Common Quality Attributes in Software Projects

Lotfi ben Othmane

Architecture Design

Architecture drivers

- Design purposes
- Quality attributes
- Primary functionalities
- Architectural concerns
- Constraints

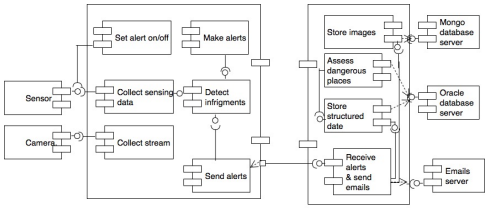
Design concepts

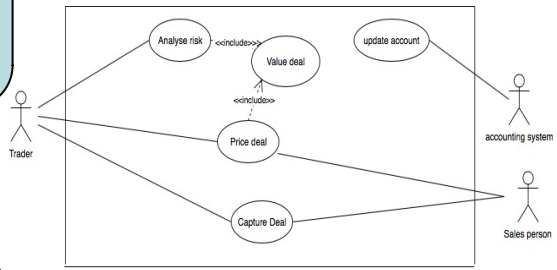
Selects and instantiates

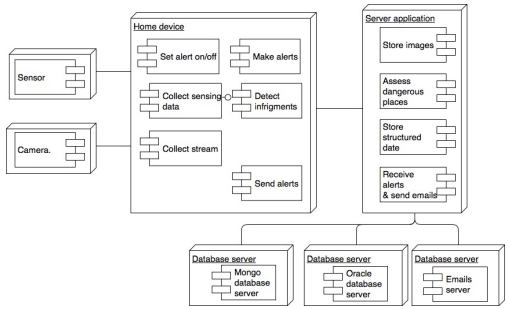
Candidate design decisions

Architect

Architecture structures





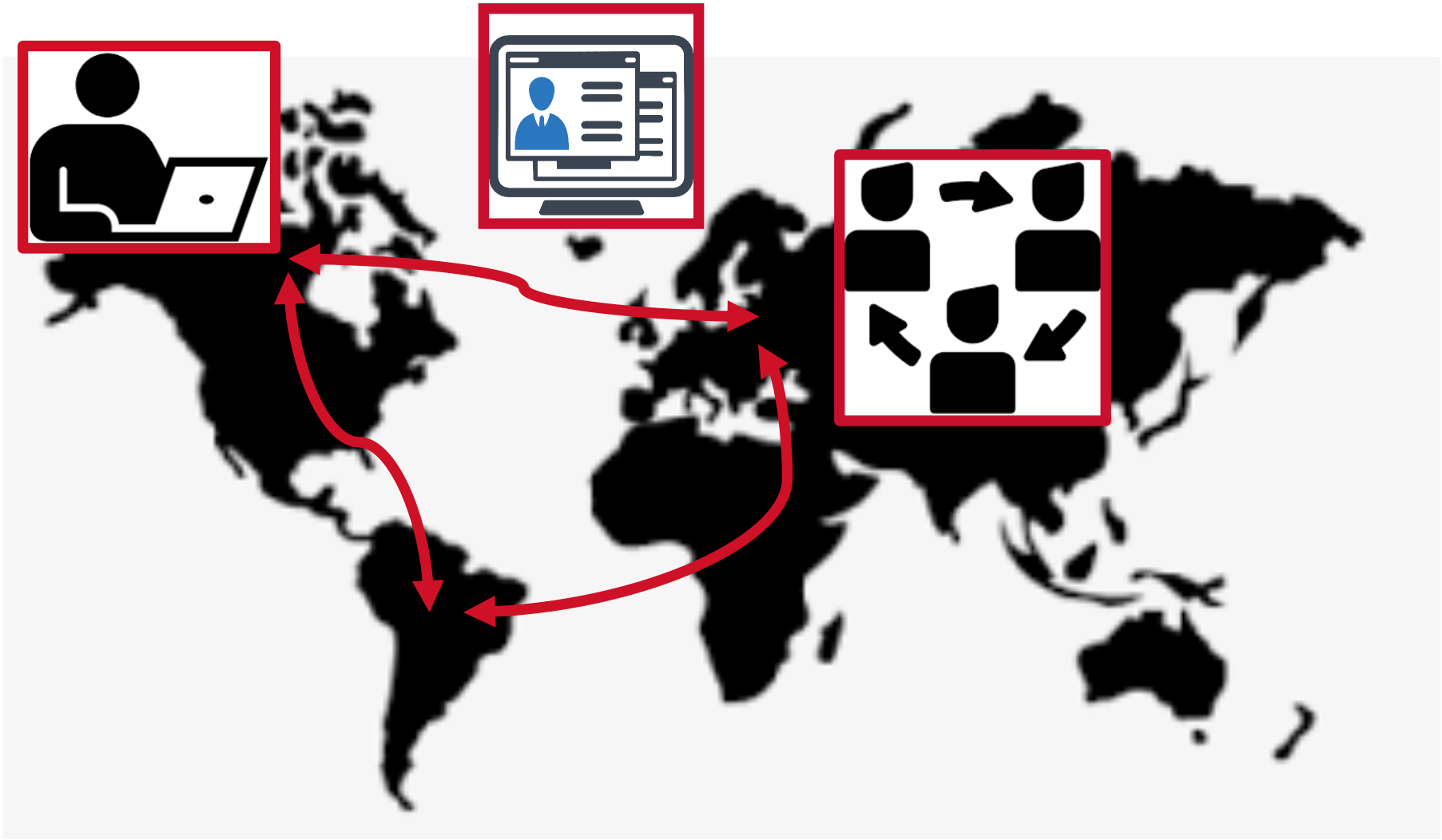


Quality Attributes

Software quality attributes (QA) characterize the usefulness of the software in a given environment.

ISO 9126 Software Quality Characteristics

Importance of Quality Attributes



Quality Attributes Categories

ISO 9126

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

CMU SEI-ATAM

- Modifiability
- Performance
- Availability
- Deployability
- Security
- Scalability
- Interoperability
- Usability
- Etc.

Example of Discussion About a Quality Attribute Scenario

- **Evaluator:** What are some of your main concerns for the system being evaluated?
- **Stakeholder:** We have to provide flexibility.
- **Evaluator:** What do you mean by “flexibility”?
- **Stakeholder:** The system shouldn’t constrain the data flow. If a particular flow of information doesn’t work well, we should be able to create a new process.
- **Evaluator:** So, the system shouldn’t provide one predetermined process but should let you create new processes or modify existing processes.
- **Stakeholder:** Exactly. We would like the system to be flexible rather than constraining.

Reminder: Quality Attribute Scenarios

Scenario: Performance scenario for submitting applications

- Stimulus: Regular application
- Stimulus source: User
- Response: Save the application, process the payment, return a confirmation to the user
- Response measure: < 5 sec
- Environment: normal and overload conditions
- Artifact: submission, payment, save file, save database

"A list such as this, derived from real project data, can help teams reason about future changes they should consider and avoid being caught off guard by an impactful, unanticipated change."

Toward Agile Architecture

Insights from 15 Years of ATAM Data

Stephany Bellomo, Ian Gorton, and Rick Kazman,
Software Engineering Institute

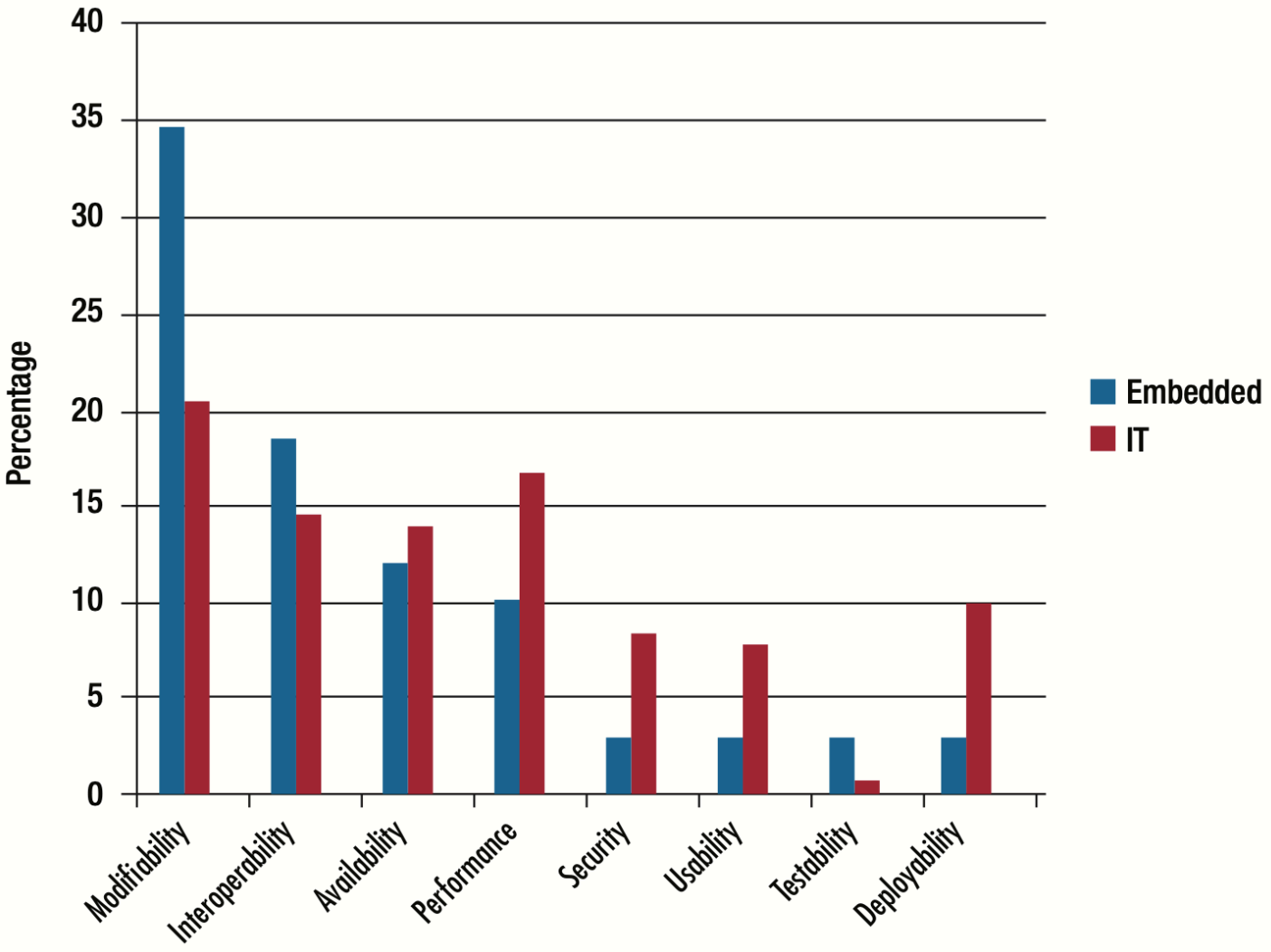
The Collection

Study	#Projects	#Scenarios	%IT	%Embedded
1 (1999-2006)	24	1072	58	42
2 (2006-2013)	9	348	78	22

Deployability-Related Scenarios

1. A new release is being planned; the final upgrade plan must be developed in a week (considerable time is spent on configuring specific installations).
2. An upgrade is being pushed out; the software is robust enough to handle 99 percent of errors, leaving 1 percent for manual handling.
3. All full controller builds must be completed in less than 5 minutes (even under the build farm's peak load).
4. The average time to create or build a new system-level test case is one day.
5. One team of developers modifies release version x, while another team makes architectural changes to versions x and x + 1 that aren't synched with the first team's changes. The first team integrates the modifications into version x + 1 within eight weeks.
6. A new version of the spec isn't compatible with previous versions. The software application and adapter should be backward- compatible, besides supporting the new version.

Frequency of Quality Attributes Concerns



Modifiability Concerns

1. Adding, replacing, or modifying a functionality or component, for example, a third-party component, new sensor, adaptor, or algorithm. (32)
2. Configurable software: order throughput, UI content, display, feature toggle, reporting, or simulation. (15)
3. Operating system or registry changes (11)
4. Replacing or changing the networked environment, for example, a new protocol or network platform. (10)
5. Changing, adding, or replacing a model or tool capability-- run-time and non-run-time. (8)
6. Software component composition or reuse from the core asset base. (7)

Modifiability Concerns—Cont.

7. Porting or integrating a new technology, device, or hardware platform--a new vehicle, a device, sensors, a dual core, and so on. (6)
8. Changing message content and formats--translation might be required (4)
9. Adding or replacing services, such as messaging. (3)
10. User- or context-driven adaptation, for example, ActiveX not allowed (3)
11. Replacing or upgrading middleware, a webserver, or a database (2)
12. Minimizing merge complexity (2)

Modifiability Concerns

- They are about anticipating changes
- Frequency of modifiability scenarios: 34% for embedded system and 21 for IT systems

- The first two concerns count 46% of the modifiability concerns

Performance Concerns

1. Maintaining response time for a system capability--simulation, display, backup, run report, and so on (25)
2. Maintaining response time for critical capabilities with limited or intermittent resources (25)
3. Maintaining response time while the hardware or platform changes, for example, a different machine or new CPU architecture (21)
4. Maintaining response time while the load or usage increases--increased load, increased service demands, increased fidelity, and so on (9)
5. Maintaining start-up or shut-down response time threshold. (8)
6. Monitoring system or network performance (6)
7. Processing transactions with an increased load within an acceptable degraded range, for example, increased transactions or larger files (4)
8. System capability working as expected (2)

Performance Concerns – Cont.

- Performance concerns count about 17% for IT projects and 10% for embedded projects
- Maintaining response time scenarios counts 80% of the performance scenarios

Availability Concerns

1. Critical operational capabilities -- server, network card, and so on-- remaining intact after a hardware failure--or blocked access. (34)
2. Detecting faults, updating logs, or monitoring outage trends, for example, a software component failure or bad node. (19)
3. Critical operational capabilities remaining intact after a software failure or bad data input. (15)
4. Restarting after a failure without losing data or state information -- rollback, resynchronize, and so on. (11)
5. Critical operational capabilities remaining intact after a network or messaging failure. (9)
6. Critical operational capabilities remaining intact after a synchronization failure, for example, clock time or data sources. (6)
7. Visibly notifying the user if hardware or software components fail. (6)

Availability Concerns – Cont.

- Availability concerns count about 14% for IT projects and 12% for embedded projects
- Maintaining critical capabilities counts 60% of the scenarios

Top Quality Attributes Concerns

Rank	Quality attribute	Concern
1	Modifiability	Reducing coupling
2	Performance	Latency
3	Interoperability	Upgrading and integrating with other system components
4	Modifiability	Designing for portability
5	Usability	Ease of operation
6	Availability	Fault detection
7	Interoperability	Ease of interfacing with other systems
8	Modifiability	Designing for extensibility
9	Availability	Fault recovery
10	Performance	Resource management
11	Deployability	Minimizing build, test, and release duration

Use of Study Results in Agile Development

- Project owners want to pay for visible functionalities

- Use as a checklist to reflect about quality attribute concerns
- Create review questions to think about design tradeoffs
  - *"Did we impact another quality attribute concern with the design decision that we are making?"*
- Brainstorm about QA deficiencies in retrospective meetings
- Propose addressing QA scenarios in Sprint planning meetings

Self-Check Questions

- What is software quality attribute?
- Give examples of scenarios for the quality attributes modifiability, performance, and security.
- How quality attributes could be waved in an Agile development process?

Thank you

Questions?