WYKŁAD 5: WARSTWY POŚREDNIE (*MIDDLEWARE*) W SYSTEMACH KONTEKSTOWYCH

Jarosław Domaszewicz
Institute of Telecommunications, Warsaw University Of Technology

Plan wykładu

- Ogólne wprowadzenie do warstw pośrednich
- Warstwy pośrednie w systemach kontekstowych
- Podstawowe informacje o modelach programistycznych
- Platformy dla aplikacji kontekstowych

OGÓLNE WPROWADZENIE DO WARSTW POŚREDNICH

Wyzwania w tworzeniu złożonych systemów oprogramowania

- Przetwarzanie rozproszone
  - uciążliwe problemy z komunikacją między elementami systemu
- Heterogeniczność
  - różnorodność zasobów/usług oferowanych przez różne węzły
  - różnorodność platform sprzętowych (różne CPU, OS, …)
- Konieczność integracji zastanych aplikacji (*legacy software*), nie podlegających modyfikacji

Specyfika systemów kontekstowych

- Przetwarzanie rozproszone jest regułą
  - wielość terminali mobilnych i węzłów wbudowanych
  - podsystem infrastrukturalny (serwery)
- Heterogeniczność jest regułą
  - różnorodność sensorów i elementów wykonawczych
- Często system powstaje ad-hoc
  - nie znamy z góry dostępnych zasobów sensorowych i wykonawczych
- Małe zasoby obliczeniowe terminali i węzłów
  - energia(!), ale także CPU, pamięć, …
- Szybko zmienny kontekst systemowy
  - energia, zmienne warunki łączności, …

Specyfika systemów kontekstowych

- Przetwarzanie rozproszone jest regułą
  - wielość terminali mobilnych i węzłów wbudowanych
  - podsystem infrastrukturalny (serwery)
- Heterogeniczność jest regułą
  - różnorodność sensorów i elementów wykonawczych
- Często system powstaje ad-hoc
  - nie znamy z góry dostępnych zasobów sensorowych i wykonawczych
- Małe zasoby obliczeniowe terminali i węzłów
  - energia(!), ale także CPU, pamięć, …
- Szybko zmienny kontekst systemowy
  - energia, zmienne warunki łączności, …

Czy każdy programista musi te problemy rozwiązywać od początku?

Jak sobie z tym radzić???

- Słowo kluczowe: ponowne użycie (*reuse*, *reusability*)
- Zidentyfikujmy często powtarzające się problemy (wyzwania dla programisty)
- Rozwiążmy to problemy raz a dobrze, tworząc komponenty oprogramowania, które mogą być wykorzystane przez wiele aplikacji
- Niech komponenty te ukrywają niewygodne szczegóły i oferują aplikacjom (programistom) usługi wygodne w użyciu

„Generyczny" opis warstwy pośredniej (*middleware*)

The role of middleware is to make application development easier,

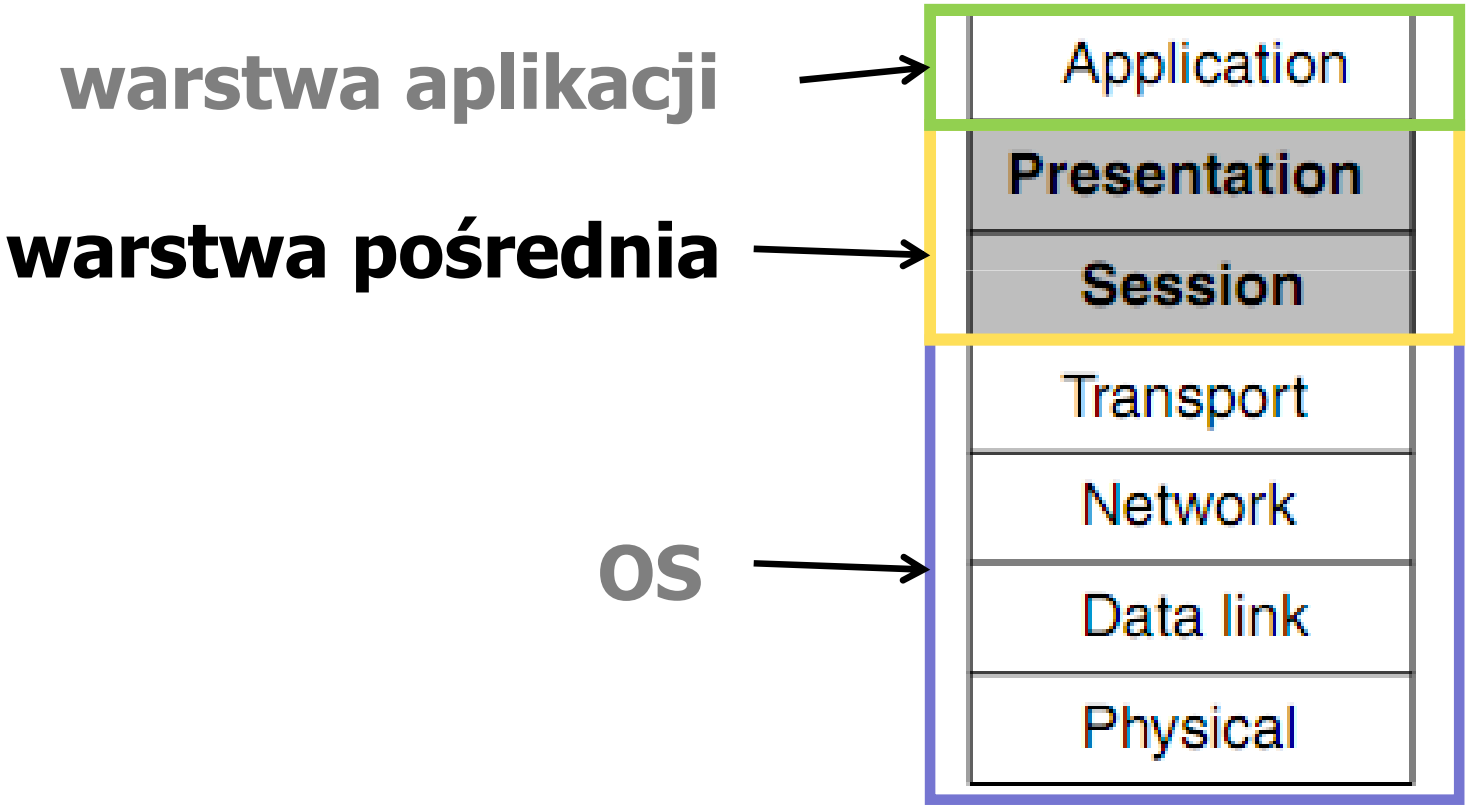
by providing common programming abstractions,

by masking the heterogeneity and the distribution
of the underlying hardware and operating systems,

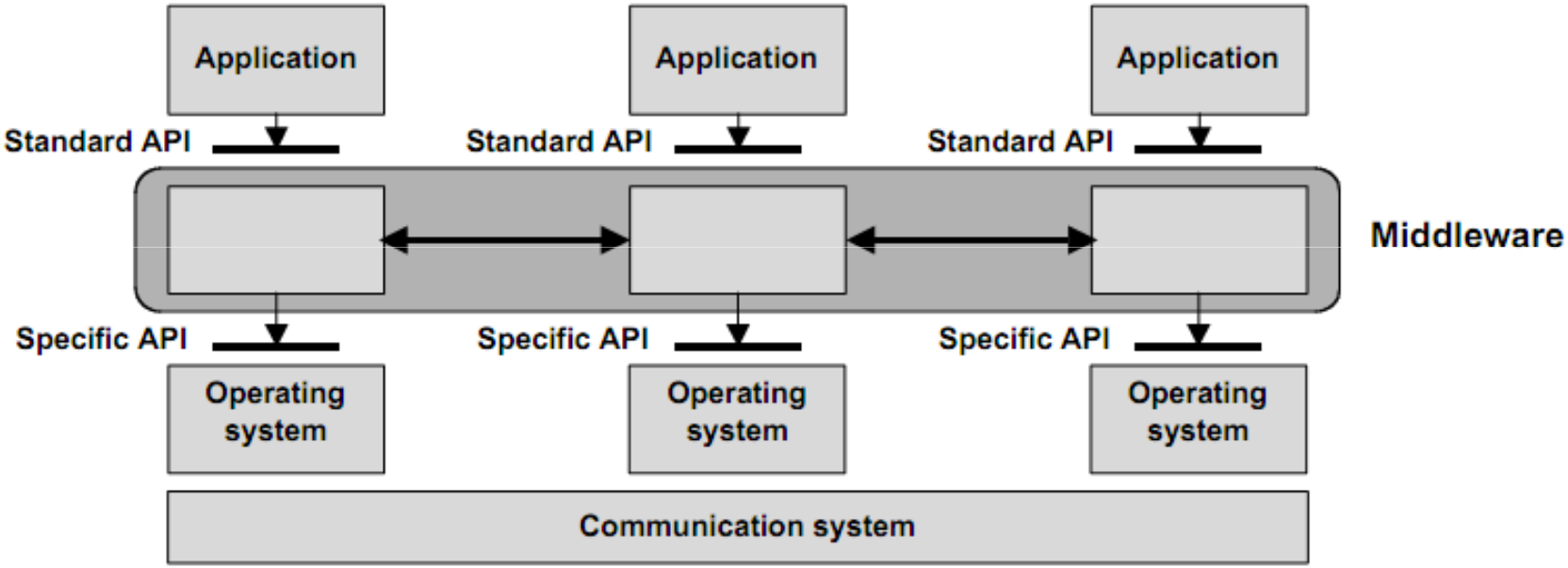
and by hiding low-level programming details.

S. Krakowiak *What is middleware* http://middleware.objectweb.org/

Warstwa pośrednia: umiejscowienie w modelu warstwowym

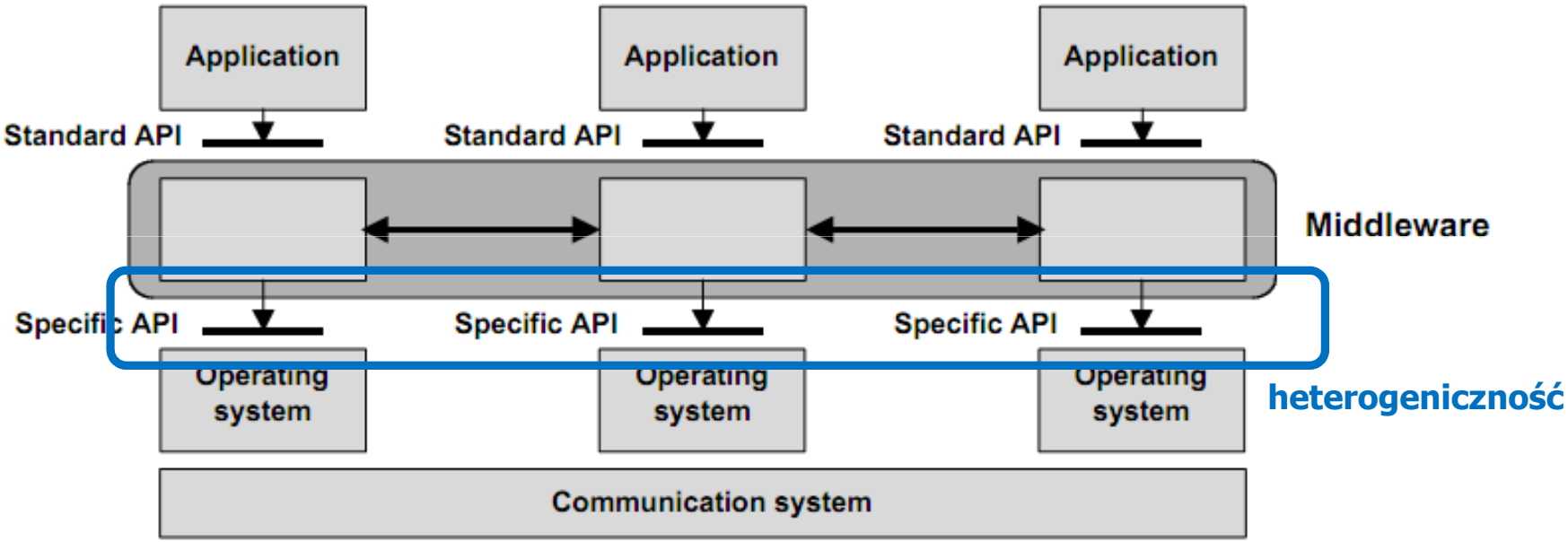


Warstwa pośrednia w systemie rozproszonym



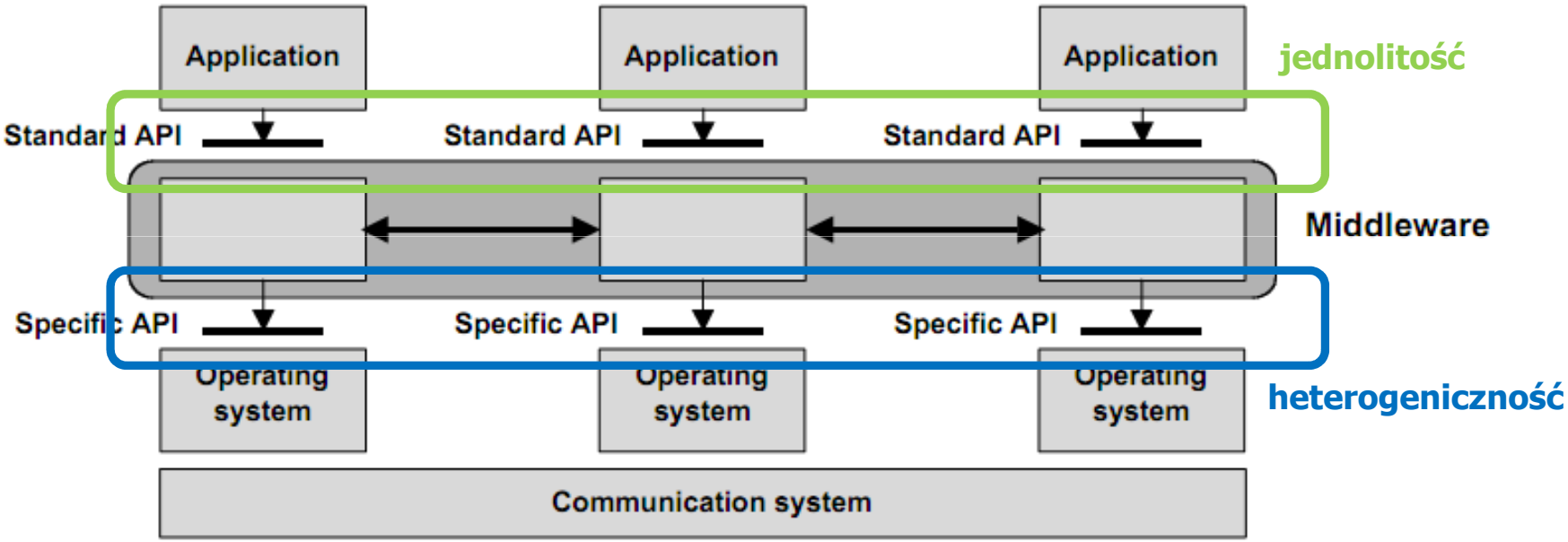
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Warstwa pośrednia w systemie rozproszonym



S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Warstwa pośrednia w systemie rozproszonym

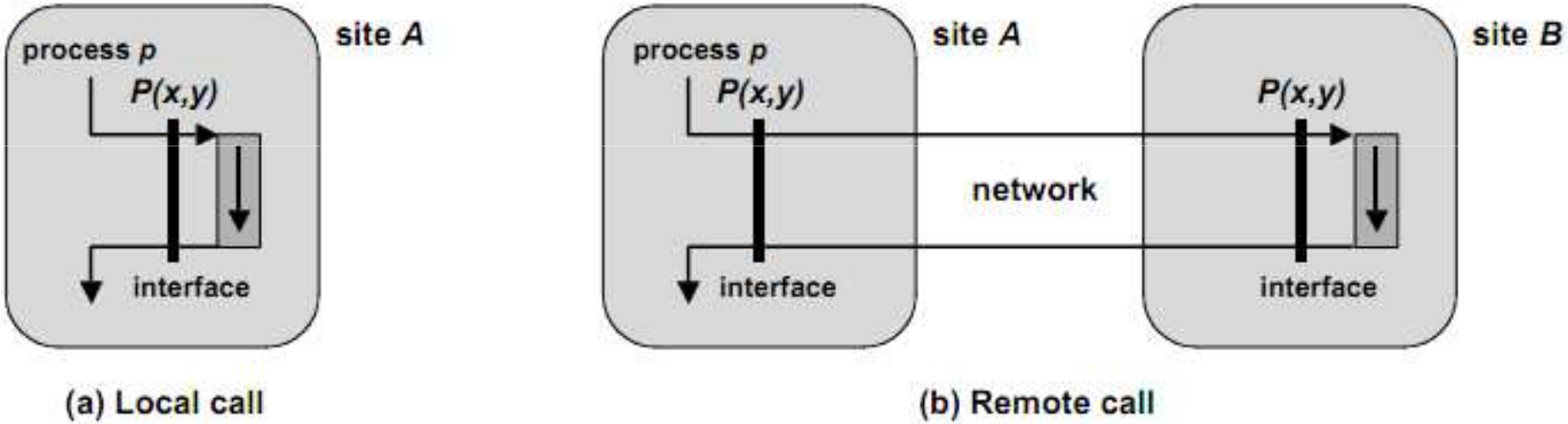


S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Warstwa pośrednia (*middleware*): czego dostarcza?

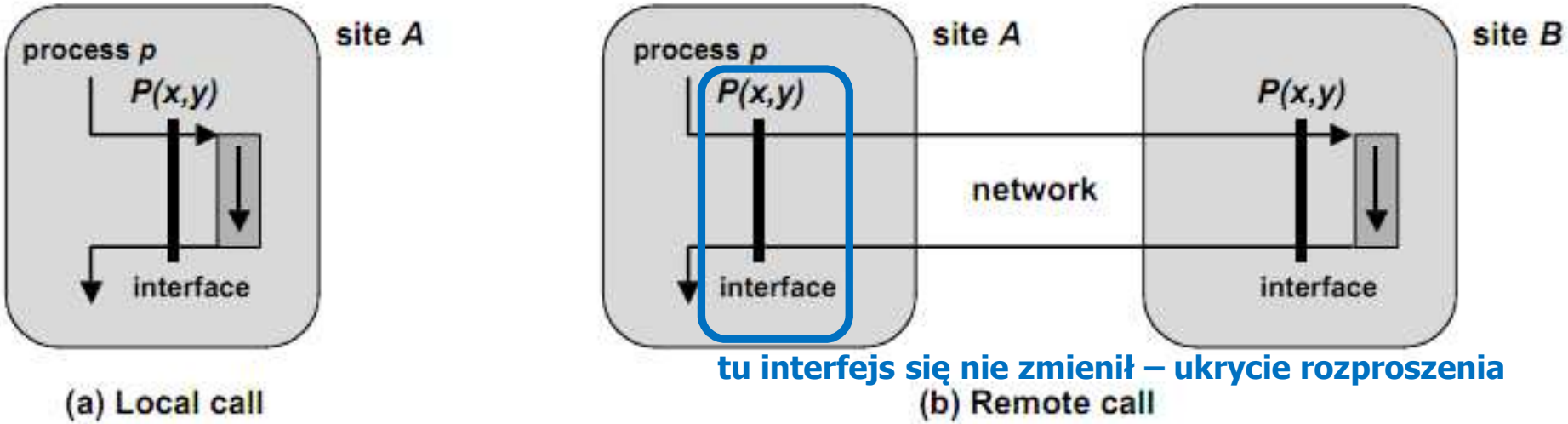
- Warstwa pośrednia podwyższa poziom abstrakcji w „poglądzie na świat" programisty aplikacji.
  - programista nie widzi (może abstrahować od) nieistotnych/niewygodnych szczegółów.
- Przykłady podwyższonego poziomu abstrakcji
  - ukrycie rozproszenia systemu
    - programista widzi cały rozproszony system jak jedną maszynę
  - ukrycie heterogeniczności (różnorodności systemów operacyjnych, zasobów, …)
  - wygodne odkrywanie dostępnych węzłów/zasobów/usług
    - np. programista nie widzi szczegółów „spontanicznego" tworzenia się systemu ad-hoc
  - ukrycie zmienności kontekstu systemowego
    - np. programista nie obsługuje zmian dostępności sieci używanych do komunikowania się elementów aplikacji

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



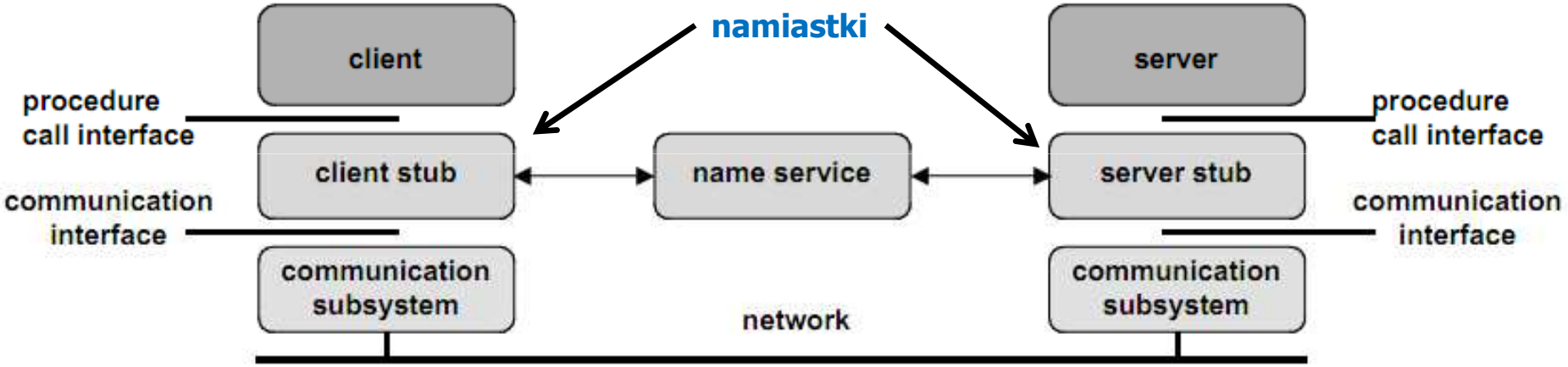
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



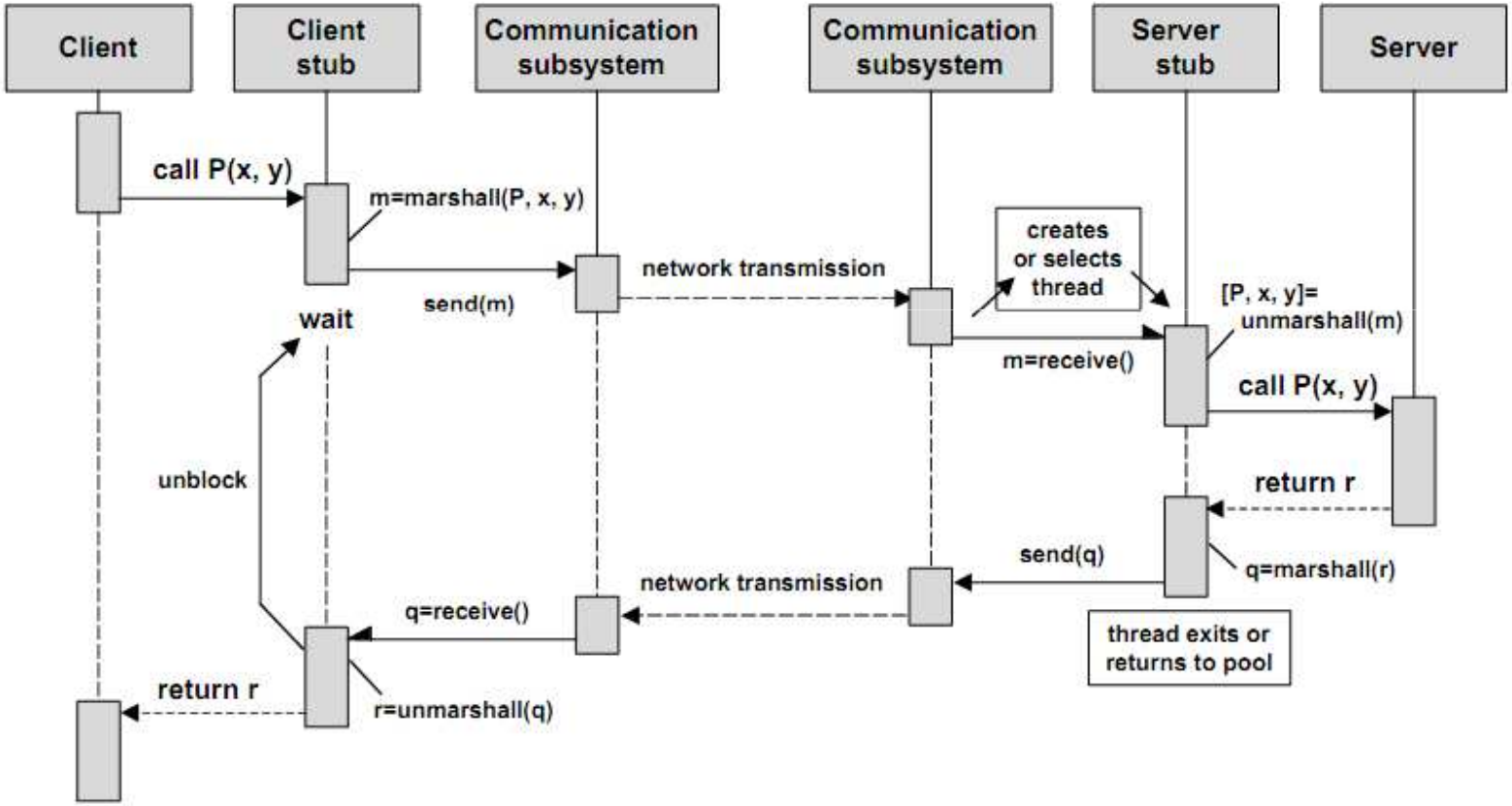
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



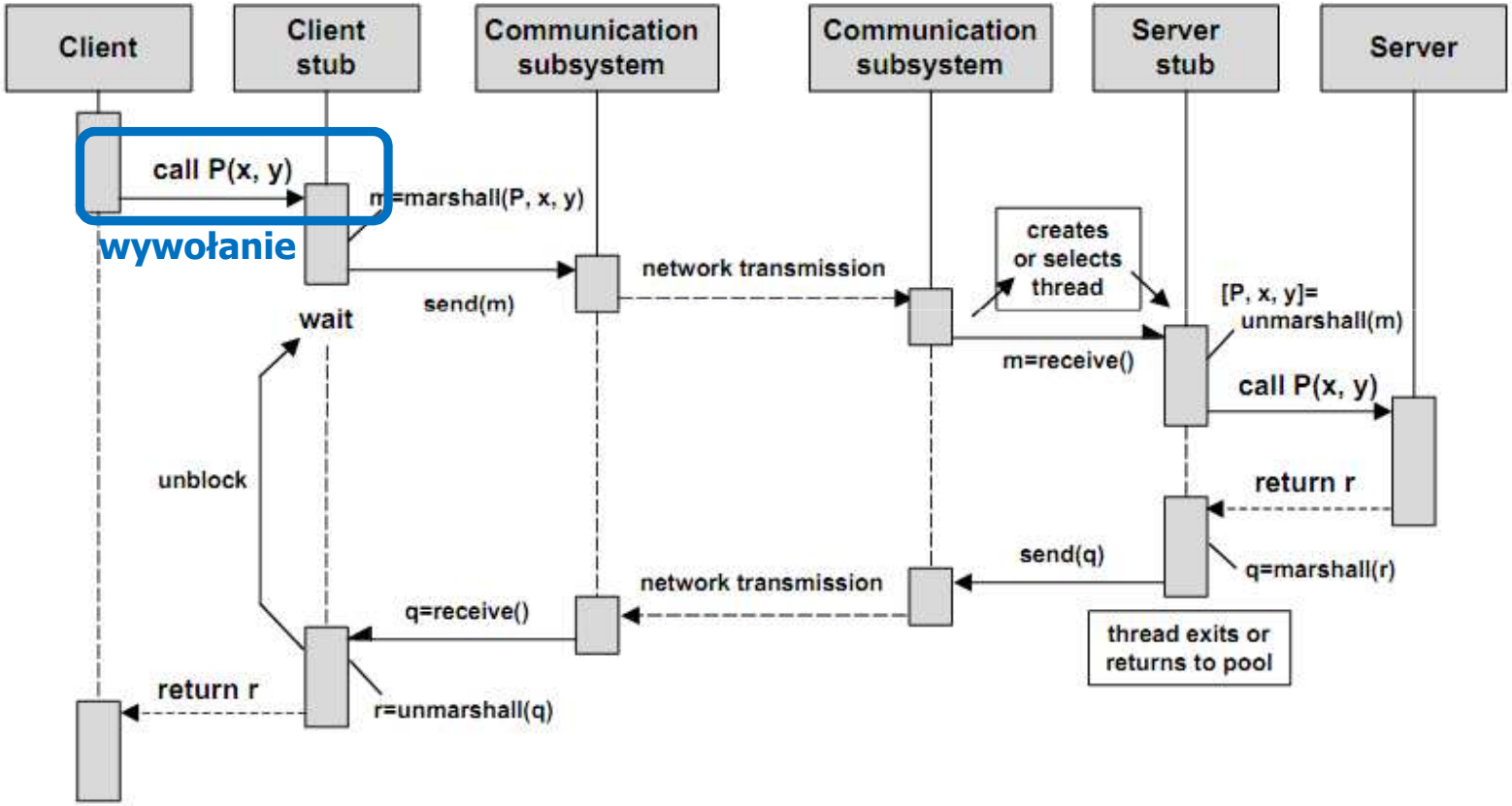
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



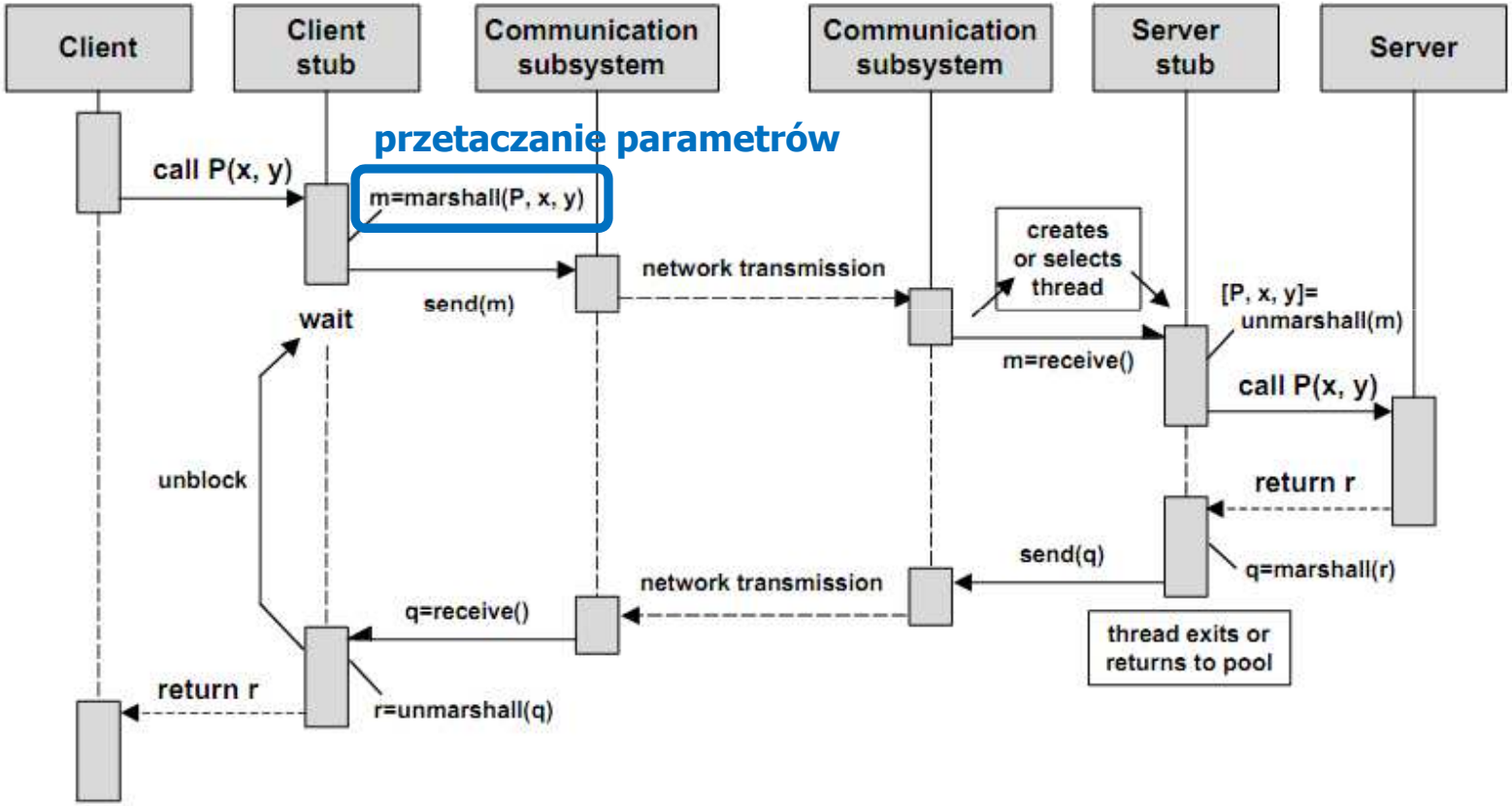
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



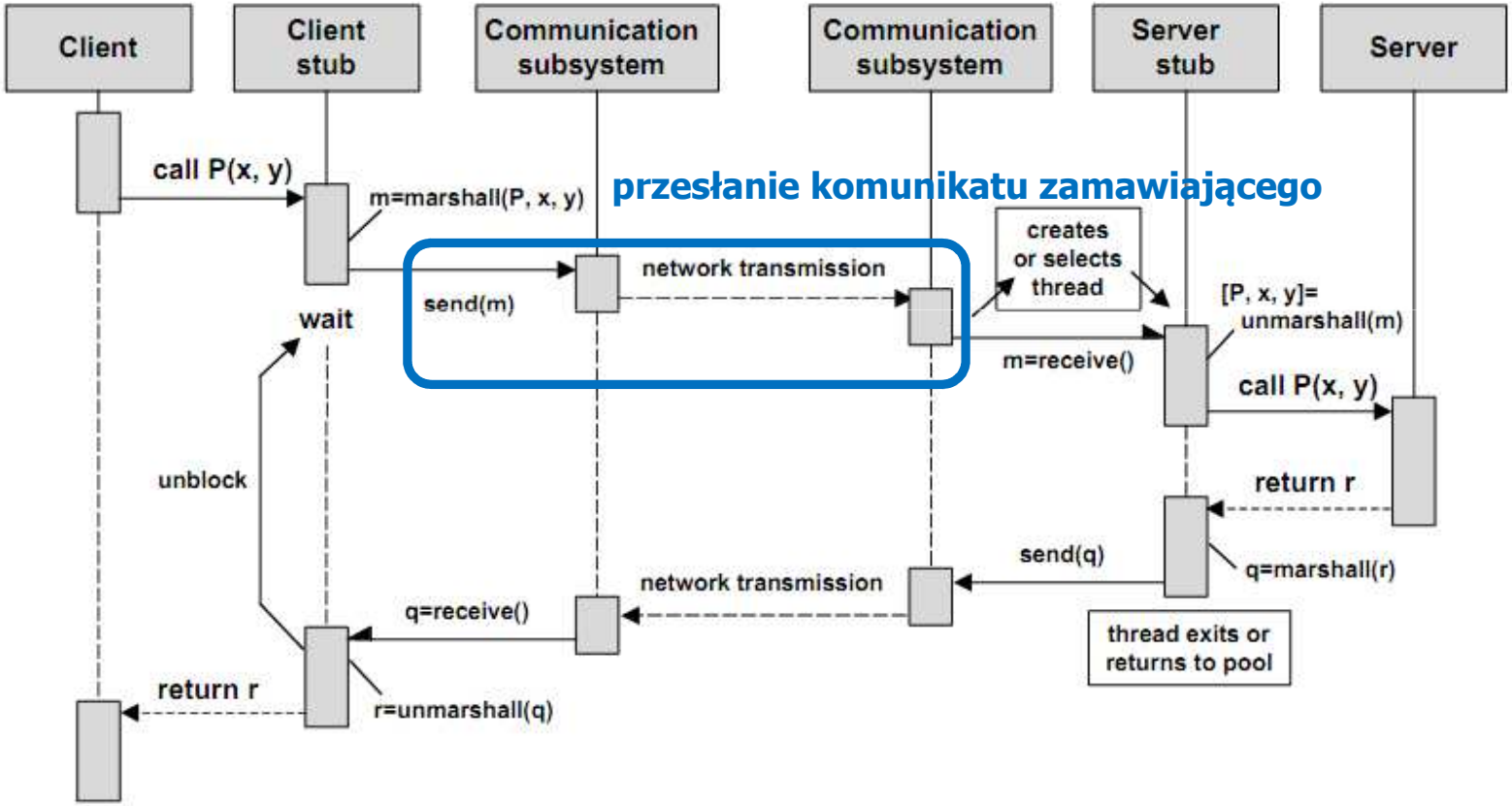
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



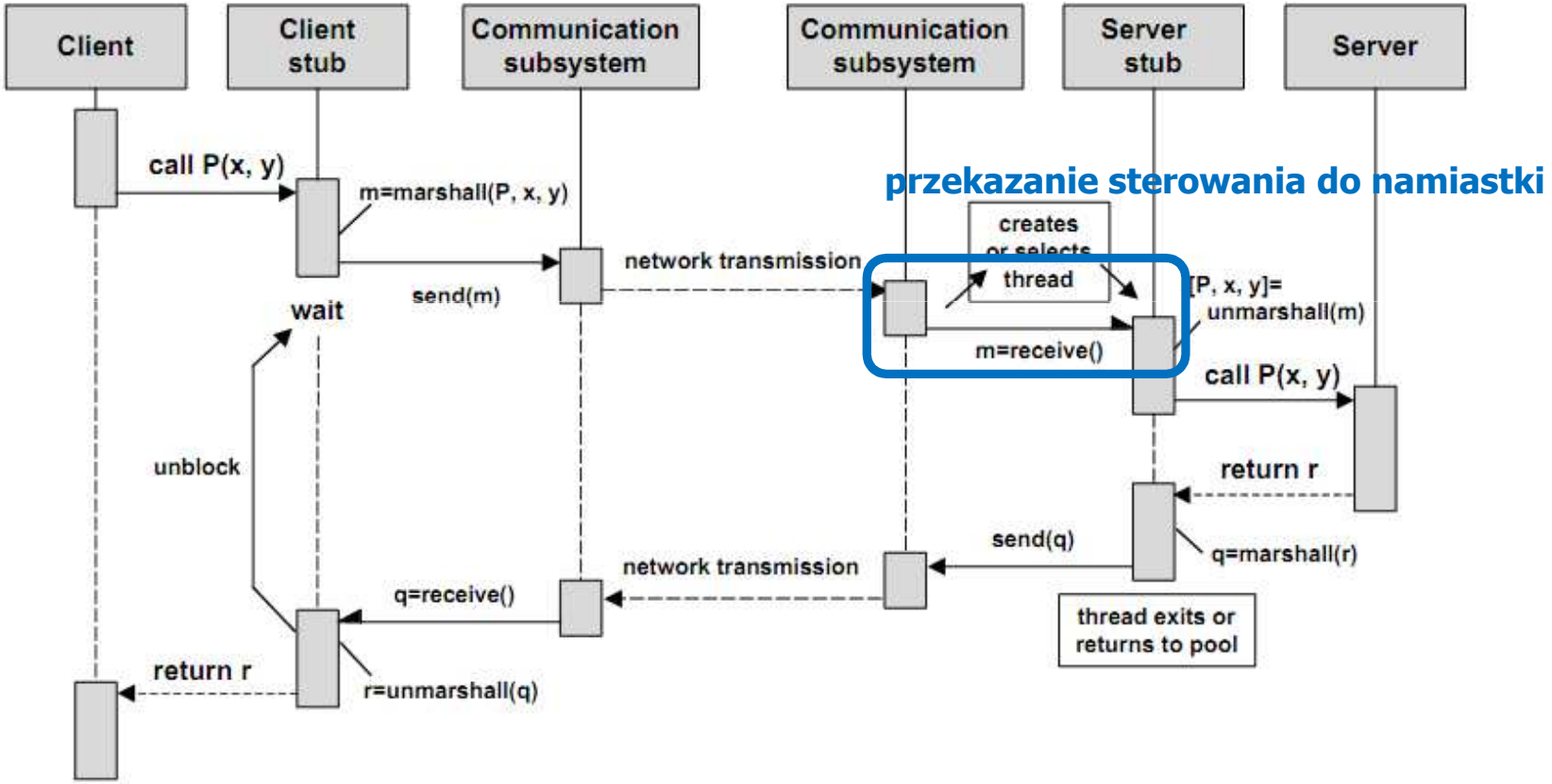
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



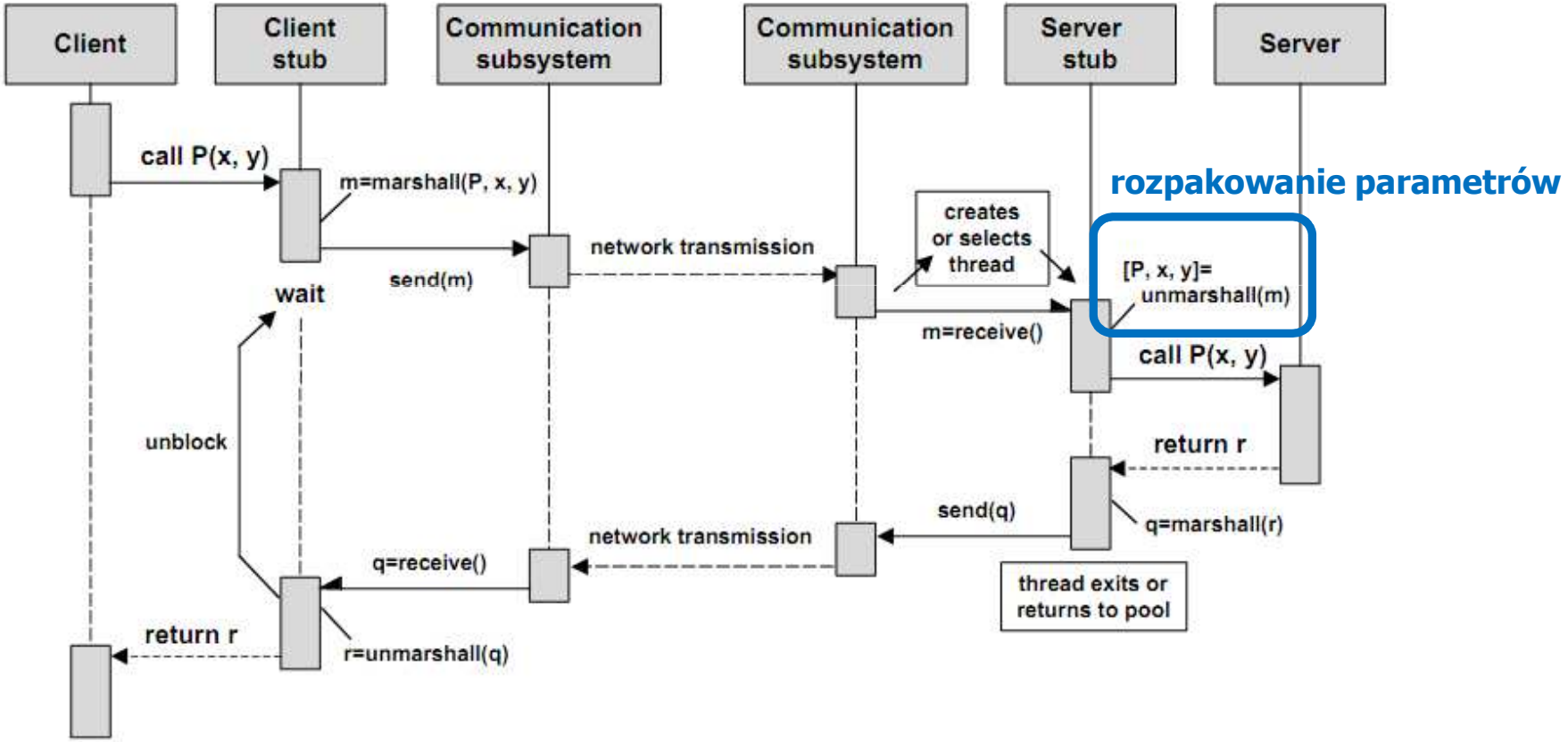
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



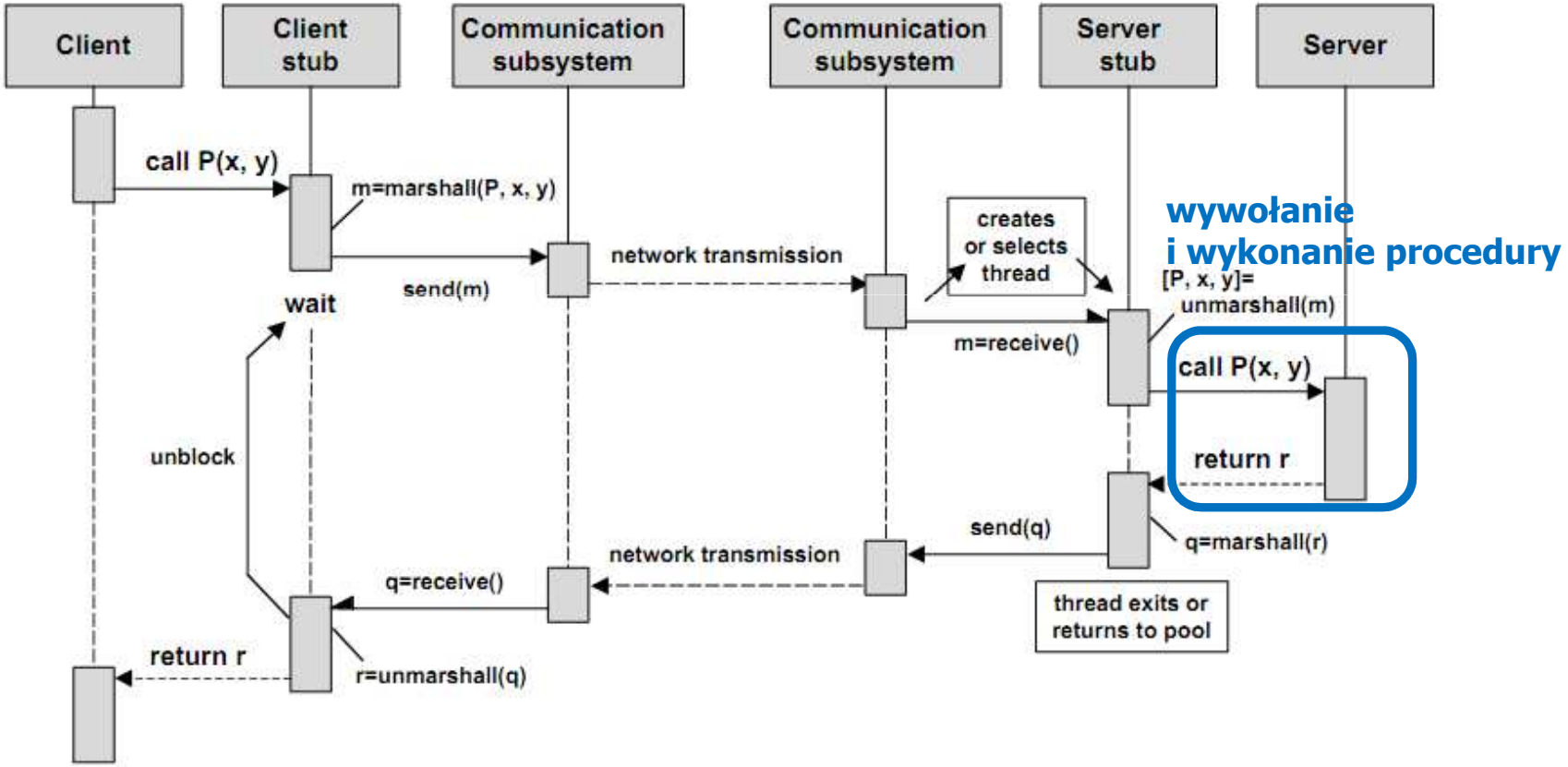
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



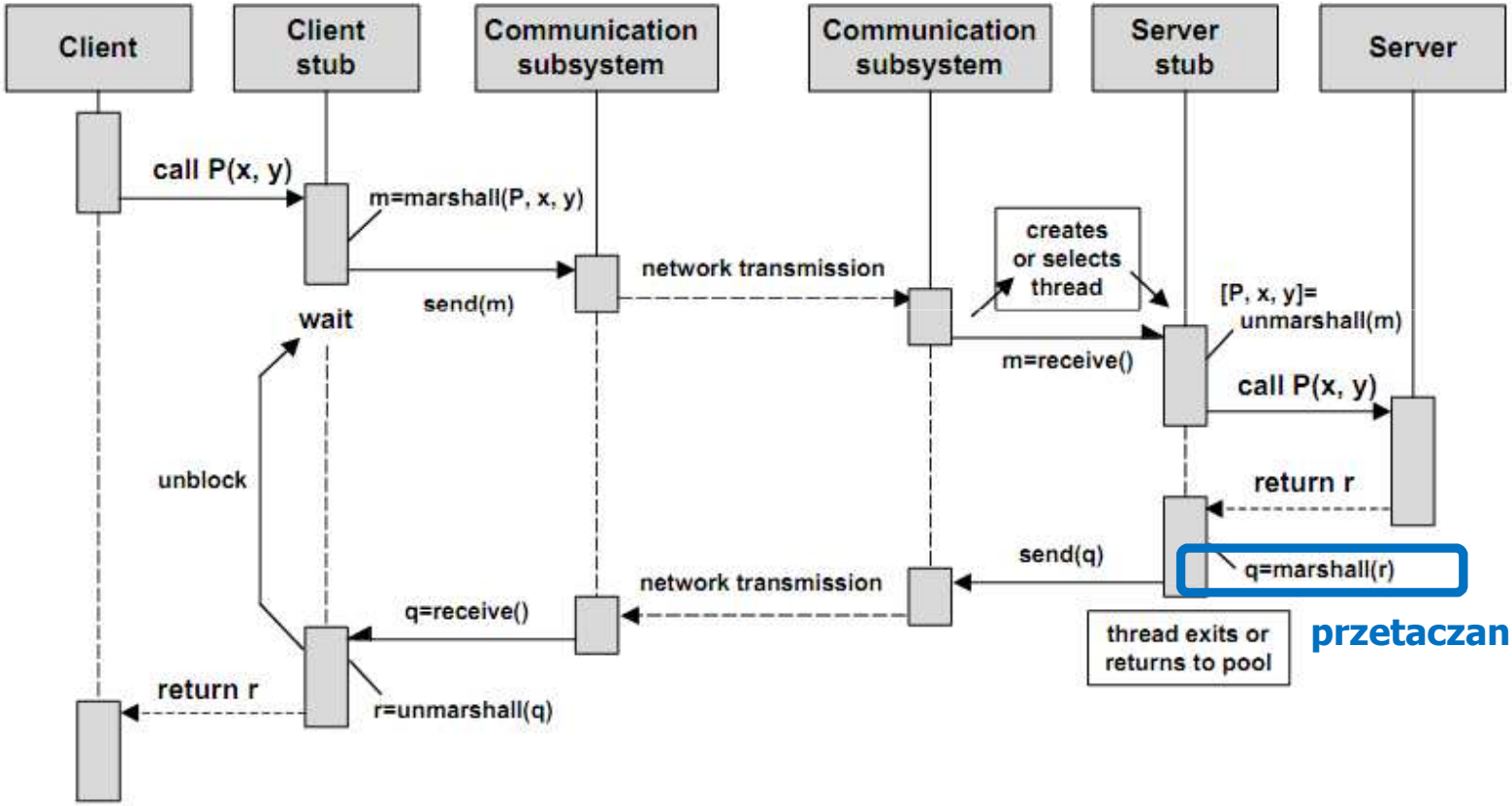
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

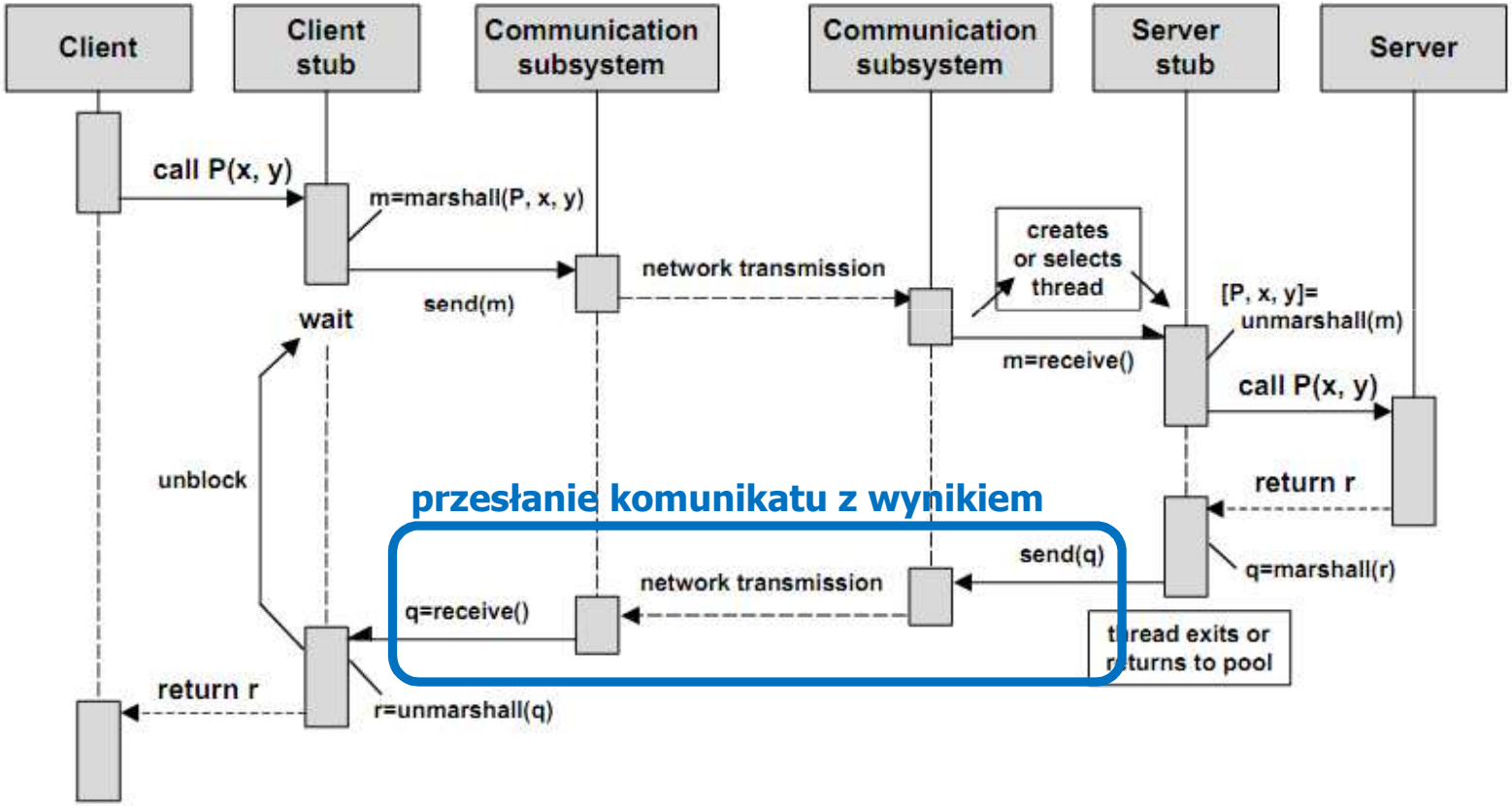
Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



przetaczanie wyniku

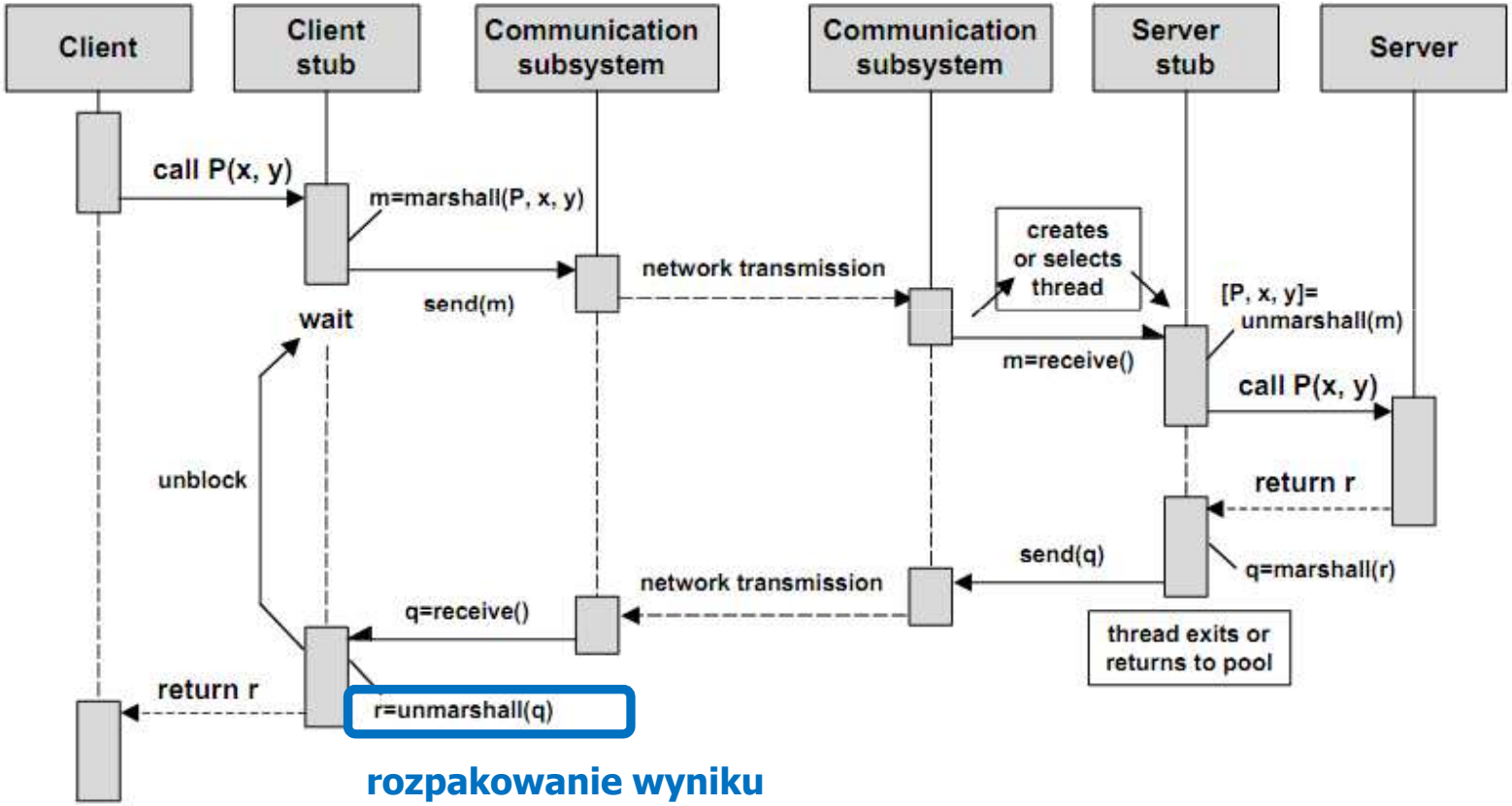
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

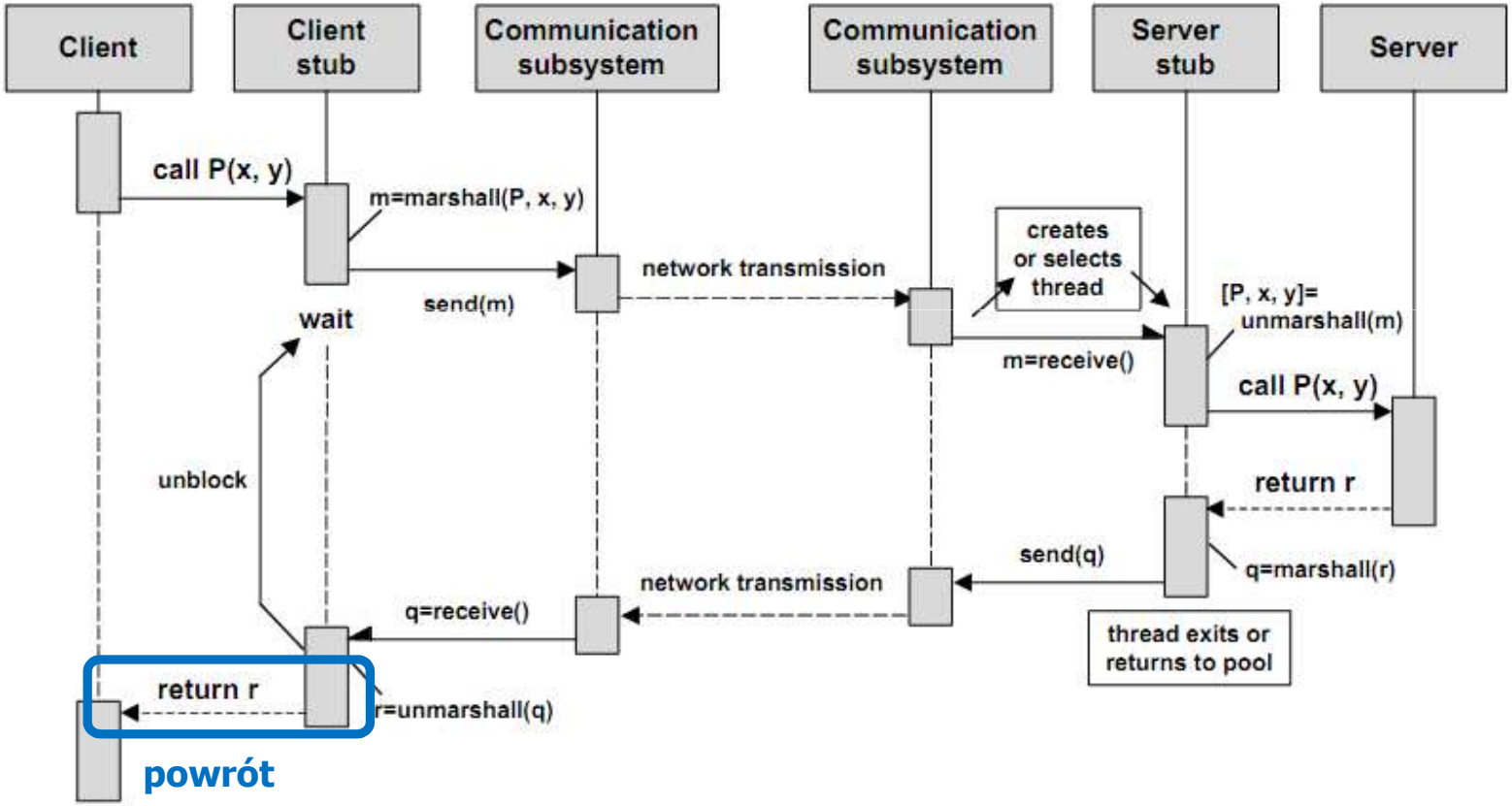
Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



rozpakowanie wyniku

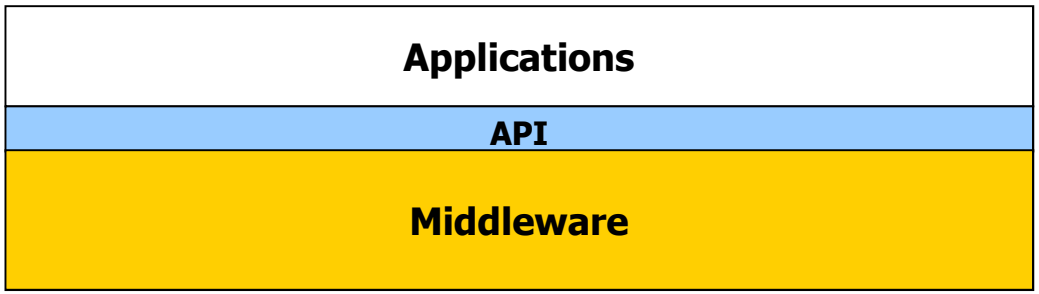
S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Przykład warstwy pośredniej: RPC (*Remote Procedure Call*)



S. Krakowiak *Middleware Architecture with Patterns and Frameworks*

Specyfikacja usług: API/model programistyczny



- API specyfikuje usługi warstwy pośredniej dla warstwy aplikacji
  - poszczególne usługi, dostępne za pomocą wywołań, nazywamy **prymitywami**
- Sposób programowania aplikacji określony przez API warstwy pośredniej nazwiemy jej **modelem programistycznym**
  - czasem mówimy o **abstrakcjach programistycznych**
- API/model programistyczny to „esencja” warstwy pośredniej
  - czasem, jak w przypadku RPC, API nie odgrywa głównej roli

Niektóre problemy z warstwami pośrednimi

- Wydajność.
  - czynniki prowadzące do suboptymalności oprogramowania …
    - dekompozycja (struktura warstwowa)
    - dążenie do ponownego użycia (tj. warunek, aby warstwa pośrednia była wykorzystywana przez wiele aplikacji)
    - wysoki poziom abstrakcji API/modelu programistycznego

- Dana warstwa pośrednia może nie być powszechnie przyjęta.
  - „nie ma z kim współpracować”
  - czy warto inwestować w poznanie nowego API/modelu programistycznego?

WARSTWY POŚREDNIE W SYSTEMACH KONTEKSTOWYCH

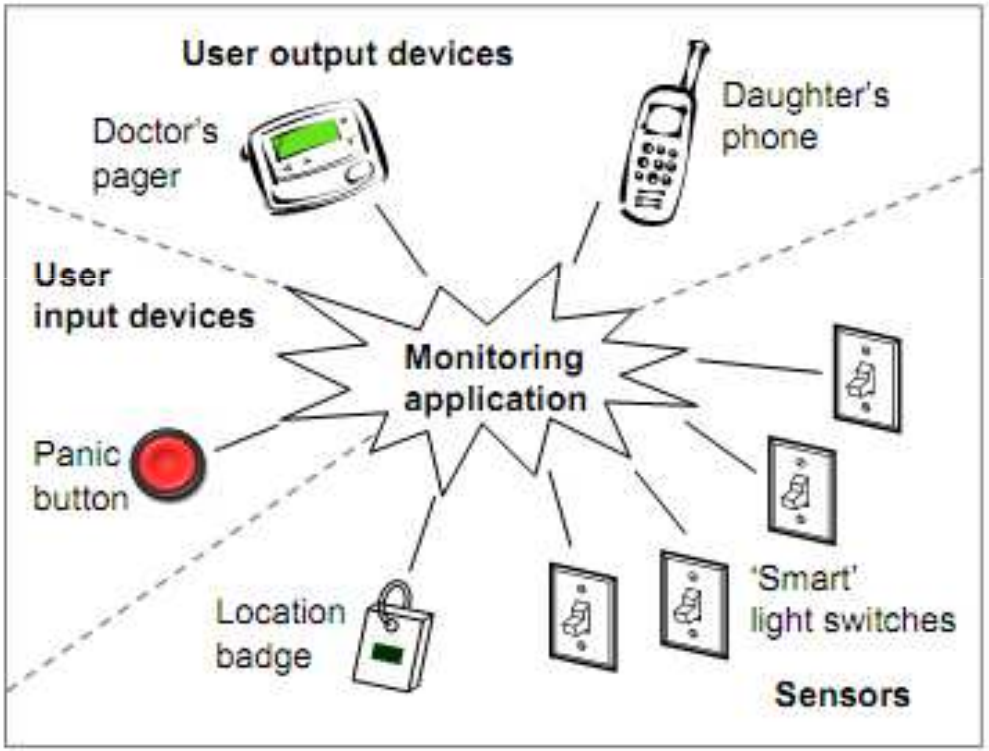
Dlaczego o tym mówimy?

- Większość prac R&D w dziedzinie aplikacji kontekstowych koncentruje się nie na samych aplikacjach, ale na budowie warstw pośrednich (platform) ułatwiających tworzenie aplikacji

- Często łatwiej jest zbudować warstwę pośrednią niż wpaść na pomysł dobrej aplikacji!

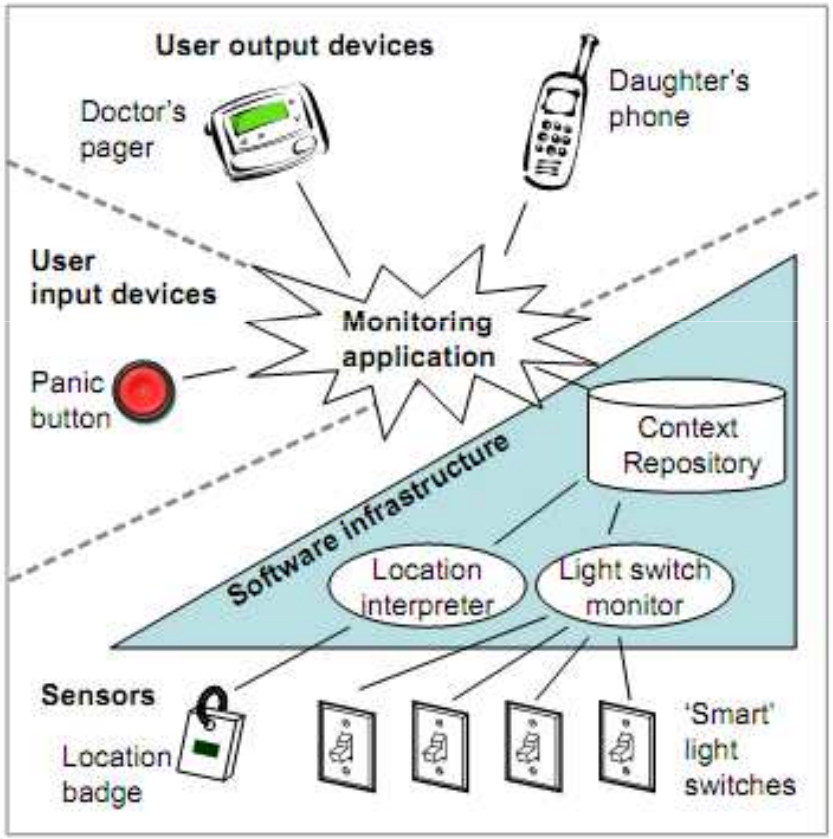
- Warto jednak poznawać warstwy pośrednie, gdyż sprzyja to dobrym nawykom w projektowaniu oprogramowania
  - ponowne użycie
  - podejście komponentowe
  - projektowanie wygodnych w użyciu interfejsów/modeli programistycznych

Budowa aplikacji kontekstowej: podejście „naiwne”



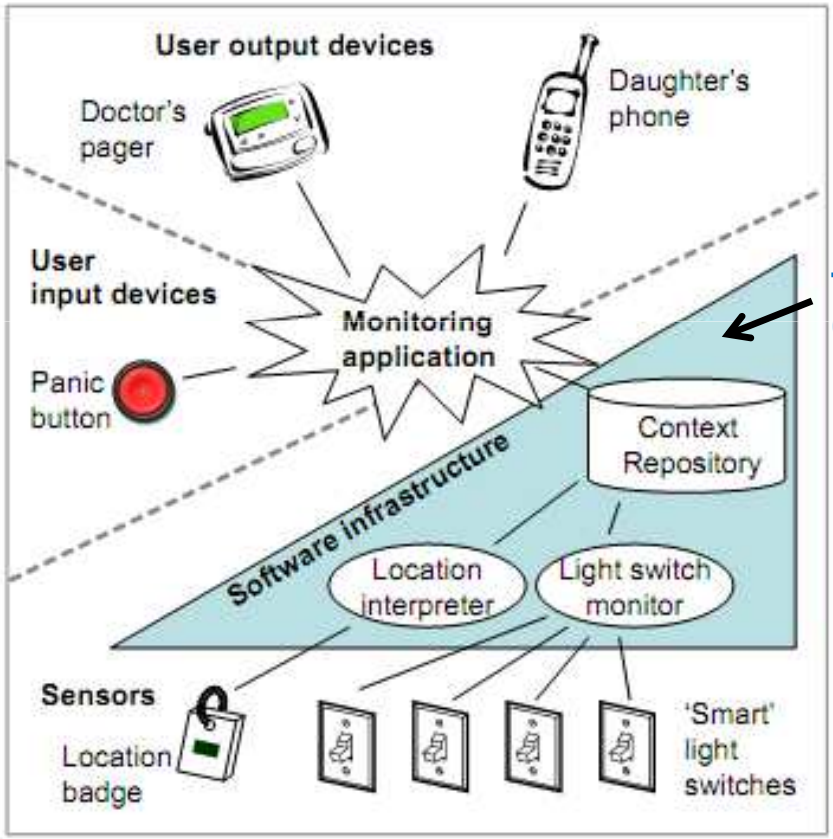
J. Indulska , K.Henricksen *Context-Awareness*

Budowa aplikacji kontekstowej: podejście z warstwą pośrednią



J. Indulska , K.Henricksen *Context-Awareness*

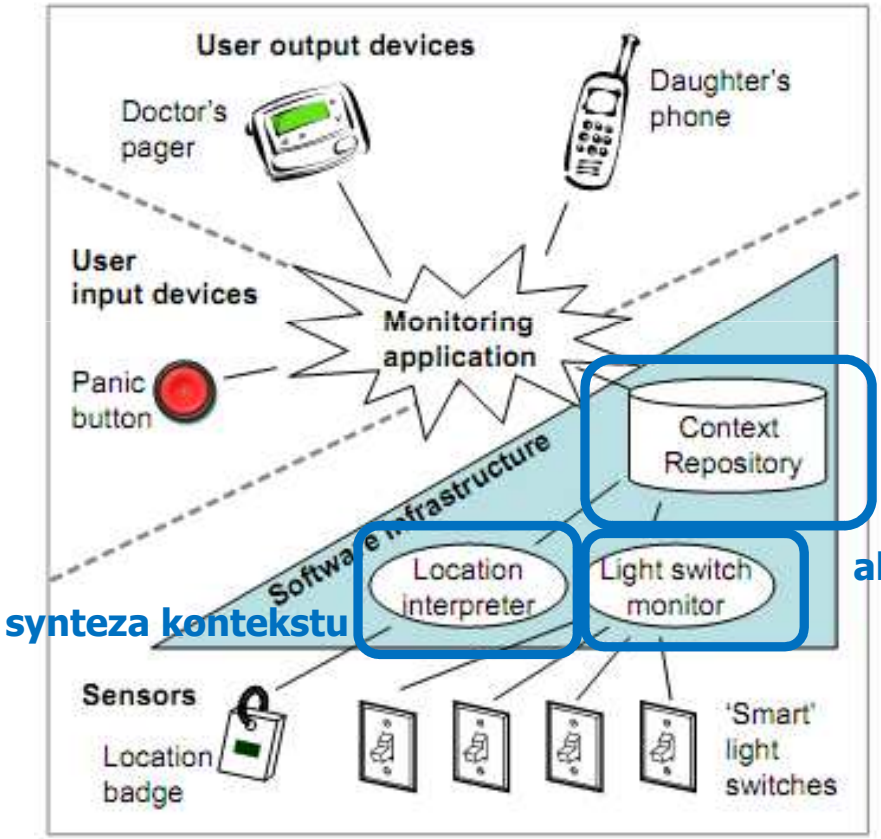
Budowa aplikacji kontekstowej: podejście z warstwą pośrednią



to może być ponownie użyte (przez inne aplikacje)!

J. Indulska , K.Henricksen *Context-Awareness*

Budowa aplikacji kontekstowej: podejście z warstwą pośrednią



przechowywanie i dystrybucja kontekstu

akwizycja danych sensorowych

synteza kontekstu

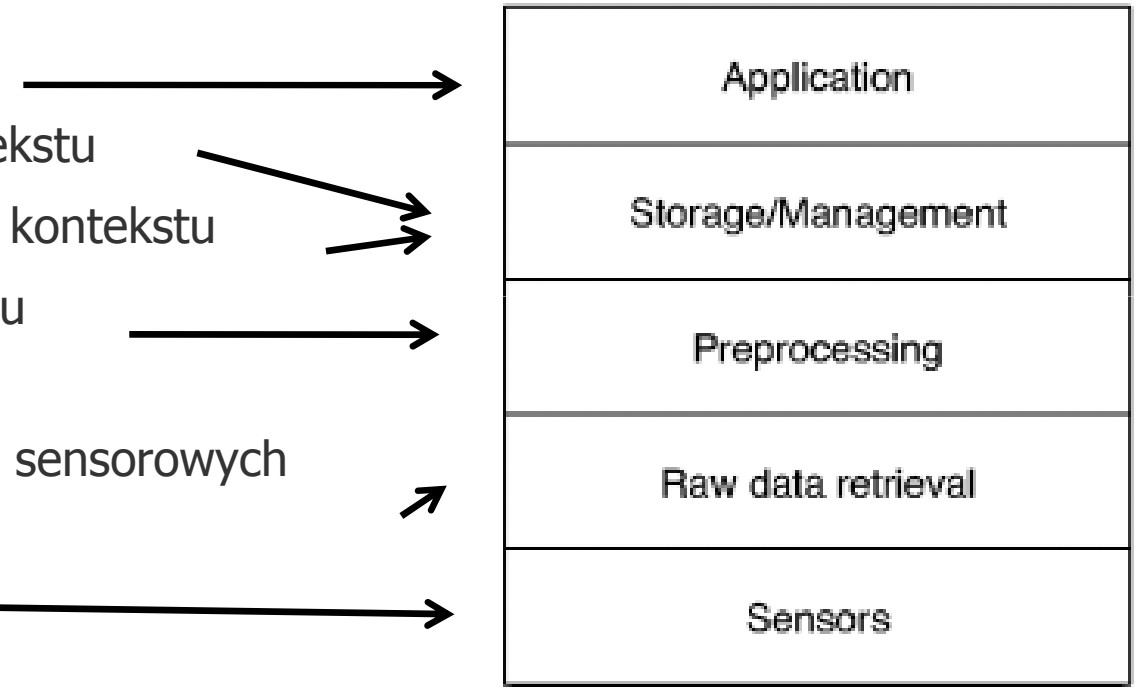
J. Indulska , K.Henricksen *Context-Awareness*

Czego dostarcza warstwa pośrednia w systemie kontekstowym

- Model kontekstu
  - nadaje strukturę repozytorium kontekstu, wpływa na API
- Wykrywanie źródeł kontekstu
- Akwizycja danych z sensorów
  - sensory fizyczne, wirtualne, profile, …
- Przechowywanie i dystrybucja informacji kontekstowej
  - repozytorium kontekstu
- Synteza kontekstu
  - sensory logiczne
- Udostępnianie informacji kontekstowej aplikacjom
  - API/model programistyczny

Warstwowa struktura aplikacji kontekstowej

- Warstwa logiki aplikacji
- Warstwa dystrybucji kontekstu
- Warstwa przechowywania kontekstu
- Warstwa syntezy kontekstu
  - sensory logiczne
- Warstwa akwizycji danych sensorowych
- Warstwa sensorów
  - sensory fizyczne
  - sensory wirtualne



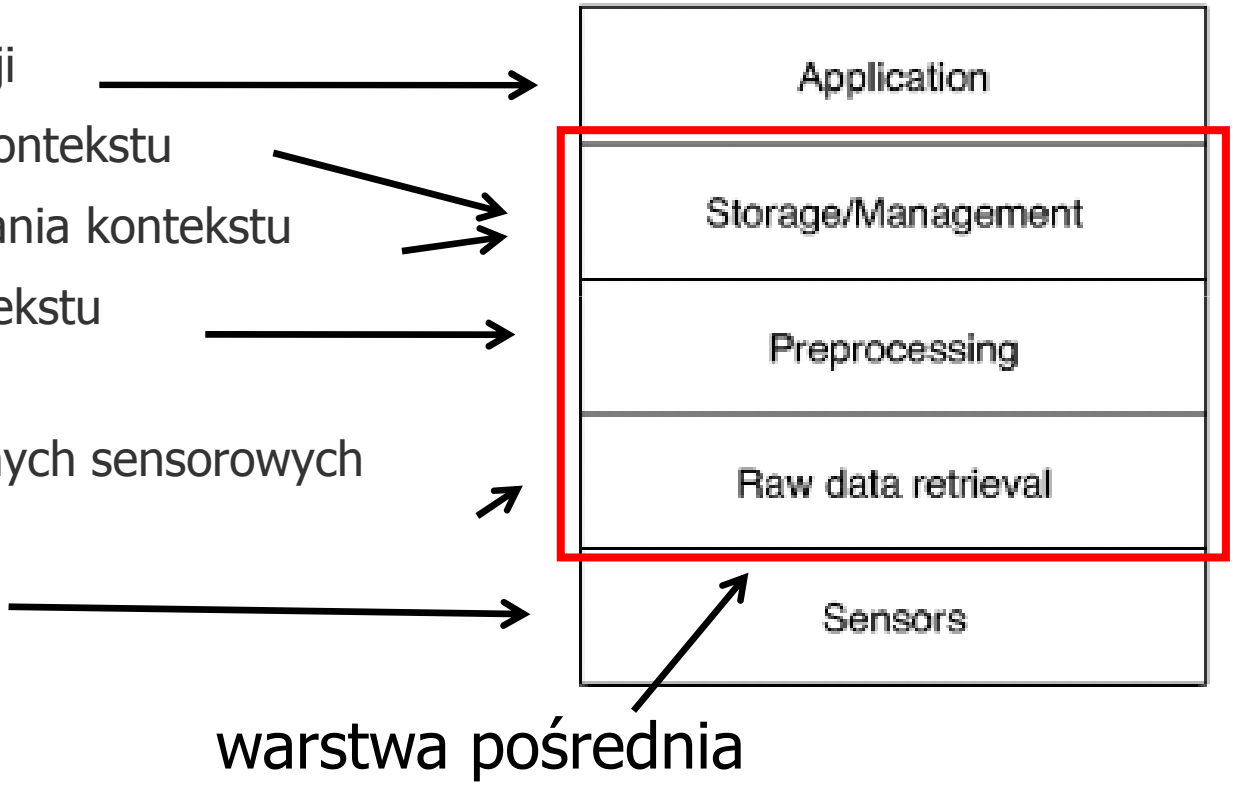
Które warstwy włączylibyśmy do warstwy pośredniej?

M. Baldauf et al. *A survey on context-aware systems*

PROJEKT!

Warstwa pośrednia w systemie kontekstowym

- Warstwa logiki aplikacji
- Warstwa dystrybucji kontekstu
- Warstwa przechowywania kontekstu
- Warstwa syntezy kontekstu
  - sensory logiczne
- Warstwa akwizycji danych sensorowych
- Warstwa sensorów
  - sensory fizyczne
  - sensory wirtualne



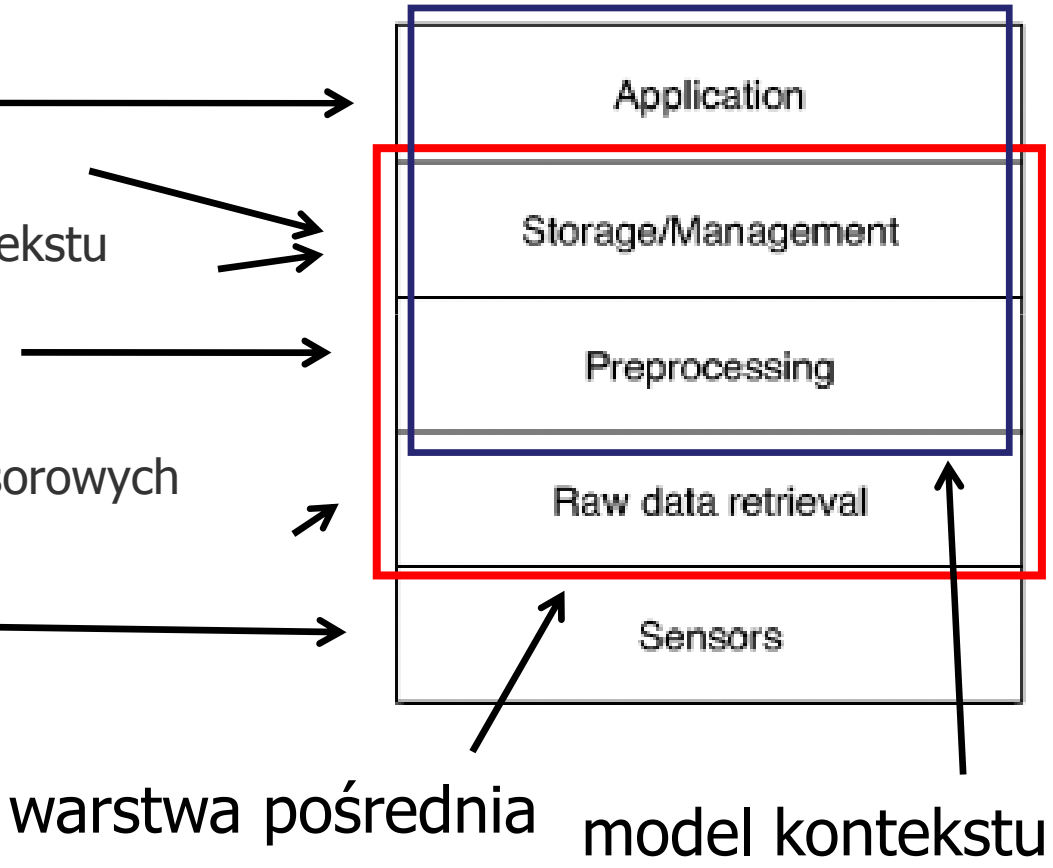
Które warstwy wykorzystują model kontekstu?

M. Baldauf et al. *A survey on context-aware systems*

PROJEKT!

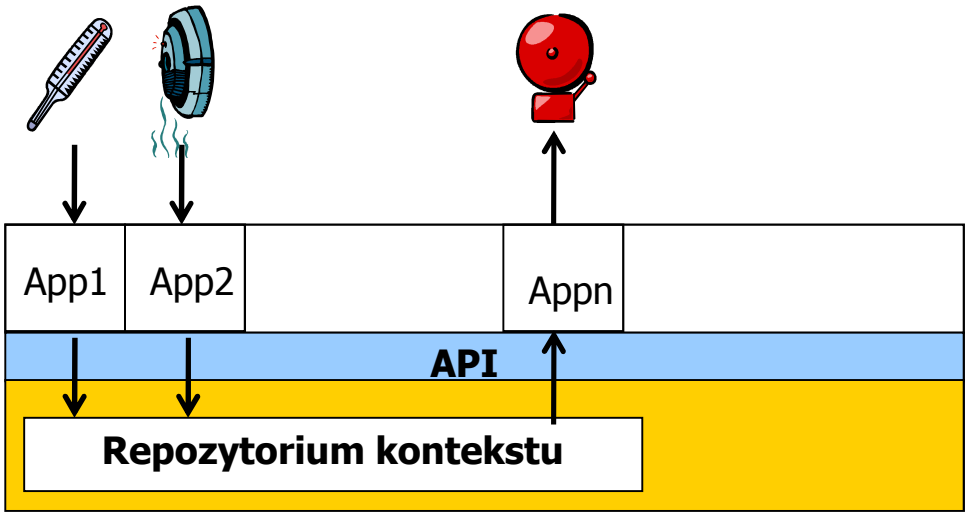
Model kontekstu systemie kontekstowym

- Warstwa logiki aplikacji
- Warstwa dystrybucji kontekstu
- Warstwa przechowywania kontekstu
- Warstwa syntezy kontekstu
  - sensory logiczne
- Warstwa akwizycji danych sensorowych
- Warstwa sensorów
  - sensory fizyczne
  - sensory wirtualne



M. Baldauf et al. *A survey on context-aware systems*

„Odwrócone" podejście do akwizycji danych sensorowych



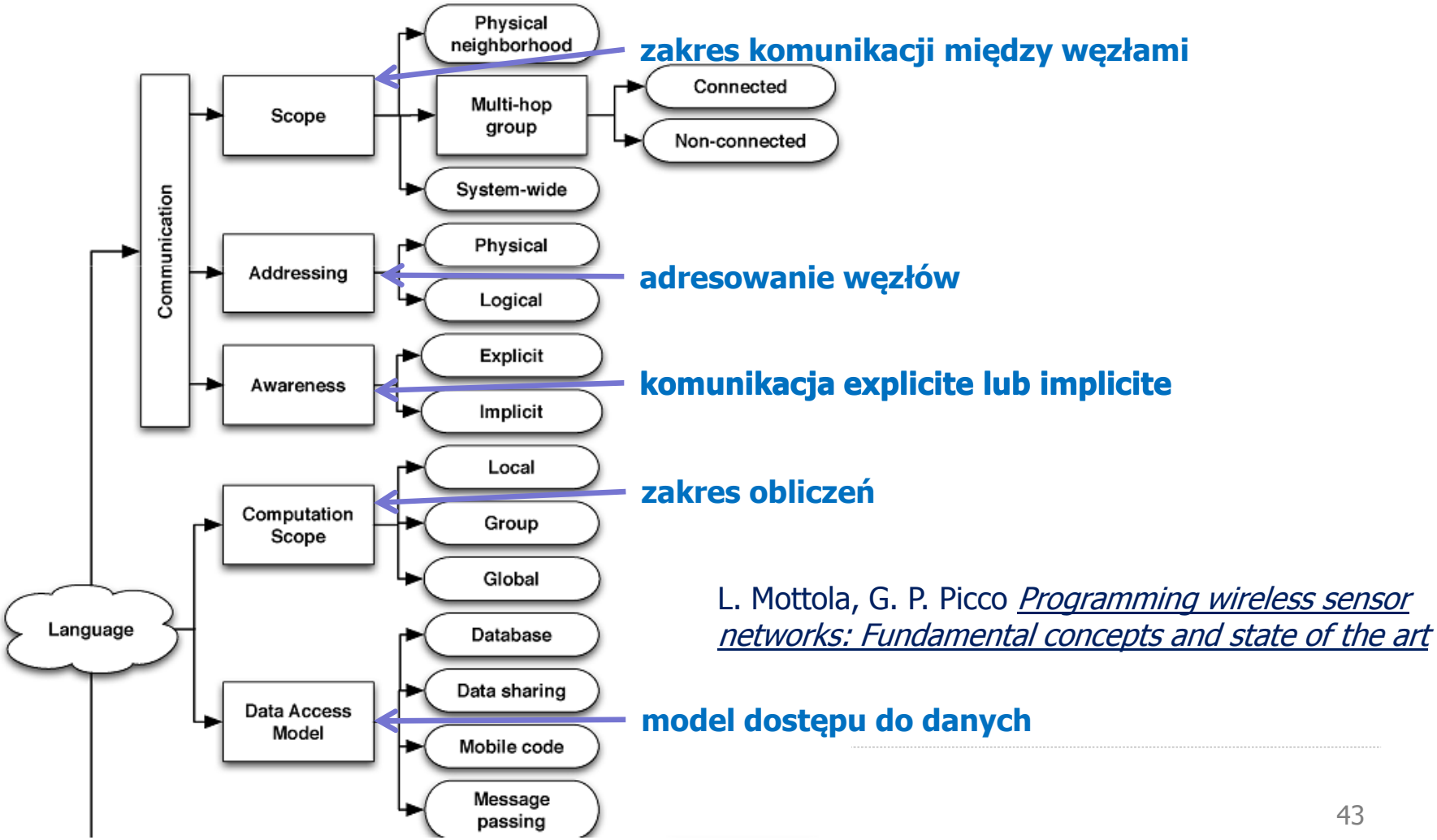
- Możliwość dodawania informacji kontekstowej przez aplikacje
  - aplikacje jako sterowniki do sensorów
- Informacja kontekstowa „wstrzykiwana" do repozytorium kontekstu w warstwie pośredniej, za pomocą jej API

Założenia co do dostępnych sensorów i elem. wykonawczych

- Przy projektowaniu platformy czynimy założenia co do dostępnych sensorów, elementów wykonawczych, punktów dostępowych, …
- Bez infrastruktury
  - nie wiemy, jakie sensory i elementy wykonawcze będą dostępne
  - stosowne aplikacje czasem nazywamy **oportunistycznymi**
  - np. POBICOS
- Z infrastrukturą
  - **smart space**, **active space**, **intelligent space**
- Z **infrastrukturą błyskawiczną** (instant)
  - błyskawiczna instalacja infrastruktury dla krótkotrwałego zastosowania w konkretnej sytuacji (np. na czas trwania imprezy masowej)

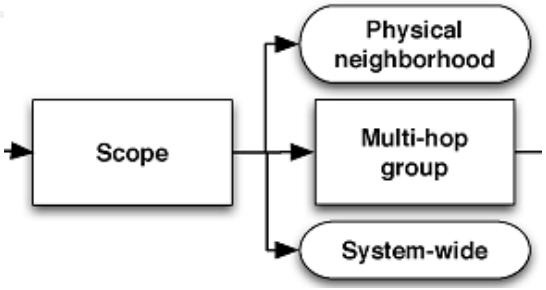
PODSTAWOWE INFORMACJE O MODELACH PROGRAMISTYCZNYCH

Przykładowa klasyfikacja modeli programistycznych



L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*

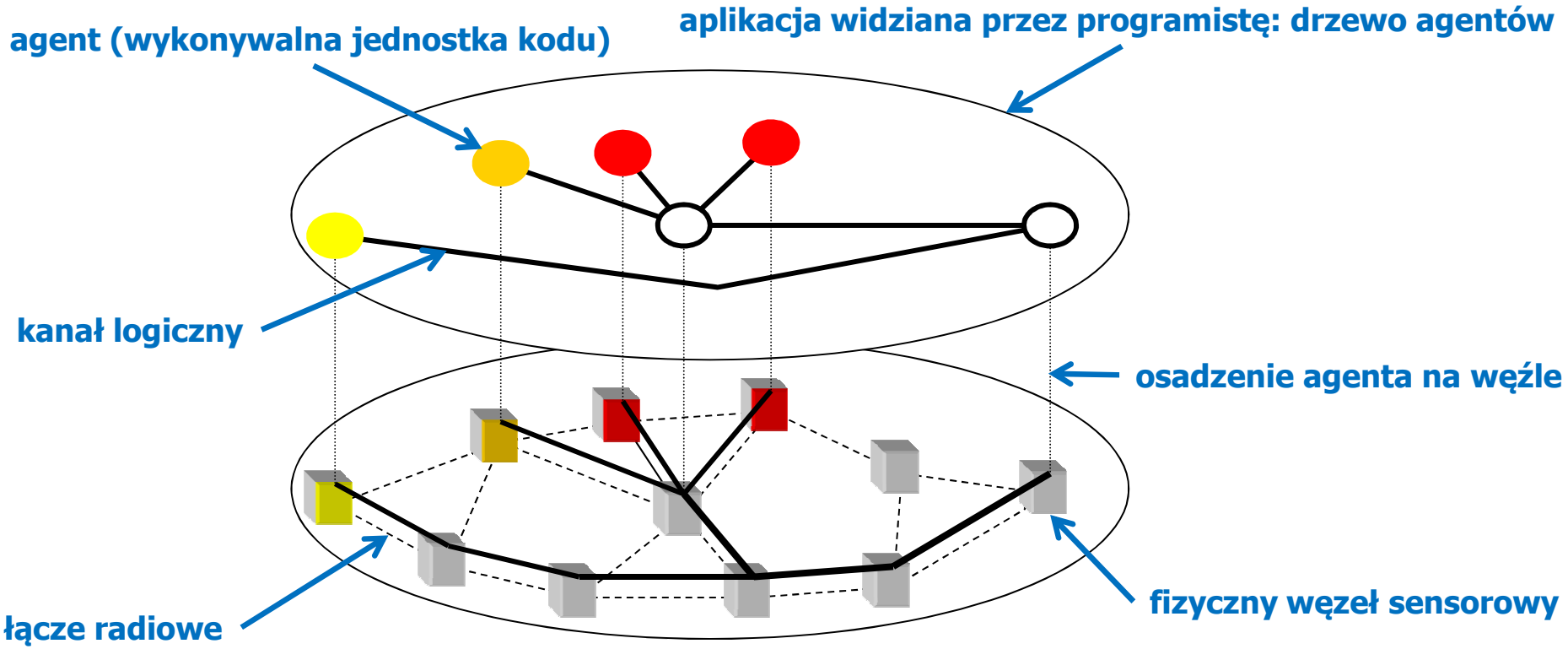
L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*



Zakres komunikacji między węzłami

- Jakie udogodnienia komunikacyjne oferuje warstwa pośrednia?
  - z którymi węzłami łatwo zaprogramować wymianę informacji?
- (a) Sąsiedztwo „fizyczne” – zasięg radiowy (1 hop)
- (b) Grupa
  - dopuszcza się „odległość” między węzłami w grupie > 1hop
  - możliwe różne sposoby określania grupy
    - np. według własności węzłów (zasoby, kontekst)
  - możliwe różne sposoby komunikowania się w obrębie grupy
    - np. rozgłaszanie do wszystkich węzłów w grupie (analogia do rozgłaszania „fizycznego”)
    - np. według logicznej struktury węzłów w grupie (patrz następny slajd)
- (c) Wszystkie węzły sieci
  - np. jedno zapytanie zbiera odczyty sensorów temperatury ze wszystkich węzłów

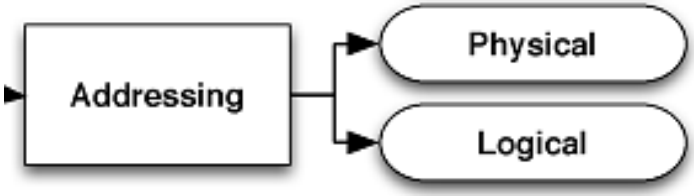
Przykład: POBICOS – komunikacja w grupie



- W POBICOS API dostępne wygodne prymitywy do przesyłania wiadomości kanałem logicznym, tj. między rodzicem a dzieckiem w drzewie agentów

POBICOS zapewnia komunikację w grupie tworzącej drzewo aplikacji.

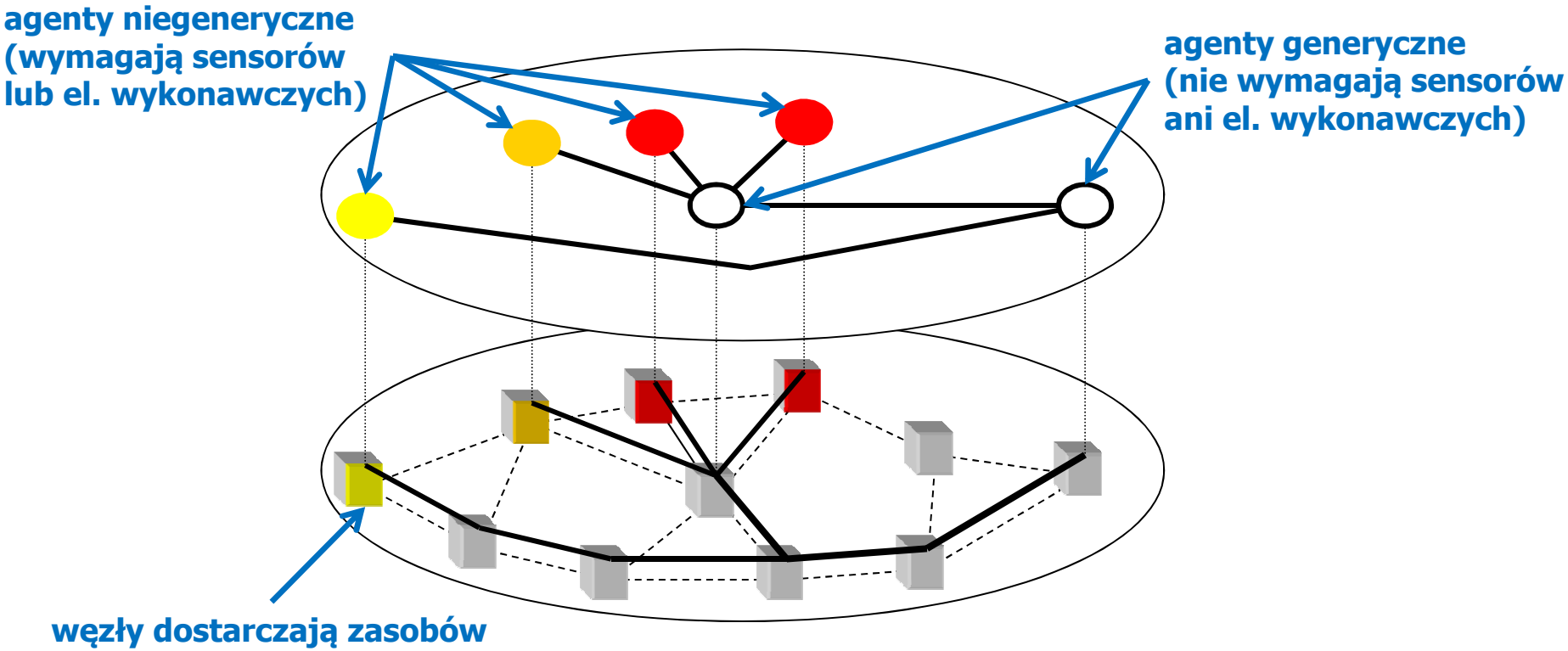
Adresowanie węzłów



L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*

- Adres fizyczny (identyfikator, id)
- „Adres" logiczny – na podstawie własności węzła
  - adresowanie wg zasobów węzła
  - adresowanie wg kontekstu węzła
    - np. na podstawie bieżącego odczytu z sensora na węźle
    - adresat (lub zbiór adresatów) może się zmienić bez zmiany adresu

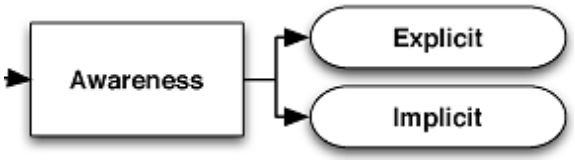
Przykład: POBICOS – adresowanie wg zasobów



- Wywołanie prymitywu POBICOS API do utworzenia agenta niegenerycznego powoduje samoczynne osadzenie go na węźle z odpowiednimi zasobami

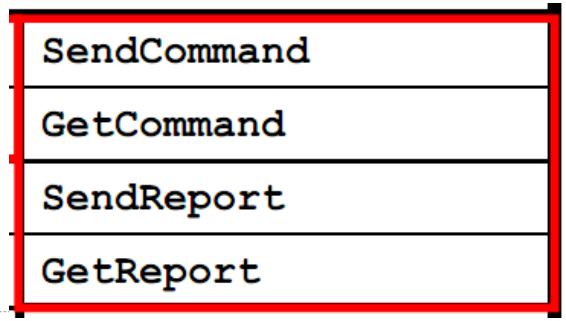
POBICOS osadza agenty na węzłach wg dostępnych zasobów sensorycznych i wykonawczych.

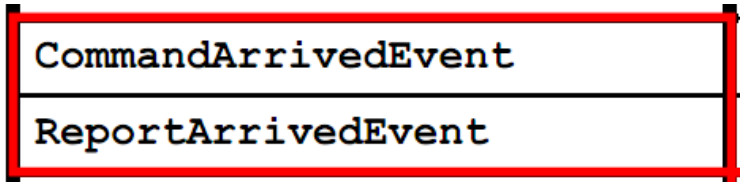
Komunikacja wyrażona explicite lub implicite



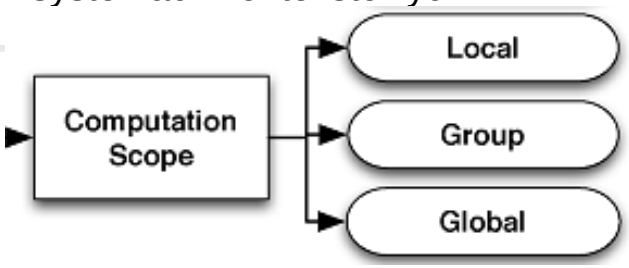
L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*

- Explicite: programista „koduje” operację komunikacji między węzłami
  - np. przygotowanie, wysłanie, odebranie, analiza wiadomości
- Implicite: programista „nie widzi” komunikacji między węzłami
  - komunikacja „ukryta” w wysokopoziomowej usłudze warstwy pośredniej
  - np. abstrakcja współdzielonej zmiennej, „widzianej” przez wiele węzłów
  - np. RPC
- Przykład: POBICOS – komunikacja wyrażona explicite





POBICOS dostarcza prymitywów do przesyłania komend i raportów kanałami logicznymi.



L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*

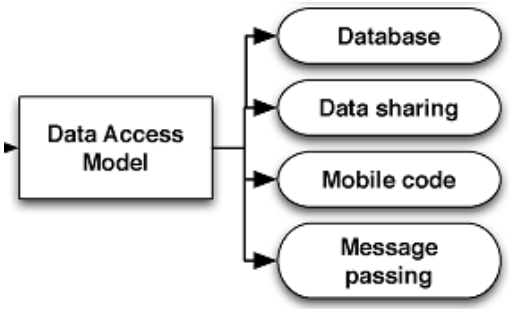
Zakres „obliczeń”

- Na jaki zbiór węzłów może mieć wpływ jedna instrukcja (operacja) programu
- Lokalny
  - jedna operacja dotyczy jednego węzła
- Grupowy
  - jedna operacja dotyczy całej grupy, np.
    - (a) zdefiniuj grupę wg odczytu sensora temperatury
    - (b) jedną operacją uśrednij odczyty temperatury w całej grupie
- Globalny
  - jedna operacja dotyczy wszystkich węzłów
  - np. jedno zapytanie zbiera odczyty sensorów temperatury ze wszystkich węzłów

- Przykład: POBICOS – zakres obliczeń lokalny

Operacje (zawarte w kodzie agenta) wykonuje się lokalnie na węźle

Model dostępu do danych



L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art*

- Abstrakcja bazy danych
  - cała sieć widziana jako baza danych
- Dane współdzielone
  - np. abstrakcja przestrzeni krotek (Linda)
- Kod mobilny
  - np. wysłanie agenta do źródła danych
- Przesyłanie wiadomości
- Przykład: POBICOS – kod mobilny i przesyłanie wiadomości

CreateGenericAgent
CreateNonGenericAgents

prymitywy dla kodu mobilnego (tworzenie instancji agenta)

SendCommand
GetCommand
SendReport
GetReport

prymitywy dla przesyłania wiadomości

PLATFORMY DLA APLIKACJI KONTEKSTOWYCH

Platforma to coś więcej niż warstwa pośrednia

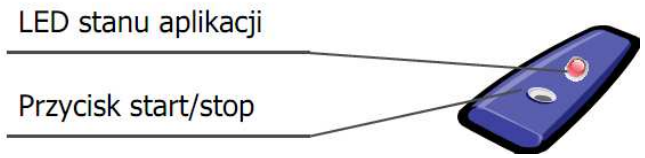
- Celem są aplikacje kontekstowe, ale samo API nie wystarczy.
- Elementy platformy:
  - model interesariuszy
  - model interakcji z użytkownikiem
  - meta-model dla modelu dziedziny (kontekstu)
  - modele dla wybranych dziedzin
  - specyfikacja API
  - warstwa pośrednia (implementacja API)
  - narzędzia
    - „kompilatory” modelu dziedziny, narzędzia do generacji kodu
    - symulatory, narzędzia diagnostyczne, itp.
- Omówimy powyższe elementy na przykładzie platformy POBICOS.

Przykład: POBICOS

- Na początku jest idea stworzenia sieci współpracujących, inteligentnych obiektów.

- Sposób realizacji: wyposażenie każdego obiektu w warstwę pośrednią.

- Dodatkowy pomysł: dystrybucja aplikacji za pomocą tzw. pigułek aplikacyjnych.

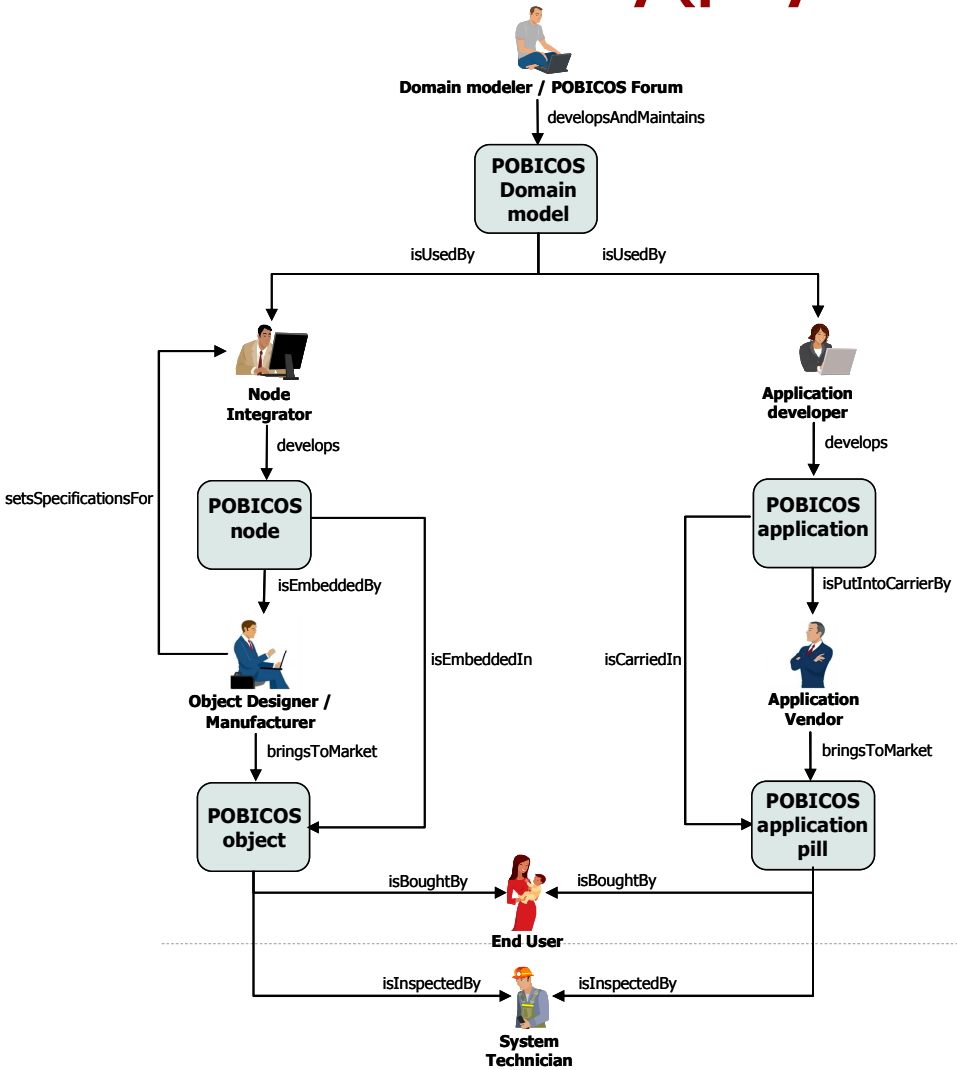


Model interesariuszy

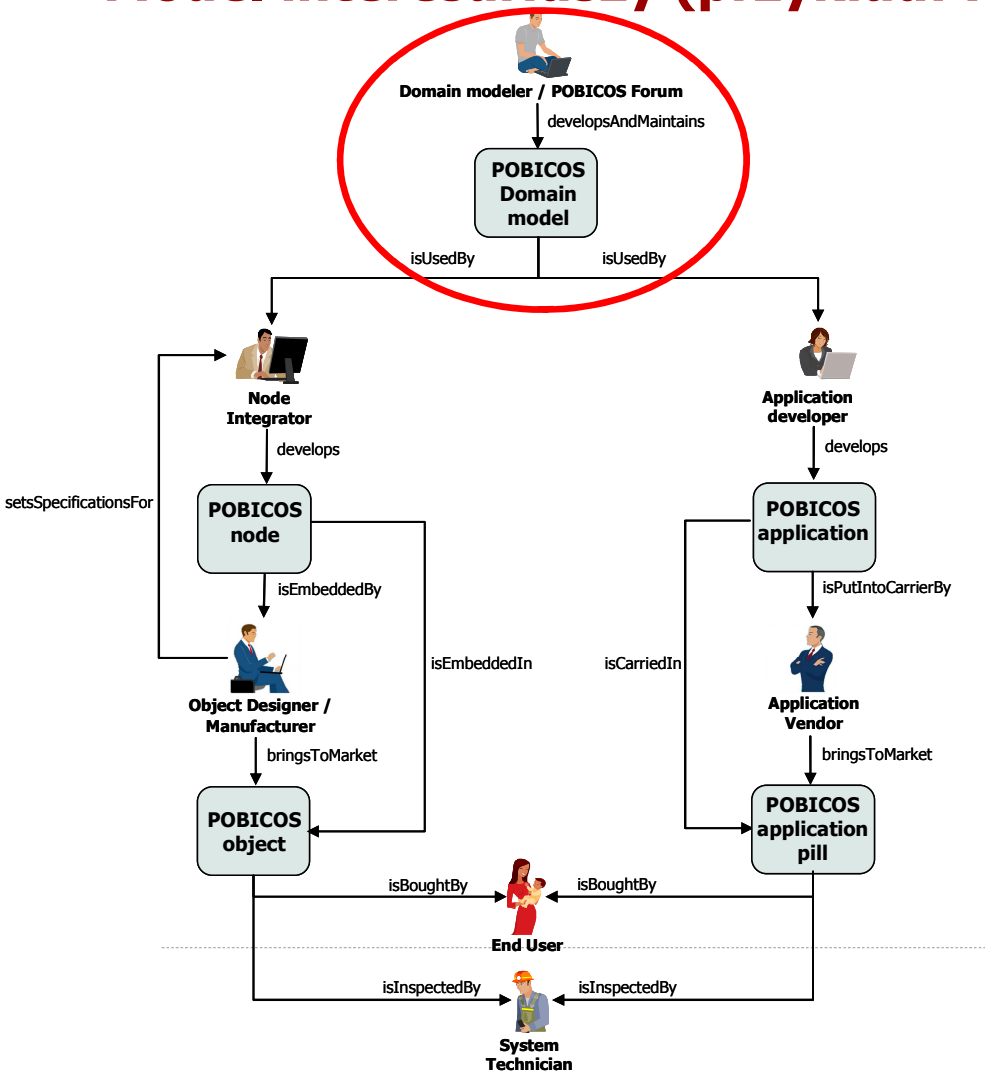
- Interesariusz organizacji (projektu, …) to osoba lub organizacja, która
  - wpływa na daną organizację (projekt, …) lub …
  - pozostaje pod jej wpływem.
- **Interesariusz platformy** – osoba lub organizacja mająca styczność (bezpośrednią lub pośrednią) z daną platformą
- Platforma aplikacji kontekstowych powinna określać pewien „ekosystem” interesariuszy (**model interesariuszy**)

- Co daje nam model interesariuszy platformy aplikacji ?
  - precyzyjne określenie ról, kompetencji, punktów widzenia
  - określenie możliwości dotarcia do użytkownika
  - ujawnienie możliwych modeli biznesowych

Model interesariuszy (przykład: POBICOS)



Model interesariuszy (przykład: POBICOS)



Domain modeler / POBICOS Forum

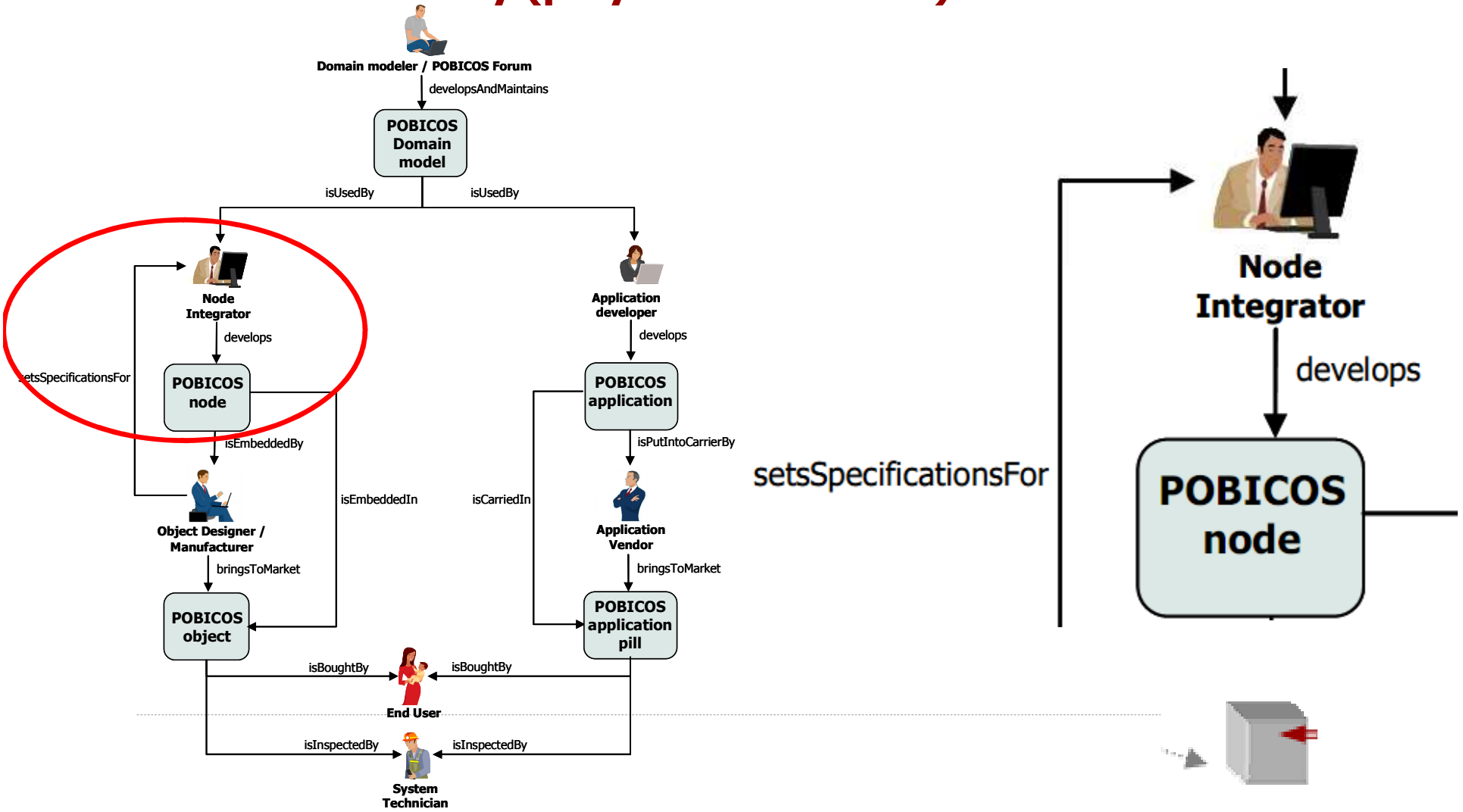
developsAndMaintains

POBICOS Domain model

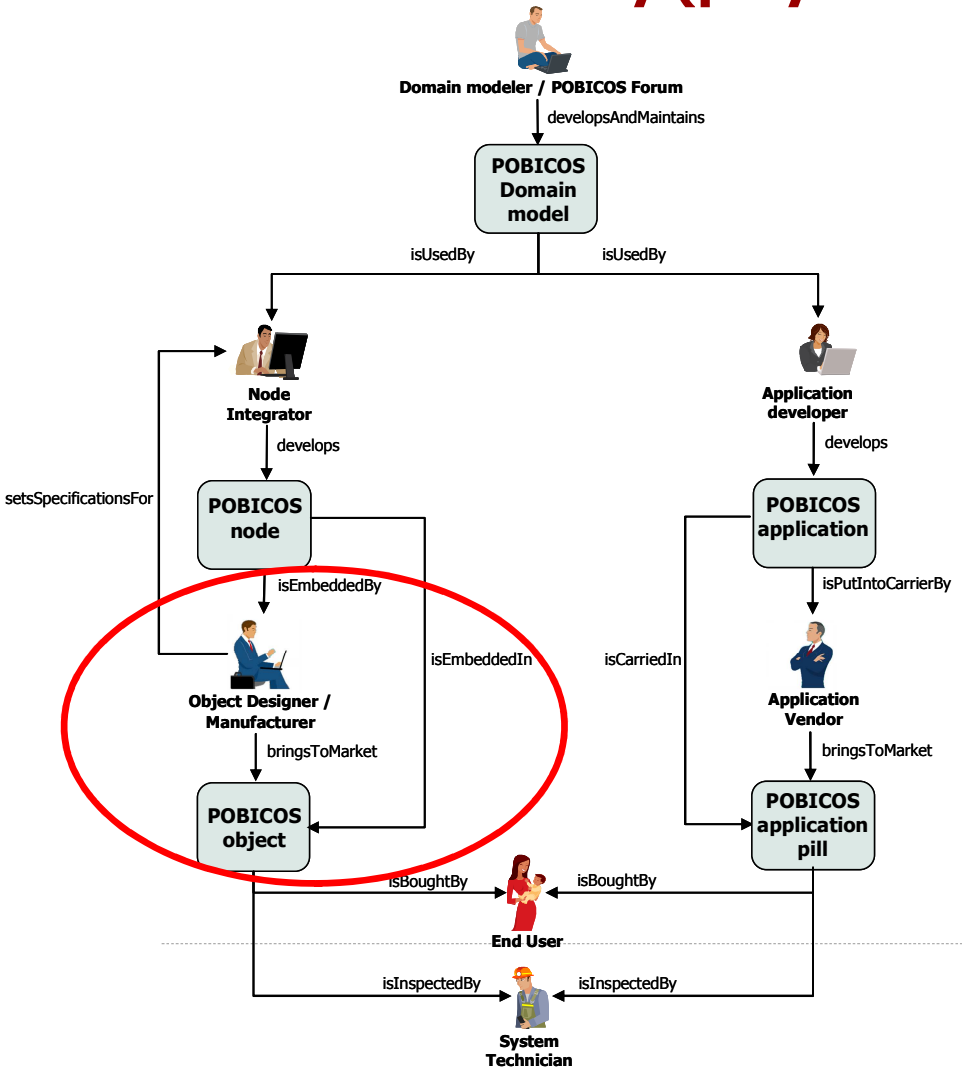
isUsedBy

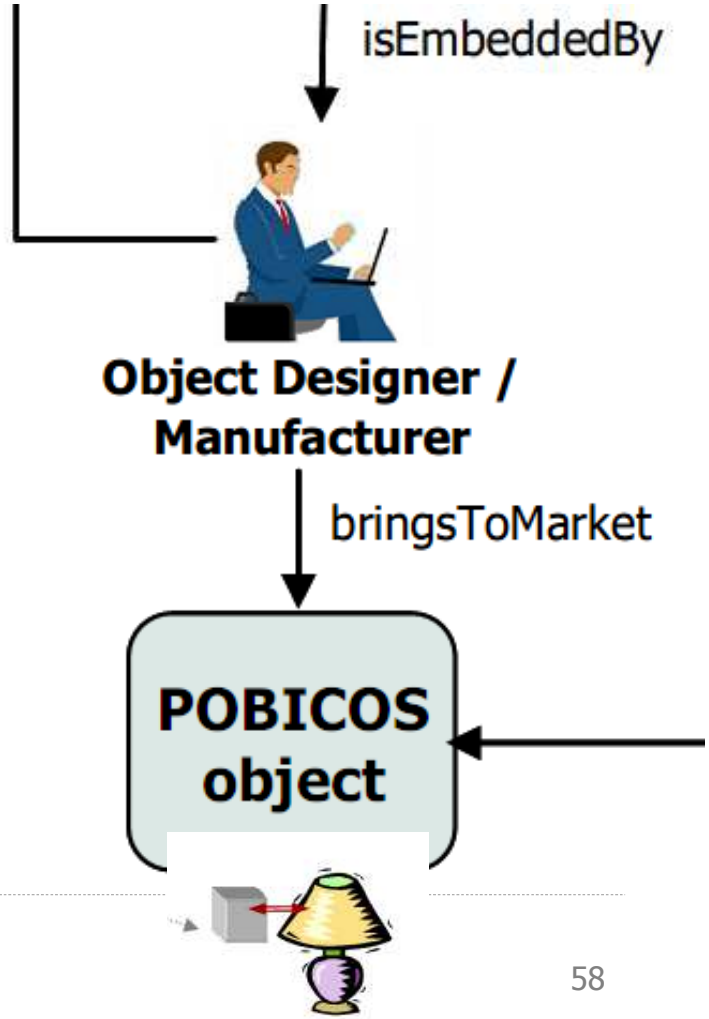
```
<!-- http://www.ict-pobicos.eu/onto/pobicos#
<owl:Class rdf:about="#alert">
    <rdfs:subClassOf rdf:resource="#Instruct
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#h
            <owl:someValuesFrom rdf:resource
        </owl:Restriction>
    </rdfs:subClassOf>
```

Model interesariuszy (przykład: POBICOS)

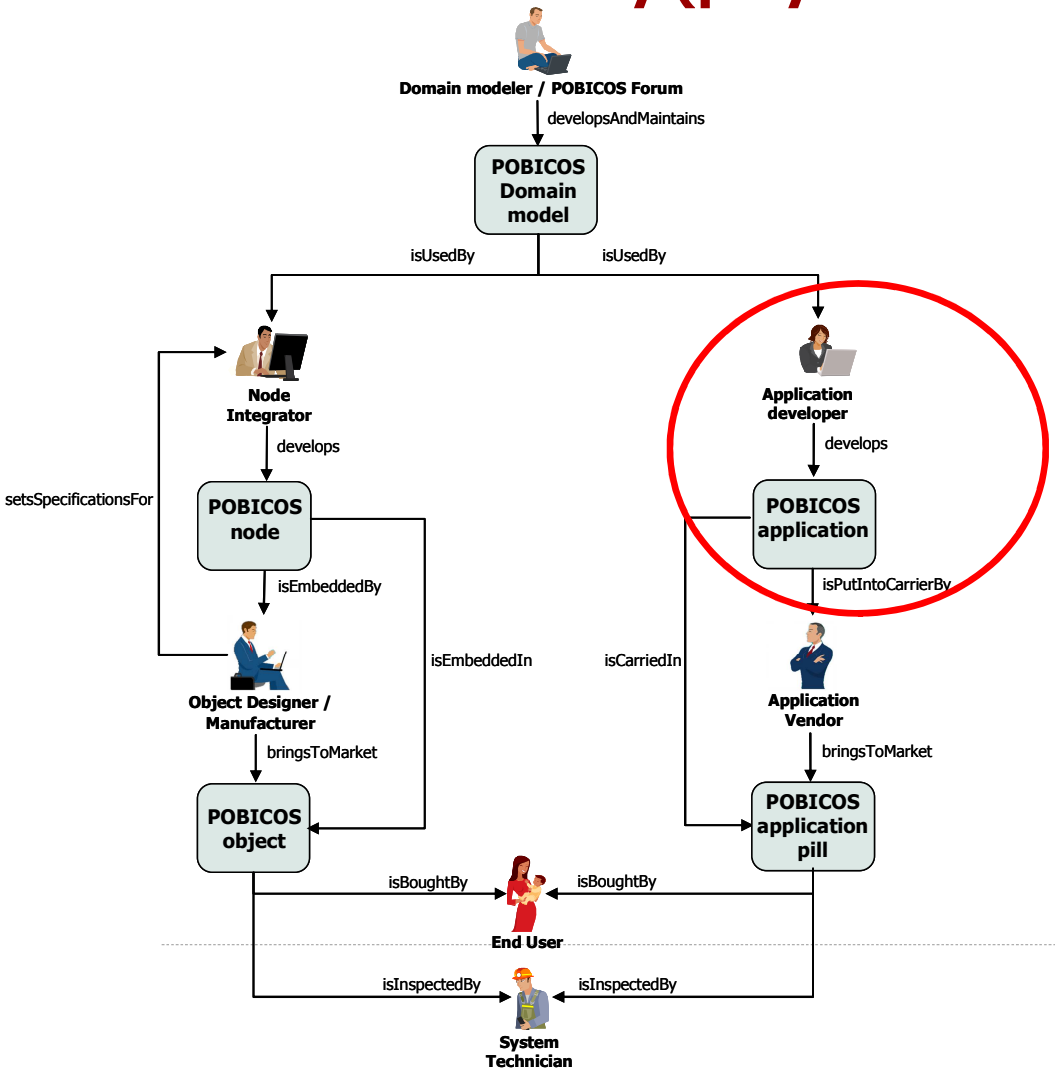


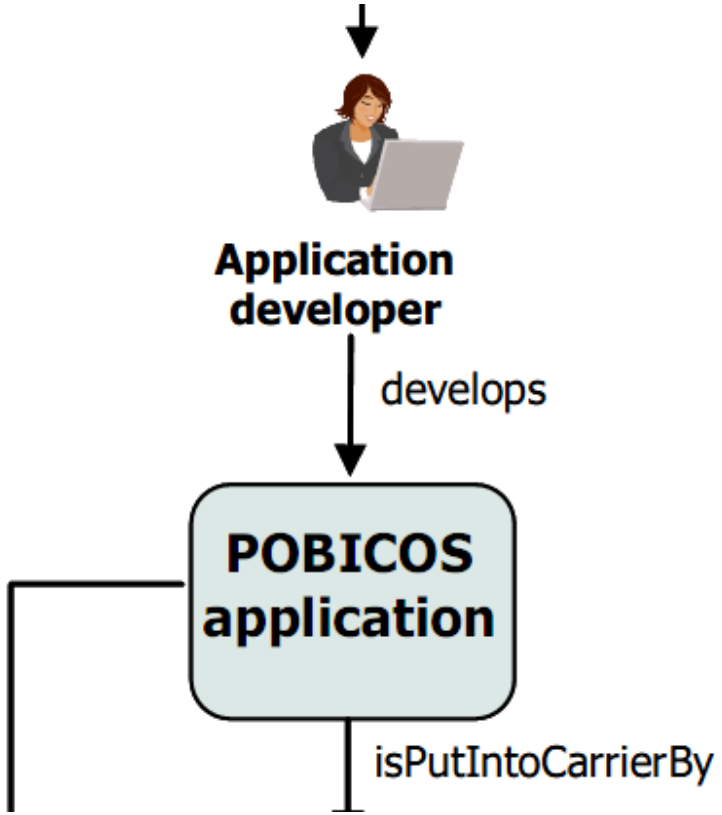
Model interesariuszy (przykład: POBICOS)



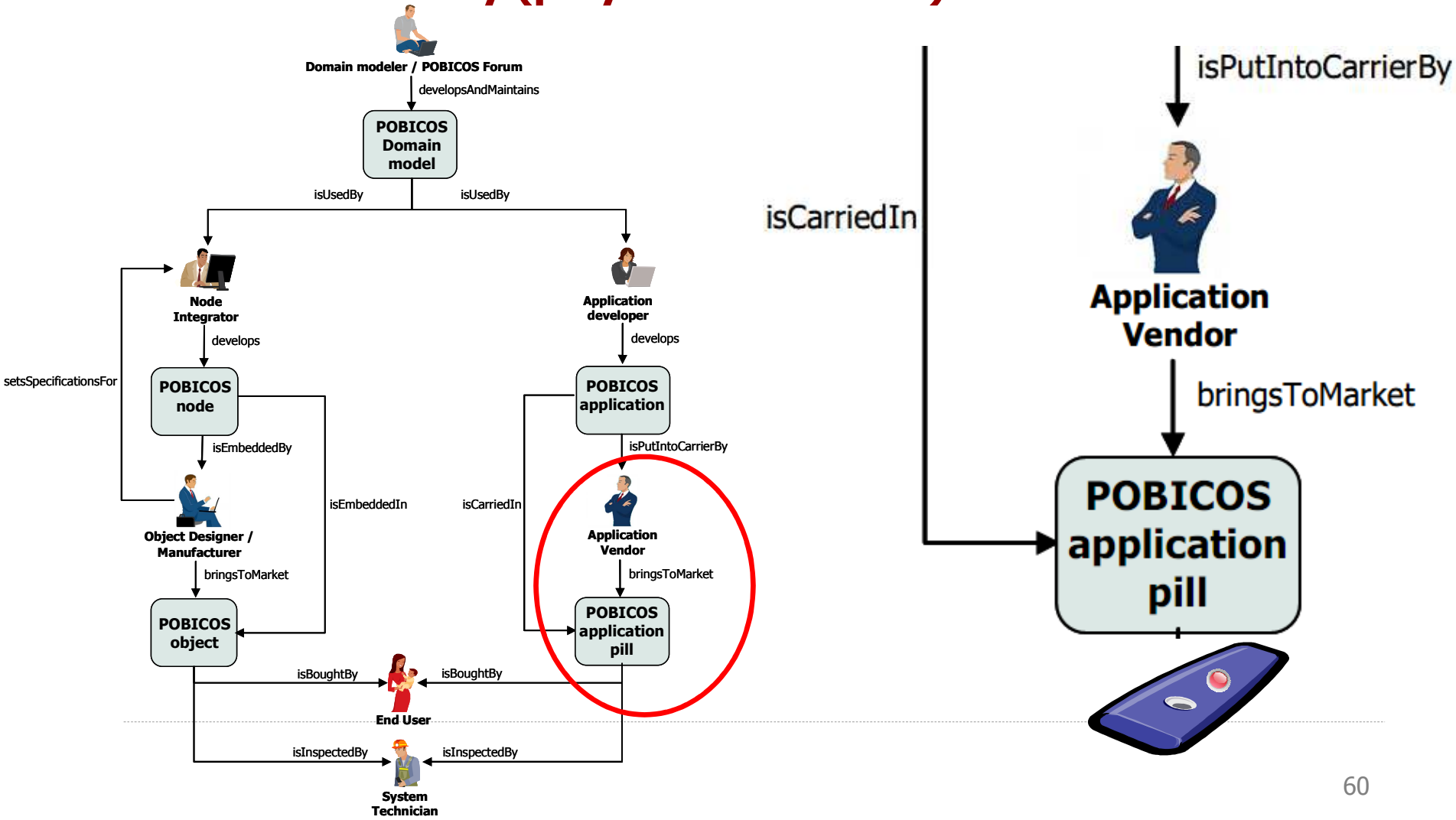


Model interesariuszy (przykład: POBICOS)

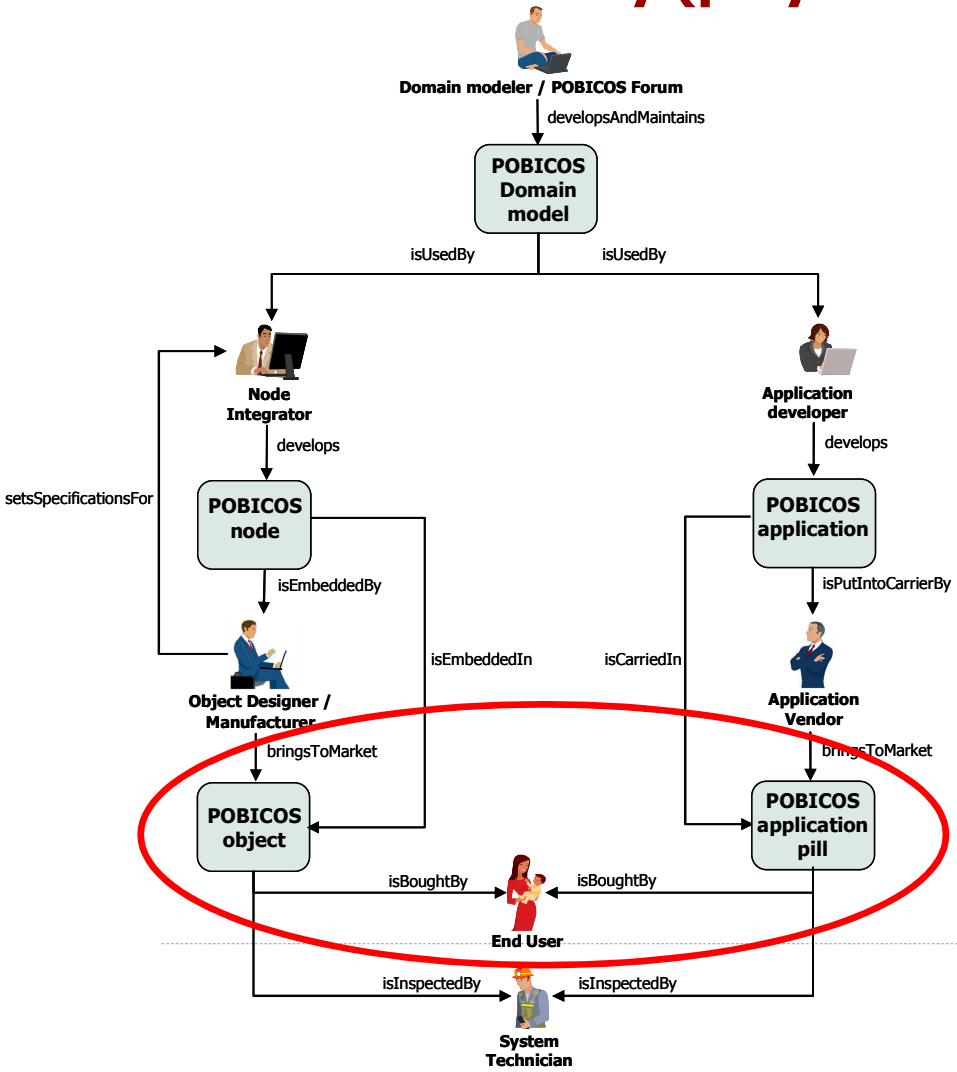


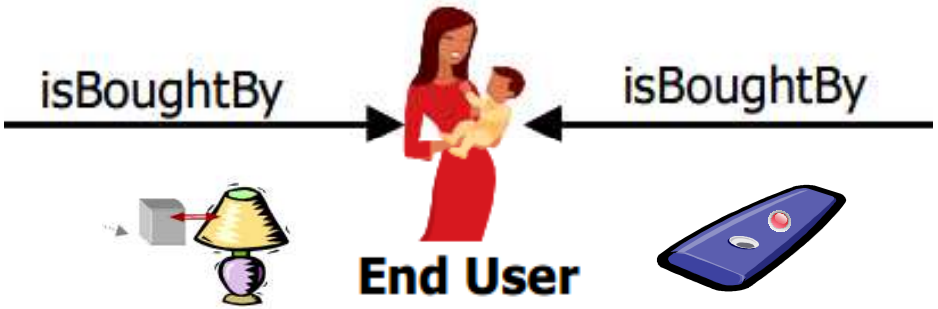


Model interesariuszy (przykład: POBICOS)

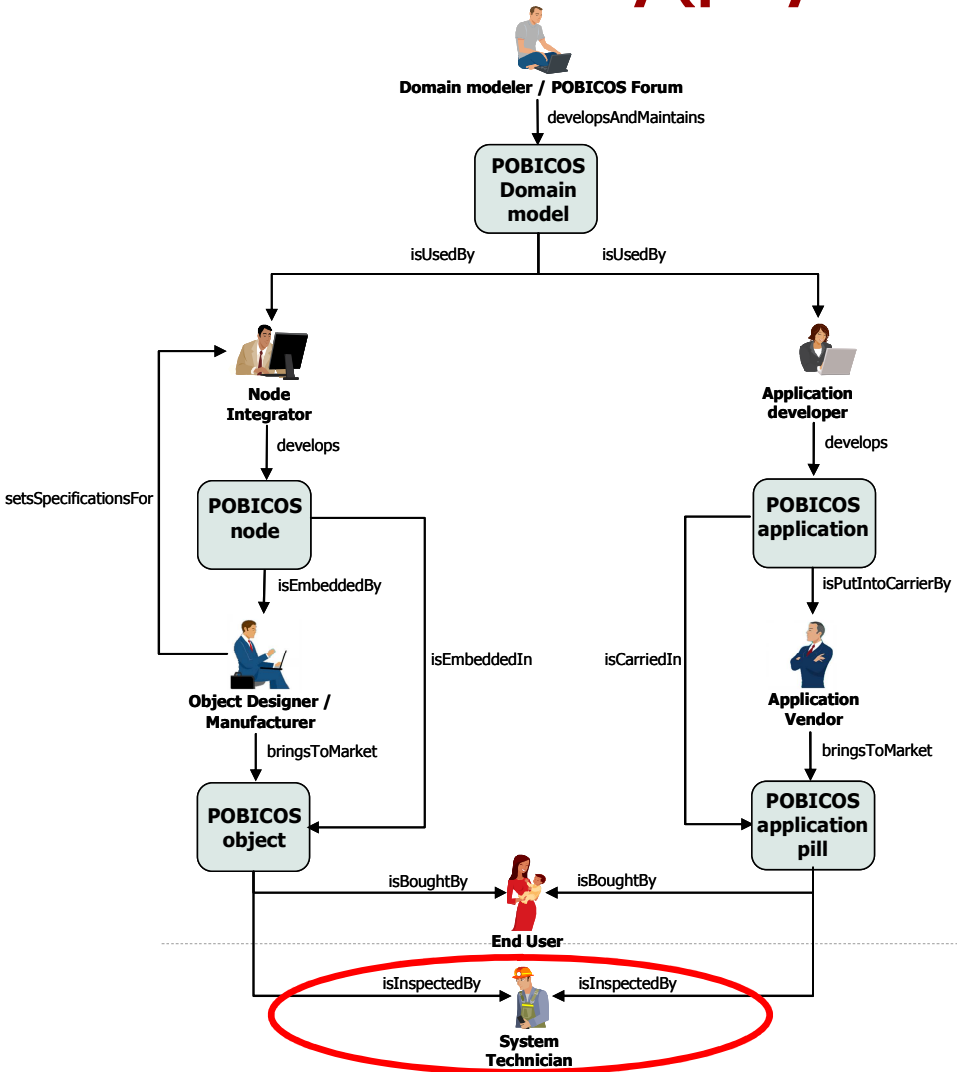


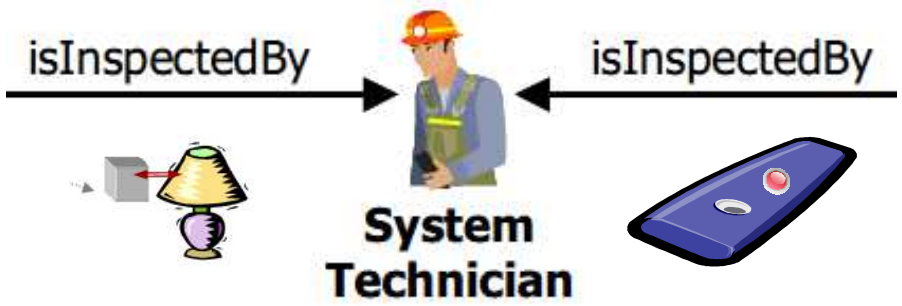
Model interesariuszy (przykład: POBICOS)





Model interesariuszy (przykład: POBICOS)

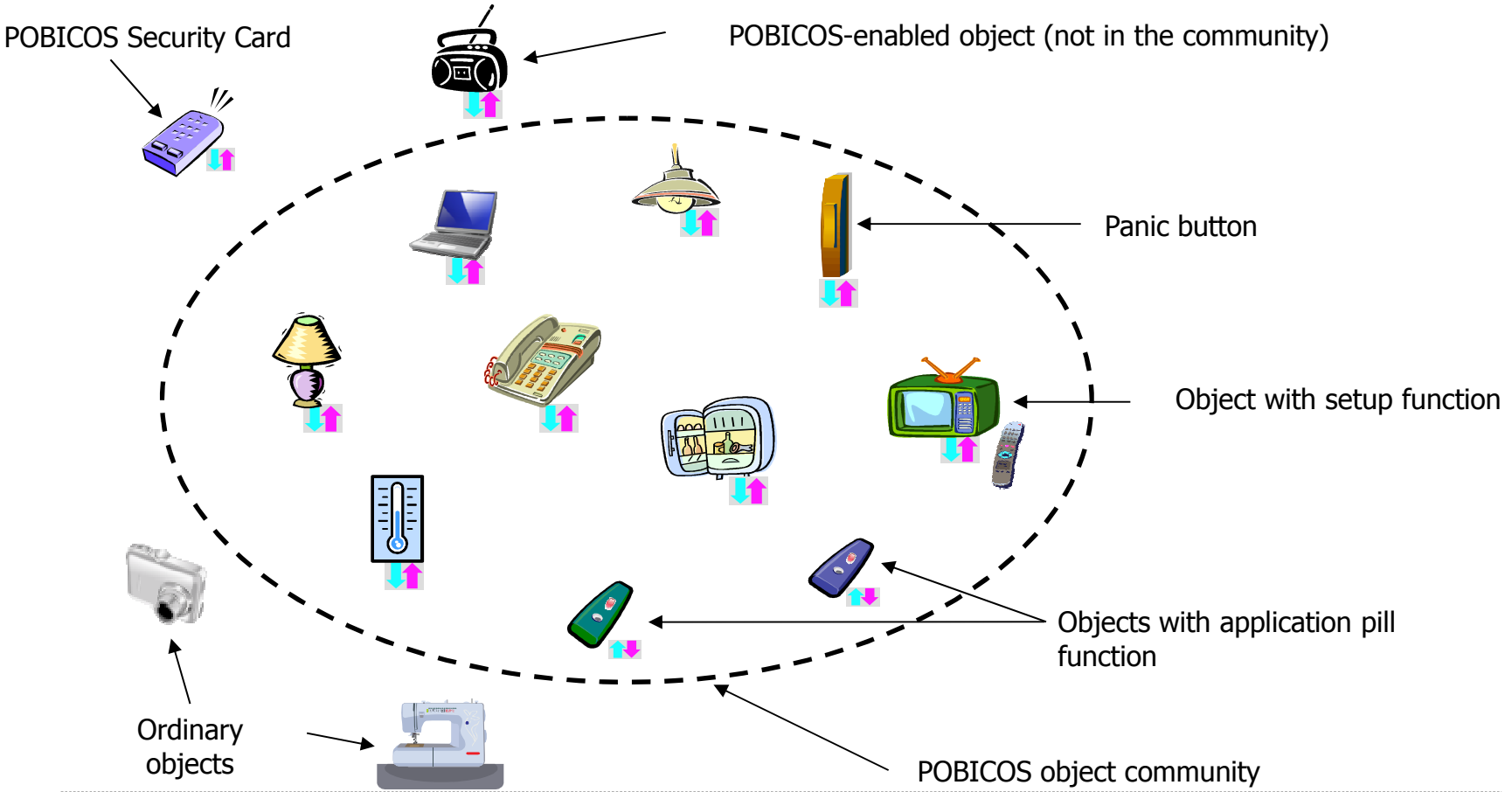




Model interakcji z użytkownikiem

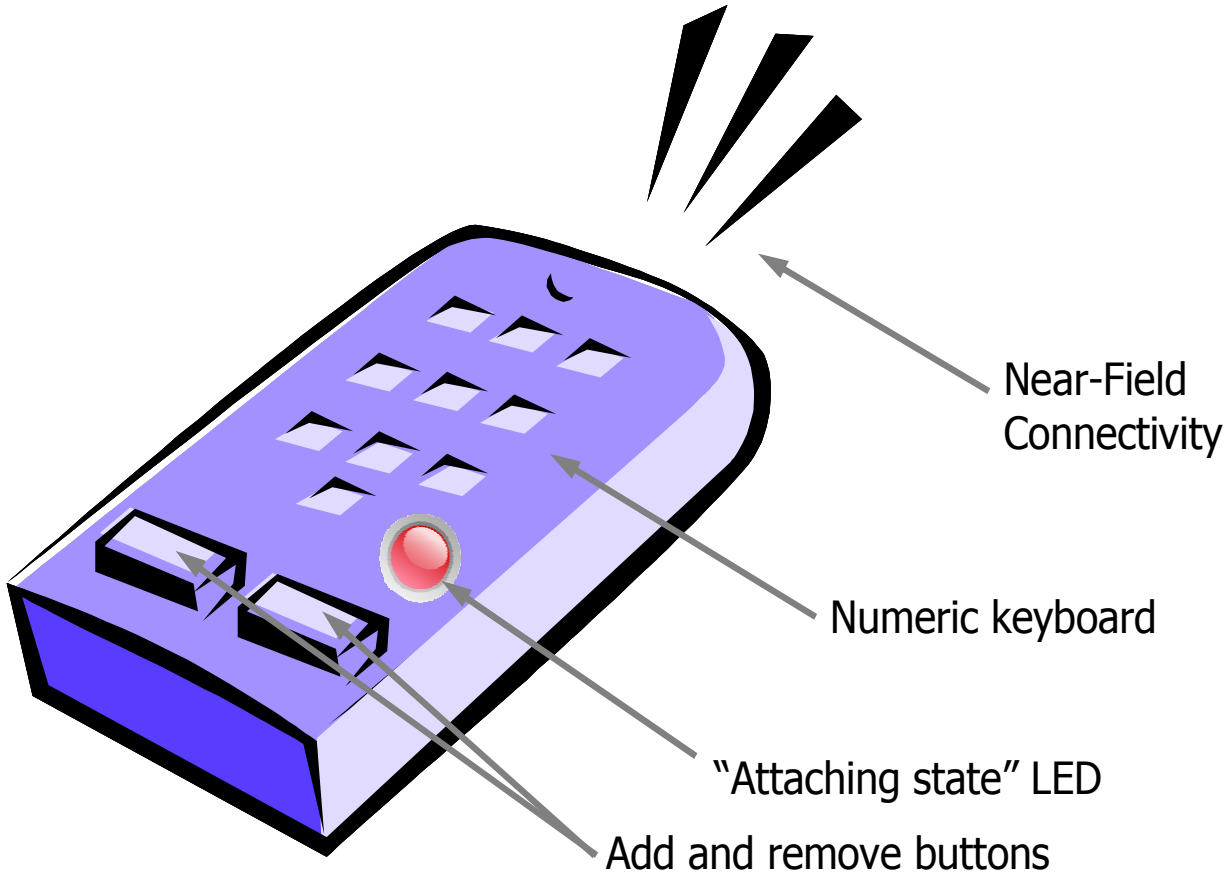
- Wspólne elementy interakcji z wieloma aplikacjami, obiektami, itp.
  - **jednolitość** (!)
- Przykład: POBICOS

End-user's view of POBICOS



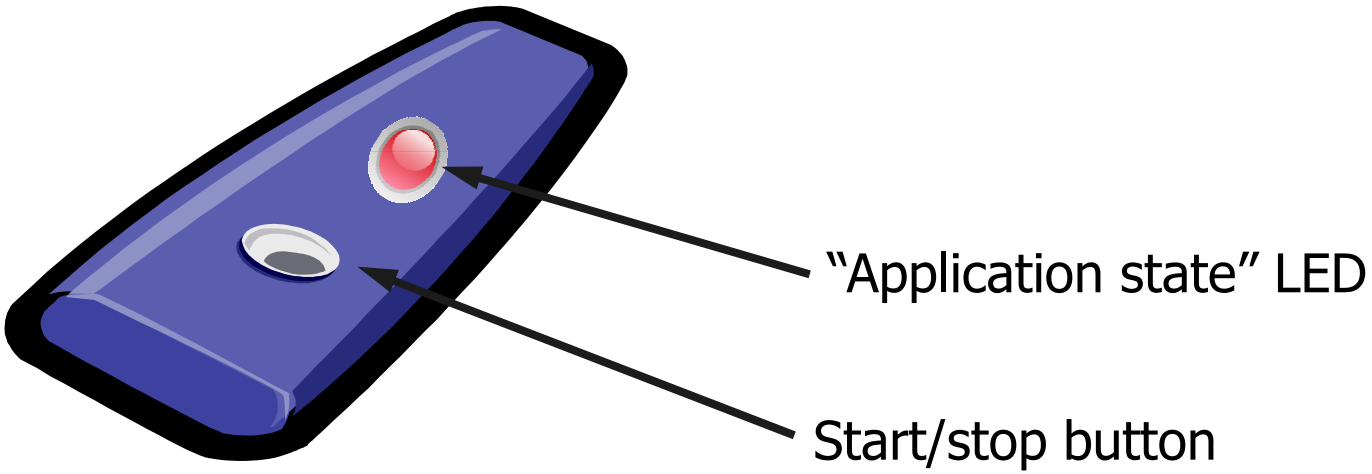
Lalis, S.; Domaszewicz, J.; Pruszkowski, A.; Paczesny, T.; Ala-Louko, M.; Taumberger, M.; Georgakoudis, G., Lekkas, K., *Tangible Applications for Regular Objects: An End-User Model for Pervasive Computing at Home*

POBICOS Security Card



Służy do zarządzania społecznością obiektów (dodawanie i usuwanie obiektów).

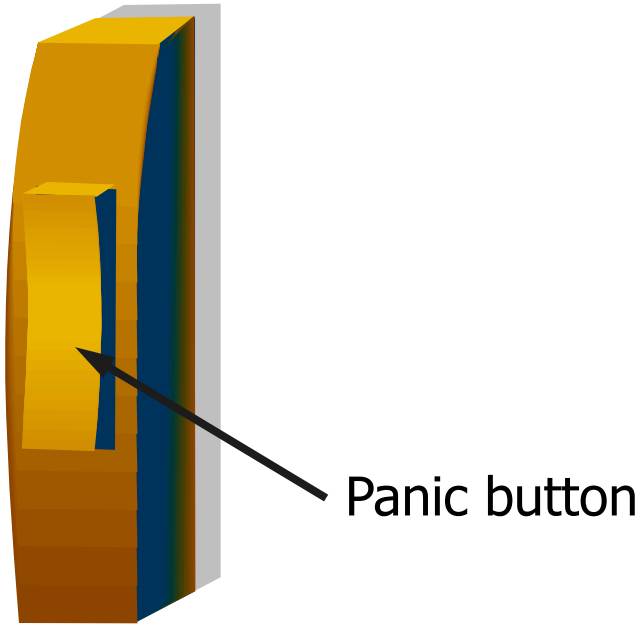
Application pill object



Domaszewicz, J.; Lalis, S.; Paczesny, T.; Pruszkowski, A.; Ala-Louko, M., *Graspable and resource-flexible applications for pervasive computing at home,*

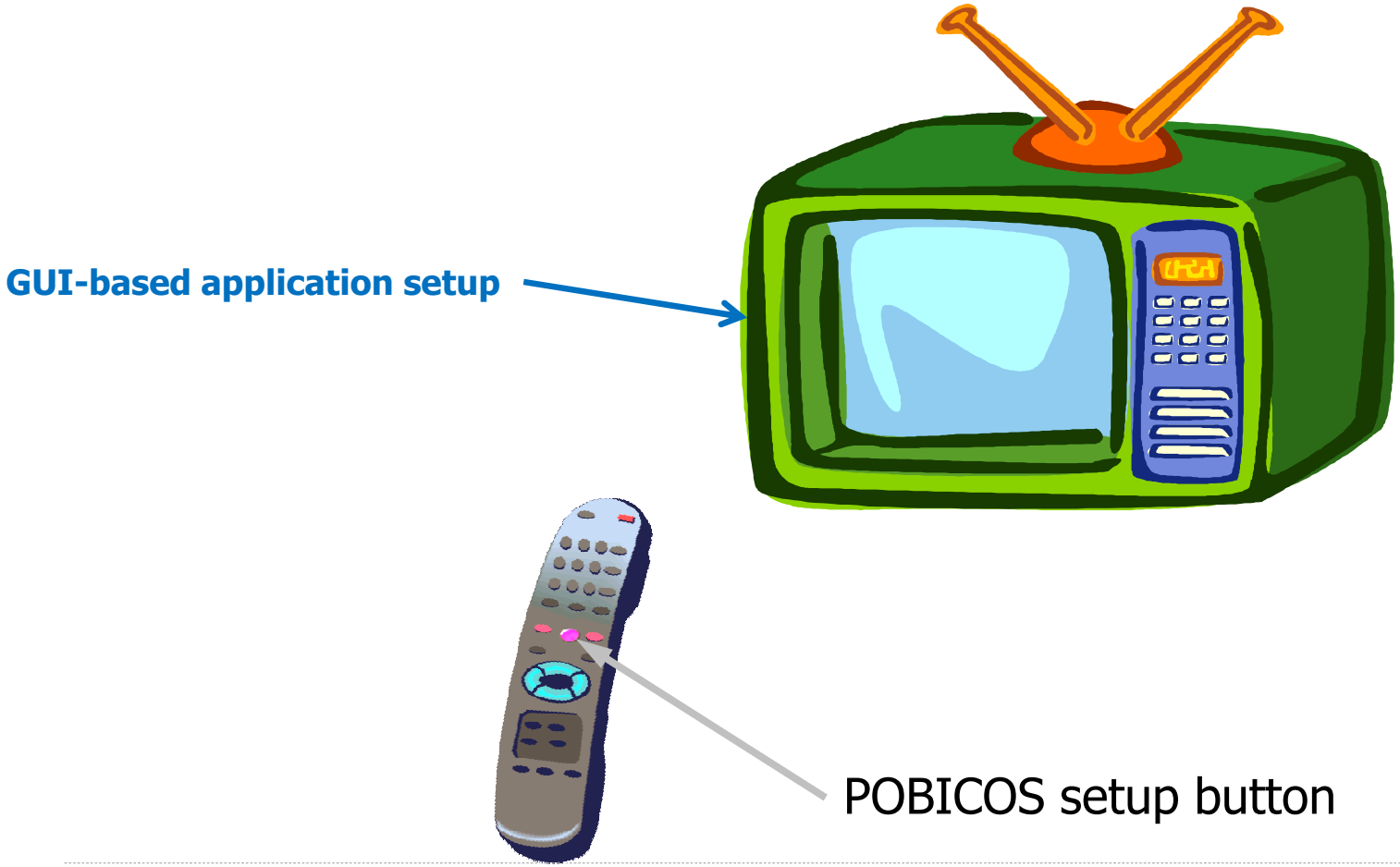
Nośnik kodu aplikacji i minimalny interfejs (aplikacja jak latarka).

Panic button object



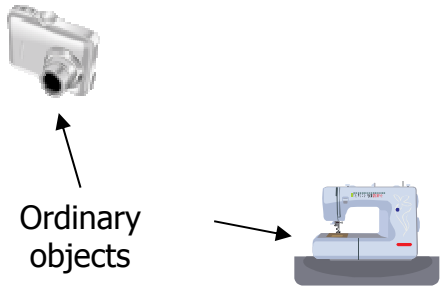
Natychmiast zatrzymaj wszystkie aplikacje.

Setup object



Obiekt z ekranem wykorzystywany do ustawienia parametrów wybranej aplikacji.

Use case: establishing an object community

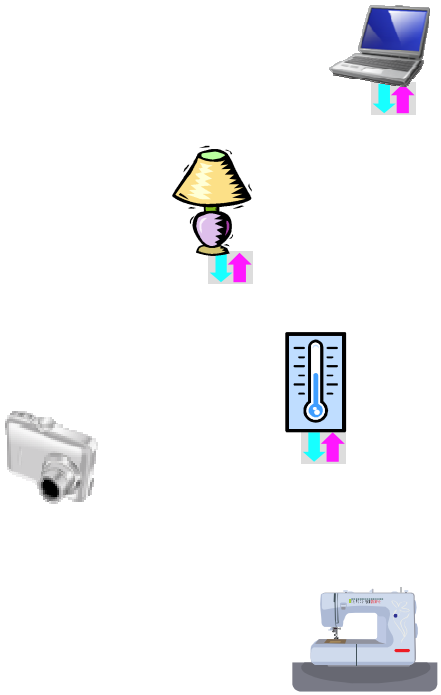


Na początku mamy tylko zwykłe obiekty.

Use case: establishing an object community

Kupiliśmy lampkę (pewnie nie wiedząc, że jest wyposażona w system POBICOS).

Use case: establishing an object community



Potrzebujemy kolejnych obiektów; kupujemy je. Znowu są z POBICOSem.

Use case: establishing an object community

I jeszcze kilka takich obiektów.

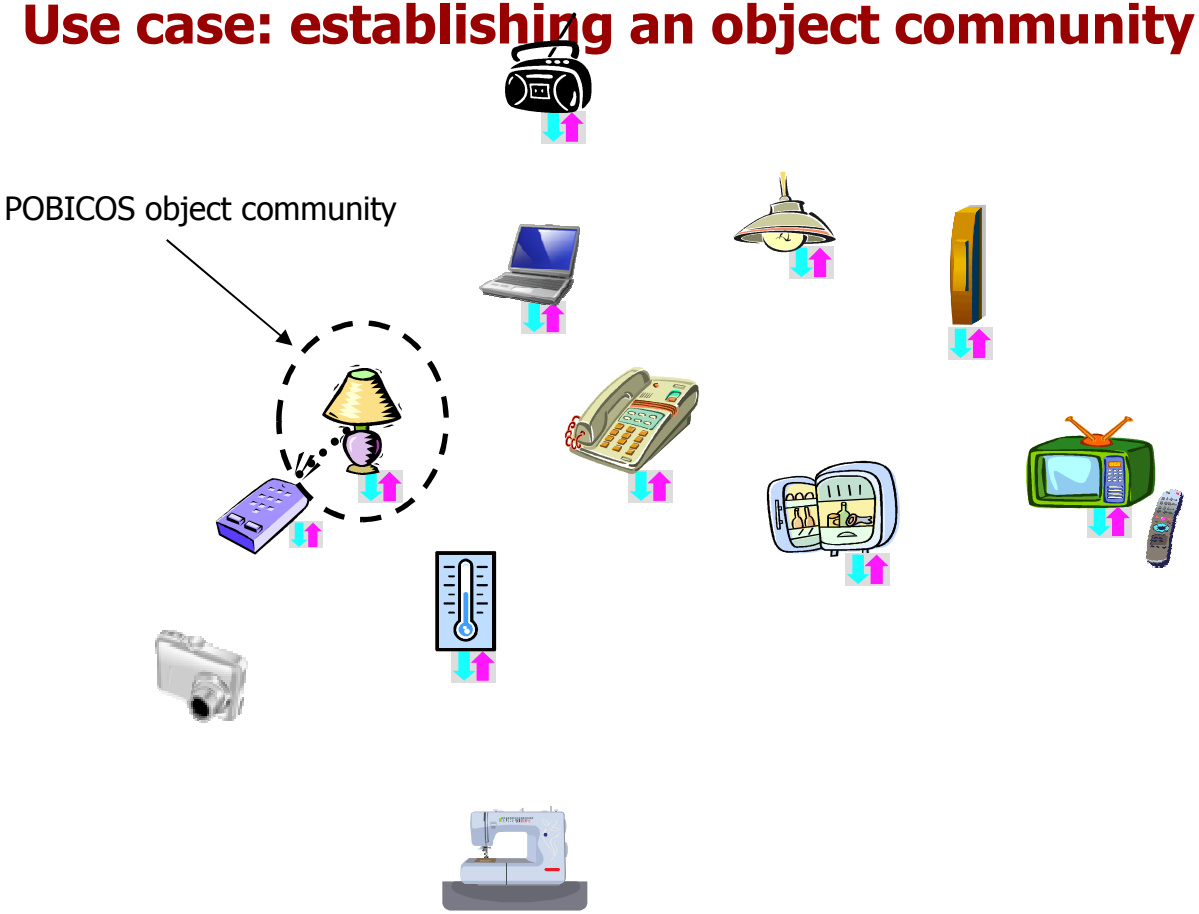
Use case: establishing an object community

Dowiadujemy się o POBICOSie i nabywamy dwa obiekty specjalne.

Use case: establishing an object community

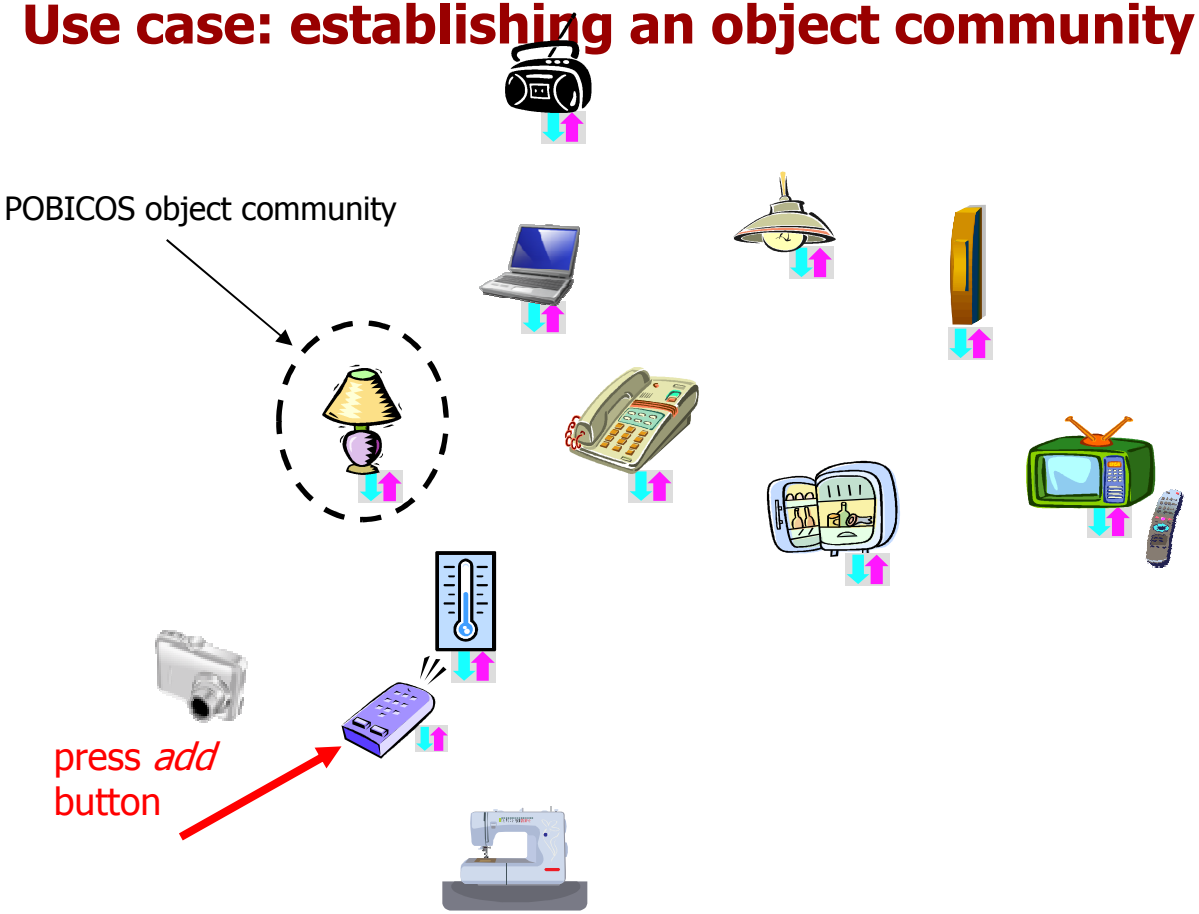
Rozpoczynamy tworzenie społeczności obiektów.

Use case: establishing an object community



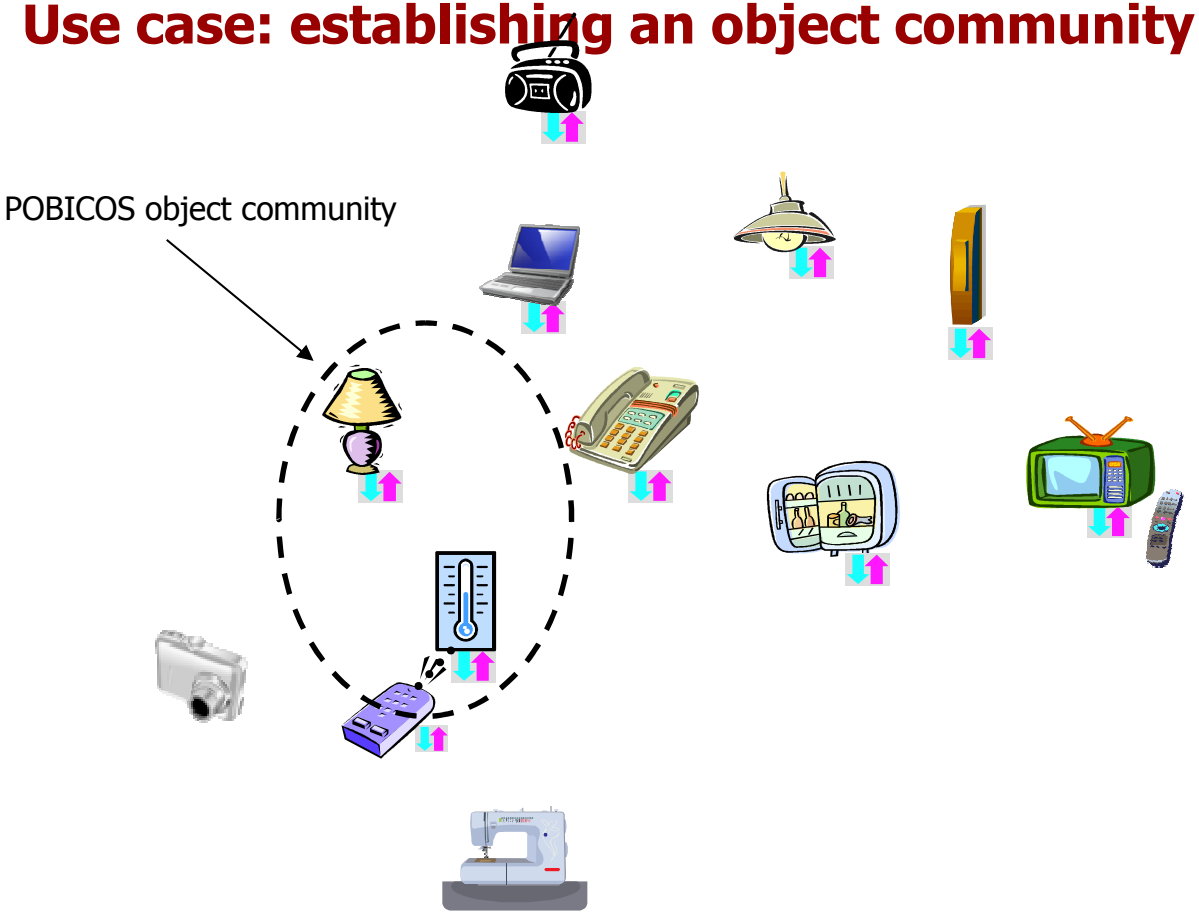
Lampka należy już do społeczności.

Use case: establishing an object community

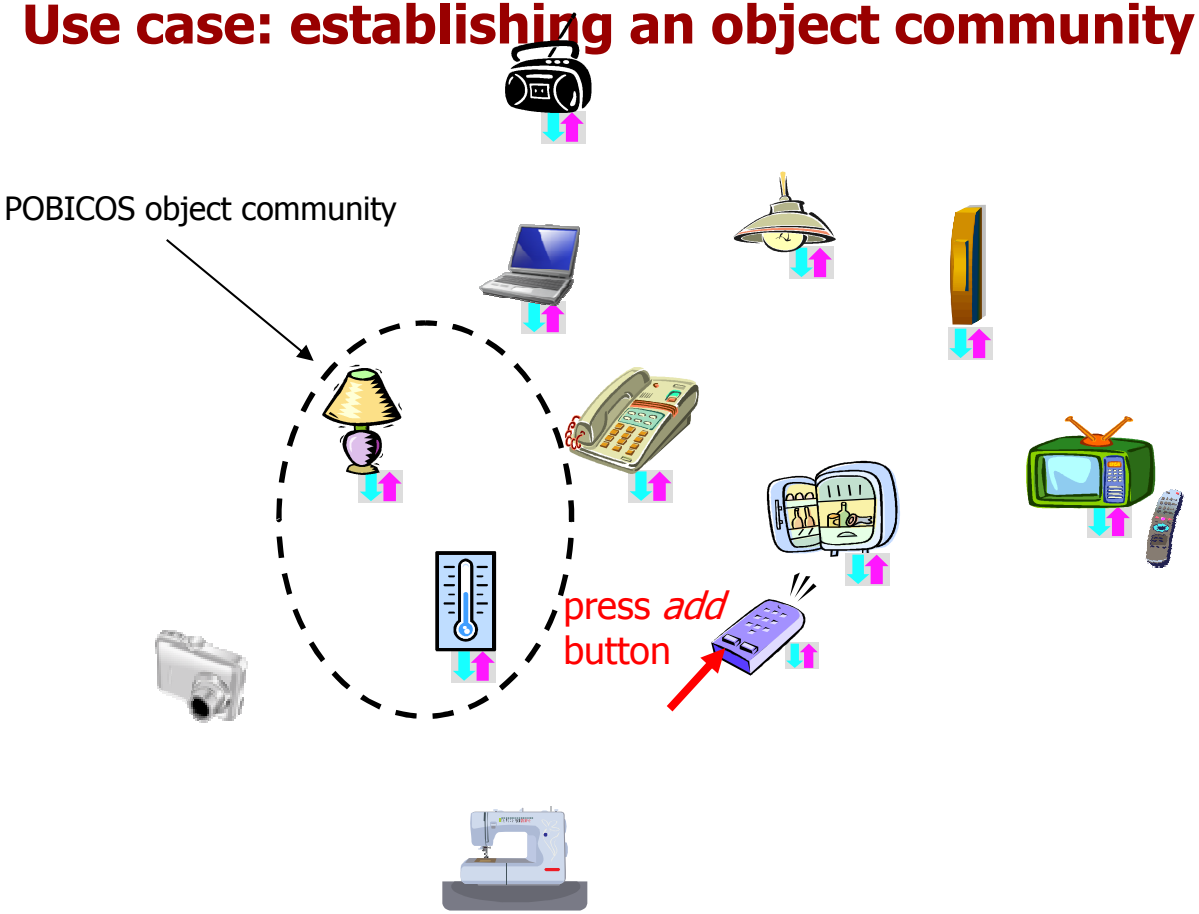


Teraz dodajemy termometr.

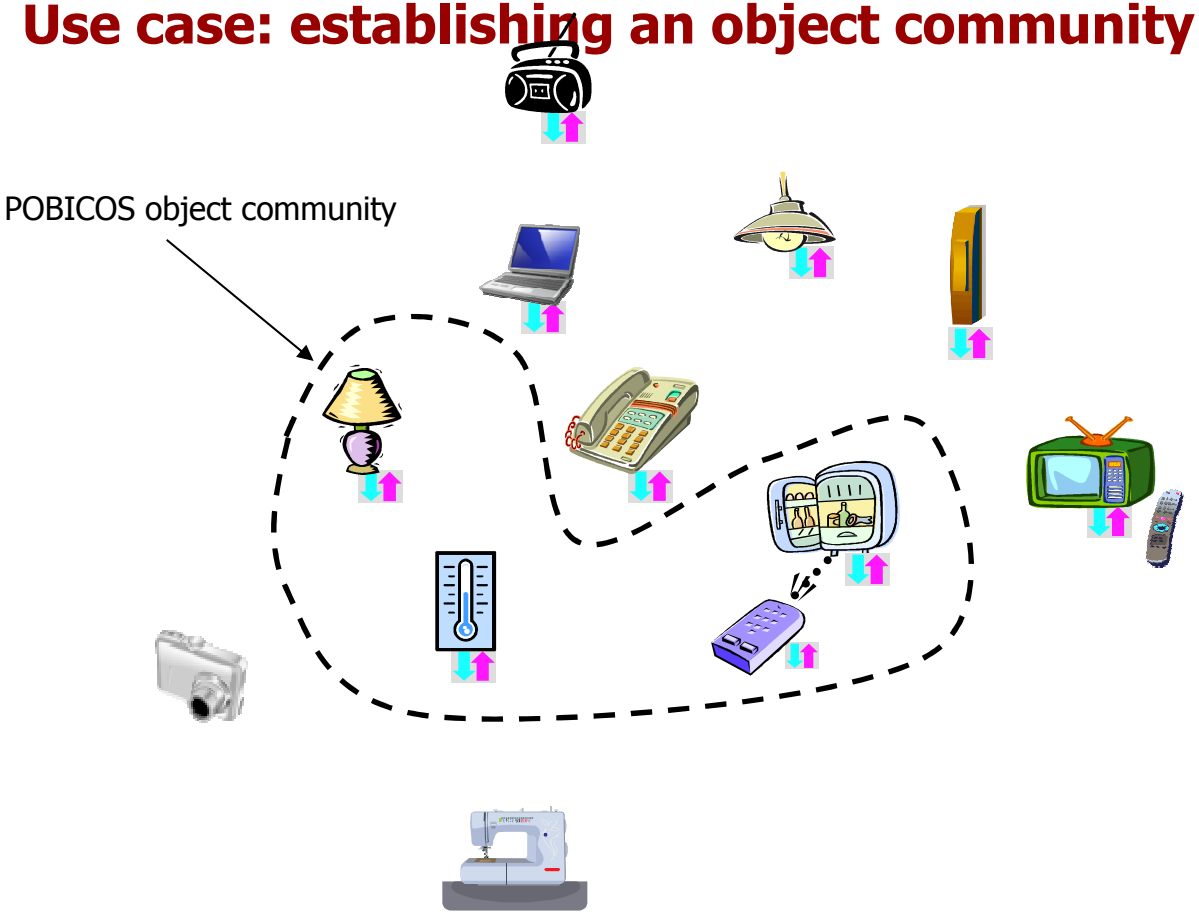
Use case: establishing an object community



Use case: establishing an object community



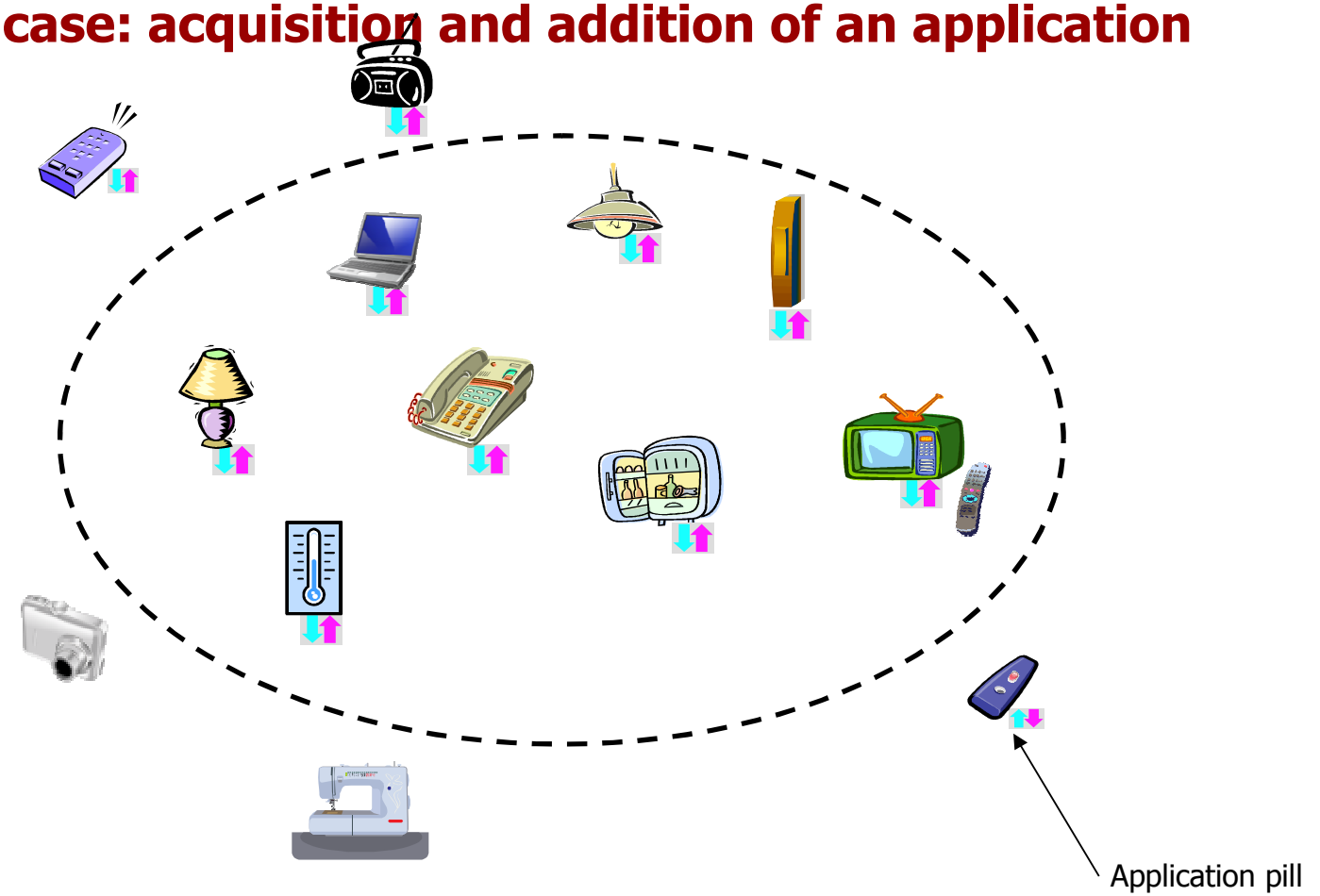
Use case: establishing an object community



Use case: establishing an object community

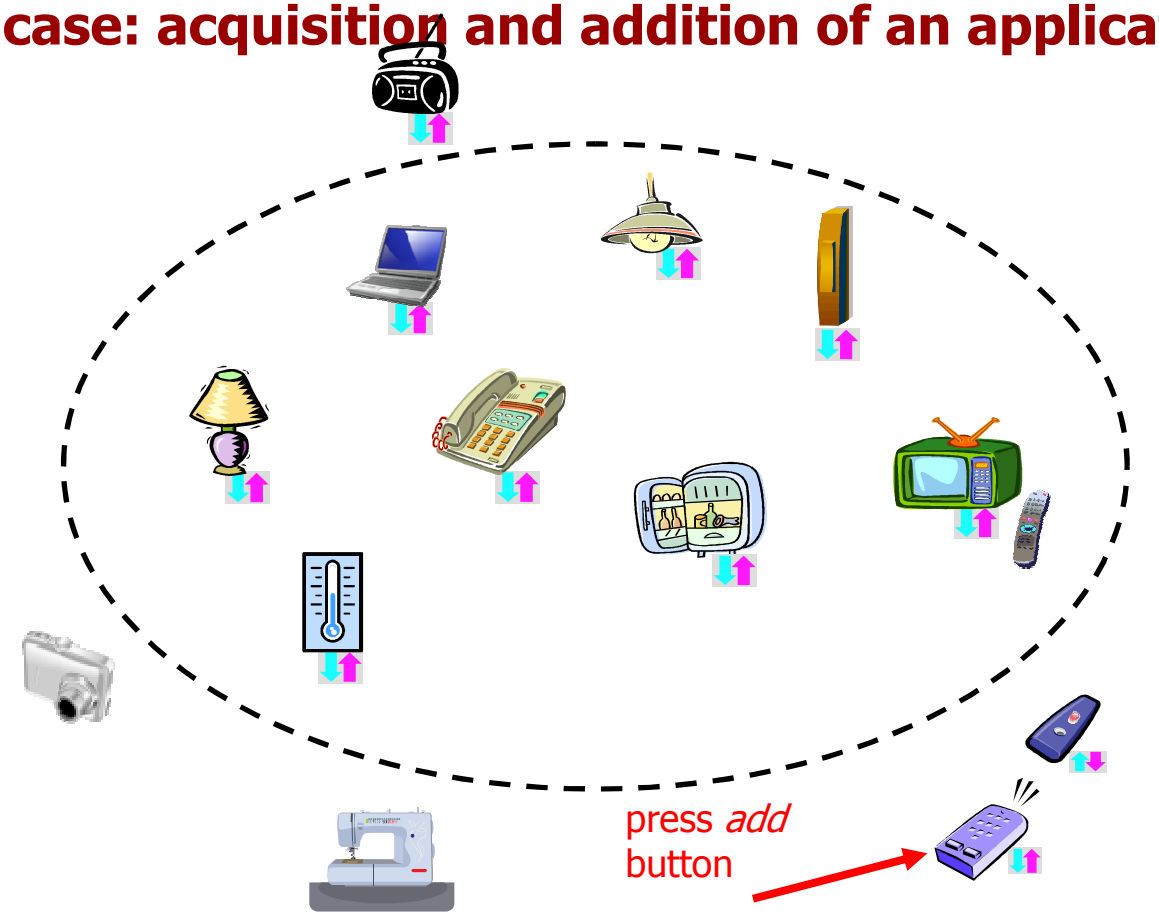
W społeczności obiektów jest już sporo zasobów sensorowych i wykonawczych.

Use case: acquisition and addition of an application



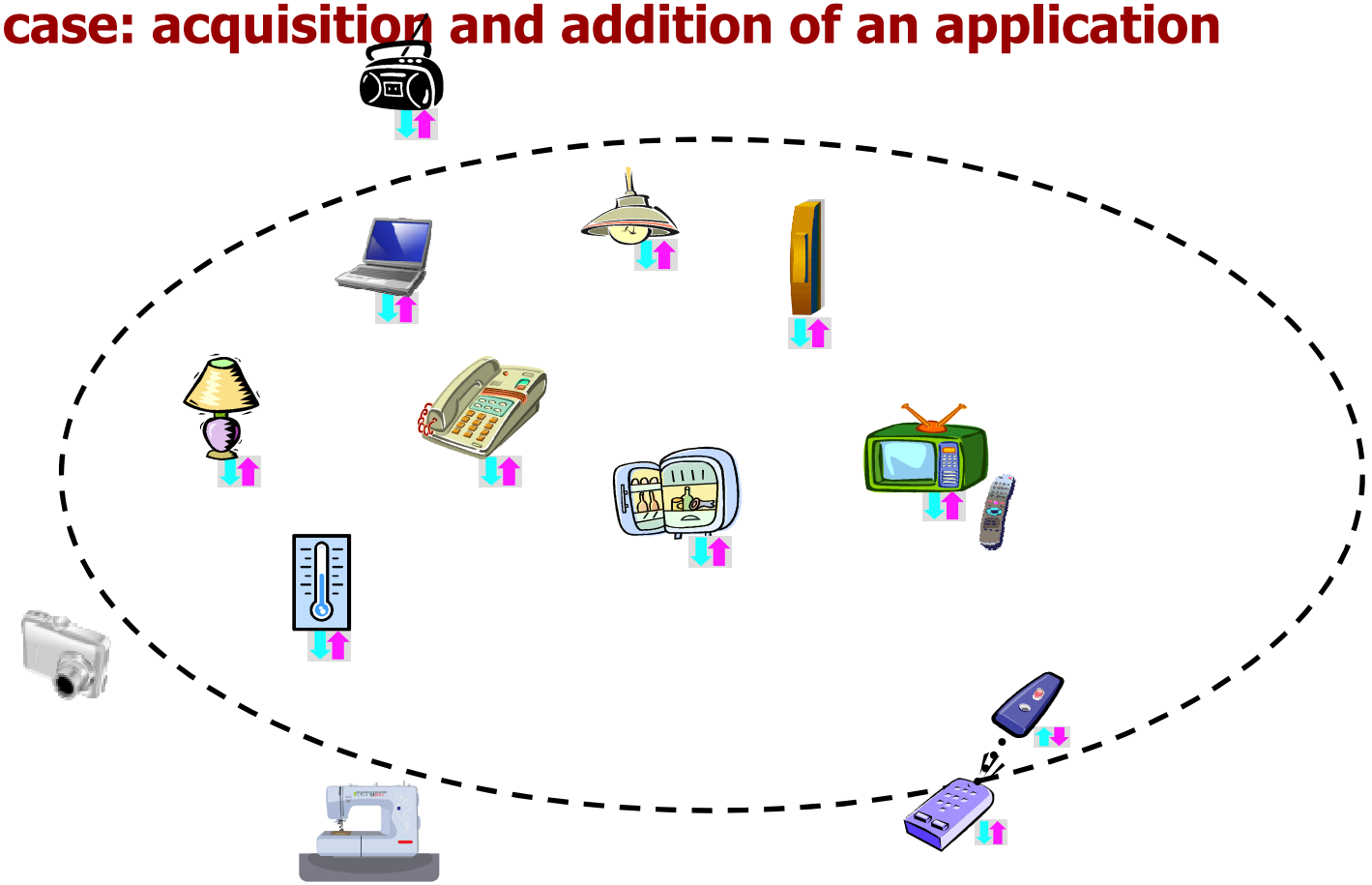
Nabywamy pierwszą aplikację (w postaci pigułki aplikacyjnej).

Use case: acquisition and addition of an application

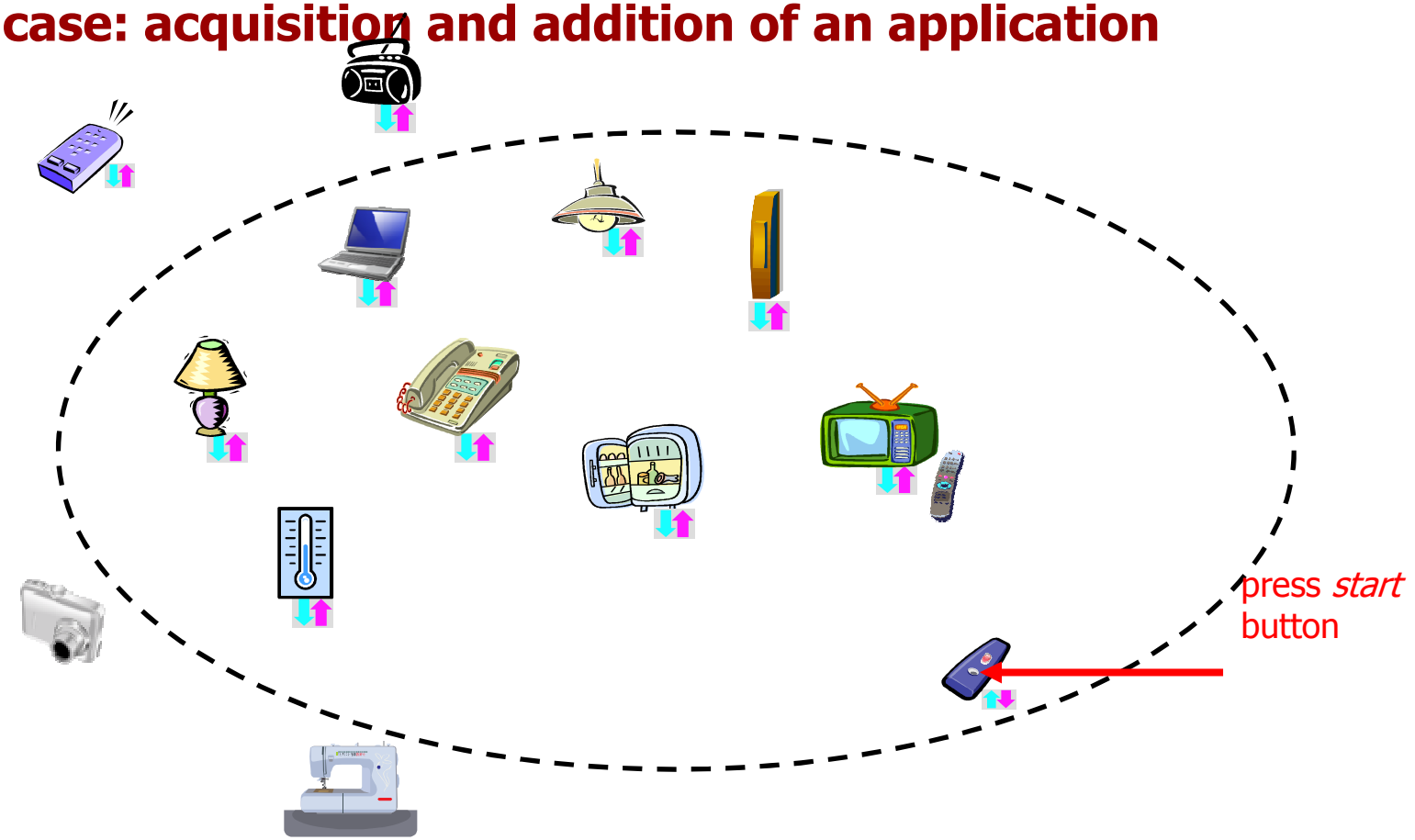


Dodajemy pigułkę aplikacyjną do społeczności obiektów.

Use case: acquisition and addition of an application

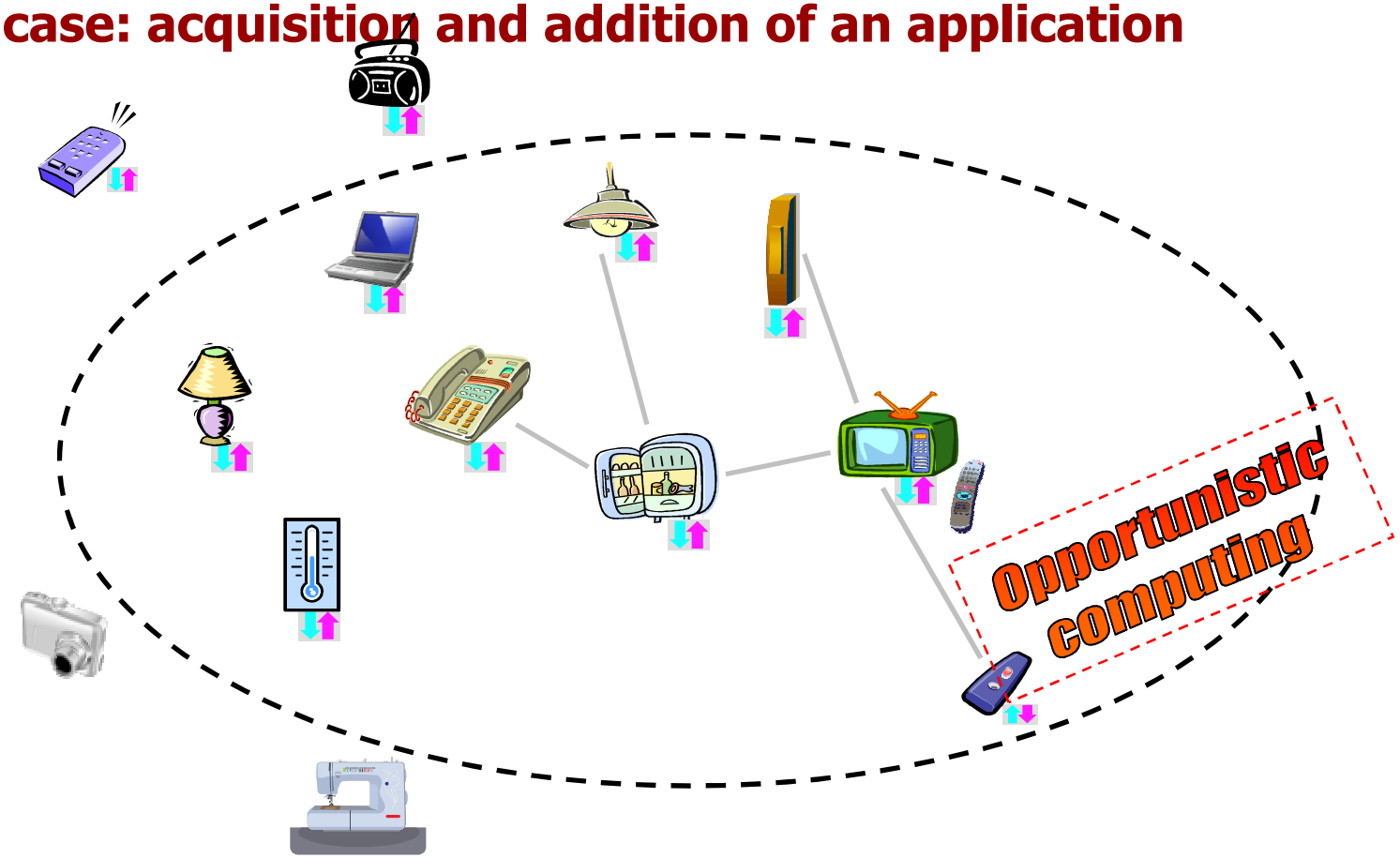


Use case: acquisition and addition of an application



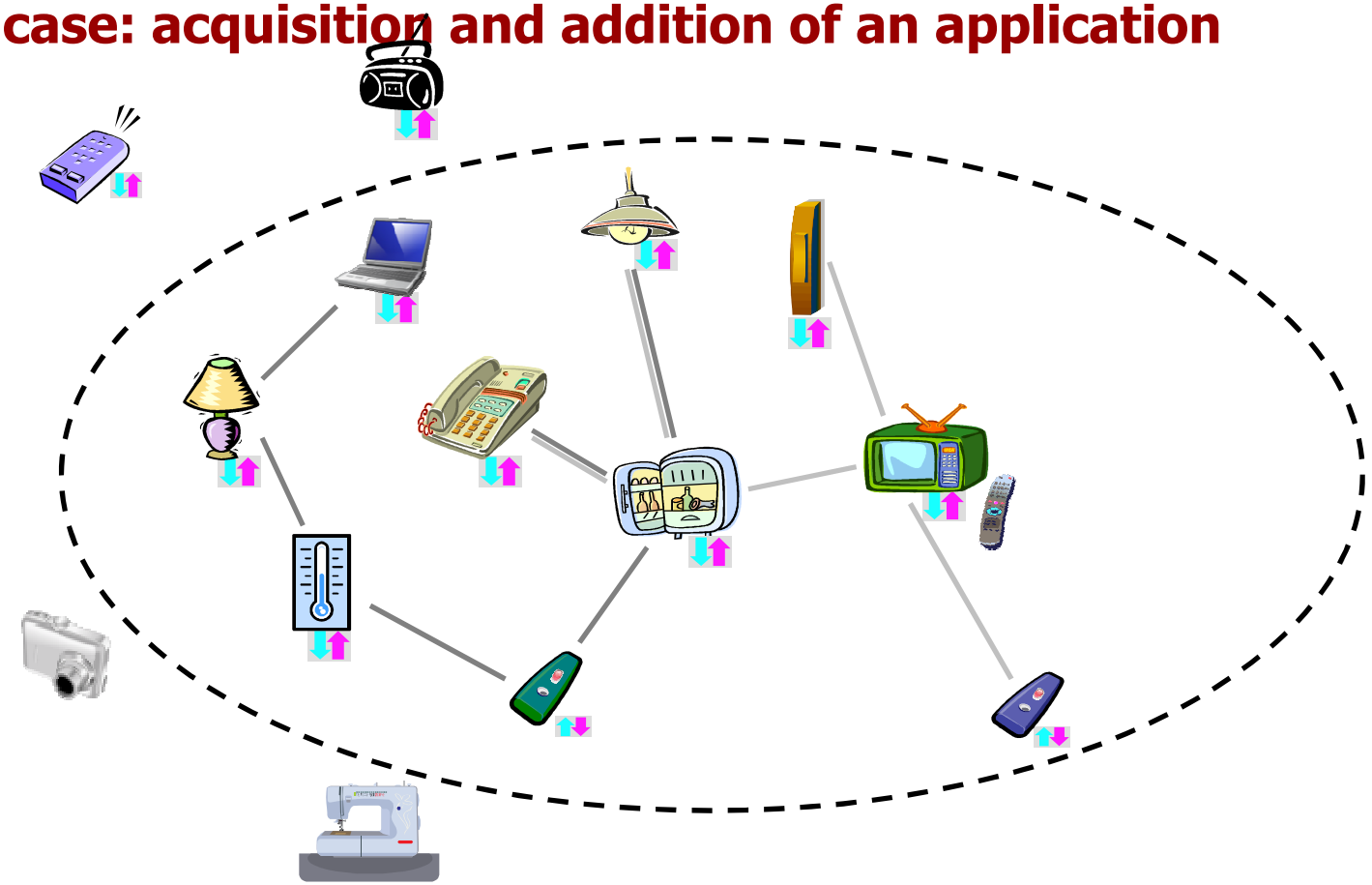
Uruchamiamy aplikację. Za chwilę pojawi się drzewo agentów.

Use case: acquisition and addition of an application



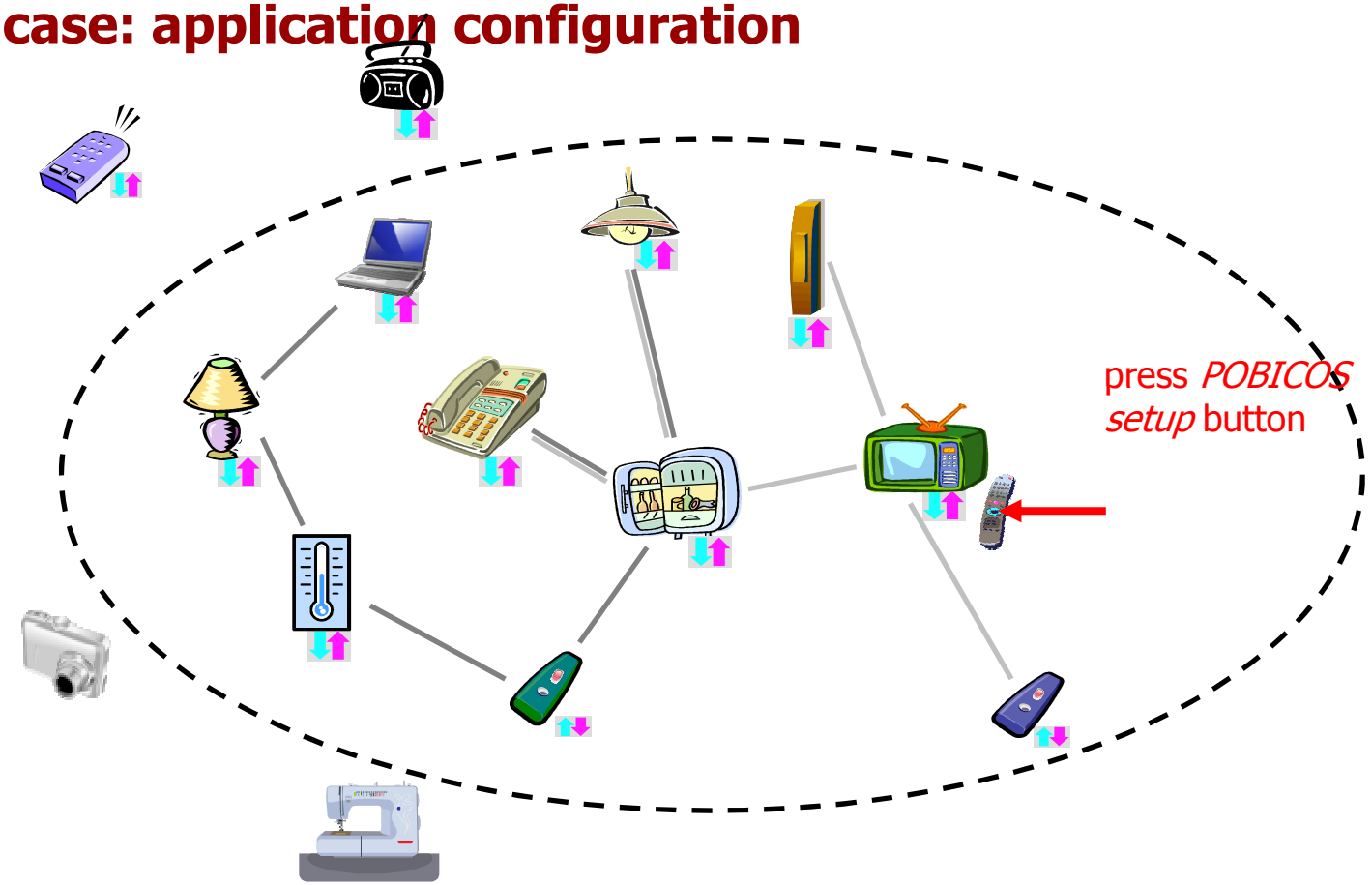
Aplikacja jak najlepiej wykorzystuje dostępne zasoby aby dostarczyć funkcjonalność.

Use case: acquisition and addition of an application



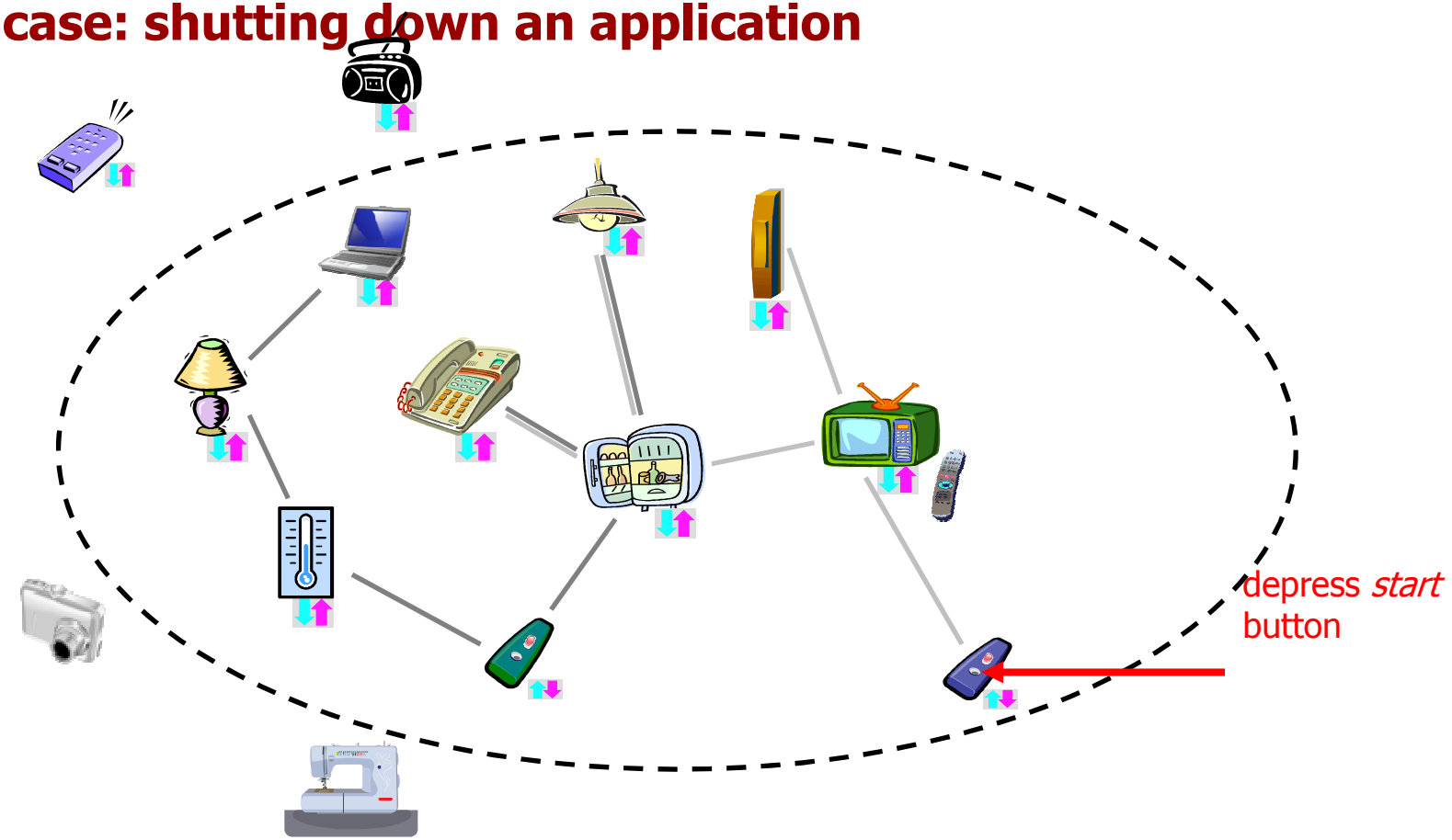
Dwie aplikacje współużywają zasobów dostarczanych przez obiekty.

Use case: application configuration



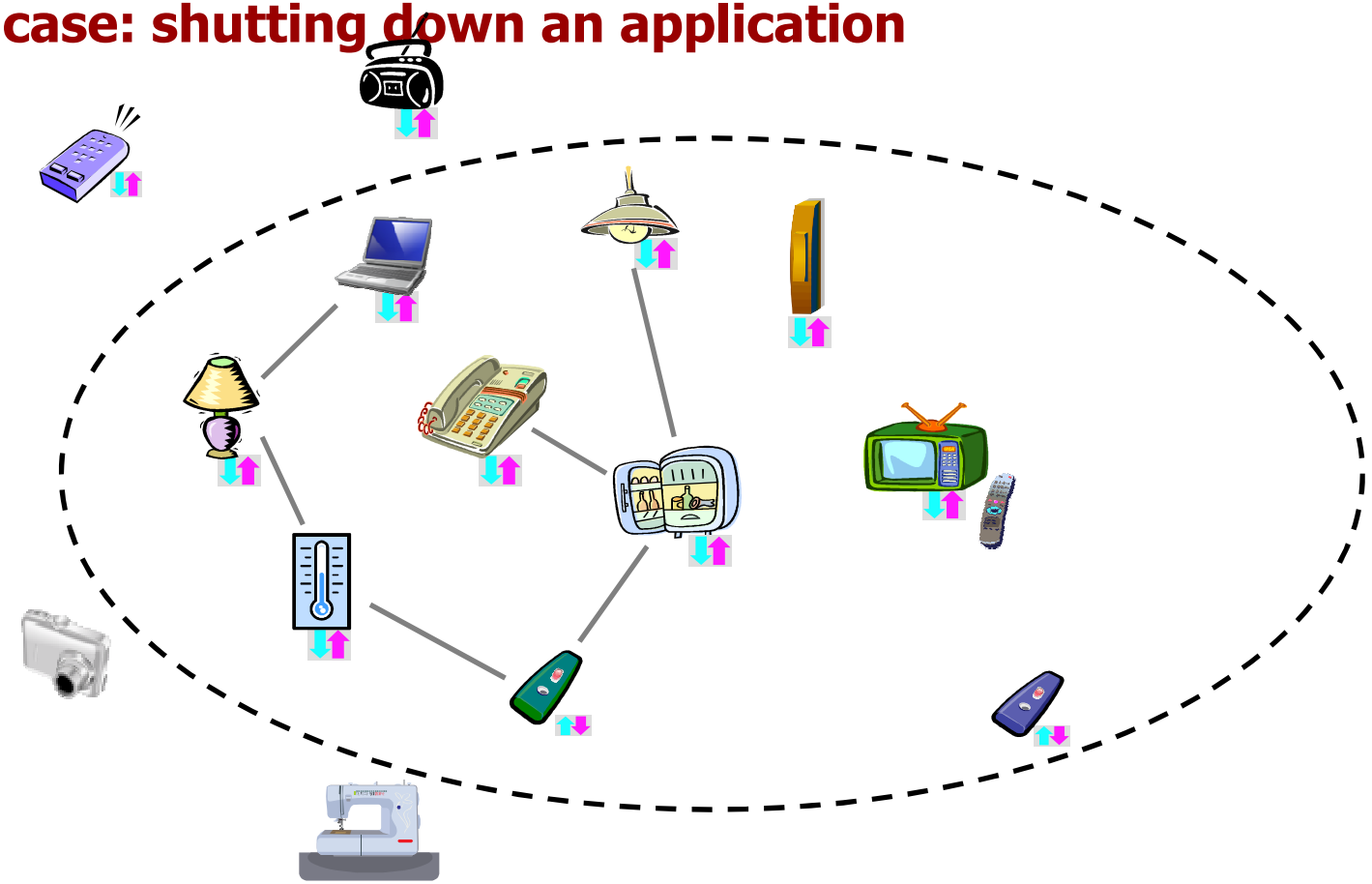
Ustawiamy parametry wybranej aplikacji.

Use case: shutting down an application



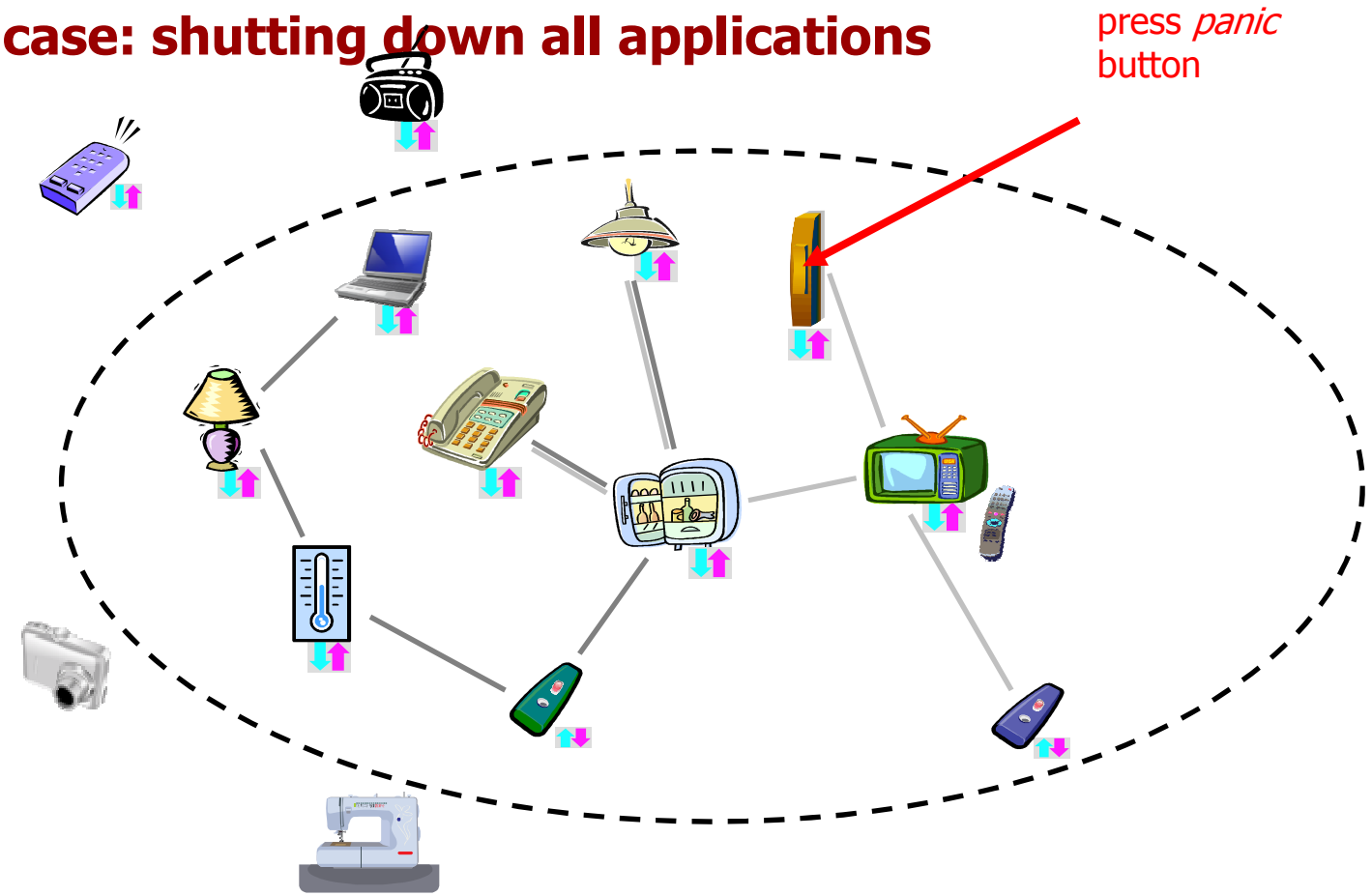
Kończymy pracę aplikacji.

Use case: shutting down an application



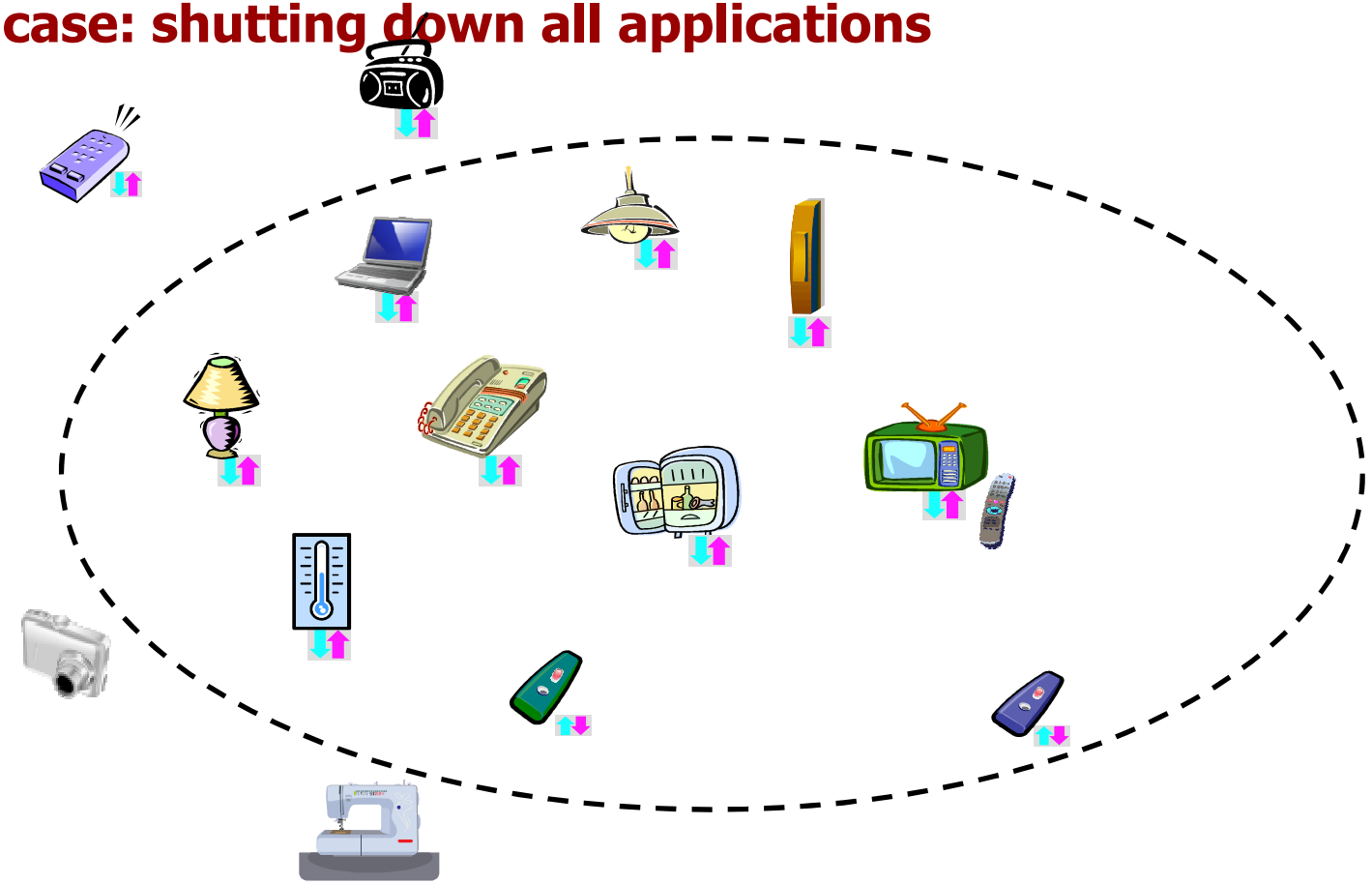
Drzewo agentów zniknęło.

Use case: shutting down all applications



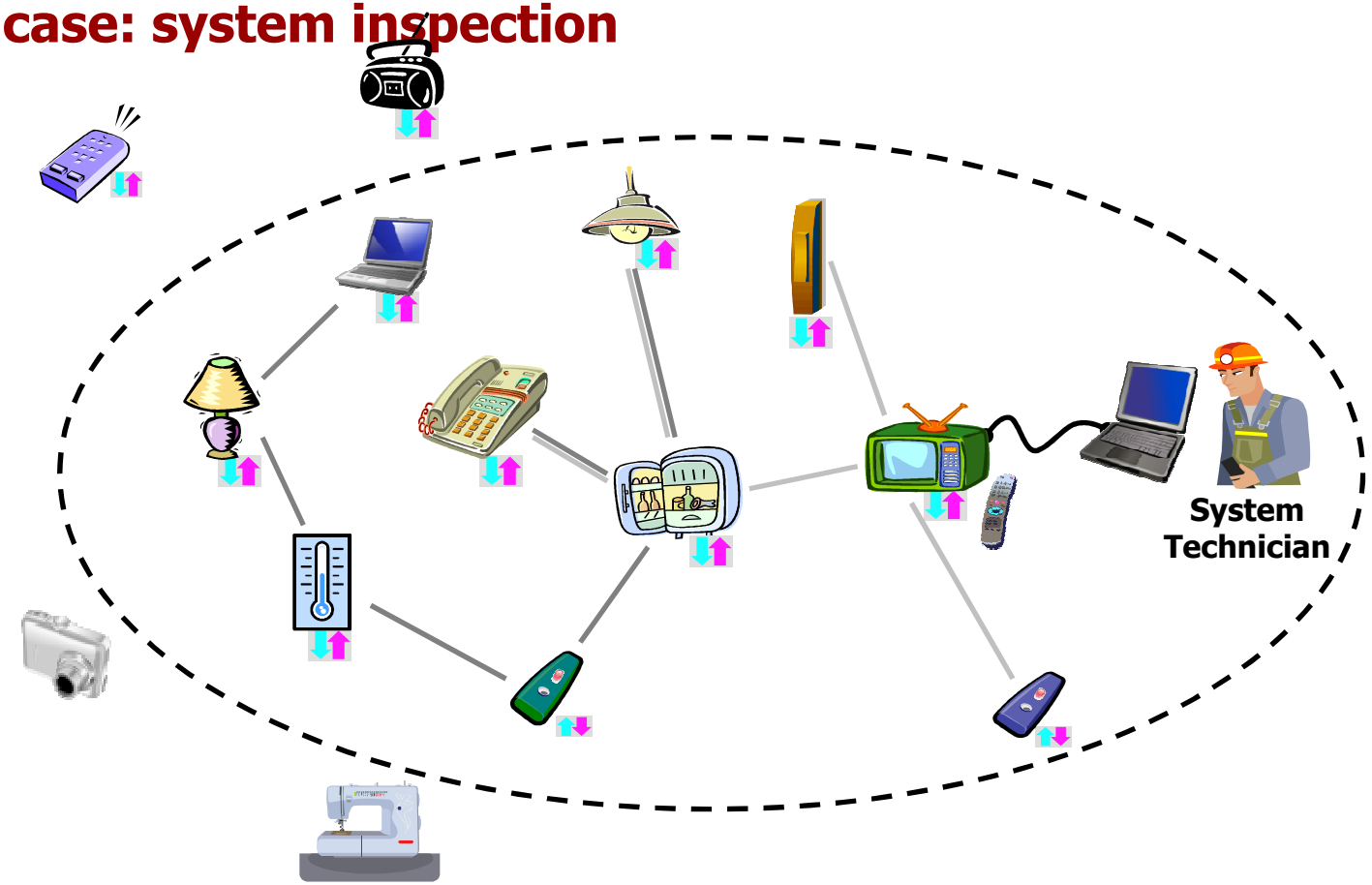
Kończymy pracę wszystkich aplikacji.

Use case: shutting down all applications



Wszystkie drzewa agentów zniknęły.

Use case: system inspection



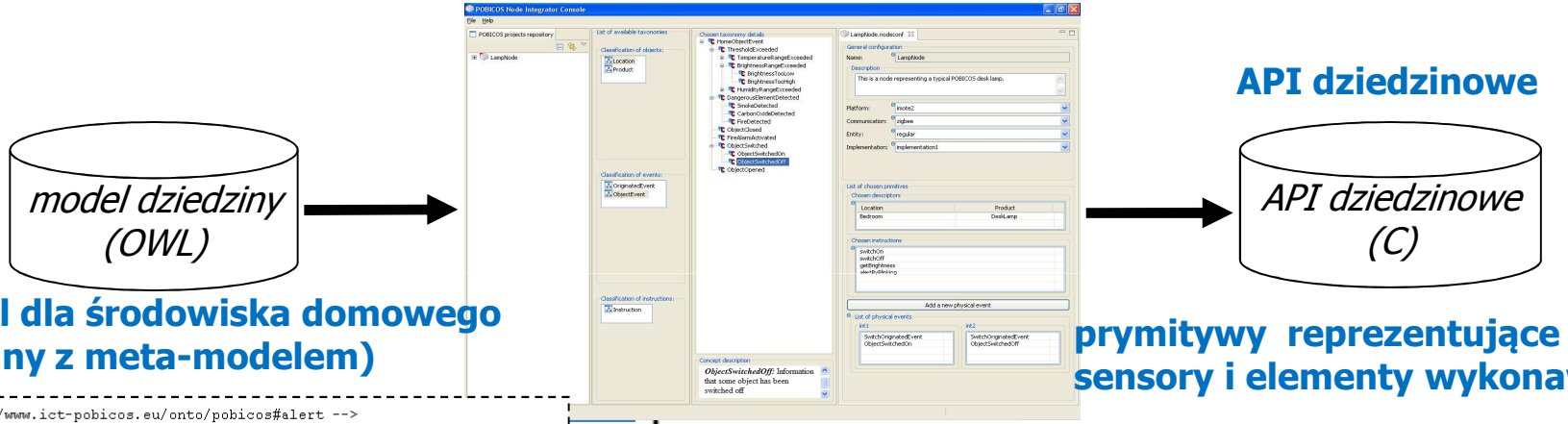
W razie potrzeby technik sprawdza stan naszej społeczności obiektów.

Meta-model dla modelu dziedziny

- Ze względu na wielokrotne użycie, platforma dla aplikacji kontekstowych powinna być do zastosowania w wielu dziedzinach
- Powinna ona być zatem neutralna co do dziedziny („generyczna") … oraz przystosowywana do dziedziny za pomocą modelu kontekstu (dziedziny)
- Uzupełnienie generycznej warstwy pośredniej modelem kontekstu nadaje warstwie pośredniej pełną funkcjonalność
- Modele kontekstu (dziedziny) muszą być zgodne z **meta-modelem**
  - meta-model określa strukturę modelu zgodnego z warstwą pośrednią

Generowanie API z modelu dziedziny (przykład: POBICOS)

kompilator modelu



API dziedzinowe

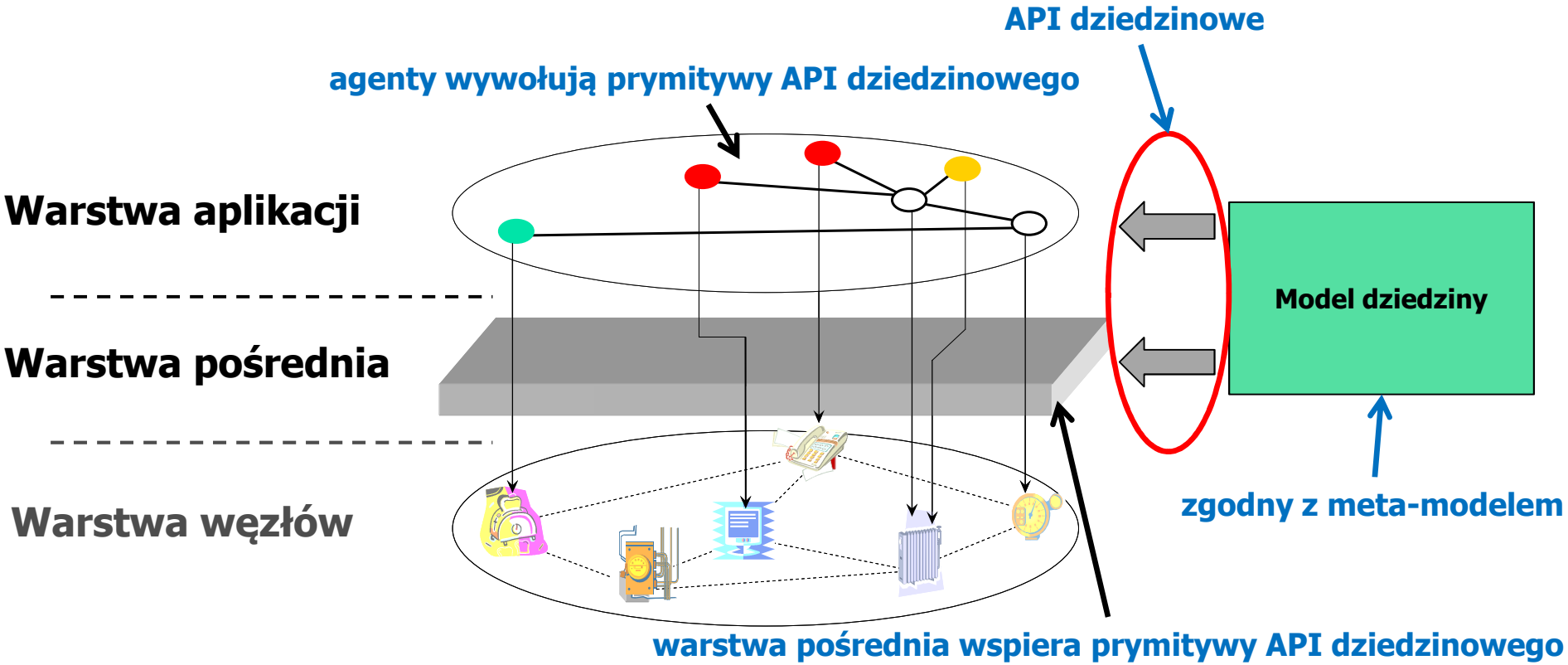
model dla środowiska domowego (zgodny z meta-modelem)

prymitywy reprezentujące obiekty, sensory i elementy wykonawcze

```
<!-- http://www.ict-pobicos.eu/onto/pobicos#alert -->

<owl:Class rdf:about="#alert">
    <rdfs:subClassOf rdf:resource="#Instruction"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasArgument"/>
            <owl:someValuesFrom rdf:resource="#alertArgument"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="&xsd;string"
        >Alert a message (by unspecified means)</rdfs:comment>
</owl:Class>



<!-- http://www.ict-pobicos.eu/onto/pobicos#alertArgument -->

<owl:Class rdf:about="#alertArgument">
    <rdfs:subClassOf rdf:resource="#Argument"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasDomainValue"/>
            <owl:someValuesFrom rdf:resource="#TextualMessage"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment
        >&quot;The text to be communicated to the users during alerting&quot;
    </rdfs:comment>
    <argNumber>1</argNumber>
</owl:Class>
```

```
// From the OriginatedEvent Taxonomy
#define PONGE_ORIGINATED_EVENT_ENVIRONMENT_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_HUMAN_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_SWITCH_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_HUMAN_PRESENCE_SENSOR_EVENT
#define PONGE_ORIGINATED_EVENT_SENSOR_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_SMOKE_DETECTOR_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_HUMIDITY_SENSOR_ORIGINATED_EVENT
#define PONGE ORIGINATED EVENT TEMPERATURE SENSOR ORIGINATED EVENT
#define PONGE_ORIGINATED_EVENT_CARBON_OXIDE_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_BRIGHTNESS_SENSOR_ORIGINATED_EVENT
#define PONGE_ORIGINATED_EVENT_MACHINE_ORIGINATED_EVENT
```

Model dziedziny a warstwa pośrednia (przykład POBICOS)



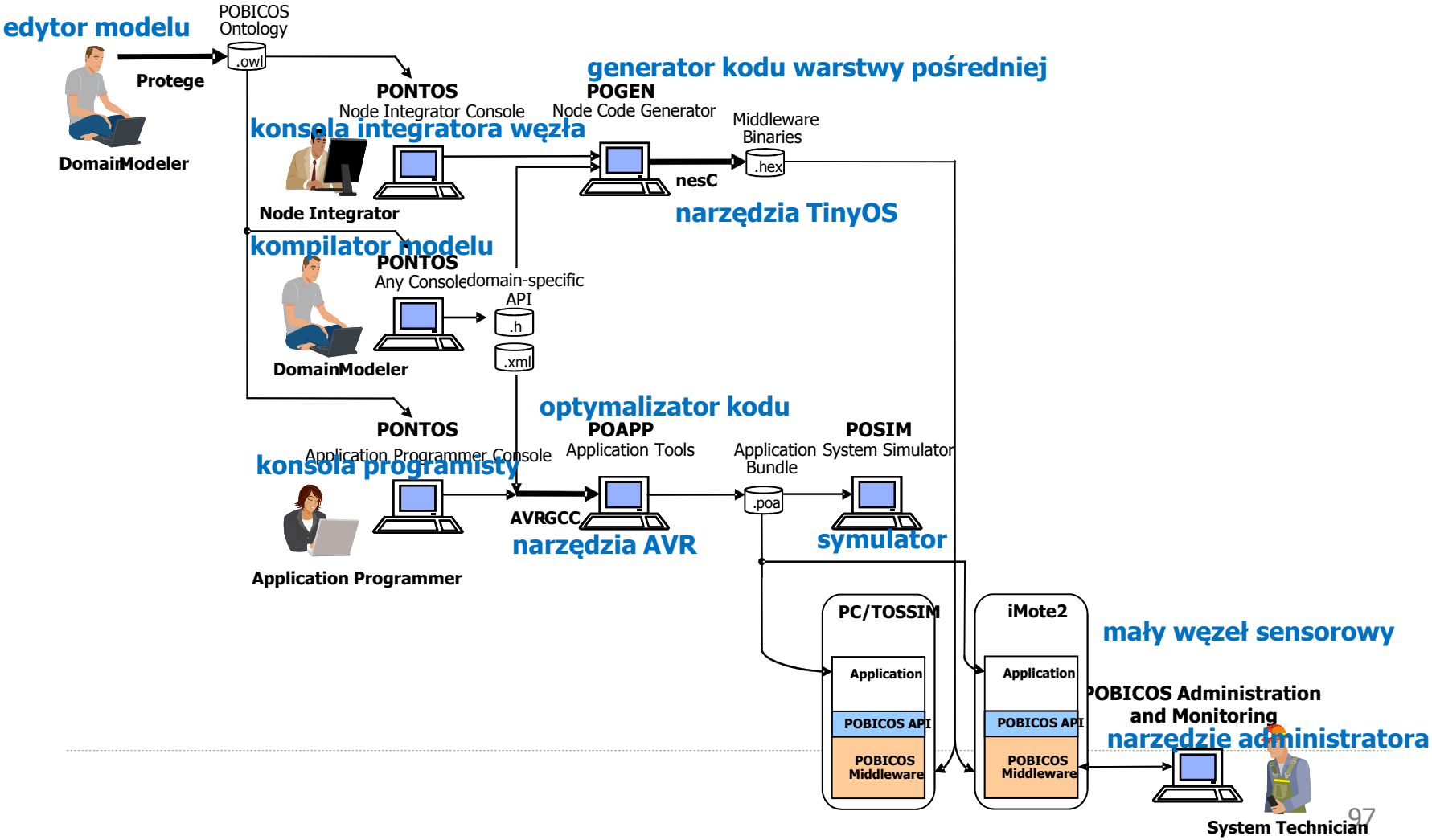
Specyfikacja API (przykład: POBICOS, API ogólne)

API ogólne (wspierane dla wszystkich dziedzin), częściowo nam już znane

events	*instructions*	
ChildCreatedEvent	CreateGenericAgent	SendCommand
ChildCreationTimeoutEvent	CreateNonGenericAgents	GetCommand
ChildUnreachableEvent	Release	SendReport
CommandArrivedEvent	GetChildInfo	GetReport
ReportArrivedEvent	GetMyID	CreateReportList
InitEvent	GetConfigSetting	DestroyReportList
FinalizeEvent	EnableEvent	AddReport
ConfigSettingsChangedEvent	DisableEvent	GetNextReport
TimeoutEvent	SetTimer	DbgStr
	GetTimerID	DbgUInt32

POBICOS API = API ogólne + API dziedzinowe (dla wybranej dziedziny)

Narzędzia (przykład: POBICOS)



Czytelnia

- S. Krakowiak *Middleware Architecture with Patterns and Frameworks Rozdz. 1 An Introduction to Middleware* http://proton.inrialpes.fr/~krakowia/MW-Book/ PRZYSTĘPNE WPROWADZENIE DO WARSTW POŚREDNICH
- J. Indulska , K.Henricksen *Context-Awareness* w *The Engineering Handbook of Smart Technology for Aging, Disability, and Independence*, Wiley, 2008 BARDZO DOBRE WPROWADZENIE DO TEMATYKI KONTEKSTU, Z UWZGLĘDNIENIEM PROBLEMATYKI WARSTW POŚREDNICH DLA APLIKACJI KONTEKSTOWYCH
- M. Baldauf, Sch. Dustdar, F. Rosenberg *A survey on context-aware systems* Int. J. Ad Hoc Ubiquitous Comput. 2, 4 (June 2007) KOLEJNE DOBRE WPROWADZENIE
- L. Mottola, G. P. Picco *Programming wireless sensor networks: Fundamental concepts and state of the art* ACM Comput. Surv. 43, 3, (April 2011) OBSZERNY PRZEGLĄD WARSTW POŚREDNICH DLA SIECI WSN/WSAN, Z NACISKIEM NA ABSTRAKCJE PROGRAMISTYCZNE
- Domaszewicz, J.; Lalis, S.; Paczesny, T.; Pruszkowski, A.; Ala-Louko, M., *Graspable and resource-flexible applications for pervasive computing at home,* Communications Magazine, IEEE , vol.51, no.6, pp.160-169, June 2013 WPROWADZENIE DO KONCEPCJI PIGUŁKI APLIKACYJNEJ (APPLICATION PILL)
- Lalis, S.; Domaszewicz, J.; Pruszkowski, A.; Paczesny, T.; Ala-Louko, M.; Taumberger, M.; Georgakoudis, G., Lekkas, K., *Tangible Applications for Regular Objects: An End-User Model for Pervasive Computing at Home,* Proceedings of UBICOMM 2010, pp. 385-390, October 2010 MODEL INTERAKCJI Z UŻYTKOWNIKIEM DLA SYSTEMU POBICOS

DZIĘKUJĘ!