Проект на архитектурата

- **Проект на архитектурата**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman

Лектор: Доц. д-р Ася Стоянова-Дойчева

Какво е архитектура?

Архитектурата не е оперативен софтуер. Тя е представяне, което дава възможност на софтуерните инженери за:

(1) анализ на ефективността на проекта в изпълнението на изискванията,

(2) обмисляне на архитектурни алтернативи в момент, когато промени в проекта все още са лесни

(3) намаление на риска свързани с изграждането на софтуера.

Защо е важна архитектурата?

- Представянето на софтуерната архитектура, позволява комуникация между всички страни (участници) интересуващи се от разработката на компютърно-базирана система.
- Архитектурата очертава ранни проектни решения, които ще имат влияние върху следващите стъпки от работата на софтуерните инженери.
- Архитектурата показва как системата е структурирана и как нейните компоненти работят заедно [BAS03].

Описание на архитектурата

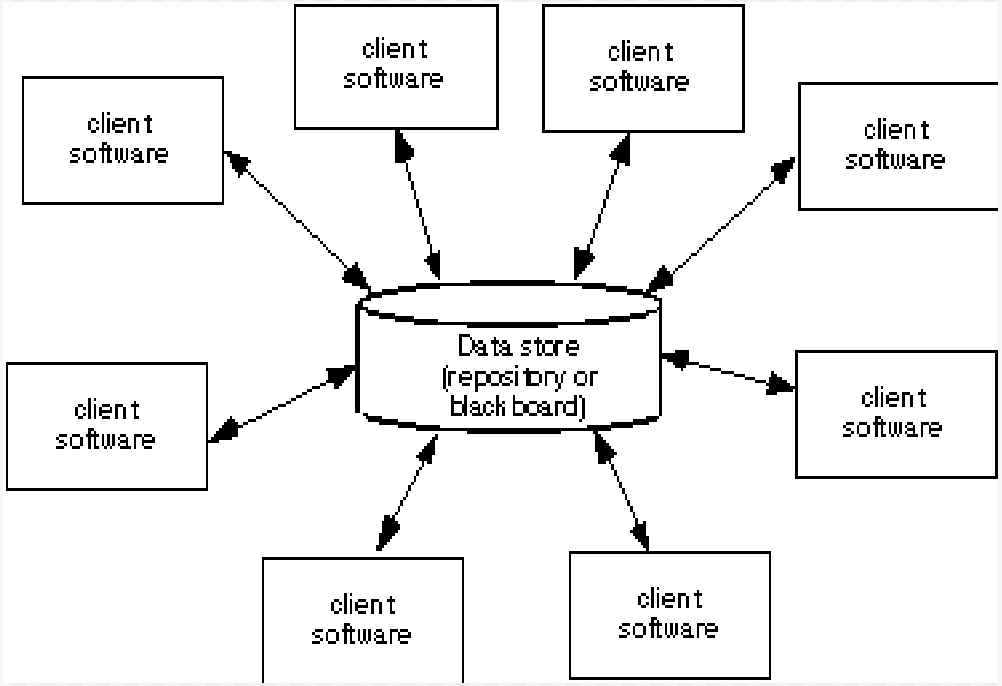
- IEEE Computer Society предлага IEEE-Std-1471-2000, *Recommended Practice for Architectural Description of Software-Intensive System*, [IEE00]
  - Да определи концептуална рамка и речник по време на проектирането на софтуерната архитектура,
  - Да предостави детайлно ръководство за представяне на архитектурно описание
  - Да окуражи архитектурните практики
- IEEE стандарта дефинира *architectural description* (AD) като „множество от продукти за документиране на архитектурата”

Архитектурни стилове

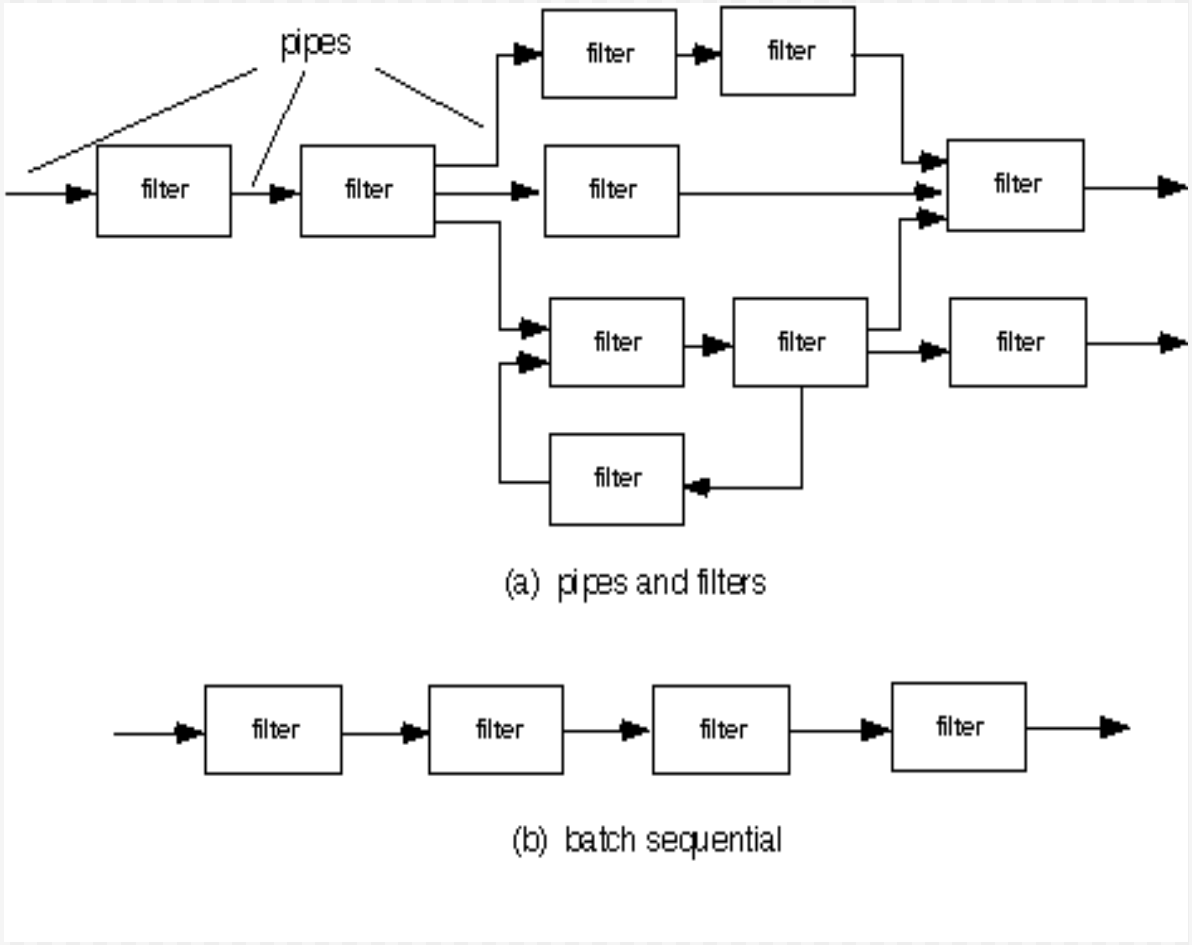
Всеки стил описва категория, която включва: (1) **множество от компоненти** (напр., база данни, изчислителни модули), които изпълняват функциите изисквани от системата, (2) **множество връзки,** които позволяват „комуникация, координация и сътрудничество“ между компонентите, (3) **ограничения,** които дефинират как компонентите могат да бъдат интегрирани, за формиране на системата и (4) **семантични модели** които позволяват на проектанта да разбере общите характеристики на системата чрез разбиране на характеристиките на съставните и части.

- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Object-oriented architectures
- Layered architectures

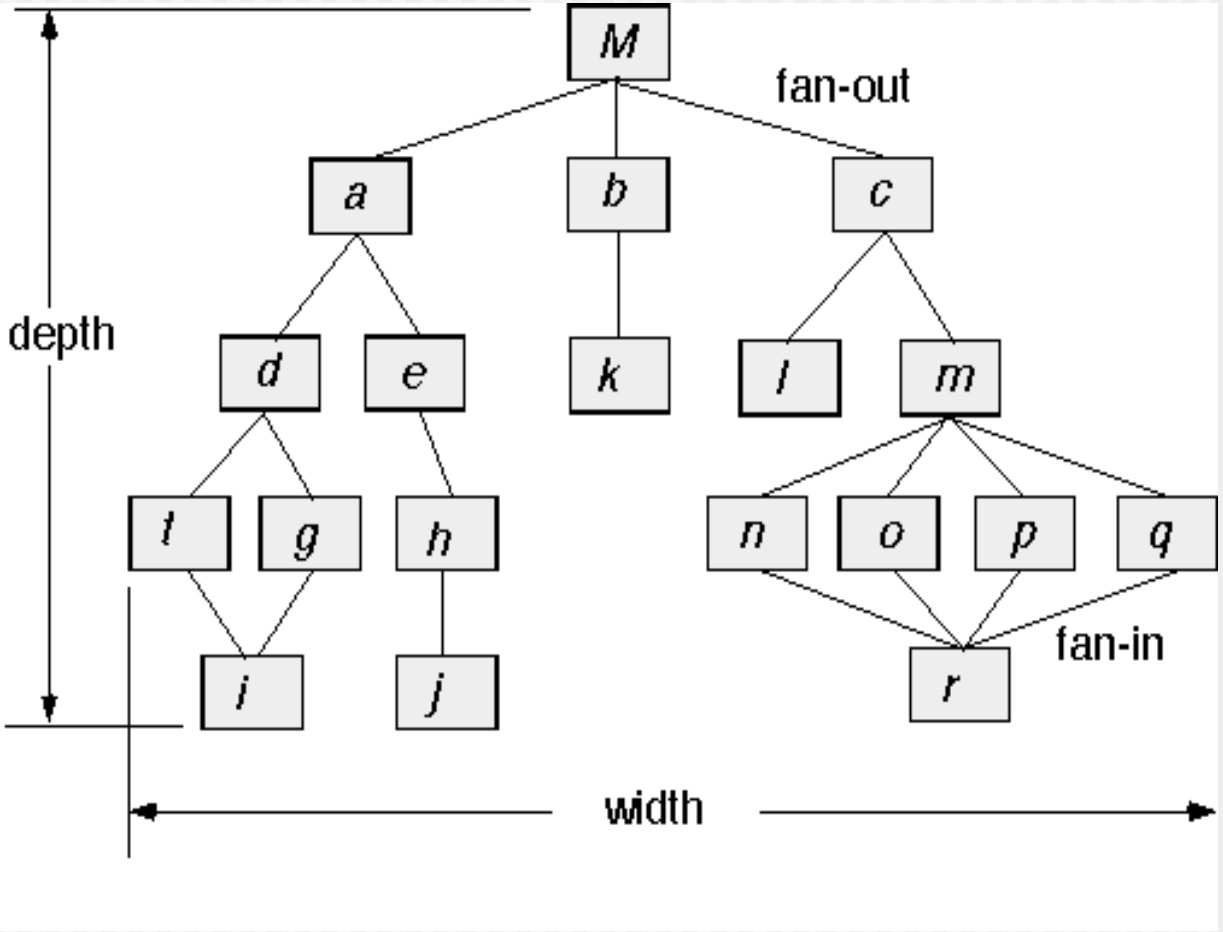
Data-Centered Architecture



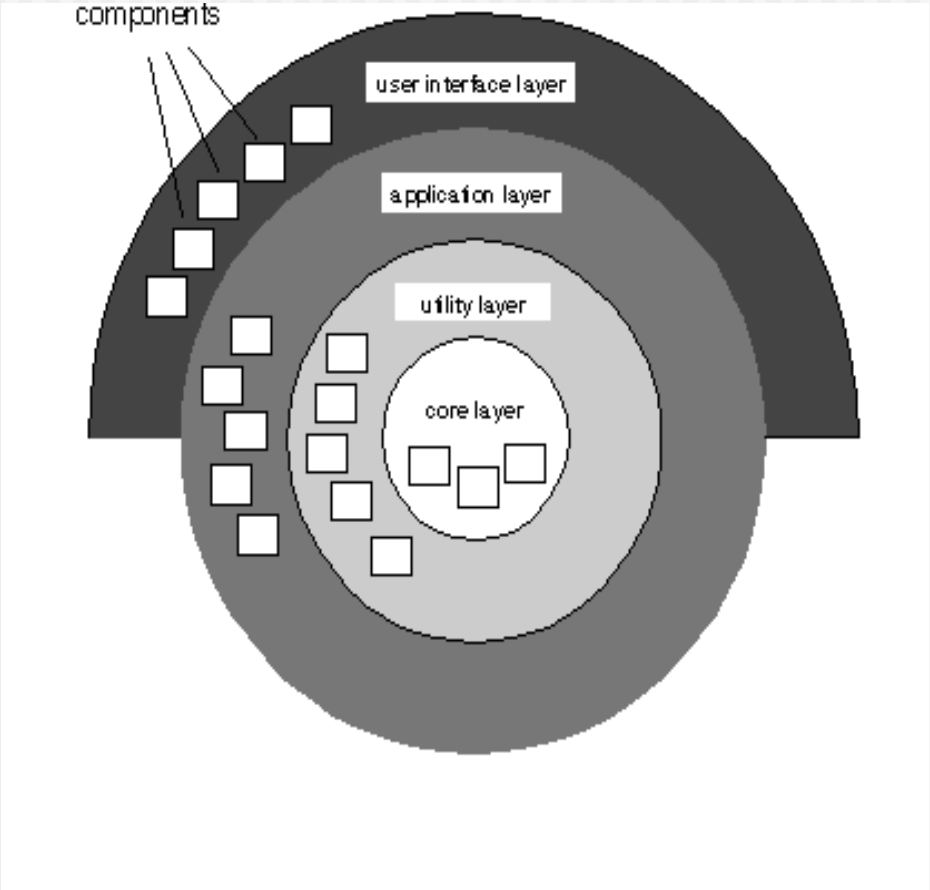
Data Flow Architecture



Call and Return Architecture



Layered Architecture



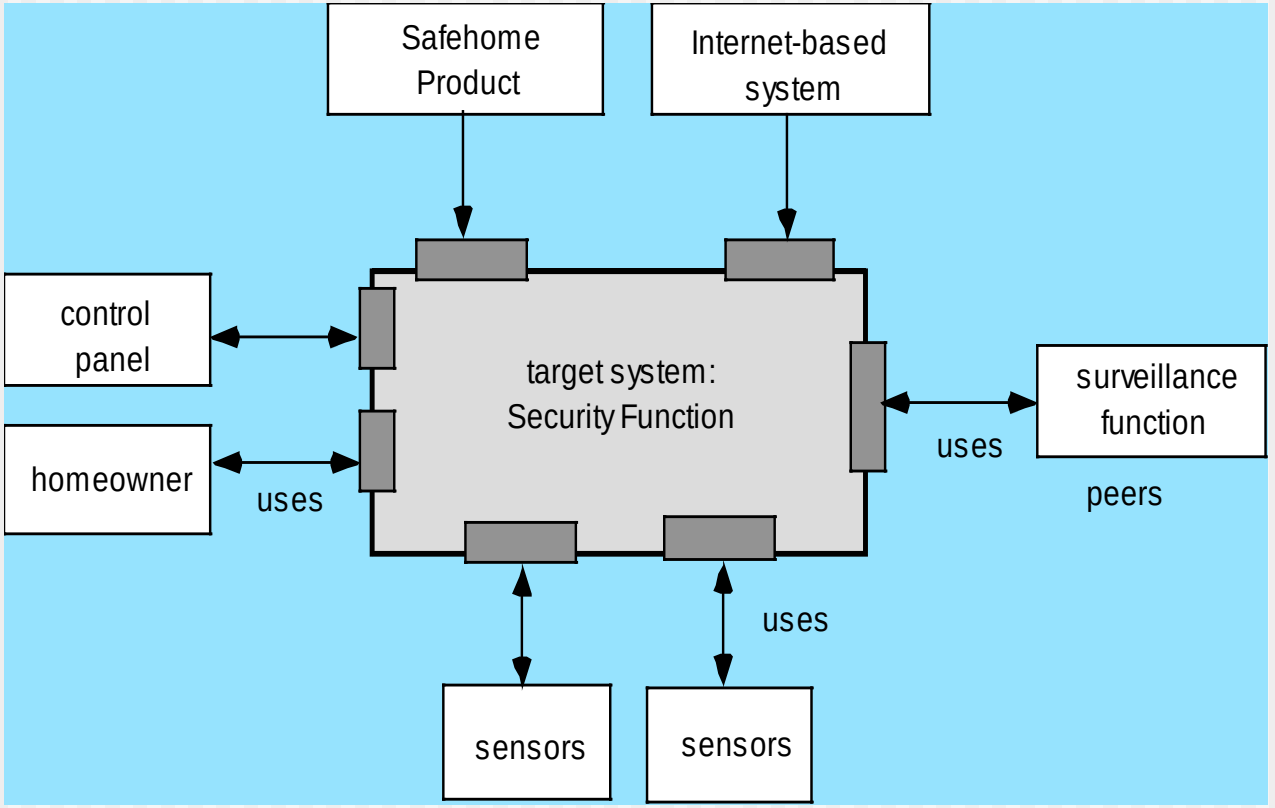
Архитектурни шаблони

- Concurrency— приложенията трябва да се справят с изпълнението на множество задачи, като симулират паралелна обработка:
  - шаблон *operating system process management*
  - шаблон *task scheduler*
- Persistence—Данните се запазват ако оцеляват след приключване на процеса, който ги е създал. Има два често използвани шаблона:
  - *database management system,* който прилага възможностите за съхраняване и извличане на данни на СУБД към архитектурата на приложението
  - *application level persistence,* който вгражда възможности за съхраняване на данни в архитектурата на приложението
- Distribution— подход, при който системи или компоненти от системи комуникират един с друг в разпределена среда:
  - *Брокер* действа като посредник между клиентски и сървърни компоненти.

Проект на архитектурата

- Софтуерът трябва да бъде поставен в контекст
  - проектът трябва да дефинира външните обекти (други системи, устройства, хора), с които софтуера си взаимодейства и естеството на взаимодействие
- Трябва да бъдат идентифицирани множество от archetypes
  - *Archetype е абстракция* (подобна на клас), която представя един елемент от поведението на системата
- Проектантът специфицира структурите на системата, чрез дефиниране и усъвършенстване на компонентите, които имплементират всеки archetype.

Контекст на архитектурата (architectural context diagram)



Archetypes

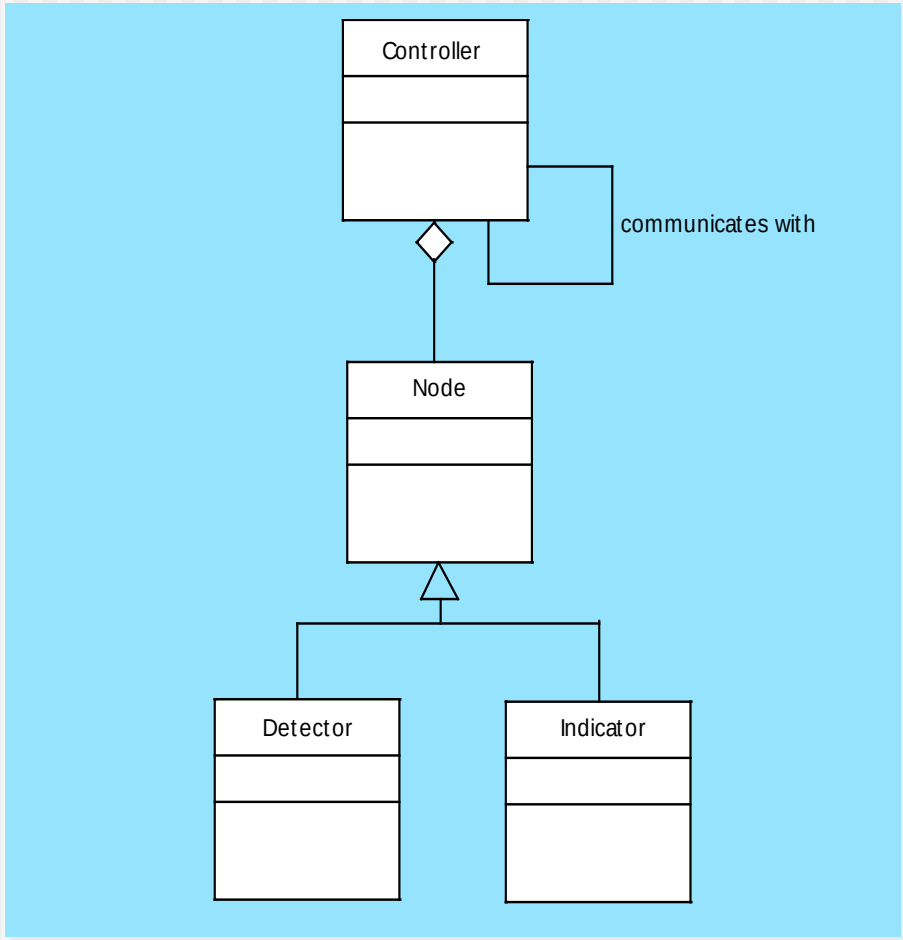
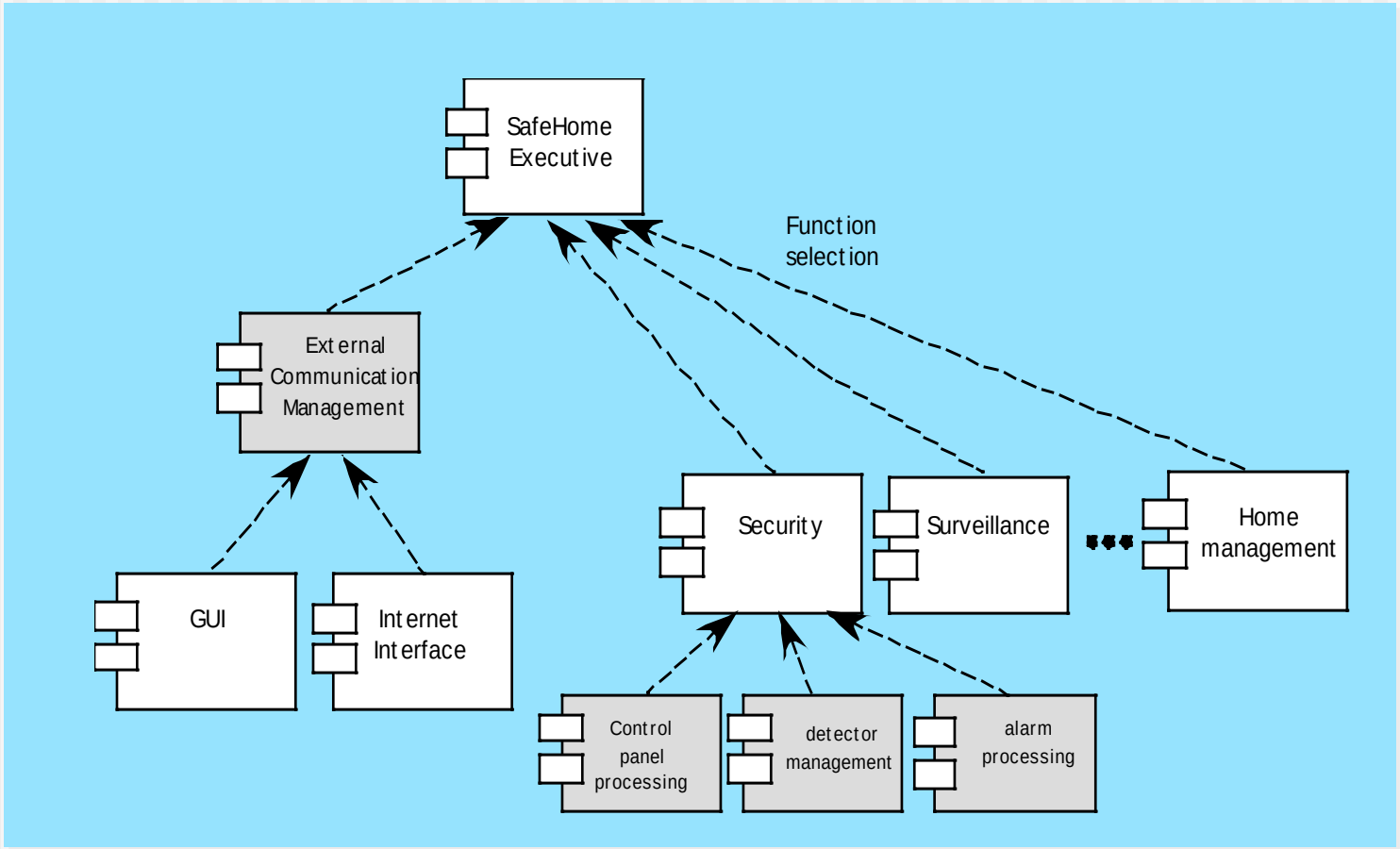
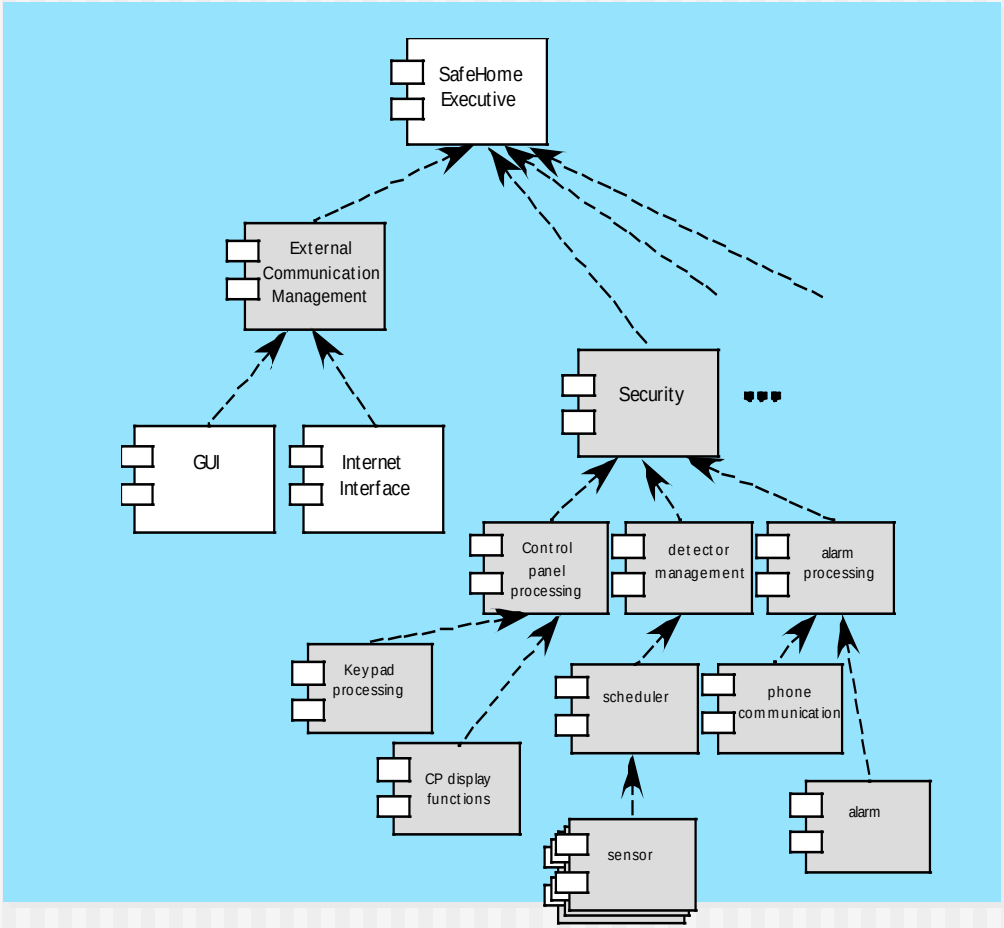


Figure 10.7 UML relationships for real-time systems

Структура на компонентите



Усъвършенстване на структурата на компонентите



ADL

- *Architectural description language* (ADL) предоставя семантика и синтаксис за описание на софтуерна архитектура
- Предоставя възможности на проектантите за:
  - да декомпозират архитектурните компоненти
  - да обединяват компоненти в големи архитектурни блокове и
  - да представят интерфейсите (механизми за свързване) между компонентите.

Анализ на архитектурния проект

1. Събиране на сценарии (use cases).
2. Извличане на изисквания, ограничения и описание на средата.
3. Описание на архитектурните стилове/ шаблони, които са избрани за сценариите и изискванията:
    - модулен изглед (скриване на информация)
    - процесен изглед (производителност)
    - Изглед на потока от данни (функционални изисквания)
4. Оценка на атрибутите за качество (надеждност, сигурност,….)
5. Определяне чувствителността на атрибутите за качество към различни архитектурни атрибути за различни архитектурни стилове.
6. Критики към архитектурите кандидати

Agile архитектура

- В класическите agile подходи имаме еволюционно проектиране – не е подходящо за големи проекти с много функционалности;
- Предложен от Madison хибриден шаблон – имаме дейности свързани с архитектурата, но и бързо движение по user stories.
  - Създаване на архитектурни user stories;
  - Определяне на приоритета заедно с product owner
  - Планиране на work units за спринтовете
- Преглед на архитектурата в края на спринта