Software Construction

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Introduction

- **Software Construction** lies between design and test and is often part of an iterative

  "design -> construct -> test"

  cycle at the heart of most development processes
- Process is one of going from a design to implemented code
- Code is generally much more complex than the design and will require a myriad of detailed **design** or **implementation** activities
- Very sensitive to the platform and the need to **"work around"** problems

Main Activities

- **Managing Complexity**: programmed systems are always much more complex than the corresponding design – this involves approaches to coping with that complexity
- **Managing Change**: the environment changes and the platform and components change regularly – this involves developing systems that are resilient to anticipated change
- **Facilitating Validation and Verification**: the final code will be subject to test, review, walkthroughs – the structure of the code is an essential influence on ease of validation
- **Standards Compliance**: important to ensure the capacity to interwork and sometimes an essential to a product

Managing Complexity

- **Avoid complexity**: by forcing a redesign to remove the complexity from the system
- **Automate complexity**: use tools to carry out complex, error-prone, tasks that are well understood, encapsulate complexity in manageable components
- **Localize complexity**: use structuring and hiding to contain complexity inside manageable boundaries. Conceptual tools like **coupling** and **cohesion** are useful in identifying and managing locality.

Managing Change

- Both the **Environment changes** and the **platform** and **components** **evolve** (some systems just freeze the system and undertake limited maintenance).
- Main management approaches:
  - Attempt to generalize the **interface** to components so the component is capable of easy adaptation
  - Experiment to attempt to identify **variability** and likely directions for change in the environment
  - Exploit **locality**- attempt to generate designs with low coupling so change in components and environments have limited impact on the overall system

Facilitating Validation

- There are a variety of ways of validating systems and
- Approaches in the code can be helpful
  - **Manual inspection**: comments, structure for reading, documentation
  - **Test**: code structure to allow good unit test and sensible interfaces so that integration test and the creation of "stubs" and "drivers" is easy
  - **Analysis tools**: use of restricted languages or structure within an existing language (e.g., SPARKAda, C#, etc.)

Complying with Standards

- Standards which (e.g., programming languages, communication methods, platforms, tools, etc.) directly affect construction issues
- **External standards** are often crucial in determining the saleability of products, e.g.:
  - XML, POSIX, CORBA, COM, DCOM, and so on in order to ensure interoperability
  - Quality standards, e.g., Capability Maturity Model (CMM) or ISO 9001 can be crucial in gaining contracts
- **Internal standards** contribute towards organization or project work practice and knowledge

Managing Construction

- **Construction Models**. Software development models (e.g., waterfall, spiral, V-model, evolutionary prototyping, extreme programming, etc.) differently emphasize construction (e.g., linear or iterative models, risk-oriented models, etc.)
  - The underlying hypotheses constrain the software construction
- **Construction Planning**. Construction methods affect the project's ability to reduce complexity, anticipate changes and construct for verification
  - Construction planning also defines the order in which (planned processes produce) deliverables (e.g., requirements, design, software, etc.) are created and integrated)
- **Construction Measurement**. Quantitative approaches support the monitoring and the assessment of construction activities and deliverables

Construction Languages

- **Software Construction**:
  - Produces the highest volume of **configuration** items that need to be managed in software projects
  - Relies on **tools** and **methods**
- **Configuration languages**: used to configure parts of the platform and many of the components used in building systems
- **Toolkit languages**: based around a particular toolkit – oriented to generating a particular component or configuration of components
- **Scripting languages**: often used to capture elements of workflow
- **Programming languages**: general purpose, oriented to creating new functionality from scratch

Summary

- Software Construction
  - Managing Complexity
  - Managing Changes
  - Facilitating Validation
  - Complying with Standards
- Managing Construction
- Construction Languages
- Reference
  - Chapter 4 of the SWEBOK on Software Construction.