COURSE CONTENT

- Introduction
- Software Lifecycle
- Requirements and Specifications
- Design
- Implementation and Deployment
- Testing
- Conclusion

User interface design

- Design issues
- The user interface design process
- User analysis
- User interface prototyping
- Interface evaluation

THE USER INTERFACE

- User interfaces should be designed to match the skills, experience and expectations of its anticipated users.

System users often judge a system by its interface rather than its functionality.

A poorly designed interface can

- cause a user to make catastrophic errors.
- is the reason why so many software systems are never used.

HUMAN FACTOR

Limited short-term memory

People can instantaneously remember about **7 items** of information.

If you present more than this, they are more liable to make mistakes.

People make mistakes

When people make mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.

People are different

People have a wide range of physical capabilities. Designers should not just design for their own capabilities.

People have **different interaction preferences**

Some like pictures, some like text.

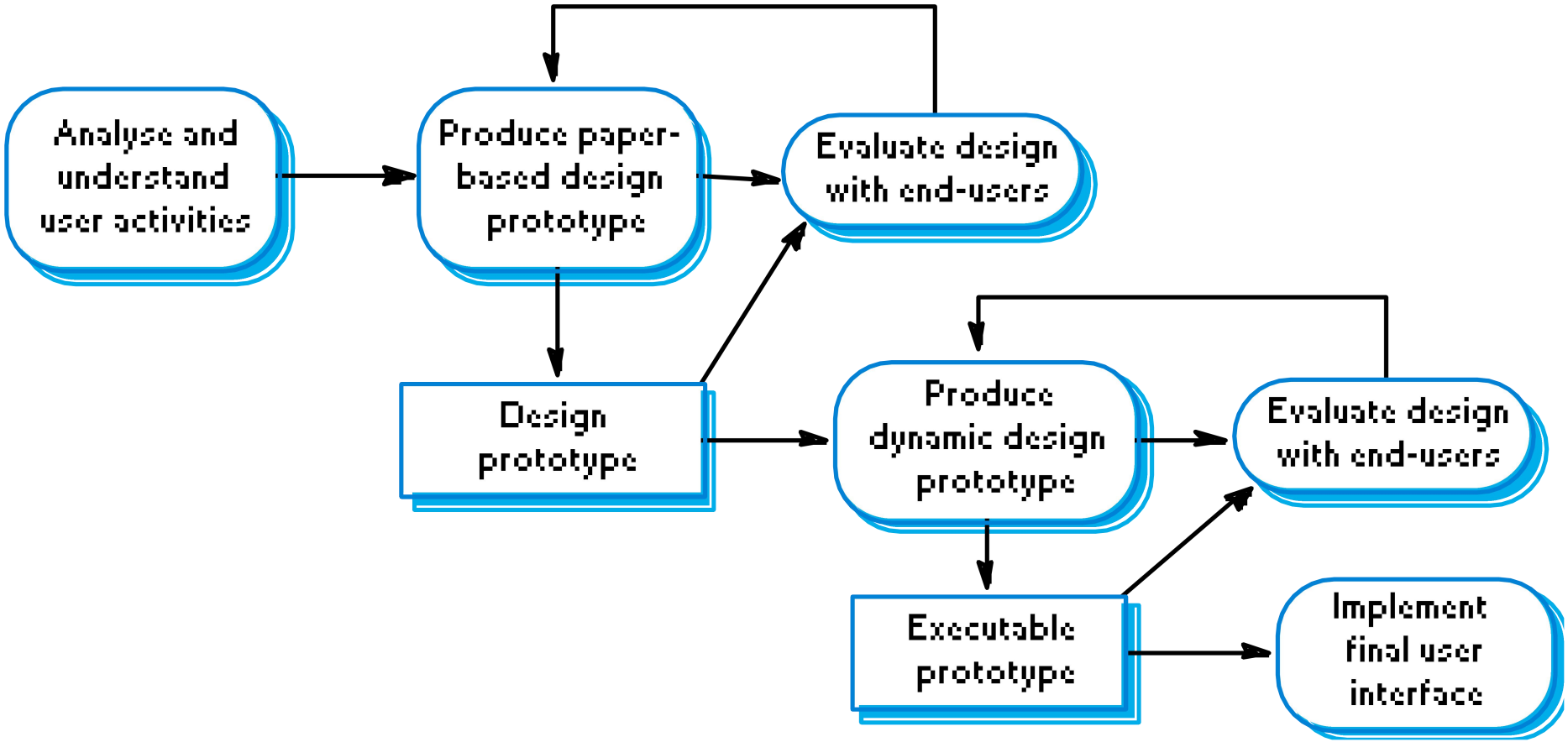
DESIGN PRINCIPLES

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.
Consistency	The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.
Minimal surprise	Users should never be surprised by the behaviour of a system.
Recoverability	The interface should include mechanisms to allow users to recover from errors.
User guidance	The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.
User diversity	The interface should provide appropriate interaction facilities for different types of system user.

DESIGN PRINCIPLES

Principle	Examples
User familiarity	Click enter always go next, click esc always close
Consistency	Undo, confirmation of destructive actions, 'soft' deletes, etc
Minimal surprise	Office system use concepts such as letters, documents, folders
Recoverability	Help section, online manuals, etc.
User guidance	Larger/smaller text/button for different people, etc.
User diversity	Commands and menus have the same format

DESIGN PROCESS



UI EVALUATION

- Some evaluation of a user interface design should be carried out to assess its suitability.
- Full scale evaluation is very expensive and impractical for most systems.
- Ideally, an interface should be evaluated against a usability specification. However, it is rare for such specifications to be produced.

Simple EVALUATION techniques

- Questionnaires for user feedback.
- Video recording of system use and subsequent tape evaluation.
- Instrumentation of code to collect information about facility use and user errors.
- The provision of code in the software to collect on-line user feedback.

IMPLEMENTATION

IMPLEMENTATION basics

Implementation is the step where we actually begin coding our project. We have detailed what objectives the system should meet, have come up with a good design, and are now ready to begin coding.

The PROGRAMMERS

Core Principals

- Use a **style guide**, so all the code looks generally the same.
- Code is written **for people**, not computers.
- Make **modules** easy to learn and understand.
- Go into everything with a **plan**. (You can experiment, but clean up after)
- Shorter code does not equal better code! **MAKE IT READABLE**.
- Break up actions into methods.

Buy vs. Build

TASK

Using the core principals, draft the implementation plan for your team. This plan should include the **style guide** for the whole team; the modules which your team is going to build or buy, and modules that each member contributes.

note: *this implementation plan is shown in your end-term report*

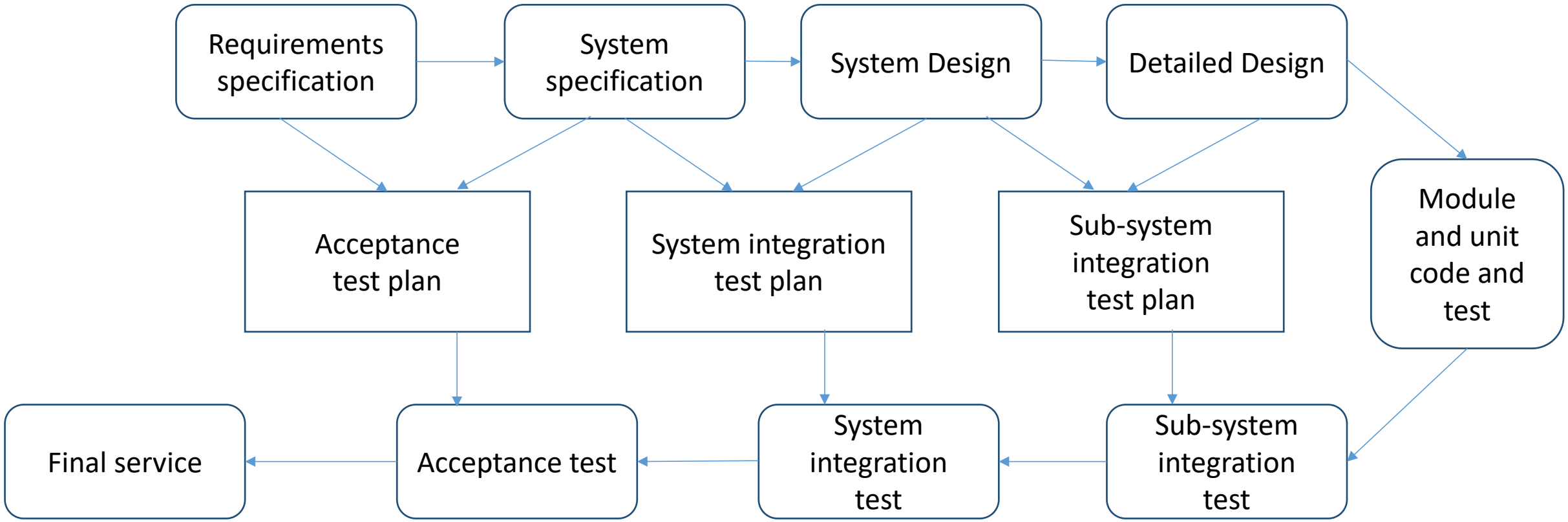
TESTING

- Definitions
- Testing process
- Unit Testing
- System Testing
- Blackbox and Whitebox Testing
- The Problem with Testing

TESTING

- Testing is the process of finding errors
- CODE, IMPLEMENTATIONS, DOCUMENTATION, OTHERS
- Testing can never prove that a system is correct. It can only show that (a) a system is correct in a special case, or (b) that it has a fault.

TESTING


Requirements specification
System specification
System Design
Detailed Design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Final service
Acceptance test
System integration test
Sub-system integration test

Testing and Debugging

Testing is most effective if divided into stages:

- Unit testing at various levels of granularity

  tests by the developer
  emphasis is on accuracy of actual code

- System and sub-system testing

  uses trial data
  emphasis is on integration and interfaces

- Acceptance testing

  uses real data in realistic situations
  emphasis is on meeting requirements

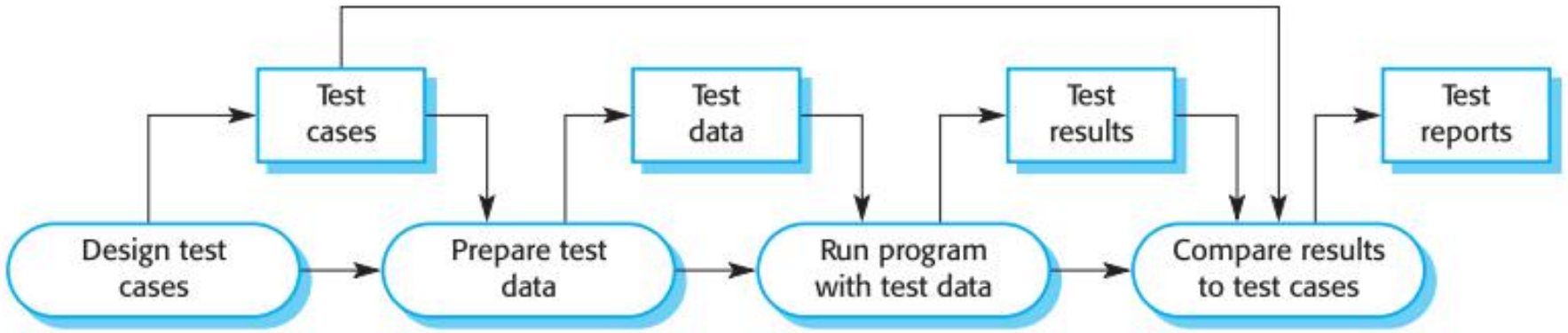
Testing Process

System and Acceptance Testing is a major part of a software project

- It requires time on the schedule
- It may require substantial investment in datasets, equipment, and test software.
- Good testing requires good people!
- Management and client reports are important parts of testing.

What is the definition of "done"?

Testing Process



DEPLOYMENT

- OVERVIEW
- DEPLOYMENT PLANNING
- DEPLOYMENT ROLLBACK