacji Iteracyjnej), których podstawowe zasady działania wyjaśniono na przykładzie. Zasady te odnoszą się do zapewnienia właściwości aplikacji wyszczególnionych w rozdziale III. Na podstawie tego opisu opracowano modele (rzeczywistego) środowiska działania aplikacji (tzn. przestrzeni inteligentnej) oraz jej kontekstu. Ich zdefiniowanie pozwoliło następnie na utworzenie modelu samej aplikacji. Zaproponowane w niniejszym rozdziale modele wykorzystują teorię zbiorów i relacji. Zachowanie aplikacji (z punktu widzenia adaptacji do nowych sytuacji mających miejsce w przestrzeni jej działania) opisano również przy pomocy automatu pozwalającego wyrazić aspekty działania w czasie rzeczywistym i równoległości wykonania akcji (adaptujących aplikację do nowych sytuacji w przestrzeni).

IV.1. Nieformalny opis IAI

Aplikacje kontekstowe należą do najbardziej zaawansowanych aplikacji z punktu widzenia modelowania ich zachowania. Ich działanie (logika aplikacji obejmująca współpracę z użytkownikiem) powinno dostosowywać się do bieżącego kontekstu (patrz rozdział III.4). Oznacza to, iż wchodzą w interakcję nie tylko z użytkownikami przestrzeni inteligentnej lecz również i z samą przestrzenią (patrz rysunek 12). Powoduje to, że muszą analizować i przetwarzać dwa rodzaje danych: kontekst oraz dane pochodzące od współpracujących z aplikacją użytkowników przestrzeni (dane wejściowe). Na podstawie pierwszego rodzaju aplikacje uzyskują informacje o sytuacjach (wyrażonych przez stan przestrzeni) jakie zachodzą w przestrzeni. Ze względu na ciągły charakter ich zmian, procesy dostosowania zachowania się aplikacji można zamodelować przy pomocy iteracji, gdzie kolejne kroki iteracyjne dotyczą przeanalizowania bieżącego kontekstu aplikacji oraz wykonania odpowiedniej akcji adaptacyjnej (w skład której wchodzi uruchomienie usług dostępnych w przestrzeni). Zatem ich struktura odpowiada klasycznemu algorytmowi iteracyjnemu (patrz własności opisane w rozdziale III). Biorąc pod uwagę interaktywność i iteracyjny charakter działania te aplikacje są oznaczane jako IAI - Interaktywne Aplikacje Iteracyjne. Istotnymi ich aspektami są również działanie w czasie rzeczywistym oraz równoległość analizy danych kontekstowych i wykonania akcji. Najbardziej ogólna idea takich aplikacji wraz z ich otoczeniem (przestrzenią inteligentną) została przedstawiona na rysunku 12.

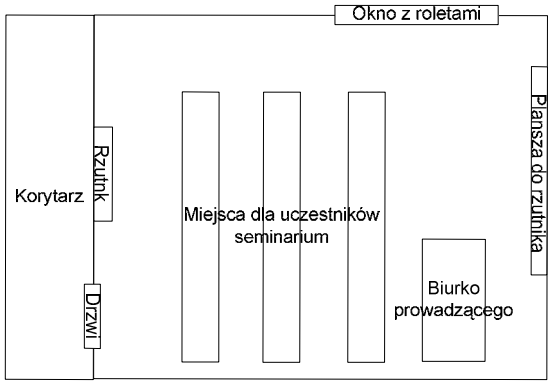


Rysunek 12 Funkcjonowanie aplikacji IAI w przestrzeni inteligentnej

Sytuacje powodowane są przez zdarzenia pojawiające się w przestrzeni, które można podzielić na zewnętrzne (tzn. takie, na które aplikacja nie ma wpływu) oraz wewnętrzne (tzn. takie, które są wynikiem działania aplikacji – wykonania akcji). Zatem działanie aplikacji rozpoczynają zewnętrzne zdarzenia (1) - np. rozpoczęcie zajęć, czy zmniejszenie zachmurzenia powodujące zwiększenie poziomu naświetlenia w sali. Powodują one zmiany w przestrzeni jej działania (w obiektach w niej istniejących - np. zmiana wartości parametru opisującego czy w sali seminaryjnej trwają zajęcia, zwiększenie poziomu naświetlenia w sali seminaryjnej) (2). Stan fragmentu przestrzeni istotny z punktu widzenia aplikacji jest odczytywany poprzez obiekty DIND pełniące rolę sensorów (3). Mogą one odczytywać zarówno dane logiczne (takie jak parametr opisujący czy w sali seminaryjnej trwają zajęcia zapisany np. w bazie danych) jak i fizyczne (jak poziom naświetlenia). Stan ten jest następnie przekazywany (przez opakowujące je usługi - patrz rozdział II.2.1) do aplikacji IAI w postaci danych kontekstowych i analizowany (pod kątem spełnienia tzw. oczekiwanych warunków wywołania akcji) przez mechanizmy wykonywania aplikacji IAI. Efektem tej analizy może być uruchomienie akcji (rozumianej jako usługa opakowująca aktuator bądź inna usługa w znaczeniu SOA) zdefiniowanej w aplikacji (4) (np. wyłączenie muzyki bądź opuszczenie rolet). W trakcie jej wykonywania może zachodzić interakcja z użytkownikami przestrzeni. Może ona również spowodować wystąpienie zdarzenia wewnętrznego (jakim jest wyłączenie muzyki lub zmniejszenie poziomu naświetlenia), rozumianego jako zmiana stanu przestrzeni na skutek działania aplikacji IAI. Zmiana taka, będąca wynikiem zaistnienia określonej sytuacji w przestrzeni oznacza, iż aplikacja IAI wchodzi w interakcję z przestrzenią inteligentną. Aspekt interaktywności

został opisany szerzej w rozdziale IV.2, gdzie znajduje się także jego odniesienie do modelu MVC. Podczas wykonywania akcji mogą pojawiać się kolejne zdarzenia zewnętrzne (1). Skutki wykonania akcji oraz zdarzenia zewnętrzne ponownie mogą zmieniać stan obiektów przestrzeni (2) i kolejne działania są kontynuowane (tworząc kolejne kroki iteracyjne). Logika działania aplikacji może zostać ograniczona do samych akcji, jeżeli głównym zadaniem aplikacji jest reagowanie na sytuacje pojawiające się w przestrzeni inteligentnej. Równie dobrze może ona jedynie podlegać pewnym modyfikacjom (wywołanym przez akcje) na skutek tych sytuacji. Na rys. 12 spośród ludzi został wyróżniony specjalny użytkownik przestrzeni – projektant, który określa na jakie sytuacje w przestrzeni (tzn. w jakim kontekście) oraz w jaki sposób aplikacja ma reagować.

Dla zilustrowania sposobu definiowania i opisu zasad działania aplikacji IAI rozpatrzymy wydzieloną przestrzeń inteligentną przedstawioną na rysunku 13. W skład rozpatrywanej przestrzeni wchodzi sala seminaryjna (o identyfikatorze Sala 101) oraz przylegający do tej sali korytarz. W przestrzeni istnieją obiekty stacjonarne, takie jak okna, rolety, drzwi oraz obiekty mobilne takie jak ludzie. Każdy z tych obiektów może być opisany pewnymi parametrami (np. w przypadku rolet istnieje parametr określający czy są opuszczone czy podniesione).



Rysunek 13 Przykładowa przestrzeń inteligentna

Przyjmijmy, że w sali seminaryjnej mogą odbywać się zajęcia (np. seminaria), które są rozdzielone przerwami (tzn. między dwoma różnymi zajęciami jest przerwa). Dla sali podano harmonogram zajęć jakie są w niej prowadzone. W zajęciach może uczestniczyć pewna liczba osób (np. 20), które są identyfikowane (podczas wchodzenia do sali) przy pomocy czytnika RFID zamontowanego w drzwiach wejściowych do sali. Po rozpoczęciu danych zajęć, następuje rozdystrybuowanie odpowiednich materiałów dydaktycznych na urządzenia przenośne ich uczestników. Materiały te muszą być zapisane przez osobę

prowadzącą zajęcia (przed ich rozpoczęciem) na serwerze dydaktycznym, do którego uczestnicy mają dostęp przez portal WWW. Podczas trwania zajęć wymagane jest sterowanie roletami w oknach, tak aby w zależności od nasłonecznienia zewnętrznego w sali było widać prezentacje wyświetlane przy pomocy rzutnika. Przyjęto założenie, że w czasie przerw pomiędzy zajęciami prowadzonymi w sali seminaryjnej mogą przebywać osoby (które np. uczestniczyły w zakończonych zajęciach). Dla nich z systemu nagłośnienia jest transmitowana muzyka, której głośność należy dostosowywać do prowadzonych w sali rozmów, tak aby uczestniczące w nich osoby nie musiały jej przekrzykiwać.

Rozpatrzmy dwie kategorie aplikacji wspomagającą prowadzenie zajęć (seminarium) w rozpatrywanej przestrzeni: niekontekstową i kontekstową. Dla aplikacji, która nie posiada cech kontekstowości (nie dostosowuje się do aktualnego stanu przestrzeni w jakiej działa), jej funkcjonowanie można opisać za pomocą następującego scenariusza:

- zapisanie przez osobę prowadzącą seminarium materiałów dydaktycznych na serwerze dydaktycznym,
- pobieranie poprzez portal WWW materiałów dydaktycznych z serwera dydaktycznego przez uczestników seminarium (muszą się oni zalogować i wskazać seminarium do którego chcą pobrać materiały),
- opuszczenie bądź podniesienie przez osobę prowadzącą rolet w oknach na skutek zmian zachmurzenia (co powoduje zmiany poziomu naświetlenia w sali),
- włączenie i wyłączenie muzyki transmitowanej przez system nagłaśniający (odpowiednio po zakończeniu i przed rozpoczęciem seminarium),
- zmniejszanie i zwiększanie głośności muzyki przez rozmówców pozostających w czasie przerwy w sali seminaryjnej, ze względu na konieczność jej przekrzykiwania.

Zestaw wymienionych funkcjonalności pozwala na prowadzenie zajęć w sali seminaryjnej. Może być on zrealizowany za pomocą jednej lub kilku aplikacji, które wymagają interwencji osób znajdujących się w sali. Jednakże niezależnie od tego co się w niej dzieje będzie ona działała w taki sam sposób (np. niezależnie od tego czy jest czy nie ma przerwy muzyka może być wyłączona – w zależności od decyzji osób znajdujących się w sali). Nie mniej aplikacja ta (bądź aplikacje) jest zarówno interaktywna (umożliwia umieszczenie i pobranie materiałów dydaktycznych na serwerze) jak i działa w czasie rzeczywistym (umieszczone materiały są natychmiast udostępnione do pobrania uczestnikom). Uczestników korzystających z portalu WWW w tym samym czasie jest wielu, także że każdy z nich obsługiwany jest w sposób równoległy.

Opisana aplikacja wzbogacona o aspekt kontekstowości będzie działać według następującego scenariusza:

- zapisanie przez osobę prowadzącą seminarium materiałów dydaktycznych na serwerze dydaktycznym,
- utworzenie listy uczestników podczas ich przechodzenia przez drzwi, które mają wbudowany odpowiedni czytnik RFID (komunikujący się z znacznikami RFID które posiadają uczestnicy),
- każda osoba uczestnicząca w seminarium dostaje wiadomość email, którą może odczytać na swoim urządzeniu przenośnym (z którym przebywa w sali), w której znajduje się link do serwera dydaktycznego umożliwiający pobranie materiałów do seminarium odbywającego się w sali (które materiały mają być pobrane jest określane na podstawie harmonogramu zajęć oraz identyfikatora sali). Materiały te są dostosowywane do formatu umożliwiającego wyświetlenie na urządzeniu przenośnym, z którego korzysta dana osoba. Osoby uczestniczące w seminarium pobierają następnie poprzez portal WWW materiały dydaktyczne z serwera,
- zmniejszenie (zwiększenie) zachmurzenia powoduje zwiększenie (zmniejszenie) poziomu naświetlenia w sali co powoduje automatyczne opuszczenie (podniesienie) rolet zamontowanych w oknach,
- w momencie rozpoczęcia przerwy pomiędzy seminariami (tzn. w momencie zakończenia seminarium - czyli w momencie gdy mija termin zakończenia seminarium wynikający z harmonogramu zajęć i z sali wyszła ponad połowa uczestników) następuje automatyczne włączenie muzyki odgrywanej przez system nagłaśniający,
- w momencie gdy mija termin rozpoczęcia seminarium wynikający z harmonogramu zajęć następuje automatyczne wyłączenie muzyki odgrywanej przez system nagłaśniający,
- w momencie rozpoznania wysokiego (niskiego) poziomu natężenia głosu osób rozmawiających w czasie przerwy, następuje automatyczne zmniejszenie (zwiększenie) głośności muzyki.

Działania podejmowane przez aplikację wspomagającą obsługę sali seminaryjnej, jakie są przedstawione w scenariuszu, są dokładnie takie same jak w poprzednim (dotyczącym aplikacji niekontekstowej), jednakże jej działanie wyraźnie się zmienia. W zależności od tego jakie zmiany zachodzą w przestrzeni jej funkcjonowania, aplikacja samoczynnie reaguje w odpowiedni sposób. Podczas przerwy przez system nagłośnienia odgrywana jest muzyka, ale poziom głośności nie jest cały czas taki sam – zmienia się razem ze zmianą

natężenia głosu osób rozmawiających. Uczestnicy zajęć zaraz po wejściu do sali otrzymują link do swoich materiałów dydaktycznych i nie muszą po zalogowaniu na portal WWW (udostępniający materiały z serwera dydaktycznego) wskazywać seminarium, w którym uczestniczą. Dodatkowo materiały są automatycznie konwertowane do formatu umożliwiającego ich wyświetlenie na urządzeniu przenośnym danego uczestnika. Należy podkreślić, że w tym przypadku wszystkie informacje pozwalające na takie działanie zostały odczytane z przestrzeni. Dla porównania, w przypadku aplikacji niekontekstowej można uzyskać bardzo podobne zachowanie, lecz wtedy wszystkie niezbędne informacje (takie jak np. identyfikator sali, w której znajduje się uczestnik seminarium i aktualny czas, pozwalające określić w jakim seminarium uczestniczy osoba) muszą być przekazane przez użytkownika jako dane wejściowe. Przedstawiony przykład dobrze obrazuje sposób konstrukcji aplikacji typu IAI, dla których podstawą działania jest rozpoznawanie kontekstu (patrz rozdział III.4) i odpowiednie reagowanie, wspólnie tworzące pojedynczy krok iteracyjny.

Dla rozpatrywanego przykładu zdarzeniami zewnętrznymi, jakie pojawiają się w przestrzeni inteligentnej podczas działania aplikacji są:

- przejście przez drzwi uczestnika seminarium,
- zwiększenie (zmniejszenie) zachmurzenia powodujące zmniejszenie (zwiększenie) poziomu naświetlenia,
- nadejście terminu rozpoczęcia bądź zakończenia seminarium,
- pojawienie się zwiększonego (zmniejszonego) poziomu głośności rozmowy.

Spowodowany przez nie (zdarzenia) nowy kontekst musi być dostępny dla aplikacji przez czas potrzebny do jego analizy. Oznacza to, iż z każdym odczytem sensora jest związany czas ważności, określający jak długo dostarczone przez niego dane mogą być analizowane przez aplikację (np. przejście przez drzwi uczestnika seminarium powinno powodować dostarczenie przez sensor danej z czasem ważności zapewniającym możliwość jej analizy do czasu dodania tej osoby do listy uczestników seminarium).

Rozpatrywana przestrzeń zawiera następujące obiekty (i opisujące ich parametry):

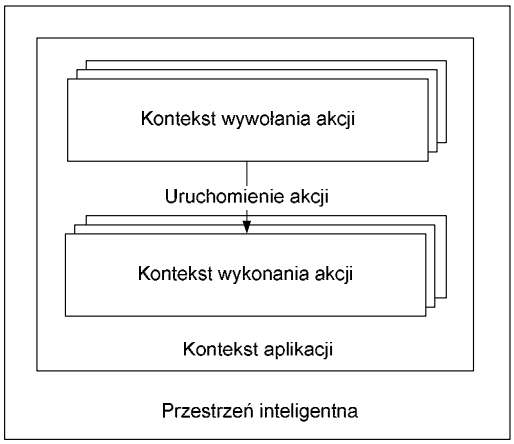
- sala seminaryjna (identyfikator, poziom naświetlenia, drzwi wejściowe, okno, czy trwają zajęcia, lista osób w sali, liczba osób w sali, czy wysłano materiały dydaktyczne do uczestników, czy pora seminarium),
- korytarz wraz z przemieszczającymi się osobami,
- drzwi z zamontowanym czytnikiem RFID (rozpoznana osoba),
- okno z zamontowanymi roletami (czy rolety są opuszczone),

- osoby (identyfikator, natężenie głosu, akceptowany przez urządzenie przenośne format dokumentów, adres email),
- system nagłośnienia (czy muzyka jest odgrywana, poziom głośności muzyki),
- rzutnik (wyświetlana prezentacja),
- serwer dydaktyczny (prezentacje dotyczące seminariów),
- portal WWW (skonfigurowany serwer dydaktyczny).

Niektóre parametry służą jedynie do powiązania obiektów przestrzeni (na przykład aby powiązać listę osób z salą seminaryjną, musi istnieć parametr sali seminaryjnej wskazujący na tę listę). Należy także zwrócić uwagę na to, iż w zależności od dostępnych w przestrzeni sensorów w skład kontekstu mogą wchodzić różne obiekty. Na przykład usługa opakowująca sensor RFID zamontowany przy drzwiach wejściowych może dostarczać całą listę osób w sali seminaryjnej, lecz równie dobrze może dostarczać informację o tym, że konkretna osoba przeszła przez drzwi. W pierwszym przypadku obiektem może być zatem sala seminaryjna i parametrem lista osób, natomiast w drugim obiektem mogą być drzwi a parametrem rozpoznana osoba. Powoduje to, że analiza kontekstu (w zależności od sposobu jego opisu) będzie realizowana dla każdego z nich inaczej. W przykładzie obserwowany fragment przestrzeni (założono, że usługa opakowująca sensor w drzwiach dostarcza listę osób w sali seminaryjnej) obejmuje następujące obiekty wraz z określonymi parametrami:

- sala seminaryjna (identyfikator, poziom naświetlenia, okno, czy trwają zajęcia, lista osób w sali, liczba osób w sali, czy wysłano materiały dydaktyczne do uczestników, czy pora seminarium),
- okno z zamontowanymi roletami (czy rolety są opuszczone),
- osoby (identyfikator, natężenie głosu, akceptowany przez urządzenie przenośne format dokumentów, adres email),
- system nagłośnienia (czy muzyka jest odgrywana).

Nie przy każdej akcji wszystkie obiekty przestrzeni i ich parametry mają znaczenie (np. dla opuszczenia rolet nie ma znaczenia czy muzyka jest odgrywana). Co więcej obiekty przestrzeni wykorzystywane do rozpoznania czy należy uruchomić akcję nie muszą być wykorzystane podczas jej wykonania. Z tego powodu dla każdej aplikacji IAI można określić różnego rodzaju konteksty: kontekst wywołania akcji, kontekst wykonania akcji oraz kontekst aplikacji. Zależność między nimi a przestrzenią inteligentną została przedstawiona na rysunku 14.



Rysunek 14 Zależność między kontekstem i przestrzenią inteligentną

Z każdą akcją (których może być w aplikacji zdefiniowanych wiele) określoną w aplikacji IAI powiązany jest pewien fragment przestrzeni inteligentnej, który składa się z dwóch części (jakie mogą się częściowo pokrywać): kontekstu wywołania akcji oraz kontekstu wykonania akcji (patrz rysunek 14). Kontekst wywołania akcji jest kreowany przez obiekty, parametry i ich wartości, które w czasie działania aplikacji spełniły oczekiwane warunki wywołania akcji a przez to spowodowały jej uruchomienie. Przykładem takiego warunku jest określenie kiedy (i do kogo) wysłać materiały dydaktyczne po rozpoczęciu seminarium (patrz wiersz 3 z tabeli 1). Zatem kontekst wywołania akcji w tym wypadku obejmuje salę seminaryjną z listą osób i jej wartość parametru określającego czy rozpoczęło się w niej seminarium. Kontekst wykonania akcji zawiera obiekty, parametry i ich wartości angażowane w czasie działania akcji. Akcją jest wysłanie wiadomości email na urządzenia przenośne uczestników seminarium, w której znajduje się link do serwera dydaktycznego umożliwiający pobranie materiałów (w formacie umożliwiającym ich wyświetlenie na danym urządzeniu przenośnym). W związku z tym, w skład kontekstu wykonania akcji wchodzi zarówno sala seminaryjna i jej identyfikator jak również lista osób wraz z ich adresami email i formatami dokumentów wspieranymi przez posiadane przez nich urządzenia przenośne. Kontekst aplikacji obejmuje z kolei wszystkie obiekty (i ich parametry) jakie są obserwowane przez aplikację, aby stwierdzić czy wywołać (poprzez sprawdzenie oczekiwanych warunków wywołania akcji) i wykonać dowolną z akcji, które są zawarte w jej definicji. Może być on zmieniony przez projektanta aplikacji poprzez dodanie nowych oczekiwanych warunków wywołania akcji i odpowiadającej im akcji, bądź modyfikację samej akcji (powodującej zmianę kontekstu wywołania bądź wykonania akcji). Kontekst (niezależnie od rodzaju) tworzony jest na podstawie danych kontekstowych (którymi są wartości parametrów obiektów dostarczanych przez obiekty DIND pełniące rolę sensorów). Odnosząc jego analizę do Modelu 2 z rysunku 11 można

na tym polu wyszczególnić następujące podejścia:

- analizowanie i przetwarzanie kontekstu wywołania akcji w aplikacji oraz analizowanie i przetwarzanie kontekstu wykonania akcji w silniku,
- analizowanie i przetwarzanie zarówno kontekstu wywołania akcji jak i kontekstu wykonania akcji w silniku.

Drugie podejście oznacza, iż aplikacja dostarcza do silnika opis warunków dotyczących oczekiwanych przez nią kontekstów, natomiast mechanizm ich analizy znajduje się bezpośrednio w silniku. Zgodnie z rozdziałem II podejście to jest charakterystyczne dla aplikacji IAI. Warto także zauważyć że analiza kontekstu aplikacji w samej aplikacji odpowiada Modelowi 1 z rysunku 11.

Akcje podejmowane przez aplikację IAI, mogą być zależne bądź niezależne od kontekstu (patrz rysunek 14). Akcje niezależne od kontekstu reprezentują tradycyjną aplikację (tzn. niekontekstową), w której wpływ na działanie ma jedynie interakcja z użytkownikiem (tą to tzw. akcje niekontekstowe). Interakcja ta, w ramach której następuje przekazywanie do akcji danych wejściowych i odbieranie wyniku jej działania, zobrazowana jest na rysunku 12. Może mieć ona także miejsce w przypadku akcji kontekstowych, jednakże wtedy oprócz danych wejściowych, akcja analizuje także kontekst wykonania akcji (i w zależności od niego zmienia swoje zachowanie). W niniejszej rozprawie akcje rozumiane są jako usługi w znaczeniu SOA (tzn. jako funkcjonalności dostępne w przestrzeni inteligentnej, które są uruchamiane i wykonywane w ramach aplikacji IAI). Akcje występujące w rozpatrywanym przykładzie aplikacji wspomagającej obsługę sali seminaryjnej to:

- umieszczenie materiałów dydaktycznych na serwerze (akcja niezależna od kontekstu),
- wysyłanie spersonalizowanych wiadomości email do uczestników seminarium,
- opuszczanie i unoszenie rolet,
- włączanie i wyłączanie muzyki w systemie nagłaśniającym,
- zmniejszanie i zwiększanie poziomu głośności muzyki w systemie nagłaśniającym.

Dla podanego przykładu działanie aplikacji IAI zostało przedstawione w tabeli 1 (ze względu na przejrzystość opisu przyjęto założenie, iż w seminarium mogą uczestniczyć 3 osoby).

Tabela 1 Szczegółowy opis działania przykładowej aplikacji IAI

Przyczyna uruchomienia akcji	Akcja oraz dane (wejściowe bądź kontekst wykonania akcji)
Dane wejściowe Przekazanie przez użytkownika danych do aplikacji.	Udostępnienie na serwerze dydaktycznym materiałów do seminarium. Dane wejściowe: Materiały dydaktyczne (dane wejściowe od użytkownika).
Zdarzenie zewnętrzne Zdarzenie zewnętrzne wygenerowane przez usługę harmonogramu oznaczające nadejście pory rozpoczęcia seminarium w Sali 101. Oczekiwane warunki wywołania akcji (oz_1): (obiekt) sala seminaryjna: • (parametr) identyfikator (warunek) == Sala 101, • lista osób w sali składa się osób o identyfikatorach Jan Kowalski, Adam Nowak lub Krzysztof Wawrzyniak, • czy zajęcia == nie, • liczba osób w sali >= 2, • czy pora seminarium == tak, Kontekst wywołania akcji: sala seminaryjna: • identyfikator = Sala 101, • lista osób w sali = {osoba1, osoba2} • czy zajęcia = nie, • liczba osób w sali = 2, • czy pora seminarium = tak, osoba1: • identyfikator = Jan Kowalski, osoba2: • identyfikator = Adam Nowak.	Wykonanie akcji a_1 powodującej zmianę wartości parametru określającego czy w sali seminaryjnej trwają zajęcia (z nie na tak) oraz wyłączenie transmitowania muzyki z systemu nagłaśniającego. Kontekst wykonania akcji: sala seminaryjna: • identyfikator = Sala 101.

<table>
<tr>
<td>Zdarzenie wewnętrzne
Zdarzenie wewnętrzne spowodowane wykonaniem przez aplikację akcji a_1, polegającej na zmianie wartości parametru opisującego czy w Sali 101 trwają zajęcia.

Oczekiwane warunki wywołania akcji (oz_2):
sala seminaryjna:
• identyfikator == Sala 101,
• lista osób w sali składa się osób o identyfikatorach Jan Kowalski, Adam Nowak lub Krzysztof Wawrzyniak,
• czy zajęcia == tak,
• czy wysłano materiały dydaktyczne == nie,

Kontekst wywołania akcji:
sala seminaryjna:
• identyfikator = Sala 101,
• lista osób w sali = {osoba1, osoba2},
• czy zajęcia = tak,
• czy wysłano materiały dydaktyczne = nie,
osoba 1:
• identyfikator = Jan Kowalski,
osoba 2:
• identyfikator = Adam Nowak.</td>
<td>Wykonanie akcji a_2, w ramach której następuje wysłanie do każdej osoby uczestniczącej w seminarium wiadomości email, w której znajduje się link do serwera dydaktycznego umożliwiający pobranie materiałów (w formacie umożliwiającym wyświetlenie na urządzeniu przenośnym z którym przyszła do sali). Wysyłane linki odnoszą się bezpośrednio do materiałów do seminarium prowadzonego w danej sali. Następuje także zmiana wartości parametru sali mówiącego o wysłaniu materiałów dydaktycznych na „tak”.

Kontekst wykonania akcji:
sala seminaryjna:
• identyfikator = Sala 101,
• lista osób w sali = {osoba1, osoba2},
osoba 1:
• identyfikator = Jan Kowalski,
• akceptowany przez urządzenie przenośne format dokumentów = PPT dla Laptopa,
• adres email = Jan.Kowalski@zak.eti.pg.gda.pl,
osoba 2:
• identyfikator = Adam Nowak,
• akceptowany przez urządzenie przenośne format dokumentów = PPT dla PDA
• adres email = Adam.Nowak@zak.eti.pg.gda.pl.</td>
</tr>
<tr>
<td>Zdarzenie zewnętrzne
Zdarzenie zewnętrzne polegające na</td>
<td>Opuszczenie rolet (a_3).</td>
</tr>
</table>

zwiększeniu poziomu naświetlenia w Sali 101. <u>Oczekiwane warunki wywołania akcji (oz_3):</u> sala seminaryjna: • identyfikator == Sala 101, • poziom naświetlenia >= 40%, • czy zajęcia == tak, <u>Kontekst wywołania akcji:</u> sala seminaryjna: • identyfikator = Sala 101, • poziom naświetlenia = 45%, • czy zajęcia = tak.	<u>Kontekst wykonania akcji:</u> sala seminaryjna: • identyfikator = Sala 101.
<u>Zdarzenie zewnętrzne</u> Zdarzenia zewnętrzne polegające na wysłaniu przez usługę harmonogramu sygnału o zakończeniu seminarium w Sali 101 oraz opuszczaniu przez uczestników seminarium Sali 101. <u>Oczekiwane warunki wywołania akcji (oz_4):</u> sala seminaryjna: • identyfikator == Sala 101, • czy zajęcia == nie, • liczba osób w sali < 2, • czy pora seminarium == nie, <u>Kontekst wywołania akcji:</u> sala seminaryjna: • identyfikator = Sala 101, • czy zajęcia = nie, • liczba osób w sali = 1, • czy pora seminarium = nie.	Wykonanie akcji a_4 powodującej włączenie odgrywania muzyki w systemie nagłaśniającym oraz zmianę wartości parametru określającego czy w sali trwają zajęcia (na wartość „nie”). <u>Kontekst wykonania akcji:</u> sala seminaryjna: • identyfikator = Sala 101.
<u>Zdarzenie zewnętrzne</u> Zdarzenie zewnętrzne polegające na	Zmniejszenie głośności muzyki (a_5).

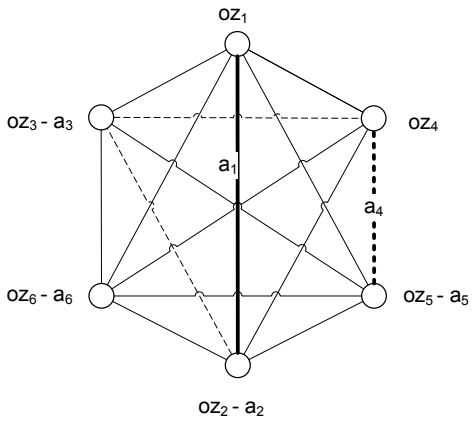
zwiększeniu poziomu głośności w Sali 101 podczas przerwy. Oczekiwane warunki wywołania akcji (oz_5): sala seminaryjna: • identyfikator == Sala 101, • lista osób w sali seminaryjnej zawiera osobę dla której natężenie głosu >= 30 %, • czy zajęcia == nie, system nagłośnienia: • czy muzyka odgrywana == tak, Kontekst wywołania akcji: sala seminaryjna: • identyfikator = Sala 101, • lista osób w sali = {osoba1, osoba2}, • czy zajęcia = nie, osoba1: • identyfikator = Jan Kowalski, • natężenie głosu = 40%, osoba2 • identyfikator = Krzysztof Wawrzyniak, • natężenie głosu = 43%, system nagłośnienia: • czy muzyka odgrywana = tak.	Kontekst wykonania akcji: sala seminaryjna: • identyfikator = Sala 101.

Tabela 1 przedstawia opis aplikacji, na który składają się pary: oczekiwane warunki wywołania akcji – akcja (kontekstowa) wraz z opisem jej działania dla konkretnych kontekstów wykonania akcji oraz pary: dane – akcja niekontekstowa (pierwszy wiersz tabeli), w których interakcja zachodzi między aplikacją i użytkownikiem (podobnie jak w aplikacjach interaktywnych opisanych w rozdziale III.3). Każda para opisana jest w osobnym wierszu i zawiera informację na temat przyczyny uruchomienia akcji, którą może być zdarzenie w przestrzeni (powodujące zaistnienie kontekstu spełniającego oczekiwane warunki wywołania akcji i przez to uruchomienie akcji kontekstowej),

albo przekazanie przez użytkownika danych wejściowych (powodujące uruchomienie akcji niezależnych od kontekstu). W przypadku akcji kontekstowych określone są oczekiwane warunki wywołania akcji oraz przedstawiony kontekst wywołania akcji, powodujący ich spełnienie. Zachowanie tak opisanej aplikacji zarówno z punktu widzenia jej funkcjonowania w przestrzeni inteligentnej, jak również samej jej postaci może być postrzegane przez pryzmat iteracyjności co jest szerzej opisane w następnym rozdziale.

IV.2. Graf stanów aplikacji IAI

Pojawianie się w przestrzeni inteligentnej kontekstu (sytuacji) spełniającego oczekiwane warunki wywołania akcji powoduje uruchomienie skojarzonej z nimi akcji. Z tego punktu widzenia można utożsamić stan aplikacji (wyrażający jaką wykonuje ona akcję i na jakich obiektach z przestrzeni) z kontekstem wywołania akcji (czyli stanem fragmentu przestrzeni), a co za tym idzie oczekiwanymi warunkami wywołania akcji. Powoduje to, iż aplikacja IAI przyjmuje postać klasycznego algorytmu iteracyjnego (jaki został opisany w rozdziale III.1), w którym etykiety odpowiadają kolejnym oczekiwanym warunkom wywołania akcji a operacje podejmowanym przez aplikację IAI akcjom. Jednakże przechodzenie pomiędzy kolejnymi stanami nie jest określone *a priori*. Aplikacja taka, wykonując pewną akcję, może powodować zmianę stanu przestrzeni, jednakże zawsze dodatkowe modyfikacje tego stanu mogą być wprowadzane przez zdarzenia zewnętrzne, na które sama aplikacja nie ma wpływu. Graf możliwych przejść stanów dla aplikacji IAI jest w związku z tym grafem pełnym. Przykład takiego grafu, utworzonego dla aplikacji opisanej w tabeli 1 został przedstawiony na rysunku 15.



Rysunek 15 Graf możliwych przejść stanów dla aplikacji IAI opisanej w tabeli 1

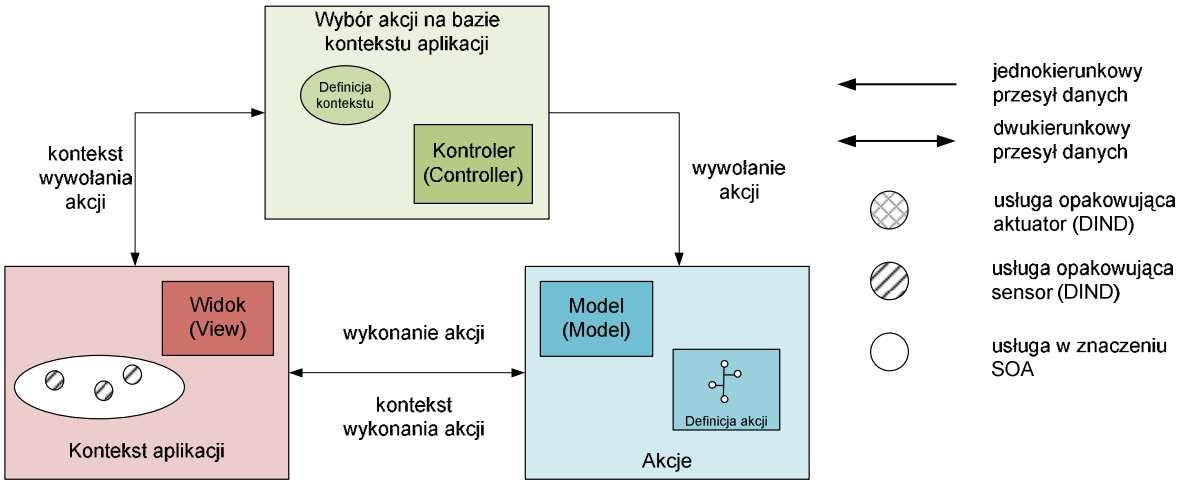
Węzły grafu reprezentują oczekiwane warunki wywołania akcji (oznaczane jako oz), które zostały spełnione przez kontekst aplikacji. Wykonane akcje są oznaczone jako a. Grubymi

krawędziami oznaczono zmiany stanu spowodowane działaniem aplikacji, cienkimi (nieprzerywanymi) zaś możliwe zmiany. Przejście aplikacji do nowego stanu może odbyć się na skutek wykonania akcji (krawędź pogrubiona), zajścia zdarzeń zewnętrznych (krawędź przerywana), bądź też zarówno akcji jak i zdarzeń zewnętrznych (pogrubiona krawędź przerywana).

W rozpatrywanym przykładzie z rozdziału IV.1 jako pierwszymi warunkami wywołania akcji rozpoznanymi podczas działania przykładowej aplikacji IAI są oz_1, odpowiadające rozpoznaniu sytuacji rozpoczęcia seminarium (jeżeli liczba osób jest większa bądź równa połowie osób mogących uczestniczyć w seminarium w czasie wynikającym z harmonogramu zajęć dla danej sali seminaryjnej). Mimo, że pierwszą akcją jaka została wykonana w ramach aplikacji było wgranie materiałów seminaryjnych przez osobę prowadzącą, to jednak nie była ona spowodowana zmianą kontekstu. Była to akcja uruchomiona przez użytkownika, w związku z tym z punktu widzenia kontekstu nie zostały rozpoznane żadne oczekiwane warunki wywołania akcji. Skutkiem wykonania akcji związanej z warunkami oz_1 jest zmiana wartości parametru opisującego czy w sali seminaryjnej trwają zajęcia. Powoduje to przejście aplikacji (i również przestrzeni) do kolejnego stanu rozpoznanego przez warunki oz_2. W tym stanie następuje wysłanie osobom będącym w sali wiadomości z linkami do materiałów dydaktycznych (lista osób wraz z akceptowanymi przez ich urządzenia przenośne formatami dokumentów została odczytana z kontekstu wykonania akcji) - a_2, jednakże nie powoduje to przejścia aplikacji do kolejnego stanu. Dopiero zwiększenie poziomu naświetlenia w sali seminaryjnej do poziomu powyżej 40% (rozpoznane przez warunki zawarte w oz_3) powoduje, iż aplikacja opuszcza rolety w oknie (a_3). W tym samym momencie czasu następuje zdarzenie zewnętrzne polegające na zakończeniu seminarium. Jednakże przejście do nowego stanu (rozpoznanego przez warunki oz_4) jest wynikiem jedynie zdarzenia zewnętrznego, ponieważ opuszczenie rolet nie ma wpływu na zakończenie seminarium. Aplikacja rozpoczyna odgrywać muzykę poprzez system nagłośnienia (a_4). W momencie gdy natężenie głosu osób prowadzących rozmowę w sali seminaryjnej przekroczy 30% następuje rozpoznanie oczekiwanych warunków wywołania akcji oz_5 (jako wyniku wykonania akcji włączenia muzyki oraz zdarzenia zewnętrznego – jej przekrzykiwania), w którym aplikacja ścisza grającą muzykę (a_5). Należy zwrócić uwagę, iż niektóre oczekiwane warunki zawarte w opisie aplikacji mogą nigdy nie zostać spełnione przez konteksty aplikacji (są one reprezentowane przez oz_6, jednakże ze względu na przejrzystość opisu nie zostały one uwzględnione w tabeli 1). Podobnie pozostałe krawędzie w grafie (reprezentowane cienką linią ciągłą) stanowią potencjalne przejścia pomiędzy stanami aplikacji (oczekiwanymi warunkami wywołania akcji). W określonych

warunkach (sytuacjach) występujących w przestrzeni przejścia te mogą zostać wykonane, ale nie muszą (zależy to od obecności zdarzeń zewnętrznych). Cechą charakterystyczną aplikacji IAI jest to, iż rzeczywiście wykonane przejścia w grafie nie muszą być powtarzalne nawet dla tych samych akcji (i co za tym idzie również kontekstów aplikacji), czego powodem jest obecność zdarzeń zewnętrznych.

Człowiek i aplikacje współistnieją obok siebie w przestrzeni inteligentnej, co powoduje iż w naturalny sposób wchodzą ze sobą (oraz z przestrzenią) w interakcję. Aplikacja IAI posiada dwie płaszczyzny interakcji z człowiekiem. Pierwsza z nich obejmuje tradycyjne dla wszystkich aplikacji interaktywnych (patrz rozdział III.3) analizowanie danych wejściowych i prezentowanie wyników (np. udostępnienie na serwerze dydaktycznym materiałów do seminarium). Działania te mogą mieć postać scenariusza, którego niektóre kroki musi wykonać człowiek (wtedy zachodzi interakcja obustronna). Druga dotyczy analizowania kontekstu aplikacji i wykonywania akcji, które następnie mogą spowodować zajście zdarzeń wewnętrznych (patrz rozdział IV.1) go zmieniających. Poprzez swoje zachowanie i podejmowane działania, człowiek również może wpływać na stan przestrzeni, w jakiej działa aplikacja IAI. Interaktywność tej (drugiej) płaszczyzny współpracy (charakterystyczną dla aplikacji IAI) można przedstawić przy pomocy wzorca MVC (rysunek 16).



Rysunek 16 Interakcja w aplikacji typu IAI w odniesieniu do wzorca MVC

Aplikacja IAI reaguje na pojawienie się w przestrzeni inteligentnej w jakiej działa jednego z kontekstów aplikacji spełniającego oczekiwane warunki wywołania akcji. Określenie czy kontekst spełnia warunki odpowiada funkcjonalności Widoku z modelu MVC. Wynikiem ich spełnienia jest następnie konieczność określenia, która akcja ma zostać wywołana.

Analogiczne analizowanie danych wejściowych przekazanych do Widoku realizowane jest w MVC przez Kontroler. Komponent Model reprezentuje logikę (sposób działania) zawartą w akcjach, które są wywoływane przez Kontroler. Na ich wykonywanie mają wpływ dane przekazane przez Widok (kontekst wykonania akcji). Podczas wykonywania akcje mogą zmieniać obiekty przestrzeni działania aplikacji co z kolei pośrednio wpływa na Widok (tzn. na kontekst aplikacji). Zmiany w Kontrolerze powodują również bezpośrednie zmiany w Widoku (jeżeli zmienią się oczekiwane warunki wywołania akcji). W następnych rozdziałach przeanalizujemy jak obserwować i jak implementować reakcję na zmiany kontekstu.

IV.3. Przestrzeń działania aplikacji kontekstowej

Jak zostało opisane w rozdziale IV.1 przestrzeń inteligentna (PI) jest środowiskiem działania aplikacji IAI, składającym się z obiektów, odpowiadających im parametrów oraz w ramach której działają usługi. Można ją zatem zdefiniować przy pomocy następującej czwórki:

$$PI = \langle O, P, \alpha, U \rangle \tag{4.1}$$

gdzie $O = \{o_1, o_2, \dots, o_{mO}\}$ są to obiekty reprezentujące obiekty przestrzeni PI, w której działa aplikacja, $P = \{p_1, p_2, \dots, p_{mP}\}$ jest to zbiór parametrów obiektów ze zbioru O, natomiast relacja α jest określona jako $o_i \alpha P_i$ i reprezentuje przyporządkowanie podzbioru parametrów $P_i \subseteq P$ do obiektów $o_i \in O$. Zbiór $U = \{u_1, u_2, \dots, u_{mU}\}$ reprezentuje usługi dostępne w przestrzeni. Obiekty środowiska mogą być zarówno fizyczne (na przykład sala) jak i niefizyczne (na przykład dane z bazy danych), a każdy z nich musi mieć przypisany przynajmniej jeden parametr – tzn.:

$$\bigwedge_{o_i \in O} \bigvee_{P_i \subset P} o_i \alpha P_i$$

Dodatkowo zostało przyjęte założenie, iż każdy parametr jest przypisany do przynajmniej jednego obiektu, czyli

$$P = \bigcup_i P_i$$

Podczas działania, aplikacja korzysta z usług istniejących w przestrzeni inteligentnej (patrz rysunek 12). Zgodnie z opisem w rozdziale IV.1 są to usługi w znaczeniu SOA, które można podzielić na następujące rodzaje:

- usługi opakowujące obiekty DIND pełniące rolę sensorów (usługi dostarczające dane kontekstowe),
- usługi opakowujące obiekty DIND pełniące rolę aktuatorów (usługi umożliwiające wykonywanie akcji),
- usługi w znaczeniu SOA (które mogą być uruchamiane podczas wykonywania akcji, albo w czasie analizy kontekstu – patrz rozdział III.4).

Dla przykładu aplikacji podanego w rozdziale IV.1 definicja *PI* (przy założeniu, że w przestrzeni znajdują się trzy osoby) wygląda następująco (tabela 2):

Tabela 2 Przykład definicji przestrzeni inteligentnej

Element definicji	Wartość
$O = \begin{Bmatrix} o_1, o_2, o_3, o_4, \\ o_5, o_6, o_7, o_8, \\ o_9, o_{10} \end{Bmatrix}$	o_1=sala seminaryjna, o_2=korytarz, o_3=drzwi w Sali 101 z czytnikiem RFID, o_4=okno w Sali 101 z zamontowanymi roletami, o_5=osoba1, o_6=osoba2, o_7=osoba3, o_8=system nagłośnienia, o_9=rzutnik, o_{10}=portal WWW na serwerze dydaktycznym.
$P = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8 \cup P_9 \cup P_{10}$	P_1= $\{p_{1_1}$= identyfikator, p_{1_2}= poziom naświetlenia, p_{1_3}= drzwi wejściowe, p_{1_4}= okno, p_{1_5}=czy zajęcia, p_{1_6}= lista osób w sali, p_{1_7}= liczba osób w sali, p_{1_8}= czy wysłano materiały dydaktyczne, p_{1_9}= czy pora seminarium$\}$ P_2= $\{p_{2_1}$= przemieszczające się osoby$\}$ P_3= $\{p_{3_1}$= czy otwarte, p_{3_2}= rozpoznana osoba$\}$ P_4= $\{p_{4_1}$= czy opuszczone$\}$ $P_5 = P_6 = P_7$=$\{p_{5_1} = p_{6_1} = p_{7_1}$= identyfikator, $p_{5_2} = p_{6_2} = p_{7_2}$= rozmówca, $p_{5_3} = p_{6_3} = p_{7_3}$= natężenie głosu, $p_{5_4} = p_{6_4} = p_{7_4}$= akceptowany przez urządzenie przenośne format dokumentów, $p_{5_5} = p_{6_5} = p_{7_5}$= adres email$\}$ P_8= $\{p_{8_1}$= czy muzyka odgrywana, p_{8_2}= poziom głośności muzyki$\}$ P_9= $\{p_{9_1}$= wyświetlana prezentacja$\}$ P_{10}= $\{p_{10_1}$= lista prezentacji dotyczących seminariów$\}$.
α	$\alpha(o_1) = P_1,\ \alpha(o_2) = P_2,\ \alpha(o_3) = P_3,\ \alpha(o_4) = P_4,\ \alpha(o_5) = P_5,$ $\alpha(o_6) = P_6, \alpha(o_7) = P_7, \alpha(o_8) = P_8, \alpha(o_9) = P_9, \alpha(o_{10}) = P_{10}.$

$U = \begin{Bmatrix} u_1, u_2, u_3, u_4, \\ u_5, u_6, u_7, u_8 \end{Bmatrix}$	u_1 = usługa odczytująca jakie osoby przechodzą przez drzwi (opakowująca sensor RFID), u_2 = usługa sprawdzająca czy minął termin rozpoczęcia seminarium, u_3 = usługa sprawdzająca liczbę osób w sali seminaryjnej, u_4 = usługa odczytująca poziom naświetlenia, u_5 = usługa umożliwiająca umieszczenie materiałów dydaktycznych na serwerze (akcja niezależna od kontekstu), u_6 = usługa wysyłająca spersonalizowane wiadomości email do uczestników seminarium, u_7 = usługa sterująca roletami, u_8 = usługa sterująca muzyką w systemie nagłaśniającym.

Jeżeli w przestrzeni znajduje się więcej obiektów tego samego typu każdy z nich należy osobno wymienić w definicji PI (tzn. jest osobnym obiektem oraz posiada osobny zbiór parametrów). Sytuacja taka zachodzi w przypadku osób (definicja zawiera 3 osoby) i jest spowodowana tym, iż obiekty te należą do możliwych wartości parametru lista osób. Zatem zmiana wartości ich (osób) parametrów oznacza zmianę stanu przestrzeni.

Stan przestrzeni inteligentnej tworzony jest poprzez przypisanie wartości (odczytywanych przez sensory umieszczone w obiektach DIND) do parametrów obiektów przestrzeni. Podobnie jak obiekty mogą być one fizyczne (na przykład natężenie światła, natężenie głosu) oraz niefizyczne (na przykład informacje z bazy danych). Zbiór wszystkich możliwych wartości jakie mogą przyjmować parametry obiektów jest określany jako $W = \{w_1, w_2, \dots\}$ (jest to zbiór nieskończony), natomiast zbiór możliwych stanów przestrzeni definiowany jest następująco:

$$SP = \{sp_1, sp_2, \dots, sp_{mSP}\} \tag{4.2}$$

gdzie każde $sp_l \in SP$ definiowane jest jako trójka:

$$sp_l = \langle PI, W_l, \beta \rangle \tag{4.3}$$

gdzie PI to przestrzeń inteligentna działania aplikacji (patrz definicja (4.1)), $W_l \subseteq W$ jest to zbiór wartości, jakie przyjmują parametry obiektów w definicji PI, natomiast relacja β jest określona jako $p_{i_j} \beta w_{l_{i_j}}$ i odzwierciedla przyporządkowanie wartości $w_{l_{i_j}} \in W_{l_i}$ (gdzie $W_{l_i} \subseteq W_l$) do parametrów $p_{i_j} \in P_i$ obiektów $o_i \in O$. Przyjęto założenie, że każdy parametr obiektu ma przypisaną jedną wartość, czyli:

$$\bigwedge_{p_{i_j} \in P_i} \bigvee_{w_{l_{i_j}} \in W_{l_i}} p_{i_j} \beta w_{l_{i_j}}$$

Wartości należące do zbioru W_l mogą być proste (na przykład wartość liczbowa), bądź złożone (na przykład zbiór obiektów wraz z parametrami i ich wartościami reprezentujący wcześniejszy stan przestrzeni). Analogicznie jak w przypadku parametrów obiektów zostało przyjęte założenie, iż każda wartość należąca do zbioru W_l jest przypisana do przynajmniej jednego parametru, dzięki czemu możliwy jest zapis:

$$W_l = \bigcup_i W_{l_i}$$

Stan przestrzeni $sp_l \in SP$ dla przykładu z rozdziału IV.1 opisujący zwiększony poziom naświetlenia został przedstawiony w tabeli 3 (definicja samej przestrzeni jest taka sama jak w tabeli 2).

Tabela 3 Przykład stanu przestrzeni

Element definicji	Wartość
PI	Definicja przedstawiona w tabeli 2.
$W_l = W_{l_1} \cup W_{l_2} \cup W_{l_3} \cup W_{l_4} \cup W_{l_5} \cup W_{l_6} \cup W_{l_7} \cup W_{l_8} \cup W_{l_9} \cup W_{l_{10}}$	W_1= {$w_{l_{1_1}}$= Sala 101, $w_{l_{1_2}}$= 45%, $w_{l_{1_3}}$= drzwi w Sali 101 z czytnikiem RFID, $w_{l_{1_4}}$= okno w Sali 101 z zamontowanymi roletami, $w_{l_{1_5}}$= zajęcia, $w_{l_{1_6}}$= {osoba1, osoba2}, $w_{l_{1_7}}$= 2, $w_{l_{1_8}}$= tak, $w_{l_{1_9}}$= tak} W_2 = {$w_{l_{2_1}}$= {osoba3}} W_3= {$w_{l_{3_1}}$= nie, $w_{l_{3_2}}$= osoba3} W_4= {$w_{l_{4_1}}$= nie} W_5= {$w_{l_{5_1}}$= Jan Kowalski, $w_{l_{5_2}}$= <brak>, $w_{l_{5_3}}$= 0%, $w_{l_{5_4}}$= PPT dla Laptopa, $w_{l_{5_5}}$= Jan.Kowalski@zak.eti.pg.gda.pl} W_6= {$w_{l_{6_1}}$= Adam Nowak, $w_{l_{6_2}}$= <brak>, $w_{l_{6_3}}$= 0%, $w_{l_{6_4}}$= PPT dla PDA, $w_{l_{6_5}}$= Adam.Nowak@zak.eti.pg.gda.pl} W_7= {$w_{l_{7_1}}$= Krzysztof Wawrzyniak, $w_{l_{7_2}}$= <brak>, $w_{l_{7_3}}$= 0%, $w_{l_{7_4}}$= <brak>, $w_{l_{7_5}}$= Krzysztof.Wawrzyniak@zak.eti.pg.gda.pl} W_8= {$w_{l_{8_1}}$= nie, $w_{l_{8_2}}$= 0%} W_9 = {$w_{l_{9_1}}$= prezentacja Jana Kowalskiego}

	W_{10} = {$w_{l_{10_1}}$= {prezentacja Jana Kowalskiego, prezentacja Adama Nowaka, prezentacja Krzysztofa Wawrzyniaka}}.
β	$p_{1_1}\beta w_{l_{1_1}}$, $p_{1_2}\beta w_{l_{1_2}}$, $p_{1_3}\beta w_{l_{1_3}}$, $p_{1_4}\beta w_{l_{1_4}}$, $p_{1_5}\beta w_{l_{1_5}}$, $p_{1_6}\beta w_{l_{1_6}}$, $p_{1_7}\beta w_{l_{1_7}}$, $p_{1_8}\beta w_{l_{1_8}}$, $p_{1_9}\beta w_{l_{1_9}}$, $p_{1_{10}}\beta w_{l_{1_{10}}}$, $p_{2_1}\beta w_{l_{2_1}}$, $p_{3_1}\beta w_{l_{3_1}}$, $p_{3_2}\beta w_{l_{3_2}}$, $p_{4_1}\beta w_{l_{4_1}}$, $p_{5_1}\beta w_{l_{5_1}}$, $p_{5_2}\beta w_{l_{5_2}}$, $p_{5_3}\beta w_{l_{5_3}}$, $p_{5_4}\beta w_{l_{5_4}}$, $p_{5_5}\beta w_{l_{5_5}}$, $p_{6_1}\beta w_{l_{6_1}}$, $p_{6_2}\beta w_{l_{6_2}}$, $p_{6_3}\beta w_{l_{6_3}}$, $p_{6_4}\beta w_{l_{6_4}}$, $p_{6_5}\beta w_{l_{6_5}}$, $p_{7_1}\beta w_{l_{7_1}}$, $p_{7_2}\beta w_{l_{7_2}}$, $p_{7_3}\beta w_{l_{7_3}}$, $p_{7_4}\beta w_{l_{7_4}}$, $p_{7_5}\beta w_{l_{7_5}}$, $p_{8_1}\beta w_{l_{8_1}}$, $p_{8_2}\beta w_{l_{8_2}}$, $p_{9_1}\beta w_{l_{9_1}}$, $p_{10_1}\beta w_{l_{10_1}}$.

W tabeli 3 został podany stan całej przestrzeni, mimo że z punktu widzenia reakcji aplikacji na zwiększenie poziomu naświetlenia (tzn. opuszczenie rolet) istotny jest jedynie jego fragment – kontekst.

Istnieje wiele opisowych definicji kontekstu (patrz rozdział III.4), jednakże aby przeprowadzić badania aplikacji go wykorzystujących należy przyjąć jego określoną matematyczną reprezentację. W niniejszej rozprawie kontekst będzie odnosił się do fragmentu przestrzeni (patrz definicja (4.1)), istotnego z punktu widzenia aplikacji – tzn. takiego, którego stan jest przez nią analizowany w celu dostosowywania się do zmian zachodzących w przestrzeni. Zbiór kontekstów można zdefiniować jako następująco:

$$K = \{k_1, k_2, \dots, k_{mK}\} \tag{4.4}$$

gdzie kontekst $k_l \in K$ jest definiowany jako piątka:

$$k_l = \langle OK_l, PK_l, \alpha_K, WK_l, \beta_K \rangle \tag{4.5}$$

gdzie $OK_l \subseteq O$ jest to podzbiór obiektów przestrzeni, w jakiej działa aplikacja (na przykład sala seminaryjna, osoby), które są istotne z jej punktu widzenia, $PK_l \subseteq P$ jest to zbiór parametrów (na przykład identyfikator) obiektów ze zbioru OK_l, których wartości są analizowane przez aplikację, natomiast relacja α_K jest określona jako $ok_{l_i}\alpha_K PK_{l_i}$ i reprezentuje przyporządkowanie podzbioru parametrów $PK_{l_i} \subseteq PK_l$ do obiektów $ok_{l_i} \in OK_l$. Definicja kontekstu zawiera także zbiór $WK_l \subseteq WK$ reprezentujący zbiór wartości parametrów obiektów wchodzących w skład k_l, oraz relację β_{SK} która jest określona jako $pk_{l_{i_j}}\beta_K wk_{l_{i_j}}$ i odzwierciedla przyporządkowanie wartości $wk_{l_{i_j}} \in WK_{l_i}$ (gdzie $WK_{l_i} \subseteq WK_l$) do parametrów $pk_{l_{i_j}} \in PK_{l_i}$ (przy czym $PK_{l_i} \subseteq PK_l$) obiektów $ok_{l_i} \in OK_l$ (zbiory OK_l i PK_l są zdefiniowane w k_l). Podobnie jak w przypadku definicji

przestrzeni (4.1) przyjęto założenie, że każdy obiekt musi mieć przypisany przynajmniej jeden parametr – tzn.:

$$\bigwedge_{ok_{l_i} \in OK_l} \bigvee_{PK_{l_i} \subset PK_l} ok_{l_i} \alpha_K PK_{l_i}$$

oraz dodatkowo, że każdy parametr obiektu ma przypisaną jedną wartość, czyli:

$$\bigwedge_{pk_{l_{i_j}} \in PK_{l_i}} \bigvee_{wk_{l_{i_j}} \in WK_{l_i}} pk_{l_{i_j}} \beta_K wk_{l_{i_j}}$$

Każdy parametr jest przypisany do przynajmniej jednego obiektu, dzięki czemu możliwy jest zapis:

$$PK_l = \bigcup_i PK_{l_i}$$

a także, iż każda wartość należąca do zbioru WK_l jest przypisana do przynajmniej jednego parametru, dzięki czemu można zapisać:

$$WK_l = \bigcup_i WK_{l_i}$$

Ponieważ obiekty należące do kontekstu mogą być zarówno fizyczne jak i niefizyczne taka definicja umożliwia określenie kontekstu odwołującego się do historycznych wartości parametrów obiektów. Istotne jest to, że kontekst nie musi zawierać wszystkich obiektów jakie zawiera przestrzeń. Dzięki temu można przy jego pomocy uzyskać różne stopnie szczegółowości opisu przestrzeni.

W czasie działania aplikacji IAI korzysta ona z trzech rodzajów kontekstu (patrz rozdział IV.1), które można pogrupować w zbiory: jednoelementowy zbiór kontekstów aplikacji (K_{IAI}), zbiór kontekstów wywołania akcji (K_{WA}) oraz zbiór kontekstów wykonania akcji K_A. Kontekst aplikacji $k_l \in K_{IAI}$ dla przykładu z rozdziału IV.1 został przedstawiony w tabeli 4.

Tabela 4 Przykład kontekstu aplikacji

Element definicji	Wartość
$OK_l = \begin{Bmatrix} ok_{l_1}, ok_{l_2}, ok_{l_3}, \\ ok_{l_4}, ok_{l_5} \end{Bmatrix}$	ok_{l_1}=sala seminaryjna, ok_{l_2}=osoba1, ok_{l_3}=osoba2, ok_{l_4}=system nagłośnienia.
$PK_l = PK_{l_1} \cup PK_{l_2} \cup PK_{l_3} \cup PK_{l_4} \cup PK_{l_5}$	PK_{l_1}= {$pk_{l_{1_1}}$= identyfikator, $pk_{l_{1_2}}$= poziom naświetlenia, $pk_{l_{1_3}}$= czy zajęcia, $pk_{l_{1_4}}$= lista osób w sali, $pk_{l_{1_5}}$= liczba osób w sali, $pk_{l_{1_6}}$= czy wysłano materiały dydaktyczne, $pk_{l_{1_7}}$= czy pora seminarium} $PK_{l_2} = PK_{l_3}$= {$pk_{l_{2_1}} = pk_{l_{3_1}}$= identyfikator, $pk_{l_{2_2}} = pk_{l_{3_2}}$= natężenie głosu, $pk_{l_{2_3}} = pk_{l_{3_3}}$= akceptowany przez urządzenie przenośne format dokumentów, $pk_{l_{2_4}} = pk_{l_{3_4}}$= adres email} PK_{l_4}= {$pk_{l_{4_1}}$= czy muzyka odgrywana}.
α_K	$\alpha_K(ok_{l_1}) = PK_{l_1}$, $\alpha_K(ok_{l_2}) = PK_{l_2}$, $\alpha_K(ok_{l_3}) = PK_{l_3}$, $\alpha_K(ok_{l_4}) = PK_{l_4}$.
$WK_l = WK_{l_1} \cup WK_{l_2} \cup WK_{l_3} \cup WK_{l_4}$	WK_1= {$wk_{l_{1_1}}$= Sala 101, $wk_{l_{1_2}}$= 30%, $wk_{l_{1_3}}$= zajęcia, $wk_{l_{1_4}}$= {osoba1, osoba2}, $wk_{l_{1_5}}$= 2, $wk_{l_{1_6}}$= tak, $wk_{l_{1_7}}$= tak} WK_2= {$wk_{l_{2_1}}$= Jan Kowalski, $wk_{l_{2_2}}$= 0%, $wk_{l_{2_3}}$= PPT dla Laptopa, $wk_{l_{2_4}}$= Jan.Kowalski@zak.eti.pg.gda.pl} WK_3= {$wk_{l_{3_1}}$= Adam Nowak, $wk_{l_{3_2}}$= 0%, $wk_{l_{3_3}}$= PPT dla PDA, $wk_{l_{3_4}}$= Adam.Nowak@zak.eti.pg.gda.pl} WK_4= {$wk_{l_{4_1}}$= muzyka wyłączona}.
β_K	$pk_{l_{1_1}}\beta_K wk_{l_{1_1}}$, $pk_{l_{1_2}}\beta_K wk_{l_{1_2}}$, $pk_{l_{1_3}}\beta_K wk_{l_{1_3}}$, $pk_{l_{1_4}}\beta_K wk_{l_{1_4}}$, $pk_{l_{1_5}}\beta_K wk_{l_{1_5}}$, $pk_{l_{1_6}}\beta_K wk_{l_{1_6}}$, $pk_{l_{1_7}}\beta_K wk_{l_{1_7}}$, $pk_{l_{2_1}}\beta_K wk_{l_{2_1}}$, $pk_{l_{2_2}}\beta_K wk_{l_{2_2}}$, $pk_{l_{2_3}}\beta_K wk_{l_{2_3}}$, $pk_{l_{2_4}}\beta_K wk_{l_{2_4}}$, $pk_{l_{3_1}}\beta_K wk_{l_{3_1}}$, $pk_{l_{3_2}}\beta_K wk_{l_{3_2}}$, $pk_{l_{3_3}}\beta_K wk_{l_{3_3}}$, $pk_{l_{3_4}}\beta_K wk_{l_{3_4}}$, $pk_{l_{4_1}}\beta_K wk_{l_{4_1}}$.

Mimo, iż w seminarium może brać udział nie więcej niż trzech uczestników (uproszczenie przyjęte dla zachowania przejrzystości przykładu), to kontekst aplikacji obejmuje 2 osoby. Jest to spowodowane tym, iż w czasie jej działania w sali znajdowały się dwie osoby. Kontekst wywołania akcji $k_l \in K_{WA}$ dla akcji wysłania wiadomości z linkami do materiałów dydaktycznych (oz_2) jest przedstawiony w tabeli 5.

Tabela 5 Przykład kontekstu wywołania akcji

Element definicji	Wartość
$OK_l = \begin{Bmatrix} ok_{l_1}, ok_{l_2}, \\ ok_{l_3} \end{Bmatrix}$	ok_{l_1}=sala seminaryjna, ok_{l_2}=osoba1, ok_{l_3}=osoba2.
$PK_l = PK_{l_1} \cup PK_{l_2} \cup PK_{l_3}$	PK_{l_1}= {$pk_{l_{1_1}}$= identyfikator, $pk_{l_{1_2}}$= czy zajęcia, $pk_{l_{1_3}}$= lista osób w sali, $pk_{l_{1_4}}$= czy wysłano materiały dydaktyczne} $PK_{l_2} = PK_{l_3}$= {$pk_{l_{2_1}} = pk_{l_{3_1}}$= identyfikator}.
α_K	$\alpha_K(ok_{l_1}) = PK_{l_1}$, $\alpha_K(ok_{l_2}) = PK_{l_2}$, $\alpha_K(ok_{l_3}) = PK_{l_3}$, $\alpha_K(ok_{l_4}) = PK_{l_4}$.
$WK_l = WK_{l_1} \cup WK_{l_2} \cup WK_{l_3}$	WK_1= {$wk_{l_{1_1}}$= Sala 101, $wk_{l_{1_2}}$= zajęcia, $wk_{l_{1_3}}$= {osoba1, osoba2}, $wk_{l_{1_4}}$= nie wysłano} WK_2= {$wk_{l_{2_1}}$= Jan Kowalski} WK_3= {$wk_{l_{3_1}}$= Adam Nowak}.
β_K	$pk_{l_{1_1}}\beta_K wk_{l_{1_1}}$, $pk_{l_{1_2}}\beta_K wk_{l_{1_2}}$, $pk_{l_{1_3}}\beta_K wk_{l_{1_3}}$, $pk_{l_{1_4}}\beta_K wk_{l_{1_4}}$, $pk_{l_{2_1}}\beta_K wk_{l_{2_1}}$, $pk_{l_{3_1}}\beta_K wk_{l_{3_1}}$.

Aby sprawdzić czy warunki oz_2 są spełnione muszą zostać przeanalizowane dane o sali seminaryjnej oraz o osobach uczestniczących w seminarium. Z kolei dla odpowiadającej im akcji (a_2) kontekst wykonania $k_l \in K_A$ wygląda następująco (patrz tabela 6):

Tabela 6 Przykład kontekstu wykonania akcji

Element definicji	Wartość
$OK_l = \begin{Bmatrix} ok_{l_1}, ok_{l_2}, \\ ok_{l_3} \end{Bmatrix}$	ok_{l_1}=sala seminaryjna, ok_{l_2}=osoba1, ok_{l_3}=osoba2.
$PK_l = PK_{l_1} \cup PK_{l_2} \cup PK_{l_3}$	PK_{l_1}= {$pk_{l_{1_1}}$= identyfikator, $pk_{l_{1_2}}$= lista osób w sali} $PK_{l_2} = PK_{l_3}$= {$pk_{l_{2_1}} = pk_{l_{3_1}}$= identyfikator, $pk_{l_{2_2}} = pk_{l_{3_2}}$= akceptowany przez urządzenie przenośne format dokumentów, $pk_{l_{2_3}} = pk_{l_{3_3}}$= adres email}.
α_K	$\alpha_K(ok_{l_1}) = PK_{l_1}$, $\alpha_K(ok_{l_2}) = PK_{l_2}$, $\alpha_K(ok_{l_3}) = PK_{l_3}$
$WK_l = WK_{l_1} \cup WK_{l_2} \cup WK_{l_3}$	WK_1= {$wk_{l_{1_1}}$= Sala 101, $wk_{l_{1_2}}$= {osoba1, osoba2}} WK_2= {$wk_{l_{2_1}}$= Jan Kowalski, $wk_{l_{2_2}}$= PPT dla Laptopa, $wk_{l_{2_3}}$= Jan.Kowalski@zak.eti.pg.gda.pl}

	WK_3= {$wk_{l_{3_1}}$= Adam Nowak, $wk_{l_{3_2}}$= PPT dla PDA, $wk_{l_{3_3}}$= Adam.Nowak@zak.eti.pg.gda.pl}.
β_{SK}	$pk_{l_{1_1}}\beta_{SK}wk_{l_{1_1}}$, $pk_{l_{1_2}}\beta_{SK}wk_{l_{1_2}}$, $pk_{l_{2_1}}\beta_{SK}wk_{l_{2_1}}$, $pk_{l_{2_2}}\beta_{SK}wk_{l_{2_2}}$, $pk_{l_{2_3}}\beta_{SK}wk_{l_{2_3}}$, $pk_{l_{3_1}}\beta_{SK}wk_{l_{3_1}}$, $pk_{l_{3_2}}\beta_{SK}wk_{l_{3_2}}$, $pk_{l_{3_3}}\beta_{SK}wk_{l_{3_3}}$.

Dla przytoczonych przykładów kontekst wywołania akcji jest różny od kontekstu wykonania akcji, ponieważ podczas uruchomienia akcji należy przekazać do niej inne dane kontekstowe niż potrzebne do rozpoznania czy akcję tę należy wywołać.

Podczas działania aplikacji stan przestrzeni może się zmieniać. Zmiany te wywołane są poprzez zdarzenia (patrz rysunek 12), które mogą mieć dwojaką naturę:

- zdarzenia wewnętrzne (należące do zbioru Z_A) - wywołane na skutek wykonania akcji,
- zdarzenia zewnętrzne (należące do zbioru Z_Z) – zdarzenia niekontrolowane przez aplikację, będące wynikiem działania czynników, na które aplikacja nie ma wpływu.

Zdarzenia rozumiane są jako matematyczna idealizacja zjawiska fizycznego polegająca na pominięciu jego rozmiarów przestrzennych i okresu trwania. Zbiór zdarzeń Z jest zdefiniowany następująco:

$$Z = Z_A \cup Z_Z \tag{4.6}$$

gdzie $Z_A = \{za_1, za_2, \dots, za_{mZA}\}$ to zbiór zdarzeń wewnętrznych związanych z akcjami aplikacji, natomiast $Z_Z = \{zz_0, zz_1, \dots, zz_{mZZ}\}$ jest to zbiór zdarzeń zewnętrznych. Ponieważ zdarzenia bezpośrednio przekładają się na zmiany stanu przestrzeni, można zatem zdefiniować dodatkowo funkcję zmiany stanu ζ:

$$\zeta: (SP, 2^Z) \rightarrow SP \tag{4.7}$$

gdzie SP to zbiór stanów przestrzeni (patrz definicja (4.2)). Nowy stan przestrzeni $sp' \in SP$ jest przy jej pomocy wyznaczany następująco:

$$sp' = \begin{cases} \zeta(sp, \{zz_1, zz_2, \dots, zz_c\})\ dla\ zz_i \in Z_Z, \mathrm{i} = 1\,..\,\mathrm{c} \\ \zeta(sp, \{za_1, \dots, za_d, zz_1, \dots, zz_c\})\ dla\ za_i \in Z_A, zz_j \in Z_Z, \mathrm{i} = 1..\,\mathrm{d}, \mathrm{j} = 1\,..\,\mathrm{c} \end{cases} \tag{4.8}$$

gdzie $sp \in SP$ to poprzedni stan przestrzeni. Przykładem jednoczesnego zajścia zdarzenia

zewnętrznego jak i wykonania akcji jest przejście między stanami aplikacji odpowiadającymi oczekiwanym warunkom wywołania akcji oz_4 i oz_5 w przykładzie z rozdziału IV.1. W momencie rozpoznania spełnienia warunków oz_4 została wykonana akcja, której wynikiem było włączenie muzyki w sali seminaryjnej. W czasie jej odgrywania osoby znajdujące się w sali zaczęły głośniej rozmawiać. W efekcie oba te zdarzenia (włączenie odgrywania muzyki i głośniejsze rozmowy) doprowadziły do spełnienia warunków oz_5. Przykładem sytuacji, gdy jedynie zdarzenie zewnętrzne powoduje zmianę stanu przestrzeni jest zwiększenie poziomu naświetlenia w sali seminaryjnej (patrz tabela 1 wiersz 4 odpowiadający warunkom oz_3). Zmiana ta została przedstawiona w tabeli 7.

Tabela 7 Przykład zmiany stanu kontekstu na skutek zajścia zdarzenia

Element definicji	Wartość
Z_A	{pojawienie się zwiększonego poziomu naświetlenia = 45%}.
Z_Z	{}.
sp	Przedstawiony w tabeli 3.
ζ	$\zeta(sp,$ {pojawienie się zwiększonego poziomu naświetlenia = 45%}) = sp'.
sp'	Zbliżony do stanu przedstawionego w tabeli 3, jednakże wartość parametru opisującego poziom naświetlenia wzrosła do 45%: β(sala seminaryjna, poziom naświetlenia) = 45%.

IV.4. Formalny opis zachowania się IAI

Sytuacja, na którą ma reagować aplikacja, jest wyrażona poprzez kontekst aplikacji $k_l \in K_{IAI}$ (patrz definicja kontekstu (4.5)). Odpowiadający mu fragment przestrzeni może dotyczyć wielu kontekstów wywołania akcji zgrupowanych w zbiór K_{WA}, zatem pojedyncza sytuacja w przestrzeni może powodować uruchomienie wielu akcji. W związku z tym, aby scharakteryzować aplikację IAI należy określić trójkę:

$$IAI = \langle OS, A, \gamma \rangle \tag{4.9}$$

gdzie zbiór $OS \supseteq (K_{WA} \cup \{\emptyset\})$ zawiera tzw. oczekiwane konteksty wywołania akcji (czyli wszystkie możliwe konteksty wywołania akcji, na których pojawienie się aplikacja ma reagować uruchomieniem akcji), $A = \{a_1, a_2, \dots, a_{mA}\}$ reprezentuje zbiór akcji aplikacji, natomiast funkcja γ definiowana jest jako:

$$\gamma: OS \to A \tag{4.10}$$

i oznacza przyporządkowanie kontekstu wywołania akcji do akcji jaka ma zostać wykonana w momencie jego rozpoznania. Każda akcja powoduje uruchomienie usługi znajdującej się w przestrzeni, zatem można zdefiniować relację odzwierciedlającą to przypisanie:

$$a_l o u_l \tag{4.11}$$

Wykonanie akcji może spowodować zaistnienie w przestrzeni zdarzenia wewnętrznego (patrz rozdział IV.3). Zbiór OS jest nadzbiorem K_{WA} ponieważ oprócz kontekstów wywołania akcji, które spowodowały uruchomienie akcji (w czasie działania aplikacji) zawiera wszystkie możliwe konteksty wywołania akcji. Interpretacja funkcji γ dla zbioru pustego odpowiada działaniu aplikacji niekontekstowej tzn. uruchomieniu akcji niezależnie od kontekstu. Istotne jest, iż podczas wykonywana akcji może dojść do zmiany oczekiwanych kontekstów wywołania akcji wraz z odpowiadającymi im akcjami (na przykład na skutek działań projektanta aplikacji – patrz rysunek 12, bądź wykonania akcji). Reprezentuje to bezpośrednią komunikację pomiędzy Kontrolerem i Widokiem w opisywanym w rozdziale IV.2 modelu MVC. Warto także zauważyć, że dzięki elastyczności definicji kontekstu (4.5) oczekiwane konteksty wywołania akcji mogą odnosić się do kontekstów historycznych. Alternatywną formą definicji aplikacji IAI jest przedstawienie jej w postaci zbioru dwu elementowych ciągów:

$$IAI = \{(os, a) : os \in OS, a \in A\} \tag{4.12}$$

gdzie os jest oczekiwanym kontekstem wywołania akcji, natomiast a oznacza akcję, jaka ma zostać uruchomiona, gdy zostanie on rozpoznany. W praktyce definicja takiej aplikacji nie zawiera oczekiwanych kontekstów wywołania akcji, lecz warunki odnoszące się do wartości parametrów obiektów wchodzących w ich skład. Jest to spowodowane tym, że parametry mogą przyjmować wartości ze zbiorów nieskończonych - na przykład poziom naświetlenia (wyrażany w procentach). Jeżeli aplikacja ma reagować, gdy przekroczy on 40% to zgodnie z definicją (4.12) należy wymienić wszystkie wartości poziomu naświetlenia powyżej tej granicy (w postaci osobnych oczekiwanych kontekstów wywołania akcji). W takich przypadkach definicja aplikacji musiałaby zawierać nieskończoną ilość par: oczekiwany kontekst wywołania akcji – akcja. Jednakże wszystkie te oczekiwane konteksty można zgrupować w pojedynczy warunek opisujący, iż aktualny (dla działającej aplikacji IAI) poziom naświetlenia ma być większy niż wartość progowa (40%). Dzięki temu spełnienie oczekiwanych warunków wywołania akcji oznacza rozpoznanie sytuacji w przestrzeni, na którą aplikacja ma zareagować. Grupowanie można wykonać wtedy, gdy oczekiwane konteksty wywołania akcji dotyczą tych samych

obiektów, posiadających te same parametry oraz mają przypisaną tą samą akcję. Warunki odnoszące się do wartości parametrów obiektów można zdefiniować jako zbiór:

$$WR = \{wr_1, wr_{2,} \dots, wr_{mWR}\} \tag{4.13}$$

gdzie każdy wr, określa kiedy warunek dotyczący wartości parametru obiektu jest prawdziwy. W celu powiązania warunku z parametrami obiektów należącymi do kontekstu wywołania akcji $k_l \in K_{WA}$ zdefiniowana została relacja ϱ:

$$\left(ok_{l_i}, pk_{l_j}\right) \varrho wr_{l_m} \tag{4.14}$$

gdzie $ok_{l_i} \in OK_l$ jest to zbiór obiektów kontekstu wywołania akcji, $pk_{l_j} \in PK_l$ jest to zbiór parametrów obiektów kontekstu wywołania akcji (patrz definicja (4.5)), $\left(ok_{l_i}, pk_{l_j}\right) \in OK_l \times PK_l$, natomiast $wr_{l_m} \in WR_l$ i $WR_l \subseteq WR$. Relacja grupowania λ jest zdefiniowana następująco:

$$OS_l \lambda oz_l \tag{4.15}$$

gdzie $OS_l \subseteq OS$ reprezentuje zbiór oczekiwanych kontekstów wywołania akcji jakie mogą być zgrupowane (tzn. przypisane do pojedynczego warunku przy pomocy funkcji ϱ), natomiast $oz_l \in OZ$ reprezentuje oczekiwany warunek wywołania akcji. Zbiór wszystkich oczekiwanych warunków wywołania akcji zdefiniowany jest następująco:

$$OZ = \{oz_1, oz_2, \dots, oz_{mOZ}\} \tag{4.16}$$

gdzie każde $oz_l \in OZ$ definiuje się jako czwórkę:

$$oz_l = \langle OK_l, PK_l, \alpha_K, WR_l, \varrho \rangle \tag{4.17}$$

w której OK_l oraz PK_l są obiektami i parametrami wchodzącymi w skład oczekiwanych kontekstów podlegających grupowaniu, α_K odzwierciedla przyporządkowanie parametrów do obiektów, $WR_l \subseteq WR$ reprezentuje zbiór warunków, natomiast relacja ϱ przyporządkowuje warunki do parametrów obiektów. Dzięki grupowaniu oczekiwanych kontekstów wywołania akcji można każdej takiej grupie przyporządkować akcję:

$$\sigma: OZ \rightarrow A \tag{4.18}$$

Zasada działania funkcji σ jest następująca:

$$\sigma(oz) = a <=> \bigwedge_{os \in OS_l \wedge \lambda(OS_l) = oz} \gamma(os) = a \quad (4.19)$$

co oznacza, że przyporządkowanie akcji do oczekiwanych warunków wywołania akcji jest równoważne przyporządkowaniu tej akcji do wszystkich zbiorów oczekiwanych kontekstów wywołania akcji należących do zbioru w skład którego wchodzą zbiory oczekiwanych kontekstów wywołania akcji, do którego przyporządkowane są oczekiwane warunki wywołania akcji. Dzięki temu aplikację IAI można przedstawić następująco:

$$IAI = \{(oz, a): oz \in OZ, a \in A\} \quad (4.20)$$

gdzie OZ to zbiór oczekiwanych warunków wywołania akcji, natomiast A jest to zbiór akcji. Przykład fragmentu aplikacji IAI przedstawionej w rozdziale IV.1 (odpowiadający parze $oz_2 - a_2$) dla tak określonej definicji aplikacji został przedstawiony w tabeli 8.

Tabela 8 Przykład definicji pary: warunki wywołania akcji – akcja aplikacji IAI

Element definicji	Wartość
IAI	$\{(oz_2, a_2)\}$.
$OK_2 = \{ok_{l_1}\}$	ok_{l_1}= sala seminaryjna.
$PK_2 = PK_{l_1}$	PK_{l_1}= {$pk_{l_{1_1}}$= identyfikator, $pk_{l_{1_2}}$= czy zajęcia, $pk_{l_{1_3}}$= lista osób w sali, $pk_{l_{1_4}}$= czy wysłano materiały dydaktyczne}.
α_K	$\alpha_K(ok_{l_1}) = PK_{l_1}$
$WR_2 = \begin{Bmatrix} wr_{21}, wr_{22}, wr_{23}, \\ wr_{24} \end{Bmatrix}$	wr_{21}(sala seminaryjna, identyfikator) = prawda jeżeli identyfikator sali seminaryjnej to Sala 101, wr_{22}(sala seminaryjna, lista osób w sali) = prawda jeżeli lista osób w sali seminaryjnej składa się z osób o identyfikatorach Adam Nowak, Jan Kowalski lub Krzysztof Wawrzyniak wr_{23}(sala seminaryjna, czy wysłano materiały dydaktyczne) = prawda jeżeli nie wysłano, wr_{24}(sala seminaryjna, czy zajęcia) = prawda jeżeli w sali seminaryjnej trwają zajęcia.
$oz_2 = \langle OK_2, PK_2, \alpha_K, WR_2, \varrho \rangle$	(sala seminaryjna, identyfikator) ϱ wr_{21}, (sala seminaryjna, lista osób w sali) ϱ wr_{22}, (sala seminaryjna, czy wysłano materiały dydaktyczne) ϱ wr_{23}, (sala seminaryjna, czy zajęcia) ϱ wr_{24}.
$A = \{a_2\}$	a_2: przedstawione w tabeli 1.

Dla każdej aplikacji IAI można zdefiniować wielkość kontekstu, który musi być analizowany, aby stwierdzić czy zostały spełnione oczekiwane warunki wywołania akcji. Wielkość ta będzie definiowana następująco:

$$\psi = \left|\{(ok_l, pk_l)\}: ok_{l_i} \in OK_l, pk_{l_i} \in PK_l, \bigvee_{wr_{l_i} \in WR_l} (ok_{l_i}, pk_{l_i}) \varrho wr_{l_i}\right| \qquad (4.21)$$

gdzie OK_l reprezentuje zbiór obiektów wchodzących w skład kontekstu aplikacji $k_l \in K_{IAI}$, PK_l reprezentuje zbiór parametrów obiektów kontekstu aplikacji natomiast WR_l jest to zbiór warunków wykorzystywanych w definicji oczekiwanych warunków wywołania akcji należący do definicji aplikacji IAI. Zatem ψ reprezentuje liczbę elementów zbioru wszystkich par: obiekt – parametr, dla których istnieje warunek wykorzystywany w definicji aplikacji IAI (dla przykładu z tabeli 1 $\psi = 9$). Innym parametrem jaki można określić dla aplikacji IAI jest miara pozwalająca określić stopień ich interaktywności. Ponieważ spełnienie oczekiwanego warunku wywołania akcji oznacza zaistnienie w przestrzeni aplikacji sytuacji, na którą musi ona zareagować a sposobem reakcji jest wywołanie akcji (ta interaktywność została opisana przy pomocy modelu MVC w rozdziale IV.2) można tę miarę zdefiniować jako liczba par: oczekiwane warunki wywołania akcji – akcja, znajdująca się w definicji aplikacji (4.20). Liczba ta jest równa liczbie oczekiwanych warunków wykorzystywanych w definicji aplikacji, dlatego też miara określająca stopień interaktywności aplikacji IAI jest to moc zbioru OZ z definicji aplikacji IAI:

$$\mu = |OZ| \qquad (4.22)$$

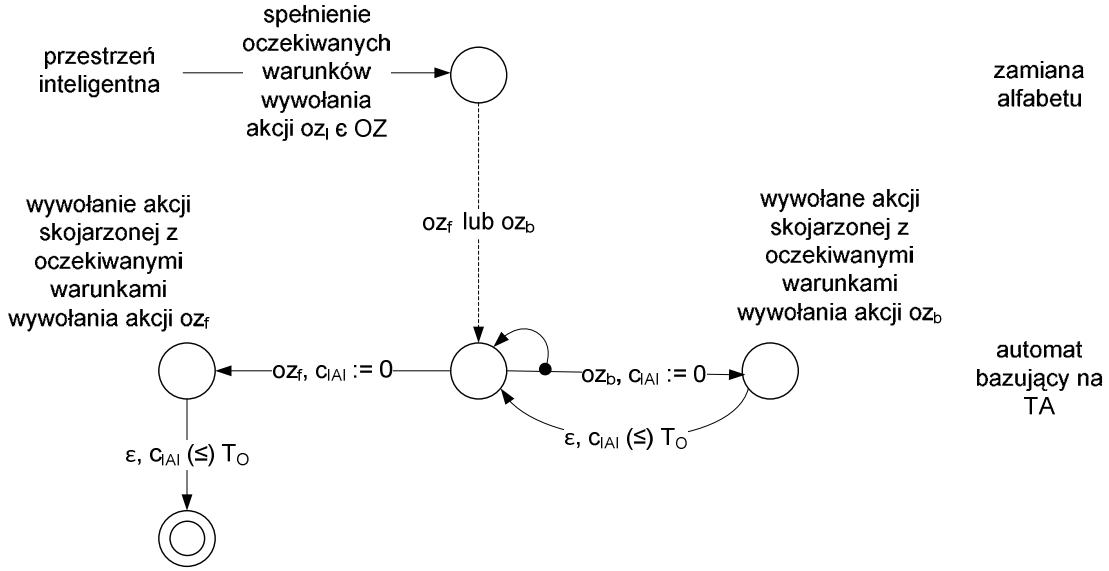
Aplikację IAI można również rozpatrywać w kategoriach klasycznej aplikacji iteracyjnej (zdefiniowanej w rozdziale III.1), gdyż jej definicję można przedstawić w następujący sposób:

$$IAI = \begin{Bmatrix} (oz_1, a_1), \\ (oz_2, a_2), \\ \dots \\ (oz_h, a_h) \end{Bmatrix} \qquad (4.23)$$

gdzie $oz_1, oz_2, \dots, oz_h \in OZ, a_1, a_2, \dots, a_h \in A$. Oczekiwane warunki wywołania akcji oz określają obiekty środowiska, na których operuje aplikacja IAI oraz mogą być interpretowane jako etykiety dla akcji (patrz graf z rys. 15). Podobnie zatem jak w klasycznym algorytmie iteracyjnym (AI) konstrukcja aplikacji IAI opiera się na etykietach (oczekiwanych warunkach), instrukcjach (zdarzeniach zewnętrznych i akcjach) oraz obiektach (wchodzących w skład kontekstu). Przejścia między kontekstami (stanami

przestrzeni) spełniającymi warunki zawarte w definicji aplikacji typu IAI (czyli także pośrednio przejścia pomiędzy tymi warunkami) opisane są poprzez funkcję ζ (patrz definicja (4.7)), co pozwala interpretować je jako odpowiednik zbioru instrukcji Q algorytmu iteracyjnego (patrz graf z rys. 15). Przejścia te są niedeterministyczne, ponieważ przestrzeń działania aplikacji może zawierać wiele kontekstów akcji, wynikających z definicji aplikacji w związku z czym w wyniku pojedynczej instrukcji aplikacja może przejść do wielu stanów (tzn. rozpoznać jednocześnie wiele oczekiwanych warunków przypisanych do różnych akcji). Dzieje się tak, dlatego że kontekst akcji jest jedynie fragmentem kontekstu aplikacji, tak że każdy z oczekiwanych warunków wywołania akcji opisanych w definicji aplikacji może operować na innym kontekście wywołania akcji. Aplikacja typu IAI ma zatem konstrukcję zgodną z klasycznym algorytmem iteracyjnym AI.

Ze względu na konieczność spełnienia przez aplikację IAI wymogów pracy w czasie rzeczywistym (w zakresie reagowania na oczekiwane konteksty) można ją zamodelować przy pomocy automatu bazującego na TA (opisanego w rozdziale III.2). Został on przedstawiony na rysunku 17.



Rysunek 17 Automat opisujący działanie aplikacji typu IAI

Aplikacja IAI reaguje na spełnienie oczekiwanych warunków wywołania akcji ($oz_l \in OZ$) dlatego też można przyjąć, iż tworzą one alfabet wejściowy automatu. Jednakże w praktyce oznacza to, iż taki alfabet jest nieskończony. Jest to spowodowane tym, iż podczas działania aplikacji mogą pojawić się nowe oczekiwane warunki wywołania akcji wprowadzone przez projektanta (będące skutkiem zmiany definicji aplikacji – patrz

rozdział IV.1), bądź będące wynikiem wykonania akcji. Aby móc skorzystać z automatu TA należy zatem zastosować podejście podobne do wykorzystywanego w automatach DA (opisanych w rozdziale III.2), polegające na zamianie alfabetu wejściowego. Polega ono na skojarzeniu z każdym z symboli nieskończonego alfabetu wejściowego jednego z symboli z alfabetu skończonego, dzięki czemu automat bazujący na TA pracuje na skończonym zbiorze symboli. W tym celu została zdefiniowana funkcja $\delta: OZ \rightarrow \Sigma^*$ która reprezentuje działanie transduktora, zamieniającego nieskończony alfabet wejściowy (oczekiwane warunki wywołania akcji) na skończony zbiór wartości $\Sigma^* = \{oz_b, oz_f\}$:

$$\delta(oz) = \begin{cases} oz_f, dla\ oz\ odpowiadającego\ warunkom\ zakończenia\ aplikacji \\ oz_b, dla\ oz\ nieodpowiadającego\ warunkom\ zakończenia\ aplikacji \end{cases}$$

Podejście to można zastosować, ponieważ z punktu widzenia sposobu działania aplikacji typu IAI (wywołanie akcji w momencie spełnienia oczekiwanych warunków wywołania akcji), konkretna reprezentacja kontekstu, nie musi znana. Najważniejsze jest to, że oczekiwane warunki zostały rozpoznane i skojarzona z nimi akcja została wywołana. W momencie pojawienia się na wejściu automatu bazującego na TA (patrz rysunek 17) symbolu oz_b, przechodzi on do stanu wywołania odpowiadającej mu akcji. Podczas tego przejścia zegar c_{IAI} jest zerowany. Ponieważ IAI są aplikacjami czasu rzeczywistego, wywołanie akcji musi nastąpić w pewnym przedziale czasu reprezentowanym przez stałą T_O (odzwierciedlającą ograniczenie czasowe), po upływie którego powinno nastąpić przejście automatu (bez odczytania symbolu z wejścia) do stanu oczekiwania na kolejny symbol (operator ($\leq$) sprawdza czy wartość zegara będącego lewym operandem jest mniejsza niż wartość przedziału czasu zawarta w prawym operandzie). Po wywołaniu akcji musi być ona wykonana w środowisku wykonywania aplikacji IAI. Dodatkowo, może się zdarzyć, że zostaną jednocześnie rozpoznane dwa oczekiwane warunki wywołania akcji – wtedy obie akcje z nimi skojarzone muszą być wykonane równolegle (na przykład zbyt wysoki poziom naświetlenia sali, powoduje opuszczenie rolet i w tym samym czasie seminarium może się zakończyć co powoduje rozpoczęcie odgrywania muzyki przez system nagłaśniający). Oznacza to konieczność równoległego analizowania i przetwarzania danych o stanie przestrzeni oraz wykonywania akcji. Zatem automat musi mieć możliwość dalszego odbierania symboli (oczekiwanych warunków wywołania akcji spełnionych przez sytuacje pojawiające się w przestrzeni) z wejścia (mimo wykonywania działania dla symbolu poprzedniego). Dlatego też w sposób równoległy przechodzi do stanu oczekiwania na te symbole. W przypadku przekazania symbolu oz_f zachowanie automatu jest podobne jak w przypadku symbolu oz_b, jednakże po wywołaniu akcji automat przechodzi do stanu końcowego (w którym już pozostaje) i dodatkowo w trakcie jej wykonywania nie oczekuje już na kolejne symbole wejściowe. Przyjęto założenie, iż

zamiana alfabetu dokonywana jest w czasie rzeczywistym – gdyż jedynie zamienia symbole alfabetu. Ponieważ aplikacja operuje na rozpoznanych oczekiwanych warunkach wywołania akcji i akcjach, oznacza to, iż automat ją opisujący powinien zawierać operacje doboru i wywołania akcji. Procesy rozpoznawania czy kontekst spełnia oczekiwane warunki oraz wykonania akcji są realizowane przez środowisko wykonywania aplikacji. Utworzony automat jest podstawą do zbudowania maszyny do wykonywania aplikacji IAI, zgodnie z którą takie środowisko jest zorganizowane. Została ona opisana w następnym rozdziale, gdzie określono jakie mechanizmy musi dostarczać oraz jakie ma charakterystyki czasowe dotyczące wykonania aplikacji IAI.

Wprowadzona w niniejszym rozdziale definicja aplikacji IAI umożliwia separację logiki aplikacji od mechanizmów analizowania kontekstu i uruchamiania oraz wykonywania akcji adaptacyjnych (usług w rozumieniu SOA). Podejście takie odpowiada Modelowi 2 zaprezentowanemu na rysunku 11 i pozwala na usystematyzowanie procesu projektowania oraz implementacji tego rodzaju aplikacji, którego punktem wyjścia jest aspekt kontekstowości. Same aplikacje IAI działają w czasie rzeczywistym, są równoległe i interaktywne, natomiast ich wykonanie posiada cechy iteracyjności i kontekstowości. Wszystkie te własności (wyszczególnione w rozdziale III) wynikają z zaproponowanej definicji. Aby można było wykonywać tak zdefiniowane aplikacje potrzebne jest dedykowane środowisko uruchomieniowe, którego działanie może być opisane przy pomocy dedykowanej maszyny. Podstawą do jej budowy jest wprowadzony w niniejszym rozdziale automat. Maszyna ta musi przyjmować definicję aplikacji i wykonywać wszystkie czynności, jakie są potrzebne do jej (aplikacji) prawidłowego wykonania (np. rozpoznawanie czy wystąpił jeden z oczekiwanych kontekstów). Budowa takiej maszyny i środowiska wykonywania aplikacji IAI dla przestrzeni inteligentnych została opisana w następnych rozdziałach.

V MODEL MASZYNY PIAI (PARALLEL IAI)

W rozdziale została opisana maszyna Parallel IAI (PIAI) służąca do przedstawienia zasad działania środowiska wykonywania aplikacji typu IAI. Jako podstawę do jej (maszyny) utworzenia wykorzystano automat wprowadzony w rozdziale IV. Przedstawiono także schemat funkcjonowania przestrzeni inteligentnej w kontekście realizacji podstawowych funkcji koniecznych do wykonania aplikacji IAI. Opisano również sposoby współpracy z użytkownikami przestrzeni. Podano właściwości opisujące maszynę PIAI przy uwzględnieniu rozproszenia logiki analizy kontekstu.

V.1. Maszyny PRAM i PIAI

Zgodnie z kierunkiem rozwoju przestrzeni inteligentnych, wykonywanie aplikacji IAI realizowane jest bezpośrednio przez przestrzeń (patrz rozdział II) poprzez zastosowanie podejścia zaprezentowanego na rysunku 11 – Model 2 (patrz rozdział III.5). Oznacza to konieczność przygotowania maszyny (pełniącej funkcję silnika z rysunku 11), dostarczającej mechanizmów (wynikających z automatu opisanego w rozdziale IV.4), dzięki której możliwe będzie wykonywanie aplikacji IAI. Maszyna taka została nazwana PIAI i wywodzi się z maszyny PRAM, omówionej w rozdziale III.2. Tak więc model IAI i maszyna PIAI stanowi podstawę do budowy środowiska wykonywania aplikacji IAI. Jeden z mechanizmów tego środowiska realizuje analizę danych kontekstowych. Następnie te dane mogą zostać dodatkowo przekształcone poprzez zagregowanie w bardziej abstrakcyjne struktury ułatwiające sprawdzenie czy oczekiwane warunki wywołania akcji zostały spełnione. Jeśli te warunki zachodzą środowisko wykonania powinno uruchomić skojarzoną z nimi (w definicji aplikacji) akcję. Podczas konstruowania maszyny przyjęto założenie, iż będzie ona operować jedynie na oczekiwanych warunkach wywołania akcji (patrz definicja (4.13) w rozdziale IV.4), ponieważ mogą być one zastosowane podczas implementacji, w przeciwieństwie do teoretycznego modelu aplikacji wykorzystującego oczekiwane konteksty wywołania akcji.

Punktem wyjścia do zbudowania maszyny służącej do wykonywania aplikacji IAI jest zbudowanie algorytmu opisującego ich działanie. Został on przedstawiony przy pomocy sekwencyjnego szablonu (zbudowanego zgodnie z regułami klasycznego algorytmu iteracyjnego omówionego w rozdziale III), utworzonego na bazie automatu opisującego działanie aplikacji IAI (patrz rozdział IV.4), w listingu 1.

```
1. if (akcja(null) != null) Thread(akcja(null));
2. do {
3.      k := kontekst();
```

```
4.      if ((k != null) and (sprawdz_oz1(k))) akcja(k);
5.      if ((k != null) and (sprawdz_oz2(k))) akcja(k);
6.          …
7.      if ((k != null) and (sprawdz_ozf(k))) akcja(k);
8. } while !(sprawdz_ozf(k))
```

Listing 1 Sekwencyjny szablon interaktywnej aplikacji kontekstowej

Sekwencyjność (w zakresie analizowania kontekstu) w zaprezentowanym na listingu 1 szablonie została wprowadzona w celu zobrazowania ogólnej zasady działania aplikacji IAI. Instrukcja `kontekst()` reprezentuje odczytanie z przestrzeni kontekstu aplikacji $k \in K_A$ (patrz definicja (4.5) w rozdziale IV.3), instrukcja `akcja(kontekst_aplikacji)` reprezentuje wykonanie przez aplikację akcji, która jest przypisana do oczekiwanych warunków wywołania akcji, natomiast instrukcja `Thread (akcja)` odpowiada równoległemu uruchomieniu wykonywania akcji. Obiekt `k` zawiera kontekst aplikacji (patrz definicja (4.5)) czyli odzwierciedla bieżącą sytuację w przestrzeni. Może on także zawierać informacje o historycznych kontekstach (w zależności od przyjętej definicji – patrz rozdział IV.3). Funkcje `sprawdz_oz()` realizują sprawdzenie czy kontekst spełnia oczekiwane warunki wywołania akcji i jeżeli tak zwracają wartość *true*. Operatory `!=` oraz `:=` oznaczają odpowiednio różność kontekstów oraz przypisanie odczytanego z przestrzeni kontekstu do obiektu, natomiast wartość `null` odpowiada pustemu kontekstowi. Z opisu aplikacji kontekstowych wyrażonego szablonem z listingu 1 wynika, iż składają się z dwóch procesów wykonywanych równolegle. Pierwszy z nich realizuje funkcjonalność (w postaci akcji) niezależną od kontekstu (tak jak w aplikacjach niekontekstowych) – reprezentowaną przez instrukcję `akcja(null)`. Jest ona odpowiednikiem akcji przypisanej do pustego zbioru oczekiwanych warunków wywołania akcji (patrz definicja (4.15)). W ramach drugiego procesu analizowane są dane kontekstowe dostarczane przez usługi opakowujące sensory. Wynikiem analizy jest określenie czy i które oczekiwane warunki wywołania akcji zostały spełnione, które następnie są interpretowane zgodnie z działaniem transduktora wchodzącego w skład automatu opisanego w rozdziale IV.4. Jeżeli warunki zostały spełnione następuje uruchomienie odpowiadającej im akcji dostosowującej działanie aplikacji do sytuacji w przestrzeni (co odpowiada części automatu bazującej na TA). Po rozpoznaniu oczekiwanych warunków wywołania akcji, które mają skutkować zakończeniem działania aplikacji (tzn. transduktor generuje symbol oz_f) maszyna PIAI uruchamia akcję i kończy działanie, natomiast w przeciwnym razie (transduktor generuje symbol oz_b) maszyna uruchamia akcję i rozpoczyna oczekiwanie na nowe dane kontekstowe. Aby zgrupować oczekiwane warunki i akcje w ramach pojedynczego

obiektu, szablon z listingu 1 można ulepszyć w następujący sposób (listing 2):

```
1. IAI = {(oz1, a1), (oz2, a2), …, (ozf, af)};
2. if (IAI.Keys.contains(null)) Thread (IAI.get(null));
3. do {
4.      k := kontekst();
5.      oz[] = sprawdz_warunki(k, IAI);
6.      for each (oz in oz[])
7.          if (oz != null)
8.              akcja(IAI.get(oz), k);
9. } while (!oz[].containsKey(ozf))
```

Listing 2 Zmodyfikowany szablon interaktywnej aplikacji kontekstowej

W listingu 2 została dodana funkcja `sprawdz_warunki()`, która na podstawie kontekstu aplikacji (zawartemu w obiekcie `k`) oraz zawartości obiektu `IAI` (opisującego sposób działania aplikacji) określa, które oczekiwane warunki wywołania akcji (`oz`) zostały spełnione. Są one reprezentowane przez tablicę `oz[]`. Następnie, dla każdego z nich zostaje uruchomiona odpowiadająca mu akcja i zostaje jej przekazany kontekst (obiekt `k`), zawierający odpowiedni dla akcji kontekst wykonania akcji (patrz rozdział IV.1). Różnicą w stosunku do poprzedniego szablonu (patrz listing 1) jest sprawdzanie czy oczekiwane warunki zostały spełnione w ramach pojedynczej funkcji (`sprawdz_warunki()`), której wykonanie nie musi być sekwencyjne. Oprócz danych kontekstowych aplikacje analizują także dane wejściowe – tak jak to zostało opisane w rozdziale IV.1 (na przykład przekazane przez człowieka bądź inne aplikacje). Jednakże te dane są przekazywane do aplikacji podczas wykonania akcji (patrz rysunek 12), dlatego też nie są bezpośrednio uwzględnione w powyższym opisie. Dla tak zorganizowanego szablonu można zauważyć, iż między różnymi aplikacjami kontekstowymi zmienia się jedynie zawartość obiektu `IAI` (czyli określenie oczekiwanych warunków i akcji), natomiast linia 2 wraz z pętlą reprezentowaną przez linie 3 – 9 pozostają niezmienne. W ten sposób zmodyfikowany szablon został podzielony na dwa elementy: zmienny opis sposobu działania aplikacji (obejmujący oczekiwane warunki wywołania akcji i akcje, jakie mają zostać wykonane w momencie ich spełnienia) oraz stałą część służącą do wykonania zawartych w tym opisie zasad działania. Część zmienna (obiekt `IAI`) definiuje w istocie aplikację IAI (patrz definicja (4.20)), natomiast część stałą można potraktować jako szablon maszyny o służącej wykonywaniu tego typu aplikacji, który może być następnie wzbogacony o aspekty równoległości.

Równoległość w działaniu maszyny PIAI musi przejawiać się w odbieraniu od usług opakowujących sensory (linia 4 z listingu 2) i analizowaniu (linia 5 w listingu 2) danych kontekstowych. Dane te mogą zostać przekazane (przez usługi) w dowolnym momencie czasu i analiza jednej z nich nie może wprowadzać opóźnienia w analizie pozostałych. Taki sposób organizacji może spowodować (jednoczesne) spełnienie różnych oczekiwanych warunków wywołania akcji zdefiniowanych w obiekcie `IAI`. Ponieważ do każdego warunku przypisana jest określona akcja to jej uruchamianie i wykonywanie także musi odbywać się w sposób równoległy. Oznacza to zmianę (w stosunku do listingu 2) dotyczącą linii 8, która polega na tym, iż maszyna jedynie uruchamia akcję i przekazuje ją do wykonania w ramach osobnej maszyny RAM, bądź PRAM (w zależności od postaci akcji). Zmianę taką (zrównoleglenie) można wprowadzić, ponieważ akcje są od siebie niezależne, a sam proces wykonywania aplikacji typu IAI nie musi wiedzieć o zakończeniu uruchomionej wcześniej akcji (patrz automat z rozdziału IV.4). Na rysunku 17, przedstawiającym automat opisujący zasadę działania aplikacji IAI, takie zachowanie zobrazowano poprzez natychmiastowy powrót do stanu oczekiwania na spełnienie oczekiwanych warunków zaraz po ich rozpoznaniu. Wprowadzenie równoległości jest zatem konieczne w odniesieniu do wszystkich mechanizmów zapewnianych przez maszynę PIAI i umożliwia wykonywanie oraz działanie aplikacji IAI w czasie rzeczywistym (patrz rozdział III.2). Z tego powodu maszyna PIAI powinna działać w sposób zorientowany na zdarzenia, gdzie każde zdarzenie odpowiada dostarczeniu przez usługę opakowującą sensor danej kontekstowej i powoduje jej przekazanie (w formie sygnału) do równolegle wykonywanego procesu analizy kontekstu. Zdarzeniem powinno być również rozpoznanie oczekiwanych warunków wywołania akcji, powodującego wygenerowanie sygnału skutkującego doborem przypisanej im akcji i jej uruchomieniem. Docelowy sposób realizacji równoległego działania maszyny PIAI został zaprezentowany na listingu 3.

```
1. //PRAM 1
2. IAI = {(oz1, a1), (oz2, a2), …, (ozf, af)};
3. PIAI_stop = false;
4. k = null;
5. if (IAI.Keys.contains(null))
6.       akcja_start(null, IAI);
7. sprawdz_warunki_thr = Thread(sprawdz_warunki);
8. kontekst_thr = Thread(kontekst);
9. while (!PIAI_stop) {
10.      //oczekiwanie na zmiany obiektu IAI
```

```
11. }
12.
13. //PRAM 2
14. void kontekstu (sensor_kontekst) {
15.      mutex_lock();
16.      k := k (+) sensor_kontekst;
17.      mutex_unlock();
18.      sprawdz_warunki_thr.start();
19. }
20.
21. //PRAM 3
22. void sprawdz_warunki () {
23.      //algorytm sprawdzania warunków
24.      //jeżeli     warunki     spełnione     uruchomienie
invoke_actions()
25. }
26.
27. //PRAM 4
28. void invoke_actions (oz[] ozs) {
29.      if (oz.containsKey(oz_f))
30.           PIAI_stop = true;
31.      for each (oz in ozs)
32.           akcja_start(IAI.get(oz), k);
33. }
```

Listing 3 Szablon maszyny PIAI

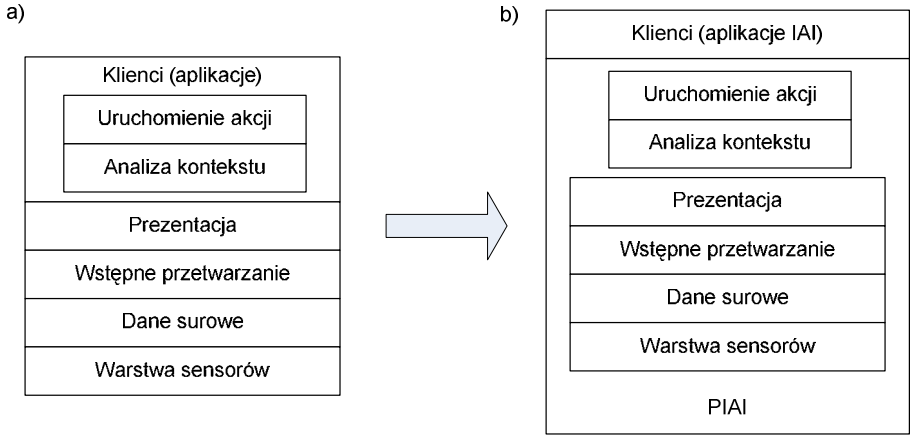
W ramach maszyny PIAI można wyodrębnić 4 maszyny PRAM. Pierwsza z nich (PRAM 1) kontroluje wykonanie (tzn. uruchamia poprzez instrukcję `Thread ()`) maszyn PRAM 2 (odbieranie danych kontekstowych od usług i uzupełnianie nimi kontekstu) i PRAM 3 (sprawdzanie czy kontekst aplikacji spełnia oczekiwane warunki wywołania akcji). Realizowane jest to (uruchamianie) odpowiednio przez instrukcję `sprawdz_warunki_thr = Thread(sprawdz_warunki)` dla PRAM 3 i `kontekst_thr = Thread(kontekst)` dla PRAM 2. Maszyna PRAM 1 przechowuje także obiekty przedstawione w liniach 2 - 4 (stanowiące jej pamięć współdzieloną) oraz zawiera pętlę, w ramach której maszyna oczekuje na zmiany definicji aplikacji IAI (czyli obiektu `IAI`) jakie mogą być dokonywane przez projektanta (patrz rozdział IV.1). Dane kontekstowe (zawarte w obiekcie `sensor_kontekst`) dostarczane

przez usługi opakowujące obiekty DIND pełniące rolę sensorów są odbierane w maszynie PRAM 2 w metodzie `sensor_kontekstu`. Uzupełnia ona nimi kontekst aplikacji (zawarty w obiekcie k poprzez wykonanie instrukcji `k := k (+) sensor_kontekst`), który jest umiejscowiony w pamięci współdzielonej. Następnie uruchamiane jest jego analizowanie w ramach maszyny PRAM 3 (poprzez wykonanie instrukcji `sprawdz_warunki_thr.start()`), która sprawdza czy zostały spełnione któreś z oczekiwanych warunków wywołania akcji, zawartych w definicji aplikacji – obiekcie `IAI`. Jest to realizowane w metodzie `sprawdz_warunki ()`. Zastosowany w tym celu algorytm analizy kontekstu (który może być także opisany maszyną PRAM) zależy bezpośrednio od jego reprezentacji (patrz definicje (4.5)). Jeżeli oczekiwane warunki zostały spełnione to następuje (w ramach maszyny PRAM 4) dobór, uruchomienie oraz wykonanie akcji (reprezentowane przez metodę `invoke_actions (oz[] ozs)`) i przekazanie im kontekstu aplikacji (zawierającego kontekst wykonania akcji). Komunikacja pomiędzy PRAM 3 i PRAM 4 odbywa się przy pomocy sygnałów. Dzięki opisanej organizacji maszyny PIAI w postaci maszyn PRAM jej działanie zostało zrównoleglone w zakresie odebrania danych kontekstowych, ich analizy oraz doboru oraz uruchomienia i wykonania akcji.

Ze względu na zrównoleglenie wykonania akcji, przedstawiony model maszyny przyjmuje postać zestawu maszyn PRAM typu CRCW (patrz rozdział III.2). Z punktu widzenia analizowania kontekstu, dla aplikacji typu IAI wykonywanych w ramach maszyny PIAI (a zatem także jej samej) pamięcią jest fragment przestrzeni inteligentnej (odpowiadający kontekstowi aplikacji) a środowisko wykonywania (zbudowane na bazie maszyny) operuje na tej pamięci za pomocą usług (patrz rysunek 12). Zatem jednoczesny odczyt danych z pamięci jest operacją bezkolizyjną, natomiast jednoczesny zapis jest typu arbitralnego (z wielu konkurencyjnych - jeden zapis kończy się powodzeniem, a pozostałe niepowodzeniem). Jednakże ze względu, na fakt iż akcje wpływają bezpośrednio na stan przestrzeni, jednoczesny zapis (czyli zmiana wartości parametrów obiektów przestrzeni inteligentnej) jest operacją mało prawdopodobną. Ponieważ akcje są niezależne oraz dane, na jakich operują, także mogą być niezależne (akcje operują na danych wejściowych od użytkowników oraz obiektach wchodzących w skład przestrzeni) to można zaklasyfikować architekturę środowiska wykonywania aplikacji typu IAI (zbudowanego na bazie maszyny z listingu 3) określić jako MIMD.

Aplikacja IAI wykonywana w ramach środowiska zbudowanego na bazie maszyny PIAI jest zatem odciążona od operacji związanych z akwizycją i analizą kontekstu. Zostało to przedstawione na rysunku 18, gdzie zobrazowano zmianę sposobu konstrukcji aplikacji IAI w stosunku do tradycyjnych aplikacji kontekstowych (których warstwowy model

został zaprezentowany w rozdziale III.4 na rysunku 10).

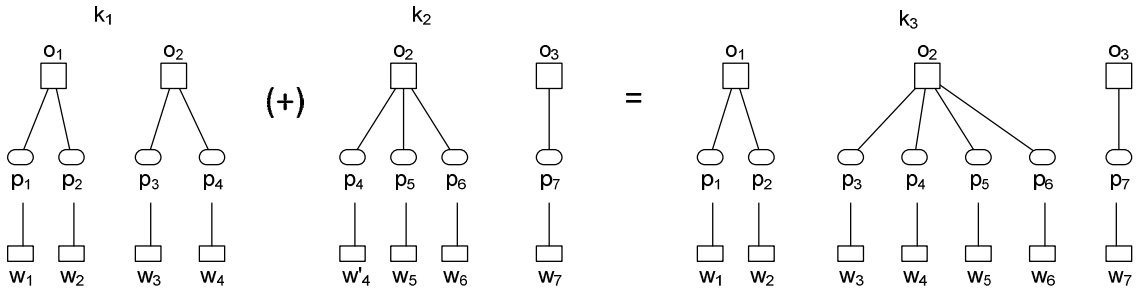


Rysunek 18 Sposoby konstrukcji aplikacji kontekstowej (a) i aplikacji IAI wykonywanej w ramach PIAI (b)

W przypadku tradycyjnych aplikacji kontekstowych (a) analiza kontekstu jak i wykonanie odpowiedniej akcji (czyli dostosowanie aplikacji do nowych warunków działania) było realizowane przez aplikację. Dodatkowe funkcje, takie jak komunikacja z sensorami oraz wstępne przetwarzanie uzyskanych od nich danych surowych, również mogły być realizowane przez aplikację bądź przez warstwę pośredniczącą (patrz rozdział II.2.4). W podejściu opartym o wykorzystanie maszyny PIAI (b), to maszyna zapewnia komunikację z usługami opakowującymi sensory, analizuje kontekst, uruchamia i wykonuje odpowiednią akcję. Definicja aplikacji typu IAI zawiera zatem opis zasad działania natomiast maszyna PIAI dostarcza odpowiednich mechanizmów ich interpretacji i realizacji. Tradycyjna aplikacja kontekstowa została w związku z tym podzielona na część opisującą zasady jej działania (odpowiadającą definicji aplikacji IAI) oraz część wykonującą (PIAI) - stanowi to nawiązanie do modelu 2 (z rysunku 11), w którym warstwę silnika do wykonywania aplikacji pełni maszyna PIAI. Powoduje to, że aplikację IAI można w opisać jako równoległą maszynę PIAI, w znaczeniu zdefiniowania zasad jej działania (par: oczekiwanych warunków wywołania akcji i akcji) oraz mechanizmów koniecznych do ich realizacji. Stanowi to wykazanie pierwszej tezy rozprawy doktorskiej. Poprzez umiejscowienie maszyny bezpośrednio w przestrzeni inteligentnej (w postaci środowiska wykonywania aplikacji IAI), wykonywanie aplikacji staje się jej integralną funkcją, co jest zgodne z trendami rozwojowymi przestrzeni inteligentnych (patrz rozdział II). Zatem sam fakt istnienia maszyny PIAI w przestrzeni powoduje, iż jest to przestrzeń inteligentna, natomiast o stopniu tej inteligencji decydują funkcjonujące w niej aplikacje IAI.

V.2. Schemat działania PIAI

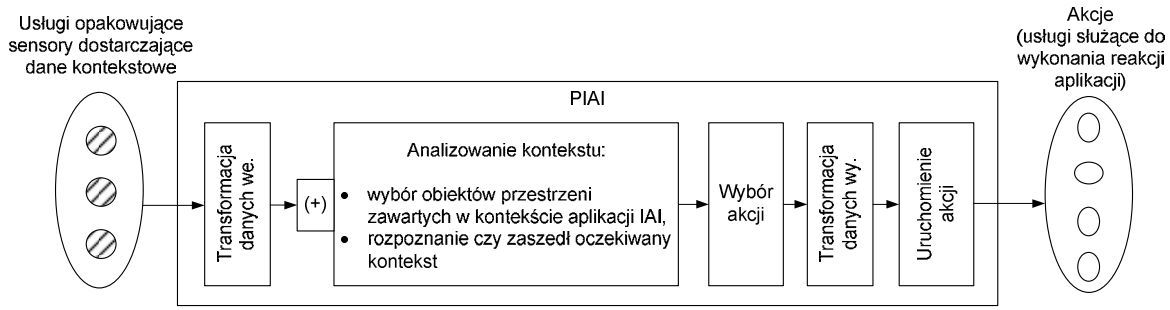
Maszyna PIAI uzyskuje dane kontekstowe od usług opakowujących sensory. Działanie każdego z nich dotyczy pewnego podzbioru obiektów wchodzących w skład kontekstu aplikacji, natomiast PIAI musi analizować pełen konteksta aplikacji (oczekiwane warunki wywołania akcji mogą dotyczyć kontekstów opisujących większy fragment przestrzeni niż obserwowany przez poszczególne sensory). Zatem można zdefiniować działanie dodawania kontekstów $(+)$, którego argumentami i wynikiem jest kontekst $((+): K \times K \rightarrow K)$. Zasada dodawania kontekstów została przedstawiona na rysunku 19.



Rysunek 19 Zasada dodawania kontekstów

Rysunek 19 przedstawia trzy konteksty $(k_1, k_2, k_3 \in K)$. Dwa z nich (k_1, k_2) są dostarczane (w postaci danych kontekstowych) przez dwie różne usługi, opakowujące obiekty DIND pełniące rolę sensorów, współpracujące z PIAI. Kontekst k_3 jest kontekstem wynikowym. Dla uproszczenia rozważań dla zobrazowania kontekstów nie posłużono się notacją z definicji (4.5), lecz wykorzystano zbiory O, P i W (opierając się na założeniu, że kontekst odpowiada fragmentowi stanu przestrzeni). Konteksty mogą mieć część wspólną – np. zarówno jeden jak i drugi sensor obserwuje ten sam obiekt o_2 i jego parametr p_4. Należy zwrócić uwagę na to, że pozostałe parametry obiektu o_2 są dla obu sensorów różne. Dodawanie kontekstów polega na utworzeniu nowego kontekstu, którego zbiory obiektów, parametrów i wartości są odpowiednio sumą zbioru obiektów, sumą zbioru parametrów i sumą zbioru wartości dodawanych kontekstów. Relacja przyporządkowania parametrów do obiektów jest określona dla nowych zbiorów analogicznie jak dla dodawanych kontekstów. Podczas wykonywania operacji dodawania kontekstów może się zdarzyć, wartości przypisane do tych samych parametrów (w przykładzie w_4 i w_4') są różne (na przykład odczyt poziomu naświetlenia dostarczany przez dwa różne sensory będzie różnił się o 1%) - patrz rozdział II.2.2. W takim wypadku zachowanie operacji zależy od decyzji podjętej podczas implementacji środowiska wykonywania opartego na maszynie PIAI (dotyczy to relacji przypisania wartości do parametrów).

Usługi działające w przestrzeni inteligentnej (patrz rozdział IV) mogą posiadać różną semantykę (znaczenie danych jakie wymieniają z PIAI) oraz syntaktykę (inny sposób wymiany danych – patrz rozdział III.3). Dlatego też, w przypadku współpracy PIAI z heterogenicznymi usługami konieczne jest wprowadzenie komponentów, umożliwiających odpowiednie dostosowanie (transformację) danych i sposobu ich przekazywania. Musi być ono wykonywane w momencie, gdy dane są dostarczane do PIAI (tzn. do środowiska wykonywania aplikacji IAI na niej zbudowanego) do ujednoliconej postaci pozwalającej na ich wspólnie analizowane (niezależnie od rodzaju sensora) pod kątem sprawdzenia czy spełniają oczekiwane warunki wywołania akcji. Podobnie podczas uruchamiania akcji, dane jakie mają być do niej przekazane muszą być odpowiednio przygotowane (przetransformowane z postaci ujednoliconej do zrozumiałej przez usługę). Ostatecznie schemat działania maszyny PIAI wygląda następująco (rysunek 20):



Rysunek 20 Proces analizy kontekstu w maszynie PIAI

Listing 4 opisuje działanie maszyny PIAI rozszerzone o aspekty transformacji danych.

```
1. //PRAM 1
2. IAI = {(oz1, a1), (oz2, a2), …, (ozk, ak)};
3. PIAI_stop = false;
4. k = null;
5. if (IAI.Keys.contains(null))
6.       akcja_start(null, IAI);
7. sprawdz_warunki_thr = Thread(sprawdz_warunki);
8. kontekst_thr = Thread(kontekst);
9. while (!PIAI_stop) {
10.      //oczekiwanie na zmiany obiektu IAI
11. }
12.
13. //PRAM 2
14. void kontekstu (sensor_kontekst) {
```

```
15.      mutex_lock();
16.      k = k (+) transformujPIAI(sensor_kontekst);
17.      mutex_unlock();
18.      sprawdz_warunki_thr.start();
19. }
20.
21. //PRAM 3
22. void sprawdz_warunki () {
23.      //algorytm sprawdzania warunków
24.      //jeżeli     warunki     spełnione     uruchomienie
invoke_actions()
25. }
26.
27. //PRAM 4
28. void invoke_actions (oz[] ozs) {
29.      if (ozos.containsKey(oz_k))
30.           PIAI_stop = true;
31.      for each (oz in ozs)
32.           akcja_start(IAI.get(oz), transformujA(k));
33. }
```

Listing 4 Szablon maszyny PIAI rozszerzony o transformacje danych

Funkcje transformujące dane są to `transformujPIAI()` oraz `transformujA()` i oznaczają odpowiednio transformację danych kontekstowych do postaci zrozumiałej przez proces analizy kontekstu w PIAI oraz transformację danych analizowanych w PIAI do postaci zrozumiałej przez usługę wykonywaną w ramach akcji. W automacie opisującym działanie aplikacji IAI (patrz rozdział IV.4) przejawiają się one w postaci mechanizmu przekazującego symbole słownika wejściowego do transduktora i stanów uruchamiania akcji.

Ze względu na charakter współpracy usług opakowujących sensory z maszyną PIAI (tzn. ze środowiskiem wykonywania zbudowanym na bazie tej maszyny) można je podzielić na dwie kategorie: synchroniczne i asynchroniczne. Praca w trybie synchronicznym oznacza dostarczenie do procesu analizowania kontekstu określonego (specyficznego dla sensora) zbioru parametrów obiektów (i ich wartości) przestrzeni „na żądanie”. Ten sposób pracy oznacza, iż mechanizm analizowania kontekstu odpytuje usługę o dane kontekstowe i to on decyduje, w którym momencie powinien nastąpić ich odczyt z przestrzeni (przez sensor). Taki model komunikacji to *request*/*response*, który wymaga rejestracji usługi w środowisku wykonywania aplikacji zbudowanym na bazie

PIAI. Model ten wykorzystywany jest również w przypadku usług opakowujących obiekty DIND pełniące rolę aktuatorów oraz innych usług w znaczeniu SOA, z którymi zachodzi komunikacja w czasie wykonywania aplikacji IAI. Przy jego wykorzystaniu reakcja aplikacji nie zależy od faktycznego momentu czasu pojawienia się zdarzenia w przestrzeni. Pomiędzy odpytywaniami może nastąpić wiele zdarzeń, powodujących zmiany w kontekście aplikacji. Jednakże spowodowany przez nie stan przestrzeni nie zostanie dostarczony do PIAI, ponieważ ulegnie zmianie na skutek zajścia kolejnych zdarzeń. Z kolei usługi, działające w trybie asynchronicznym, same decydują o tym kiedy przekazać wartość parametru obiektu kontekstu do procesu analizowania kontekstu (co bardziej wpisuje się w nurt systemów rozproszonych). W przypadku usług asynchronicznych, można dodatkowo wprowadzić podział na takie, które wysyłają informacje o zmianie stanu przestrzeni – komunikacja z aplikacją na zasadzie *publish/subscribe* (usługa zostaje zarejestrowana w środowisku wykonywania i w momencie zmiany kontekstu wysyła odpowiedni komunikat do procesu analizowania kontekstu w PIAI) oraz takie, które ją regularnie (w jednakowych odstępach czasu) odświeżają – komunikacja z aplikacją także na zasadzie *publish/subscribe*, jednakże ze stałym interwałem. Podejście oparte na odświeżaniu wydaje się mniej dopasowane do aplikacji kontekstowych, gdyż sensory są narażone na utratę łączności z systemem, awarię, zanik zasilania itp. W takich przypadkach mogło by się okazać, iż usługa opakowująca obiekt DIND pełniący rolę sensora nie zdąży zaktualizować przekazywanej przez siebie danej i wykorzystująca ją aplikacja IAI będzie w efekcie działała na nieaktualnym kontekście. Za każdym razem podczas odczytywania danej z otoczenia (wartości parametru obiektu) usługa określa jaki jest czas trwania jej ważności (patrz rozdział IV.1) i przekazuje ją do PIAI (środowiska wykonania zbudowanego na jej podstawie). Na przykład dla usług odświeżających wartości parametrów obiektów przestrzeni czas ten powinien może być równy interwałowi odświeżania.

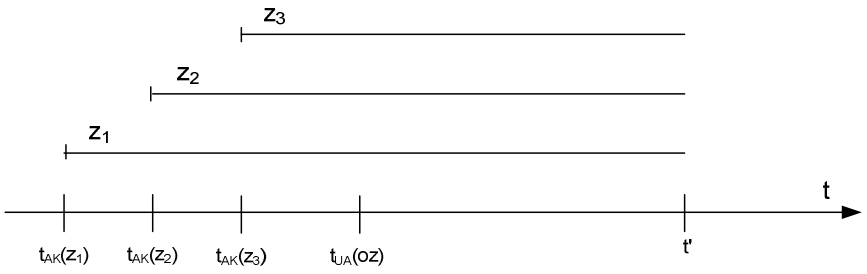
V.3. Własności PIAI

V.3.1. Elementy procesu analizowania danych

W odniesieniu do funkcjonalności zawartych w maszynie PIAI (transformacji i analizy danych kontekstowych, ich przetwarzania oraz wywoływania akcji) można dokonać jakościowej (w znaczeniu efektywności – patrz rozdział I.2) analizy procesu wykonywania każdej aplikacji typu IAI. Zgodnie ze schematem działania PIAI z listingu 4, można wyróżnić następujące instrukcje (które występują po sobie sekwencyjnie) wykonywane w ramach PIAI związane z analizowaniem danych kontekstowych oraz wywołaniem akcji (dla każdej z nich podana została odpowiadająca funkcja z listingu 4):

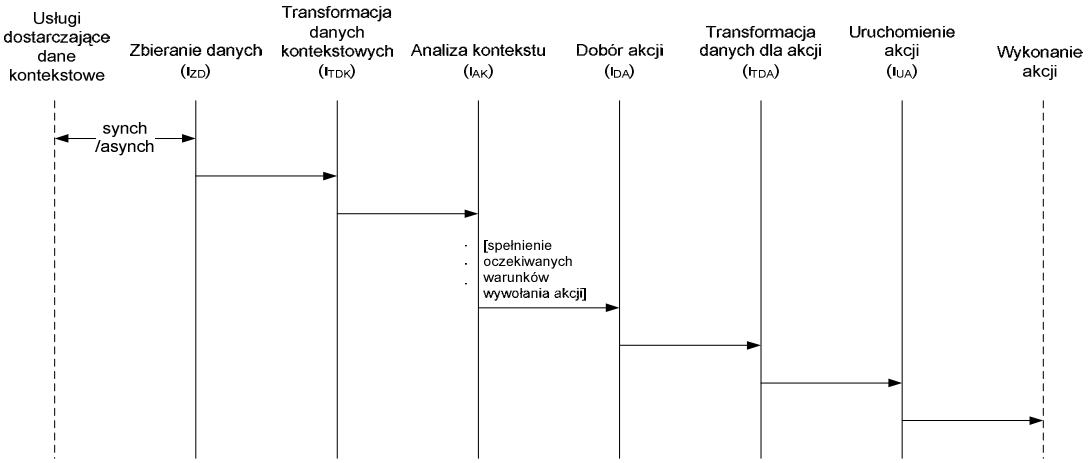
- ι_{ZD} - zbieranie danych kontekstowych - `kontekst()`,
- ι_{TDK} – transformacja danych kontekstowych – `transformujPIAI()`,
- ι_{AK} – analiza kontekstu (sprawdzenie czy zostały spełnione oczekiwane warunki wywołania akcji) - zawiera wykonanie funkcji $(+)$ - `sprawdz_warunki()`,
- ι_{DA} – dobór (wyszukanie) akcji (zdefiniowanej dla oczekiwanych warunków wywołania akcji) – pętla `for each` z funkcją `IAI.get()`,
- ι_{TDA} – dostosowanie danych do wybranej akcji – `transformujA()`,
- ι_{UA} – uruchomienie akcji - `akcja_start()`.

Zbieranie danych (ι_{ZD}) odpowiada pobraniu danych kontekstowych od usług i różni się w zależności od sposobu komunikacji usługi ze środowiskiem wykonywania aplikacji IAI (patrz synchroniczna i asynchroniczna komunikacja z usługami opisana w rozdziale V.2). Instrukcja ta ma znaczenie w przypadku komunikacji synchronicznej. W przypadku komunikacji asynchronicznej nie wprowadza ona żadnych opóźnień. Następnie wykonywana jest transformacja przekazanych danych kontekstowych (ι_{TDK}) i po ich uwzględnieniu w kontekście, jego analiza (ι_{AK}). Instrukcje te (ι_{ZD}, ι_{TDK} i ι_{AK}) są związane z danymi kontekstowymi. Jeżeli wynikiem analizy kontekstu jest rozpoznanie spełnienia oczekiwanych warunków wywołania akcji, następuje jej wyszukanie - ι_{DA}. Instrukcja ta jest związana rozpoznanymi warunkami, podobnie jak dwie kolejne (ι_{TDA} i ι_{UA}). Z punktu widzenia działania aplikacji IAI wykonywanej w PIAI (od momentu pojawienia się zdarzenia w środowisku, do zakończenia wykonywania akcji) można także zdefiniować instrukcje odpowiadające pobraniu (z przestrzeni) przez sensor wartości parametrów obiektów i wykonaniu akcji. Jednakże nie dotyczą one bezpośrednio maszyny PIAI (patrz rysunek 20) dlatego też instrukcje te nie są ujęte w jej analizie. Dodatkowo, należy zwrócić uwagę na to, że kategorie ι_{DA}, ι_{TDA} i ι_{UA} dotyczą oczekiwanych warunków wywołania akcji a nie danych kontekstowych. Z tego powodu przekazanie danych nie zawsze skutkuje uruchomieniem wszystkich instrukcji PIAI. Następuje to dopiero w momencie gdy PIAI otrzyma dane kontekstowe kreujące kontekst aplikacji powodujący spełnienie warunków. Zostało to zobrazowane na rysunku 21.



Rysunek 21 Zdarzenia powodujące wystąpienie kontekstu a czas uruchomienia akcji

Do maszyny PIAI przekazywane są dane kontekstowe związane z pojawieniem się w przestrzeni działania aplikacji IAI zdarzeń z_1, z_2, z_3 (patrz definicja (4.6) z rozdziału IV.3). Jak zostało to opisane w rozdziale IV.1 rozpoczynają one działanie aplikacji. Na podstawie tych danych tworzony jest kontekst aplikacji, który powoduje spełnienie oczekiwanych warunków wywołania akcji oz. Dla przejrzystości rozważań przyjęto, iż czas rozpoczęcia wykonywania instrukcji ι_{AK} odnosi się do zdarzeń, jednakże w rzeczywistości dotyczy danych kontekstowych, jakie zostały dostarczone do PIAI na skutek ich (zdarzeń) zaistnienia w przestrzeni. Dodatkowo założono, iż rozpoczęcie analizy kontekstu rozpoczyna się w tym samym momencie czasu co pojawienie się zdarzenia w otoczeniu. Na rysunku 21 zostały zaznaczone czasy rozpoczęcia tej analizy – są to wartości funkcji $t_{AK}(z)$ dla $z \in \{z_1, z_2, z_3\}$. Mimo że zdarzenie z_1 pojawiło się w przestrzeni wcześniej niż z_3, to z punktu widzenia spełnienia oczekiwanych warunków wywołania akcji jego analizowanie musi zostać wstrzymane do czasu pojawienia się z_3, jako zdarzenia powodującego (wraz z z_1 i z_2) zaistnienie kontekstu je spełniającego. Odpowiadająca im akcja jest wykonywana dopiero od momentu $t_{UA}(oz)$ i kończona w t'. W między czasie mogą być równolegle analizowane inne warunki wywołania akcji oraz uruchamiane inne akcje (patrz rozdział V.1). Diagram sekwencji obrazujący wykonywane w ramach maszyny PIAI instrukcje, które są związane z przekazaniem do niej danej kontekstowej został przedstawiony na rysunku 22.



Rysunek 22 Diagram sekwencji dla instrukcji wykonywanych w PIAI

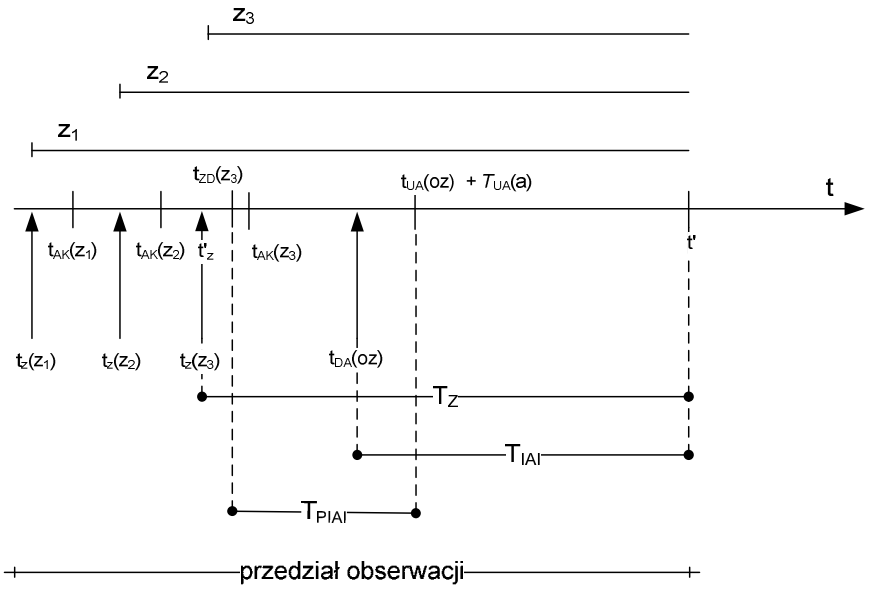
Na bazie powyższego diagramu można określić całkowity czas wykonania instrukcji maszyny PIAI zawiązany z dostarczeniem danej kontekstowej, sprawdzeniem czy kontekst aplikacji spełnia oczekiwane warunki wywołania akcji i uruchomieniem przypisanej im akcji (w definicji aplikacji IAI):

$$T_S = T_{ZD} + T_{TDK} + T_{AK} + T_{DA} + T_{TDA} + T_{UA} \tag{5.1}$$

gdzie T_{ZD}, T_{TDK}, T_{AK}, T_{DA}, T_{TDA} oraz T_{UA} oznaczają czasy wykonania odpowiadające poszczególnym instrukcjom. W zależności od przyjętego sposobu komunikacji środowiska wykonywania aplikacji IAI zbudowanego na bazie maszyny PIAI z usługami dostarczającymi dane czas T_{ZD} będzie przyjmował różne wartości (np. dla komunikacji synchronicznej musi uwzględniać maksymalny interwał odpytywania). Ma to bezpośredni wpływ na utrzymanie charakteru wykonania aplikacji IAI w czasie rzeczywistym (patrz rozdział III.2) ponieważ, gdy interwały odpytywania usługi są zbyt długie, wykonanie aplikacji typu IAI nie będzie spełniało wymagań czasowych.

V.3.2. Czas reakcji a ważność danych

Określenie czasu reakcji może odnosić się do przedziału czasu między pojawieniem się zdarzenia w przestrzeni do reakcji na nie (T_Z), do przedziału czasu działania maszyny PIAI (T_{PIAI}), oraz do przedziału czasu działania samej aplikacji typu IAI (T_{IAI}). Są to trzy różne perspektywy czasu reakcji, które obejmują różne aspekty wykonania aplikacji typu IAI w środowisku zbudowanym na bazie maszyny PIAI. Ich interpretacja, dla definicji aplikacji IAI składającej się z pojedynczej pary: oczekiwane warunki wywołania akcji (oz) – akcja (a) (patrz definicja (4.20)), została przedstawiona na rysunku 23:



Rysunek 23 Czasy reakcji w odniesieniu do wykonania aplikacji typu IAI

W przykładzie zaprezentowanym na rysunku 23 założono, że kolejność pojawiania się zdarzeń w przestrzeni inteligentnej nie jest istotna z punktu widzenia aplikacji IAI. Podobnie jak na rysunku 21 dane związane z pojedynczym zdarzeniem nie powodują spełnienia oczekiwanych warunków wywołania akcji. Dopiero kontekst będący wynikiem zajścia wszystkich trzech zdarzeń powoduje ich spełnienie. Zostały dodatkowo wprowadzone funkcje oznaczające przypisanie do zdarzenia momentu czasu jego pojawienia się w przestrzeni ($t_Z(z)$) i momentu czasu rozpoczęcia wykonywania instrukcji zbierania danych (ι_{ZD}) przez PIAI związanych z jego zajściem ($t_{ZD}(z)$). Momenty czasu t'_Z i t'_{ZD} oznaczają odpowiednio moment pojawienia się w przestrzeni inteligentnej zdarzenia, które powoduje zaistnienie kontekstu spełniającego oczekiwane warunki wywołania akcji oz (w przykładzie z rysunku 23 wartość $t'_Z = t_Z(z_3)$) oraz moment rozpoczęcia wykonywania instrukcji zbierania danych kontekstowych ι_{ZD} w ramach maszyny PIAI dla tego zdarzenia, czyli wartości parametrów obiektów przestrzeni jakie zostały zmienione w wyniku jego zajścia (w przykładzie z rysunku 23 wartość $t'_{ZD} = t_{ZD}(z_3)$). Po spełnieniu oczekiwanych warunków wywołania akcji oz następuje rozpoczęcie dobierania akcji wynikającej z definicji aplikacji IAI. Czas ten przypisywany jest warunkom oz przy pomocy funkcji $t_{DA}(oz)$, natomiast funkcja $t_{UA}(oz)$ przypisuje im czas rozpoczęcia uruchamiania wybranej akcji. Z kolei funkcja $\tau_{UA}(a)$ przypisuje akcji całkowity przedział czasu jej uruchamiania (tzn. przedział czasu od początku do końca uruchamiania akcji). Funkcja ta pozwala wyznaczyć wartość T_{UA} dla określonej akcji. Zaznaczony na rysunku 23 czas t' odpowiada momentowi zakończenia wykonywania akcji. Czas reakcji T_Z

(określany jako rzeczywisty czas reakcji) odnosi się do działania aplikacji typu IAI z perspektywy użytkownika bądź innej współpracującej aplikacji (użytkownika przestrzeni) i jest definiowana następująco:

$$T_Z = t' - t'_Z \tag{5.2}$$

Obejmuje on przedział czasu od momentu pojawienia się zdarzenia w przestrzeni, które skutkuje zaistnieniem kontekstu spełniającego oczekiwane warunki wywołania akcji (t'_Z), do momentu kiedy zostanie wykonana odpowiednia akcja (t'). W podanym wzorze składnik t'_Z można pominąć, jeżeli pomiar czasu rozpoczyna się w momencie zajścia sytuacji w przestrzeni. Maszyna PIAI, nie ma wpływu na szybkość, z jaką sensor wykrywa zdarzenia w przestrzeni i kiedy przekazuje dane kontekstowe z nim związane (patrz rozdział V.2). Co więcej PIAI jedynie uruchamia akcję, natomiast nie ma wpływu na jej przebieg (czyli faktyczną zmianę zachowania aplikacji IAI – np. odpowiednią interakcję z człowiekiem, bądź z inną aplikacją). Zatem z punktu widzenia maszyny PIAI czas jej reakcji zdefiniuje się w następujący sposób:

$$T_{PIAI} = t_{UA}(oz) + T_{UA} - t'_{ZD} \tag{5.3}$$

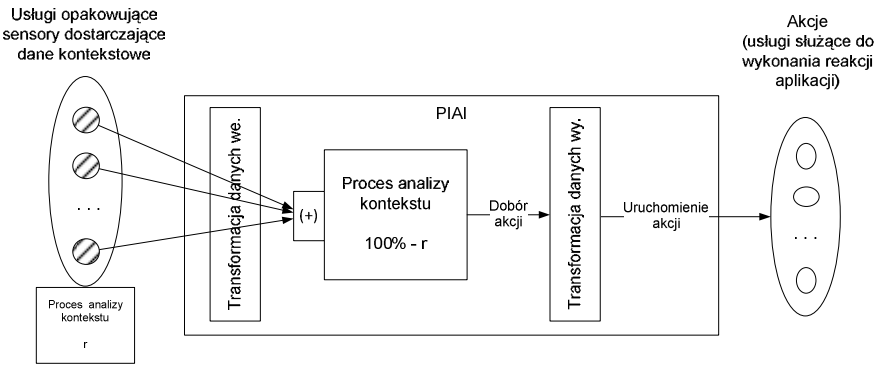
Jest on określony w odniesieniu do przedziału czasu od momentu otrzymania danych kontekstowych (ściśle, rozpoczęcia wykonywania instrukcji zbierania danych w PIAI), związanych z pojawieniem się zdarzenia powodującego zaistnienie kontekstu, spełniającego oczekiwane warunki wywołania akcji (t'_{ZD}) do momentu zakończenia uruchomiania związanej z tym kontekstem akcji ($t_{UA}(oz) + T_{UA}$). Trzecim wyszczególnionym rodzajem czasu reakcji jest czas reakcji dotyczący aplikacji typu IAI wykonywanej w ramach maszyny PIAI - T_{IAI}, który definiowany jest następująco:

$$T_{IAI} = t' - t_{DA}(oz) \tag{5.4}$$

Aplikacja nie rozpoznaje bezpośrednio kontekstu – wykonuje to maszyna. Czas reakcji aplikacji można zatem liczyć od momentu wykrycia przez maszynę oczekiwanego kontekstu wywołania akcji (co pokrywa się z czasem rozpoczęcia doboru akcji - $t_{DA}(oz)$) do momentu zakończenia wykonania akcji (t'). Dla ludzi i innych użytkowników przestrzeni inteligentnej wchodzących w interakcję z aplikacją typu IAI najważniejszy jest rzeczywisty czas reakcji T_Z, ponieważ określa (z ich punktu widzenia) szybkość reakcji na sytuacje w przestrzeni.

V.3.3. Stopień rozproszenia analizy kontekstu

W przestrzeni inteligentnej, w której działa aplikacja IAI, uzyskuje się różne stopnie rozproszenia analizy i przetwarzania kontekstu (r) pomiędzy usługi opakowujące obiekty DIND pełniące rolę sensorów i środowisko wykonywania aplikacji zbudowane w oparciu o maszynę PIAI. Im większe rozproszenie tym więcej oczekiwanych warunków (i związanych z nimi przetwarzania) jest sprawdzanych w usługach. Zostało to przedstawione na rysunku 24.



Rysunek 24 Rozproszenie logiki analizy kontekstu pomiędzy PIAI i sensory

To czy proces analizy (i ewentualnego przetwarzania) kontekstu jest obecny w usługach opakowujących sensory i jaki jest stopień jego zaawansowania uzależnione jest od możliwości obliczeniowych i decyzji projektowych podjętych w czasie tworzenia zarówno usług jak i środowiska wykonywania aplikacji IAI. Można wyróżnić 3 charakterystyczne sposoby rozproszenia logiki analizy i przetwarzania kontekstu:

- pełne rozproszenie (LS – logika w całości zawarta w usługach) - $r = 100\%$,
- brak rozproszenia (LK – logika w całości zawarta w PIAI) - $r = 0\%$,
- częściowe rozproszenie - $0\% < r < 100\%$.

Podejście LS jest oparte na wykorzystaniu usług opakowujących sensory, które w całości wykonują analizę (oraz ewentualne przetwarzanie) kontekstu. Ich działanie odpowiada sprawdzeniu czy kontekst spełnia oczekiwane warunki wywołania akcji zawarte w definicji aplikacji. Proces analizy kontekstu w PIAI, jest zatem obciążony w niewielkim stopniu. Jednakże nie ma on dostępu do surowych danych (tzn. danych, odczytanych przez sensory z przestrzeni inteligentnej) ponieważ nie są one przesyłane przez usługi (jedynie wynik mówiący o tym że warunki zostały spełnione). Nie przeprowadza on analizy otrzymanych od usług danych lecz musi jedynie wygenerować sygnał (patrz rozdział V.1) o konieczności uruchomienia akcji. Przeniesienie całości logiki analizy danych

kontekstowych do PIAI (podejście LK) powoduje, że sensory odczytują wartości parametrów obiektów środowiska, wchodzących w skład kontekstu aplikacji i następnie usługi je opakowujące (bez ich przetworzenia bądź analizy) wysyłają je do PIAI. W efekcie środowisko wykonywania aplikacji jest obciążone ich analizą, jednakże może przy wykorzystaniu tych danych uzyskiwać informacje o różnym poziomie abstrakcji (zależnym np. od innych wykonywanych równolegle aplikacji IAI). W podejściu mieszanym (które jest najczęściej stosowane) część logiki analizy jest zawarta w usługach opakowujących sensory a część w procesie analizy kontekstu w PIAI. W każdym z wymienionych podejść jest obecna funkcja dodawania kontekstu $(+)$, ponieważ każda z usług może dostarczać dane kontekstowe związane z fragmentem kontekstu aplikacji a PIAI musi analizować pełen kontekst aplikacji. Rozproszenie logiki analizy kontekstu r jest wyrażane przy pomocy odpowiednio mniejszej wielkości kontekstu (ψ - patrz definicja (4.21)) analizowanego w ramach PIAI:

$$\psi_{PIAI} = \begin{Bmatrix} \psi \; dla \; r = 0\% \\ 1 \; dla \; r = 100\% \\ \psi * (100\% - r) \; dla \; r \in (0\%, 100\%) \end{Bmatrix} \tag{5.5}$$

Wielkość kontekstu, który trzeba przeanalizować w ramach maszyny PIAI (ψ_{PIAI}), jest to fragment wielkości kontekstu aplikacji (ψ), odpowiadający części kontekstu aplikacji jaka nie jest analizowana w usługach opakowujących sensory. Dla $r = 0\%$ (co oznacza iż nie ma rozproszenia logiki – podejście LK) całość kontekstu aplikacji jest analizowana w ramach PIAI, w przypadku gdy całość kontekstu aplikacji analizowana jest przez sensory ($r = 100\%$ podejście LS) to wielkość kontekstu analizowana w ramach PIAI ma wartość 1. Przypadek ten oznacza, iż istnieje jedna usługa opakowująca obiekty DIND pełniące rolę sensorów, która analizuje cały kontekst aplikacji i do maszyny PIAI wysyła pojedynczą daną świadczącą o tym, iż oczekiwane warunki wywołania akcji zostały spełnione. Dla częściowego rozproszenia ($r \in (0\%, 100\%)$) wartość ψ_{PIAI} można wyrazić przy pomocy proporcjonalnie (do $100\% - r$) pomniejszonej wielkości kontekstu ψ. Zatem stopień rozproszenia jest zmienną zależną, którą można wyrazić przy pomocy wielkości kontekstu ψ.

Przykładem, obrazującym zagadnienie rozproszenia logiki analizy kontekstu, jest rozpoznawanie sytuacji rozpoczęcia seminarium (oz_1). Dla podejścia LS sensor odczytuje liczbę i listę osób w sali po czym usługa go opakowująca sprawdza czy godzina rozpoczęcia seminarium już minęła. Jeżeli wszystkie oczekiwane warunki wywołania akcji zostały spełnione przesyła ona cały kontekst akcji do PIAI, która następnie uruchamia akcję. W przypadku zastosowania podejścia LK usługi wysyłają jedynie dane kontekstowe

informujące o tym, że minął czas rozpoczęcia seminarium i o tym ile oraz jakie osoby są w sali, natomiast sprawdzenie czy oczekiwane warunki są spełnione jest wykonywane w maszynie PIAI (tzn. w środowisku wykonywania aplikacji IAI na tej maszynie zbudowanym).

Przekazanie danych z usług opakowujących sensory do środowiska wykonywania aplikacji IAI (zbudowanego na bazie maszyny PIAI) jest operacją chwilową i powoduje uruchomienie wykonywania instrukcji wymienionych w rozdziale V.3.1 w tym ι_{AK} odpowiadającej analizie kontekstu. Jednakże taka analiza musi dotyczyć kontekstu uwzględniającego dane kontekstowe jakie już zostały dostarczone (przez inne bądź tę samą usługę). Dlatego też niezbędne jest wprowadzenie pojęcia czasu ważności danych kontekstowych (patrz rozdział IV.1). Jest to czas przez jaki dane mogą kreować kontekst i tym sensie być analizowane pod kątem spełnienia oczekiwanych warunków wywołania akcji. Liczba danych kontekstowych, jakie aktualnie są analizowane w PIAI jest oznaczana jako s. Liczba zdarzeń pojawiających się w przestrzeni działania aplikacji, może wpływać na s, ale nie musi. Jest to spowodowane tym, że liczba dostarczanych do PIAI danych zależy od sensorów a nie od zdarzeń - na przykład, gdy sensory współpracują z PIAI w trybie synchronicznym, albo asynchronicznym ze stałym interwałem (na przykład odczyt poziomu naświetlenia co 1 sekundę). Wszystkie zdarzenia zewnętrzne, jakie zmieniły wartość poziomu naświetlenia w ciągu sekundy (a mogło być ich więcej niż jedno) zostaną przez środowisko wykonywania aplikacji zbudowane na bazie maszyny PIAI niezauważone, ponieważ usługa opakowująca sensor dostarczy jedynie dane o wartości poziomu naświetlenia po ostatniej zmianie.

Przedstawiona w niniejszym rozdziale maszyna PIAI zawiera mechanizmy konieczne do wykonania aplikacji IAI. Zostały one szczegółowo opisane w odniesieniu do zasad funkcjonowania maszyny a także podano jej własności. Można przy jej pomocy opisać taką aplikację (tzn. zasady jej działania zgodnie z definicją aplikacji IAI oraz mechanizmów koniecznych do ich realizacji) co stanowi wykazanie pierwszej tezy rozprawy doktorskiej. Maszyna ta jest teoretycznym modelem, będącym podstawą do utworzenia architektury środowiska wykonywania aplikacji IAI, jakie może być umieszczone w przestrzeni inteligentnej (jako silnik z Modelu 2 z rysunku 11), dzięki czemu o inteligencji przestrzeni mogą decydować aplikacje IAI w niej działające (patrz rozdział II). Przestrzeń inteligentna dotycząca określonej dziedziny zastosowań, w ramach której działa środowisko wykonywania aplikacji IAI (SWIAI) zbudowane na bazie maszyny PIAI oraz istnieją wymienione w rozdziale IV.3 rodzaje usług jest nazywana dedykowaną przestrzenią usług. Jej budowa została opisana w następnym rozdziale.

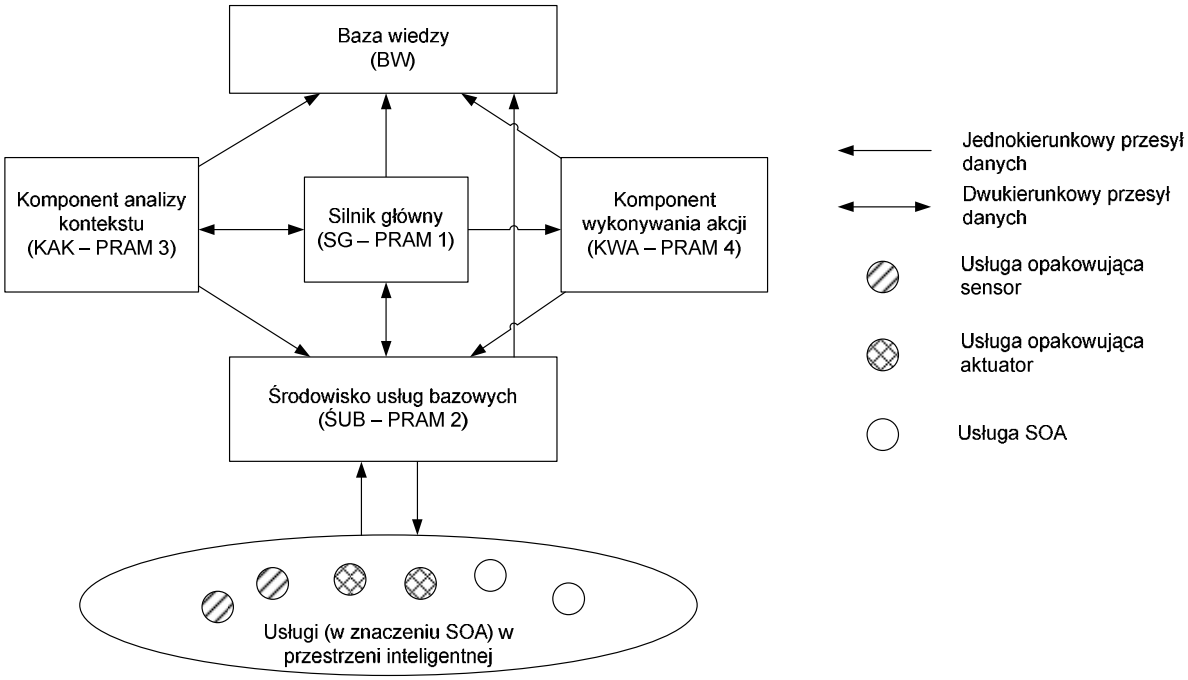
VI BUDOWA EKSPERYMENTALNEJ PRZESTRZENI INTELIGENTNEJ

Maszyna PIAI stanowi podstawę do zbudowania środowiska wykonywania aplikacji IAI (SWIAI). Opisano architekturę takiego środowiska oraz jej podstawowe komponenty funkcjonalne. Określono zasady ich komunikacji oraz utworzono procedury dla najistotniejszych aspektów ich działania. Wymieniono niezbędne typy usług, które są wykorzystywane w trakcie wykonania aplikacji IAI. W ten sposób utworzono przestrzeń inteligentną, mogącą wykonywać aplikacje IAI. Zostały także przedstawione najważniejsze problemy, które należało rozwiązać podczas jej budowy.

VI.1. Architektura IAK i opis komponentów

Analizując opis maszyny PIAI przedstawiony na listingu 4 można zauważyć, że korzysta ona z następujących funkcji: `kontekst()`, `sprawdz_warunki()`, `akcja_start()` i `transformujPIAI()` oraz `transformujA()`, które są wykonywane w sposób równoległy (tzn. w ramach różnych maszyn PRAM). Pierwsza z nich odpowiada za zbieranie danych (instrukcja ι_{ZD}) przesyłanych przez usługi opakowujące obiekty DIND pełniące rolę sensorów. W czasie wykonania akcji następuje wywołanie usługi z przestrzeni inteligentnej (co zostało opisane w rozdziale IV.1). Usługi te mogą być wykorzystane także w czasie analizy kontekstu (patrz rozdział III.4). Implementacja maszyny PIAI musi zatem zapewniać komunikację z usługami SOA znajdującymi się w przestrzeni (tzw. usługami bazowymi – patrz rozdział VI.1.2), której postać może być różna (np. jedne usługi przesyłają odczyty przy wykorzystaniu standardu Web Service a inne przy wykorzystaniu interfejsu CORBA). Co więcej mogą one w różny sposób odczytywać dane z przestrzeni i je organizować (co wynika z różnych ontologii opisu danych). Z tego względu podczas implementacji maszyny PIAI należy wyodrębnić osobną warstwę abstrakcji, która będzie realizowała zarówno komunikację z usługami, jak i transformowanie danych przez nie dostarczanych do wspólnej (standardowej) postaci (w przypadku zbierania danych odpowiadającą złożeniu funkcji `transformujPIAI(kontekst())`, natomiast w przypadku uruchamiania akcji - funkcji `transformujA()`). Postać ta zawiera opisy obiektów występujących w przestrzeni i ich parametrów (zgodnie z definicją przestrzeni inteligentnej z rozdziału IV.3), a także reguły opisujące zależności pomiędzy nimi. Można ją zatem określić jako bazę wiedzy o przestrzeni (patrz rozdział III.4). Funkcja `sprawdz_warunki()` odpowiada instrukcji sprawdzenia czy kontekst spełnia oczekiwane warunki wywołania akcji i jeżeli tak, to jej

wynikiem jest uruchomienie akcji (`akcja_start()`). Mimo, że z punktu widzenia maszyny PIAI najważniejsze jest uruchomienie akcji to jednakże implementacja środowiska wykonywania aplikacji IAI zbudowanego w oparciu o tę maszynę musi dodatkowo umożliwiać jej wykonanie. Ostatnim zakresem funkcjonalności maszyny, jaki musi być zawarty w systemie ją implementującym, jest przechowywanie definicji aplikacji (w listingu 4 jest to obiekt `IAI`) oraz nadzorowanie wykonania pętli `while {…}` (w maszynie PRAM 1) dzięki której projektant (patrz rysunek 12) może modyfikować definicje aplikacji. Każda z wymienionych funkcjonalności (analizowanie i przetwarzanie kontekstu, uruchamianie wraz z wykonaniem akcji, organizacja współpracy z usługami bazowymi, baza wiedzy oraz przechowywanie definicji aplikacji i komunikacja z projektantem) może zostać zorganizowana w postaci osobnego komponentu. W ten sposób powstała Interaktywna Architektura Komponentowa (IAK) jako architektura dla implementacji środowiska wykonywania aplikacji IAI, bazująca na maszynie PIAI. Architektura ta została przedstawiona na rysunku 25 (wraz z zaznaczonymi maszynami PRAM 1 – 4 opisanymi w listingu 4).



Rysunek 25 Interaktywna Architektura Komponentowa

Architektura IAK składa się z następujących komponentów, realizujących poszczególne instrukcje związane z przetwarzaniem i analizą danych kontekstowych (zdefiniowane w rozdziale V.1), wraz z odpowiadającymi im maszynami PRAM (zaznaczonymi w listingu 4):

- komponent silnika głównego (SG) – PRAM 1 - stanowiący centralny punkt systemu, poprzez który następuje komunikacja z projektantem aplikacji. Utrzymuje on ciągłość działania systemu (nadzoruje działanie pozostałych komponentów) oraz dobiera (na podstawie spełnionych oczekiwanych warunków) akcje do uruchomienia – realizuje instrukcję ι_{DA},
- środowisko usług bazowych (ŚUB) – PRAM 2 – komponent, poprzez który odbywa się komunikacja z usługami bazowymi – realizuje instrukcje ι_{ZD}, ι_{TDK}, ι_{TDA},
- komponent analizy kontekstu (KAK) – PRAM 3 - komponent odpowiedzialny za analizę danych kontekstowych i decyzję czy kontekst spełnia oczekiwane warunki wywołania akcji – realizuje instrukcję ι_{AK},
- komponent wykonywania akcji (KWA) – PRAM 4 - odpowiada za wykonanie akcji (ι_{UA}),
- baza wiedzy (BW) – komponent, zawierający wiedzę na temat przestrzeni inteligentnej. Jest on wykorzystywany przez pozostałe komponenty do interpretacji analizowanych (i przetwarzanych) przez nie danych.

Poszczególne komponenty funkcjonalnie odpowiadają mechanizmom zdefiniowanym w maszynie PIAI koniecznym do wykonania aplikacji IAI (na przykład KAK realizuje sprawdzenie oczekiwanych warunków wywołania akcji). Mogą być one zrealizowane w sposób rozproszony, tak iż każdy z nich będzie działać niezależnie od pozostałych i komunikować się z różnymi komponentami tego samego typu.

VI.1.1. Komponent silnika głównego (SG)

Silnik główny jest centralnym komponentem systemu, którego podstawową funkcją jest zapewnienie ciągłości funkcjonowania środowiska zbudowanego zgodnie z architekturą IAK (tzn. realizacja linii 2 – 11 z listingu 4) oraz komunikacja z projektantem (umożliwienie zmiany zawartości obiektu `IAI` – patrz rysunek 12). Projektant przekazuje do SG definicje aplikacji IAI, tzn. oczekiwanych warunków wywołania akcji oraz przypisanych im akcji (w formie dokumentów XML – jest to opisane w rozdziale VI.2), które są następnie przekazywane do właściwych dla nich komponentów środowiska (definicja oczekiwanych warunków do KAK a definicja akcji do KWA). W momencie spełnienia oczekiwanych warunków wywołania akcji komponent KAK (za pośrednictwem SG) przekazuje do KWA informację, o tym które z nich zostały spełnione (wraz z kontekstem jaki spowodował ich spełnienie – patrz rozdział V).

Wszystkie pozostałe komponenty IAK muszą najpierw zarejestrować się w silniku głównym (SG posiada rejestr wszystkich komponentów) i dopiero wtedy mogą rozpocząć

działanie w obrębie środowiska wykonywania aplikacji IAI. Podczas tego procesu negocjowane są klucze sesyjne wykorzystywane w czasie wymiany danych pomiędzy poszczególnymi komponentami. System rejestracji umożliwia również implementację mechanizmów niezawodnościowych (tzn. wymiany komponentu na skutek jego awarii). Ponieważ rejestracja komponentów jest scentralizowana w silniku głównym, to odpowiada on także za propagację informacji o rejestracji każdego nowego komponentu. Jednakże nie dla wszystkich komponentów propagacja ta wygląda tak samo (nie każdy komponent musi posiadać informację na temat wszystkich innych komponentów). Jej zasady przedstawiają się następująco:

- informacja o rejestracji lub wyrejestrowaniu komponentu analizy kontekstu (KAK) – nie jest propagowana,
- informacja o rejestracji lub wyrejestrowaniu komponentu wykonywania akcji (KWA) – nie jest propagowana,
- informacja o rejestracji lub wyrejestrowaniu środowiska usług bazowych (ŚUB) – propagowana jest do komponentu analizy kontekstu (KAK) oraz komponentu wykonywania akcji (KWA),
- informacja o rejestracji lub wyrejestrowaniu komponentu bazy wiedzy (BW) – jest propagowana do komponentów KAK, KWA i ŚUB.

Przedstawione zasady powodują, iż przesyłanie wszystkich danych kontekstowych odbywa się poprzez silnik główny (SG), a nie bezpośrednio do komponentu analizy kontekstu (KAK).

W celu umożliwienia realizacji opisanych powyżej funkcjonalności i dostarczenia ich innym komponentom IAK, określono następujące funkcje interfejsu wejściowego i wyjściowego SG:

- interfejs wejściowy:
  - rejestrację i wyrejestrowanie definicji aplikacji IAI,
  - odebranie przetransformowanej danej kontekstowej od ŚUB,
  - odebranie od KAK sygnału dotyczącego rozpoznania, iż oczekiwane warunki wywołania akcji zostały spełnione,
- interfejs wyjściowy:
  - rejestrację i wyrejestrowanie definicji oczekiwanych warunków wywołania akcji w KAK,
  - rejestrację i wyrejestrowanie definicji akcji w KWA,
  - przekazanie przetransformowanych danych kontekstowych do KWA,

- przekazanie do KWA sygnału dotyczącego rozpoznania, iż oczekiwane warunki wywołania akcji zostały spełnione,
- przekazanie informacji wynikających z propagacji informacji o komponentach.

VI.1.2. Środowisko usług bazowych (ŚUB)

Komponent ŚUB stanowi warstwę komunikacji z usługami bazowymi. Są to usługi (w znaczeniu SOA) wykorzystywane podczas wykonywania aplikacji IAI (patrz rozdział IV.3). W zależności od sposobu współpracy z ŚUB zostały wyróżnione dwa ich rodzaje: usługi dedykowane i usługi dodatkowe. Usługi dedykowane posiadają określony interfejs komunikacyjny, pozwalający na ich rejestrację w środowisku wykonywania aplikacji IAI zbudowanym zgodnie z IAK oraz dostarczanie do niego danych kontekstowych (w zależności od sposobu pracy – patrz rozdział V.2). Usługi dodatkowe obejmują pozostałe usługi (w znaczeniu SOA), które nie posiadają takiego interfejsu i w związku z tym wymagają istnienia mechanizmu umożliwiającego ich wykorzystanie (w tym także rejestrację) w ramach środowiska. Ze względu na opisywaną w rozdziale V.2 różnorodność usług, konieczne jest transformowanie przesyłanych przez nie danych. Jego sposób jest określany podczas rejestracji usługi w środowisku zbudowanym w oparciu o architekturę IAK, która odbywa się poprzez komponent ŚUB. W tym celu przekazywany jest jej opis (zgodny z modelem usługi opisanym w rozdziale VI.2), określający w jaki sposób wyrazić dane dostarczane bądź dane pobierane przez usługę przy wykorzystaniu opisu zawartego w bazie wiedzy. Po transformacji dane (w ujednoliconej formie) są przekazywane do silnika głównego i następnie mogą być uwzględnione w kontekście aplikacji – linia 16 z listingu 4. Zatem zakres funkcjonalny interfejsu wejściowego i wyjściowego tego komponentu, umożliwiający wykorzystanie dostarczanych przez ŚUB funkcjonalności innym komponentom architektury IAK, obejmuje:

- interfejs wejściowy:
  - odebranie informacji wynikających z propagacji informacji o komponentach od SG,
  - rejestrację i wyrejestrowanie komponentu ŚUB w SG,
  - odebranie danej kontekstowej od usługi opakowującej sensor (w formie komunikatu),
  - zarejestrowanie i wyrejestrowanie usługi bazowej przez projektanta,
- interfejs wyjściowy:
  - przekazanie danej kontekstowej do SG,
  - pobranie danej kontekstowej od usługi, jaka nie jest dedykowana (usługi

dodatkowej) do współpracy z IAK,
- zarejestrowanie i wyrejestrowanie usługi bazowej w BW,
- wyszukanie usługi bazowej w BW.

VI.1.3. Baza wiedzy (BW)

Baza wiedzy jest to komponent zawierający wiedzę na temat przestrzeni działania aplikacji IAI. Zawiera ona pojęcia opisujące przestrzeń inteligentną (znajdujące się w niej obiekty i ich parametry) wraz z relacjami pomiędzy nimi. Tak zbudowany opis przestrzeni wykorzystywany jest podczas transformacji danych wykonywanej w celu ujednolicenia znaczenia wszystkich danych kontekstowych, jakie są analizowane i przetwarzane w środowisku utworzonym w oparciu o architekturę IAK. Dochodzi do niej podczas komunikacji z usługami bazowymi. Jednakże oprócz informacji na temat semantyki i syntaktyki danych, jakie muszą być przekazane bądź odczytane z usługi, potrzebne są także informacje dotyczące samego sposobu jej wywołania (np. w przypadku usług sieciowych dotyczy to sposobu konstrukcji komunikatu SOAP). W tym celu baza wiedzy zawiera również opisy usług, które znajdują się w przestrzeni inteligentnej, przez co stanowi semantyczny ich rejestr. Rejestr ten przechowuje informacje o sposobie komunikacji z daną usługą w postaci jej modelu (został on opisany w rozdziale VI.2) uwzględniającego wymagane transformacje danych (reprezentowane przez funkcje `transformujPIAI()` i `transformujA()` – patrz listing 4). Aby komponent BW mógł pełnić opisane funkcje zakres funkcjonalny jego interfejsu wejściowego obejmuje:

- rejestrację i wyrejestrowanie komponentu w SG,
- zarejestrowanie i wyrejestrowanie usługi bazowej (przekazane od ŚUB),
- wyszukanie usługi bazowej (przekazane od ŚUB),
- pobranie wiedzy o kontekście przez KAK,
- zmiana wiedzy o przestrzeni inteligentnej (przekazana od projektanta).

Komponent ten nie korzysta z innych komponentów, dlatego też nie posiada interfejsu wyjściowego.

VI.1.4. Komponent analizy kontekstu (KAK)

Głównym zadaniem tego komponentu jest interpretacja definicji oczekiwanych warunków wywołania akcji otrzymanych z silnika głównego (SG) oraz sprawdzanie czy kontekst aplikacji (zbudowany na podstawie danych kontekstowych) spełnia te warunki. Podczas interpretacji oczekiwanych warunków, komponent może komunikować się z bazą wiedzy (BW), aby uzyskać dodatkowe informacje na temat wykorzystywanych w nich obiektów i parametrów (patrz definicja (4.14), na przykład KAK może odnajdować

synonimy pojęć je opisujących). Jak to zostało opisane w rozdziale III.4 w czasie analizowania kontekstu może nastąpić wywołanie usług znajdujących się w przestrzeni (w przypadku IAK są to usługi bazowe, z którymi komunikacja odbywa się poprzez ŚUB) i przekazanie im danych kontekstowych do analizy. Sytuacja taka występuje jeżeli analiza danych kontekstowych musi dostosowywać się do bieżących możliwości przestrzeni, lub podczas wykonania aplikacji konieczne jest przeanalizowanie dodatkowych danych, do których środowisko wykonywania aplikacji IAI zbudowane zgodnie z architekturą IAK nie ma dostępu, lecz mogą być wykorzystane w analizie poprzez wywołanie usług bazowych. Wywołania te możliwe są dzięki mechanizmowi komunikacji opartemu na wykorzystaniu opisów usług zawartych w bazie wiedzy (BW), zorganizowanych na podstawie modelu usługi (patrz rozdział VI.2). W momencie, gdy oczekiwane warunki wywołania akcji zostały spełnione komponent KAK wysyła sygnał do SG wraz z bieżącym kontekstem aplikacji.

Zakres funkcjonalny interfejsów wejściowego i wyjściowego KAK, umożliwiający jego wykorzystanie przez pozostałe komponenty architektury IAK, obejmuje:

- interfejs wejściowy:
  - odebranie informacji wynikających z propagacji informacji o komponentach od SG,
  - rejestrację i wyrejestrowanie komponentu w SG,
  - rejestracja i wyrejestrowanie definicji oczekiwanych warunków wywołania akcji od SG,
  - odebranie danej kontekstowej od SG,
- interfejs wyjściowy:
  - przekazanie sygnału o spełnieniu oczekiwanych warunków wywołania akcji do SG,
  - pobranie wiedzy o kontekście (komunikacja do BW).

VI.1.5. Komponent wykonywania akcji (KWA)

Komponent ten otrzymuje od silnika głównego (SG) komunikaty, zawierające definicje akcji jakie zostały przekazane przez projektanta aplikacji IAI w jej definicji. Zadaniem KWA jest ich interpretacja i przetworzenie w taki sposób, aby mogły zostać wykonane przy wykorzystaniu usług znajdujących się w przestrzeni inteligentnej, jakie zostały zarejestrowane poprzez ŚUB. W czasie ich uruchomienia może zajść konieczność dodatkowej transformacji danych (patrz rozdział V.2), która jest możliwa dzięki BW. Ponieważ usługi bazowe mogą także korzystać z innych usług znajdujących się w przestrzeni, wykonanie akcji może przyjąć formę scenariusza – patrz rozdział III.3. Jej

uruchomienie jest skutkiem spełnienia przez kontekst aplikacji oczekiwanych warunków wywołania akcji co powoduje wysłanie z komponentu KAK sygnału sterującego (patrz rozdział V.1) do SG, który przesyła go dalej do KWA. Wraz z otrzymaniem tego sygnału przesyłany jest kontekst wykonania akcji (zawarty w kontekście aplikacji – patrz listing 4).

Zakres funkcjonalny interfejsu wejściowego i wyjściowego tego komponentu, umożliwiający jego działanie w ramach architektury IAK, obejmuje:

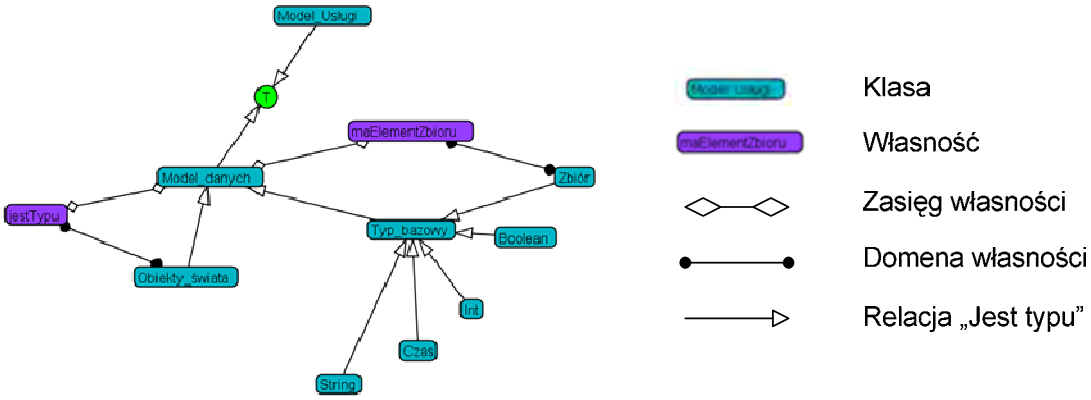
- interfejs wejściowy:
  - odebranie informacji wynikających z propagacji informacji o komponentach od SG,
  - rejestrację i wyrejestrowanie komponentu w SG,
  - rejestracja i wyrejestrowanie definicji akcji od SG,
  - odebranie sygnału informującego o spełnieniu oczekiwanych warunków wywołania akcji od SG,
- interfejs wyjściowy:
  - pobranie wiedzy o kontekście (komunikacja do BW),
  - wyszukanie usług bazowej (do ŚUB),
  - uruchamianie usługi bazowej.

Wymienione komponenty odpowiadają mechanizmom i koniecznym do ich realizacji procedurom, jakie muszą być zapewnione przez środowisko zbudowane na bazie maszyny PIAI (w celu wykonania aplikacji IAI). Jest to wykazanie drugiej tezy rozprawy doktorskiej mówiącej o tym, iż implementacja maszyny PIAI jest możliwa i prowadzi do architektury komponentowej IAK, składającej się z opisanych komponentów.

VI.2. Implementacja przestrzeni

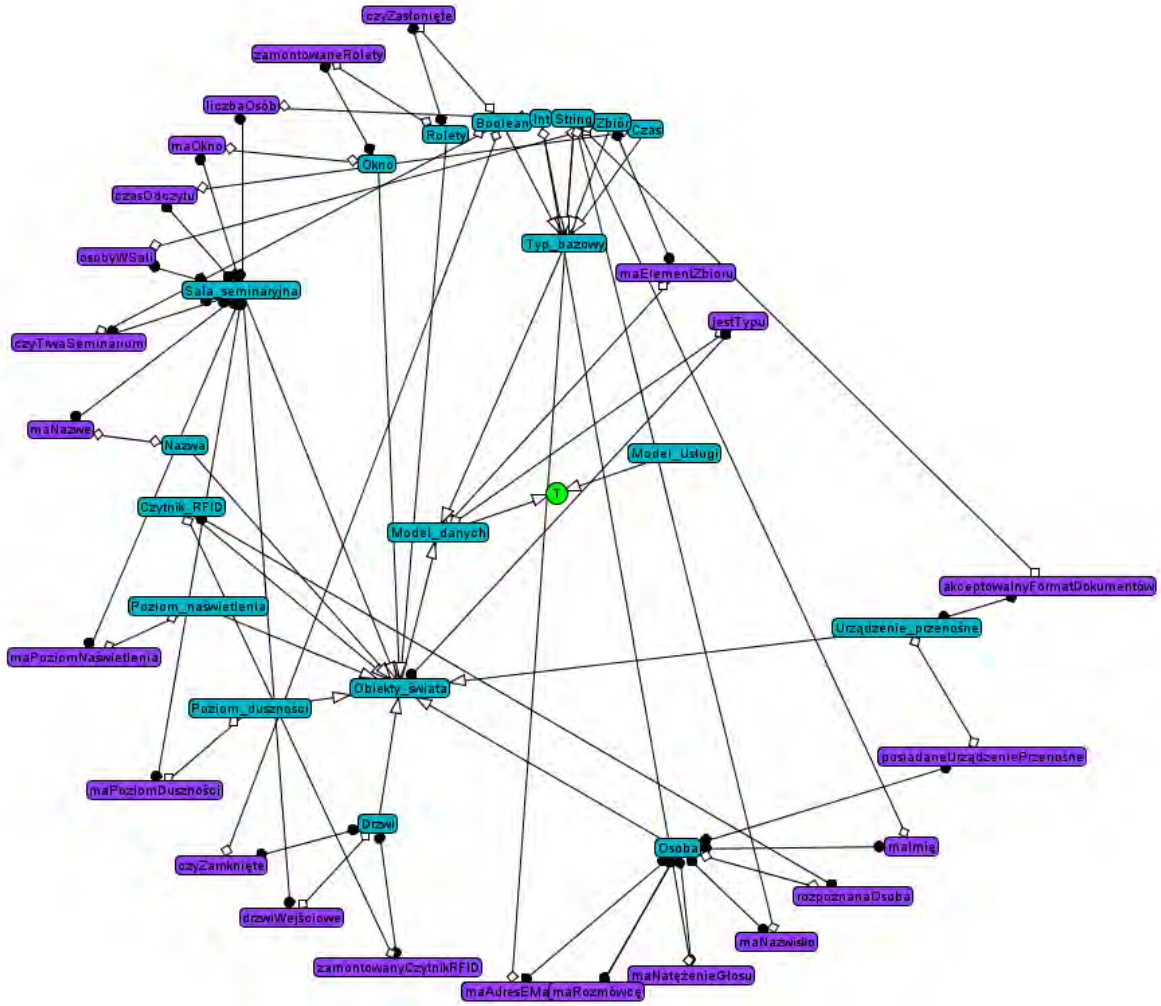
Przedstawioną architekturę IAK zaimplementowano w postaci systemu SWIAI (System Wykonywania Interaktywnych Aplikacji Iteracyjnych). Komponenty systemu zostały opracowane w formie usług sieciowych (baza wiedzy w technologii Java, natomiast pozostałe komponenty w technologii .NET). Wykorzystywane usługi bazowe również działają zgodnie ze standardem Web Service. Jako serwery aplikacyjne posłużyły IIS oraz Apache z modułem Axis, natomiast do opisu danych zastosowano język XML. Baza wiedzy zorganizowana jest w postaci ontologii (patrz rozdział III.4) zapisanej w języku OWL a do jej przeszukiwania wykorzystano język SPARQL. Do utworzenia i utrzymywania ontologii wykorzystano aplikację Protege natomiast do jej wykorzystania (dodawania oraz usuwania pojęć i wykonywania zapytań podczas działania SWIAI) została użyta biblioteka Jena. Ogólna postać ontologii wykorzystywanej w implementacji

została przedstawiona na rysunku 26.



Rysunek 26 Ogólna postać ontologii dla bazy wiedzy

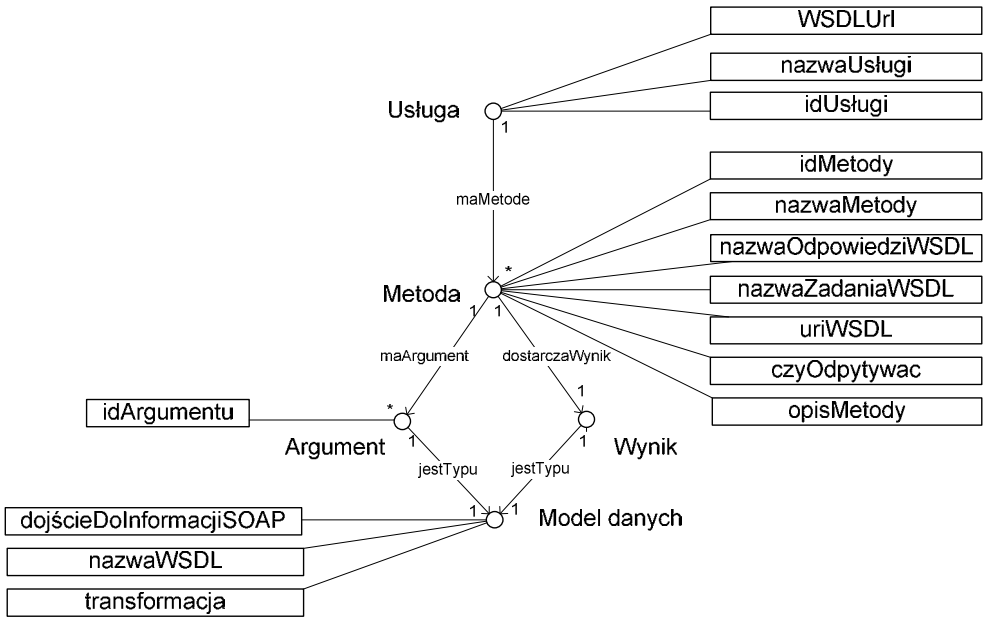
Ontologia zawiera dwa główne pojęcia: Model_danych oraz Model_usługi. Pierwsze z nich jest pojęciem bazowym dla wszystkich innych jakimi można opisać przestrzeń funkcjonowania aplikacji (tzn. obiekty i ich parametry). Pojęcia te podzielone są dalej na dwie klasy: Obiekty_świata oraz Tyb_bazowy. Obiekty_świata (przestrzeni) opisują obiekty znajdujące się w przestrzeni, takie jak sala seminaryjna, drzwi, okno, osoba itp., natomiast Typ_bazowy reprezentuje typy wartości, jakie mogą przyjmować parametry obiektów przestrzeni (zbiór zawierający obiekty typu Model_danych, Czas, String, Int, itd.). W tej ogólnej ontologii wprowadzono dwie własności (typu ObjectProperty): maElementZbioru oraz jestTypu. Własność maElementZbioru dotyczy zbioru obiektów przestrzeni (takich jak zbiór osób w sali seminaryjnej) i łączy go z dowolnym obiektem z modelu danych (dzięki czemu można wyrazić iż obiekt należy do zbioru). Druga własność (jestTypu) pozwala opisać obiekty należące do opisu przestrzeni przy pomocy pojęć typu Typ_bazowy. Dzięki temu można w usystematyzowany sposób opisać, że na przykład identyfikator pokoju jest typu znakowego (String), natomiast poziom naświetlenia jest typu liczbowego (Int). Fragment zaimplementowanego opisu przestrzeni został przedstawiony na rysunku 27. Odpowiada on przykładowi wprowadzonemu w rozdziale IV.1.



Rysunek 27 Fragment opisu przestrzeni dla przykładu z rozdziału IV

Opis zawarty w bazie wiedzy (w wyrażony przy pomocy ontologii) jest wspólny dla wszystkich operacji wykonywanych w ramach przetwarzania i analizy danych kontekstowych (patrz rozdział V.3.1). Wszystkie pochodzące od usług opakowujących sensory dane kontekstowe, niezależnie od ich pierwotnej postaci, są do tego wspólnego opisu transformowane. Jest to możliwe dzięki opisom usług bazowych (zbliżonym do standardu OWL-S), przekazywanym w czasie ich rejestracji. Opisy te zapisywane są w ontologii w postaci *individuals* (odnoszących się do obiektów dziedziczących z Model_danych), dzięki czemu pozwalają wyrazić z jakich danych korzysta usługa oraz w jaki sposób należy przetransformować odczytywane i wysyłane do niej dane. Ze względu na to, iż sposoby zarówno transformacji danych jak i komunikacji z usługą mogą być różne dla różnych usług opracowano model usługi (patrz rys. 28), który w uniwersalny sposób opisuje jej syntaktykę (tzn. sposób komunikacji) i semantykę (tzn. znaczenie

wymienianych danych).



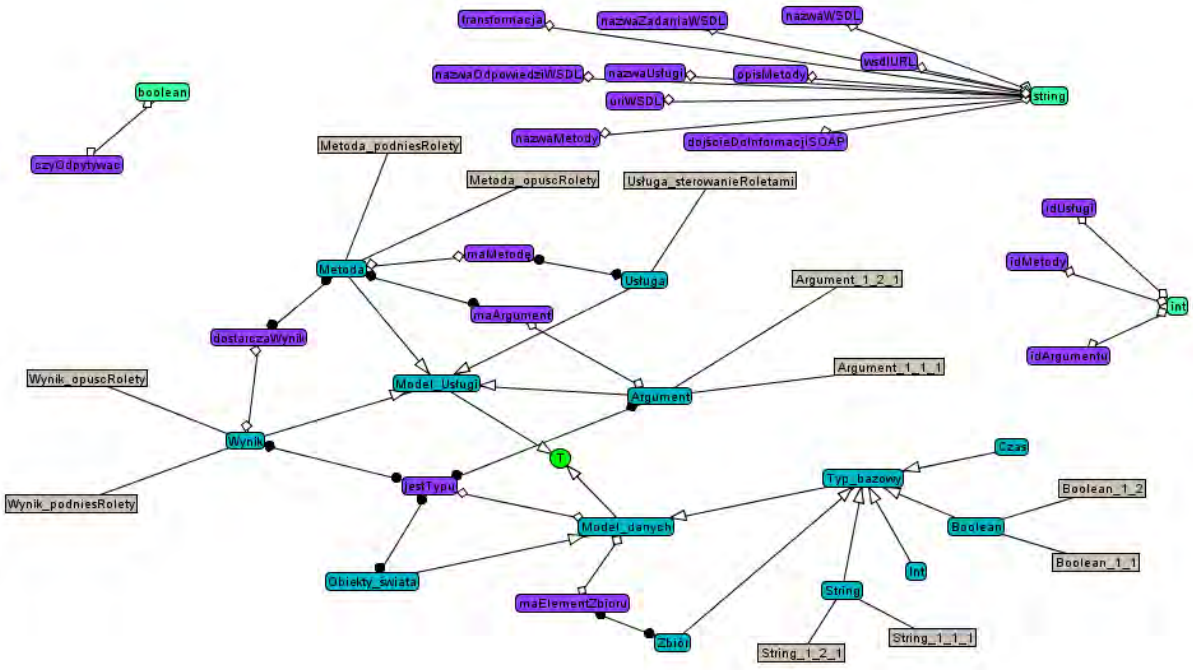
Rysunek 28 Model usługi

Usługa ma pewien globalny (względem wszystkich usług jakie są zarejestrowane w środowisku wykonywania aplikacji IAI zbudowanym zgodnie z architekturą IAK) identyfikator, jaki jest określany w czasie jej rejestracji w BW. Ze względu na wykorzystanie usług sieciowych (Web Service), przyjęto iż każda usługa posiada także pewien opis w standardzie WSDL, do którego odnośnik URL znajduje się w atrybucie WSDLUrl oraz nazwę (odpowiadającą polu *service name*) umożliwiającą jej wywołanie. Pojedyncza usługa Web Service udostępnia jedną bądź więcej metod. Podobnie jak usługi, metody posiadają identyfikatory oraz nazwy. Dodatkowo, komunikacja przy wykorzystaniu standardu Web Service zakłada, że wszystkie komunikaty zawarte są w ramach przestrzeni nazw. Jest ona zawarta w atrybucie uriWSDL. Opis metody (zawarty w atrybucie opisMetody z rysunku 28) pozwala na jej semantyczne wyszukiwanie (np. podczas analizy kontekstu – patrz rozdział III.4), natomiast atrybut czyOdpytywać wykorzystywany jest w celu określenia czy metoda jest dedykowana do pracy w ramach środowiska zgodnego z architekturą IAK czy nie (na przykład należy do dowolnej usługi w znaczeniu SOA pochodzącej z Internetu. Pozostałe dwa atrybuty dotyczące metod (nazwaZadaniaWSDL oraz nazwaOdpowiedziWSDL) wykorzystywane są do konstruowania komunikatów SOAP w zakresie odnoszącym się do wywołania metody. Każda metoda usługi posiada zero bądź więcej argumentów (które są identyfikowane przy pomocy identyfikatora idArgumentu) oraz zwraca jeden wynik. Zarówno argumenty jak i wynik odwołują się do pojęć dziedziczących z pojęcia Model_danych (patrz rysunek 27).

Do zawarcia w modelu informacji na temat transformacji (patrz rozdział VI.1.3) służą atrybuty nazwaWSDL (określający nazwę pola odpowiadającego przesyłanej bądź odbieranej od usługi danej w wywołaniu SOAP), dojścieDoInformacjiSOAP (pozwalający określić w komunikacie SOAP położenie struktury, która reprezentuje daną) oraz transformacja, dzięki którym dane z komunikatu SOAP przekształcane są do postaci zgodnej z opisem przestrzeni zawartym w ontologii (bazie wiedzy). W przypadku przesyłania danych do usługi (tzn. konstruowaniu argumentów jej wywołania w IAK) te same atrybuty wykorzystywane są do zbudowania odpowiedniego wywołania SOAP. Zawarte w bazie wiedzy informacje wykorzystywane są podczas działania środowiska zbudowanego zgodnie z architekturą IAK w odniesieniu do następujących procesów:

- wyszukiwanie i komunikacja z usługą w czasie analizy kontekstu i w trakcie wywoływania usług podczas uruchamiania akcji:
    - odnalezienie potrzebnej metody (na podstawie jej semantycznego opisu zawartego w BW który został przekazany przez dostawcę usługi w czasie jej rejestracji w ŚUB oraz opisu warunków w definicji aplikacji) oraz usługi przez jaką jest udostępniana,
    - sprawdzenie zgodności interfejsu wejściowego i wyjściowego metody z opisem warunku zwartym w definicji aplikacji. Opisy interfejsów jak i warunków zawarte są w bazie wiedzy (w formie *individuals*), dlatego też sprawdzenie polega na porównaniu jej fragmentów,
    - w przypadku zgodności uruchomienie metody udostępnianej przez usługę, w ramach którego następuje konstrukcja odpowiedniego komunikatu SOAP (na podstawie informacji o interfejsie wejściowym usługi zawartym w bazie wiedzy),
- odbieranie danych od usług opakowujących sensory:
    - dla usług dedykowanych do współpracy z IAK, następuje odebranie danych przesyłanych przez usługę opakowującą sensor i przetworzenie danych do standardowej postaci (zgodnej z bazą wiedzy). Proces przetworzenia wykorzystuje atrybuty nazwaWSDL i dojscieDoInformacjiSOAP,
    - w przypadku usług dodatkowych, następuje ich wywoływanie przez środowisko wykonywania aplikacji IAI zbudowane zgodnie z architekturą IAK. Po odczytaniu z usługi danych kontekstowych następuje ich przetworzenie analogicznie jak w przypadku usług dedykowanych,

Przykład zarejestrowanej w środowisku SWIAI usługi został przedstawiony na rysunku 29 (ze względu na przejrzystość rysunek nie przedstawia własności typu DataProperty).



Rysunek 29 Przykład zarejestrowanej w SWIAI usługi sterującej roletami

Jest to usługa umożliwiająca podnoszenie i opuszczanie rolet w oknach. Udostępnienia ona dwie metody (Metoda_opuscRolety i Metoda_podniesRolety). Każda z nich przyjmuje jeden argument (identyfikator sali – będąca wartością typu *String*) oraz zwraca rezultat podniesienia bądź opuszczenia rolet (mówiący o tym czy się udało czy nie – będący wartością logiczną typu *Boolean*). Przykład opisu usługi, jaki musi być dostarczony w czasie jej rejestracji, został przedstawiony w listingu 5.

```
    <ServiceDescription>
        <OntologyDefinition>
           <![CDATA[
            <rdf:RDF
                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
                xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
                xmlns:Ont="http://ont.owl#">
               <owl:Ontology rdf:about="http://ont.owl"/>
```

```
                <Ont:Usluga
rdf:about="http://ont.owl#Usluga_sterowanieRoletami">
                 <Ont:maMetode>
                   <Ont:Metoda
rdf:about="http://ont.owl#Metoda_opuscRolety">
                     <Ont:dostarczaWynik>
                       <Ont:Wynik
rdf:about="http://ont.owl#Wynik_opuscRolety">
                         <Ont:jestTypu>
                           <Ont:Boolean
rdf:about="http://ont.owl#Boolean_%SID%_1">

<Ont:dojscieDoInformacjiSOAP>[opuscRoletyResponse/opuscRolety
Result]</Ont:dojscieDoInformacjiSOAP>
                           </Ont:Boolean>
                         </Ont:jestTypu>
                       </Ont:Wynik>
                     </Ont:dostarczaWynik>
                     <Ont:maArgument>
                       <Ont:Argument
rdf:about="http://ont.owl#Argument_%SID%_1_1">
                         <Ont:jestTypu>
                           <Ont:String
rdf:about="http://ont.owl#String_%SID%_1_1">

<Ont:nazwaWSDL>identyfikatorSali</Ont:nazwaWSDL>
                           </Ont:String>
                         </Ont:jestTypu>
                         <Ont:idArgumentu>1</Ont:idArgumentu>
                       </Ont:Argument>
                     </Ont:maArgument>
                     <Ont:idMetody>1</Ont:idMetody>

<Ont:nazwaOdpowiedziWSDL>opuscRoletyResponse</Ont:nazwaOdpowi
edziWSDL>

<Ont:nazwaMetody>opuscRolety</Ont:nazwaMetody>
                     <Ont:nazwaZadaniaWSDL>        opuscRolety
</Ont:nazwaZadaniaWSDL>

<Ont:uriWSDL>http://tempuri.org/</Ont:uriWSDL>
```

```
                    <Ont:czyOdpytywac>nie</Ont:czyOdpytywac>
                    <Ont:opisMetody>Opuszcza
rolety</Ont:opisMetody>
                  </Ont:Metoda>
                </Ont:maMetode>
                <Ont:maMetode>
                  <Ont:Metoda
rdf:about="http://ont.owl#Metoda_podniesRolety">
                    <Ont:dostarczaWynik>
                      <Ont:Wynik
rdf:about="http://ont.owl#Wynik_ podniesRolety">
                        <Ont:jestTypu>
                          <Ont:Boolean
rdf:about="http://ont.owl#Boolean_%SID%_2">

<Ont:dojscieDoInformacjiSOAP>[podniesRoletyResponse/podniesRo
letyResult]</Ont:dojscieDoInformacjiSOAP>
                        </Ont:Boolean>
                      </Ont:jestTypu>
                    </Ont:Wynik>
                  </Ont:dostarczaWynik>
                  <Ont:maArgument>
                    <Ont:Argument
rdf:about="http://ont.owl#Argument_%SID%_2_1">
                      <Ont:jestTypu>
                        <Ont:String
rdf:about="http://ont.owl#String_%SID%_2_1">

<Ont:nazwaWSDL>identyfikatorSali</Ont:nazwaWSDL>
                        </Ont:String>
                      </Ont:jestTypu>
                      <Ont:idArgumentu>1</Ont:idArgumentu>
                    </Ont:Argument>
                  </Ont:maArgument>
                  <Ont:idMetody>2</Ont:idMetody>

<Ont:nazwaOdpowiedziWSDL>podniesRoletyResponse</Ont:nazwaOdpo
wiedziWSDL>

<Ont:nazwaMetody>podniesRolety</Ont:nazwaMetody>
```

```
<Ont:nazwaZadaniaWSDL>podniesRolety</Ont:nazwaZadaniaWSDL>

<Ont:uriWSDL>http://tempuri.org/</Ont:uriWSDL>
                    <Ont:czyOdpytywac>nie</Ont:czyOdpytywac>
                    <Ont:opisMetody>Podnosi
rolety</Ont:opisMetody>
                  </Ont:Metoda>
                </Ont:maMetode>
                <Ont:idUslugi>%SID%</Ont:idUslugi>

<Ont:WSDLUrl>http://localhost/Test/Service1.asmx?wsdl</Ont:WS
DLUrl>
            <Ont:maNazwe>SterowanieRoletami</Ont:maNazwe>
                </Ont:Usluga>
             </rdf:RDF>
           ]]>
         </OntologyDefinition>
    </ServiceDescription>
```

Listing 5 Przykład opisu usługi

Opis usługi zorganizowany jest w BW zgodnie z modelem przedstawionym na rysunku 31. Przed jego dodaniem do ontologii następuje przyporządkowanie unikalnego identyfikatora (w listingu 5 reprezentowanego jako `%SID%`) i następnie zapisanie całego opisu jako *individuals* (odnoszące się do pojęć dziedziczących z pojęcia Model_danych). Wygenerowane identyfikatory wykorzystywane są podczas konstruowania nazw dla każdego *individual* wynikającego z modelu jak i dla typów danych wykorzystywanych w definicji (na przykład `Boolean_%SID%_1`). Dzięki temu możliwe jest późniejsze porównywanie obiektów i parametrów opisanych przy pomocy tych pojęć na poziomie bazy wiedzy.

Ze względu na konieczność zapewnienia komunikacji SWIAI z dowolną usługą sieciową (patrz rozdział VI.1.2), został przygotowany mechanizm *proxy*, który pobiera w czasie rejestracji usługi plik WSDL ją opisujący i na jego podstawie tworzy obiekt służący do komunikacji. Wszystkie wywołania usługi odbywają się poprzez specjalną metodę tego obiektu, do której przekazywany jest dokument XML, opisujący zawartość wywołania (wartości argumentów). Dokument ten konstruowany jest na podstawie opisu usługi zawartego w ontologii. Dla usług dedykowanych (posiadających odpowiedni interfejs do komunikacji z IAK) tworzony jest obiekt PassiveWrapper, który odpowiada za komunikację z daną usługą, natomiast dla pozostałych usług w znaczeniu SOA

tworzony jest obiekt ActiveWrapper, który z jednej strony wykonuje ich rejestrację a z drugiej strony pozwala na ich odpytywanie o dane kontekstowe. Opisany sposób komunikacji oraz wykorzystanie języka XML do opisu danych kontekstowych spowodował, iż wykorzystano standard XSLT oraz wyrażenia XPath do ich transformacji.

Na potrzeby implementacji został opracowany format XML definicji aplikacji IAI przekazywanej przez projektanta, tzn. pary: oczekiwane warunki wywołania akcji – akcja. Format ten został przedstawiony na listingu 6 (odpowiada parze $oz_3 - a_3$ z przykładu z rozdziału IV.1):

```
    <UserContextScenarioDescription>
      <ContextDefinition>
        <Variables>
          <Variable                    name="salaSeminaryjna"
def="Ont:SalaSeminaryjna"/>
        </Variables>
        <Relations>
          <Relation                                  name="rel1"
expr="czySeminarium([var:salaSeminaryjna;//Ont:identyfikator]
,[var:salaSeminaryjna;//Ont:poraSeminarium]") groups=""/>
          <Relation                                  name="rel2"
expr="czyWiększe([var:salaSeminaryjna;//Ont:poziomNaswietleni
a],{http://ont.owl#intGlobal}:40)" groups=""/>
          <Relation                              name="mainRel"
expr="and([rel:rel1],[rel:rel2])" main="true" groups="" />
        </Relations>
      </ContextDefinition>
      <ScenarioDefinition type="nativeInvoke">
        <InputDefinition> <![CDATA[
          <sterowanieRoletami
xmlns="http://www.eti.pg.gda.pl/wsdl/SterowanieRoletami">

<nazwaPokoju>[salaSeminaryjna/Ont:maIdentyfikator]</nazwaPoko
ju>
          </sterowanieRoletami>]]>
        </InputDefinition>

<WSDLUrl>http://localhost:8081/ode/processes/Akcja</WSDLUrl>
        <ServiceName>SterowanieRoletami</ServiceName>
        <MethodName>opuscRolety</MethodName>
      </ScenarioDefinition>
```

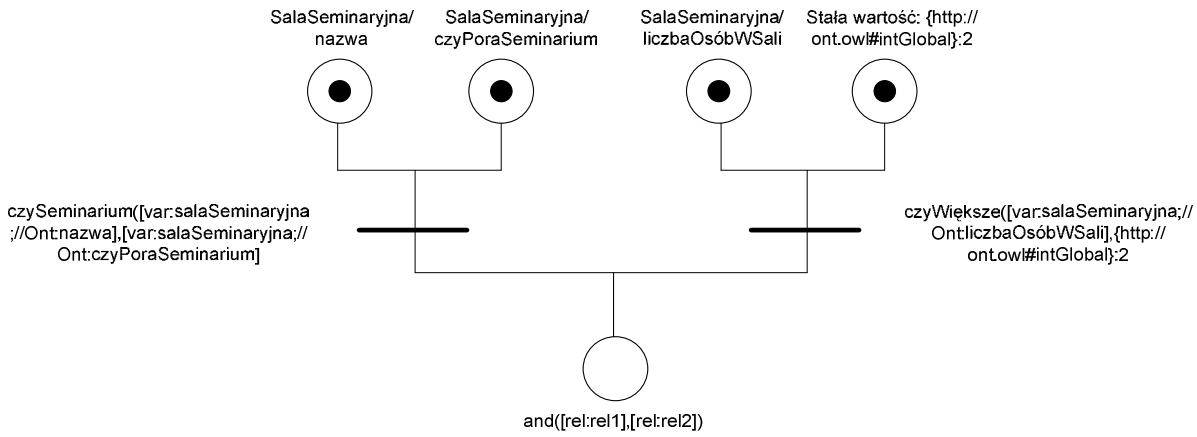
```
</UserContextScenarioDescription>
```

Listing 6 Definicja pojedynczej pary: oczekiwane warunki wywołania akcji – akcja (w definicji aplikacji typu IAI)

Definicja składa się z dwóch głównych elementów. Pierwszy z nich (ContextDefinition) odpowiada definicji oczekiwanych warunków wywołania akcji. Zawiera on listę zmiennych (Variables), które mogą przechowywać informacje na temat obiektów i parametrów oraz relacji (Relations). Relacje umożliwiają wykonywanie operacji na wartościach zmiennych, jakie są ustawiane na podstawie danych kontekstowych przekazywanych przez usługi opakowujące sensory i dzięki temu pozwalają na wyrażenie oczekiwanych warunków. W przykładzie zmienną jest salaSeminaryjna (reprezentująca obiekt zdefiniowany w ontologii (Ont:SalaSeminaryjna)) natomiast relacje odnoszą się do parametrów przypisanych tej zmiennej o nazwach identyfikator, poraSeminarium i liczbaOsóbWSali. W ramach relacji wykorzystywane są semantyczne opisy funkcji, jakie należy wykonać na przekazanych im argumentach (w implementacji ograniczono się do maksymalnie dwóch argumentów w każdej relacji) – na przykład czySeminarium, zawiera dwa argumenty opisane przy pomocy pojęć z bazy wiedzy. Podczas wykonywania operacji określonych w relacji są wyszukiwane i wywoływane usługi bazowe (ich metody) zwracające wartość logiczną. Proces wyszukiwania składa się z dwóch kroków, z których pierwszy obejmuje odnalezienie usługi o semantycznym opisie funkcjonalności (na przykład czySeminarium) takim samym jak wskazany w relacji, natomiast w ramach drugiego kroku następuje sprawdzenie zgodności interfejsów (argumenty oraz zwracany wynik). Wyszukiwana usługa musi operować na danych kontekstowych, jakie wynikają z definicji relacji (opisanych przy pomocy pojęć z opisu świata na przykład Ont:salaSeminaryjna.Ont:identyfikator) oraz zwracać wartość logiczną. Zarówno interfejs usługi, którego opis dostarczony jest w trakcie jej rejestracji jak i wymagany interfejs usługi opisany w definicji relacji wyrażone są przy pomocy pojęć z bazy wiedzy (każdy interfejs opisany jest przy pomocy drzewa, złożonego z pojęć opisujących przestrzeń i relacji między nimi). Zatem dopasowanie interfejsu odbywa się poprzez porównanie fragmentów ontologii (np. Ont:salaSeminaryjna.Ont:identyfikator z Ont:salaSeminaryjna_1_1_1. Ont:identyfikator_1_1_1), gdzie pierwszy opis jest zawarty w definicji relacji, natomiast drugi jest wynikiem rejestracji usługi w bazie wiedzy a dane przez nią dostarczane są do takiej postaci transformowane. W przykładzie założono, iż usługa opakowująca sensor dostarcza jednocześnie wszystkie informacje wymagane do analizy kontekstu (nazwę sali seminaryjnej, czy jest pora seminarium oraz liczbę osób w sali). Drugi z elementów definicji (ScenarioDefinition) to definicja akcji wraz ze sposobem transformacji danych kontekstowych potrzebnych do jej wykonania. Akcje są

usługami (patrz rozdział IV.1) bazowymi, które w utworzonej implementacji zawierają scenariusze wywołań innych usług (Web Service) zdefiniowane przy wykorzystaniu języka BPEL. Scenariusze te są wykonywane następnie w ramach silnika Apache Ode.

Sposób analizy danych kontekstowych został zbudowany na podstawie zmodyfikowanej sieci Petrie'go (patrz rozdział III.4). Został on zobrazowany na rysunku 30.



Rysunek 30 Zasada działania mechanizmu analizy kontekstu

Proces analizy danych kontekstowych rozpoczyna się od przekazania przez SG definicji oczekiwanych warunków wywołania akcji. Na ich podstawie automatycznie tworzona jest zmodyfikowana sieć Petrie'go (z adnotacjami). Sieć ta składa się z dwóch warstw: pierwsza odpowiada za generowanie tokenów, natomiast druga warstwa odpowiednio przesyła je przez sieć. Każdy z tokenów posiada pewien czas ważności (patrz rozdział IV). Analiza danych kontekstowych w sieci jest zrównoleglona – tzn. każda tranzycja wykonywana jest w sposób równoległy. Każda z nich posiada kolejki wejściowe, do których zapisywane są tokeny (będące wynikiem działania poprzedzającej tranzycji bądź dostarczenia danych kontekstowych). Podczas dostarczenia danej kontekstowej do sieci następuje jej podział na części przypisane do zmiennych (na przykład dane o sali seminaryjnej, są dzielone na listę osób w sali, nazwę sali, poziom naświetlenia itd.) w zależności od definicji przekazanej przez projektanta (patrz listing 6). Zmienne te przekazywane są do pierwszej warstwy, skąd rozpoczyna się sprawdzenie czy spełniają oczekiwane warunki wywołania akcji (wyrażone przy pomocy relacji). Warunki te mogą zawierać odwołania do usług bazowych (o których informacje oraz sposób wywołania zawarte są w bazie wiedzy). Jeżeli wynikiem zwróconym przez warunek jest prawda to zmienne analizowane przez ten warunek zamieniane są na token i trafiają do drugiej

warstwy. W niej następuje sprawdzenie warunków logicznych (*and* i *or*) – zgodnie z definicją relacji (wynikającą z oczekiwanych warunków wywołania akcji) i grup (aby można było określić które zmienne muszą pochodzić z tych samych odczytów). Jeżeli również te warunki są pełnione, token przepisywany jest na kolejną tranzycję z drugiej warstwy (co odpowiada sprawdzeniu kolejnego warunku logicznego). W momencie, gdy jest to ostatni warunek (decydujący o tym czy oczekiwane warunki wywołania akcji zostały spełnione) następuje wysłanie sygnału (patrz rozdział V.1) do silnika głównego SG wraz z kontekstem aplikacji zawierającym kontekst akcji).

VI.3. Problemy konstrukcyjno-implementacyjne

Podstawowym problemem jaki należało rozwiązać podczas implementacji SWIAI było zbudowanie mechanizmu komunikacji z usługami sieciowymi (Web Service), dołączanymi w czasie działania środowiska. Mechanizm ten opiera się na obiekcie Proxy (patrz rozdział VI.2) tworzonym (tzn. dynamicznie kompilowanym) na podstawie opisu WSDL (patrz listing 7).

```
    public class Proxy {
     public object invokeMethod (String methodName, object[]
args) {
          //wywołanie metody
     }
    }

    Proxy createProxy (String WSDLUrl, String serviceName) {
     String WSDLContent = download (WSDLUrl);     //pobranie
                                                  opisu        WSDL
                                                  usługi
     Object o = compileWSDL(WSDLContent);         //dynamiczna
                                                  kompilacja  opisu
                                                  WSDL
     Type t = o.GetType();
     return t.getServiceByName (serviceName);     //zwrócenie
                                                  obiektu      typu
                                                  Proxy
                                                  umożliwiającego
                                                  wywoływanie
                                                  metod usługi
    }
```

Listing 7 Zasada działania dynamicznego wywoływania usług sieciowych

Dla każdej dodawanej do SWIAI usługi sieciowej (w czasie jej rejestracji) tworzony jest nowy obiekt Proxy, umożliwiający komunikację z usługą. Proces rejestracji usługi w SWIAI został przedstawiony na listingu 8. Przedstawia on operacje wykonywane w komponencie ŚUB.

```
public class Service {
 Proxy proxy;
 IWrapper wrapper;
 public Service (Proxy proxy, IWrapper wrapper) {
      this.proxy = proxy;
      this.wrapper = wrapper;
 }
}

Service registerService (String serviceDescription,
String WSDLUrl, String serviceName) {
 Proxy proxy = createProxy (WSDLUrl, serviceName);//
                                    utworzenie obiektu
                                    umożliwiającego
                                    wywoływanie metod
                                    dynamicznie
                                    skompilowanej usługi
 ServiceModel                  model                =
BW.registerService(serviceDescription, WSDLUrl, serviceName);
                                    //rejestracja w BW
                                    Nowej usługi i
                                    otrzymanie od BW
                                    ustandaryzowanego jej
                                    opisu
 IWrapper wrapper = null;
 if (model.isNative)                //sprawdzenie czy
 usługa
                                        jest dedykowana
                                    do współpracy z SWIAI
      wrapper = new PassiveWrapper (model);  //
                                    utworzenie wrapper'a
                                    pasywnego dla usług
                                    dedykowanych
 else
      wrapper = new ActiveWrapper (model);   //
                                    utworzenie wrapper'a
```

```
                                        aktywnego dla usług
                                        dodatkowych
 return new Service (proxy, wrapper);
}
```

Listing 8 Rejestracja usługi w SWIAI

W czasie rejestracji usługi przekazywane są informacje na temat jej opisu (którego przykład został przedstawiony w listingu 5), lokalizacji dokumentu XML zawierającego opis WSDL oraz nazwy usługi. Po utworzeniu obiektu `Proxy` wykonywana jest rejestracja usługi w komponencie BW. Przetwarza on dostarczony opis, zapisuje go w bazie wiedzy (tworząc semantyczny rejestr usług) i zwraca do ŚUB jej ustandaryzowany opis (w postaci obiektu `ServiceModel`). Opis ten zawiera informacje na temat sposobu transformacji danych wymienianych z usługą oraz czy jest ona dedykowana do współpracy z SWIAI. W zależności od tego czy usługa jest dedykowana czy nie (patrz rozdział VI.1.2) tworzone są odpowiednie Wrapper'y, które wraz z obiektem `Proxy` umożliwiają wykorzystanie usługi w środowisku (podczas działania aplikacji).

Duża liczba danych kontekstowych dostarczanych w czasie działania SWIAI powoduje konieczność wielu jednoczesnych wywołań usług (zarówno bazowych jak i odpowiadających komponentom IAK). Ich uruchamianie w sposób kaskadowy (jednej usługi w ramach innej) jest niemożliwe ze względu na konieczność utrzymywania dużej liczby połączeń TCP. W związku z tym podczas wywoływania metod pomiędzy komponentami (szczególnie przy wymianie danych pomiędzy ŚUB, SG i KAK) należało zapewnić, aby możliwie szybko kończyły one swoje działanie bez oczekiwania na zakończenie instrukcji wynikających z maszyny PIAI (patrz rozdział V.3.1). W przeciwnym wypadku powodowało by to, iż metoda, od której rozpoczęła się komunikacja musi oczekiwać na zakończenie (utrzymując połączenie TCP) do momentu, aż wszystkie te instrukcje zostaną wykonane. Tego problemu uniknięto poprzez uruchomienie dodatkowych wątków przejmujących dane od wywoływanych metod i osobno te dane przetwarzających. Na listingu 9 przedstawiono zasadę działania mechanizmu kolejkowania danych.

```
public class QueueThread : Thread {
 private bool stop = false;
 private Blocker blocker;
 private Processor processor;
 private List<object> queue;

 public QueueThread() {
```

```
        this.Start();                   //uruchomienie wątku
                                        obsługi kolejki
    }

    public void insertData (object o) {
        queue.Add(o);                   //dodanie do kolejki
                                        danej wejściowej
        blocker.Set();                  //odblokowanie
                                        pobierania danych z
                                        kolejki do analizy i
                                        przetwarzania
    }

    public void run () {
        while (!stop) {                 //sprawdzenie warunku
                                        czy zakończyć
                                        działanie wątku
            blocker.Wait();             //oczekiwanie na
                                        odblokowanie
                                        pobierania danych z
                                        kolejki do analizy i
                                        przetwarzania
            while (queue.Count > 0) {   //dla wszystkich
                                        obiektów
                object o = queue[0];    //z kolejki
                                        uruchomienie
                queue.Remove(o);        //ich analizy i
                processor.Analyze(o);   //przetwarzania
            }
        }
    }
}
```

Listing 9 Kolejkowanie danych wymienianych pomiędzy komponentami SWIAI

Wątek (którego działanie reprezentowane jest przez metodę `run()`) zaczyna działać w momencie utworzenia obiektu (`this.Start()`) i jest wykonywana dopóki nie została ustawiona flaga oznaczająca zakończenie działania wątku. Oczekuje na obiekcie `blocker` (`blocker.Wait()`) do momentu dostarczenia nowych danych. Metoda wywołująca ogranicza się jedynie do umieszczenia obiektu z danymi w kolejce wejściowej (`queue`) i odblokowania (poprzez wykonane instrukcji `blocker.Set()`) działania

wątku. Wtedy, dla wszystkich danych z kolejki wejściowej, uruchamiane jest działanie reprezentowane przez metodę `processor.Analyze(o)` i dane są usuwane z kolejki.

Przestrzeń inteligentna zawiera środowisko wykonywania aplikacji IAI zbudowane zgodnie z architekturą IAK. W celu jego implementacji konieczne jest opracowanie komponentów i mechanizmu komunikacji (pomiędzy nimi - w ramach IAK jak i z usługami w przestrzeni). W poszczególnych komponentach wykonywane są operacje związane z procedurą analizy kontekstu uwzględniające transformację danych kontekstowych. W tabeli 9 zostały zebrane aspekty implementacji architektury IAK oraz dotyczące ich sposoby realizacji, jakie zostały opracowane w ramach tego rozdziału.

Tabela 9 Aspekty implementacji architektury IAK

Aspekt	Sposób realizacji
Komponenty	Komponent Analizy Kontekstu, Komponent Wykonania Akcji, Silnik Główny, Środowisko Usług Bazowych, Baza Wiedzy
Mechanizm komunikacji	Mechanizm sesji, Kolejkowanie danych, Komunikaty (sygnały), Komunikacja z usługami przez Proxy, Wykorzystanie modelu usługi
Procedury	Analiza kontekstu uwzględniająca transformacje danych
Formaty	Definicja pary aplikacji IAI, Model usługi
Wymagane typy usług w przestrzeni	Usługi (dedykowane i dodatkowe) dostarczające dane kontekstowe, Usługi umożliwiające wykonywanie akcji, Usługi wspomagające analizę kontekstu

Zatem maszyna PIAI (i wynikająca z niej architektura IAK) stanowi bazę do budowy przestrzeni inteligentnej, w której znajduje się środowisko SWIAI z wydzielonym zbiorem instrukcji i usług wspomagających działalność tej przestrzeni, co stanowi wykazanie drugiej tezy rozprawy doktorskiej. Mając przygotowaną implementację przestrzeni można wykonać założone badania eksperymentalne, które zostały opisane w następnym rozdziale.

VII PRZEPROWADZENIE BADAŃ EKSPERYMENTALNYCH

Przedstawiono eksperymenty, jakie zostały przeprowadzone w celu zweryfikowania czy efektywność działania (czas reakcji) SWIAI zależy od wielkości kontekstu, rozproszenia jego analizy pomiędzy maszynę lub usługi, a także od liczby zdarzeń jednocześnie zachodzących w przestrzeni. W tym celu wykorzystano środowisko wykonywania aplikacji IAI opisane w rozdziale VI. Przedstawione zostały założenia, dotyczące poszczególnych eksperymentów, sposób ich wykonania oraz zaprezentowano i przeanalizowano otrzymane w skutek ich przeprowadzenia wyniki.

VII.1. Założenia dotyczące eksperymentów

Przedmiot badań

Przedmiotem badań była efektywność działania środowiska SWIAI i jej zmiany w zależności od wielkości kontekstu, rozproszenia jego analizy pomiędzy maszynę i usługi, a także od liczby zdarzeń jednocześnie zachodzących w przestrzeni. Efektywność działania rozumiana jako czas reakcji typu T_{PIAI} - patrz rozdział V.3. Została ona wybrana do badań ponieważ umożliwia (w przeciwieństwie do pozostałych zdefiniowanych czasów reakcji) uniezależnienie ich wyników od usług znajdujących się w przestrzeni inteligentnej.

Wykorzystanie usług

Przeprowadzone eksperymenty nie dotyczyły badania szybkości działania usług (opakowujących sensory oraz koniecznych do wykonywania akcji) znajdujących się w dedykowanej przestrzeni usług. Mimo, że mają one wpływ na efektywność wykonania aplikacji IAI to z punktu widzenia użytkownika (efektywność wyrażoną poprzez czas reakcji T_Z - patrz rozdział V.3), są one zewnętrzne w stosunku do maszyny PIAI a przez to także implementacji SWIAI i w związku z tym nie mają wpływu na ich właściwości. Dodatkowo, usługi te mogą się różnić pomiędzy dziedzinami zastosowań przestrzeni inteligentnych. Sposób działania usług opakowujących sensory zależy od technologii, w jakiej te sensory są zbudowane, ich zasobów sprzętowych oraz zasad działania. Podobnie akcje mogą składać się z uruchomienia pojedynczej usługi bądź wielu usług (w postaci scenariusza). W implementacji SWIAI przyjęto, iż akcje są usługami, które wykonują scenariusz BPEL składający się z wywołania pojedynczej usługi. Są to aspekty, na które architektura IAK nie ma wpływu.

Dedykowana przestrzeń usług

Aplikacje IAI, jakie posłużyły do przeprowadzenia badań, wykonywane były w ramach środowiska SWIAI zaimplementowanego zgodnie z architekturą IAK, które są opisane w rozdziale VI. Środowisko to zostało umieszczone w dedykowanej przestrzeni usług (patrz rozdział V), w obrębie której funkcjonowały następujące usługi (patrz rozdział IV.3):

- dedykowane do pracy z SWIAI usługi opakowujące sensory, dostarczające dane odczytane przez sensory. Usługi te dostarczały do SWIAI dane kontekstowe związane ze zdarzeniem zachodzącym w przestrzeni, dokładnie w momencie jego pojawienia się. W rozpatrywanym przykładzie obejmowały one: usługę odczytującą poziom natężenia głosu osób prowadzących rozmowy w sali seminaryjnej oraz usługę reagującą na zakończenie seminarium,
- usługi potrzebne do wykonania akcji: dla rozpatrywanego przykładu były to usługa ściszająca muzykę odgrywaną przez system nagłośnienia oraz usługa włączająca odgrywanie muzyki i zmieniająca wartość parametru określającego czy w sali trwają zajęcia.

Zgodnie z rozdziałem IV.3 w dedykowanej przestrzeni usług znajdują się także usługi, które mogą być wykorzystywane do analizy kontekstu. Jednakże, aby uniezależnić badania od usług znajdujących się w przestrzeni, wszystkie operacje konieczne do analizy kontekstu wykonywane są przez SWIAI.

Analiza kontekstu

Analizowanie kontekstu w SWIAI jest oparte na zmodyfikowanej sieci Petrie'go, jednakże zastosowane w tym celu podejście może się różnić pomiędzy różnymi implementacjami architektury IAK. Dotyczy to zwłaszcza charakterystyk czasowych takich jak czas sprawdzenia pojedynczego warunku zawartego w definicji aplikacji, czy czas analizy pojedynczej danej kontekstowej. Jednakże, ponieważ analizowanie kontekstu jest kluczową funkcją maszyny PIAI to czas jej wykonania został wzięty pod uwagę podczas pomiarów wykonywanych w czasie eksperymentów (zgodnie z definicją T_{PIAI}).

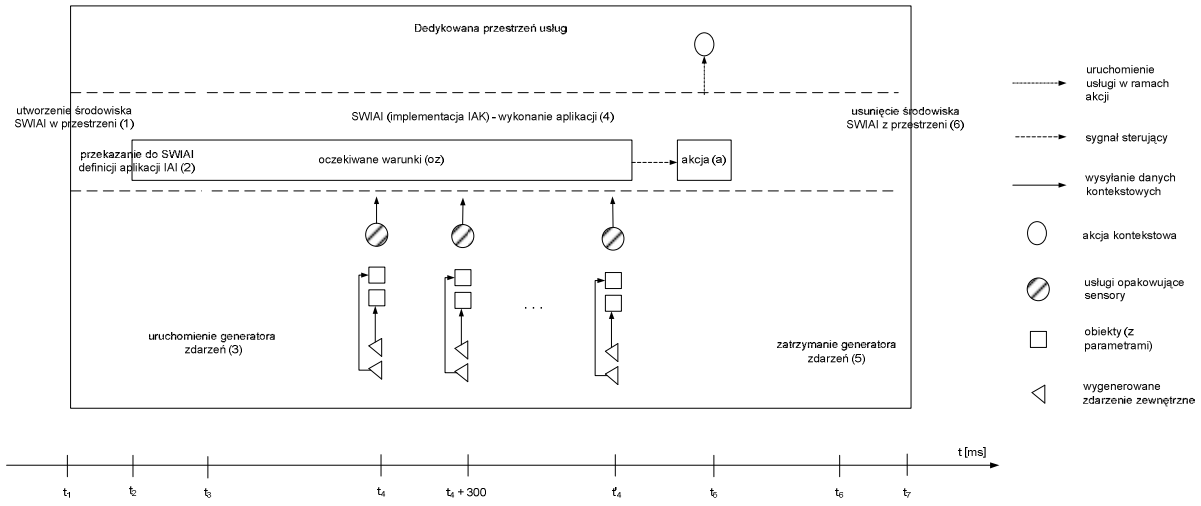
Zdarzenia w przestrzeni

Do przeprowadzenia eksperymentów został wykorzystany generator zdarzeń zewnętrznych (które zostały zdefiniowane w rozdziale IV.3), umożliwiający generowanie wielu jednoczesnych zdarzeń w różnych momentach czasu. Podczas badań odstęp między zdarzeniami był stały i wynosił 300 ms (to znaczy co 300 ms generator generował

określoną liczbę zdarzeń zewnętrznych – zależną od prowadzonego eksperymentu, patrz rozdział VII.2). Przyjęcie tej wartości gwarantowało, iż wykonywanie instrukcji maszyny odnoszących się do analizy danej kontekstowej związanej z pojawieniem się zdarzenia zostanie zakończone przed otrzymaniem następnych danych od usług opakowujących sensory (związanych z kolejnymi wygenerowanymi zdarzeniami). Dzięki temu otrzymane wyniki są porównywalne. Każde wygenerowane zdarzenie powodowało zmianę wartości pojedynczego parametru obiektu znajdującego się w przestrzeni i wchodzącego w skład kontekstu badanej aplikacji. Liczba jednocześnie generowanych zdarzeń była ustalana osobno dla każdego eksperymentu (jej wartości zostały podane w dalszej części rozdziału).

VII.2. Schemat badań

Schemat przeprowadzonych badań został przedstawiony na rysunku 31.



Rysunek 31 Zasada przeprowadzenia pojedynczego badania dla dwóch jednoczesnych zdarzeń

Każde badanie, było powtarzane dla różnych wartości wielkości kontekstu, różnego poziomu rozproszenia logiki analizy kontekstu oraz dla różnej liczby jednoczesnych zdarzeń pojawiających się w przestrzeni i składało się z następujących kroków:

(1) utworzenie środowiska SWIAI w przestrzeni inteligentnej, w której znajdują się usługi opakowujące sensory i usługi umożliwiające wykonanie akcji (są to usługi SOA zgodne ze standardem Web Service – patrz rozdział VI.2) – dzięki temu przestrzeń staje się dedykowaną przestrzenią usług dla aplikacji IAI,
(2) przekazanie do SWIAI definicji aplikacji IAI, co powoduje utworzenie obiektów umożliwiających analizę kontekstu i wykonanie akcji w komponentach IAK (patrz

rozdział VI),
(3) uruchomienie generatora zdarzeń zewnętrznych,
(4) wykonanie aplikacji IAI polegające na odbieraniu danych kontekstowych oraz wykonywaniu poszczególnych instrukcji maszyny PIAI zdefiniowanych w rozdziale V.3. Wykonanie aplikacji kończy się w momencie, gdy zakończy się wykonywanie akcji związanych z wszystkimi oczekiwanymi warunkami wywołania akcji zawartymi w przekazanej definicji aplikacji,
(5) zatrzymanie generatora zdarzeń,
(6) usunięcie komponentów środowiska SWIAI z przestrzeni.

W momencie zajścia zdarzenia zewnętrznego w przestrzeni, usługa opakowująca sensor wysyła do SWIAI informację na temat nowej wartości parametru obiektu. Jeżeli w przestrzeni zaszło jednocześnie więcej niż jedno zdarzenie, informacje dotyczące wszystkich parametrów przesyłane są w jednym komunikacie. Pomiar czasu, obejmujący wszystkie instrukcje wchodzące w skład analizy danej kontekstowej w ramach SWIAI (patrz rozdział V.3.1), był zapisywany w obiekcie monitorującym z dokładnością do nanosekundy. Wartości przedziałów czasów wykonania instrukcji maszyny PIAI uzyskane dla różnych zdarzeń w ramach pojedynczego eksperymentu były uśredniane. W przypadku, gdy wielkość kontekstu nie była podzielna przez przyjętą do danego eksperymentu liczbę jednoczesnych zdarzeń, wtedy ostatni sensor wysyłał tylko tyle danych kontekstowych ile wynosiła wielkość kontekstu modulo liczba jednoczesnych zdarzeń.

Dane kontekstowe, jakie docierają do środowiska i pozostają w procesie analizowania kontekstu (po wykonaniu wszystkich akcji zdefiniowanych w aplikacji), mogą wpływać na szybkość rozpoznania czy zostały spełnione inne oczekiwane warunki wywołania akcji należące do innej aplikacji (wykonywanej na przykład podczas kolejnego badania). Z tego powodu zdecydowano, że po zakończeniu wykonywania aplikacji IAI (tzn. wszystkich jej akcji) środowisko SWIAI będzie usuwane z przestrzeni (tzn. wszystkie jej komponenty są zatrzymywane i usuwane z serwera aplikacji). Dzięki temu uzyskano powtarzalność eksperymentów.

Scenariusze badań

Badane aplikacje składały się z jednej bądź dwóch par: oczekiwane warunki wywołania akcji – akcja (tzn. $\mu = 1$ lub $\mu = 2$) – patrz rozdział IV.4 i dotyczyły przykładowej aplikacji opisanej w rozdziale IV.1:

- aplikacja 1 ($\mu = 1$) - reagowała na zwiększenie poziomu natężenia głosu osób

prowadzących rozmowy w sali seminaryjnej (poprzez ściszenie muzyki),

- aplikacja 2 ($\mu = 2$) - reagowała na zwiększenie poziomu natężenia głosu osób prowadzących rozmowy w sali seminaryjnej (poprzez ściszenie muzyki) oraz dodatkowo na zakończenie seminarium (poprzez włączenie odgrywania muzyki i zmianę parametru określającego czy w sali trwają zajęcia).

Do eksperymentów wybrano te pary, ponieważ związane z nimi oczekiwane warunki mogą zostać spełnione w krótkich odstępach czasu (na potrzeby badań przyjęto, że w tym samym momencie czasu), co pozwala obserwować równoległe analizowanie kontekstu związanego z oczekiwanymi warunkami wykonania akcji oraz wykonanie akcji (należących do różnych par w definicji aplikacji).

Scenariusz 1

W ramach pierwszego scenariusza wykonywano badania w odniesieniu do aplikacji 1 jak i aplikacji 2, w czasie których zmieniała się wielkość kontekstu ψ (od 10 do 200 obiektów - co 10) oraz liczba jednocześnie pojawiających się w przestrzeni zdarzeń f (od 1 do 50 co 1). Każdy obiekt należący do kontekstu miał przypisany jeden parametr, wykorzystany następnie w oczekiwanych warunkach wywołania akcji. Każdy z tych parametrów opisywał inny rodzaj poziomu natężenia głosu rozmówców (np. ze względu na umiejscowienie sensora) bądź inny rodzaj sygnału o zakończeniu seminarium jaki był odczytywany przez usługi opakowujące sensory. Aby zbadać czas reakcji SWIAI dla większych kontekstów, ich interpretacje były teoretyczne (np. 200 parametrów opisujących różne rodzaje poziomu natężenia głosu). Dla różnych wielkości kontekstu dokonano następnie pomiarów czasu reakcji w zależności od liczby jednocześnie pojawiających się w przestrzeni zdarzeń, powodujących zmianę poziomu natężenia głosu rozmówców lub wygenerowanie sygnału o zakończeniu seminarium (powodującego zmianę wartości parametru sali). W czasie prowadzenia badań zgodnie ze scenariuszem 1 kontekst był analizowany w całości przez SWIAI (tzn. stopień rozproszenia kontekstu $r = 0\%$).

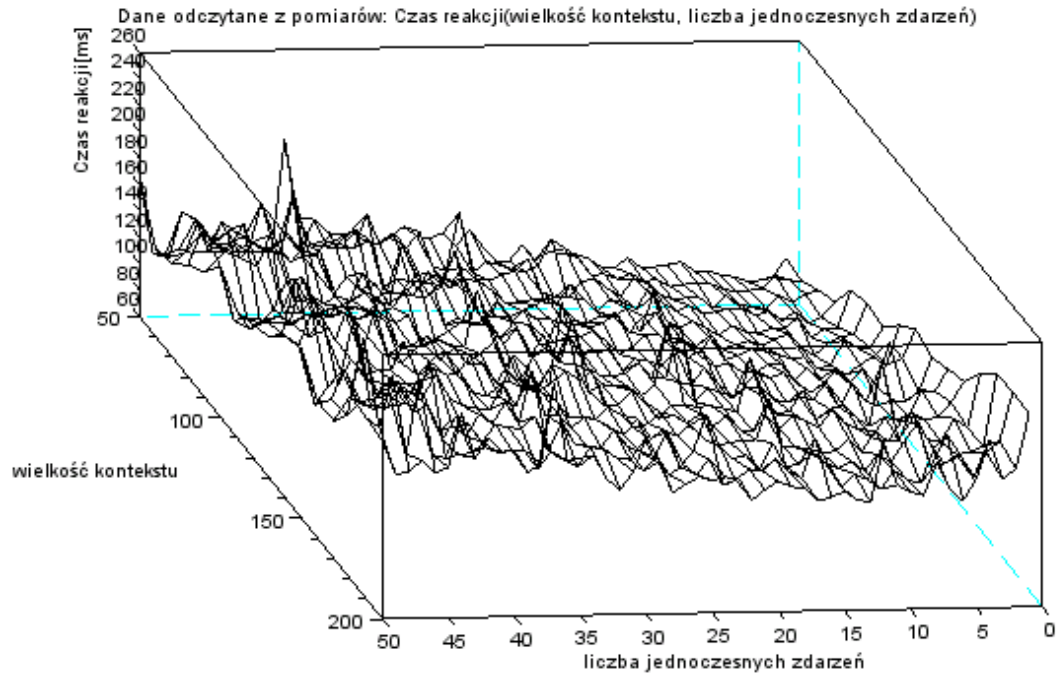
Badania prowadzone w ramach pierwszego scenariusza pozwoliły również stwierdzić, która z instrukcji zdefiniowanych w ramach PIAI (patrz rozdział V.3.1) jest najbardziej znacząca dla czasu działania maszyny (tzn. czas jej wykonania w największym stopniu wpływa na jej czas reakcji). W tym celu dokonywano pomiaru momentu rozpoczęcia i zakończenia wykonywania poszczególnych instrukcji, uzyskując tym samym przedział czasu ich wykonania. Wynik następnie uśredniono dla wszystkich pomiarów.

Scenariusz 2

Drugi scenariusz badań odnosił się do aplikacji 1 i pozwolił na określenie czy czas reakcji środowiska SWIAI zależy od stopnia rozproszenia logiki pomiędzy środowisko wykonywania aplikacji IAI a usługi SOA znajdujące się w przestrzeni usług dedykowanych (r). W tym celu wykonano badania dla różnych wielkości kontekstu (od 10 do 100) przy stałej wartości liczby jednoczesnych zdarzeń ($f = 2$) i poziomu rozproszenia logiki (0%) oraz badania dla różnych poziomów rozproszenia (od 10% do 100% - co 10%) przy stałej wielkości kontekstu ($\psi = 100$) i stałej liczbie jednoczesnych zdarzeń ($f = 2$). Dla każdej wartości wielkości kontekstu oraz poziomu rozproszenia logiki analizy kontekstu badanie powtórzono 100 razy i następnie uśredniono wynik.

VII.3. Pomiary i analiza wyników eksperymentów

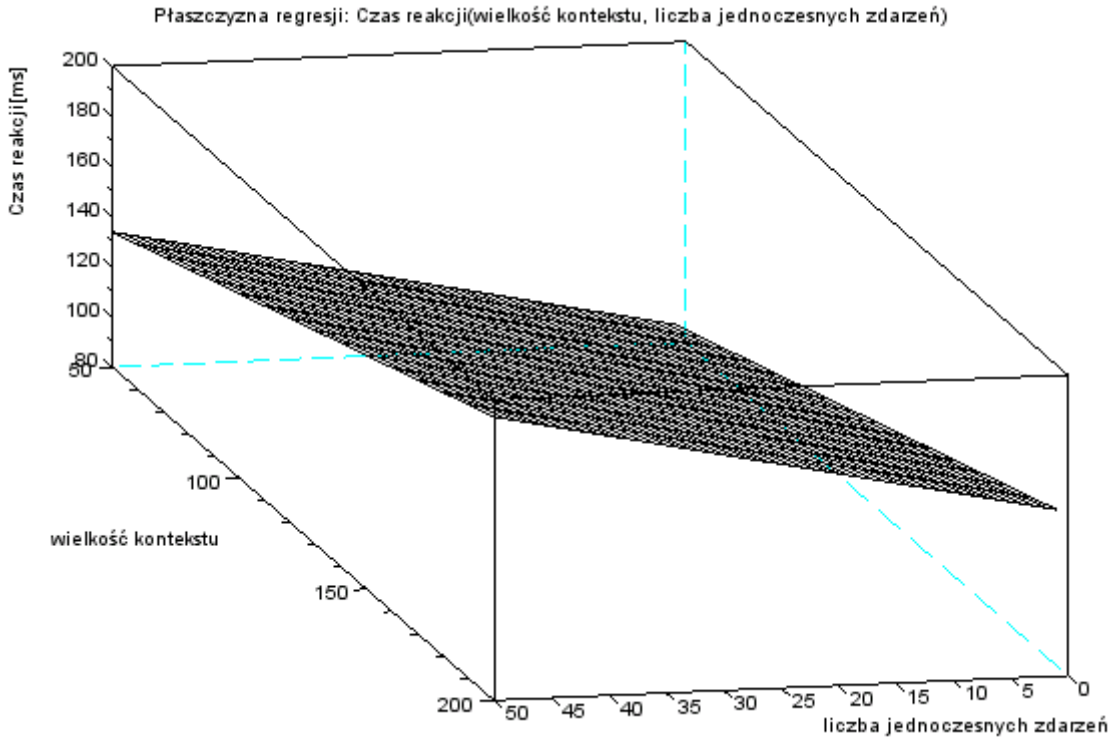
W pierwszej kolejności dokonano pomiarów zgodnie ze scenariuszem 1 dla aplikacji 1 ($\mu = 1$). Pomiary dotyczyły czasu reakcji typu T_{PIAI} wykonywanej w ramach SWIAI aplikacji dla różnych wielkości kontekstu (ψ) oraz liczby jednocześnie pojawiających się w przestrzeni zdarzeń (f). Wyniki zostały zobrazowane na rysunku 32.



Rysunek 32 Czas reakcji SWIAI w zależności od wielkości kontekstu aplikacji i liczby jednoczesnych zdarzeń dla aplikacji 1

Czas reakcji był mierzony w nanosekundach. Aby lepiej zobrazować wyniki, poszczególne punkty reprezentujące wyniki pomiarów połączono liniami. Ze względu na wykonywanie

pomiarów dla 50 jednoczesnych zdarzeń wyniki przedstawiono dla wielkości kontekstu większej bądź równej 50. Jest to konieczne, aby dane kontekstowe związane ze wszystkimi zdarzeniami były przeanalizowane w czasie sprawdzania oczekiwanych warunków wywołania akcji. Dla danych zobrazowanych na powyższym wykresie wyznaczono (metodą regresji) płaszczyznę trendu. Została ona przedstawiona na rysunku 33.



Rysunek 33 Płaszczyzna regresji dla zależności czasu reakcji od wielkości kontekstu i liczby jednoczesnych zdarzeń dla aplikacji 1

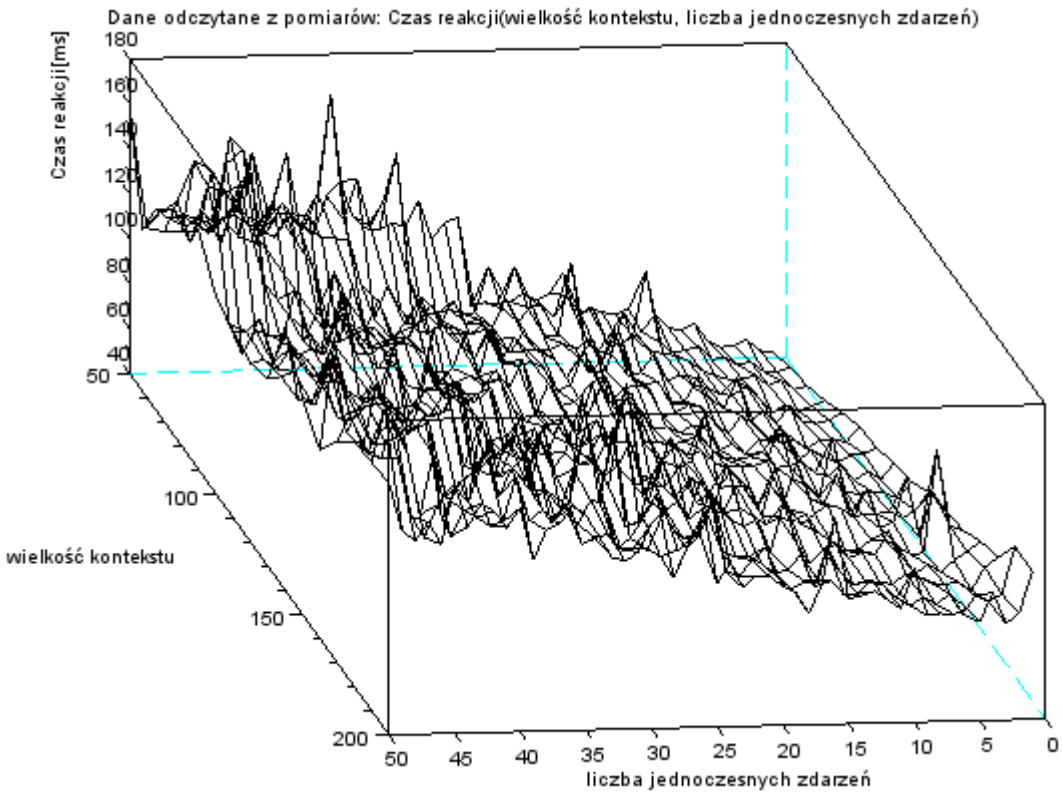
Współczynniki opisujące płaszczyznę wyznaczono przy wykorzystaniu metody regresji liniowej (wykorzystującej metodę najmniejszych kwadratów). Płaszczyzna ta jest opisana wzorem:

$$\pi(f,\psi) = 0.950052 * f + 0.3950633 * \psi + 66.579846$$

Uzyskane wartości współczynników są większe od 0 zatem płaszczyzna jest pochylona (w górę) w obu kierunkach (f i ψ). Można zatem stwierdzić, iż dla przygotowanej implementacji SWIAI aplikacje posiadające stopień interaktywności $\mu = 1$ wykonują się coraz wolniej (spada efektywność ich wykonania – tzn. rośnie czas reakcji typu T_{PIAI}) wraz ze wzrostem wielkości kontekstu oraz liczby jednoczesnych zdarzeń.

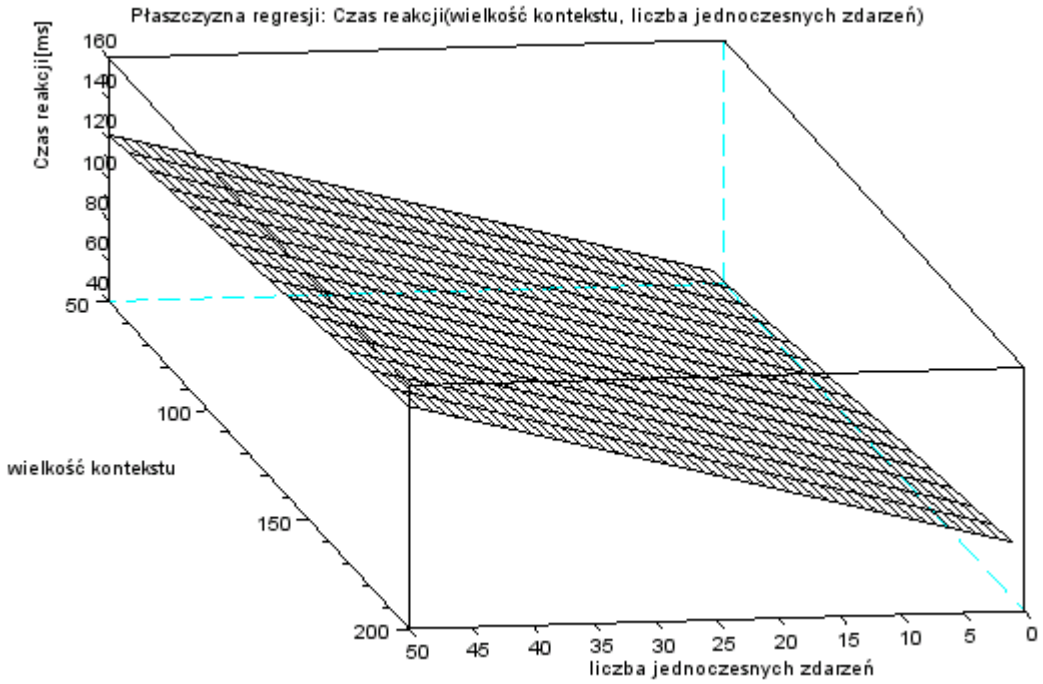
Analogiczne pomiary zostały powtórzone dla aplikacji 2 ($\mu = 2$). Dane odczytane

z pomiarów zostały uśrednione (dla tych samych wartości wielkości kontekstu i liczby jednoczesnych zdarzeń) i są przedstawione na rysunku 34.



Rysunek 34 Czas reakcji SWIAI w zależności od wielkości kontekstu aplikacji i liczby jednoczesnych zdarzeń dla aplikacji 2

Podobnie jak w przypadku aplikacji 1 wielkość kontekstu (ψ) ograniczono do przedziału od 50 do 100, a poszczególne punkty reprezentujące wyniki pomiarów połączono liniami. Zmniejszenie czasu reakcji zarówno w kierunku osi f jak i ψ widać po wyznaczeniu płaszczyzny regresji dla danych zebranych w czasie pomiarów. Płaszczyzna ta została przedstawiona na rysunku 35.



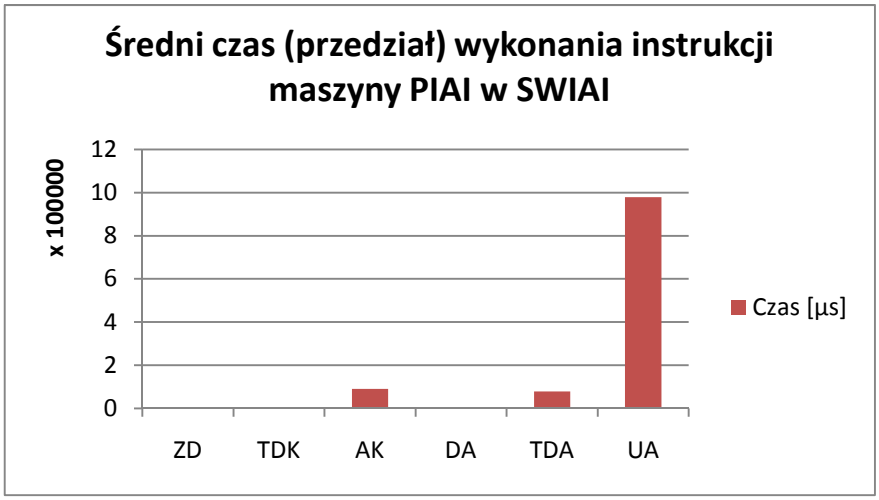
Rysunek 35 Płaszczyzna regresji dla zależności czasu reakcji od wielkości kontekstu i liczby jednoczesnych zdarzeń dla aplikacji 2

Przedstawiona płaszczyzna opisana jest następującym wzorem:

$$\pi(f,\psi) = 1.5401152 * f + 0.184849 * \psi + 35.812873$$

Podobnie jak dla płaszczyzny regresji (wyznaczonej poprzez metodę najmniejszych kwadratów tzn. płaszczyzny dla której suma kwadratów jej odległości od wszystkich punktów przedstawiających wyniki pomiarów jest najmniejsza) utworzonej dla pierwszej aplikacji IAI (ze stopniem interaktywności równym 1) unosi się ona w kierunku osi f oraz ψ. Analogicznie jak w przypadku wyników pomiarów, płaszczyzny regresji dla obu aplikacji mogą być interpretowane jedynie dla dyskretnych wartości f oraz ψ.

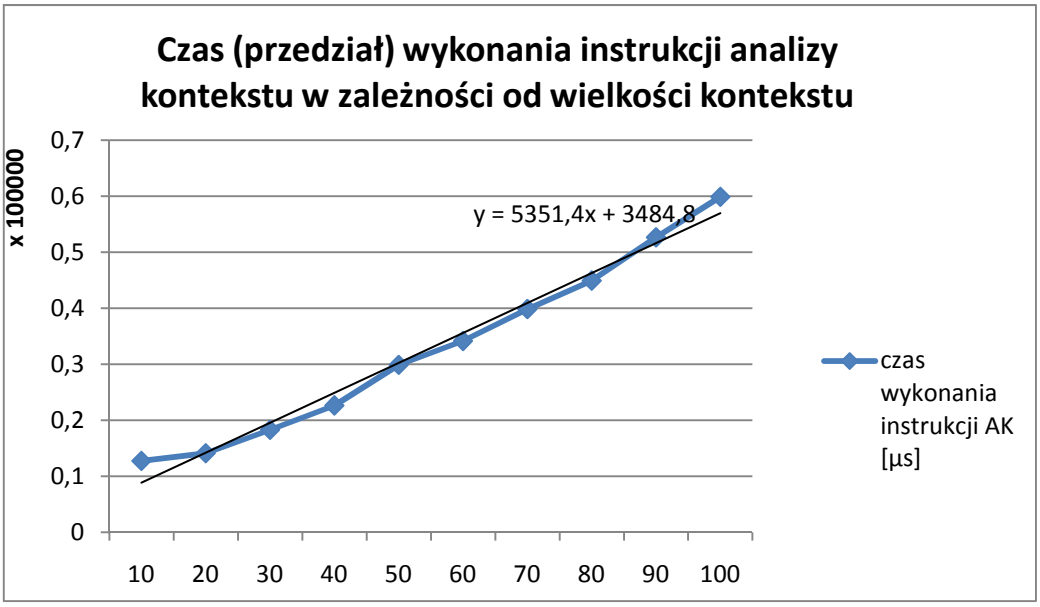
Podczas badań przeanalizowano także, które z instrukcji maszyny PIAI wykonywane w czasie analizy pojedynczej danej kontekstowej (zdefiniowane w rozdziale V.3.1) najbardziej wpływają na proces analizy kontekstu i co za tym idzie na efektywność wykonania aplikacji IAI. Wyniki zostały przedstawione na rysunku 36.



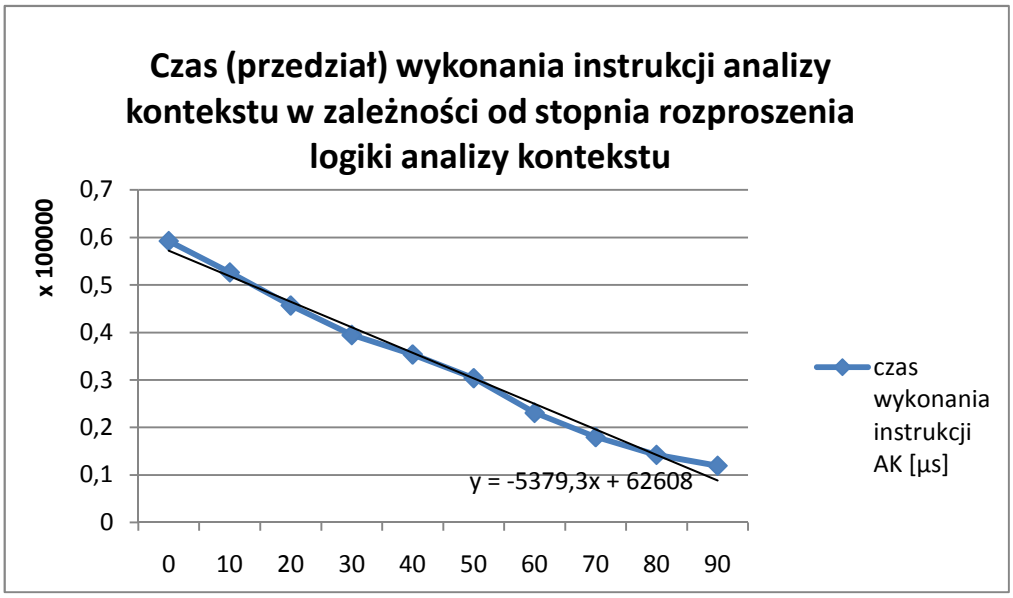
Rysunek 36 Średni czas (przedział) wykonania instrukcji PIAI w ramach SWIAI

Pomierzone zostały przedziały czasu wykonania każdej instrukcji PIAI związanej z analizą danej kontekstowej (zdefiniowane w rozdziale V.3.1). Pomiar czasu został dokonany przed rozpoczęciem wykonania każdej z instrukcji i zaraz po jej zakończeniu przy wykorzystaniu obiektu monitorującego. Można zauważyć, iż czasy związane z instrukcjami ι_{ZD}, ι_{TDK} oraz ι_{DA} są pomijalnie małe (na wykresie przedstawione są jako wartość 0, co wynika z tego, iż miały one wartość poniżej 100 milisekund). Największy wpływ na czas reakcji SWIAI miały składowe analizy kontekstu (instrukcja ι_{AK}) oraz w drugiej kolejności uruchomienia akcji (instrukcja ι_{UA}) i transformacji danych dla akcji (instrukcja ι_{TDA}).

W czasie przeprowadzania badania opisanego w scenariuszu 2 (którego wynikiem jest określenie czy rozproszenie logiki analizy kontekstu pomiędzy SWIAI i usługi opakowujące sensory także wpływa na czas reakcji) analizowany był przedział czasu wykonania instrukcji ι_{AK}. Zgodnie z rysunkiem 36 jest to jedna z trzech instrukcji, których przedziały czasów wykonania najbardziej wpływają na czas reakcji T_{PIAI}, jednakże na wykonanie instrukcji ι_{TDA} i ι_{UA} nie wpływa stopień rozproszenia logiki analizy kontekstu (patrz rysunek 24). Wyniki pomiarów zostały zaprezentowane na rysunkach 37 i 38.



Rysunek 37 Czas (przedział) wykonania instrukcji analizy kontekstu w zależności od wielkości kontekstu



Rysunek 38 Czas (przedział) wykonania instrukcji analizy kontekstu w zależności od stopnia rozproszenia logiki analizy kontekstu

Wyniki uzyskane w trakcie pomiarów świadczą o tym, iż zwiększenie stopnia rozproszenia logiki analizy kontekstu przekłada się na odpowiednie zmniejszenie wielkości kontekstu jaki jest analizowany w SWIAI (tzn. przez maszynę PIAI). Oba wykresy przedstawiają

zależności jakie można opisać funkcjami liniowymi, które posiadają przybliżone (co do wartości bezwzględnej) współczynniki kierunkowe różniące się znakami (dla wykresu z rysunku 38 jest to 5351,4 natomiast dla wykresu z rysunku 39 jest to -5379,3). W celu dalszego porównania wyników dotyczących pomiarów uzyskanych w odniesieniu do scenariusza 2 zestawiono wartości przedziałów czasu (w mikrosekundach) wykonania instrukcji ι_{AK} w tabeli 10.

Tabela 10 Zestawienie wyników pomiarów dotyczących scenariusza 2

	ψ=10, r=90%	ψ=20, r=80%	ψ=30, r=70%	ψ=40, r=60%	ψ=50, r=50%
ψ	12750,578	14118,514	18295,474	22648,195	29884,858
r	11960,929	14247,304	17966,014	23097,947	30404,501
Δ=| ψ – r |	789,64871	128,79028	329,46009	449,75256	519,64286
Δ/((ψ + r)/2)	**3%**	**0%**	**1%**	**1%**	**1%**
	ψ=60, r=40%	ψ=70, r=30%	ψ=80, r=20%	ψ=90, r=10%	ψ=100, r=0%
ψ	34156,409	39856,704	44934,43	52655,087	59877,147
r	35395,249	39497,214	45710,949	52665,713	59266,455
Δ=| ψ – r |	1238,8398	359,48962	776,51962	10,625849	610,69178
Δ/((ψ + r)/2)	**2%**	**0%**	**1%**	**0%**	**1%**

Wartości wielkości kontekstu i stopnia rozproszenia logiki jego analizy, dla których podano wartości przedziałów czasu (w kolumnach), wynikają ze wzoru (5.5). Dodatkowo, wyliczono wartość bezwzględną różnicy tych wartości oraz jakim jest ona procentem w stosunku do wartości ich średniej. Nie przekracza ona 3%, co stanowi potwierdzenie, iż stopień rozproszenia logiki analizy kontekstu można wyrazić poprzez wielkość kontekstu (zgodnie ze wzorem (5.5)), a w związku z tym wpływa on na wartość przedziału czasu wykonania instrukcji ι_{AK}.

Zebrane w czasie eksperymentów wyniki, pozwalają stwierdzić, że wielkość kontekstu jak i liczba jednocześnie zachodzących w przestrzeni inteligentnej zdarzeń wpływają na czas reakcji środowiska SWIAI, w ramach którego wykonywane są aplikacje

IAI. Wpływ ten można opisać przy pomocy płaszczyzny trendu. Im większy jest kontekst i im więcej zdarzeń pojawia się jednocześnie w przestrzeni, w której działa aplikacja (i jest jednocześnie przekazywana z obiektów opakowujących usługi do SWIAI), tym dłużej trwa wykonywanie instrukcji maszyny PIAI. Jest to spowodowane tym, że (w rozpatrywanym przykładzie) większy kontekst oznacza większą liczbę oczekiwanych warunków wywołania akcji zawartych w definicji aplikacji. Podobnie, wzrost liczby jednoczesne dostarczanych przez usługi opakowujące sensory informacji o zdarzeniach powoduje wzrost jednoczesne wykonywanych operacji w ramach równolegle działających wątków SWIAI (np. wykorzystywanych w trakcie analizy kontekstu, gdzie każdy warunek jest sprawdzany w osobnym wątku). Z tego powodu zwiększenie liczby jednoczesnych zdarzeń powoduje zwiększenie czasu reakcji.

Wyznaczone płaszczyzny trendu pozwalają zaobserwować, w jaki sposób ulegają zmianie wartości współczynników kierunkowych dla zmieniających się wartości μ w przypadku, gdy warunki dotyczące różnych par zawartych w definicji aplikacji są spełnione jednocześnie. Wraz ze wzrostem wartości μ wartość współczynnika kierunkowego odnoszącego się do wielkości kontekstu pozostaje dodatnia, jednakże maleje co jest związane z tym, iż ulegają zrównolegleniu instrukcje maszyny PIAI dla odpowiednio mniejszej wielkości kontekstu (np. dla $\mu = 1$ i $\psi = 100$ maszyna analizuje 100 warunków, których przedziały czasów analizy zostały zsumowane, natomiast dla $\mu = 2$ i $\psi = 50$ w SWIAI zostały przeanalizowane w sposób równoległy 2 zestawy 50-ciu warunków, a przedziały czasów analizy każdego z zestawów uśredniony). Inny charakter mają zmiany wartości współczynnika kierunkowego dotyczącego jednoczesnej liczby zdarzeń - wzrost wartości μ powoduje także wzrost wartości współczynnika kierunkowego. Z jednej strony jest to rezultat sposobu wykonywania pomiarów, uwzględniającego czas od momentu rozpoczęcia analizy danej kontekstowej do momentu jej zakończenia. Ten przedział czasu dotyczy operacji wykonywanych przez serwer aplikacji (np. tworzenie obiektów w pamięci). Czas wykonania tych operacji w największym stopniu wpływa na analizę danych kontekstowych dostarczonych jako pierwsze przez usługi opakowujące sensory. Z drugiej strony zwiększenie wartości μ powoduje zwiększenie liczby akcji a co za tym idzie (w rozpatrywanym przykładzie) zmniejszenie liczby warunków określających czy daną akcję uruchomić (przy stałej wielkości kontekstu). Ponieważ warunków jest mniej, zatem mniej zdarzeń umożliwia sprawdzenie czy zostały one spełnione. W efekcie mniejsza liczba komunikatów przesyłanych od usług powoduje spełnienie wszystkich warunków i dostarczenie pierwszego z nich oraz jego analiza w większym stopniu (niż dla mniejszych wartości μ) wpływa na całkowity czas wykonywania operacji maszyny PIAI (ze względu na uruchomienie operacji związanych z działaniem serwera aplikacji).

Wykazano także, iż stopień rozproszenia logiki analizy kontekstu można wyrazić poprzez wielkość kontekstu, co powoduje iż on również wpływa na przedział czasu wykonania instrukcji analizy kontekstu maszyny PIAI. Jest to zgodne z teoretycznymi rozważaniami (przeprowadzonymi w rozdziale V.3.3) co, wraz z przeprowadzeniem badań dla dwóch różnych aplikacji, świadczy o poprawności implementacji SWIAI. Spośród jego instrukcji, w największym stopniu na czas wykonywania aplikacji (a w związku z tym na efektywność) wpływa analiza danych kontekstowych (kontekst) oraz uruchamianie akcji. Stanowi to wykazanie 3 tezy rozprawy.

Podczas badań przyjmowano różne wartości parametrów pomiarów (tzn. wielkości kontekstu i liczby jednoczesnych zdarzeń), aby sprawdzić uniwersalność działania SWIAI. Było to możliwe dzięki uniezależnieniu się od usług (w tym opakowujących sensory, poprzez zastosowanie generatora zdarzeń), wykorzystaniu ontologii jako formy reprezentacji kontekstu oraz przyjęciu teoretycznej interpretacji parametrów obiektów. Oznacza to, iż pomiary można powtórzyć dla dowolnych obiektów (i ich parametrów) znajdujących się w przestrzeni inteligentnej, których stan zmieniany jest przez zdarzenia. Dlatego też przykładowe aplikacje jakie podlegały badaniom są reprezentatywne.

VIII UWAGI KOŃCOWE

Aplikacje typu IAI stają się coraz powszechniejsze, co jest spowodowane rozwijającymi się przestrzeniami inteligentnymi i stawianymi im wymaganiami. Praktycznie każda aplikacja, która dostosowuje swoje działanie do bieżącego stanu przestrzeni inteligentnej w jakiej funkcjonuje i dodatkowo wchodzi w interakcję z innymi jej użytkownikami może być zamodelowana jako aplikacja IAI. Wokół problematyki tego rodzaju aplikacji skupia się rozprawa doktorska, której podsumowanie stanowi niniejszy rozdział. Zostały w nim opisane uzyskane osiągnięcia teoretyczne oraz praktyczne, a także przedstawione dalsze, wciąż otwarte problemy badawcze.

Osiągnięcia teoretyczne

Model aplikacji IAI – w rozprawie został zaproponowany model aplikacji działających w przestrzeni inteligentnej [69], które są zarówno interaktywne jak i iteracyjne oraz korzystają w czasie swojego działania z kontekstu (patrz rozdział IV), który może się zmieniać w czasie. Jego zaletą jest rozdzielenie logiki aplikacyjnej (tzn. zasad działania), opisującej kiedy i w jaki sposób dostosować działanie aplikacji do sytuacji zachodzących w przestrzeni, od aspektów programistycznych związanych z procesem analizy kontekstu i wykonaniem akcji. Dzięki niemu została poszerzona wiedza na temat tego rodzaju aplikacji w odniesieniu do sposobu ich projektowania i implementacji, dla którego punktem wyjścia jest kontekstowość (inaczej niż w przypadku modelu SOM – patrz rozdział III.4). Dodatkowo wprowadzony model umożliwia określenie i porównywanie poziomu interaktywności aplikacji jeszcze przed ich implementacją. Wydzielenie logiki aplikacyjnej otwiera z kolei możliwość jej formułowania (tzn. definiowania aplikacji IAI) przez użytkowników, nieposiadających programistycznej wiedzy (na przykład przy wykorzystaniu języka naturalnego). Ma to duże znaczenie, ponieważ przestrzenią funkcjonowania aplikacji jest przestrzeń inteligentna, która z założenia ma wspierać wszystkich jej użytkowników bez względu na ich umiejętności.

Model przestrzeni działania aplikacji IAI - utworzono model maszyny służącej do wykonywania aplikacji IAI (PIAI – patrz rozdział V) zbudowany w oparciu o zestaw równolegle działających maszyn RAM. Na jej podstawie opracowano następnie architekturę IAK (patrz rozdział VI) i wykonano jej implementację (środowisko SWIAI) przy wykorzystaniu technologii Java i .NET. Środowisko umożliwia zastosowanie równoległego przetwarzania w odniesieniu do rozpoznawania oczekiwanych warunków wywołania akcji oraz wykonywania akcji określonych w różnych parach (patrz rozdział IV) zawierających się w definicji aplikacji (czyli zwielokrotnienia komponentów KAK i

KWA – patrz rozdział VI). Podobnie komunikacja z usługami jest zrównoleglona. SWIAI może być umieszczone w przestrzeni inteligentnej, co umożliwia dostarczenie przez nią mechanizmów i procedur koniecznych do analizy kontekstu i wykonywania akcji zdefiniowanych w aplikacjach IAI. Zatem zastosowanie maszyny PIAI umożliwia wykonywanie przez przestrzeń inteligentną aplikacji IAI, które (dzięki zastosowaniu podejścia SOA) może dynamicznie dostosowywać się do możliwości przestrzeni. Definicje aplikacji mogą być dostarczane przez użytkownika przestrzeni, dzięki czemu może on dynamicznie „programować" przestrzeń (bez konieczności posiadania programistycznej wiedzy) – tzn. określać w jaki sposób ma się ona dostosowywać do różnych sytuacji (innymi słowy określać stopień inteligencji przestrzeni). Jest to zgodne z kierunkiem rozwoju przestrzeni inteligentnych jaki został opisany w rozdziale II.

Wyniki badań mają wpływ na projektowanie aplikacji IAI, ponieważ uzyskano dzięki nim informacje na temat skalowalności środowiska SWIAI. Płaszczyzny trendu (patrz rozdział VII) i wartości ich współczynników kierunkowych, pozwalają oszacować jaką wartość będzie miał czas reakcji środowiska (tzn. uruchomienia akcji) jeszcze przed implementacją aplikacji IAI. W tym celu należy przyjąć określoną wielkość kontekstu, liczbę oczekiwanych warunków, które będą jednocześnie sprawdzane oraz liczbę jednocześnie pojawiających się w przestrzeni zdarzeń.

Środowisko SWIAI jest uniwersalne - tzn. może zostać wykorzystane do wykonywania każdej aplikacji IAI i może istnieć w dowolnej przestrzeni inteligentnej, w której można wyróżnić usługi opakowujące dostarczające dane kontekstowe, służące do wspomagania procesu analizy kontekstu i wykonujące akcje. Jest to możliwe dzięki temu, że środowisko (jak i jego architektura) zbudowane jest na bazie maszyny PIAI, której instrukcje wynikają z teoretycznego modelu aplikacji IAI.

Procedury analizy kontekstu – podczas prac nad rozprawą określono jakie są niezbędne mechanizmy i procedury konieczne do wykonywania aplikacji IAI bezpośrednio przez przestrzeń inteligentną, w której znajdują się określone usługi (zostały one wyszczególnione w rozdziale V). Jedną z takich procedur jest analiza kontekstu, dla której zaproponowano algorytm oparty na zmodyfikowanej sieci Petrie'go [70]. Może on zostać wzbogacony o mechanizm wykorzystania usług znajdujących się w przestrzeni inteligentnej do analizy kontekstu [71].

Osiągnięcia praktyczne

Implementacja przestrzeni ukierunkowanej na przeprowadzenie eksperymentów – zaprojektowano i zaimplementowano środowisko wykonywania aplikacji IAI (SWIAI) zbudowane na bazie architektury IAK – zostało ono opisane w rozdziale VI. Środowisko to

posłużyło do wykonania badań dotyczących różnych aplikacji IAI, co potwierdza, iż określone zasady działania (patrz rozdział V) i jego konstrukcja (patrz rozdział VI) umożliwiają wykonywanie tego rodzaju aplikacji w przestrzeni inteligentnej (jego implementacja jest poprawna).

Przeprowadzenie eksperymentów – wyniki uzyskane podczas badań (przedstawionych w rozdziale VII) świadczą o tym, że model aplikacji IAI może zostać wykorzystany do projektowania i implementacji aplikacji dla różnych przestrzeni inteligentnych, reagujących na zachodzące w niej zmiany. Zmiany te (wyrażone przez kontekst) mają z kolei wpływ na wykonywanie takich aplikacji. Im jest ich więcej i im większego fragmentu przestrzeni (istotnego z punktu widzenia aplikacji) dotyczą, tym mniejsza jest efektywność wykonania aplikacji IAI. Zależy ona przede wszystkim od algorytmów zastosowanych do analizy kontekstu oraz uruchamiania (i wykonania) akcji. Jak wynika z pomiarów, to te dwie instrukcje maszyny PIAI w największym stopniu wpływają na efektywność wykonania aplikacji IAI (rozumianą jako czas reakcji maszyny PIAI – patrz rozdział V) w SWIAI. Jednakże nie musi być to prawdziwe dla każdej implementacji, ze względu na to, że istotny wpływ może mieć także transformacja danych kontekstowych.

Potwierdzenie tez dla każdej przestrzeni o określonych warunkach działania – Przedstawione w rozdziale I tezy rozprawy doktorskiej zostały wykazane dla zaproponowanych modeli IAI (patrz rozdział IV) oraz PIAI (patrz rozdział V). Pierwsza z nich dotyczy możliwości opisu aplikacji IAI przy pomocy maszyny PIAI. Jak zostało to przedstawione w rozdziale IV taki opis, zawierający oczekiwane warunki wywołania akcji oraz akcje, można przygotować dla każdej aplikacji IAI. Druga teza mówi o tym, iż maszyna PIAI stanowi bazę do budowy środowiska wykonywania SWIAI, jakie może zostać umieszczone w przestrzeni inteligentnej z wydzielonym zbiorem instrukcji i usług wspomagających jej działalność. Zostało to wykazane w rozdziale VI, opisującym architekturę IAK (zbudowaną na podstawie modelu PIAI) oraz jej implementację SWIAI jak została umieszczona w przestrzeni inteligentnej, ukierunkowanej na przeprowadzenie badań eksperymentalnych. Stanowią one również podstawę wykazania prawdziwości ostatniej – trzeciej tezy rozprawy (patrz rozdział VII) mówiącej o tym, że efektywność działania (czas reakcji) SWIAI zależy od wielkości kontekstu, rozproszenia logiki jego analizy pomiędzy maszynę i usługi znajdujące się przestrzeni inteligentnej, a także od liczby zdarzeń jednocześnie zachodzących w przestrzeni. W ten sposób osiągnięto cel rozprawy postawiony w rozdziale I związany z określeniem jaki jest wpływ kontekstu na efektywność wykonania aplikacji IAI w dedykowanej przestrzeni usług (co jest tytułem rozprawy).

Wprowadzone w niniejszej rozprawie modele IAI oraz PIAI otwierają nowe problemy

badawcze. Pierwszym z nich jest konieczność klasyfikacji danych wykorzystywanych podczas działania aplikacji na dane wejściowe i kontekstowe (patrz rozdział IV). Dodatkowo, analiza kontekstu może odnosić się do danych historycznych, co oznacza konieczność rozbudowy architektury IAK, w ramach której musiałby istnieć komponent, odpowiadający za zbieranie i zapamiętywanie danych kontekstowych. Dane te mogą następnie być przetwarzane i analizowane przy pomocy reguł z bazy wiedzy, tak żeby wychwycić zależności, jakie między nimi istnieją. Przy założeniu istnienia takiej bazy wiedzy oraz konieczności transformacji danych kontekstowych do standardowej postaci pojawia się również problem skali – im większy jest fragment przestrzeni inteligentnej, z którym aplikacja wchodzi w interakcję tym więcej danych kontekstowych musi zostać przeanalizowanych. Problem ten może zostać rozwiązany poprzez wyodrębnienie domen (np. odpowiadającym dziedzinom zastosowaniom aplikacji) i modyfikację komponentu BW (patrz rozdział VI), polegającą na utrzymywaniu różnych baz wiedzy dla poszczególnych domen. Architektura IAK może również zostać uszczegółowiona w kierunku zapewnienia ciągłości pracy w przypadku awarii któregoś z komponentów albo usług bazowych. Istotnym aspektem jest także wiarygodność w odniesieniu do zapewnienia określonych charakterystyk wykonania aplikacji IAI określonych przez projektanta (np. czasu wykonania, wykorzystania zaufanych usług, itp.), które mogą zostać uzyskane poprzez zastosowanie mechanizmów QoS.

BIBLIOGRAFIA

[1] Ahamed S.I., Zulkernine M., Anamanamuri S., "A dependable device discovery approach for pervasive computing middleware", Availability, Reliability and Security, pp. 66-73, 2006,

[2] Ailisto H., Alahuhta P., Haataja V., Kyllönen V., Lindholm M., "Structuring Context Aware Applications: Five-Layer Model and Example Case", Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing, 2002,

[3] Al-Bahlal M., Al-Muhtadi J., "A Middleware for Personal Smart Spaces", Computer Software and Applications Conference Workshops, pp. 299-304, 2010,

[4] Alur R., Dill D.L., "A theory of timed automata", Theoretical Computer Science, vol. 126 issue 2, pp. 183-235, 1994,

[5] Autebert J.M., Berstel J., Boasson L., "Context-Free Languages and Push-Down Automata", Handbook of Formal Languages, vol. 1, pp. 111-174, 1997,

[6] Baldauf M., Dustdar S., "A survey on context-aware systems", International Journal of Ad Hoc and Ubiquitous Computing, vol.2 issue 4, pp. 263-277, 2004,

[7] Banavar G., Beck J., Gluzberg E., Munson J., Sussman J., Zukowski D., "Challenges: An Application Model for Pervasive Computing", Proceedings of 6th annual international conference on Mobile computing and networking, pp. 266-274, 2000,

[8] Bell M., "Service-Oriented Modeling: Service Analysis, Design, and Architecture", Wiley, 2008,

[9] Bing Y., Yong L., Guo-yan Y., "Research on middleware technology for pervasive computing", Computer Application and System Modeling, vol. 7, pp. 132-134, 2010,

[10] Bojanczyk M., Muscholl A., Schwentick T., Segoufin L., David C., "Two-Variable Logic on Data Words", Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science, pp. 7-16, 2006,

[11] Brscic D., Hashimoto H., "Tracking of Humans Inside Intelligent Space using Static and Mobile Sensors," Industrial Electronics Society, pp. 10-15, 2007,

[12] Cao L., Li Q., Sui Q., "A Service-oriented Adaptive Framework for Pervasive Computing", Pervasive Computing and Applications, pp. 631–634, 2006,

[13] Chang J., Na S., Yoon M., "Intelligent Context-Aware System Architecture in Pervasive Computing Environment", Parallel and Distributed Processing with Applications, pp. 745–750, 2008,

[14] Chen G., Li G., Wu B., Pei S., "PConG: A novel platform available for pervasive computing based on GPU", Pervasive Computing, pp. 547-550, 2009,

[15] Chen H., "An Intelligent Broker Architecture for Context-Aware Systems", PhD Dissertation proposal, 2003,

[16] Chen H., Finin T., Anumap J., "An ontology for context-aware pervasive computing environments", The Knowledge Engineering Review, vol. 18 issue 3, pp. 197–207, 2003,
[17] Chen H., Finin T., Joshi A., "The SOUPA Ontology for Pervasive Computing", Ontologies for Agents: Theory and Experiences, pp. 233-258, 2005,
[18] Chen H., Perich F., Finin T., Joshi A., "SOUPA: standard ontology for ubiquitous and pervasive applications", Mobile and Ubiquitous Systems: Networking and Services, pp. 258-267, 2004,
[19] Chen Q., Cordea M.D., Petriu E.M., Whalen T.E., Rudas, I.J., Varkonyi-Koczy A.R., "Hand-Gesture and Facial-Expression Human-Computer Interfaces for Intelligent Space Applications", Medical Measurements and Applications, pp. 1-6, 2008,
[20] Chen Z., Ge L., Wang H., Huang X., Lin J., "A Trust-Based Service Evaluation and Selection Model in Pervasive Computing Environment", Pervasive Computing and Applications, pp. 641-646, 2006,
[21] Chin J.S., Callaghan V., Clarke G., "A Pervasive Computing Programming Approach for Non-Technical Users", Pervasive Computing and Applications, pp. 235-240, 2006,
[22] Ciarletta L., Dima A., "A conceptual model for pervasive computing", Parallel Processing, pp. 9-15, 2000,
[23] Cheng J., Goto Y., Someya M., Endo T., "Persistent Computing Systems as an Infrastructure for Pervasive Services", Pervasive Computing and Applications, pp. 104-109, 2006,
[24] Clements-Croome D.J., "Intelligent buildings: design, management and operation", Thomas Telford, London, 2004,
[25] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C., "Introduction to Algorithms", The MIT Press, 2002,
[26] Coutaz J., "PAC, and Object Oriented Model for Dialog Design", Human-Computer Interaction, pp. 431-436, 1987,
[27] Daszczuk W.B., "Real Time Model Checking Using Timed Concurrent State Machines", International Journal of Computer Science & Applications, pp. 1-12, 2007,
[28] Dey A. K., "Context-aware computing: the cyberdesk project", AAAI 1998 Spring Symposium on Intelligent Environments, Tech. Report SS-98-02, pp. 51-54, 1998,
[29] Dey A.K., "Understanding and using context", Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing, vol. 5 issue 1, pp.4-7, 2001,
[30] Dey A. K., Abowd G. D., "Towards a better understanding of context and context-awareness", Proceedings of the 1st international symposium on Handheld and

Ubiquitous Computing, pp. 304-307, 1999,
[31] Dey A.K., Abowd G.D., Salber D., "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Human-Computer Interaction, vol. 16 issue 2, pp. 97–166, 2001,
[32] Dey A.K., Mankoff J., Gregory D., "Distributed mediation of ambiguous context in aware environments", Proceedings of the 15th annual ACM symposium on User interface software and technology, pp. 121-130, 2002,
[33] Dhingra V., Arora A., "Pervasive Computing: Paradigm for New Era Computing", Emerging Trends in Engineering and Technology, pp. 349-354, 2008,
[34] Dong Y., Li Q., Shi Y., "Research on the Architecture of Ontology-based Context-aware Application in Pervasive Environment", Pervasive Computing and Applications, pp. 128-132, 2007,
[35] Dourish P., "What we talk about when we talk about context", Personal and Ubiquitous Computing, vol.8 issue 1, pp. 19-30, 2004,
[36] Du W., Wang L., "Context-aware application programming for mobile devices", Proceedings of C3S2E'2008, pp.215-227, 2008,
[37] Erickson T., "Some problems with the Notion of Context-Aware computing", Communications of the ACM - Ontology: different ways of representing the same concept, vol. 45 issue 2, pp. 102-104, 2002,
[38] Farrell J., "Microsoft Visual C# 2010: An Introduction to Object-Oriented Programming", Course Technology; 4 edycja, 2010,
[39] Finkenzeller K., "RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification 2nd Edition", Wiley, 2003,
[40] Flynn M.J., "Some Computer Organizations and Their Effectiveness", Computers, vol. C-21 issue 9, pp. 948-960, 1972,
[41] Fortune S., WyllieJ., "Parallelism in random access machines", Proceedings of the tenth annual ACM symposium on Theory of computing, pp.114-118, 1978,
[42] Franklin D., Flaschbart J., "All gadget and no representation makes Jack a dull environment", AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02, pp. 155-160, 1998,
[43] Gassmann O., Meixner H., "Sensors in intelligent buildings: Overview and trends," Sensors in Intelligent Buildings, vol. 2, pp. 3-25, 2004,
[44] Gouin-Vallerand C., Abdulrazak B., Giroux S., Mokhtari M., "A Software Self-Organizing Middleware for Smart Spaces Based on Fuzzy Logic", High Performance Computing and Communications, pp. 138-145, 2010,
[45] Greenberg S., "Context as a dynamic construct", Human-Computer Interaction, vol.16 issue 2, pp. 257-268, 2001,

[46] Grumberg O., Kupferman O., Sheinvald S., "Variable Automata over Infinite Alphabets", Language and Automata Theory and Applications, Lecture Notes in Computer Science, vol. 6031, pp. 561-572, 2010,
[47] Guarino N., Giaretta P., "Ontologies and Knowledge Bases: Towards a Terminological Clarifcation", Knowledge Acquisition, IOS Press, vol. 1 issue 9, pp. 25-32, 1995,
[48] Guoguang Z., "A direct iterative algorithm for linear programming", Control Conference, pp. 75-77, 2010,
[49] Han L., Jyri S., Ma J., Yu K., "Research on Context-Aware Mobile Computing", Advanced Information Networking and Applications - Workshops, pp. 24-30, 2008,
[50] Hashimoto H., "Intelligent space: a commentary on research trends", Industrial Electronics, vol. 1, pp. 11-16, 2002,
[51] Hashimoto H., "Intelligent space - how to make spaces intelligent by using DIND", Systems, Man and Cybernetics, vol. 1, pp. 14-19, 2002,
[52] Henricksen K., Indulska J., Rakotonirainy A., "Modeling Context Information in Pervasive Computing Systems", Proceedings of First International Conference on Pervasive Computing, pp. 79-117, 2002,
[53] Hopcroft J.E., Motwani R., Ullman J.D., "Introduction to Automata Theory, Languages, and Computation (2nd Edition)", Addison Wesley, 2000,
[54] Hsu C.L., "Constructing intelligent living-space controlling system with blue-tooth and speech-recognition microprocessor", Expert Systems with Applications: An International Journal, vol. 36 issue 5, pp. 9308-9318, 2009,
[55] Hsu C.L., Hsu T.Y., Ho K.Y., Wu W.B., "Practical Design of Intelligent Living-Space Based on Blue-Tooth System", Advanced Information Networking and Applications – Workshops, pp. 1505-1510, 2008,
[56] Hsu H.J., Wu S.Y., Wang F.J., "A Methodology to Developing Situation-Aware Pervasive Applications with Service Oriented Architecture", Services, pp. 657-660, 2010,
[57] Hull R., Neaves P., Bedford-Roberts J., "Towards situated computing", Proceedings of 1st International Symposium on Wearable Computers, pp. 146-153, 1997,
[58] Jennings N.R., Wooldridge M.J., "Agent technology: foundations, applications, and markets", Springer, 2002,
[59] Jansen E., Yang H.I., Abdulrazak B., Helal S., "A context-driven programming model for pervasive spaces", Proceedings of the 5th international conference on Smart homes and health telematics, pp. 31-43, 2007,
[60] Jih W., Chen L., Hsu J.Y., "A Context-aware Service Platform in a Smart Space", 2007 ACM International Workshop on Agent-Based Ubiquitous Computing, 2007,

[61] Kalasapur S., Kumar M., Shirazi B., “Evaluating service oriented architectures (SOA) in pervasive computing”, Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 276-285, 2006,
[62] Kallio P., Latvakoski J., “Challenges and requirements of ubiquitous computing”, WSEAS Transactions on Information Science and Applications, vol. 1 issue 1, pp. 234-239, 2004,
[63] Kawaji K., Niitsuma M., Kosaka A., Hashimoto H., “Acquisition of objects' properties in Intelligent Space”, SICE Annual Conference, pp. 259–263, 2007,
[64] Kemmerer R.A., Kolano, P.Z., “Formally specifying and verifying real-time systems”, Formal Engineering Methods, pp. 112-120, 2002,
[65] Kim M.J., Kumar M., Shirazi B.A., Chong H.J., “A Novel Architecture for Provisioning Basic Services in Heterogeneous Pervasive Environments”, World of Wireless, Mobile and Multimedia Networks, pp. 1-4, 2007,
[66] Kohonen T., “The Self-Organizing Map”, Proceedings of the IEEE, vol. 78 no. 9, pp. 1464-1480, 1990,
[67] Korpipaa P., Hakkila J., Kela J., Ronkainen S., Kansala I., “Utilising context ontology in mobile device application personalization”, Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, pp. 133-140, 2004,
[68] Korpipää P., Mäntyjärvi J., “An Ontology for Mobile Device Sensor-Based Context Awareness” Modeling and Using Context, Lecture Notes in Computer Science, vol. 2680, pp. 451-458, 2003,
[69] Krawczyk H., Nasiadka S., “A New Model for context-aware applications analysis and design”, The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, ThinkMind, pp. 211-217, 2011,
[70] Krawczyk H., Nasiadka S., ”Metoda wyznaczania kontekstu dla aplikacji sterowanych zdarzeniami”, „Metody Informatyki Stosowanej”, PAN oddział w Gdańsku., 2008,
[71] Krawczyk H., Nasiadka S., “Service oriented approach to complex context analysis”, Proceedings of the 2011 Second International Conference on Pervasive Computing, Signal Processing and Applications, 2011, w druku,
[72] Kurkovsky S., “Pervasive computing: Past, present and future”, Information and Communications Technology, pp. 65-71, 2007,
[73] Lee J.H., “A New Localization Scheme based on Cooperative Environment”, PhD Thesis, University of Tokyo, 1999,
[74] Lee J.H., Ando N., Hashimoto H., “Design Policy of Intelligent Space”, Systems, Man, and Cybernetics, vol. 3, pp. 1077–1082, 1999,
[75] Lee J.H., Ando N., Yakushi T., Nakajima K., Kagoshima T., Hashimoto H., “Applying intelligent space to warehouse-the first step of intelligent space project”,

Advanced Intelligent Mechatronics, vol. 1, pp. 290-295, 2001,
[76] Lee J.H., Hashimoto H., "Global Positioning System for Mobile Robots with Distributed Sensors", Proceedings of the International Conference IROS, pp. 1733-1738 , 1999,
[77] Lee J.H., Hashimoto H., "Intelligent space", Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1358-1363, 2000,
[78] Lee J.H., Hashimoto H., "Intelligent space - concept and contents", Advanced Robotics, vol. 16, no. 3, pp. 265-280, 2002,
[79] Lee J.H., Hashimoto H., "Intelligent Space, its past and future", Industrial Electronics Society, vol. 1, pp. 126-131, 1999,
[80] Lee J.H., Morioka K., Ando N., Hashimoto H., "Cooperation of Distributed Intelligent Sensors in Intelligent Environment", IEEE/ASME Transactions On Mechatronics, vol. 9, no. 3, pp. 535–543, 2004,
[81] Lee S.O., Sakurai R., Nishizawa T., Lee J.H., Park G.T., "A spatial history storing system based on Intelligent Space", Industrial Electronics, pp. 3397–3402, 2008,
[82] Leung W., Vanijjirattikhan R., Li Z., Xu L., Richards T., Ayhan B., Chow M.Y., "Intelligent Space with Time Sensitive Applications", Advanced Intelligent Mechatronics, pp. 1413–1418, 2005,
[83] Li H., Wang H., "A Method of Service Description and Discovery in Pervasive Computing Environments", Pervasive Computing and Applications, pp. 604-607, 2006,
[84] Liu B., Wang F.Y., Geng J., Yao Q., Gao H., Zhang B., "Intelligent spaces: An overview", Vehicular Electronics and Safety, pp. 1-6, 2007,
[85] Liu K., Lin C., Qiao B., "A multi-agent system for intelligent pervasive spaces", Service Operations and Logistics, and Informatics, pp. 1005-1010, 2008,
[86] Liu X., Matsikoudis E., Lee E.A., "Modeling Timed Concurrent Systems", Concurrency Theory, pp. 27-30, 2006,
[87] Ma C., Dai G., Wang H., "Capturing Invisible Design: Modeling Multi-modal Interaction in Pervasive Computing", Pervasive Computing and Applications, pp. 162-167, 2006,
[88] Magott J., "Techniki opisu formalnego systemów informatycznych czasu rzeczywistego", Wydawnictwa Komunikacji i Łączności, 2005,
[89] Makris D., Ellis T., "The potential of using computer vision systems in intelligent interactive buildings", Intelligent Environments, pp. 289–296, 2006,
[90] Malatras A., Hirsbrunner B., "A Context-Aware Framework to Enable Adaptation in Pervasive Computing Environments", Network-Based Information Systems, pp. 182-187, 2009,

[91] Mankoff J., Gregory D. A., Hudson S. E., "OOPS: A Toolkit Supporting Mediation Techniques for Resolving Ambiguity in Recognition-Based Interfaces", Computers and Graphics, vol. 24 issue 6, pp. 819-834, 2000,

[92] Mazurkiewicz A., "Problemy przetwarzania informacji", Tom 2, Wydawnictwa Naukowo-Techniczne, 1974,

[93] Mäntyjärvi J., Tuomela U., Kansala I., Hakkila J., "Context-studio – Tool for Personalizing Context-Aware Applications in Mobile Terminals", Proceedings of Australasian Computer Human Interaction Conf., Addison-Wesley Longman, pp. 64-73, 2003,

[94] Mendelbaum H.G., Yehezkael R.B., "Using 'parallel automaton' as a single notation to specify, design and control small computer based systems", Engineering of Computer Based Systems, pp. 152-159, 2001,

[95] Morioka K., Hashimoto H., "Appearance Based Object Identification for Distributed Vision Sensors in Intelligent Space", Intelligent Robots and Systems, vol.1, pp.199-204, 2004,

[96] Morioka K., Mao X., Hashimoto H., "Global Color Model Based Object Matching in the Multi-Camera Environment", Intelligent Robots and Systems, pp.2644-2649, 2006,

[97] Nasiadka S., „Aplikacje i środowiska kontekstowe: Środowisko OSGi oraz definiowalna architektura SOA", Praca zbiorowa Katedry Architektury Systemów Komputerowych (KASKBOOK), pp. 149-164, 2008,

[98] Nasiadka S., Dziubich K., Dziubich T., Budnik Ł., "Dynamically configurable platform for service integration", Proceedings of the 1st International Conference on Information Technology, pp. 51-56, 2008,

[99] Niitsuma M., Hashimoto H., Hashimoto H., "Spatial Memory as an Aid System for Human Activity in Intelligent Space", IEEE Trans. on Industrial Electronics, vol.54, pp. 1122-1131, 2007,

[100] Niitsuma M., Hashimoto H., "Spatial memory for collaboration support system in intelligent space -User identification based on spatial memory", SICE Annual Conference, pp. 3073-3079, 2007,

[101] Niitsuma M., Kawaji K., Yokoi K., Hashimoto H., "Extraction of Human – Object Relations in Intelligent Space", Robot and Human Interactive Communication, pp. 520–525, 2008,

[102] Niitsuma M., Yokoi K., Hashimoto H., "Describing Human-Object Interaction in Intelligent Space", Human System Interactions, pp. 395–399, 2009,

[103] Orriëns B., Yang J., Papazoglou M.P., "A Framework for Business Rule Driven Service Composition", Technologies for E-Services, pp. 14-27, 2003,

[104] Palafox, L.F., Hashimoto H., “A movement profile detection system using self organized maps in the Intelligent Space” , Advanced Robotics and its Social Impacts, pp. 114-118, 2009,

[105] Papazoglou M.P., Traverso P., Dustdar S., Leymann F., “Service-Oriented Computing: State of the Art and Research Challenges”, Computer, vol. 40 issue 11, pp. 38-45, 2007,

[106] Parashar M., Hariri S., “Autonomic Computing: An Overview”, Unconventional Programming Paradigms, pp. 247-259, 2005,

[107] Peterson J.L., “Petri Net Theory and the Modeling of Systems”, Prentice Hall, 1981,

[108] Potel M., “MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java”, Taligent Inc, 1996,

[109] Presecan S., Tomai N., “Distributed context provisioning and reaction middleware”, Intelligent Computer Communication and Processing, pp. 351-354, 2009,

[110] Qiao B, Liu K., Guy C., "A Multi-Agent System for Building Control,"Proceedings of the IEEEIWIC/ACM International Conference on Intelligent Agent Technology, vol. 00, pp. 653-659, 2006,

[111] Rehman K., Stajano F., Coulouris G., “An Architecture for Interactive Context-Aware Applications”, Pervasive Computing, vol. 6 issue 1, pp. 73-80, 2007,

[112] Rodden T., Cheverst K., Davies K., Dix A., “Exploiting context in HCI design for mobile systems”, Proceedings of Workshop on Human Computer Interaction with Mobile Devices, 1998,

[113] Ronzani D., “THE BATTLE OF CONCEPTS : UBIQUITOUS COMPUTING , PERVASIVE COMPUTING AND AMBIENT INTELLIGENCE IN MASS MEDIA”, Computing, vol. 4 issue 3, 2000,

[114] Rosenberg D., Stephens M., “Use Case Driven Object Modeling with UML: Theory and Practice”, Apress, 2007,

[115] Ross T.J., “Fuzzy Logic with Engineering Applications, Third Edition”, Wiley, 2010,

[116] Ryan N.S., Pascoe J., Morse D.R., “Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant”, Computer applications in Archaeology, 1997,

[117] Saha D., Mukherjee A., “Pervasive computing: a paradigm for the 21st century”, Computer, vol. 36 issue 3, pp. 25-31, 2003,

[118] Sasaki T., Hashimoto H., “Hierarchical Framework for Implementation of Intelligent Space”, Industrial Electronics Society, pp. 28-33, 2007,

[119] Schilit B., Theimer M., “Disseminating active map information to mobile hosts”,

Network, vol. 8 issue 5, pp. 22-32, 1994,
[120] Schroeder A., van der Zwaag M., Hammer M., "A Middleware Architecture for Human-Centred Pervasive Adaptive Applications", Self-Adaptive and Self-Organizing Systems Workshops, pp. 138-143, 2008,
[121] Shadbolt N., "Ambient Intelligence", IEEE Intelligent Systems, pp. 2-3, 2003,
[122] Sharifi M., Tabatabaei S., Movahednejad H., Manaf A.A., Ibrahim S., "A Security Conscious Service Discovery Framework in Pervasive Computing Environments", Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 256–262, 2009,
[123] Skliarova I., Sklyarov V., "Recursive versus Iterative Algorithms for Solving Combinatorial Search problems in Hardware", VLSI Design, pp. 255-260, 2008,
[124] Soylu A., De Causmaecker P., "Merging model driven and ontology driven system development approaches pervasive computing perspective", Computer and Information Sciences, pp. 730-735, 2009,
[125] Snir M., "On parallel searching", Proceedings of the first ACM SIGACT-SIGOPS symposium on Principles of distributed computing, pp. 242-253, 1982,
[126] Soylu A., Causmaecker P., Desmet P., "Context and Adaptivity in Context-Aware Pervasive Computing Environments", Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, pp. 94-101, 2009,
[127] Stotts P.D., Pugh W., "Parallel Finite Automata for Modeling Concurrent Software Systems", Journal of Systems and Software, vol. 27, pp. 27-43, 1994,
[128] Sui Q., Wang H.Y., "A Service Oriented Flow Pattern in Pervasive Computing Environments", Pervasive Computing and Applications, pp. 625–630, 2006,
[129] Syed A.A., Lukkien J., Frunza R., "An architecture for self-organization in pervasive systems", Design, Automation & Test in Europe Conference & Exhibition, pp. 1548-1553, 2010,
[130] Tate A., Chen-Burger Y.H., Dalton J., Potter S., Richardson D., Stader J., Wickler G., Bankier I., Walton C., Williams P., "I-Room: A Virtual Space for Intelligent Interaction", Intelligent Systems, vol. 25 issue 4, pp. 62-71, 2010,
[131] Trivedi M.M., Huang K.S., Mikic I., "Dynamic context capture and distributed video arrays for intelligent spaces", Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 35 issue 2, pp. 145-163, 2005,
[132] Toth A. A., Varkonyi-Koczy A. R., "Hand gesture controlled interface for Intelligent Space applications", Intelligent Engineering Systems, pp. 239–244, 2009,
[133] Vanijjirattikhan R., Chow M.Y., Szemes P., Hashimoto H., "Mobile agent gain scheduler control in inter-continental Intelligent Space", The 2005 IEEE International Conference on Robotics and Automation, pp. 1115-1120, 2005,

[134] Varkonyi-Koczy A.R., Tusor B., “Man-machine communication via CFNN-based hand sign models”, Applied Machine Intelligence and Informatics, pp. 329-334, 2010,
[135] Wang F.Y., “Driving into the Future with ITS”, Intelligent Systems, vol. 21 issue 3, pp. 94-95, 2006,
[136] Wang F.Y., Zeng D., Yang L., “Smart Cars on Smart Roads: An IEEE Intelligent Transportation Systems Society Update”, Pervasive Computing, pp. 68-69, 2006,
[137] Wang J., Yang Z., Chen Y., Kou W., Zhang Z., “A Trust Model in Pervasive Computing”, Pervasive Computing and Applications, pp. 370-374, 2008,
[138] Want R., Borriello G., Pering T., Farkas K.I., “Disappearing Hardware”, IEEE Pervasive Computing, vol. 1 issue 1, pp. 36-47, 2002,
[139] Weiser M., “The computer for the 21st century”, ACM SIGMOBILE Mobile Computing and Communications Review - Special issue dedicated to Mark Weiser, vol. 3 issue 3, pp. 3-11, 1999,
[140] Winograd T., “Architectures for context”, Human-Computer Interaction, vol. 16 issue 2, pp. 401-419, 2001,
[141] WWW: Burbeck S., “Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC)”, http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html,
[142] WWW: OSI, http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip,
[143] WWW: OWL, http://www.w3.org/TR/owl-features/,
[144] WWW: SOAP, http://www.w3.org/TR/soap/,
[145] WWW: WebService, http://www.w3.org/TR/ws-arch/,
[146] WWW: WSCDL, http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/,
[147] Xie P., Zhou J.; “Research on Heterogeneous Component Platform for Pervasive Computing”, Environmental Science and Information Application Technology, pp. 590-593, 2009,
[148] Xu W., Xin Y., Lu G., “A System Architecture for Pervasive Computing”, Natural Computation, pp. 772-776, 2007,
[149] Xu W., Xin Y., Lu G., “A Trust Framework for Pervasive Computing Environments”, Wireless Communications, Networking and Mobile Computing, pp. 2222–2225, 2007,
[150] Xu W., Xin Y., Lu G., Chen Q., “A New Service Description, Matching and Selection Mechanism for Pervasive Computing”, Fuzzy Systems and Knowledge Discovery, pp. 509–514, 2009,
[151] Yamahara H., Soma T., Harada F., Takada H. Shimakawa H., Shimada Y., “

TaggedWorld: An Intelligent Space Providing Services by Interaction between a User and an Environment", Proceedings of the 2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 333-342, 2008,

[152] Yang H.I., Helal A., "Safety Enhancing Mechanisms for Pervasive Computing Systems in Intelligent Environments", Pervasive Computing and Communications, pp. 525-530, 2008,

[153] Yang H.I., Jansen E., Helal S., "A comparison of two programming models for pervasive computing", Applications and the Internet Workshops, pp. 4, 2006,

[154] Yang L., Wang F.Y., "Driving into Intelligent Spaces with Pervasive Communications", Intelligent Systems, vol. 22 issue 1, pp. 12-15, 2007,

[155] Yao Q., Chang S., Qi Y., Ming L., "New Programming Model for Pervasive Computing", e-Business Engineering, pp. 325-332, 2008,

[156] Yin S., Ray I., Ray I., "A Trust Model for Pervasive Computing Environments", Collaborative Computing: Networking, Applications and Worksharing, pp. 1-6, 2007,

[157] Yin Q., Hu H., Li J., Lu J., "A Middleware Approach for Behavior Consistent Composition of Services in Smart Space", Service-Oriented Computing and Applications, pp. 233-240, 2007,

[158] Ying X., Fu-yuan X., "Research on Context Modeling Based on Ontology", Computational Intelligence for Modelling, Control and Automation, pp. 188-188, 2006,

[159] Zhang K., Li Q., Sui Q., "A Goal-driven Approach of Service Composition for Pervasive Computing", Pervasive Computing and Applications, pp. 593–598, 2006,

[160] Zhenjiang M., Baozong Y., "A human-centered pervasive computing system model", Wireless, Mobile and Multimedia Networks, pp. 1-4, 2006,

[161] Zhou J., Riekki J., "Context-Aware Pervasive Service Composition", 2010 International Conference on Intelligent Systems, Modelling and Simulation, pp. 437-442, 2010,

[162] Zhou F., Tian G., Yang Y., Xiao H., Chen J., "Research and implementation of voice interaction system based on PC in intelligent space", Automation and Logistics, pp. 679-684, 2010,

[163] Zhou J., Gilman E., Ylianttila M., Riekki J., "Pervasive Service Computing: Visions and Challenges", Computer and Information Technology, pp. 1335–1339, 2010,

[164] Zhou J., Riekki J., Ylianttila M., "Modeling service composition and exploring its characteristics", 3rd International Workshop on Web Service Composition and Adaptation, pp. 446-451, 2009.