

The background image shows a large, modern university building at night. The building has a mix of architectural styles, including a prominent glass-enclosed circular structure with a dome and a long, low-profile section with a glass roof. The lights from the building are reflected in the dark water in front of it, creating a symmetrical pattern.

SOFTWARE ENGINEERING TEST REQUIREMENTS ELICITATION

SPRING 2018
CARSTEN RUSENG JAKOBSEN



AALBORG UNIVERSITY
DENMARK

TESTING PROCESSES



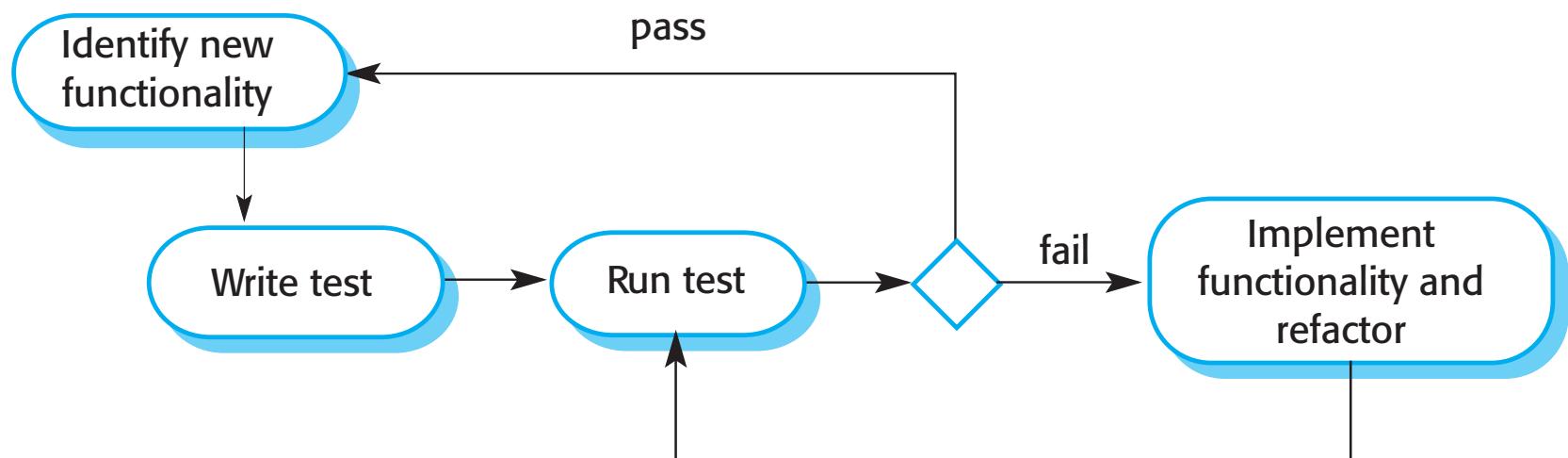
AALBORG UNIVERSITY
DENMARK

Test-driven development

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- Tests are written before code and ‘passing’ the tests is the critical driver of development.
- You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.



Test-driven development



TDD process activities

- Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.
- Write a test for this functionality and implement this as an automated test.
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- Implement the functionality and re-run the test.
- Once all tests run successfully, you move on to implementing the next chunk of functionality.



Benefits of test-driven development

- Code coverage
 - Every code segment that you write has at least one associated test so all code written has at least one test.
- Regression testing
 - A regression test suite is developed incrementally as a program is developed.
- Simplified debugging
 - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- System documentation
 - The tests themselves are a form of documentation that describe what the code should be doing.



Regression testing, requires frameworks

- Regression testing is testing the system to check that changes have not ‘broken’ previously working code.
- In a manual testing process, regression testing is expensive but, with automated regression testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.

The screenshot shows a code editor with two snippets of Java code. The top snippet is a test method:

```
@Test  
public void lookupEmailAddresses() {  
    assertThat(new CartoonCharacterEmailLookupService().getResults("loc...  
        not(empty()),  
        containsInAnyOrder(  
            allOf(instanceOf(Map.class), hasEntry("id", "56"), hasEntry("...  
            allOf(instanceOf(Map.class), hasEntry("id", "76"), hasEntry("...  
        )  
    )  
}
```

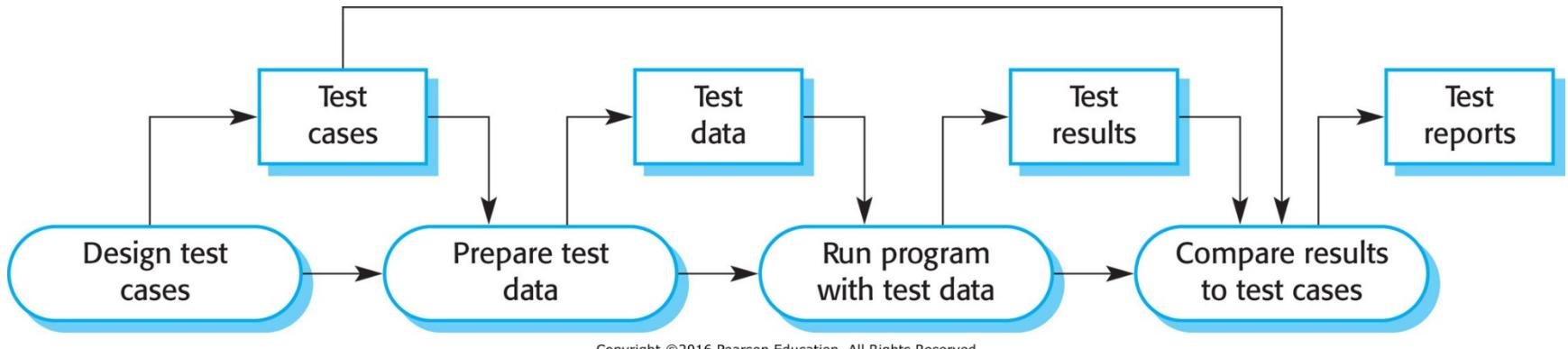
The bottom snippet is a class definition:

```
0 references  
public class CheckingAccount  
{  
    double m_balance;  
    0 references  
    public void Withdraw(double amount)  
    {  
        if (m_balance >= amount)  
        {  
            m_balance -= amount;  
        }  
        else  
        {  
            throw new Argueme...  
        }  
    }  
}
```

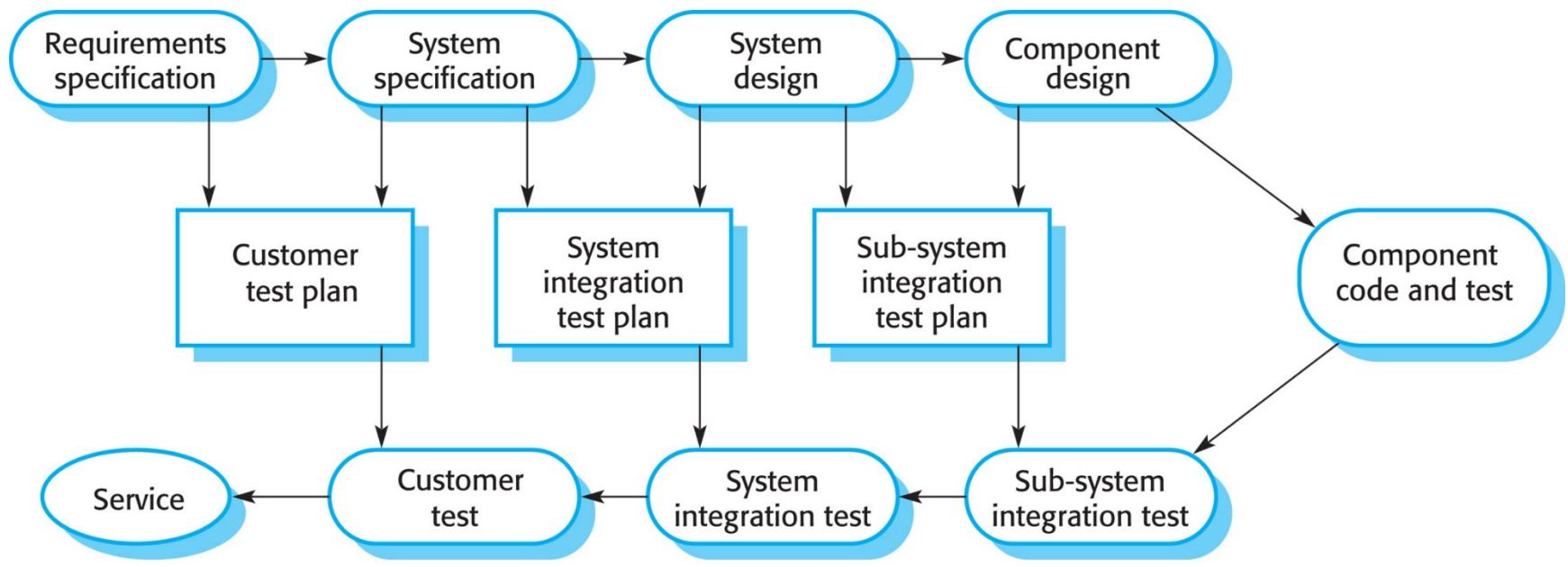
A context menu is open over the code, with the 'Create Unit Tests' option highlighted in yellow.



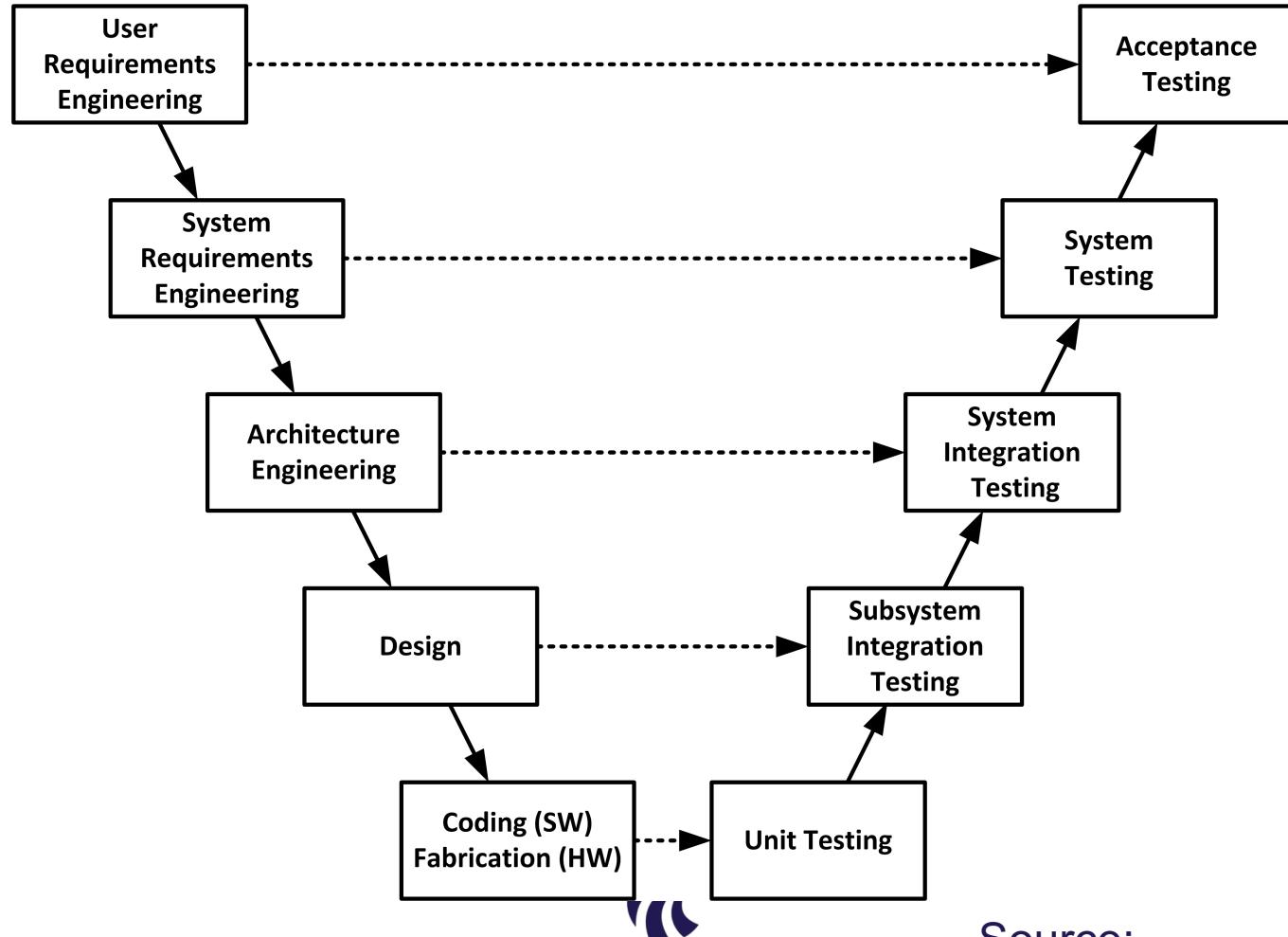
Plan-Driven Testing



Plan-Driven: V Model

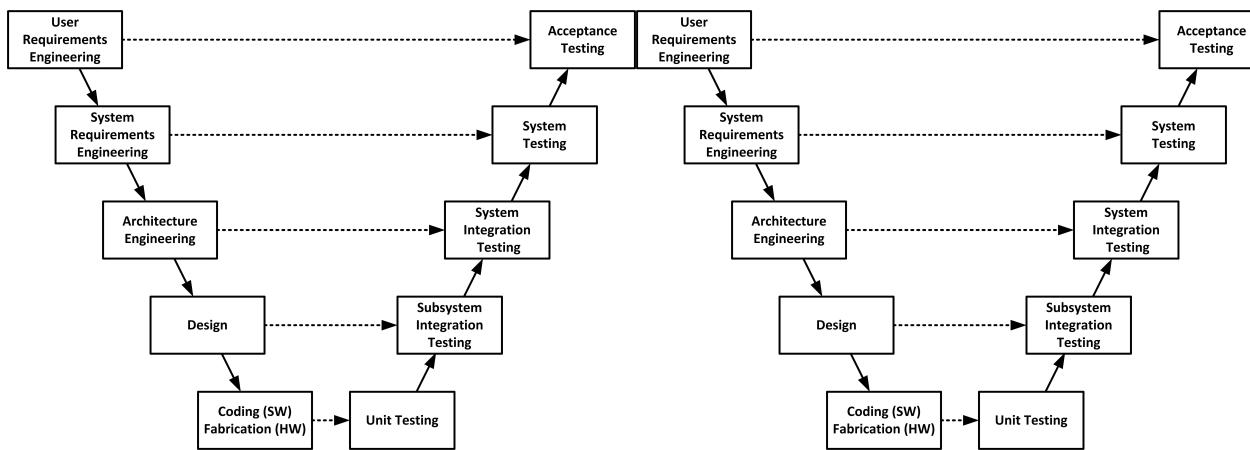


Plan-Driven: V Model



Source:
Software Engineering Institut

Agile: W Model

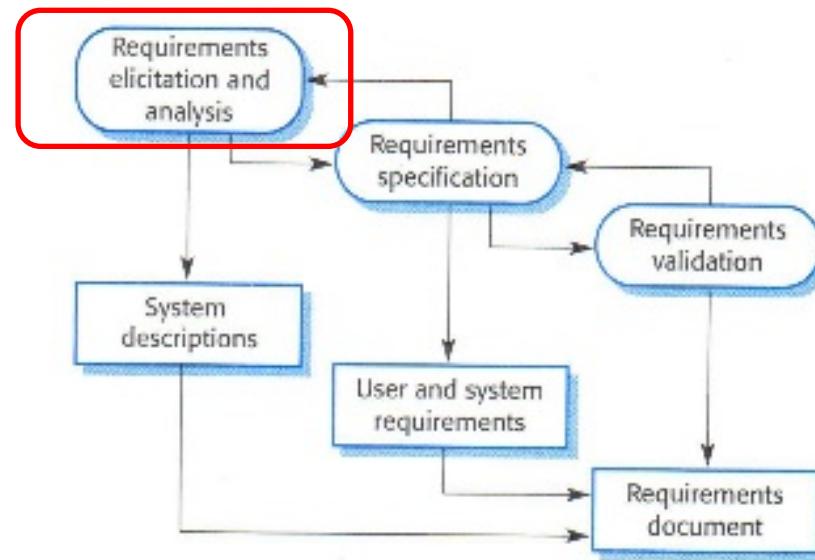


REQUIREMENTS ELICITATION



Requirements Elicitation

- From Lecture 1: Elicitation is part of Requirements Engineering



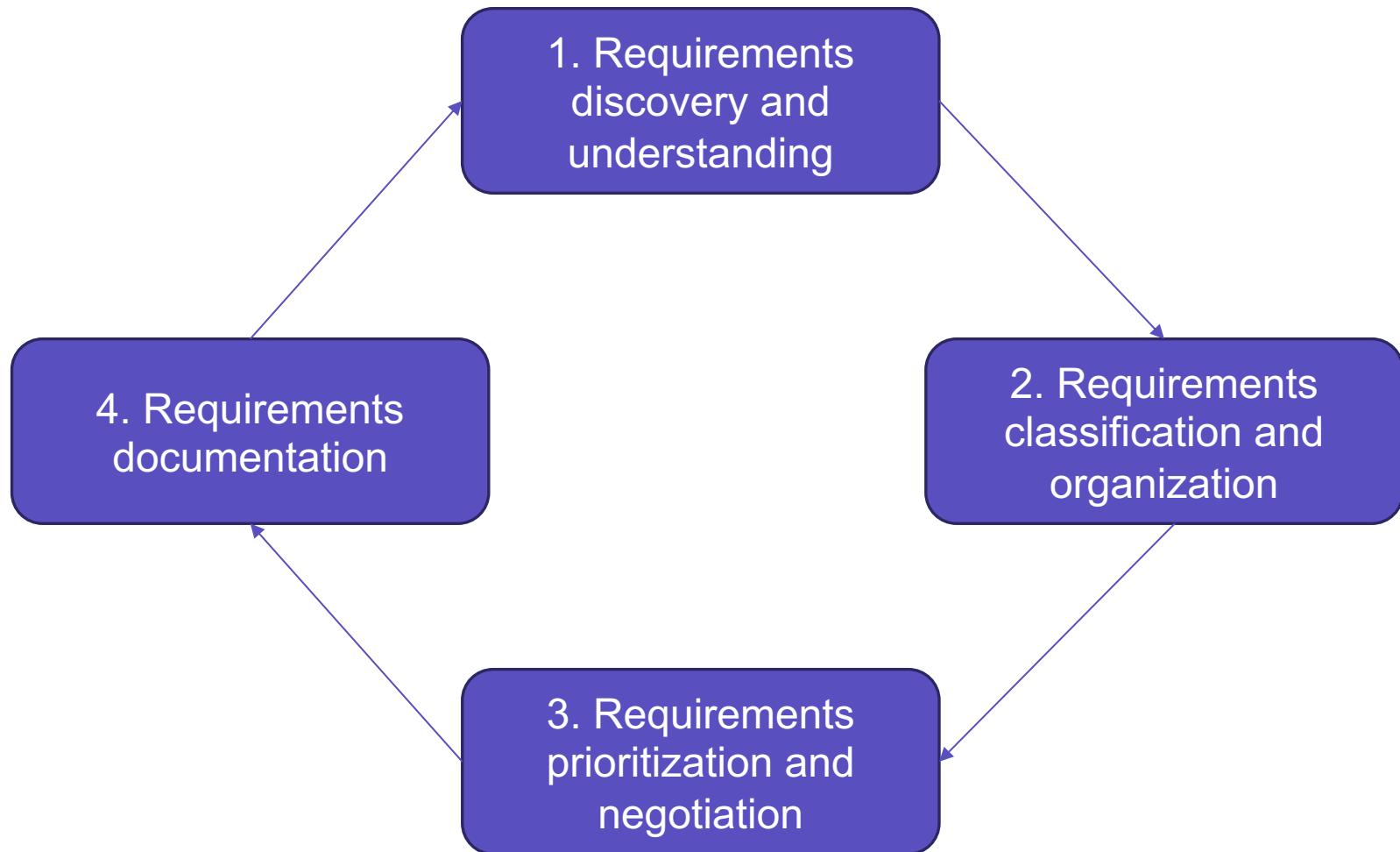
Requirements Elicitation

Requirements Elicitation is difficult because:

- Stakeholders don't know what they want
- Stakeholders use own language and implicit knowledge
- Different stakeholders with diverse or conflicting requirements
- Political factors
- Economic and business environment is dynamic



Requirement Elicitation and analysis process

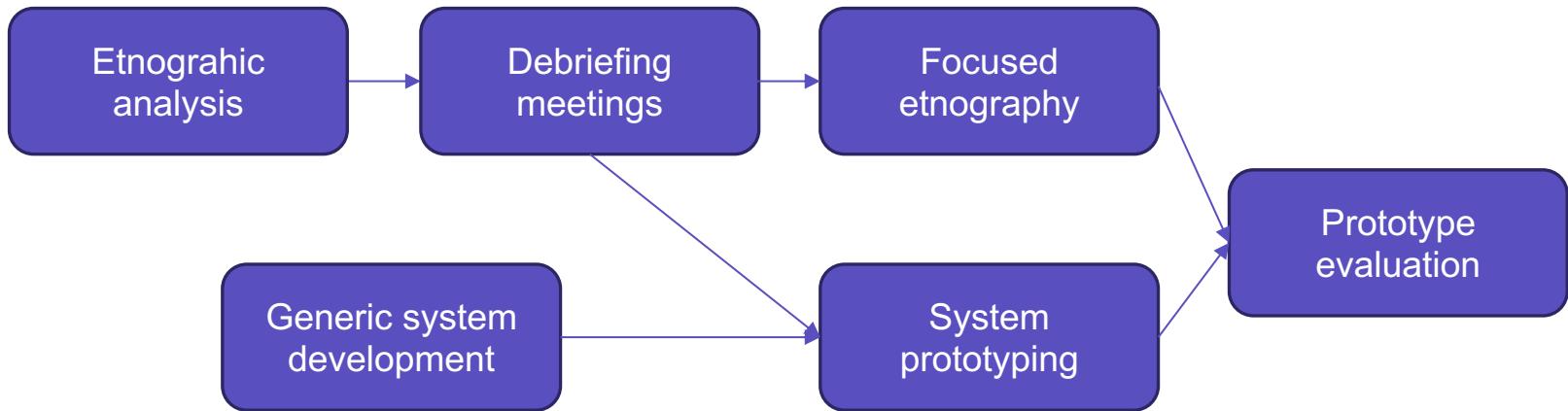


Requirements Elicitation Techniques

- Interview – Talk to people about what they do
 - Closed interviews with predefined questions
 - Open interviews with no predefined agenda
- Observation or Ethnography – watch people do their job.
Good at:
 - Observing how the work is actually performed
 - Requirements derived from cooperation and awareness of other people
 - Understand existing systems



Etnography and prototyping for requirements analysis



Etnography is not effective for discovering broader organizational or domain requirements or for suggesting innovation



Stories and scenarios

- How the system can be used for a specific task
- A scenario may include:
 - What the system and users expect when the scenario starts
 - Normal flow of events in the scenario
 - What can go wrong and how resulting problems are handled
 - Information about other activities at the same time
 - System state when the scenario ends

People find it easier to relate to real-life examples than abstract descriptions

