Inżynieria oprogramowania

Wykład 9

Projektowanie oprogramowania

Metody projektowania oprogramowania

Cztery kategorie metod:

1. zorientowane na dane,
2. zorientowane na funkcje,
3. zorientowane obiektowo,
4. formalne.

Dwie pirwsze kategorie są to metody klasyczne.

Metody projektowania oprogramowania, cd.

- Wszystkie metody mają ten sam cel – przekształcenie zestawu wymagań dotyczących przetwarzania danych do postaci programu komputerowego.
- Jedną z wad głównych metod projektowania jest to, że są one oparte na procesie, a nie na rygorystycznej nauce – nie istnieje matematycznych dowodów poprawności oprogramowania.
- Więc, jakość projektu znajduje się przede wszystkim pod wpływem intuicji i doświadczenia projektanta.

Metody projektowe zorientowane na dane

- Projektowanie zorientowane na dane jest również znane jako **inżynieria informacji**.
- Analiza jest przeprowadzana na encjach danych systemowych, aby wyodrębnić wymagania do danych.
- Wymogi dotyczące danych następnie napędzają projekt oprogramowania.

Metody projektowe zorientowane na dane, cd.

- Encje są ustalane dla każdego podsystemu, wówczas są badane wzajemne relacje encji w celu opracowania dodatkowych encji, potrzebnych do obsługi relacji.
- Iteracyjny proces analizy trwa do momentu, aż relacje encji nie można będzie rozszerzać.

Metody projektowe zorientowane na dane, cd.

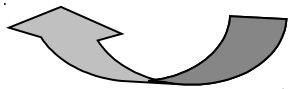
Projektowanie zorientowane na dane jest przydatne dla systemów, które przetwarzają duże ilości danych.

Przykład: aplikacje bazodanowe bankowe.

Takie systemy były programowane wcześniej za pomocą języków proceduralnych, np. COBOL'u.

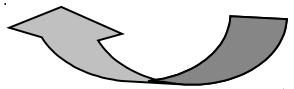
Proces inżynierii informacji

Planowanie



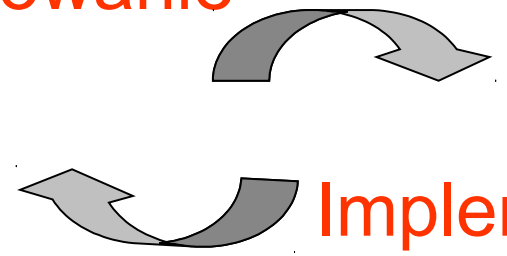
Analiza

1. Rozpoznaj czynniki planu strategicznego
   a. Cele
   b. Czynniki sukcesu
   c. Obszary problemowe
2. Rozpoznaj obiekty planu korporacyjnego
   a. Jednost. orzanizac.
   b. Lokalizacje
   c. Funkcje biznesowe
   d. Typy encji
3. Opracuj model przedsięwzięcia
   a. Dekompozycja funkcji
   b. Diagram encja-związek (E-R)
   c. Matryca log. projektu



Projektowanie

1. Opracuj model konceptualny (detalizacja diagramu E-R)
2. Opracuj model procesowy (diagramy przepływu danych)



Implementacja

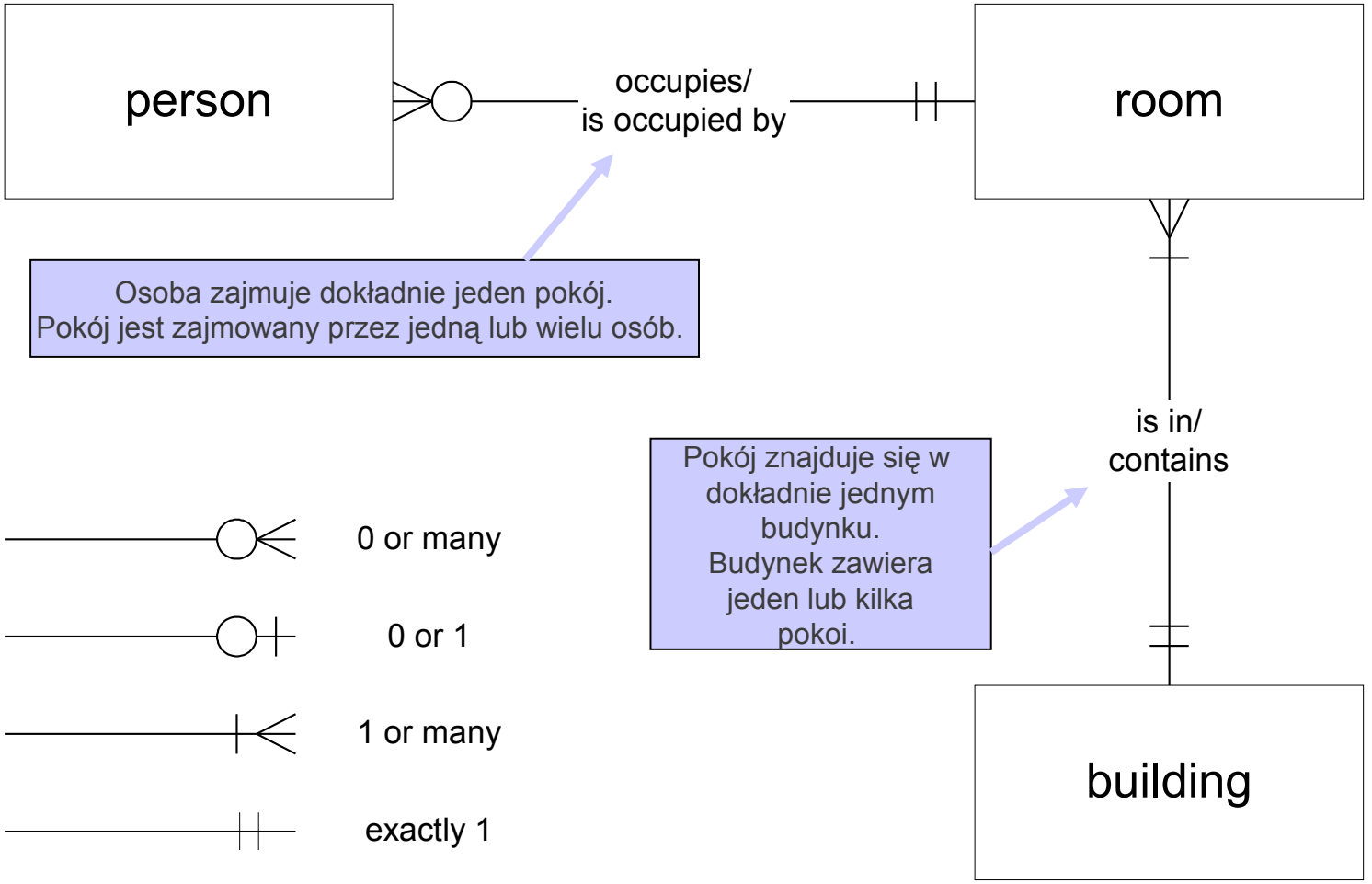
1. Zaprojektuj bazy danych (normalizacja relacji)
2. Zaprojektuj procesy
   a. Diagramy aktywności
   b. Interfejsy użytkownika: menu, ekrany, raporty

1. Zbuduj definicję baz danych
   (tabele, indeksy)
2. Wygeneruj aplikację
   (kod źródłowy, testy)

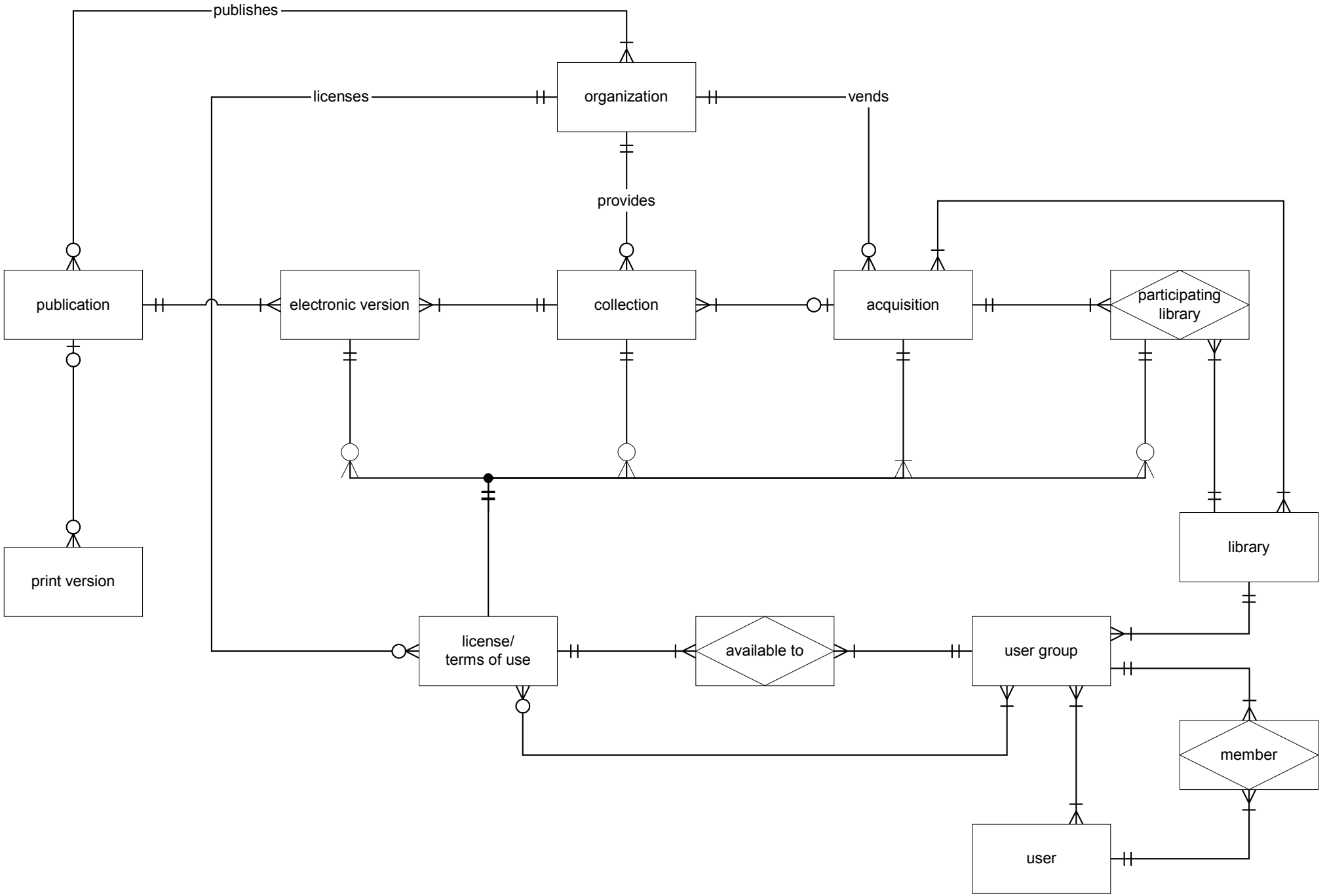
Opracowanie diagramu encja-związek

Materiał pochodzi z: Nathan D.M. Robertson, Entity-Relationship Diagram for Electronic Resource Management, NISO/DLF Workshop on Standards for Electronic Resource Management, 10 May 2002

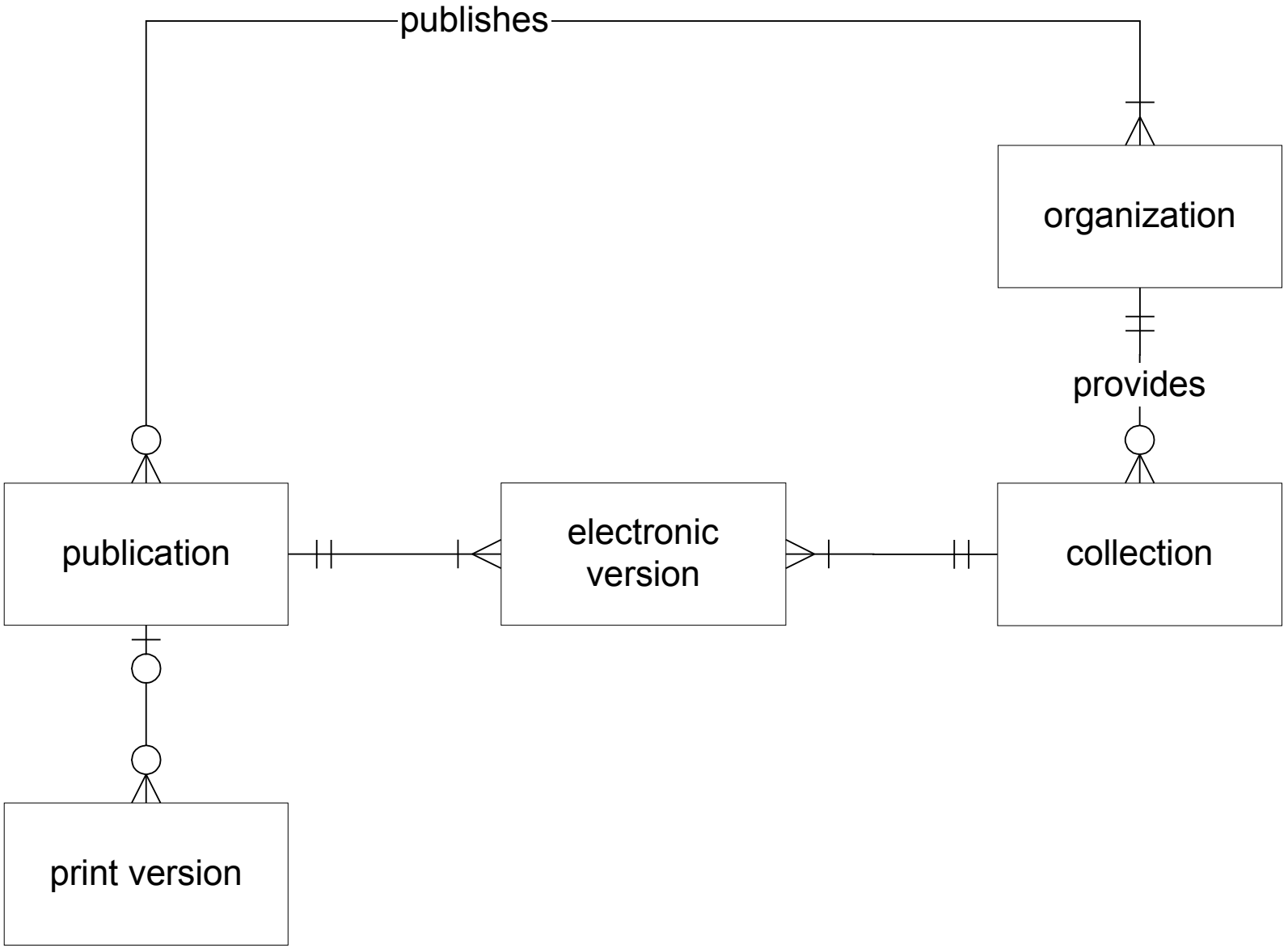
Notacja modelu encja-związek w ramach „inżynierii informacji”



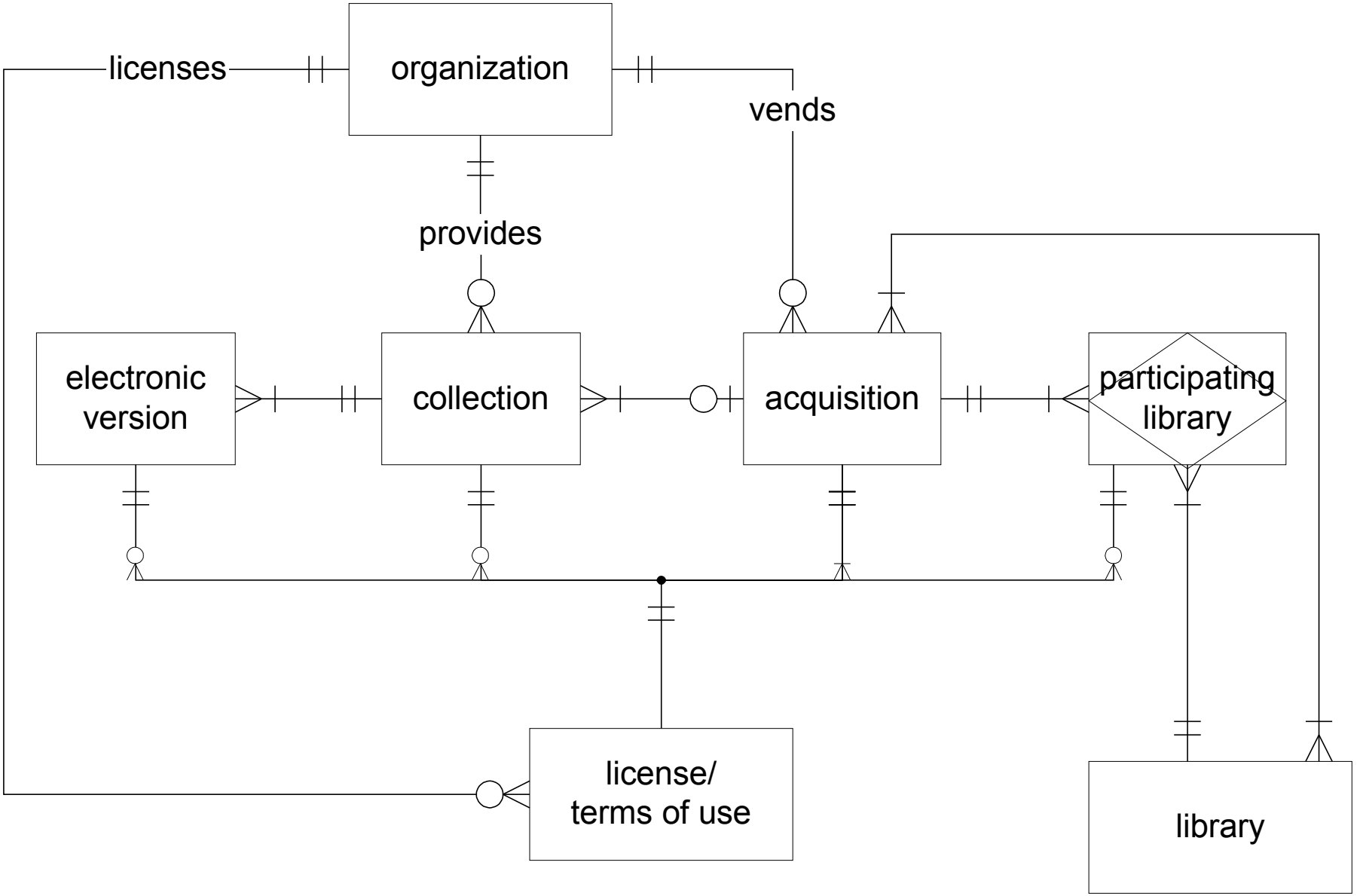
System zarządzania zasobami elektronicznymi



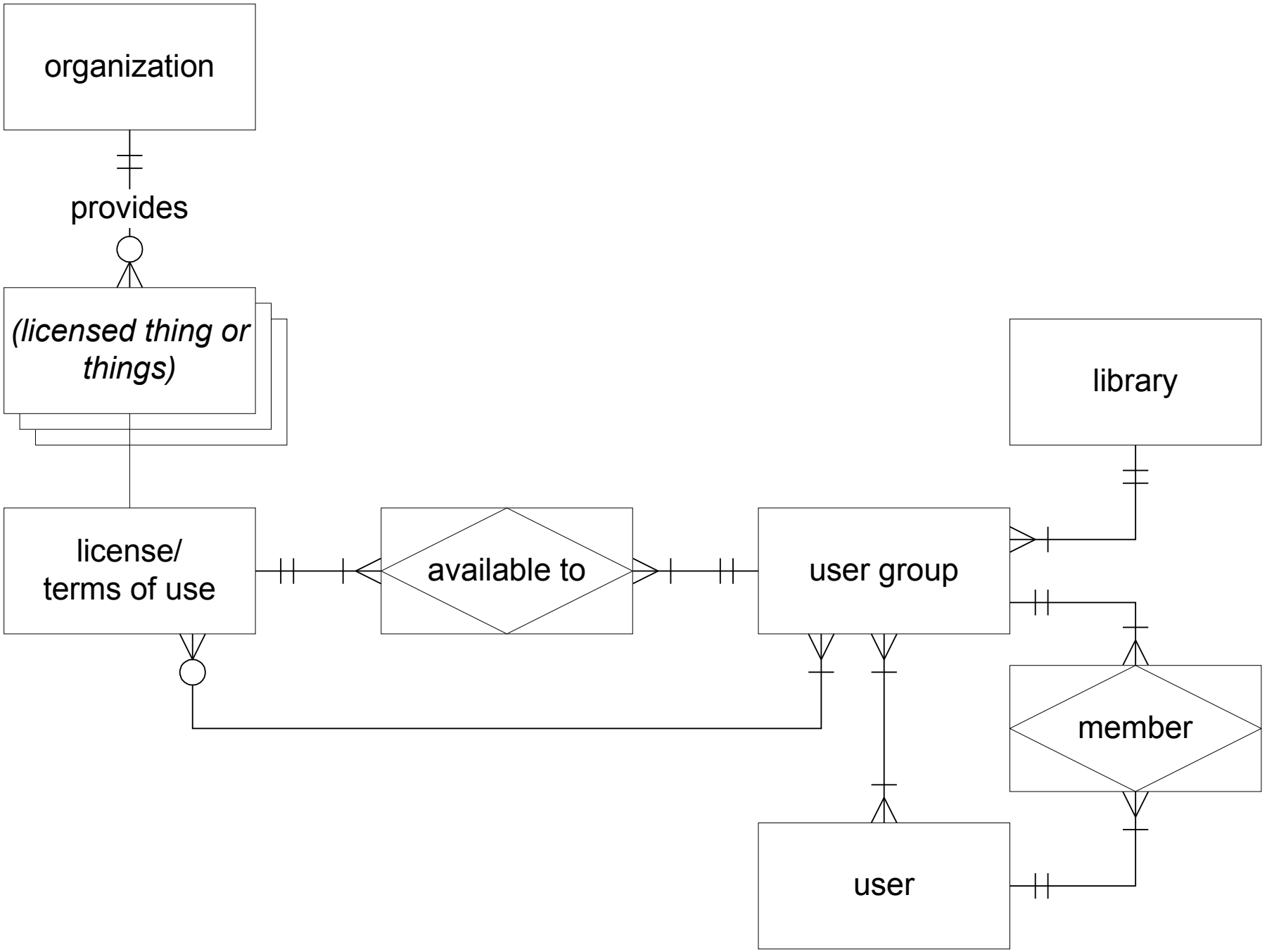
Encje i relacje dla modułu opisu



Encje i relacje dla modułu licencjonowania



Encje i relacje dla modułu dostępu



Metody projektowe zorientowane na funkcje

- Projektowanie zorientowane na funkcje jest również znane jako **analiza strukturalna**.
- Podstawą metody jest dekompozycja procesów, która służy do wyodrębniania wymagań procesowych systemu.
- Wymagania procesowe następnie napędzają projekt oprogramowania.

Metody projektowe zorientowane na funkcje, cd.

- Funkcje procesowe są identyfikowane, a następnie przypisywane do fizycznych grup. Zatem są poddawane reorganizacji w celu skutecznej implementacji programu.
- Konstrukcja parametrów danych jest dopracowywana po pełnej dekompozycji modułów procesu.

Metody projektowe zorientowane na funkcje, cd.

Metody zorientowane na funkcje są przydatne dla systemów z intensywnymi procesami.

Były one początkowo programowane za pomocą języków proceduralnych, np. FORTRAN'a.

Projektowanie obiektowe

- Rozwój metod zorientowanych obiektowo (OO metody), jak praktycznie wszystkie pojęcia w informatyce, był ewolucyjny.
- Metody OO są hybrydą projektowania zorientowanego na dane i na funkcje.
- Metody OO dzielą system na obiekty, które są spójnymi jednostkami pokrewnych metod i atrybutów.

Projektowanie obiektowe: historia

- Terminy "obiekt" i "atrybut" sięgają prac z zakresu sztucznej inteligencji wykonanych w 50-tych.
- Prekursorem nowoczesnego projektowania obiektowego było wprowadzenie do języka Simula hermetyzacji w roku 1966.
- Obecnie projektowanie obiektowe jest tak samo popularne jak metody strukturalne były popularne w latach 70-tych.

Projektowanie obiektowe: różnice w podejściu

- Wszystkie OO metody są podobne, ich celem jest przekształcenie zestawu wymagań do modelu obiektów.
- Jednakże różnią się one w terminologii oraz liczbie warstw i elementów w modelu obiektowym.
- Metody OO wykorzystują iteracyjne i przyrostowe podejście do opracowania obiektowego modelu systemu.
- Różne techniki mogą być wykorzystywane do projektowania elementów każdej warstwy w modelu obiektowym.

Projektowanie obiektowe: różnice w podejściu, cd.

- Ogólnie rzecz biorąc, model obiektowy można podzielić na 3 warstwy:
  - model domeny jest wierzchnia warstwa,
  - model interfejsu jest warstwa środkowa,
  - model implementacji jest dolna warstwa.
- Ogólnie rzecz biorąc, projekt rozpoczyna się od modelu domeny, zatem projektowana jest warstwa interfejsu, a następnie warstwy implementacji.

Projektowanie obiektowe: model domeny

- Model domeny jest najbardziej abstrakcyjną warstwa modelu obiektowego.
- Być może, jest to najbardziej krytyczna warstwa, ponieważ błędy projektu i wymogów, wprowadzone tutaj, będą propagować na cały model obiektowy.
- W modelu domeny wyjaśniane są interakcje wysokiego poziomu między klasami obiektów przy użyciu dziedziczenia, współpracy i relacji interfejsowych.

Projektowanie obiektowe: model interfejsu

- Model interfejsu wykorzystuje model domeny do tworzenia bardziej szczegółowych abstrakcyjnych i konkretnych klas obiektów.
- Nowe relacje dziedziczenia i współpracy mogą być również rozwijane w tej fazie.

Projektowanie obiektowe: konstruowanie obiektów

- W fazie implementacji obiekty są tworzone na podstawie diagramów klas, opracowanych w poprzednich dwóch etapach.
- Definiowana jest strukturalna kompozycja metod i atrybutów.
- Do rozwoju metod są wykorzystywane pojęcia (koncepty) projektowania napędzanego procesem.
- Do rozwoju atrybutów – pojęcia projektowania napędzanego danymi.

Projektowanie obiektowe: zastosowanie

- Metody OO są przydatne dla każdego systemu, który zostanie zaprogramowany przy użyciu obiektowego języka, takiego jak C++ czy Java.
- Jednakże szczególnie dobrze nadają się one do rozwoju dużych systemów oprogramowania.

Metody projektowe formalne

- Metody formalne używają formalnego języka do opisania artefaktów oprogramowania, takich jak specyfikacja, projekt architektoniczny lub kod źródłowy.
- Język formalny umożliwia stosowanie dowodów formalnych w celu oceny poprawności artefaktu.

Metody projektowe formalne, cd.

Metody formalne, znane również jako "clean-room approach", używają języka dyskretnej matematyki do udowodnienia poprawności programów.

Wymagania i programy są tłumaczone na matematyczną notację, gdzie mogą one być analizowane za pomocą narzędzi zwanych metodami formalnymi.

Metody formalne, cd.

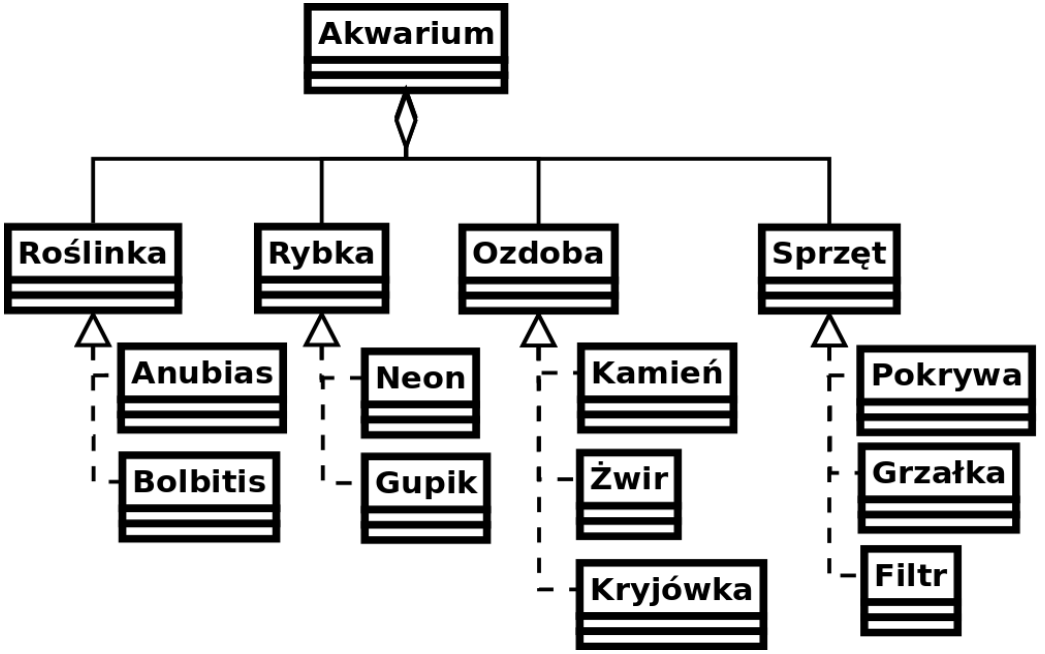
- Formalne metody są uważane za alternatywę dla OO i metod klasycznych, ale ich stosowanie jest kosztowne i roszczenia do obniżenia błędów pozostają niepotwierdzone.
- Jednak zdolność do formalnego sprawdzenia poprawności w artefakcie oprogramowania jest atrakcyjna i badania metod formalnych są w toku.
- Formalne metody zostały wykorzystane do opracowania najbardziej krytycznych sekcji projektu sterowania ruchem lotniczym Urzędu Cywilnego Lotnictwa Wielkiej Brytanii.

Przykład podejścia do projektowania oprogramowania w oparciu o dane

(Data driven development)

Na podstawie http://www.targetprocess.com/blog/2005/12/data-driven-development.html

Obiektowy model akwariumu



- Architektura programu jest bardzo elastyczna. Na przykład, łatwo stworzyć nową rybkę lub roślinkę, dziedzicząc właściwy obiekt i przeciążając tylko kilka metod.
- Problem – pracochłonna modyfikacja kodu: jeśli potrzebujemy dodać nowy atrybut do rybki, to musimy sprawdzać wszystkie istniejące klasy rybek.

C++ kod zainspirowany danymi

```
enum FISH_FOOD
{
INSECTS = 0x01,
WORMS = 0x02,
GRUBS = 0x04,
};

enum BEHAVIOR
{
SHOAL = 0x01,
CAMOUFLAGE = 0x02,
FEAR = 0x04,
};
```

```
struct FISH
{
const char * name;
int effective_size;
int food;
int behavior;
} fish[ ] = {
{"guppy", 10, INSECTS GRUBS, SHOAL},
{"neon", 12, WORMS INSECTS, SHOAL
CAMOUFLAGE FEAR},
{"golden fish", 8, GRUBS, FEAR},
{NULL, 0, 0}
};
```

```
class Fish
{
void Eat() { }
public:
Fish(const char * name) {}
void Move() { }
};

int main()
{
Fish guppy1("guppy");
Fish guppy2("guppy");
return 0;
}
```

- Rozwiązanie problemu – brak dziedziczenia dla każdego typu ryb. Możemy stworzyć jedną klasę Fish, która zostanie zainicjowana z tabeli struktur FISH. Nowy typ rybki to tylko jedna linijka danych, może ona całkiem pochodzić ze specyfikacji wymagań.
- Mankament – utrata elastyczności, trudno stworzyć unikatowe zachowanie każdej rybki.

Rozwój idei optymalizacji danych

```
struct STRATEGY {
int stragey_name;
int food;
int behavior;
} strategies[ ] =
{
{AGRESSIVE, GRUBS | WORMS | INSECTS, SHOAL},
{COMMON, GRUBS | WORMS | INSECTS, SHOAL},
{AFRAID, GRUBS, 0}
}
```

```
struct FISH {
const char * fish_name;
int effective_size;
int strategy;
} fishes[ ] =
{
{"guppy", 10, AGRESSIVE},
{"golden fish", 6, AFRAID},
}
```

- Przez wspólne strategie możemy usunąć powtarzające się dane z tabeli fishes[ ].
- Ponadto, jeśli dodamy więcej właściwości do strategii, to będziemy mieć mniej miejsc do zmodyfikowania.