Software Requirements Engineering (SE2001)

Dr. Syed Muazzam Ali Shah

HEC Approved Ph.D. Supervisor

(Assistant Professor)

Department of Software Engineering
NUCES-FAST Karachi

Contact#: ***(021) 111-128-128 Ext. 130***
Website: sites.google.com/view/muazzam-kazmi/home
Official page: https://khi.nu.edu.pk/personnel/dr-syed-muazzam-ali-shah-2/
Google Scholar: https://scholar.google.com.tw/citations?hl=en&user=0vcfR-IAAAAJ

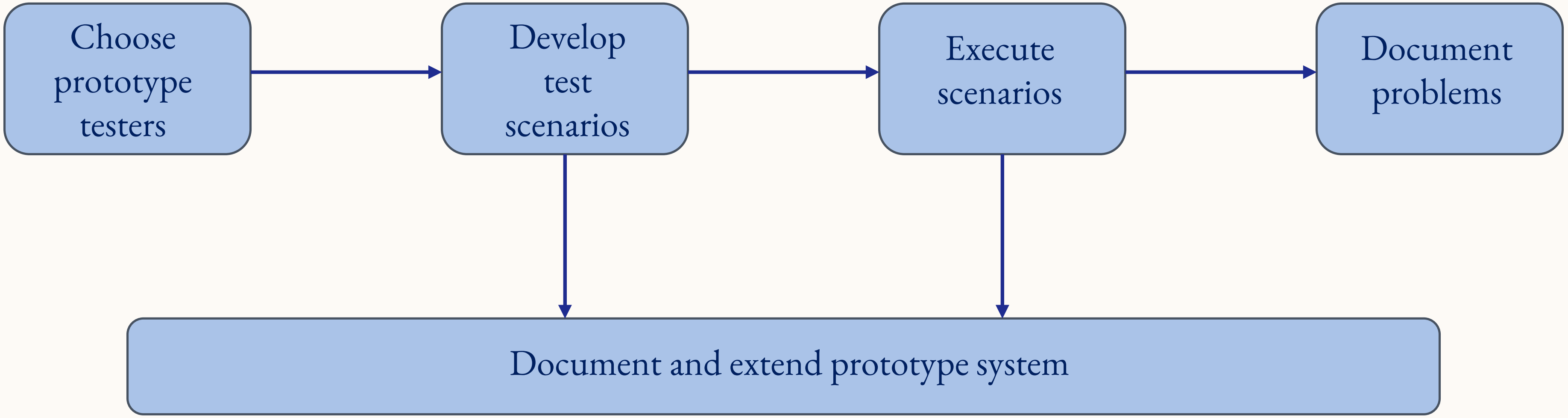
Validation Techniques

- ❖Review checklists
- ❖Prototyping
- ❖User manual development
- ❖Model validation
- ❖Requirements testing

Prototyping

- Prototypes for requirements validation demonstrate the requirements and help stakeholders discover problems
- Validation prototypes should be complete, reasonably efficient and robust.
- It should be possible to use them in the same way as the required system
- User documentation and training should be provided

Prototyping for Validation



Prototyping Activities - 1

- ❖ **Choose prototype testers:**
  - □ The best testers are users who are fairly experienced and who are open-minded about the use of new systems.
  - □ End-users who do different jobs should be involved so that different areas of system functionality will be covered

Prototyping Activities - 2

❖ Develop test scenarios:

- Careful planning is required to draw up a set of test scenarios which provide broad coverage of the requirements.
- End-users shouldn’t just play around with the system as this may never exercise critical system features

Prototyping Activities - 3

- ❖ **Execute scenarios:**
  - □ The users of the system work, usually on their own, to try the system by executing the planned scenarios
- ❖ **Document problems:**
  - □ Its usually best to define some kind of electronic or paper problem report form which users fill in when they encounter a problem

User Manual Development - 1

- Writing a user manual from the requirements forces a detailed requirements analysis and thus can reveal problems with the document.

User Manual Development - 2

❖ **Information in the user manual:**

- □ Description of the functionality and how it is implemented
- □ Which parts of the system have not been implemented
- □ How to get out of trouble
- □ How to install and get started with the system

System Models

- For some projects, system models may be developed based on the agreed set of requirements
- These models may be data-flow models of the system’s functionality, object models, event models, entity-relation models

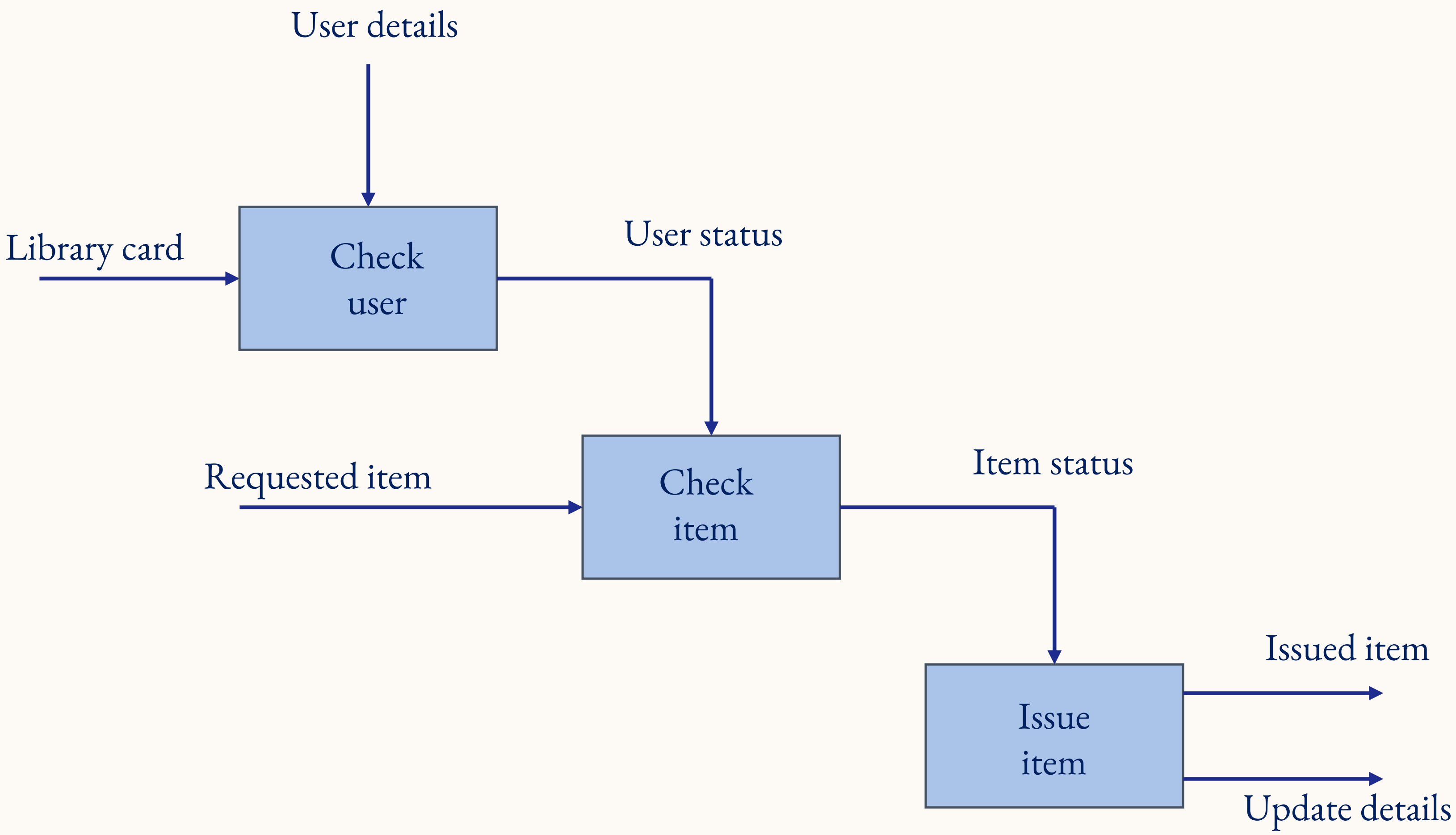
System Models

- Validation of system models is an essential part of the validation process
- Some checking is possible with automated tools
- Paraphrasing the model is an effective checking technique

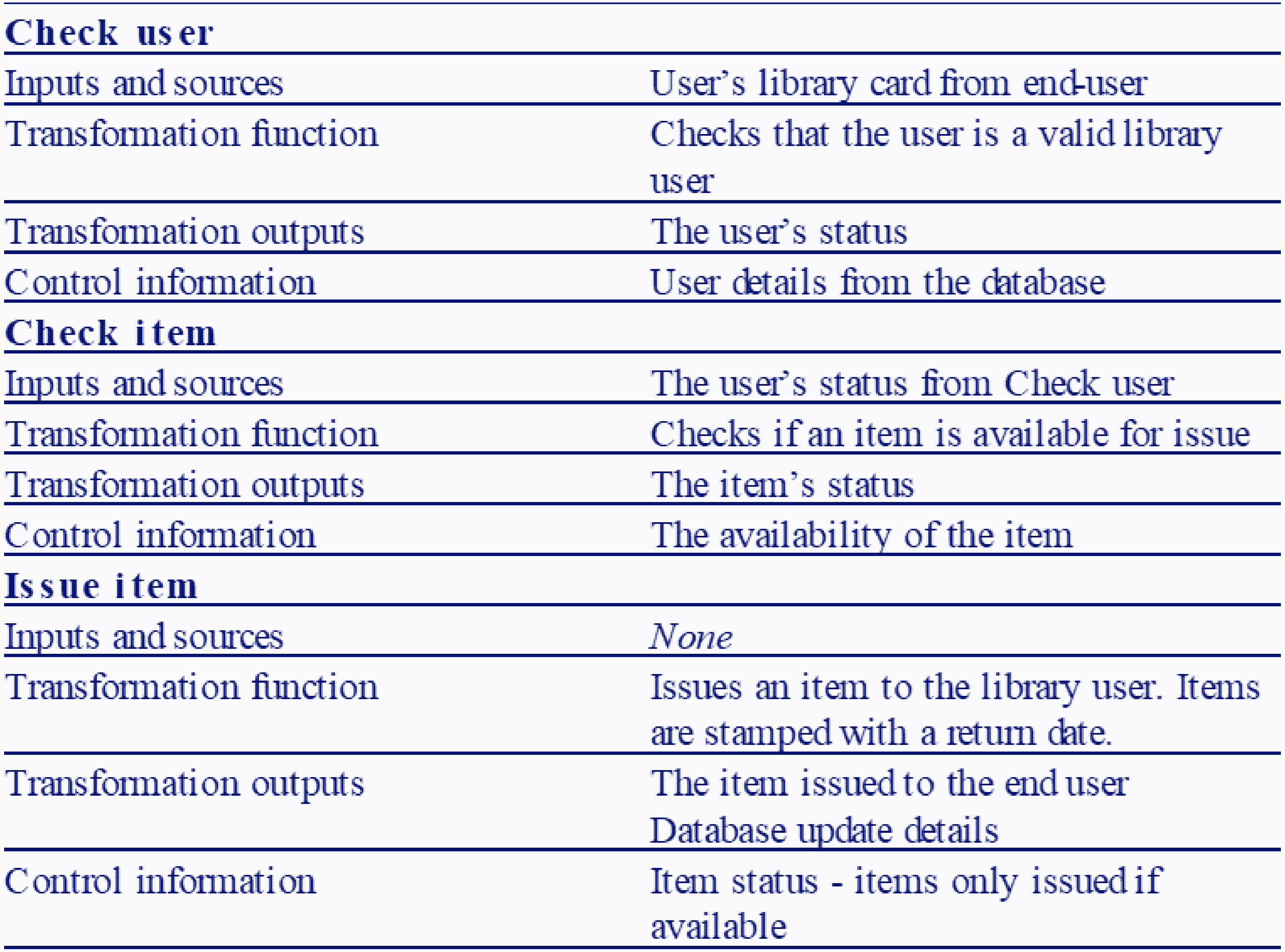
Objectives of Model Validation

- To demonstrate that each model is self-consistent
- If there are several models of the system, to demonstrate that these are internally and externally consistent
- To demonstrate that the models accurately reflect the real requirements of system stakeholders. This is very difficult

Data-flow Diagram for Issue



Paraphrased Description



Check user	
Inputs and sources	User's library card from end-user
Transformation function	Checks that the user is a valid library user
Transformation outputs	The user's status
Control information	User details from the database
Check item	
Inputs and sources	The user's status from Check user
Transformation function	Checks if an item is available for issue
Transformation outputs	The item's status
Control information	The availability of the item
Issue item	
Inputs and sources	*None*
Transformation function	Issues an item to the library user. Items are stamped with a return date.
Transformation outputs	The item issued to the end user Database update details
Control information	Item status - items only issued if available

Requirements Testing

- Each requirement should be testable i.e., it should be possible to define tests to check whether or not that requirement has been met
- Inventing requirements tests is an effective validation technique as missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Each functional requirement should have an associated test

Test Case Definition

- What usage scenario might be used to check the requirement?
- Does the requirement, on its own, include enough information to allow a test to be defined?
- Is it possible to test the requirement using a single test or are multiple test cases required?
- Could the requirement be re-stated to make the test cases more obvious?

Hard-to-test Requirements

- System requirements
- Exclusive requirements
- Some non-functional requirements

Hard-to-test Requirements

❖ System requirements:

- ➢ Requirements which apply to the system as a whole.
- ➢ In general, these are the most difficult requirements to validate irrespective of the method used as they may be influenced by any of the functional requirements.
- ➢ Tests, which are not executed, cannot test for non-functional system-wide characteristics such as usability

Hard-to-test Requirements

- **Exclusive Requirements:**
  - These are requirements which exclude specific behavior.
  - For example, a requirement may state that system failures must never corrupt the system database. It is not possible to test such a requirement exhaustively

Hard-to-test Requirements

❖ **Some Non-Functional Requirements:**

- Some non-functional requirements, such as reliability requirements, can only be tested with a large test set.
- Designing this test set does not help with requirements validation

THANK YOU

Dr. Syed Muazzam Ali Shah

muazzam.ali@nu.edu.pk