U O W

Software Requirements, Specifications and Formal Methods

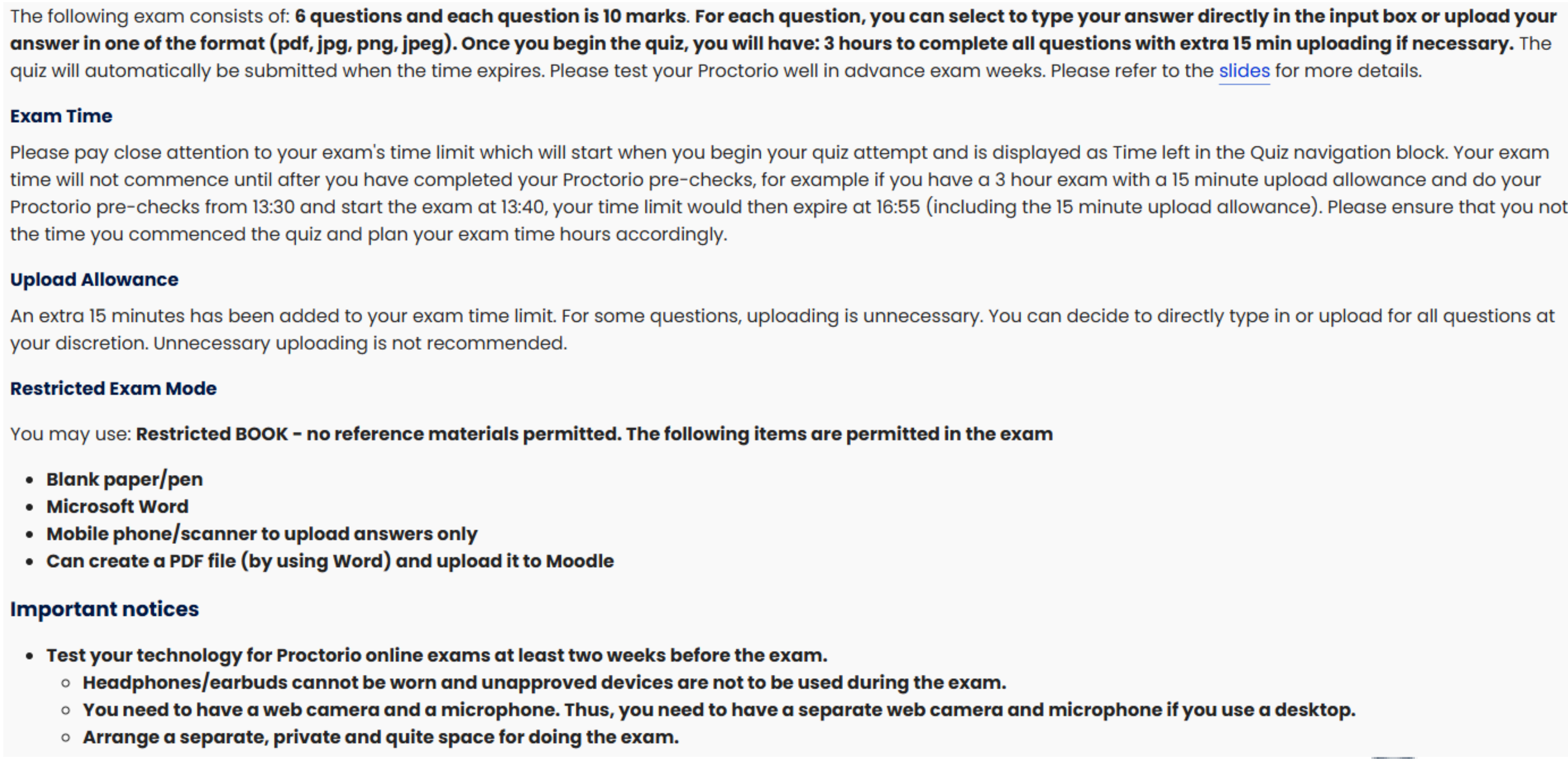
Dr. Shixun Huang

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Exam

Time: 13:30 pm, 16th, June Coverage: the whole subject

Answering mode: answer your questions in the Moodle site, no downloading is required.



The following exam consists of: **6 questions and each question is 10 marks. For each question, you can select to type your answer directly in the input box or upload your answer in one of the format (pdf, jpg, png, jpeg). Once you begin the quiz, you will have: 3 hours to complete all questions with extra 15 min uploading if necessary.** The quiz will automatically be submitted when the time expires. Please test your Proctorio well in advance exam weeks. Please refer to the slides for more details.

Exam Time

Please pay close attention to your exam's time limit which will start when you begin your quiz attempt and is displayed as Time left in the Quiz navigation block. Your exam time will not commence until after you have completed your Proctorio pre-checks, for example if you have a 3 hour exam with a 15 minute upload allowance and do your Proctorio pre-checks from 13:30 and start the exam at 13:40, your time limit would then expire at 16:55 (including the 15 minute upload allowance). Please ensure that you not the time you commenced the quiz and plan your exam time hours accordingly.

Upload Allowance

An extra 15 minutes has been added to your exam time limit. For some questions, uploading is unnecessary. You can decide to directly type in or upload for all questions at your discretion. Unnecessary uploading is not recommended.

Restricted Exam Mode

You may use: **Restricted BOOK - no reference materials permitted. The following items are permitted in the exam**

- **Blank paper/pen**
- **Microsoft Word**
- **Mobile phone/scanner to upload answers only**
- **Can create a PDF file (by using Word) and upload it to Moodle**

Important notices

- **Test your technology for Proctorio online exams at least two weeks before the exam.**
  - **Headphones/earbuds cannot be worn and unapproved devices are not to be used during the exam.**
  - **You need to have a web camera and a microphone. Thus, you need to have a separate web camera and microphone if you use a desktop.**
  - **Arrange a separate, private and quite space for doing the exam.**

Revision

Some Major Topics:

- Software process models
- Requirement process and elicitation
- Requirement specification
- DFD, FSM
- Petri Net, Colored Petri Net
- Formal Specification Method: Z

Software process models

- The waterfall model
  – Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  – Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
  – The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

Types of requirement

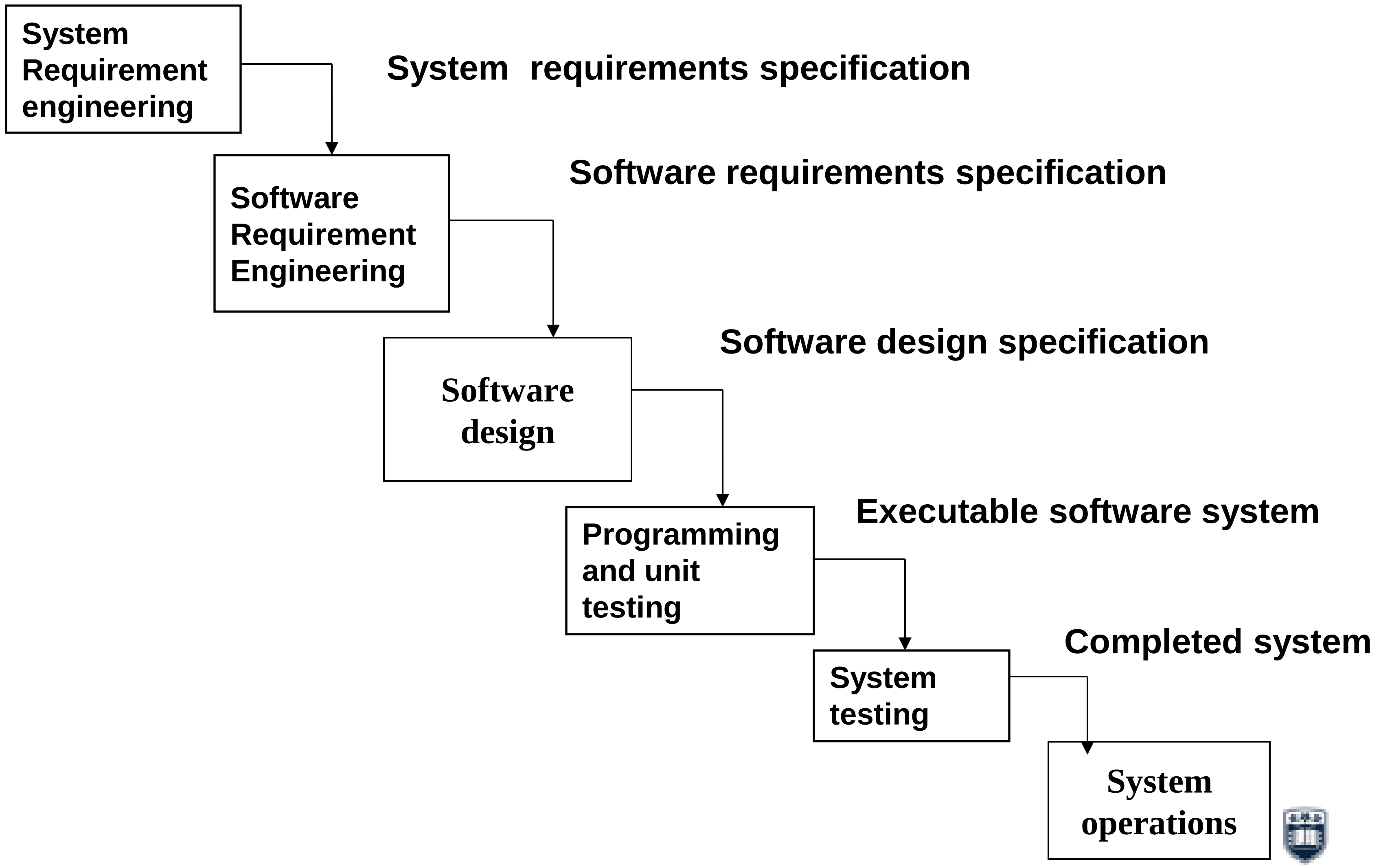
- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
  - A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Functional and non-functional requirements

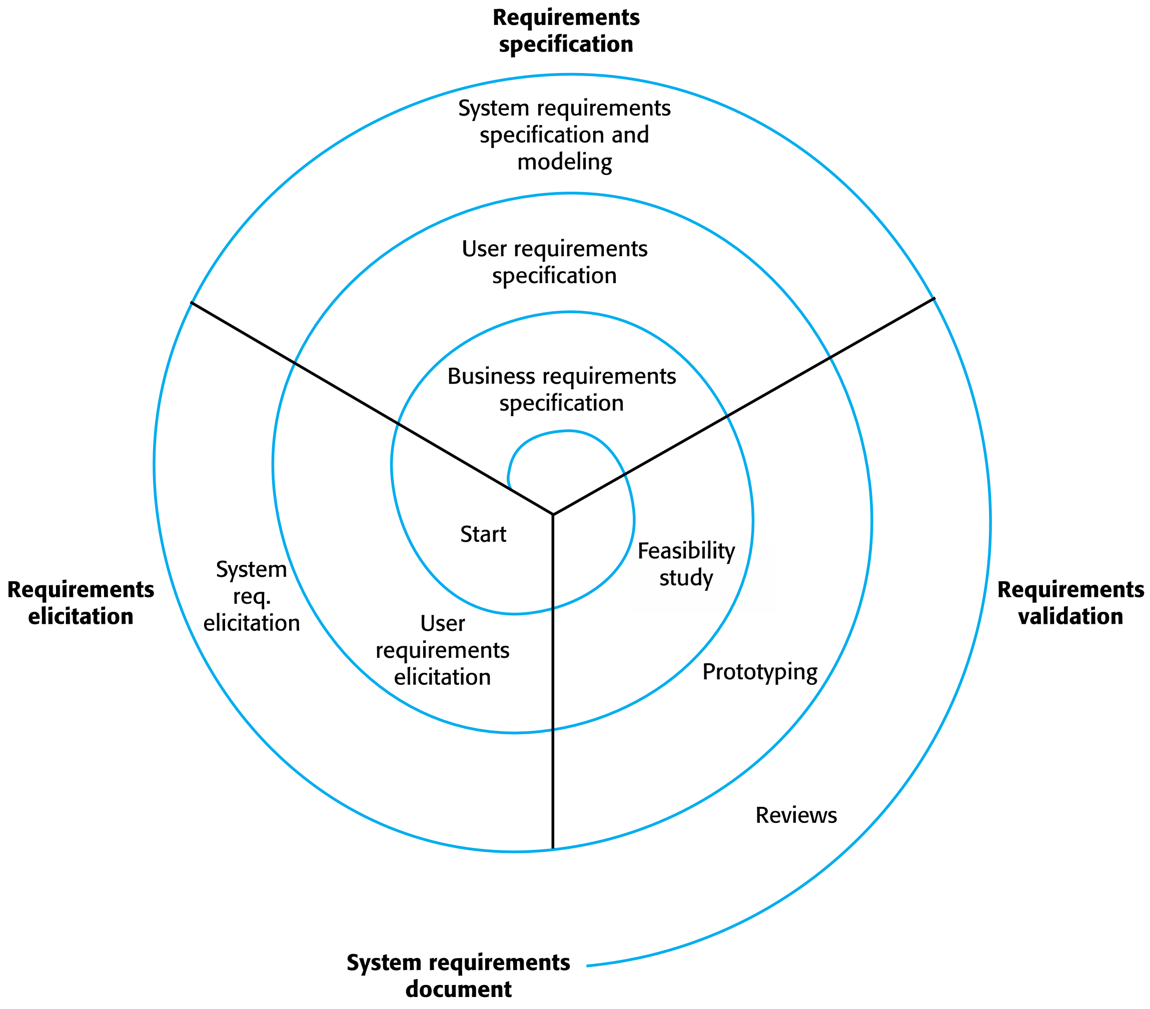
- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

UNIVERSITY OF WOLLONGONG AUSTRALIA

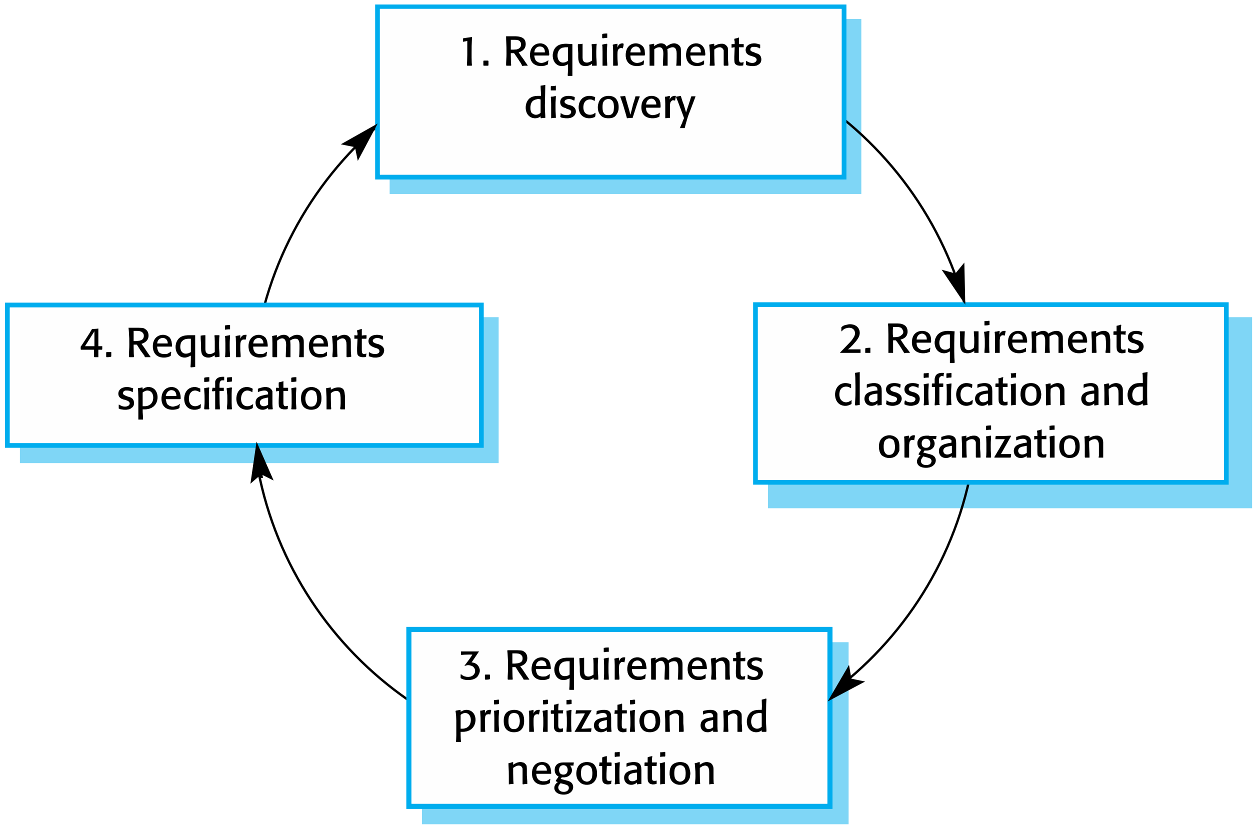
Waterfall model of the software process


System Requirement engineering
System requirements specification
Software Requirement Engineering
Software requirements specification
Software design
Software design specification
Programming and unit testing
Executable software system
System testing
Completed system
System operations

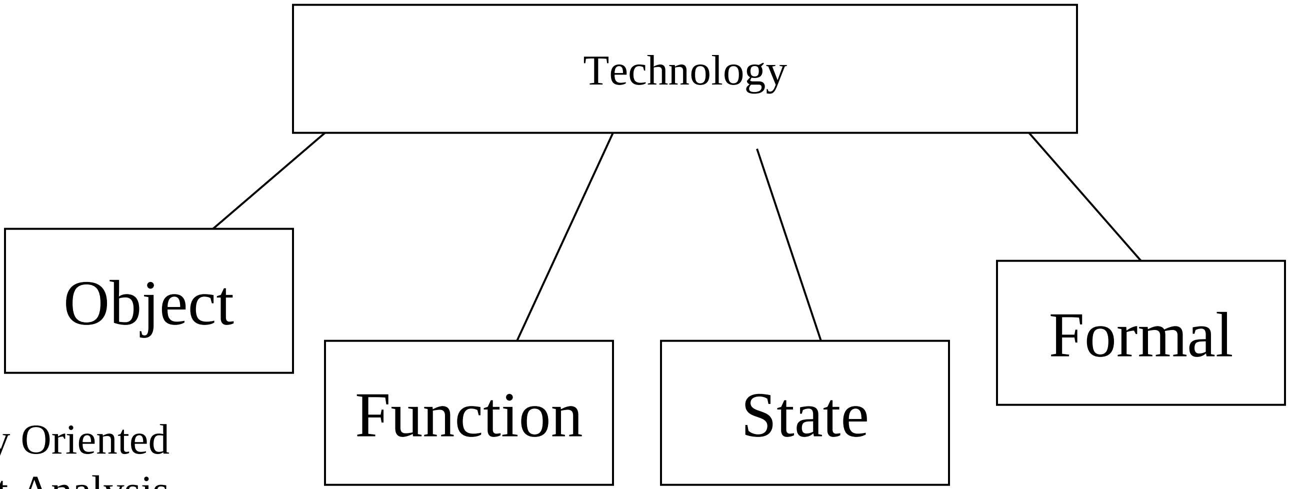
A spiral view of the requirements engineering process


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

The requirements elicitation and analysis process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

Methods for Requirements Engineering



- Ontology Oriented Requirement Analysis
- Concurrent Design Approach for Real-time System

- **Dataflow diagrams**
- Decision table
- Process specification
- Specification and Description Graphs
- Structure AnalysisS And Design Technique diagrams

- **Finite State Machine**
- **Petri nets**
- State chart

- Petri nets
- Constraint Satisfaction Problem
- Vienna Development Method
- **Z notation**

Data Flow Diagrams

A diagram that specifies the processes (also referred as bubbles, transforms, transactions, activities, operations) and flow of data between them is called a Data Flow Diagram (DFD).

A DFD exhibits possible forms of information flow in a system, storage locations for data, and transformations of data as it flows through a system.

Data Flow Diagram: Guidelines

- all icons must be labeled with meaningful names
- the DFD evolves through a number of levels of detail
- always begin with a context level diagram (also called level 0)
- always show external entities at level 0
- always label data flow arrows
- do not represent procedural logic

Constructing a DFD—Step I

- review user scenarios and/or the data model to isolate data objects and use a grammatical parse to determine "operations"
- determine external entities (producers and consumers of data)
- create a level 0 DFD

Constructing a DFD— Step II

- Write a narrative describing the transform
- Parse to determine next level transforms
- “Balance” the flow to maintain data flow continuity
- Develop a level 1 DFD
- Use a 1:5 (approx.) expansion ratio

Data Dictionary

- A data dictionary stores information about data items found in a DFD.
- A data dictionary supplies information such as a data typing, required accuracy of data useful o designers and implementers.
- A data dictionary can be used to check the completeness and consistency of DFDs. Whenever all bubbles, arrows, and databases (repository) have labels and all arrows have sources and destinations, DFD is considered complete.

Data Dictionary Features

Information Stored	**Explanation**
Name	Identifies data item
Alias	Identifies other names, abbreviations used to identify a data item
Data structure (type)	Type of data (e.g. integer, queue)
Description	Indicates how (why) a data item is used
Duration (begins)	Life span of data (when created)
Accuracy	High, medium, low accuracy
Range of values	Allowable values of data item
Data flows	Identifies process that generate/receive data

Finite State Machines (FSM)

At its lowest level, observable behavior of a system can be described with Finite State Machines (FSMs), which are special cases of automata (abstract machine).

The state of a system can be determined by checking the value of one or more state variables.

Suppose that a system might have state variable q1, q2, q3, q4 with values from the set {waiting, reading, writing, searching}. The possible state sequence for the system might be:

$$\begin{tikzcd}
\text{waiting} \arrow[r] & \text{reading} \arrow[r] & \text{searching} \arrow[r] & \text{writing} \arrow[r] & \text{waiting} \\
\text{(state q1)} & \text{(q2)} & \text{(q3)} & \text{(q4)} & \text{(q1)}
\end{tikzcd}$$

Definition of a finite-state machine

A finite-state machine M is defined by a five-tuple (Q, F, q0, S, δ) where

- Q is a finite set of states {q0, q1, q2, …, qn}
- F is a subset of final (accepting) state of Q
- q0 is a single start state in Q
- S is an input/output alphabet
- δ: Q x S $\rightarrow$Q maps the current input and current state into the next state

State table

The behaviors of FSMs can be represented in a table form. For example, a complete representation of the behavior of machine (b) is given in the following table.

State \ Input	a	b
q1	q2	q3
q2	q2	Φ
q3	Φ	Φ

Petri Nets

We have learnt:

- Principle of constructing a Petri Net
- Firing rule
- Basic properties of Petri Nets
- Problem solving by Petri Nets (assignments, examples)

PN definition

A Petri Net is a bipartite directed multi-graph, $G=(V,A)$, where $V= \{v_1, v_2, \ldots, v_s\}$ is a set of vertices and $A = \{ a_1, a_2, \ldots, a_r\}$ is a bag of directed arcs, $a_i = \{v_j, v_k\}$ with $v_j, v_k \in V$.

The set V are partitioned into two disjoint sets P *(circle nodes, i.e. places)* and T *(bar nodes, i.e. transactions)* such that $V = P \cup T$, $P \cap T = \emptyset$, and for each directed arc, $a_i \in A$, if $a_i = (v_j,v_k)$, the either $v_j \in P$ and $v_k \in T$ or $v_j \in T$ and $v_k \in P$.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Definition

A *Petri net structure*, C, is a four-tuple, $C=(P,T,I,O)$.

$P=\{p_1,p_2,\ldots,p_n\}$ is a finite set of places, $n \geq 0$.

$T=\{t_1,t_2,\ldots,t_m\}$ is a set of transitions, $m \geq 0$.

The set of places and the set of transitions are disjoint,

$P \cap T = \emptyset$.

$I: T \rightarrow P^{\infty}$ is the input function, a mapping from transitions to bags of places, and indicates the input places of a transaction.

$O: T \rightarrow P^{\infty}$ is the output function, a mapping from transitions to bags of places, and indicates the output places of a transaction.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Marked PN

- A marked PN contains tokens
- Tokens are depicted graphically by dots (•) and reside in places
- A marking of a PN is a mapping that assigns a non negative integer (the number of tokens) to each place of the net
- The marking characterizes the state of the Petri Net
- The initial marking is referred to as μ

Execution Rules for Petri Nets

- A transition t is called *enabled* in a certain marking, if:
  - For every arc from an input place p to transaction t, there exists a distinct token in the marking
- An enabled transition can be *fired* and result in a new marking
- Firing of a transition *t* in a marking is an atomic operation

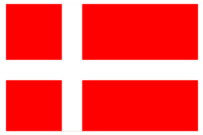
UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Petri net analysis

- Petri net reachability
- Petri net liveness
- Petri net soundness
- Petri net safeness
- Petri net conservation
- Petri net conflict
- Structural Analysis: P-invariants and T-invariants
- Petri Nets Modelling

What is a Coloured Petri Net?

- *Modelling language* for systems where ***synchronisation***, ***communication***, and ***resource sharing*** are important.
- Combination of *Petri Nets* and *Programming Language.*
  - *Control structures*, *synchronisation*, *communication*, and *resource sharing* are described by *Petri Nets.*
  - *Data* and *data manipulations* are described by *functional programming language.*
- CPN models are *validated* by means of *simulation* and *verified* by means of *state spaces* and *place invariants.*
- Coloured Petri Nets is developed at *University of Aarhus*, *Denmark* over the last 25 years.



Colour sets = Types

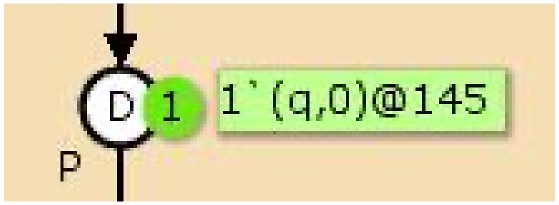
- We use data types to specify the kinds of tokens which we allow on the individual places.
- Types can be arbitrarily complex:
  - *Atomic* (e.g., integers, strings, Booleans and enumerations).
  - *Structured* (e.g., products, records, unions, lists, and subsets).
- The use of types allows us to make more readable descriptions with mnemonics type names such as:
  - **PROD, CONS, PACKETS**
- We also get more correct descriptions.
  - Automatic type checking of arc expressions.

Timed CPN

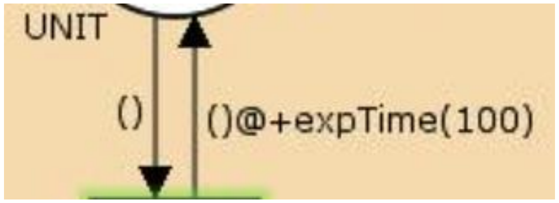
- Time-related information are required.
- Can CPN measure time?
- Can transitions be fired by considering a time stamp?

The answer is YES.

- We can put time stamps on the tokens, and

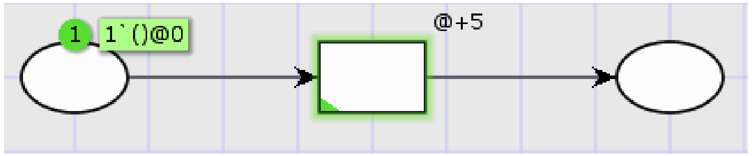


- We can modify the transitions and add arc functions to consider the time stamp.

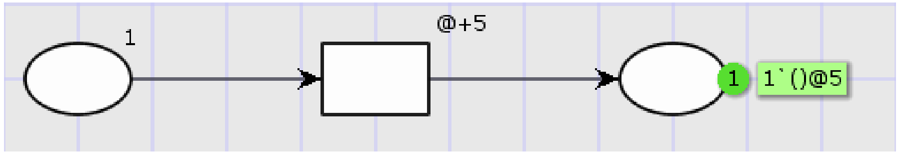


Time Intervals

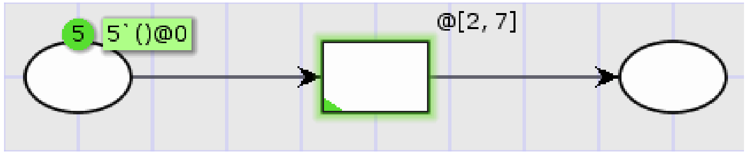
Timed models allow us to specify a duration for each transition, like this:



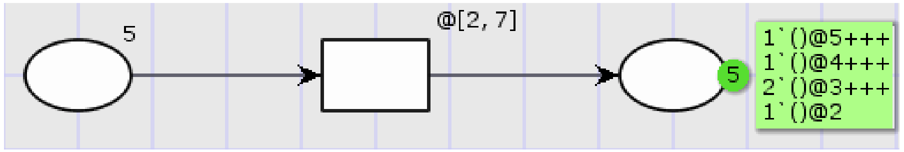
When we execute the transition, the produced token will have a time-stamp that is 5 time units into the future, indicating we cannot consume the token before time 5:



In CPN Tools 4, we can also specify an interval, indicating that the actual value of the time stamp should be taken at random from the interval:



Now, if we execute the transition (5 times), we get this state:

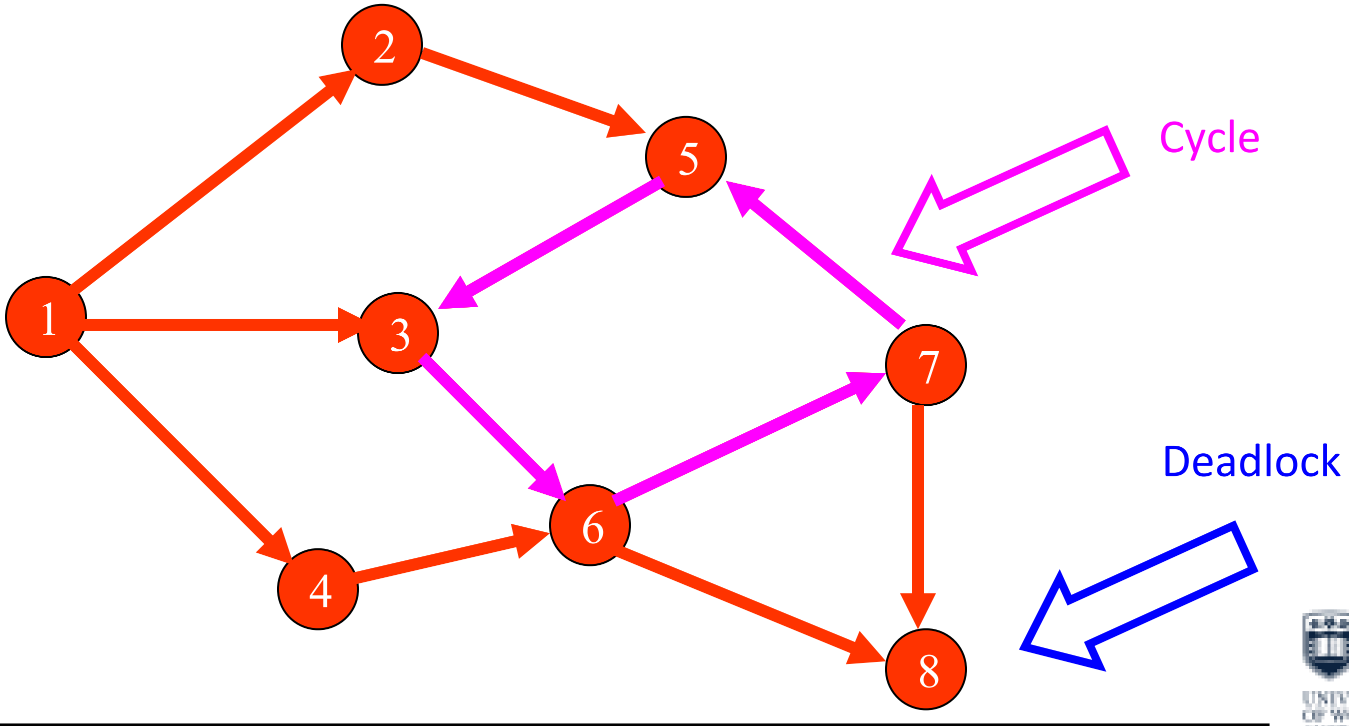


We see that the tokens have different time stamps; they are chosen at random (uniformly) from the interval specified. We could of course achieve something similar using one of the ~~probability distribution functions~~:

State spaces

- A *state space* is a *directed graph* with:
  - A *node* for each *reachable marking* (i.e., state).
  - An *arc* for each *occurring binding element.*

transition + binding



Model-based specification

- Defines a model of a system using well-understood mathematical entities such as sets and functions
- The state of the system is not hidden (unlike algebraic specification)
- State changes are straightforward to define
- Z are the most widely used model-based specification languages
- First order logic and set theory

Z method

- Z method is also called as Z notation.
- Z method can be used for requirement specification or design specification, in software process.
- The mathematical foundation of Z is typed set theory.
- Schemas are specification building blocks.
- Graphical presentation of schemas make Z specifications easier to understand,

UNIVERSITY OF WOLLONGONG AUSTRALIA

Z schemas

- Schemas introduce specification entities and defines invariant predicates over these entities.
- Schemas can be included in other schemas and may act as type definitions.
- (Ξ) and (Δ) notation are important symbols to indicate different types of schemas inclusions.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Steps for Z Model

1. Define the data type
    1. free type
    2. basic type
2. Define the axiomatic descriptions
    1. global variables
    2. global constrains
3. Define the system state schema
    1. local variables
    2. system state invariants
4. Initialise the system state schema
    1. Initialise all local variables and global variables
5. Create full operation schema
    1. successful scenarios (when all pre-conditions are true)
    2. non-successful scenarios (when each pre-condition is false)
    3. combine the successful and non-successful scenarios

Operation specification in Z

- Operations may be specified incrementally as separate schemas then the schemas combined to produce the complete specification
  - Define the ‘normal’ operation as a schema
  - Define schemas for exceptional situations
  - Combine all schemas using the disjunction (or) operator