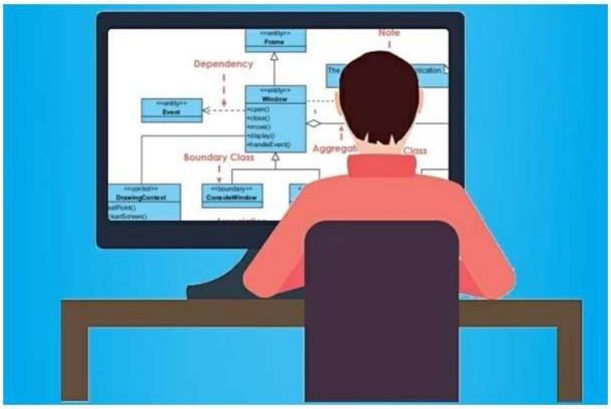
Software Design & Architecture
Spring 2022 - Week-13



مدرس: مہندس ماجد کلیم

جامعہ بحریہ، واقعگاہ کراچی

Engr. Majid Kaleem

WEEKLY AGENDA

TENTATIVE WEEKLY DATES		TENTATIVE TOPICS
1	Mar 7th – Mar 11th	INTRODUCTION TO THE COURSE; DEFINING SOFTWARE ARCHITECTURE & DESIGN CONCEPTS
2	Mar 14th – Mar 18th	DESIGN PRINCIPLES; OBJECT-ORIENTED DESIGN WITH UML
3	Mar 21st – Mar 25th	SYSTEM DESIGN & SOFTWARE ARCHITECTURE; OBJECT DESIGN, MAPPING DESIGN TO CODE
4	Mar 28th – Apr 1st	FUNCTIONAL DESIGN; UI DESIGN; WEB APPLICATIONS DESIGN **ASSIGNMENT & QUIZ #1**
5	Apr 4th – Apr 8th	MOBILE APPLICATION DESIGN; PERSISTENCE LAYER DESIGN
6	Apr 11th – Apr 15th	CREATIONAL DESIGN PATTERNS
7	Apr 18th – Apr 22nd	STRUCTURAL DESIGN PATTERNS **ASSIGNMENT & QUIZ #2**
8	Apr 25th – Apr 29th	BEHAVIORAL DESIGN PATTERNS
← MID TERM EXAMINATIONS →		
9	May 9th – May 13th	INTERACTIVE SYSTEMS WITH MVC ARCHITECTURE; SOFTWARE REUSE
10	May 16th – May 20th	ARCHITECTURAL DESIGN ISSUES; ARCHITECTURE DESCRIPTION LANGUAGES (ADLS)
11	May 23rd – May 27th	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES
12	May 30th – Jun 3rd	ARCHITECTURAL STYLES/PATTERNS & DESIGN QUALITIES **ASSIGNMENT & QUIZ #3**
13	Jun 6th – Jun 10th	QUALITY TACTICS; ARCHITECTURE DOCUMENTATION
14	Jun 13th – Jun 17th	ARCHITECTURAL EVALUATION TECHNIQUES
15	Jun 20th – Jun 24th	MODEL DRIVEN DEVELOPMENT **ASSIGNMENT (PRESENTATIONS) & QUIZ #4**
16	Jun 27th – Jul 1st	REVISION WEEK
← FINAL TERM EXAMINATIONS →		

QUALITY TACTICS

- The quality attribute requirements specify the responses of the system that realize the goals of the business.
- An architect can use techniques to *achieve* the required quality attributes.
- *These techniques are called architectural tactics.*
- A tactic is a design decision that influences the achievement of a quality attribute response—*tactics directly affect the system's response to some stimulus.*
- Tactics impart portability to one design, high performance to another, and integrability *(quality of being integrated)* to a third.

3

QUALITY TACTICS

- In general, a tactic is a conceptual action aiming at the achievement of a goal.
- An architectural tactic is a design decision that influences the achievement of a quality attribute response, as a design goal.
  - Building blocks of architectural styles and patterns
  - No consideration of tradeoffs

4

QUALITY TACTICS

- Different tactics for each quality attribute
- The same tactic could be relevant to many quality attributes

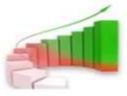
Security	Modifiability	Availability
Tactic 1	*Tactic 5*	*Tactic 3*
Tactic 2	*Tactic 6*	*Tactic 8*
Tactic 3	*Tactic 7*	*Tactic 9*
Tactic 4		*Tactic 10*
		Tactic 11
		Tactic 12

5

QUALITY TACTICS

Example Quality Attributes

Availability


Performance


Interoperability

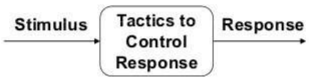

Modifiability


Security

6

QUALITY TACTICS

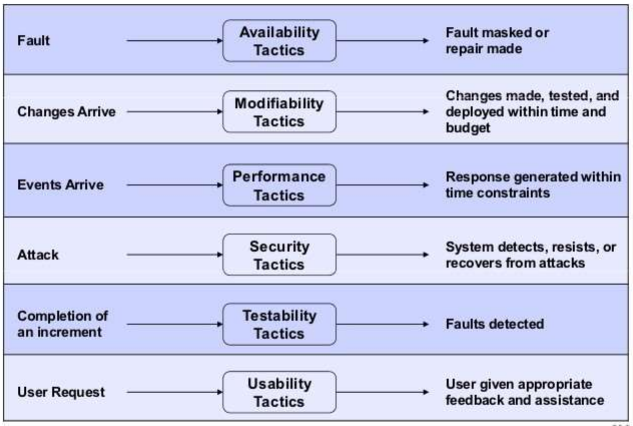
- *Quality attribute decisions are also known as tactics.*
- Relationship is shown below:



7

QUALITY TACTICS

- Tactics are decisions to achieve quality attribute requirements



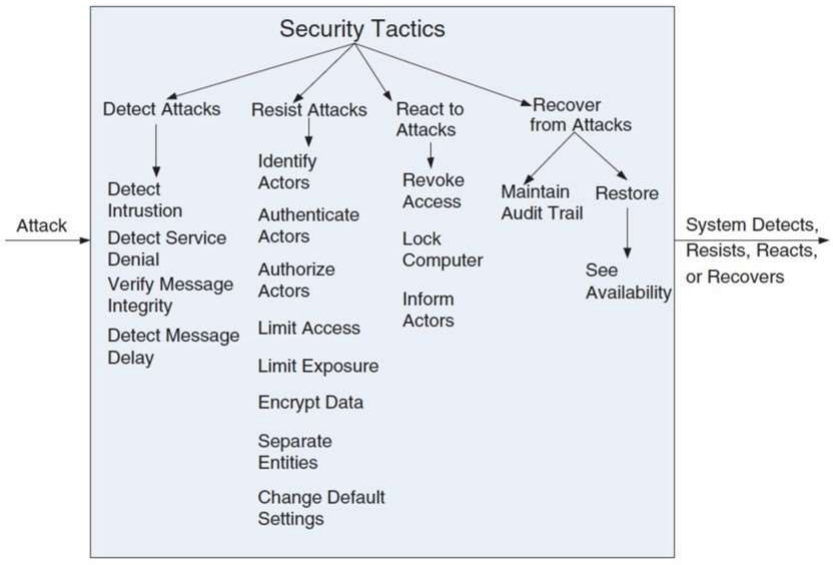
8

QUALITY TACTICS

- Each tactic is a design option for the architect.
- *For example, one of the tactics introduces redundancy to increase the availability of a system*.
- Usually achieving high availability through redundancy implies a associated need for synchronization (to ensure that the redundant copy can be used if the original fails).

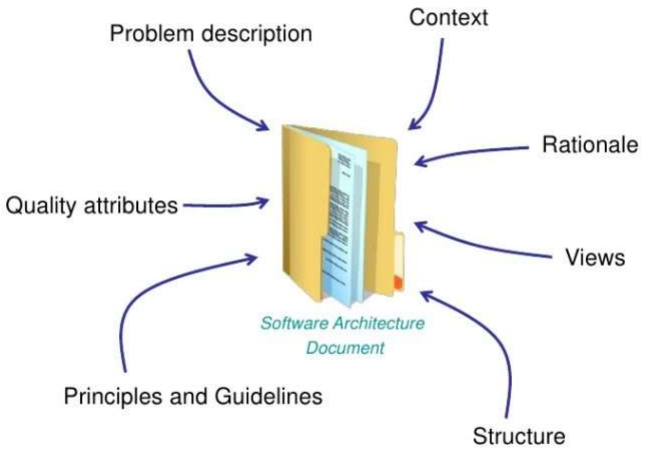
9

EXAMPLE – SECURITY TACTICS



10

ARCHITECTURAL DOCUMENTATION



11

ARCHITECTURAL DOCUMENTATION

- *"Documentation is to write from the point of view of the reader"*
- The architecture of the system depends on the requirement levied on it.
- Should be sufficiently abstract.
- Should be detailed also to serve as blueprint.
- Understand who the stakeholder are, this will help in documentation.

12

ARCHITECTURAL DOCUMENTATION - VIEWS

- A software architecture for a system is *"the structure or the structures of the system , which comprise elements , the externally visible properties of those elements , and relationship among them"*
- **View** – representation of a coherent set of architectural elements as written and read by stakeholders.

13

ARCHITECTURAL DOCUMENTATION - VIEWS

- Documenting an architecture is a matter of documenting the relevant view and then adding documentation that applies to more than one view.
- Backbone of the architecture documentation:-
  - Choosing the relevant view.
  - Documenting the view.
  - Documenting information that applies to more than one view.

14

ARCHITECTURAL DOCUMENTATION - VIEWS

- Documenting an architecture is a matter of documenting the relevant view and then adding documentation that applies to more than one view.
- Backbone of the architecture documentation:-
  - Choosing the relevant view.
  - Documenting the view.
  - Documenting information that applies to more than one view.

14

ARCHITECTURAL VIEWS – PHILLIP KRUCHTEN (4+1) VIEW

- **LOGICAL VIEW**: Shows the parts that comprise the system, as well as their interactions.
- Represents set abstractions.
- Emphasizes classes and objects.
- Logical view describes system's *object model*.
- UML diagrams that show the logical view include:
  - *Class diagrams*
  - *State diagrams*
  - *Object diagrams*
  - *Communications diagrams*

16

ARCHITECTURAL VIEWS – PHILLIP KRUCHTEN (4+1) VIEW

- **PROCESS VIEW**: Describes system's processes.
- Shows any communications between those processes.
- Explores what needs to happen inside the system.
- Particularly helpful when your system will have a number of simultaneous threads or processes.
- UML *activity* diagram represents the process view.
- **PHYSICAL VIEW**: Models the system's execution environment.
- Maps software artifacts onto the hardware that hosts them.
- UML *deployment* diagrams are used to model the physical view of a system.

17

ARCHITECTURAL VIEWS – PHILLIP KRUCHTEN (4+1) VIEW

- **DEPLOYMENT VIEW**: Describes the system's modules, or components, including packages, sub-systems and class libraries.
- Gives a building-block view of the system.
- Helpful for managing the system's layers.
- UML diagrams that show the development view include:
  - Component diagrams
  - Package diagrams

18

ARCHITECTURAL VIEWS – PHILLIP KRUCHTEN (4+1) VIEW

- **USE CASE VIEW**: Shows the system's functionality.
- Captures user goals and scenarios.
- Helpful in defining and explaining the structures and functionality in the other four views.
- UML use case diagrams (along with written use cases and requirement specifications) provide the use case view.

19

ARCHITECTURE DOCUMENTATION CHALLENGES

- How do you decide which architectural views to document?
- What information do you record about an architectural view beyond the box-and-line diagram?
- How do you specify an architectural element's software interface? What information do you record?
- How do you specify an element's behavior?
- What notations are available for documenting a view, an interface or behavior?

20

IEEE – SOFTWARE DESIGN DESCRIPTION (SDD)

- The SDD usually contains the following information:
- The *data design* describes structures that reside within the software. Attributes and relationships between *data objects* dictate the choice of *data structures.*
- The *architecture design* uses information flowing characteristics, and maps them into the program structure. The transformation mapping method is applied to exhibit distinct boundaries between incoming and outgoing data. The data flow diagrams allocate control input, processing and output along three separate modules.
- The *interface design* describes internal and external program interfaces, as well as the design of the *human interface*. Internal and external interface designs are based on the information obtained from the analysis model.

21

IEEE – SOFTWARE DESIGN DESCRIPTION (SDD)

- The *procedural design* describes structured programming concepts using graphical, tabular and textual notations.
- These design mediums enable the designer to represent procedural detail, that facilitates translation to code. This blueprint for implementation forms the basis for all subsequent software engineering work.
- **IEEE 1016-2009** Sample SDD Template (*attached separately*)

22

```
If(anyQuestions)
{
  askNow();
}
else
{
  thankYou();
  submitAttendance();
  endClass();
}
```


REFERENCES

1. ***Software Architecture***, *Perspectives on an Emerging Discipline By Mary Shaw & David Garlan*
2. ***The Art of Software Architecture***, *Design Methods & Techniques By Stephen T. Albin*
3. ***Essential Software Architecture***, *By Ian Gorton*
4. ***Microsoft Application Architecture Guide***, *By Microsoft*
5. ***Design Patterns***, *Elements of Reusable Object-Oriented Software By by Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides*
6. ***Refactoring, Improving the Design of Existing Code***, *By Martin Fowler & Kent Beck*