Principles of Software Architecture

Techniques of Software Architecture

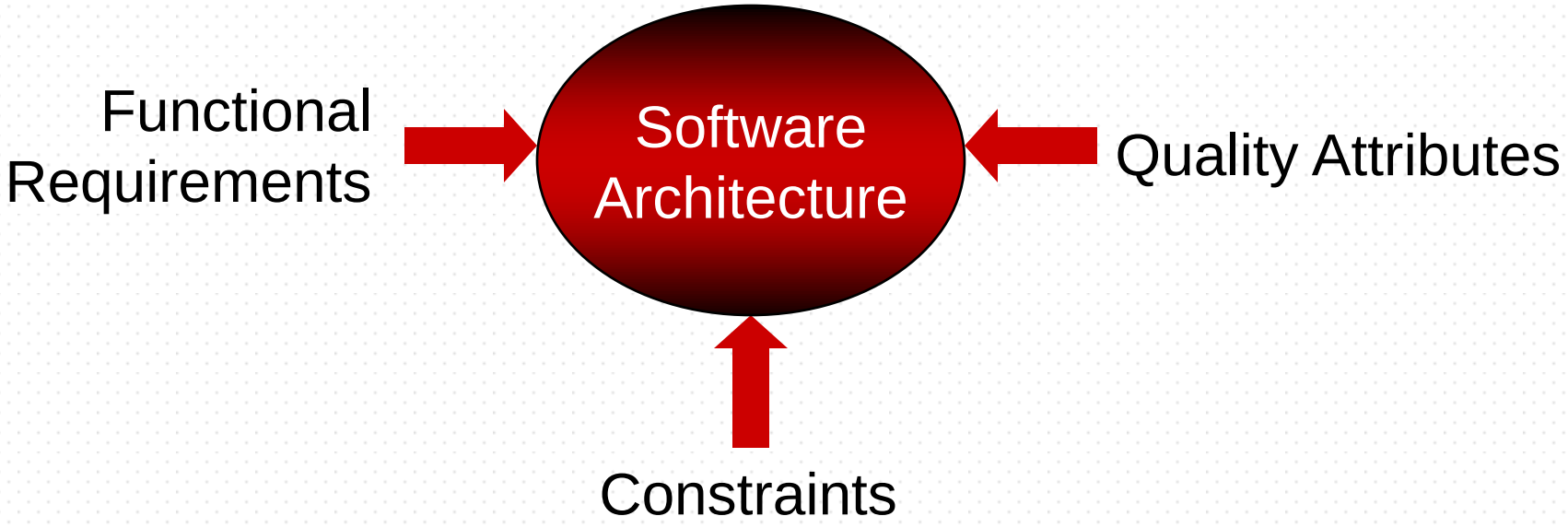
David Garlan, Carnegie Mellon University

Techniques

1. Architectural drivers
2. Evaluation of architectures
3. Documentation
4. Product lines

Architectural Drivers – 1

- *Architectural drivers* are requirements that will shape the software architecture
- Architectural drivers include:



Architectural Drivers – 2

- Functional Requirements – what the system must do.
  - At design time, we are concerned with high level function not implementation details.
- Constraints – design decisions already made for the designers.
- Quality Attributes – characteristics that the system must possess in addition to the functionality.

Functional Requirements

- *What* a system must compute
  - Describes the core functionality of the system from a user's perspective
  - Avoids details of implementation
- Usually available as part of a standard requirements specification for a product
  - However, some functional requirements may be "hidden"
  - Examples: error behavior

Constraints

- Describes the context
  - Decisions that are already made about the design
  - Factors that influence the design
- Categories
  - Technical constraints, such as platform, hardware, …
  - Business constraints, such as time to market, amount of human resources available, …

Quality Attributes - 1

- Quality Attributes define the essential qualities that a system should have beyond its functionality
- Examples
  - availability
  - modifiability
  - performance
  - security

Quality Attributes – 2

- Functionality and quality attributes are largely orthogonal.
  - Two systems can have different structures and provide the same function.
  - Each will support different quality attributes dependent upon their structure.
- Systems are decomposed to achieve a variety of things other than functionality.

Quality Attributes – 3

- Quality attributes must be designed into the system.
  - The achievement of various quality attributes is dependent upon early architecture design decisions.
  - In general, you can't implement the functionality, and then go back and add the quality attributes.
    - Classic examples in industry include retro-designing for security and performance.

Quality Attributes & Architecture

- Architectural choices implement functionality, promote some quality attributes, and inhibit others.
  - A change in structure that improves one quality attribute may promote or inhibit the achievement of others.
    - these are tradeoffs
  - Architecture is critical to balancing *quality attribute tradeoffs* before detailed design and implementation.

Describing Quality Attributes – 1

- Quality attributes are the most difficult architectural drivers to discover and document.
- Names of quality attributes alone are ***not*** enough.
  - there is no widely accepted standard – *terminology varies widely*
  - descriptions usually are vague, lacking measures to evaluate achievement

Describing Quality Attributes – 2

- Stakeholders may have different interpretations of "ilities".
- Relationships between quality attributes can be complex.
  - System failure can be an aspect of availability, security, and usability.
  - Simple textbook descriptions of quality attributes are usually too naive in practice.

Describing Quality Attributes – 3

- Consider the requirement:

  "The system shall be modifiable."

- This is a meaningless requirement.
  - You cannot design a system that is modifiable for all possible changes.
  - Every system is modifiable with respect to *some* set of changes and not with respect to some *other* set of changes.
  - What inhibits change is cost and time.

Describing Quality Attributes – 4

- The real issue is more than modifiability - How can we make the time and cost of change acceptable?
  - The first step is to quantify the quality attributes.
  - Next we need to prioritize them.
  - Only then is it possible to design an architecture to balance all of the architectural drivers.

Describing Quality Attributes – 5

- Software architecture affects most qualities, but not all aspects of all qualities.
- Consider *usability*.
  - Choice of radio buttons, dialog boxes, or command lines affects usability, but these decisions are not architectural.
  - Isolating those decisions so the user interface is interchangeable is architectural.

Capturing Quality Attribute Requirements

- It is common practice today to capture functional requirements using *Use Cases.*
- A solution to the problem of describing quality attributes is to use *Quality Attribute Scenarios* as a means to better characterize quality attributes.

Quality Attribute Scenarios

- A quality attribute scenario is a short story that describes a system's quality attribute response to a specific stimulus.
- Example:

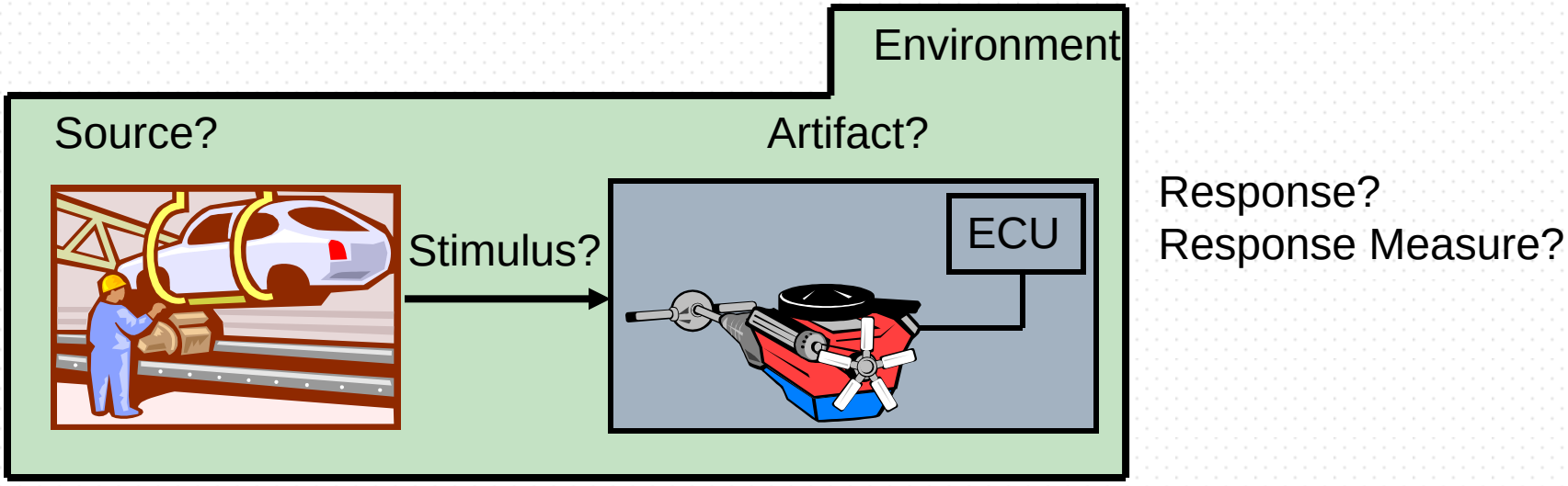
  "An external customer needs a specific power train coordination model. In a normal production environment, our software architecture is able to accommodate the customer's specifications within a 12 month development cycle and with no changes to the fundamental architectural framework."

Six Parts of a Quality Attribute Scenario – 1

1. **Stimulus** – A condition that affects the system.
2. **Source of the stimulus** – The entity that generated the stimulus.
3. **Environment** – The condition under which the stimulus occurred.
4. **Artifact stimulated** – The artifact that was stimulated by the stimulus.
5. **Response** – The activity that results because of the stimulus.
6. **Response measure** – The measure by which the system's response will be evaluated.

Six Parts of a Quality Attribute Scenario – 2

"An external customer needs a specific power train coordination model. In a normal production environment, our software architecture is able to accommodate the customer's specifications within a 12 month development cycle and with no changes to the fundamental architectural framework."



Availability – 1

- Definition: *Availability is concerned with system failure and its associated consequences. A system failure occurs when a system no longer delivers a service that is consistent with its specification.*

Availability – 2

- Areas of concern include
  - preventing catastrophic system failures
  - detecting system failures
  - recovering successfully from system failures
  - the amount of time needed to recover from system failures
  - the frequency of system failures
  - degraded modes of operation due to system failures

Availability – 3

Example Stimuli

- omission, crash, timing, events, response

Example Sources

- internal, external, software, hardware

Example Artifacts

- systems, processors, software, hardware, storage, networks

Example Environments

- operational, test, development, load, quiescence

Example Responses

- detect and notify, record, disable, be unavailable, continue operation in degraded mode

Example Response Measures

- critical time intervals when the system must be available
- availability time
- time intervals in which the system can operate in a degraded mode
- repair time
- response time

Availability – 4

- Example Availability Scenario:

  "An unanticipated oil low signal is received by the initialization process during engine startup. The initialization process prevents an engine start and illuminates the check engine light in 1 second."

Availability – 5

Example Stimuli

- omission, crash, timing, **event**, response

Example Sources

- external, **internal**, software, hardware

Example Artifacts

- systems, processors, **software**, hardware, storage, networks

Example Environments

- test, **operational**, development, load, quiescence

Example Responses

- detect and notify, record, **disable, be unavailable**, continue operation in degraded mode

Example Response Measures

- critical time intervals when the system must be available
- availability time
- time intervals in which the system can operate in a degraded mode
- repair time
- **response time**

Modifiability – 1

- Definition: *Modifiability is about the cost of change and refers to the ease with which a software system can accommodate changes.*

Modifiability – 2

- Areas of concern include
  - Identifying what can change.
    - functions, platforms, hardware, operating systems, middleware, systems it must operate with, protocols, and so forth
    - quality attributes: performance, reliability, future modifiability, and so forth
  - When will the change be made and who will make it.

Modifiability – 3

Example Stimuli

- add/delete/modify functionality or quality attribute

Example Sources

- stakeholders: developer, system administrator, end user,..

Example Artifacts

- systems, OS, hardware, software

Example Environments

- runtime, compile time, build time, design time

Example Responses

- locate places to be modified
- make modifications without side affects
- test the modification with minimal effort
- deploy the modification with minimal effort

Example Response Measures

- cost in terms affected, components, effort, and money
- extent to which this modification affects other functions and/or quality attributes

Modifiability – 4

- Example Modifiability Scenario:

  "The existing engine control processor used for 4 cylinder motors must also be used for 6 and 8 cylinder motors. The engine control software is able to accommodate the use of duel processors with no changes to the engine controller's base source code."

Performance – 1

- Definition: *Performance is about timing and how long it takes the system to respond when an event occurs.*

Performance – 2

- Areas of concern include
  - varying sources of events: interrupts, messages, requests from users, transactions, and so forth
  - varying arrival rates and patterns: sporadic, periodic, stochastic, or some combination
  - response time: elapsed time from event arrival to system reaction

Performance – 3

Example Stimuli

- periodic, sporadic, or stochastic event arrival

Example Sources

- internal, external, software, hardware

Example Artifacts

- systems, OS, hardware, software

Example Environments

- operational, test, development, load, quiescence

Example Responses

- processes stimuli

Example Response Measures

- latency, deadline, throughput, jitter, miss rate, data loss

Performance – 4

- Example Performance Scenario:

  "500 users initiate 1,000 transactions per minute stochastically under normal operating conditions and each transaction is processed with an average latency of two seconds."

References

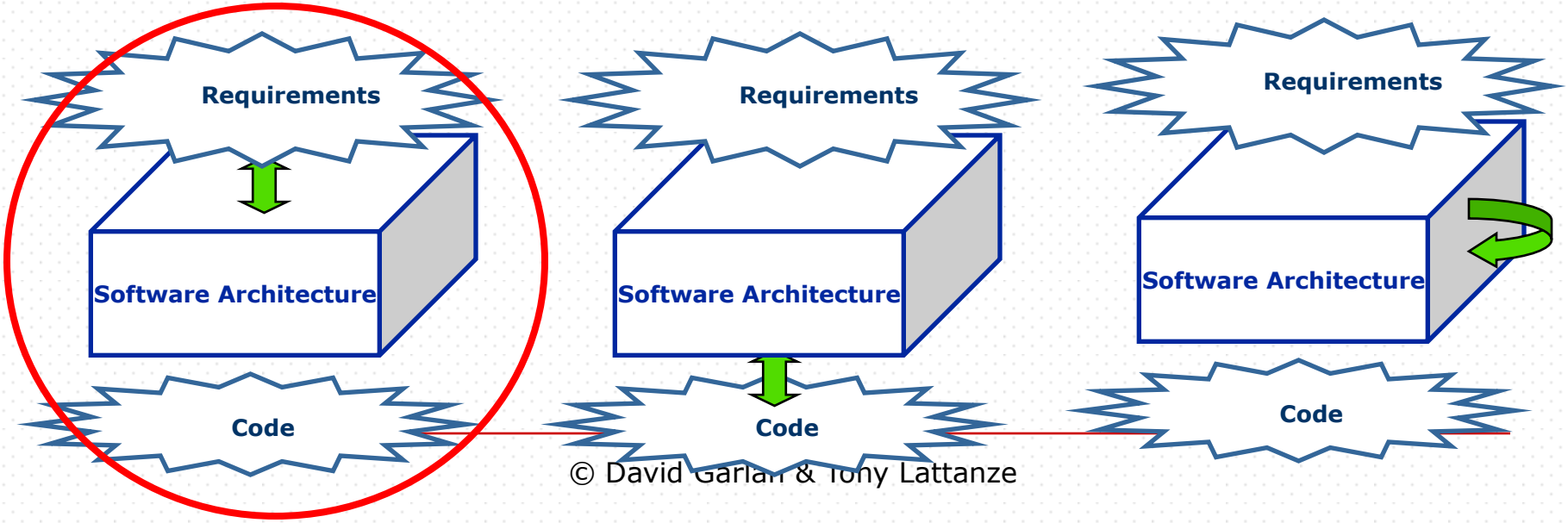
- Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.
- Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; Wood, W. *Quality Attribute Workshops (QAWs), Third Edition.* Technical Report CMU/SEI-2003-TR-016: Software Engineering Institute, 2003.

Techniques

1. Architectural drivers
2. Evaluation of architectures
3. Documentation
4. Product lines

Architecture Evaluation

- An architectural design must bridge the gap between requirements and implementations
- Three important relationships
  - between requirements and architecture
  - between architecture and implementations
  - between alternative architectures



Benefits of Architecture Reviews

- "In our experience, the average [architecture] review pays back at least twelve times its cost."

  [Daniel Starr and Gus Zimmerman, *STQE Magazine*, July/August 2002]

- "We estimate that projects of 100,000 non-commentary source lines of code have saved an average of US$1 million each by identifying and resolving problems early."

  [Maranzano et al, *IEEE Software*, March/April 2005]

- Beneficial side effects:
  - Cross-organizational learning is enhanced
  - Architectural reviews get management attention without personal retribution
  - Architectural reviews assist organizational change
  - Greater opportunities exist to find different defects in integration and system tests. [Maranzano et al, *IEEE Software*, 2005]

Benefits of Architecture Reviews

"Project teams found that the preparation for the review **helped them get a clearer understanding of their projects**. In several reviews the people on the project team asked more questions than the reviewers. Often, this was one of the few opportunities for the project team members to have in-depth discussions of the technical issues about the project. The review served as a catalyst to bring them together."

AT&T: "Best Current Practices: Software Architecture Validation", 1991.

"The architecture review process has **helped train people to become better architects** and helped establish a consistent view across our companies, both of what architecture is and what good architecture's characteristics are."

Maranzano et al, Architecture Reviews: Practice and Experience, IEEE Software, March/April 2005.

Many Methods Are Available

- There are many software architecture evaluation methods
  - SAAM, ATAM, ALMA, FAAM,...
  - many are domain-specific
  - many are proprietary
- In this course we will discuss two evaluation methods – one developed by AT&T and one by the SEI.

Architectural Design Reviews: The ATT Experience*

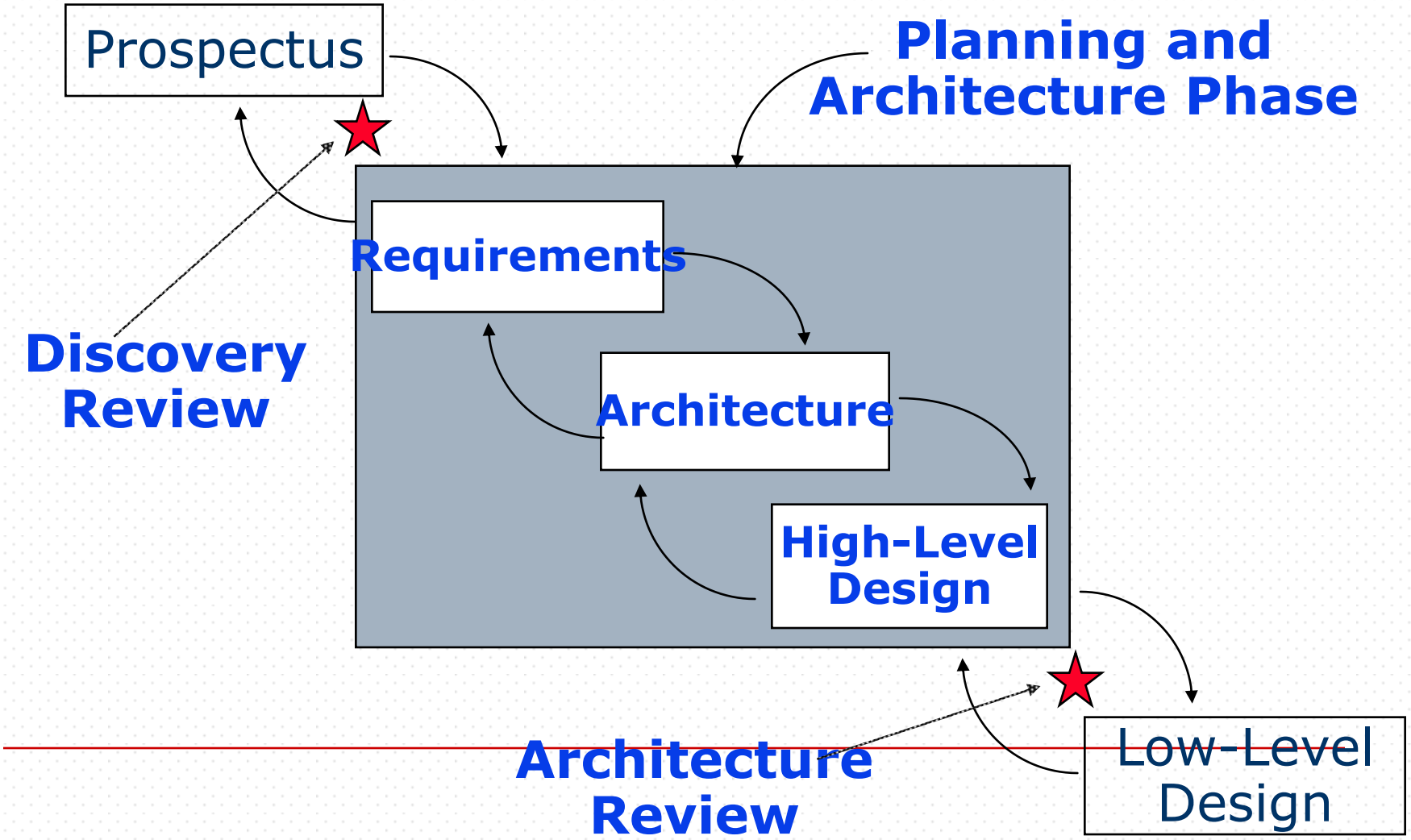
- Design Reviews at ATT
  - Architecture Review Board (ARB) created 1988
- Process consisted of two reviews
  - Approximate costs: 70 staff-days (distributed across about 15 people)
- Results
  - Reviewed over 350 projects; voluntary participation
  - "A correct architecture has the largest single impact on cost and quality of the product" (Maranzano 1995)

*Source: *Best Current Practices, Architectural Evaluation,* Joe Maranzano, 1995.

Beyond ATT

- Key participants in ATT ARB left the company
  - Lucent, Avaya, Millennium
- Adapted the ATT methods in their new jobs
  - Some differences, but many similarities
- IEEE Software article reports on results from 700 reviewed projects
  - Architecture Reviews: Practice and Experience.” J. Maranzano, et al., *IEEE Software,* April 2005

Two Design Reviews Conducted



Discovery Reviews

- Occur as business goals, problem statement, &high-level requirements are set
- Cover in depth the customer problem statement, functionality, and high-level options of the architecture
- Discussion of alternatives is key to simplification
- Purpose is to validate match between problem statement and requirements, and make sure the architecture is headed in right direction.

Architecture Reviews

- Occur after the major decisions on architecture have been made
- Cover in depth the areas of
  - Functionality
  - Performance
  - Error recovery
  - “Operations, Administration & Management” (OA&M)

How an ATT Architecture Review is Conducted - 1

- Before the review
  - ARB Chairperson meets with the project to determine the technical focus
  - Chairperson assembles a review team of subject matter experts - voluntary participation, but encouraged by corporate management
  - Project sends out review material two weeks before review
  - ARB generates a list of questions and concerns several days before the review

How a Review is Conducted - 2

- A 2- or 3-day review is conducted by the ARB Team
  - Detailed talks are presented by project personnel on key technical areas
  - Raised issues are recorded on “snow cards”
- Immediate feedback is given to the team at the end of the review
  - Cards are grouped by “Things done right”, “Issues”, and “Recommendations”

How a Review is Conducted - 3

- Chairperson follows up with a written report and a presentation to project management
  - Project team develops action plan based on findings
  - Project team and ARB team present evaluation and action plan to project management and ARB directors

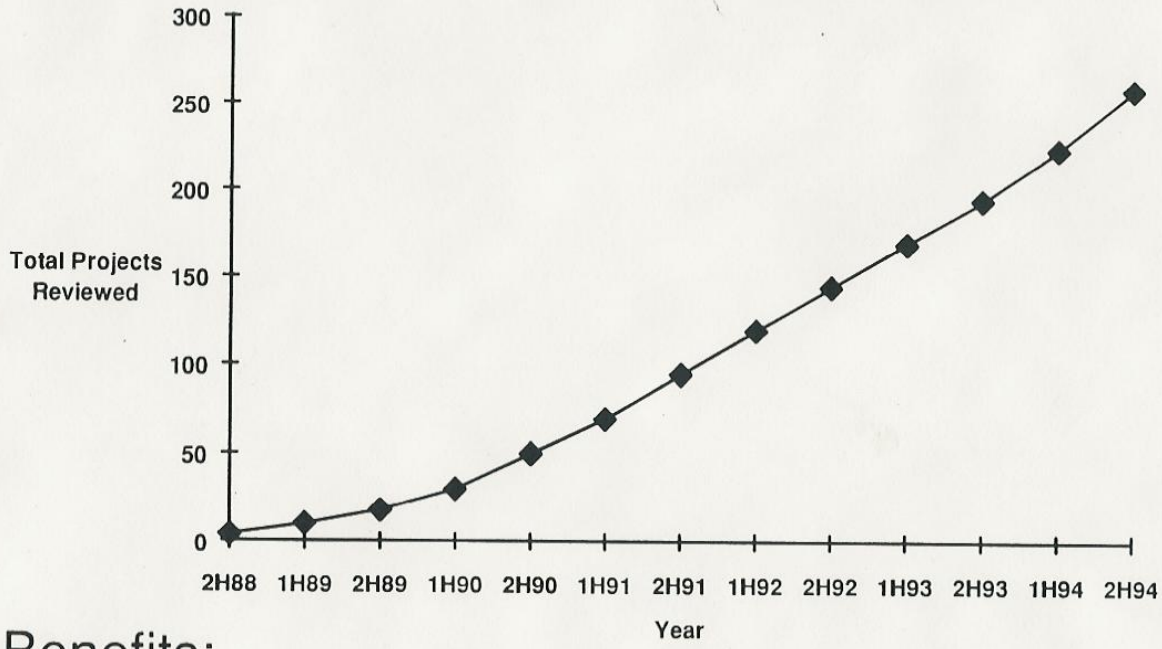
What they look for in an Architecture

- The *purpose* of the system
- The major *functional components* and/or *platforms*
- The *relationships* and *dynamic interplay* of control and communication between components
- The system's *ease of use and operation*
- *Data storage* and *flow of data* between components
- *Resources* consumed by each component

Kind of Systems Reviewed

- Mostly IT Domains
  - Transaction processing/customer care
  - Decision support systems
  - Event management systems
  - Data accumulation or transformation systems
  - Integrated collection of systems
- Range of sizes
  - 7 staff members to 100s of staff members

Architecture and Discovery Reviews



Benefits:

- Average 10% savings per reviewed project
- Substantially larger savings on a few reviewed projects

Checklists: Error Recovery

1. If your system is running in a primary/secondary relationship, is there anything in the system to prevent two systems from coming up at the same time and attempting to become primary? **[Architecture, Dist. Sys., Sys. Admin]**
2. Can the system operate without the database? Can input be accepted for later processing when the database is restarted? What are the manual overrides in the system? **[Database]**
3. How complex is the database conversion? **[Database]**
4. How critical is the database? What happens to the system if the database is corrupted? **[Database]**
5. If other systems are dependent upon your system for updating their databases, how do you ensure that the interface between the systems will remain constant as my system changes? Is positive acknowledgment that they received the update required? Should you be able to retransmit prior to updates that the downstream systems missed? **[Database, Architecture]**

More Checklists

6. If audits are being done in a distributed system, have you performed a checksum on a block of records? **[Database, Dist. Sys., Sys. Admin]**
7. What is the error rate expected in parsing data input by other systems? What is the cost of incorrect data? Is manual handling required to correct the errors? **[Database, Dis. Sys., User I/F?]**
8. Do you also have database unconversion if you have to go backwards? **[Database, Migration]**
9. In a distributed environment, what are the failure modes? Have they been documented? What actions are required to recover from each? **[Dist. Sys., Fault Isolation]**
10. Is this data being sent between systems in a consistent unit of information? (More than 50% of the trouble reports in some systems are related to communications interfaces within them.) **[Dist. Sys., I/O]**

References

- *Best Current Practices, Architectural Evaluation,* J. Maranzano, ATT Technical Report.
- *Architecture Reviews: Practice and Experience*. J. Maranzano, S. Rozsypal, G. Warnken, D. Weiss, P. Wirth, and G. Zimmerman. IEEE Software 2005.