CDT413: Course Update

- Mon 28 Jan 15:15-17:00, Lecture: “Introduction to Software Architecture”
- Wed 30 Jan, Assignment 1 – Deadline Wed 13 Feb
- Fri 1 Feb 11:00-12:00, Extra lecture by Judith Stafford: “Documented Software Architecture”
- Mon 4 Feb 13:15-15:00, Lecture: “Software Architecture Design” + presentation of Seminar 1

Introduction to Software Architecture

- David Parnas on Software Design (1972)
- Shaw and Garlan on Software Architecture (1994)
- Bass et. al. on Software Architecture (2003)
  - Motivation and terminology

David Parnas on Software Design

- “On the Criteria to Be Used in Decomposing Systems into Modules” (1972)
- Illustrates how the criteria used when deciding what modules to include in a system affects how easy the system will be to maintain
- Uses a simple example: KWIC (Key Word in Context)

The KWIC Index System

The KWIC [Key Word in Context] index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be “circularly shifted” by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order

[Parna 1972]
The KWIC Index System

- A simple example that can easily be implemented by one programmer
- Still useful to illustrate how a system may be decomposed into modules that can potentially be assigned to different development teams

KWIC: Functional Decomposition

- Read input
- Create circular shifts
- Sort in alphabetical order
- Write output

KWIC: Modularization 1

- One subroutine for each of the four functions
- Main program “Master control” calls each of the subroutines in turn
- The subroutines operates on shared data

KWIC: Modularization 1

- Simple and intuitive
- Follows the basic functional decomposition
- Shared data enables efficient use of memory
- There is no problem with concurrent access since the subroutines executes one at a time

Problems with Modularization 1

- Changing the format for data storage
  - E.g., lines can be stored in different ways and circular shifts may be stored explicitly or implicitly as index/offset pairs
- Changes in processing algorithm
  - E.g., lines can be shifted as they are read or after all lines have been read and sorting can be done incrementally
- Functional enhancements
  - E.g., ignoring words like “in” and “to”
  - Such changes will require updates to several modules
KWIC: Modularization 2

• One module for each “design decision”
  • Input format
  • Line storage
  • Circular shifts
  • Alphabetical shifts
  • Output format
• Each module has its own data storage and provides procedures that other modules can call

KWIC: Modularization 2

• Algorithms and data storage formats can be changed in individual modules without affecting others
• Illustrates the use of “information hiding”
  • The system should be decomposed such that each module hides one (or a few) design decision(s)
  • Is now often used with object-oriented design
• May result in less efficient use of time and memory

Shaw and Garlan on Software Architecture

• “An Introduction to Software Architecture” (1994)
• Structure and organization of software systems
  • Higher level of abstraction than algorithms and data structures
  • Becomes more important with increasing size and complexity

Software Architecture (Shaw & Garlan)

• Identifies a number of commonly occurring styles or patterns of software architecture
• Defines a framework for discussing architectures and styles in terms of components and connectors
• Terminology inspired by box-and-line diagrams commonly used by practitioners to describe software architectures

Pipes and Filters

• Components are called filters and each has a set of inputs and a set of outputs. They operate on streams of input data and produce streams of output data
• Connectors are called pipes and connect the outputs of filters to the inputs of other filters
• There is no sharing of data between filters except through the pipes
Pipes and Filters
- Filters are independent
- Reusable
- Easy to maintain and extend systems
- Difficult to use for interactive systems
- No shared data – may not be efficient

Abstract Data Types or Objects
- The components are instances of abstract data types which encapsulate data representation and expose abstract operations to other components
- The connectors are references between objects

Layered Systems
- The components are called layers
- Each layer is “connected” to at most two other layers, providing services to the layer above it and consuming the services of the one under it

Abstract Data Types or Objects
- The components are instances of abstract data types which encapsulate data representation and expose abstract operations to other components
- The connectors are references between objects

KWIC: Abstract Data Type Solution
- Input
- Circular Shift
- Alphabetizer
- Output
- Input Medium
- Output Medium

Bass et. al on Software Architecture
The Architecture Business Cycle

- Architectures are influenced by a number of factors
  - Different system stakeholders
  - Developing organization
  - Technical environment
  - Background and experience of the architect(s)
- Architectures affect the factors that influence them
  - Thus forming a cycle

Software Architecture (Bass et al.)

- The software architecture of a program or computing system is the structure or structures of the system which comprise software elements, the externally visible properties of those elements, and the relationships among them

Software Architecture Structures

- Complex software cannot be represented by one single structure
- Three main types of structures
  - Module structures, which consist of static software elements and relationships
  - Component-and-connector structures of dynamic software elements and relationships
  - Allocation structures, which map software elements to external resources