Introduction to CITS4401

Software Requirements and Design

CITS4401

Lecture 1

Outline of this Lecture

What you’ll be learning in CITS4401

Software Engineering

Requirements

Design

Software Methodologies

Software Processes

How you’ll be learning in CITS4401

Introductions

- Unit Coordinator: **Dr Mehwish Nasim**
- CITS4401 consultation : **Mondays 3:15pm in CSSE Rm 1.16 (after the lecture)**
- My research: Network Modeling, Serious games, Information Visualisation
- Office: **CSSE Room: 1.16**
- Email: cits4401-pmc@uwa.edu.au

What you’ll be learning in CITS4401

CITS4401 unit overview

Requirements and design are important phases of software development because errors or misunderstandings of software requirements or designs are expensive to correct during later stages and may lead to project failure.

This unit introduces the theory and practice of software requirements and design.

The content comprises

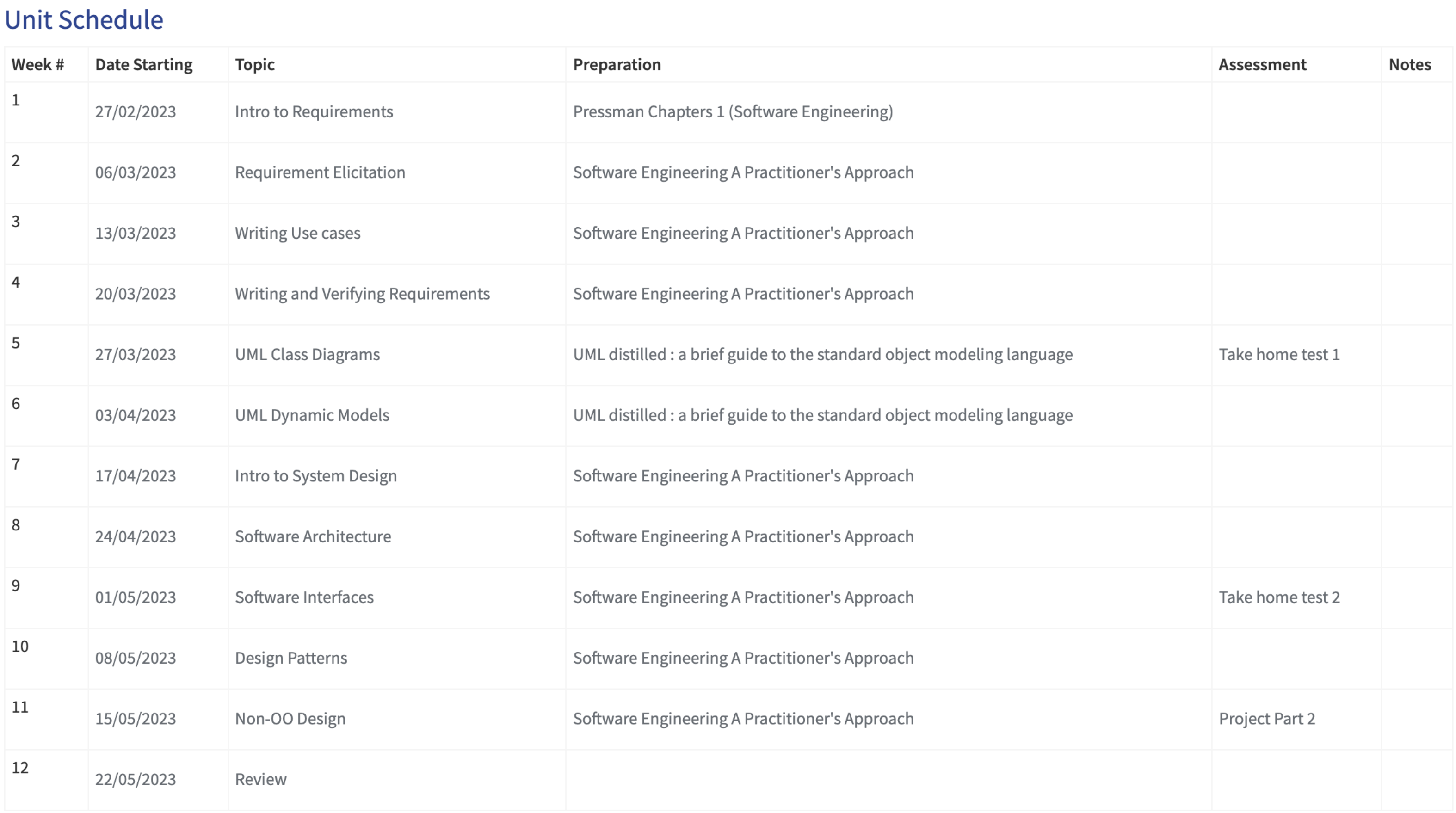
(1) requirements engineering

(2) software design

(3) software architectures; and

(4) design patterns and idioms.

CITS4401 Schedule (see LMS)



Unit Schedule

Week #	Date Starting	Topic	Preparation	Assessment	Notes
1	27/02/2023	Intro to Requirements	Pressman Chapters 1 (Software Engineering)		
2	06/03/2023	Requirement Elicitation	Software Engineering A Practitioner's Approach		
3	13/03/2023	Writing Use cases	Software Engineering A Practitioner's Approach		
4	20/03/2023	Writing and Verifying Requirements	Software Engineering A Practitioner's Approach		
5	27/03/2023	UML Class Diagrams	UML distilled : a brief guide to the standard object modeling language	Take home test 1	
6	03/04/2023	UML Dynamic Models	UML distilled : a brief guide to the standard object modeling language		
7	17/04/2023	Intro to System Design	Software Engineering A Practitioner's Approach		
8	24/04/2023	Software Architecture	Software Engineering A Practitioner's Approach		
9	01/05/2023	Software Interfaces	Software Engineering A Practitioner's Approach	Take home test 2	
10	08/05/2023	Design Patterns	Software Engineering A Practitioner's Approach		
11	15/05/2023	Non-OO Design	Software Engineering A Practitioner's Approach	Project Part 2	
12	22/05/2023	Review			

Software Engineering

- Two words: *Software* and *Engineering*
  - **Software** is a set of instructions, data or programs used to operate computers and execute specific tasks
  - **Engineering** applies systematic, scientific and well-defined processes to produce a good quality product.

- SE is a *creative* process in which
  - there are few *right/wrong* answers
  - but nonetheless some requirements and designs are (much) better than others.
- **Choices must be evaluated and justified.**

Software Engineering Definition

software engineering 3.3810 [ISO standard]

1. systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software

ISO/IEC 2382:2015, Information technology — Vocabulary

2. application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software

ISO/IEC TR 19759:2016, Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK)

Requirements

software requirement 3.3847

1. **software capability** **needed** by a user to solve a problem or to achieve an objective

2. software capability **that must be met or possessed** by a system or system component to satisfy a contract, standard, specification, or other formally imposed document

software requirements analysis 3.3848

1. **process** of studying user needs to arrive at a definition of system, hardware, or software requirements

Design

design 3.1125

1. [**process**] to define the **architecture, system elements, interfaces**, and other characteristics of a system or system element

2. **result** of the process in (1)

software requirements analysis 3.3848
1. **process** of studying user needs to arrive at a definition of system, hardware, or software requirements

Software Methodologies

methodology 3.2438

1. a system of practices, techniques, procedures, and rules used by those who work in a discipline

[A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition]

2. specification of the process to follow together with the work products to be used and generated, plus the consideration of the people and tools involved, during an information-based domain development effort

[ISO/IEC 24744:2014 Software Engineering — Metamodel for development methodologies, 3.2]

A series of related methods (ie systematic procedures) or techniques

[Websters Dictionary quoted by Cockburn]

Software Methodologies

"All organisations have a methodology – it is simply how they do business."

"Your methodology is everything you regularly do to get your software out … the conventions your group agrees to."

[Cockburn: Agile SW Development]

Methodology Concepts [Cockburn]

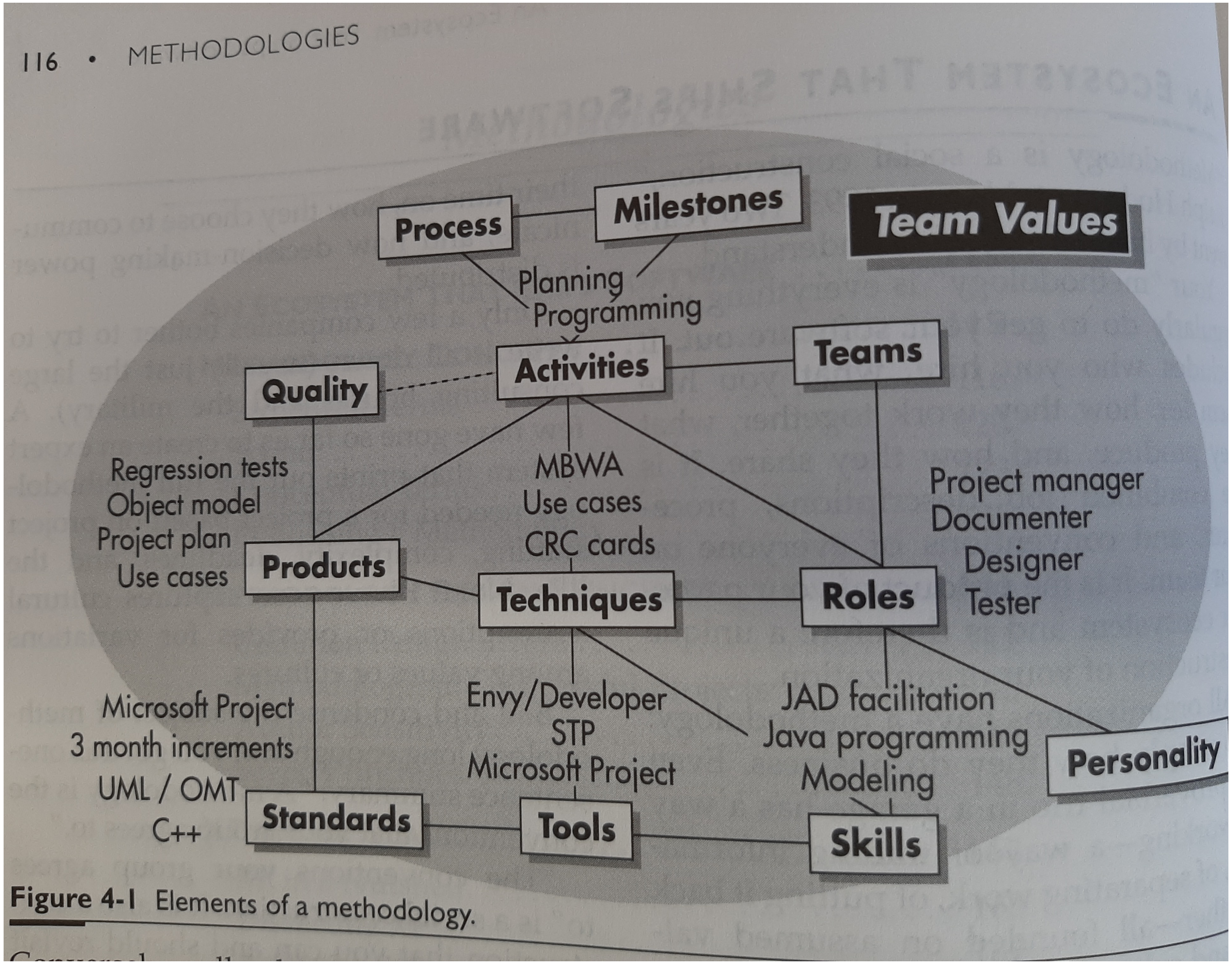


Figure 4-1 Elements of a methodology.

Software Processes

- A **software process** is a set of interrelated activities and tasks that **transform input work products** into **output work products**.
  [SWE page 8-2]

- **Activities** are how the people spend their days.
  e.g. planning, programming, testing, meeting. *[Cockburn]*

- A **process** is how activities fit together time, often with pre- and post-conditions for the activities.
  e.g. design review happens 2 days after designs are sent to participants and produces a list of recommendations
  [Cockburn]

Why bother with requirements and design?

Why Bother….because:

a. ***EVERY*** project has ***uncertainty***….Requirements are one way of dealing with it!

b. Engineers need ***something concrete to work toward*** (many engineers fear uncertainty).

c. The requirement that rules them all (and is implied by the Trade Practices Act). “Is the solution fit-for-purpose?”

d. A completed test program that ***demonstrates requirements are met*** is an effective tool to ***help get paid***!

How you'll be learning in CITS4401/3301

CITS4401 Learning Outcomes

1. **Classify** types of software requirements and designs
2. **Apply** requirements and design processes appropriate for a given scenario
3. **Assess** quality attributes of given requirements and designs
4. **Utilise** design patterns and idioms
5. **Document** software design rationale using discourse conventions of the discipline
6. **Select** a software architecture appropriate for a particular context

CITS4401 Lectures Mon@2pm

- Lectures will present an *overview* of *problems, theory,* and *techniques* for selected topics in SE, with a specific focus on requirements and design
- Lectures will NOT be live-streamed (ms-teams)
- Recorded lectures will be available (Echo via LMS)
- Some pre-recorded mini-lectures will be also provided in LMS
- Text books

  Pressman, Software Engineering
  Fowler, UML distilled
- The texts and other recommended reading is available from **UWA unit readings** via LMS. Online versions are available so please use them.

CITS4401 Workshops Wed@2pm

- Workshop classes are practical sessions for students applying SE requirements and design techniques
- Workshops start from Week 1, Wed 2pm
- Class work usually in groups
- Guest presenters from industry
- Submit your workshop answers in LMS (by Thu 4PM)
- Two take home tests in week 5 and 9 contribute 20% to your final mark

CITS4401 Assessment

Take home tests (20%)

Two **individual** take home test based on lectures and workshops

Due weeks 5 and 9

Submit in LMS

Requirements and Design Group Project (30%)

Group project with 2 deliverables

Due weeks 7 and 11

Teams of 5 students assigned by the unit coordinator

Submit in LMS

Final home exam (50%)

Individual closed book exam based on lectures and workshops

During the standard Exam week at UWA

The small print: See the unit outline for academic conduct rules, late penalties, covid contingencies etc

Summary of this Lecture

What you’ll be learning in CITS4401

Software Engineering

Requirements

Design

Software Methodologies

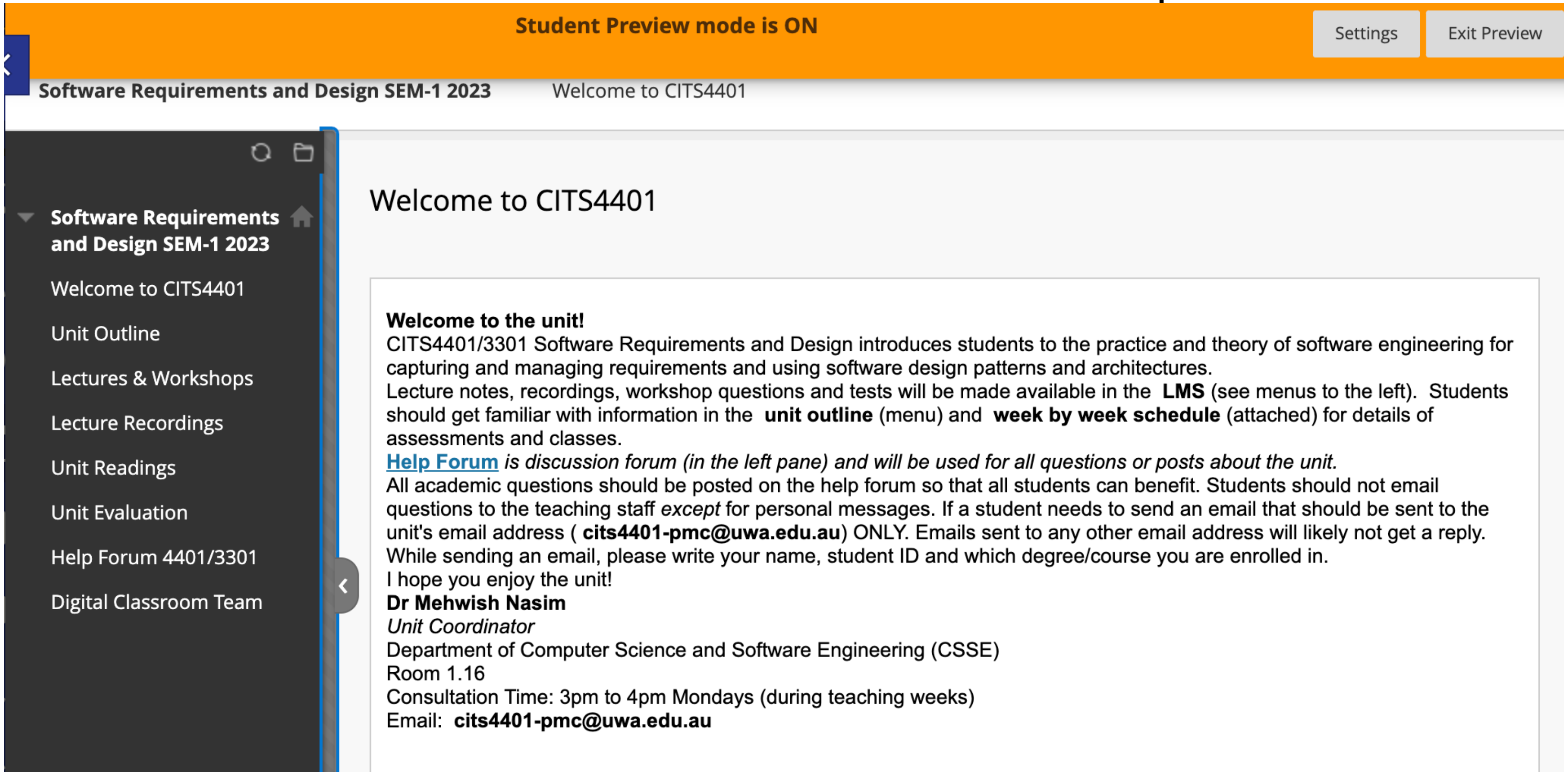
Software Processes

How you’ll be learning in CITS4401

Learning Resources
Get to know these now!

Learning resources 1: LMS

https://lms.uwa.edu.au/

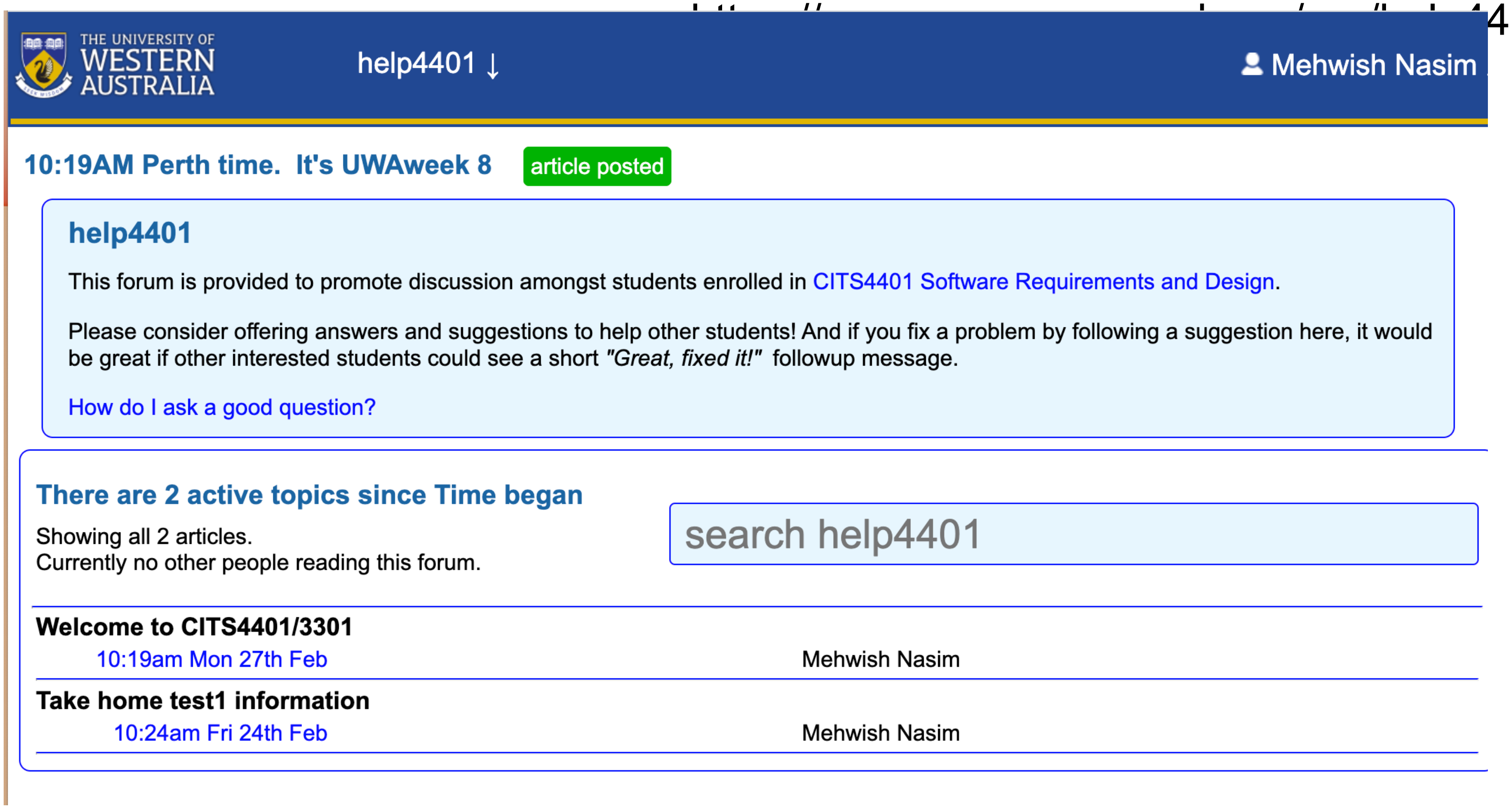


Welcome to CITS4401

Welcome to the unit!
CITS4401/3301 Software Requirements and Design introduces students to the practice and theory of software engineering for capturing and managing requirements and using software design patterns and architectures.
Lecture notes, recordings, workshop questions and tests will be made available in the **LMS** (see menus to the left). Students should get familiar with information in the **unit outline** (menu) and **week by week schedule** (attached) for details of assessments and classes.
Help Forum *is discussion forum (in the left pane) and will be used for all questions or posts about the unit.*
All academic questions should be posted on the help forum so that all students can benefit. Students should not email questions to the teaching staff *except* for personal messages. If a student needs to send an email that should be sent to the unit's email address (**cits4401-pmc@uwa.edu.au**) ONLY. Emails sent to any other email address will likely not get a reply.
While sending an email, please write your name, student ID and which degree/course you are enrolled in.
I hope you enjoy the unit!
Dr Mehwish Nasim
Unit Coordinator
Department of Computer Science and Software Engineering (CSSE)
Room 1.16
Consultation Time: 3pm to 4pm Mondays (during teaching weeks)
Email: **cits4401-pmc@uwa.edu.au**

Learning resources 2: help4401

401



10:19AM Perth time. It's UWAweek 8 article posted

help4401

This forum is provided to promote discussion amongst students enrolled in CITS4401 Software Requirements and Design.

Please consider offering answers and suggestions to help other students! And if you fix a problem by following a suggestion here, it would be great if other interested students could see a short *"Great, fixed it!"* followup message.

How do I ask a good question?

There are 2 active topics since Time began

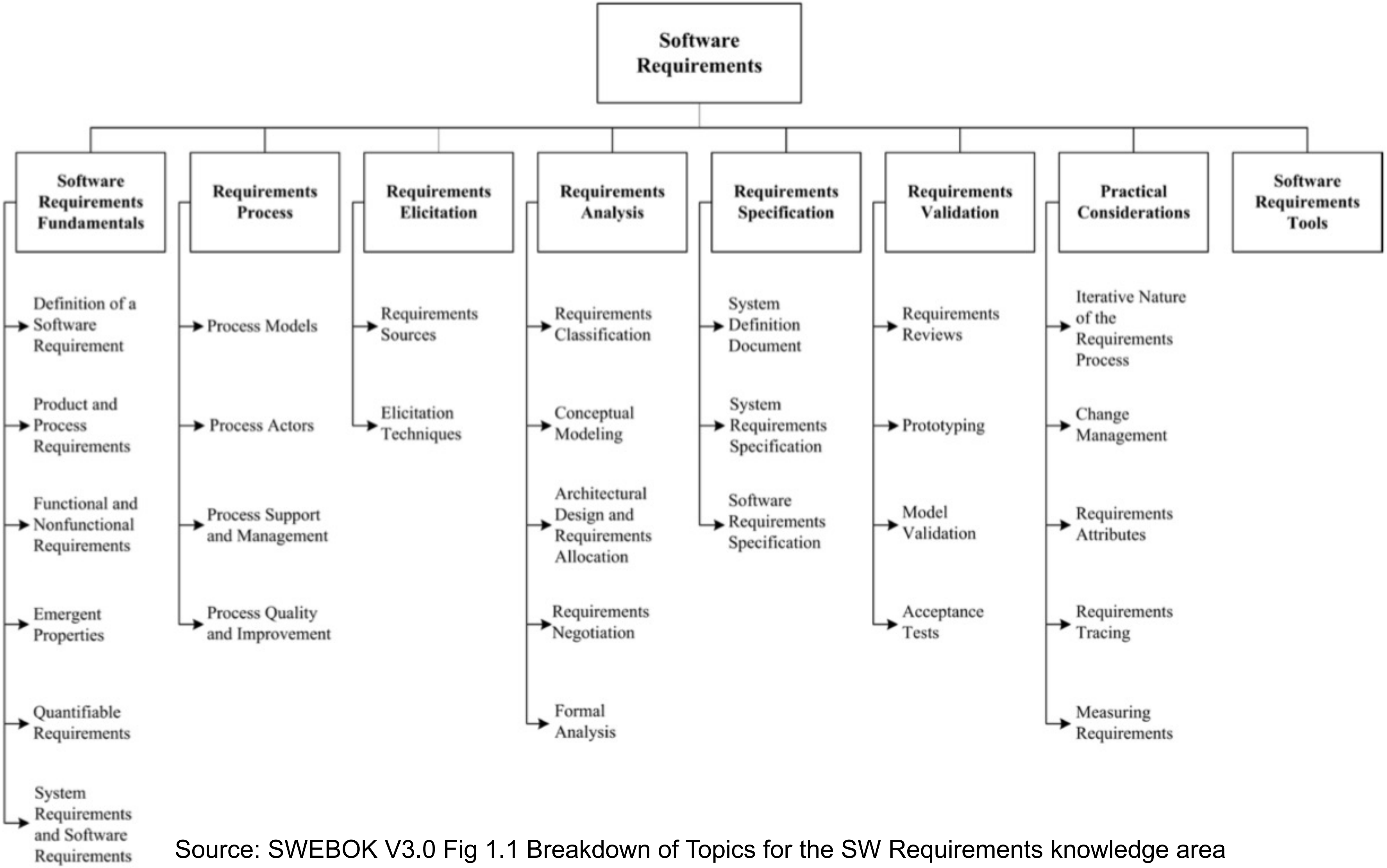
Showing all 2 articles.
Currently no other people reading this forum.

search help4401

Welcome to CITS4401/3301	
10:19am Mon 27th Feb	Mehwish Nasim
Take home test1 information	
10:24am Fri 24th Feb	Mehwish Nasim

Software Engineering Body of Knowledge

SW Requirements Knowledge



Software Requirements

Software Requirements Fundamentals
- Definition of a Software Requirement
- Product and Process Requirements
- Functional and Nonfunctional Requirements
- Emergent Properties
- Quantifiable Requirements
- System Requirements and Software Requirements

Requirements Process
- Process Models
- Process Actors
- Process Support and Management
- Process Quality and Improvement

Requirements Elicitation
- Requirements Sources
- Elicitation Techniques

Requirements Analysis
- Requirements Classification
- Conceptual Modeling
- Architectural Design and Requirements Allocation
- Requirements Negotiation
- Formal Analysis

Requirements Specification
- System Definition Document
- System Requirements Specification
- Software Requirements Specification

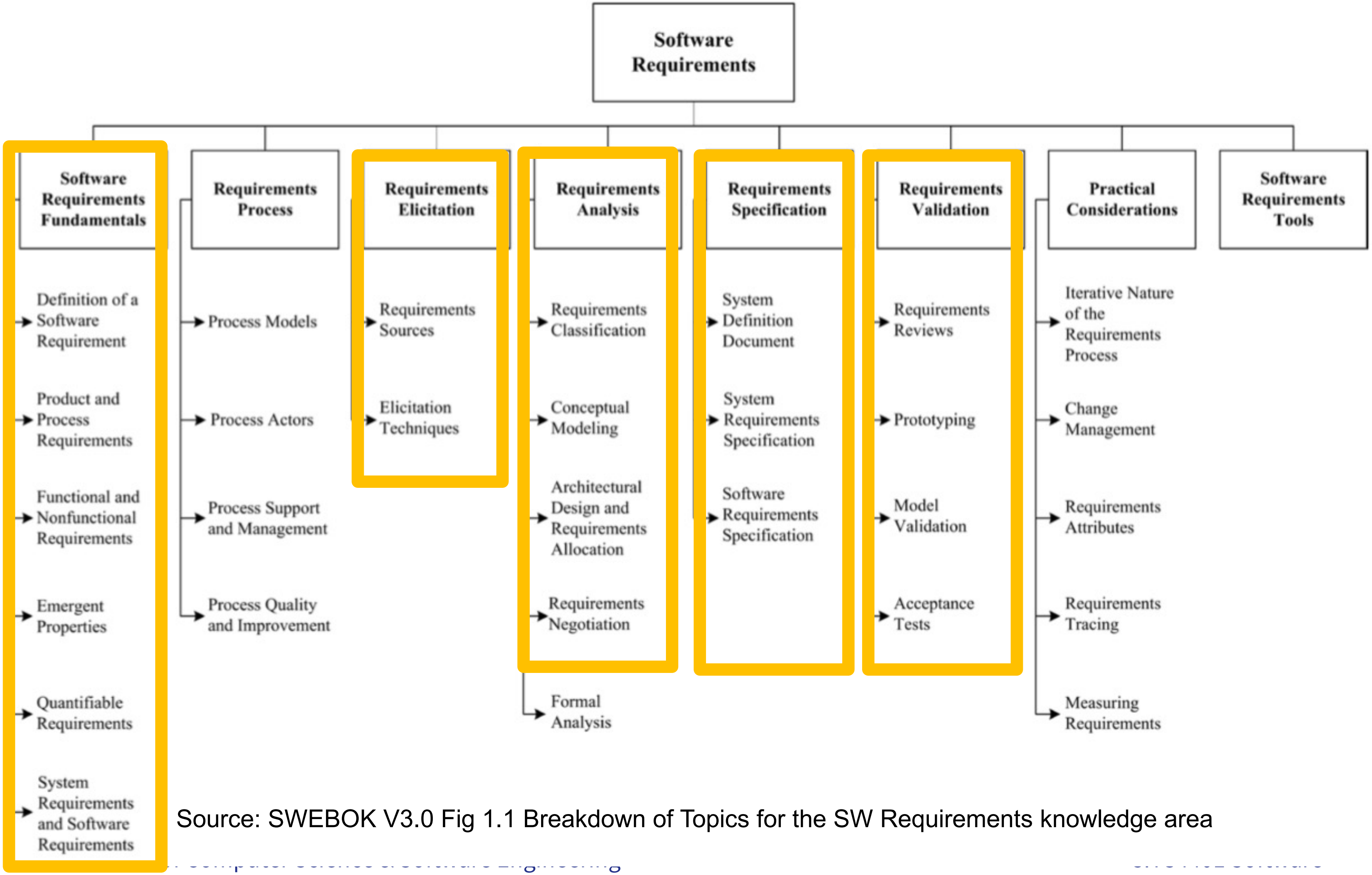
Requirements Validation
- Requirements Reviews
- Prototyping
- Model Validation
- Acceptance Tests

Practical Considerations
- Iterative Nature of the Requirements Process
- Change Management
- Requirements Attributes
- Requirements Tracing
- Measuring Requirements

Software Requirements Tools

Source: SWEBOK V3.0 Fig 1.1 Breakdown of Topics for the SW Requirements knowledge area

CITS4401 Requirements

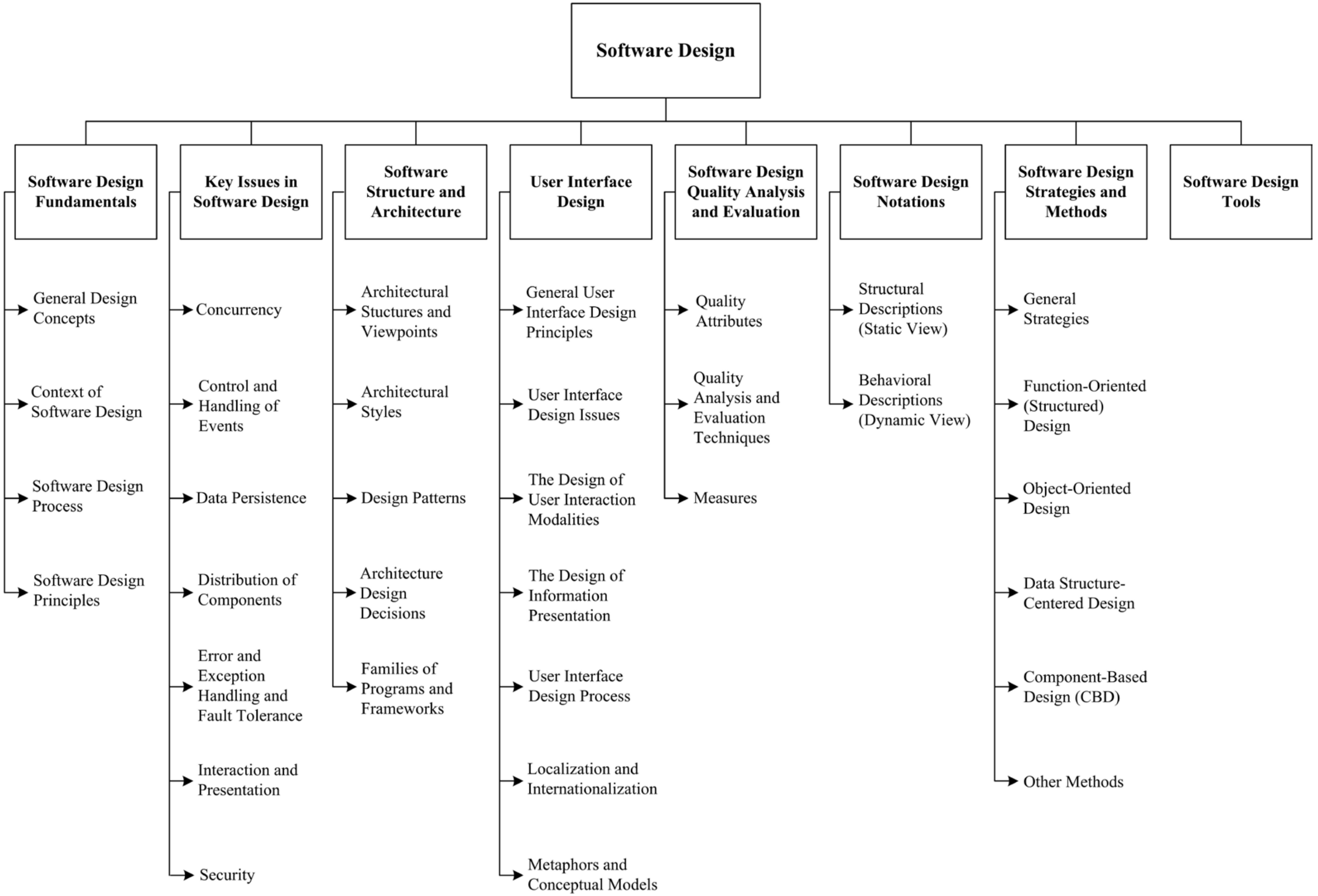


Software Requirements

- Software Requirements Fundamentals
  - Definition of a Software Requirement
  - Product and Process Requirements
  - Functional and Nonfunctional Requirements
  - Emergent Properties
  - Quantifiable Requirements
  - System Requirements and Software Requirements
- Requirements Process
  - Process Models
  - Process Actors
  - Process Support and Management
  - Process Quality and Improvement
- Requirements Elicitation
  - Requirements Sources
  - Elicitation Techniques
- Requirements Analysis
  - Requirements Classification
  - Conceptual Modeling
  - Architectural Design and Requirements Allocation
  - Requirements Negotiation
  - Formal Analysis
- Requirements Specification
  - System Definition Document
  - System Requirements Specification
  - Software Requirements Specification
- Requirements Validation
  - Requirements Reviews
  - Prototyping
  - Model Validation
  - Acceptance Tests
- Practical Considerations
  - Iterative Nature of the Requirements Process
  - Change Management
  - Requirements Attributes
  - Requirements Tracing
  - Measuring Requirements
- Software Requirements Tools

Source: SWEBOK V3.0 Fig 1.1 Breakdown of Topics for the SW Requirements knowledge area

Software Design Knowledge



Software Design

Software Design Fundamentals
- General Design Concepts
- Context of Software Design
- Software Design Process
- Software Design Principles

Key Issues in Software Design
- Concurrency
- Control and Handling of Events
- Data Persistence
- Distribution of Components
- Error and Exception Handling and Fault Tolerance
- Interaction and Presentation
- Security

Software Structure and Architecture
- Architectural Stuctures and Viewpoints
- Architectural Styles
- Design Patterns
- Architecture Design Decisions
- Families of Programs and Frameworks

User Interface Design
- General User Interface Design Principles
- User Interface Design Issues
- The Design of User Interaction Modalities
- The Design of Information Presentation
- User Interface Design Process
- Localization and Internationalization
- Metaphors and Conceptual Models

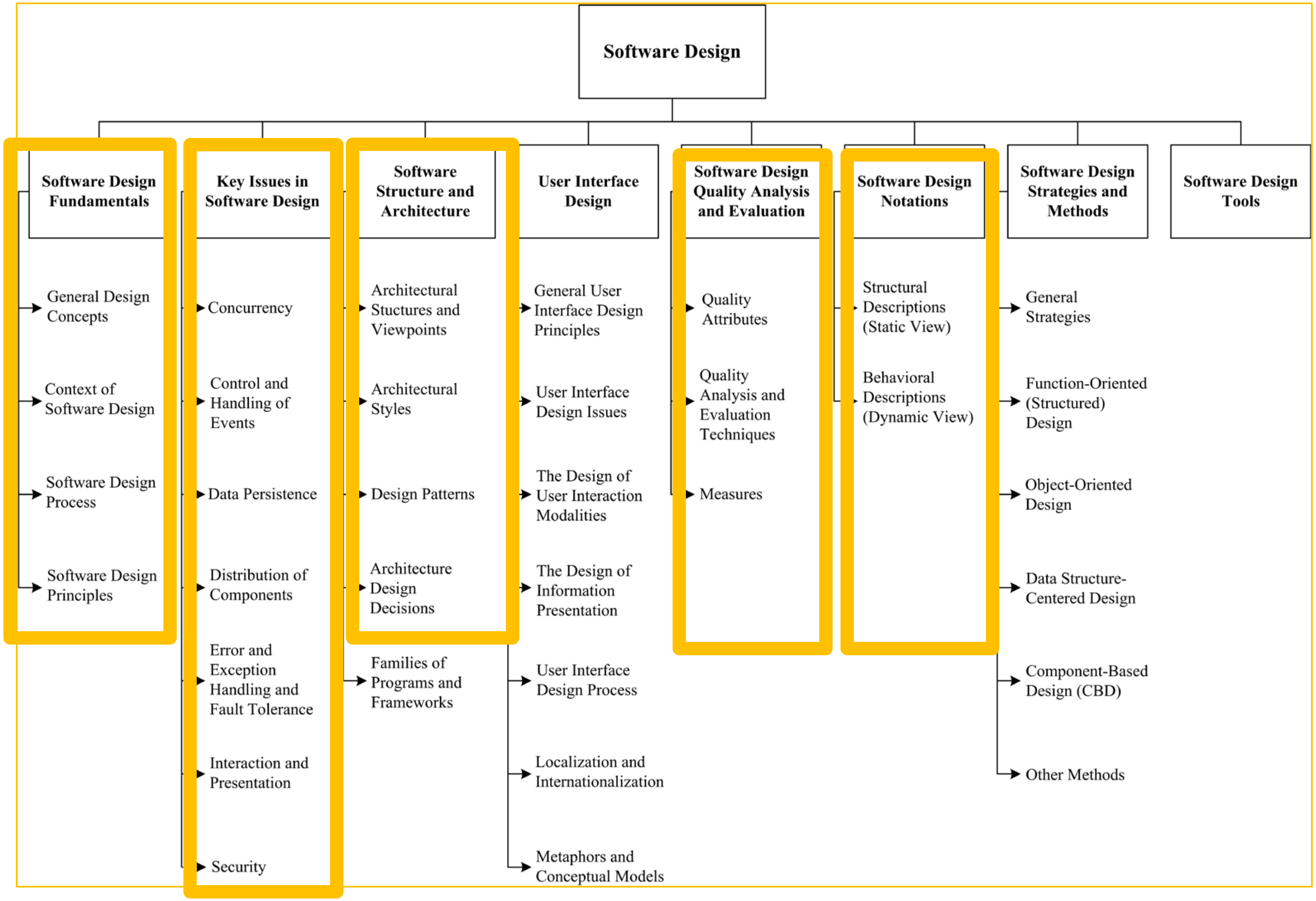
Software Design Quality Analysis and Evaluation
- Quality Attributes
- Quality Analysis and Evaluation Techniques
- Measures

Software Design Notations
- Structural Descriptions (Static View)
- Behavioral Descriptions (Dynamic View)

Software Design Strategies and Methods
- General Strategies
- Function-Oriented (Structured) Design
- Object-Oriented Design
- Data Structure-Centered Design
- Component-Based Design (CBD)
- Other Methods

Software Design Tools

CITS4401 (Design)



Software Design
Software Design Fundamentals
General Design Concepts
Context of Software Design
Software Design Process
Software Design Principles
Key Issues in Software Design
Concurrency
Control and Handling of Events
Data Persistence
Distribution of Components
Error and Exception Handling and Fault Tolerance
Interaction and Presentation
Security
Software Structure and Architecture
Architectural Stuctures and Viewpoints
Architectural Styles
Design Patterns
Architecture Design Decisions
Families of Programs and Frameworks
User Interface Design
General User Interface Design Principles
User Interface Design Issues
The Design of User Interaction Modalities
The Design of Information Presentation
User Interface Design Process
Localization and Internationalization
Metaphors and Conceptual Models
Software Design Quality Analysis and Evaluation
Quality Attributes
Quality Analysis and Evaluation Techniques
Measures
Software Design Notations
Structural Descriptions (Static View)
Behavioral Descriptions (Dynamic View)
Software Design Strategies and Methods
General Strategies
Function-Oriented (Structured) Design
Object-Oriented Design
Data Structure-Centered Design
Component-Based Design (CBD)
Other Methods
Software Design Tools

Learning SW engineering

In this unit you will be learning a number of new software engineering methods and techniques. All have strengths and weaknesses.

In Agile SW Development Cockburn discusses 3 levels of understanding new methods and skills

1. **Following:** you know a detailed procedure that works and can follow it exactly
2. **Detaching:** locate the limits of your single procedures; look for rules that explain when it works well and when it does not; adapt your use of the procedure as needed
3. **Fluent:** understand the desired end effect and work towards it; understand trade-offs and selection

In this unit you will mostly be working at the following level with some detaching. Fluency takes years of experience. But keep it in mind as the long term goal.