UNIVERSITY OF TECHNOLOGY SYDNEY

EXECUTION ARCHITECTURE

Tom McBride

UTS:
ENGINEERING AND INFORMATION TECHNOLOGY

Contents

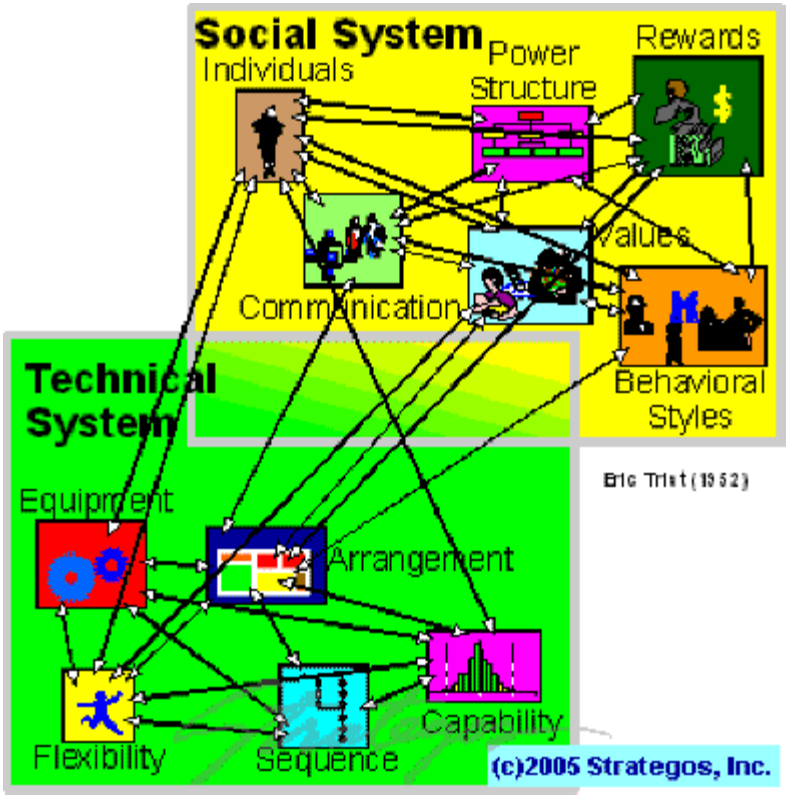
- What is execution architecture?
- What does an execution architecture enable
- How are the elements identified
- Notations
- Aggregating objects into components
- Evaluating an execution architecture

What you already know

- The architecture context
  - Functional requirements
  - Quality Requirements
  - Constraints
  - Risks
- Conceptual architecture
  - Model of the problem (or proposed solution), unencumbered by implementation concerns

An execution architecture

- Transfer the theoretical model to a computer based system within the realities and constraints of computers, software components, hardware components, networks, interface devices.
- Try to preserve the essential characteristics of the theoretical model – those essential to realizing the solution's benefits.



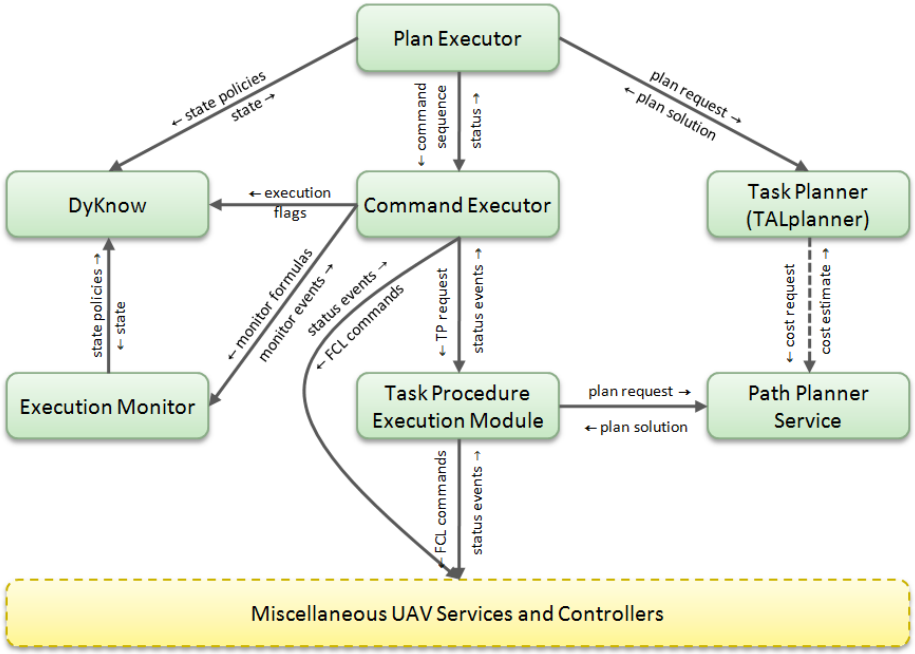
WHAT IS AN EXECUTION ARCHITECTURE

What is the execution architecture?

- An execution architecture models how the system could be implemented as a computer-based system
- An architecture that captures components that exist during run time
  - Subsystems
  - Processes
  - Threads
- An execution architecture captures the dynamics of the system
  - Communication between sub-systems and between components
  - How does information move about the system

From conceptual to execution

- Aggregate the objects into coherent components
- Decide which components will be COTS and which will be developed
- Determine communications among the components



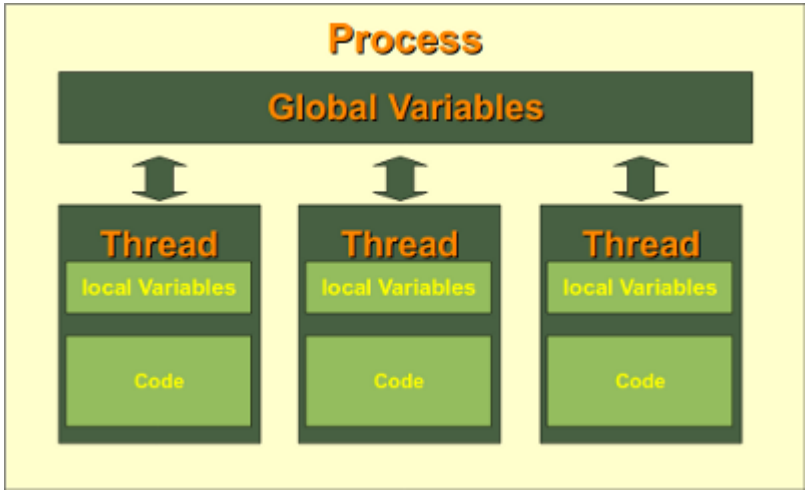
Designing for non-functional characteristics

- The conceptual architecture ignored non-functional requirements and implementation constraints.
- Now it is time to deal with them.
- Which are the more important qualities and how can they be realized
  - Usability vs Security
  - Performance vs Modifiability & testability
  - Availability vs Performance
  - Scalability vs Security

ELEMENTS OF AN EXECUTION ARCHITECTURE

Concurrent activity is the basic architecture component

- The basic component of an execution architecture is the concurrent activity
  - Concurrent subsystems – large-grain units of execution that perform a complete set of related functions
  - Processes, which have their own memory space and are resident on a single processing node
  - Threads, share the memory space of a single process



Concurrent subsystems

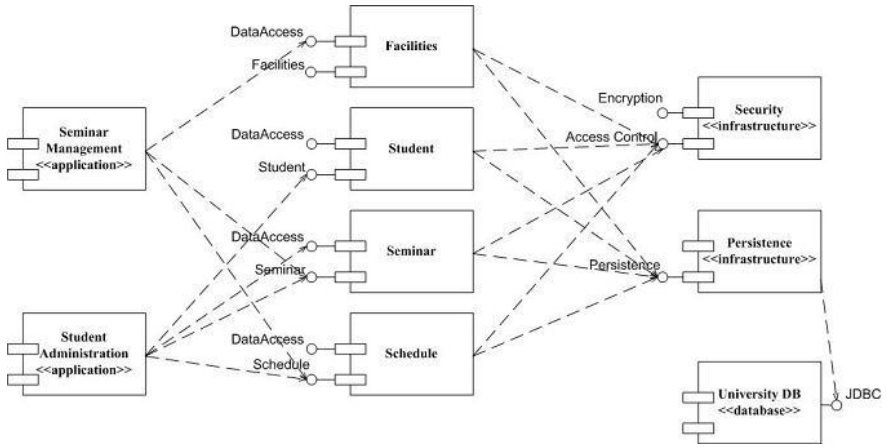
- Subdivide the system into a manageable number of subsystems that must execute concurrently
  - A large number of similar processes can be dealt with more easily if they are treated as a unit. For example, a “data acquisition subsystem” rather than the numerous processes that comprise it.
  - A process that has a high degree of internal concurrency. For example, a database server
  - Existing systems are often best treated as a concurrent subsystem
- Concurrent subsystems are always long lived.
- Modern systems tend toward the concurrent subsystem style.

Examples of concurrent sub-systems

- In general, if you can put it on a separate server, it is a subsystem
  - Mail server
  - Database server
  - Domain name server
  - User interface
  - Business processes

Process models

- Useful to show the “actual” execution structure of the system
- Processes have their own resources
- Does not have lower level processes
- Is not spread across compute nodes
- Corresponds well with the conceptual architecture
- Can be complex interactions between processes



Examples of processes

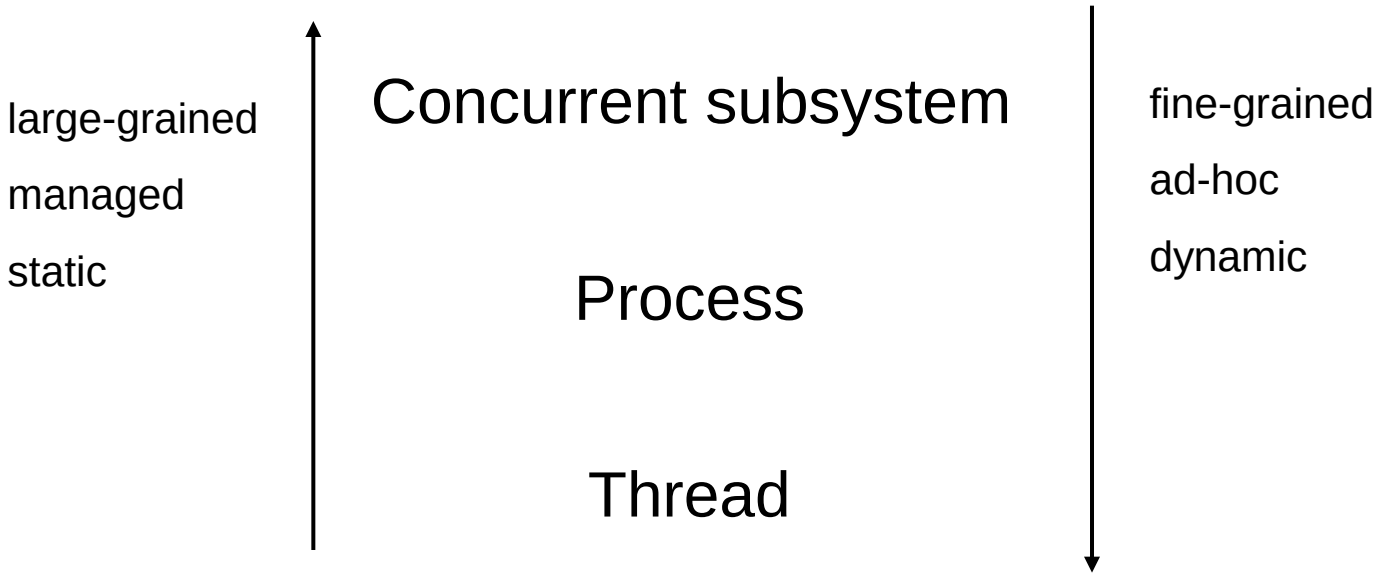
- Business processes – usually executing as a single process
- Single applications (usually)
- Client applications (usually)

Threads

- Threads are a single execution pass through a common process
- Execute concurrently within the memory space of a parent process
- So, a process may have threads and threads are contained in a process
- No particular notation for threads

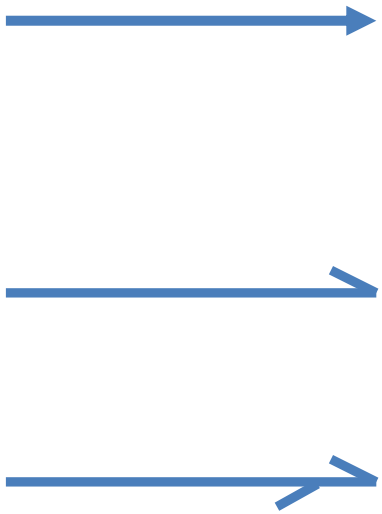
- Note. Experienced system developers advise against threads if they can be avoided. They are difficult to implement and get right.

Characteristics of concurrent components



Communication among architecture components

- Explains how components communicate
  - Synchronous, where caller waits for a response
  - Asynchronous, where no response is required
  - Callback, caller sets the means to receive a response but does not wait for it



Communication speed among components

Communication between	Relative speed
Within a process	Relatively fast.
Between processes on the same server	Slower
Between subsystems	Comparatively slow

Summary

- An execution architecture has concurrent subsystems, processes and threads
- Objects are aggregated into concurrent subsystems and processes

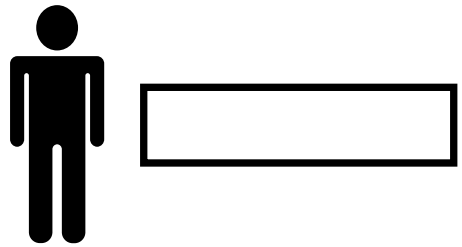
- Very important for real-time and embedded systems

NOTATIONS

Execution stereotypes

- Concurrent activities behave in different ways
- Useful to discriminate between them through their initiation and response
- User-initiated
  - The component performs some action because of input from a user.
  - Usually involves concurrent activity e.g. redraw the user interface and communicate with other concurrent components to perform the user requested action
- Active
  - Usually continuous loop awaiting inputs or interrupts
  - Inputs or interrupts rigger events
  - Initiates other concurrent activity while continuing monitoring
  - Usually in real-time systems
- Services
  - Provides a service on request.
  - Often best considered as concurrent sub-systems
  - Examples: databases, web servers, directory servers
- A component may have more than one stereotype
- Use the stereotype that best captures the component's primary purpose

Stereotype notation



User initiated

Active

Service

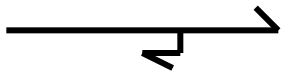
- This notation is not mandatory. Different drawing tools have different shapes available.
- Use anything that clearly distinguishes the different stereotypes

Diagramming conventions

Component

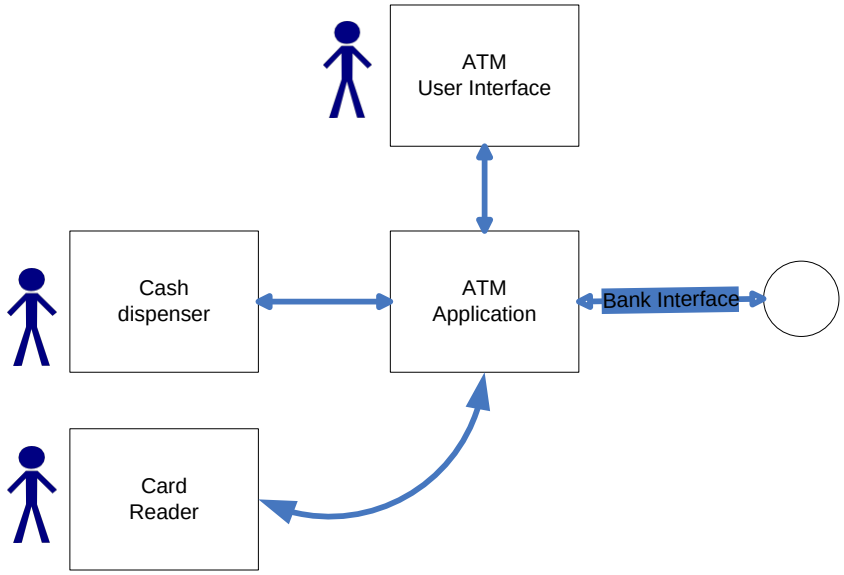
Synchronous call

Asynchronous call



Callback

ATM execution architecture



Summary

- Notations to show
  - User initiated activity
  - Continuous activities
  - Services
- Notations to show synchronisation dependencies

COMPONENTS IN EXECUTION ARCHITECTURES

Identifying concurrent components

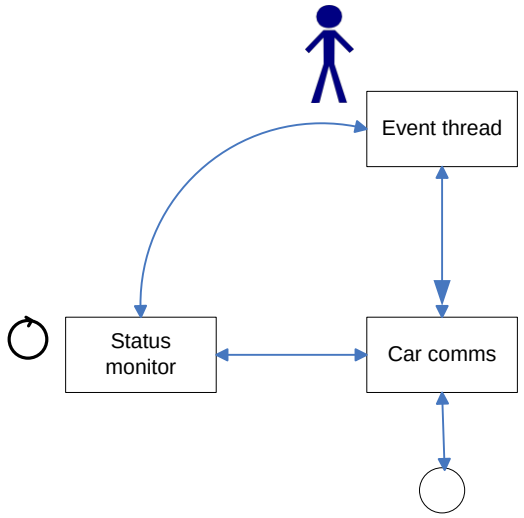
- First discover where concurrency may be required without regard to whether they should be subsystems, processes or threads
- Mostly a concern for real-time systems
- Try to establish what needs to be happening at the same time.
- From quality attributes
  - Components with high reliability or high criticality needs should not be mixed with components that have neither requirement
  - Real-time responsibilities should not be mixed with non-real-time requirements
  - Practical constraints. Need for separate development or testing can indicate the need for separate components.

What indicates a need for concurrent activity?

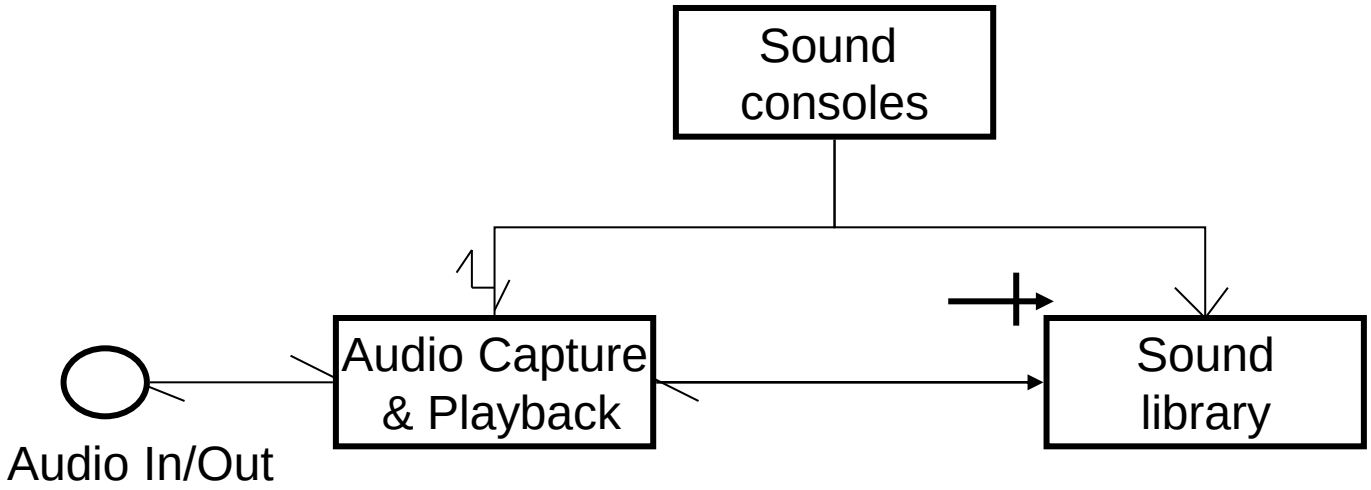
- Forks in a use-case map generally indicate a need for concurrent activity
- Components that perform significant amounts of work. May need to scale up through additional servers or processes
- Components that have to wait for “real world” inputs.
- Components known to require separate hardware

Concurrency in the model car controller

- Components are
  - user interface,
  - status monitor,
  - car communications
- Threads used for all concurrent components



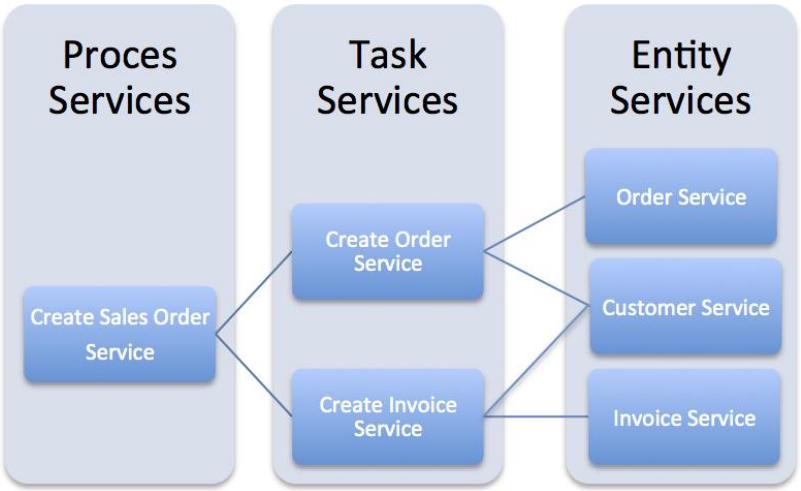
Example execution architecture for digital audio system



- Sound consoles handle user actions, sending requests to the Audio component and the Sound Library, both of which respond
- Sound library provides services in response to requests synchronously
- Audio capture/playback is an active component that monitors and responds to user requests asynchronously

Concurrent components arising from non-functional requirements

- Horizontal scalability is achieved with components.
- Security subsystem usually a candidate for an independent component.
- Modification, maintenance and release (devops) is pushing toward an architecture with independent components.



Why you shouldn't create concurrent components

- Performance liabilities
  - Context switching between concurrent activities is significantly more expensive than a function call.
- Reliability and maintainability concerns
  - generally dictate that concurrency should not be used indiscriminately
  - Where possible, concurrency should be encapsulated into components with clearly specified usage policies

Summary

- Conceptual architecture shows what the system can do and which part of the system does it.
- An execution architecture captures the dynamics of the system
  - How does the system behave
  - How is the system controlled
  - How does information move about the system
- Execution architecture allows you to reason about dynamic behaviour

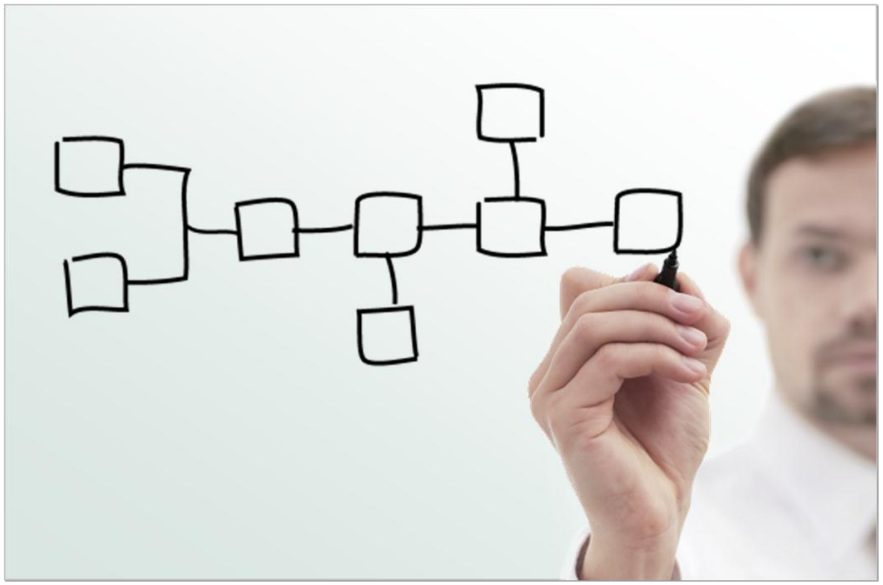
EXECUTION ARCHITECTURE EVALUATION

What can be evaluated

- Functionality was evaluated with the conceptual architecture.
- Has the transformation to a computer system compromised any of the functionality?
- Does the execution architecture provide the best tradeoff among the required non-functional objectives?

Scenario based evaluation

- Hard to beat a good selection of scenarios
  - Scenarios for normal functions
  - Scenarios for abnormal functions
  - Scenarios for tradeoffs among non-functional requirements



Evaluation methods

- Functional walk-through
  - E.g. CRC card walk-through
- ATAM
  - Also requires scenarios but concentrates on critical alternatives
- HAZOP
  - Rigorous and thorough