SAPM Lecture 3

Stuart Anderson

What is “good” Architecture

- The architecture is appropriate for the context of use. E.g. a 3-tier e-commerce architecture is not appropriate for an avionics project
- Guidance on “good architecture” focusses on:
  - Process
  - Structure
- Architecture should capture the **principal** design decisions about the system.
- The blueprint – focussing on Structure, Component Behaviour, Component Interaction and how that influences Quality Attributes of Systems

Process

- The architect team is small and maintains the integrity of the architecture
- The architecture is justified in relation to a prioritized list of quality attributes that need to be managed
- Document using views that reflect stakeholder interest
- Evaluate the architecture in terms of how well it delivers the quality attributes
- Choose architectures that allow incremental implementation

Structure

- Use good modular structure: hide information, separate concerns, good robust interfaces that are unlikely to change
- Use well known patterns and tactics (see later) to achieve quality attributes
- Don’t depend on particular versions of tools
- Modules producing data should be separate from those consuming data

Structure

- Don’t expect simple mapping between modules (static structure) and components (dynamic structure)
- Don’t depend on special features in the deployment environment unless essential
- Architecture should use a small number of ways of interaction between components
- Should be clearly identified resource contention issues e.g. if network capacity is a potential issue the architect should budget capacity across components or some other management approach

Importance of Architecture

- Software Architecture:
  - Enables us to manage the key attributes of a system
  - Allows reasoning about and managing change
  - Allows prediction of the key quality attributes
  - Allows better communication among stakeholders
  - Carries the earliest (most fundamental) design decisions
  - Defines constraints on implementation

Importance of Architecture

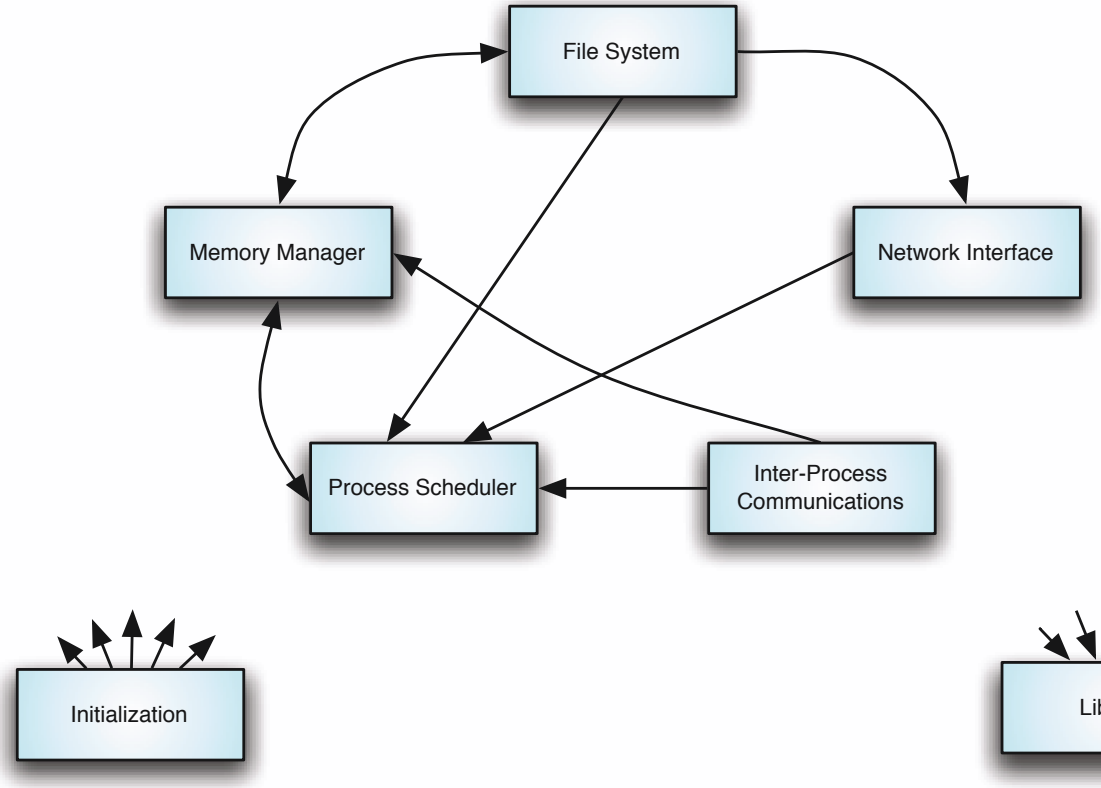
- Software Architecture:
  - Reflects the structure of an organisation
  - Provides the basis for evolutionary prototyping
  - Is the key artifact in reasoning about cost and scheduling
  - Can be used as the transferrable, reuseable model at the heart of a product line
  - Focusses on the assembly of components rather than on the creation of the components
  - Restricts design alternatives and channels developer effort in a coordinated way
  - Provides the basis for training new team members.

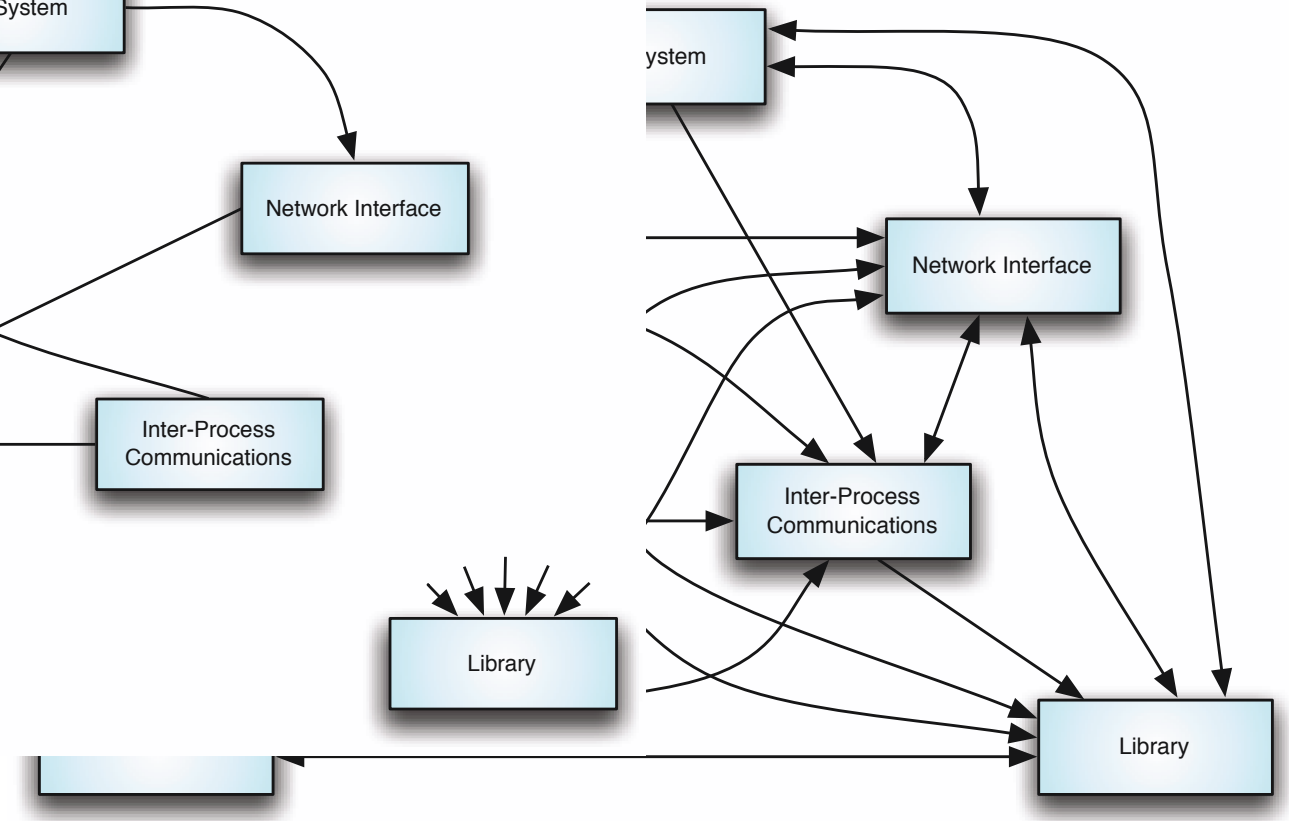
Structure

- Structure is slippery to describe
- Sometimes we want to be **prescriptive** (this is how it should be) – often too tidy
- Sometimes we want to be **descriptive** (this is how it is) – often a mess.

What about “Real” Examples?

Linux – Prescriptive Architecture





Linux – Descriptive Architecture

Managing Attributes

- Enables us to manage the key attributes of a system:
  - If performance is important we need to manage timing behaviour
  - If modifiability is important it is important to try to localise functionality so that modification is localised to change a function.
  - If managing project risk is important it is important to have clear responsibility for delivery of components and good traceability

Managing Change

- Allows reasoning about and managing change:
  - We can see three levels:
    - Inside an element
    - Between elements maintaining the architecture
    - Requiring architectural change
  - These distinctions help in assessing the cost of change
  - We can identify likely changes and try to architect to minimize architectural change

Prediction of Attributes

- Allows prediction of the key quality attributes
  - Should be able to build models on the basis of the architecture that prdict key attributes
  - E.g. if we want a system to be scalable:
    - the architecture should include:
      - capacity information,
      - dynamic component creation, and
      - dynamic deployment characteristics
    - These should be sufficiently detailed to be able to predict how increases in demand will be serviced

Improves Communication

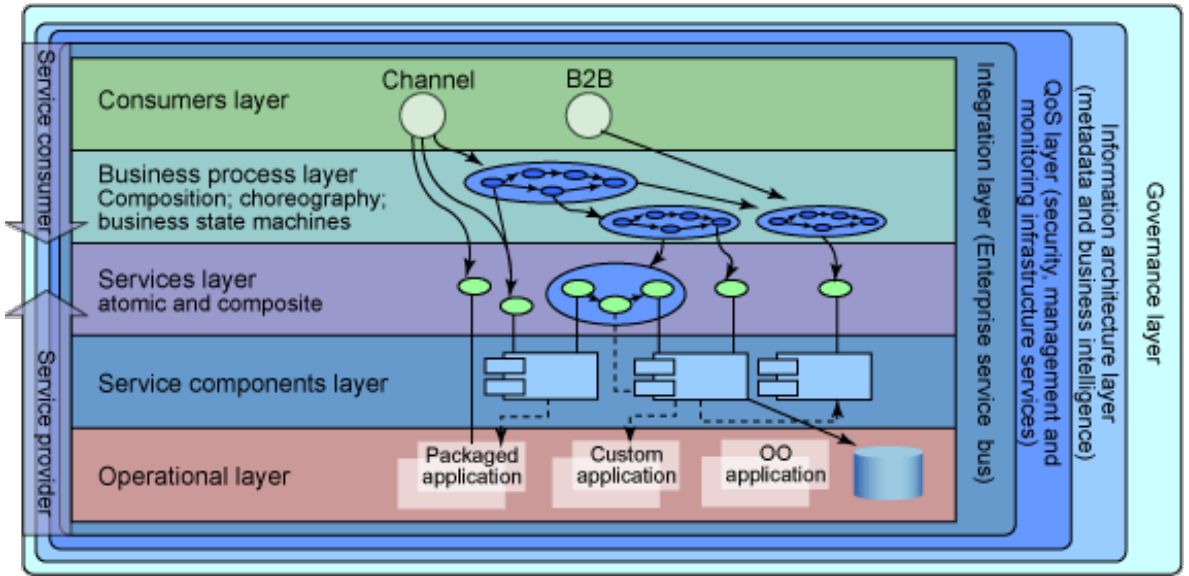
- Allows better communication among stakeholders:
  - User has particular requirements in terms of user experience
  - Customer needs to know about schedule, budget, meeting regulation in their market
  - Project manager needs to know the depndencies in terms of modules and components
- These might be accommodated by different views of the system that are consistent

Early Design

- Carries the earliest (most fundamental) design decisions:
  - What are the key quality attributes?
  - What architecture gives us control over thses attributes?
  - How do we characterise the behaviour of architecture elements?
  - E.g. suppose the attribute is that we want to know whether the system is faulty what does this say about architecture?

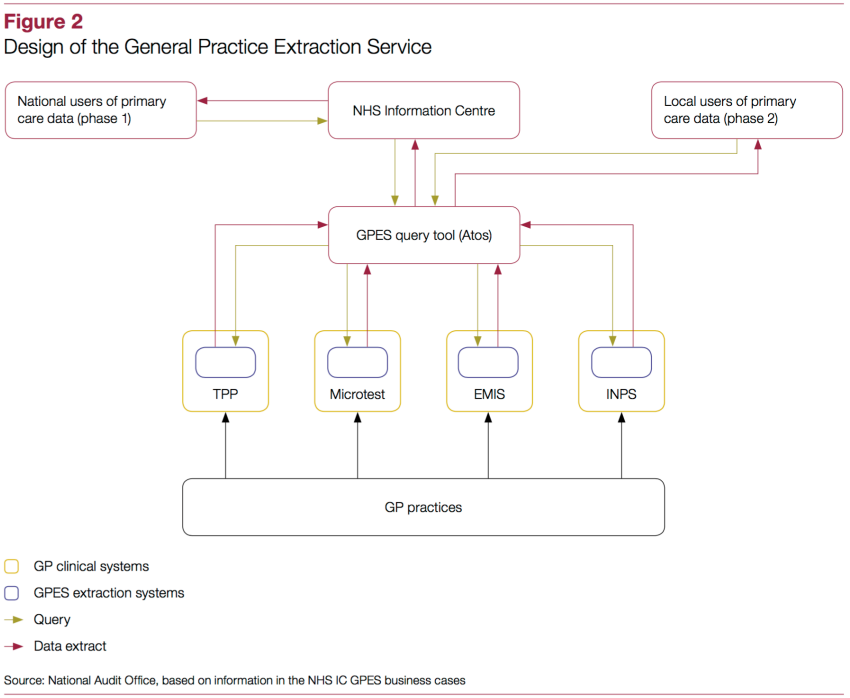
Constraints

- Defines constraints on implementation:
  - Architecture specifies the elements and their interaction
  - For example, layered architecture usually constrain access to be between adjacent layers



Structure of Organisation

- Reflects the structure of an organisation:
  - Of the development organisation.
  - Of the overall organisation
  - The work organisation shapes the architecture
  - The architecture shapes the work organisation.



Evolutionary Prototyping

- Provides the basis for evolutionary prototyping:
  - For example:
    - Plug and play – early experience of the base functionality + extensibility
    - Real time architectures – early experience with scheduling – worst case execution times guide design, and deployment

Cost and Scheduling

- Is the key artifact in reasoning about cost and scheduling:
  - Capture dependencies
  - Estimate required efforts etc
  - Allocate effort to elements
  - Understand how elements influence each other
  - Use architecture to interpret bottom-up estimates from teams working on elements

Product Line

- Can be used as the transferrable, reuseable model at the heart of a product line
  - Elements are assets that can copmpose to give new functionality
  - Architecture provides the means to compose the elements
  - Planned approach to the reuse of architectural elements

Component Level

- Focusses on the assembly of components rather than on the creation of the components:
  - With well designed elements and architecture we can combine elements from different producers provided they conform to interface standards. Brings benefits:
    - Decrease time to market
    - More reliability
    - Lower cost
    - Flexibility e.g. using multiple or alternate suppliers for a component.

Channels Development

- Restricts design alternatives and channels developer effort in a coordinated way
  - Provides a defined context for the developer
  - Well defined interfaces and clear ide of the functionality and the quality attributes required
  - Clarity on what is an architectural decision and what is a development decision (e.g if fault tolerance is important we don't want to do that inside a module) we should use architecture.

Training Resource

- Provides the basis for training new team members:
  - Multiple views provide different stakeholder perspectives
  - Records interactions between elements
  - Abstracts from detail to provide an overview

Summary

- This is a long list of good things we get from software architecture.
- This is an overview of some of the potential benefits of architecture.
- We will dig a bit deeper in subsequent weeks