Role of Architecture in Disciplined Development

SW Engineering of Software Architecture

- What are we trying to gain/maintain control of in the Architectural Design phase?
- Profoundly effect system and business qualities
- Requires making tradeoffs
- Control implies achieving system qualities by choice not chance
  - Understanding what the tradeoffs are
  - Understanding the consequences of each choice
  - Making appropriate choices at appropriate times

Implications for the Development Process

Implies need for disciplined process for SW architecture covering these tasks

- Creating the “business case” for the system
- Understanding the requirements
- Designing the architecture
- Representing and communicating the architecture
- Analyzing or evaluating the architecture
- Implementing the system based on the architecture
- Ensuring the implementation conforms to the architecture

Related Design Questions

- Create business case for the system
  - What is the “business” rationale or goal?
- Understanding the requirements
  - What are the design goals?
- Creating or selecting the architecture
  - What are appropriate components and relations?
  - What are the decomposition principles?
- Representing and communicating the architecture
  - How are the components and relations represented?
- Analyzing or evaluating the architecture
  - How do we decide if the architecture is any good?

Architectural Design

What does “design an architecture” imply?

1. That we design to a (well-defined) purpose
   - System qualities from requirements
   - Strategic goals from business context
2. That we must choose appropriate components and relations for the required qualities
3. That we need appropriate representations to make the design decisions visible (support manipulation and communication)
4. That we need a decomposition method (design approach)
   - Methods describing what to do and when
   - Principles to guide making appropriate choices
5. That we need some metric of “goodness” relative to purpose

Architectural Structures

- To describe an architecture, we must answer:
  - What are the components? Relations? Interfaces?
  - To design one, must also know what it is good for - i.e., what quality attributes are determined by that structure.
- E.g., Calls Structure
  - Components: procedures
  - Relation: calls or invokes
  - Interface: procedure interface (parameters) and visible effects
  - Determines: execution flow (e.g., bottlenecks)
- Commonalties
  - Each portrays some aspects of the system, abstracts from others
  - The purpose is to support design of some attribute (or related set) of the system that is important to some stakeholder
  - Exactly what is visible and what is abstracted away depends on which attributes must be engineered

An Example:
The FWS Module Structure

Floating Weather Stations (FWS)

Floating weather stations are seagoing buoys that are equipped with sensors to monitor wind speed. Each FWS has an on-board computer that maintains a history of recent wind speed data. At regular intervals the buoy transmits the wind speed using a radio transmitter.

We will be developing the software for a new class of low-cost floating weather stations. These stations are designed to be dropped from aircraft to drift with the currents. It is anticipated that we will produce floating weather stations in a variety of configurations to meet different customer needs. In particular, we will produce versions with a small number of sensors that are low cost but low accuracy. We will also develop higher cost versions with more sensors and/or more accurate sensors that produce more accurate results.

Expected changes

Behavior

V 1. The formula used for computing w*ind speed* from the sensor readings may vary. In particular, the weights used for the high resolution and low resolution sensors may vary, and the number of readings of each sensor used (the history of the sensor) may vary.
V2. The format of the messages that an FWS sends may vary.
V3. The *transmission period* of messages from the FWS may vary.
V4. The rate at which sensors are scanned may vary.

Devices

V4. The number and types of *wind speed* sensors on a FWS may vary.
V5. The resolution of the win*d speed* sensors may vary.
V6. The win*d speed* sensor hardware on a FWS may vary.
V7. The transmitter hardware on a FR'S may vary.
V8. The method used by sensors to indicate their reliability may vary.

Elements of Architectural Design

- Business rationale
  - What properties must the system have and why?
- Design goals
  - What are we trying to accomplish in the decomposition?
- Structure
  - What are the components, relations, interfaces?
- Method of communication
  - How are the choices of components, relations and interfaces communicated to the stakeholders?
- Decomposition principles
  - What decomposition principles support the objectives?
  - What do I do next?
  - Am I done yet?
- Evaluation criteria
  - How do I tell a good design from a bad one?

Summary Mapping

Business Rationale	
What properties must the system have?	**1. Support rapid, low cost development of new versions.** **2. Support easy customization**
Design Goals	
What are we trying to accomplish in the decomposition?	**1. Should be easy to understand what parts of the system need to change** **2. Changes should be localized**
Structure	
What are the components, relations, interfaces?	**1. Modules (objects)** **2. Submodule-of** **3. Services, assumptions, secrets of each module**
Method of Communication	
How are the choices of components, relations and interfaces communicated to the stakeholders?	**1. Module Guide Document** **2. Module Interface Specifications**
Decomposition Principle	
What decomposition principles support the objectives?	**1. Information hiding** **2. Abstraction**
Evaluation Criteria	
How do I tell a good design from a bad one?	**Evaluate against expected changes for:** **1. Completeness** **2. Ease of change**

Module Structure Design Goals

- Divide software into set of work assignments with the following properties:
  - *Easy to Understand:* Each module's structure should be simple enough that it can be understood fully.
  - *Easy to Change (mutability):* It should be possible to change the implementation of one module without knowledge of the implementation or affecting the behavior of other modules.
  - *Proportion:* Effort of making a change should be in (reasonably) direct proportion to the likelihood of that change being necessary.
  - *Independence:* It should be possible to make a major change as a set of independent changes to individual modules (except for interface changes)

Modular Structure

- Components
  - Called modules
  - Leaf modules are work assignments
  - Non-leaf modules are the union of their submodules
- Relations (connectors)
  - submodule-of => implements-secrets-of
  - The union of all submodules of a non-terminal module must implement all of the parent module's secrets
  - Constrained to be acyclic tree (hierarchy)
- Interfaces (externally visible component behavior)
  - Defined in terms of access procedures (services or method)
  - Only access to internal state

Method of Communication

Module Guide

- Documents the module *structure*:
  - The set of modules
  - The responsibility of each module in terms of the module's secret
  - The "submodule-of relationship"
  - The rationale for design decisions
- Document purpose(s)
  - Guide for finding the module responsible for some aspect of the system behavior
    - Where to find or put information
    - Determine where changes must occur
  - Baseline design document
  - Provides a record of design decisions (rationale)

Method of Communication

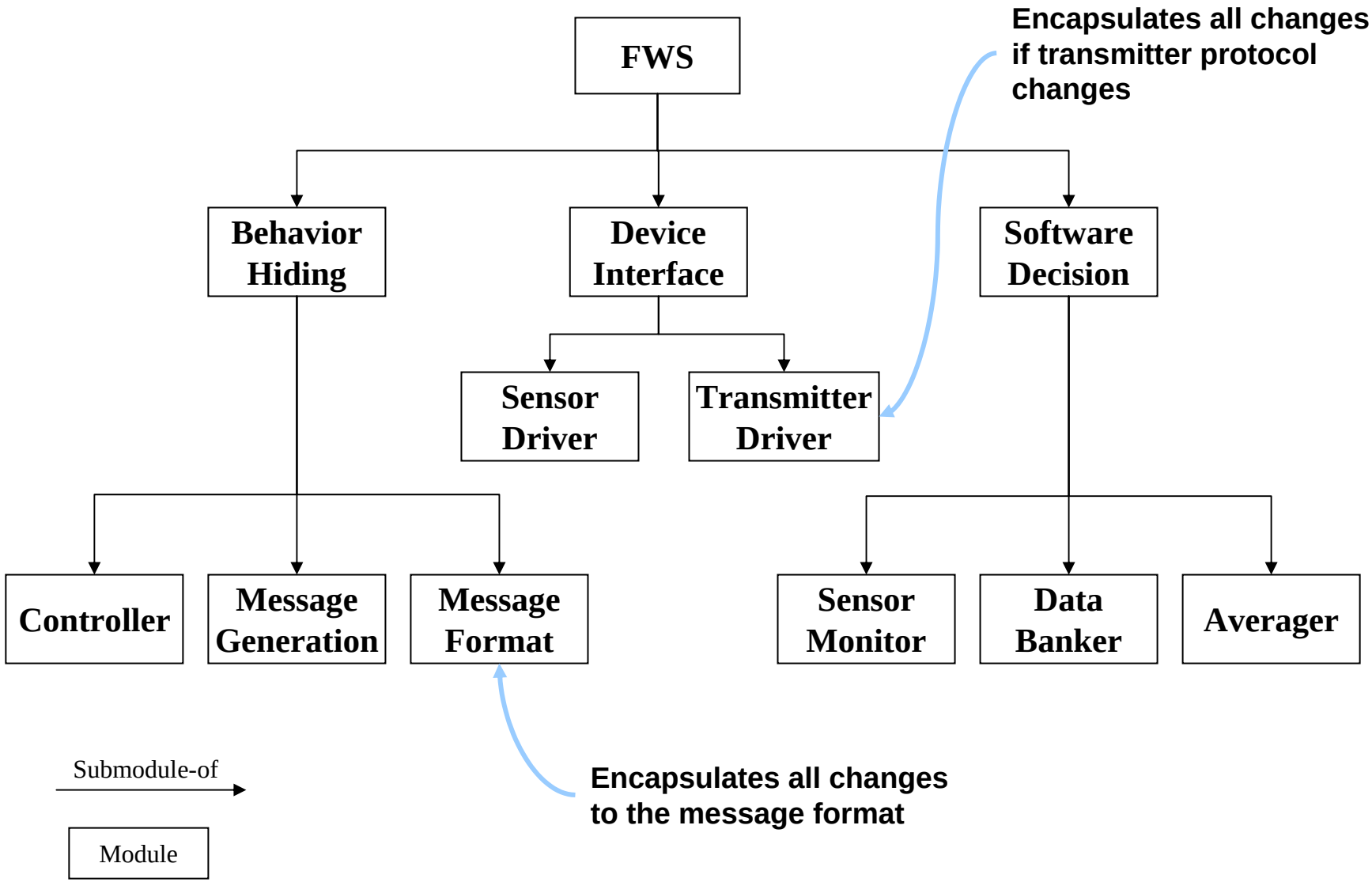
Module Interface Specifications

- Documents all assumptions user's can make about the module's externally visible behavior (of leaf modules)
  - Access programs, events, types, undesired events
  - Design issues, assumptions
- Document purpose(s)
  - Provide all the information needed to write a module's programs or use the programs on a module's interface (programmer's guide, user's guide)
  - Specify required behavior by fully specifying behavior of the module's access programs
  - Define any constraints
  - Define any assumptions
  - Record design decisions

Decomposition Criteria

- Principle: information hiding
  - System details that are likely to change independently should be encapsulated in different modules.
  - The interface of a module reveals only those aspects considered unlikely to change.
- What do I do next?
  - For each module, determine if its secret contains information that is likely to change independently
- Stopping criteria
  - Each module is simple enough to be understood fully
  - Each module is small enough that it makes sense to throw it away rather than re-do.

FWS Modular Structure



Evaluation Criteria

- Completeness
  - Is every aspect of the system the responsibility of one module?
  - Do the submodules of each module partition its secrets?
- Ease change
  - Is each likely change hidden by some module?
  - Are only aspects of the system that are very unlikely to change embedded in the module structure?
  - For each leaf module, are the module's secrets revealed by it's access programs?
- Usability
  - For any given change, can the appropriate module be found using the module guide?

…etc.

Summary

- The critical problem is separating architectural concerns from other development issues
- Choosing an appropriate architectural structure depends on the particular system design goals to be addressed
- Effective deployment of any structure requires appropriate representation, method, principles, and evaluation criteria
- Understanding which structures to apply and how best to apply them requires both training and experience

End