Achieving System Qualities Through Software Architecture

The architectural design process

Quality requirements

Choosing architectural views

Project 2 Proposals

Software Architecture

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

From *Software Architecture in Practice,* Bass, Clements, Kazman

Remember as: ***Components, Interfaces,*** *and* ***Relations***

Effects of Architectural Decisions

- System qualities are founded in architecture
- System run-time properties
  - Performance, Security, Availability, Usability
- System static properties
  - Modifiability, Portability, Reusability, Testability
- Production properties? (effects on project)
  - Work Breakdown Structure, Scheduling, time to market
- Business/Organizational properties?
  - Lifespan, Versioning, Interoperability

Importance to Stakeholders

- There are many interested parties (stakeholders) with many diverse and often conflicting interests
- Important because their interests defy mutual satisfaction
  - There are inherently tradeoffs in most architectural choices
  - E.g. Performance vs. security, initial cost vs. maintainability
- Making successful tradeoffs requires understanding the nature, source and priority of these constraints

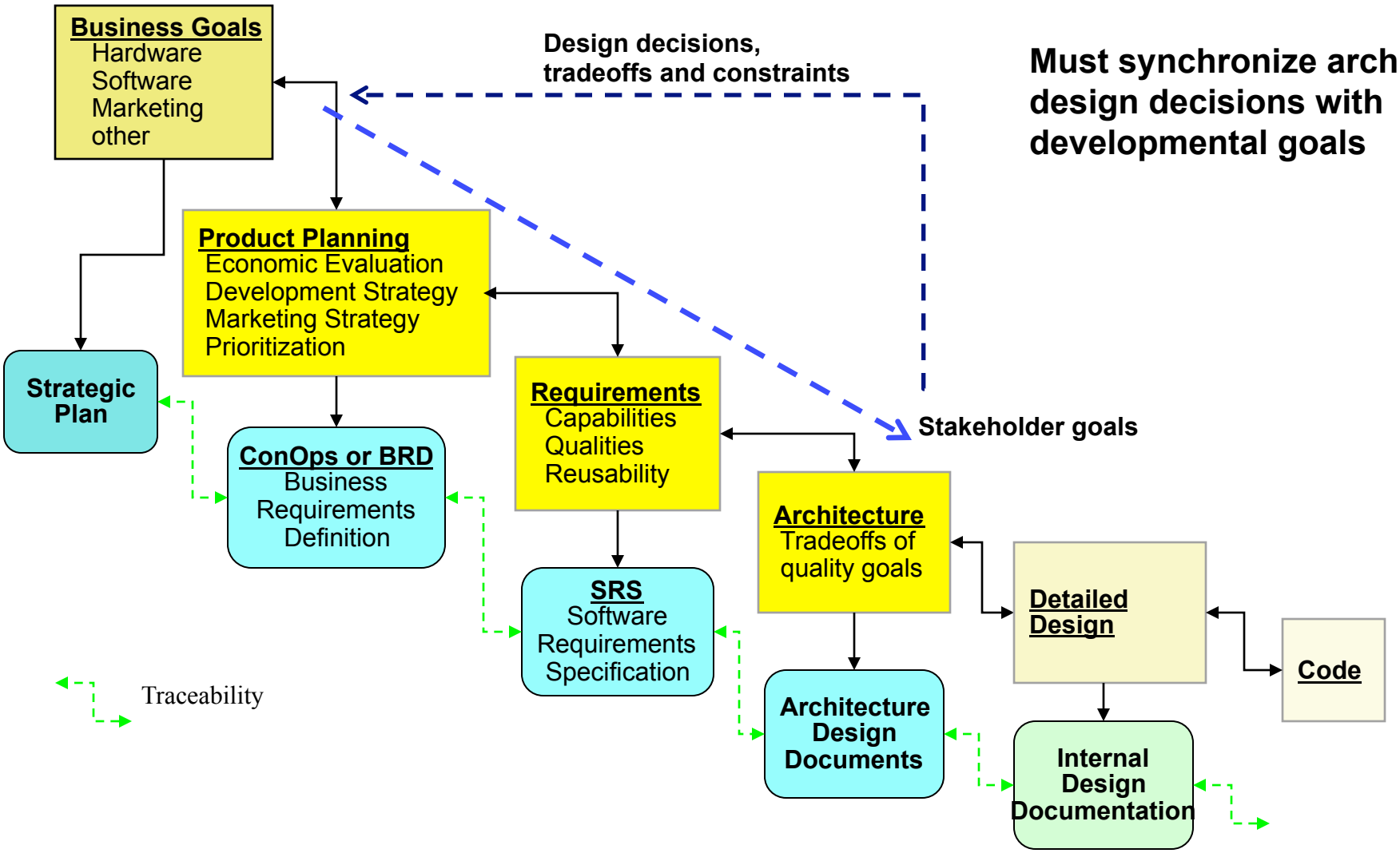
Functionality, Architecture, and Quality Attributes

- Functionality and quality attributes are orthogonal
- Achieving quality attributes must be considered throughout design, implementation, and deployment
- Satisfactory results depends on:
  - Getting the big picture (architecture) right
  - Then getting the details (implementation) right

Example: Performance

- Ex: Performance depends on
  - How much inter-component communication is necessary (Arch)
  - What functionality has been allocated to each component (Arch)
  - How shared resources are allocated (Arch)
  - The choice of algorithms to implement functionality (Non-arch)
  - How algorithms are coded (Non-arch)

Product Development Cycle and Architecture



Software Engineering Architecture

- Goal is to keep developmental goals and architectural capabilities in synch
- Proceed from an understanding of desired qualities to an *acceptable* system design
  - Balance of stakeholder priorities and constraints
  - Requires making design tradeoffs

Implications for the Development Process

Implies need to address architectural concerns in the development process:

- Understand the goals for the system (e.g., business case or mission)
- Understand/communicate the quality requirements
- Design architecture(s) that satisfy quality requirements
- Represent and communicate arch. Design decisions
- Evaluate the architecture
- Implement the system based on the architecture

Quality Requirements

Terminology

- Avoid “functional” and non-functional" classification
- Behavioral Requirements – any information necessary to determine if the run-time behavior of a given implementation constitutes an acceptable system
  - All quantitative constraints on the system's run-time behavior
  - Other objective measures (safety, performance, fault-tolerance)
  - In theory all can be validated by observing the running system and measuring the results
- Developmental Quality Attributes - any constraints on the system's static construction
  - Maintainability, reusability, ease of change (mutability)
  - Measures of these qualities are necessarily relativistic (I.e., in comparison to something else

Behavioral vs. Developmental

Behavioral (observable)

- Performance
- Security
- Availability
- Reliability
- Security
- Usability

Properties resulting from the properties of components, connectors and interfaces that exist at run time.

Developmental Qualities

- Modifiability(ease of change)
- Portability
- Reusability
- Ease of integration
- Understandability
- Independent work assignments

Properties resulting from the properties components, connectors and interfaces that exist at design time *whether or not they have any distinct run-time manifestation*.

Specifying Quality Requirements

- Traditionally done in natural language, usually badly
  - "The system shall be easy to maintain."
  - "The system shall maximize the number of user transactions."
- What's wrong with these?

Specifying Quality Requirements

- When using natural language, write *objectively verifiable* requirements when possible
  - Maintainability: “The following kinds of requirement changes will require changes in no more than one module of the system…”
  - Performance:
    - “System output X has a deadline of 5 ms from the input event.”
    - “System output Y must be updated at a frequency of no less than 20 ms.”
- Consider: what do we really mean by “maintainable?”

Timing Requirements

5.2. TIMING REQUIREMENTS FOR DEMAND FUNCTIONS

For all the demand functions, the rate of demand is so low that it will not constitute a significant CPU-load.

For the starred entries, the desired maximum delay is not known; the entry is the maximum delay in the current OFP, which we will use as an approximation. In one case, both the current and desired values are given. The current value would be good enough to satisfy requirements, but the desired rate would be preferred.

Function name	Maximum delay to completion
IMS:	
Switch AUTOCAL light on/off	*200 ms
Switch computer control on/off	*200 ms
Issue computer failure	not significant
Change scale factor	*200 ms
Switch X slewing on/off	*200 ms
Switch Y slewing on/off	*200 ms
Switch Z slewing on/off	*200 ms
Change latitude-greater-than-70-degrees	*200 ms
Switch INA light on/off	*200 ms
FLR:	
Enable radar cursor	200 ms
Slave or release slave	40 ms

Architectural Views

Architectural Development Process

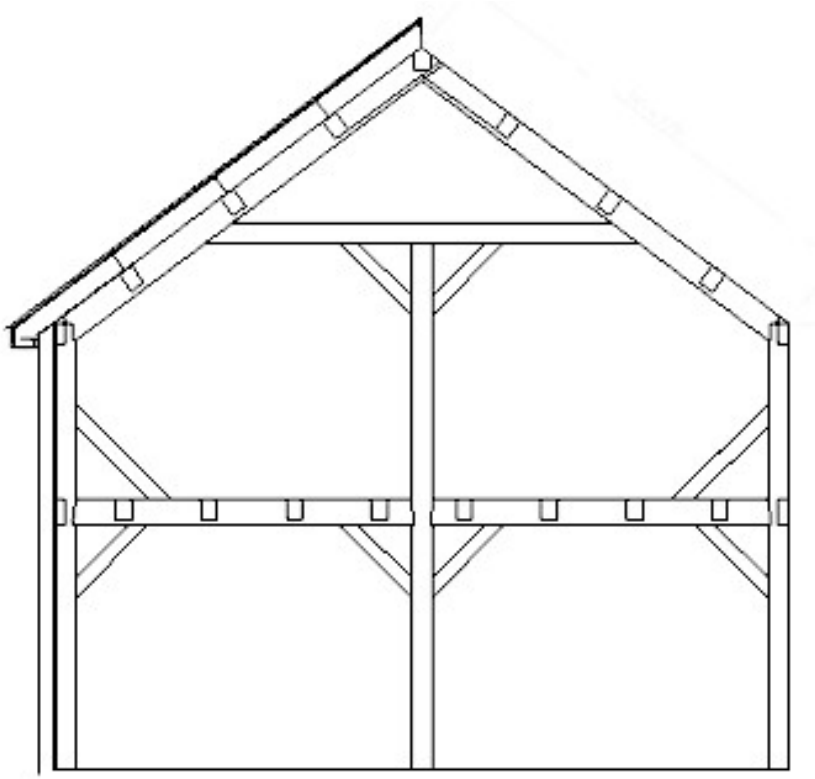
- Understand the goals for the system (e.g., business case or mission)
- Understand/communicate the quality requirements
- Design architecture(s) that satisfy quality requirements
  - Choose appropriate architectural structures
  - Design structures to satisfy qualities
  - Document to communicate design decisions
- Evaluate the architecture
- Implement the system based on the architecture

Which structures should we use?

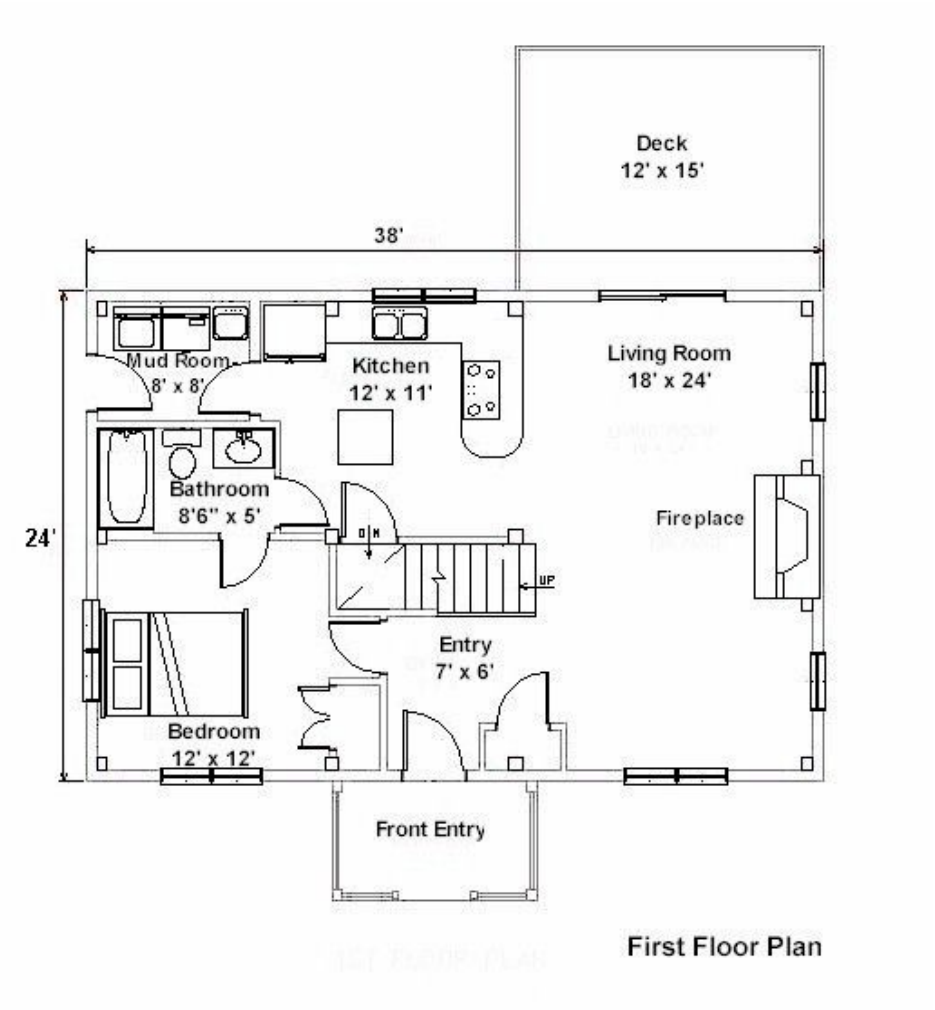
Structure	Components	Interfaces	Relationships
Calls Structure	Programs (methods, services)	Program interface and parameter declarations	Invokes with parameters (A calls B)
Data Flow	Functional tasks	Data types or structures	Sends-data-to
Process	Sequential program (process, thread, task)	Scheduling and synchronization constraints	Runs-concurrently-with, excludes, precedes

- Choice of structure depends the *specific design goals*
- Compare to architectural blueprints
  - Different blueprint for load-bearing structures, electrical, mechanical, plumbing

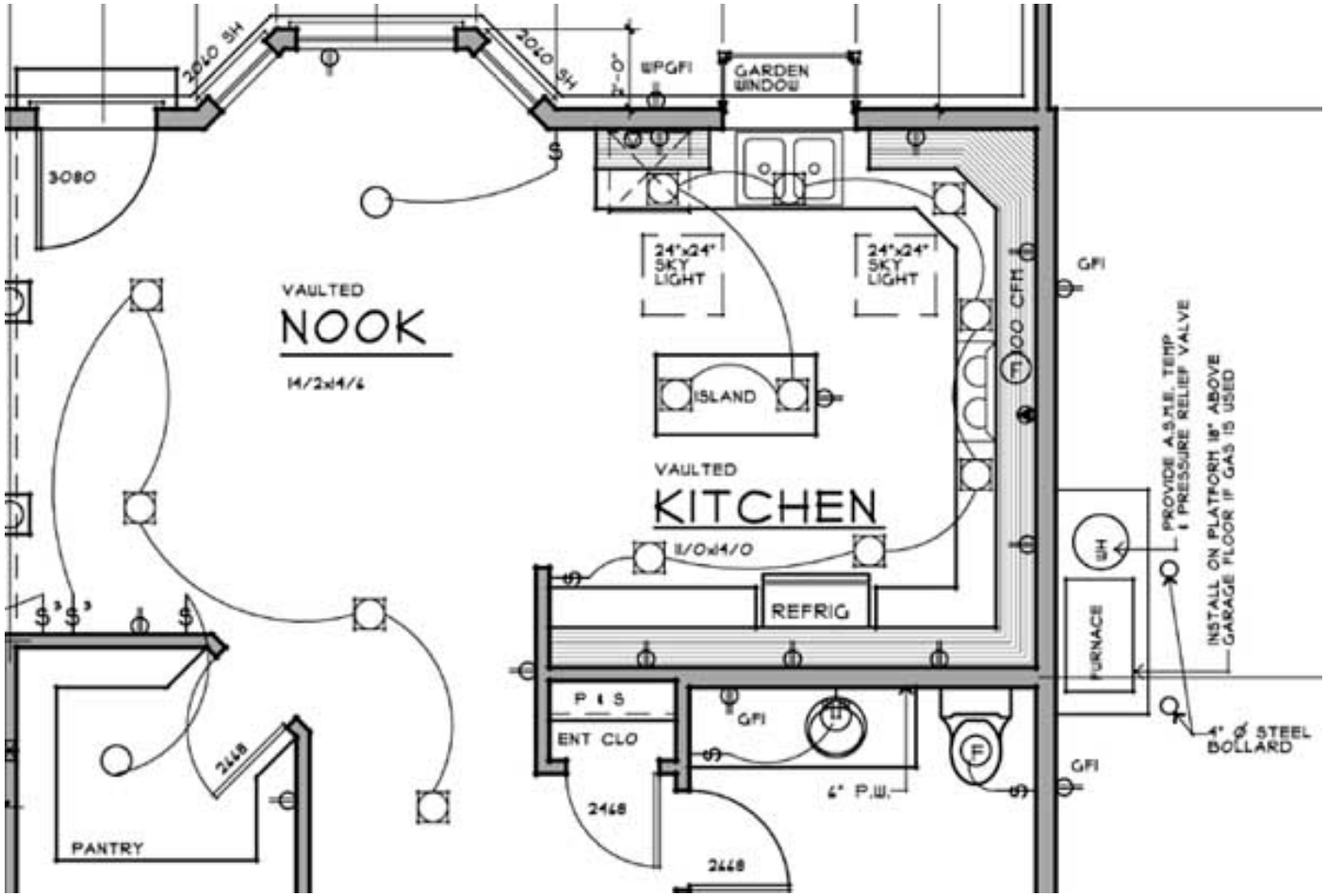
Elevation/Structural



Floor Plan


Deck
12' x 15'
38'
Mud Room
8' x 8'
Kitchen
12' x 11'
Living Room
18' x 24'
Bathroom
8'6" x 5'
24'
Fireplace
UP
Entry
7' x 6'
Bedroom
12' x 12'
Front Entry
First Floor Plan

Electrical Plan



Models/Views

- Each is a view of the same house
- Different views answer different kinds of questions
  - How many electrical outlets are available in the kitchen?
  - What happens if we put a window here?
- Designing for particular software qualities also requires the right architectural model or “view”
  - Any model can present only a subset of system structures and properties
  - Different models allows us to answer different kinds of questions about system properties
  - Need a model that makes the properties of interest and the consequences of design choices visible to the designer, e.g.
    - Process structure for run-time property like performance
    - Module structure for development property like maintainability

For Your Projects

Behavioral Qualities

- Performance
- Security
- Availability
- Reliability
- Security
- Usability

Developmental Qualities

- Modifiability(ease of change)
- Portability
- Reusability
- Ease of integration
- Understandability
- Independent work assignments
- Subsetability

Which qualities are of interest for your projects?
Which structures should you use?

Examples of Key Architectural Structures

- Module Structure
  - Decomposition of the system into work assignments or information hiding modules
  - Most influential design time structure
    - Modifiability, work assignments, maintainability, reusability, understandability, etc.
- Uses Structure
  - Determine which modules may use one another's services
  - Determines subsetability, ease of integration

Questions?