Designing Software Architectures for Usability

Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Tutorial Contents

- Introduction
- Engineering Architectural Design
- Basic Usability Concepts
- Elaborating Usability for Architecture
  - Usability Properties
  - Usability Patterns
- Assessing Usability in Architecture
- Designing for Usability

Tutorial Area

- Over the last decade, software architecture has become an area of intense research in the software engineering community
- Architectural analysis of quality attributes aims to bring forward critical decisions that affect different quality attributes to the architectural design phase
- Usability is a quality attribute that can be supported by architecture

Tutorial Sources

- European Research Project:
  STATUS (Software Architecture that supports Usability)
- Duration:
  12/01-12/04
- Partners:
  - Technical University of Madrid (Spain)
  - University of Groningen (Netherlands)
  - Imperial College (England)
  - IHG (Spain)
  - LogicDis (Greece)

Tutorial Goals

- Usability: the major omission of developers
- Basic usability concepts
- Predict usability in the architecture
- Improve design for usability

Software Architecture

- Software architecture
  Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.
  [IEEE Standard P1471]
- Definition contains two main parts:
  - structural part – components and connectors
  - principles part – principles, design rules, design constraints, etc.

Software Architecture and QAs

- The architecture of a software system constrains the quality attributes:
  - performance
  - maintainability
  - usability
  - ...
- Therefore, the driving quality attributes influence the software architecture (more than the functional requirements)

Why software architecture?

First class representation of SA during

- design
  - stakeholder-based assessment
  - assessment of quality attributes
- implementation
  - configuration management
  - software product lines
- run-time
  - dynamic software architectures, post-fielding upgrades, dynamic reorganization, etc.

Being able to design for usability

Design process depends on:

- **Determining** when the software design process is finished (requires assessment techniques)
- Development/identification of design solutions that **improve** usability

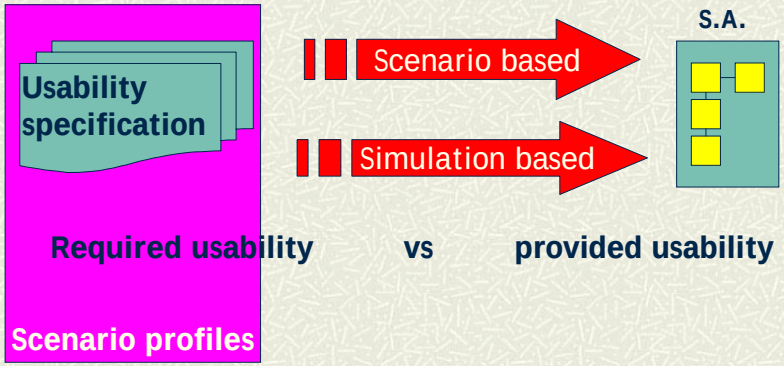
Architecture assessment

3+1 distinct types of assessment:

- scenario based
- simulation based
- mathematical modeling based
- (experience based assessment)

Architecture assessment

Assessment techniques



Conclusion

- Tutorial goal and contents
- Software architecture
- STATUS project
- Software architecture design method

Usability and Usability Attributes

Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Basic usability concepts

- Quality Criteria
  - Functionality
  - Reliability
  - Performance
  - Usability
  - Maintainability
  - Portability

(According to ISO)

Basic usability concepts

- Definition
  - Quality in use (ISO 14598,99)
  - User-software relationship
  - Not only interface, but interaction

Basic usability concepts

- Usability Attributes

A precise and measurable component of the abstract concept that is usability

Usability decomposition into attributes

ATTRIBUTE\SOURCE	CONSTANTINE 99	HIX 93	ISO 9126_00	NIELSEN 93	PREECE 94	SHACKEL 91	SCHNEIDERMAN 98	WIXON 97
LEARNABILITY	Learnability	Learnability	Learnability	Learnability (ease of learning)	Learnability (ease of learning)	Learnability (time to learn)	Time to learn	Learnability (initial performance)
EFFICIENCY	Efficiency in use	Long-term performance	Operability (?)	Efficiency of use	Throughput	Effectiveness (speed)	Speed of performance	Efficiency (long-term performance)
MEMORABILITY	Rememberability	Retainability	--	Memorability	-	Learnability (retention)	Retention over time	Memorability
RELIABILITY	Reliability in use	-	Operability (?)	Errors	Throughput	Effectiveness (errors)	Rate of errors by users	Error rates
SATISFACTION	User satisfaction	Long-term user satisfaction	Attractiveness	Satisfaction	Attitude	Attitude	Subjective Satisfaction	Satisfaction or likability
UNDERSTANDABILITY			Understandability	-	-	-	-	-
ADAPTABILITY	-	-	-	-	Flexibility	Flexibility	-	Flexibility
FIRST IMPRESSION	-	First impression	-	-	-	-	-	First impressions
EXTRA FEATURES	-	Advanced feature usage	-	-	-	-	-	Advanced feature usage
INITIAL EFFICIENCY	-	Initial performance	-	-	-	-	-	Learnability (initial performance)
EVOLVABILITY	-	-	-	-	-	-	-	Evolvability

Usability decomposition into attributes

- Learnability
  How quickly and easily users can begin to do productive work with a system that is new to them, combined with the ease of remembering the way a system should be operated
- Efficiency of use
  Number of tasks per unit time that the user can perform using the system
- Reliability in use
  Error rate in using the system and the time it takes to recover from errors
- Satisfaction
  Subjective opinions that users form in using the system

Elaborating usability for architecture consideration

- Usability Properties
  - These properties embody the heuristics and design principles that researchers in usability have found to have a direct influence on system usability
  - Usability requirements of a system that are more directly related to the solution domain and which have a direct relationship to software design decisions

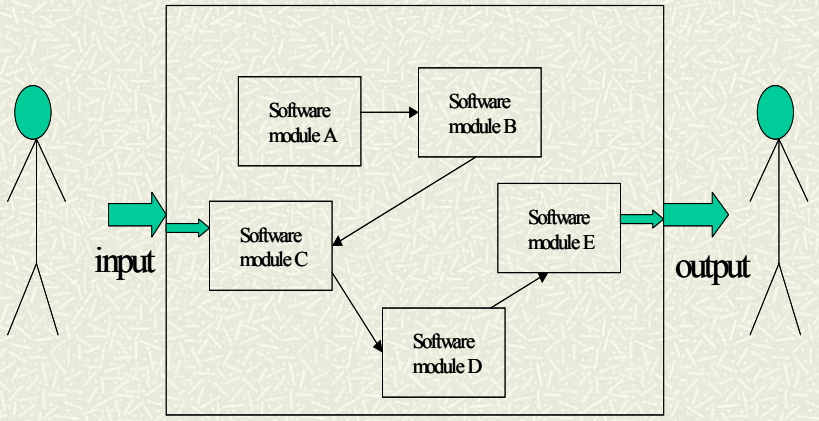
Usability properties

- **Providing feedback**
- **Error management**
  - Error prevention
  - Error correction
- **Consistency**
  - UI consistency
  - Functional consistency
- **Guidance**
- **Minimize cognitive load**
- **Explicit user control**
- **Natural mapping**
  - Predictability
  - Semiotic significance
  - Ease of navigation
- **Accessibility**
  - Disabilities
  - Multi-channel
  - Internationalization
- **Adaptability**
  - Match user preferences
  - Match user expertise

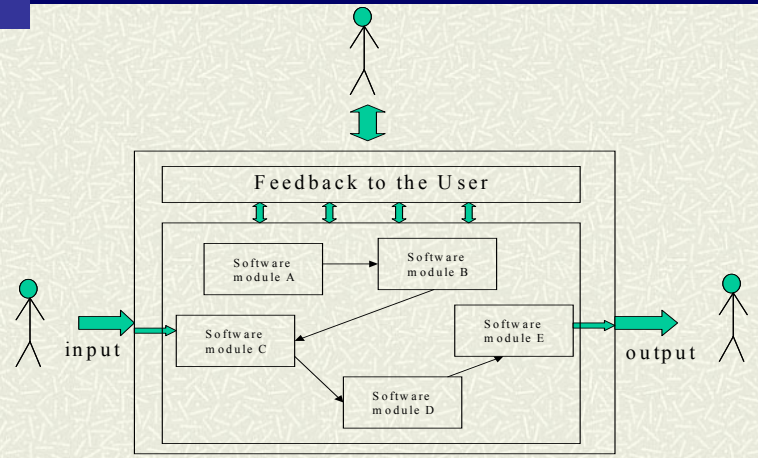
Example of the influence of a property on architecture

Suppose that we have a software system. Imagine that the process to be performed by the system is complex and takes several minutes. In this case, the feedback attribute is important for improving user satisfaction.

Example of influence



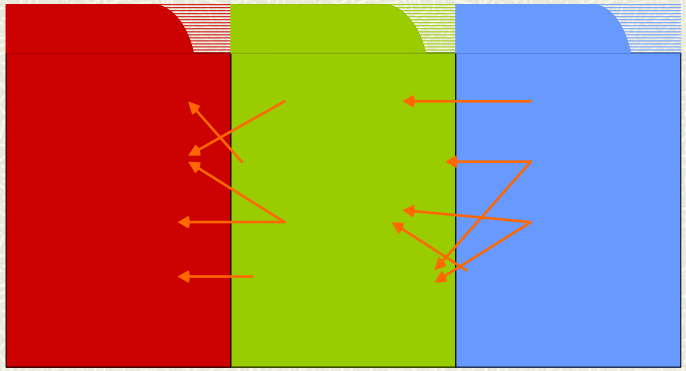
Example of influence



Elaborating usability for architecture consideration

- Usability Pattern
  - A technique or mechanism that can be applied to the design of the architecture of a software system in order to address a need identified by a usability property at the requirements stage

Usability Decomposition: attributes/properties/patterns



Conclusion

- What is usability
- Usability attributes
- Usability properties
- Usability patterns

Specifying Usability

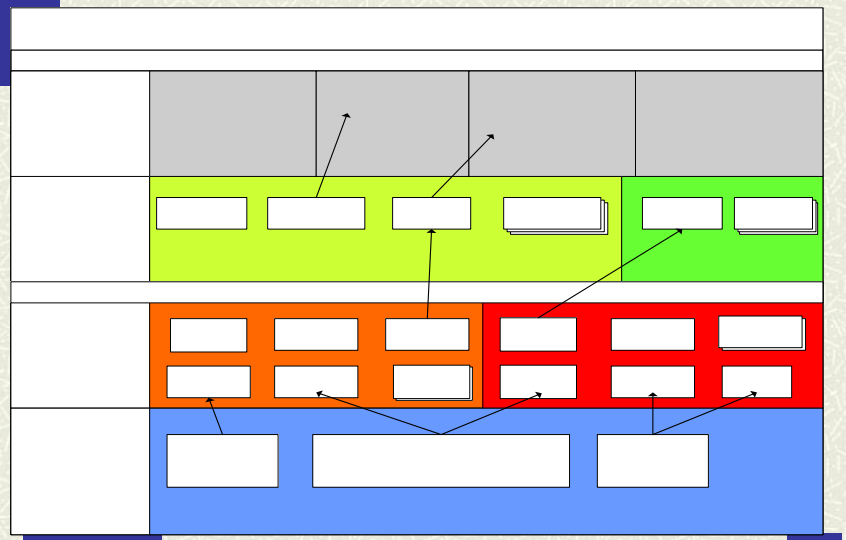
Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Layered model of usability



ISO

- effectiveness
- learnability
- retention
- flexibility

Traditional usability specification techniques

Traditional techniques [Preece, 94], [Hix, 93]

- Metric based. Specified in measurable from end user perspective. Unsuitable for S.A. assessment
- Difficult to relate to S.A. need other way of describing required usability

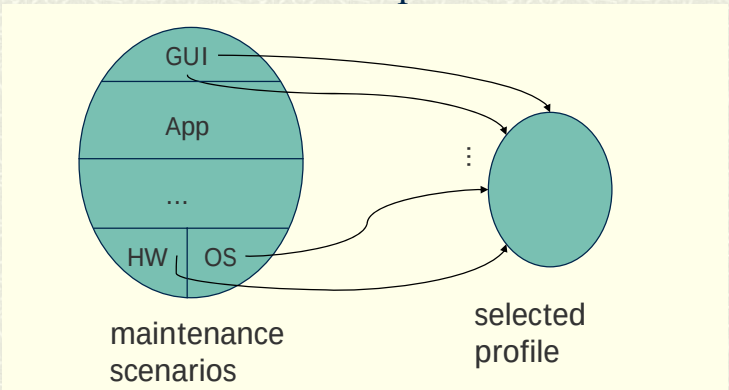
Benefits of using **scenarios**:

- Scenarios are more specific than “abstract” usability req.
- Scenarios defined over all user perceived functionality/user population/contexts

- efficiency
- memorability

Scenario profiles

- absolute versus selected profiles



Scenario profiles

- Notion of equivalence class
- Scenario categories

Generic versus system specific QA specification:

- Generic – e.g. McCabe's cyclomatic complexity metric
- Specific – scenario profiles

Scenario profiles

- Top-down or bottom-up
- Top-down profile development
  - (pre-)define scenario categories
  - selection and definition of scenarios for each category
  - each scenario is assigned a weight (either based on historical data or estimated)

Scenario profile development

- Bottom-up profile development
  - interview stakeholders
  - categorize scenarios
  - assign weights to scenarios
  - iterate until sufficient coverage
- Stopping criterion
  - coverage

Scenario profiles - QAs

- performance: usage profile
- maintainability: maintenance profile
- reliability: usage profile
- safety: hazard profile
- security: authorization profile
- usability: usability profile

Elicitation techniques

- QA level: scenarios that correspond to changes with high likelihood
- Risk assessment: scenarios that search for difficult/impossible changes
- Comparison: scenarios that highlight differences

Dialysis system maintenance profile

Category	Scenario Description (Weight)
Market Driven	Change measurement units from Celsius to Fahrenheit for temperature in a treatment. (0.043)
Hardware	Add second concentrate pump and conductivity sensor. (0.043)
Hardware	Replace blood pumps using revolutions per minute with pumps using actual flow rate (ml/s). (0.087)
Hardware	Replace duty-cycle controlled heater with digitally interfaced heater using percent of full effect. (0.174)
Safety	Add alarm for reversed flow through membrane. (0.087)
Medical Adv.	Modify treatment from linear weight loss curve over time to inverse logarithmic. (0.217)
Medical Adv.	Change alarm from fixed flow limits to follow treatment. (0.087)
Medical Adv.	Add sensor and alarm for patient blood pressure (0.087)
Com.and I/O	Add function for uploading treatment data to patient's digital journal. (0.043)
Algorithms	Change controlling algorithm for concentration of dialysis fluid from PI to PID. (0.132)

Definition scenario profile

scenario profile = "description of the semantics of software quality factors for a particular system in terms of a set of *scenarios*"

SP's defined by software architect as part of architectural assessment

Definition scenario profile

scenario = "Scenarios refer to interactions between independent entities. Entities can be stakeholders, the system [Carroll 1995] (or possibly parts of it such as hardware, software, subsystems, objects) and the environment [ITU 1996]"

Stakeholder = (IEEE) an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system. examples: users, customers, developing organization, project management and so on.

Scenario profile properties

Types of SP:

- **Complete**: "all scenarios that can possibly occur"
- **Selected**: "a representative subset of the population of all possible scenarios"

Scenario profile properties

Types of context for SP creation:

- **Greenfield context**: depends on experience/skill/ creativeness individuals/groups
- **Experienced context**: more accurate because historical data available as additional input

Synthesizing scenario profiles

- Creation of profiles is subjective
- No means to verify representativeness of the profile

costs vs. representativeness

Profile creation_experiment			
	costs	Productivity/ average created profile length	variation
individually	1 x 1 session	41	13
Unprepared Group	n x 1 sessions	43	6
Prepared group	n x 2 sessions	74	3

Scenario profiles for Q.A. assessment

- Practice and experience [Bosch & Bengtsson, 2001]
- *Maintenance profile* = set of change scenarios

Same underlying principles can be applied to usability:

Define: *usage profile*

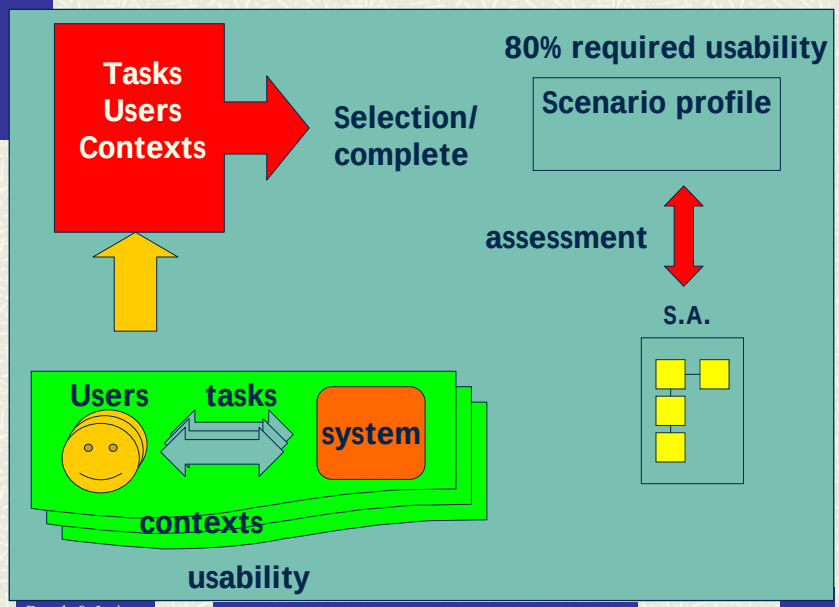
Usage profile definition

usage profile: "a description of the semantics of **usability** for a particular system in terms of *scenarios*"

usage profile = set of *usage* scenarios

(bottom up approach) S.P. defined over:

- (User perceived) Functionality (tasks) of system
- Stakeholders (users)
- Context in which users operate



Usage scenario formal definition

Scenario: "an interaction (task) between the independent entities: users, the system, and its contexts"

$$f(u,t,c) \rightarrow u \otimes t \otimes c.$$

where:

u = users

t = tasks

c = contexts

Shortcomings of formal definition

- Need to find a way to express the *usability issues* a user has while performing particular a task in a particular context
- Solution: relate to usability attributes
  - Learnability
  - Efficiency of use
  - Reliability
  - Satisfaction

Attribute Preference Table (APT)

APT relates scenarios to usability

attribute preference table						
User	Task	Context	L	E	R	S
novice	Insert order	training	5			
expert	Search	helpdesk		5		

Usage scenario formal definition

$f(u,t,c,APT) \rightarrow (u \otimes t \otimes c \otimes APT)$

where:

u = users

t = tasks

c = context

APT = Attribute Preference Table

Synthesizing APT

Determination of preference values

- Expert based
- Survey/interview stakeholders

Quantification of usability

- Assign values (1-5)
- Pair wise comparison (learnability/efficiency)

Experiment settings

More people involved → more representative

Example APT

Scenario attribute preference table

User	Task	Context	Learnability	Efficiency of use	Reliability	satisfaction
A	T1	C1	5	2	4	3
A	T2	C2	5	5	3	2
A	T3	C1	1	1	3	3
A	T4	C2	1	…	3	3
B	T1	C1	….		…	…
B	T1	C2				
B	T2	C1				
…	…	…				

Scenario reduction

Problem:

- nr of scenarios can be quite large (5 users, 10 tasks and 4 contexts → 200 scenarios)

Scenario reduction techniques:

- Create separate attribute tables for task and context
- Abstract scenario profiles

Create separate attribute preference tables

f(t, TAP, u, c, CAP) → ((t ⊗ TAP) ⊗ u ⊗ (c ⊗ CAP))

t= task, u = users, c = context

TAP = task attribute prioritization

CAP = context attribute prioritization

TAP ⊗ CAP results in SAP (scenario attribute prioritization)

- Fewer scenarios / less information about req usability
- How to compose SAP from TAP & CAP ?

TAP				
Task	L	E	R	S
Insert order			5	
Search		5		

CAP				
Context	L	E	R	S
Helpdesk		5		
Training	5			

apt						
User	Task	Context	L	E	R	S
novice	Search	Helpdesk		10		
expert	Search	Helpdesk		10		

less information about required usability

Abstract S.P. Task abstraction

Define abstract tasks: Group similar tasks into "abstract/generic" tasks with the same level of representativeness

PRE: to maintain representativeness : usability concerns related to tasks which are abstracted must be the same. (expert based prediction)

example: Tasks — abstract task

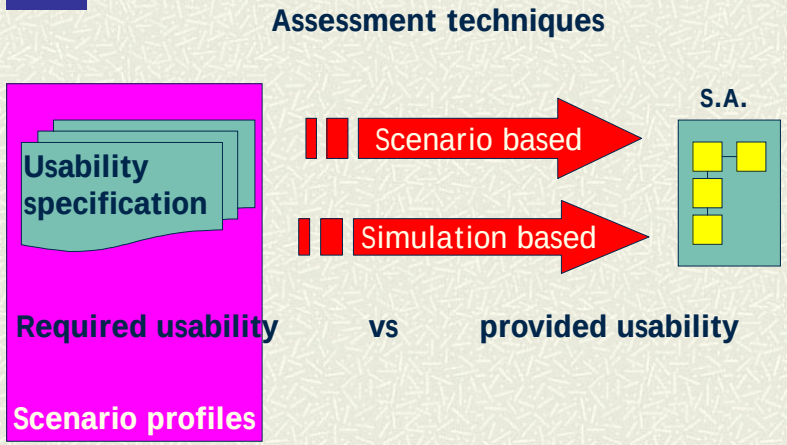
- Insert customer
- Insert order
- Insert salesman

} insert "business object"

Generating usage profiles

1. Identify the context of scenario profile generation.
2. Identify the users.
3. Identify the tasks.
4. Identify the contexts.
5. Create attribute preference table APT.
6. Select scenarios / Create scenario profile

Architecture Assessment



Conclusion

- Traditional usability techniques
- Scenario profiles
- Usability profile
  - attribute preference table
- Generating usage profile

Assessing Software Architectures for Usability

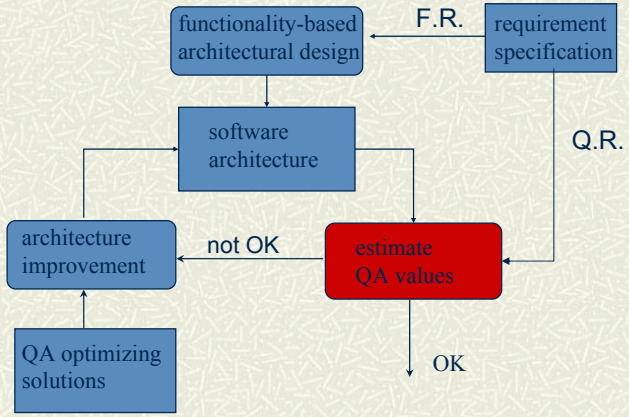
Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

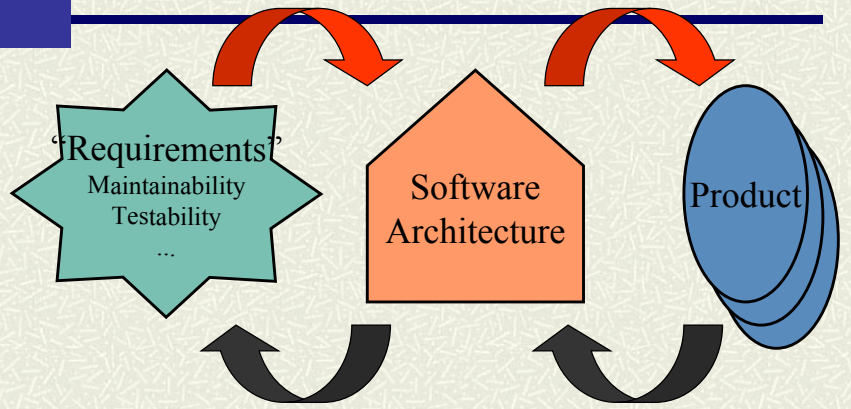
Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Architecture Design Method



Architecture assessment



Software Architecture Validation

Software Architecture Verification

Architecture assessment

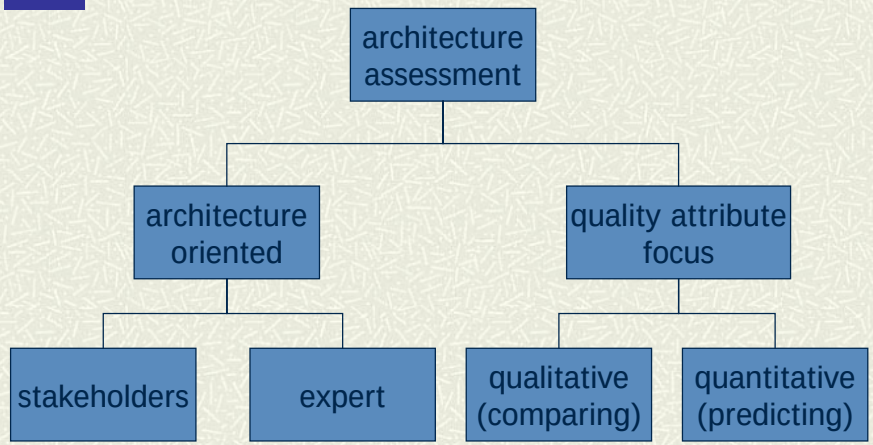
Two approaches:

- after each design iteration
- as a 'toll-gate' before starting next phase

Goals for assessment:

- quality attribute satisfaction
- stakeholder satisfaction
- support for software product line
- software system acquisition

Architecture assessment



Assessing quality attributes

- Assessment goals:
  - Relative assessment
  - Absolute assessment
  - Assessment of theoretical maximum

	A	B
performance	+	-
maintainability	-	+

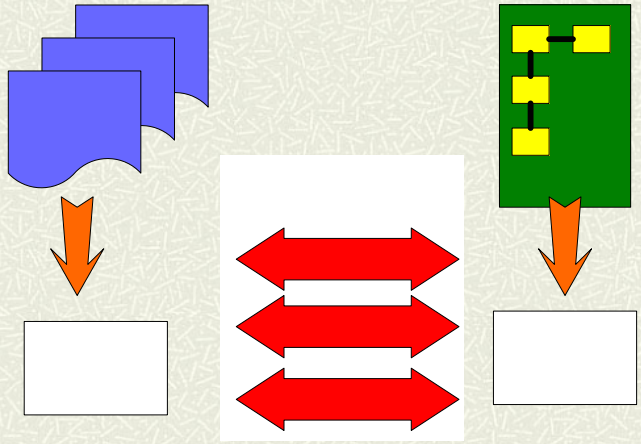
Generic assessment method

1. Set goal: determine the aim of the analysis
2. Describe software architecture: give a description of the relevant parts of the software architecture
3. Elicit scenarios: find the set of relevant scenarios
4. Evaluate scenarios: determine the effect of the set of scenarios
5. Interpret the results: draw conclusions from the analysis results

Goals for assessment

- Quantitative quality attribute prediction
- Risk assesment
- Comparative analysis

Architecture assessment techniques



SALUTA

The Scenario based Architecture Level UsabiliTy Analysis method (SALUTA)

1. Determine the goal of the assessment.
2. Create usage profile.
3. Describe the software architecture.
4. Evaluate usage scenarios: determine the support for the scenarios.
5. Interpret the results: draw conclusions from the analysis results.

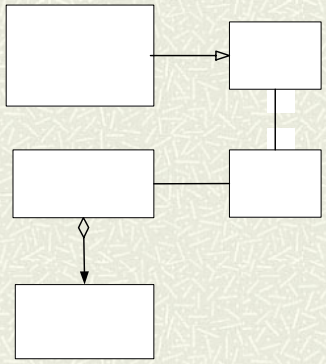
1. Determine the goal of the assessment

- **Predict the level of usability**: give an accurate indication of the architecture to support for usability.
- **Risk assessment**: detect usability issues for which the software architecture is inflexible.
- **Software architecture selection**: compare two candidate software architectures and select the optimal candidate which has the best support for usability

2. Create usage profile (1)

Steps for usage profile creation:

1. Identify the context of scenario profile (SP) generation.
2. Identify all scenario entities for usability
3. Create usage scenarios
4. Scenario elicitation



Create usage profile (2)

Activities defined for profile creation

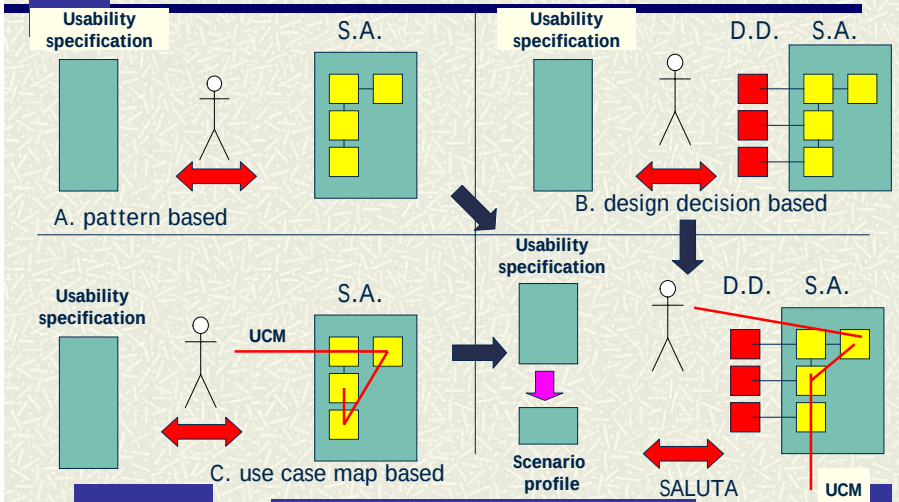
1. Identify the users
2. Identify the tasks
3. Identify the contexts of operation
4. Create attribute preference table
5. Scenario selection

Scenario attribute preference table						
Stakeholder	Task	Context	Learnability	Efficiency of use	Reliability	satisfaction
A	T1	C1	1	5	4	3
A	T2	C2	5	1	3	2

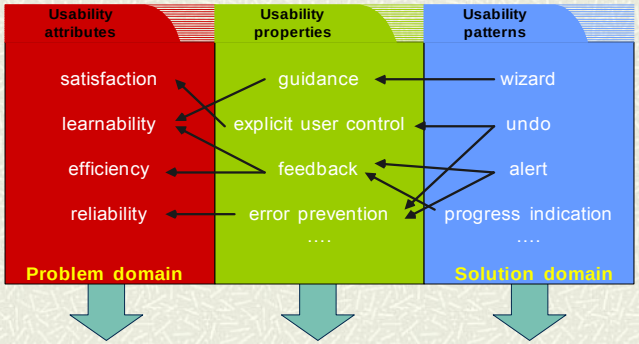
3. Describe the software architecture

- Analysis requires architectural information that allows the analysis to evaluate the scenarios defined for that quality attribute
- Only require architectural information related to usability → use framework to extract required information
- Defined 3 types of evaluation techniques based upon architectural information available or one is willing to extract

Scenario based assessment

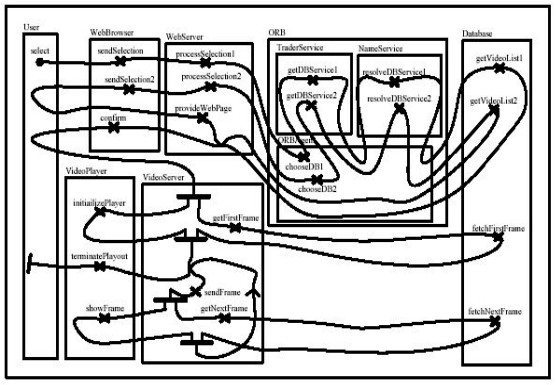


Use framework to extract information



3. UCM based 2. DD based 1. Pattern based

Use case maps: example



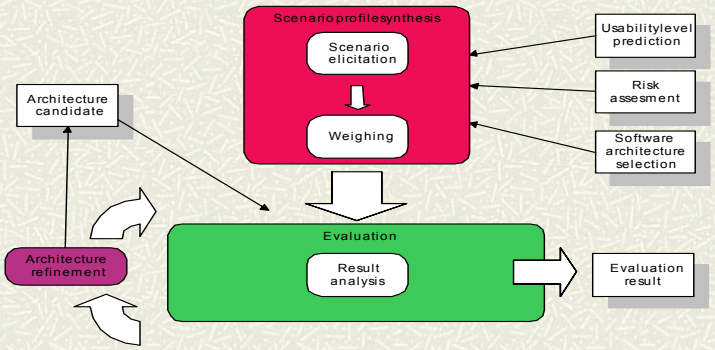
Operational indicators Indicators of efficiency

- Number of steps
- Path length

Indicators of learnability

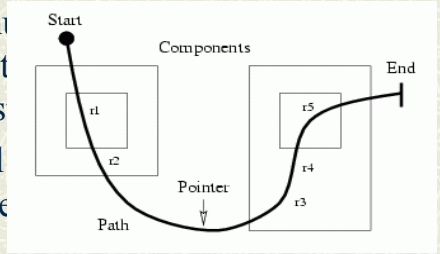
- Time to perform task 1

Overall evaluation process



Use UCM to describe the SA

- UCM describes behavioral and structural aspects of a systems at a **high** (architectural level) of abstraction
- Easy to learn & understand but precise.
- UCM Can show mu
  diagram and the int
  (reason about a sys
- UCM is a informal
  are interested in the



UCM

- High level Abstraction
- Trimming
- View separation/integration
- Aggregation of behavior

UCM appears to be the most promising candidate to capture abstract graphical representations of scenarios in an architectural context. especially in relation with abstract scenario profiles!

Conclusion

- Architecture assessment
- Different approaches to assessment
- SALUTA
- Scenario-based assessment
  - pattern-based
  - design decision-based
  - use-case map-based

Improving Software Architectures for Usability

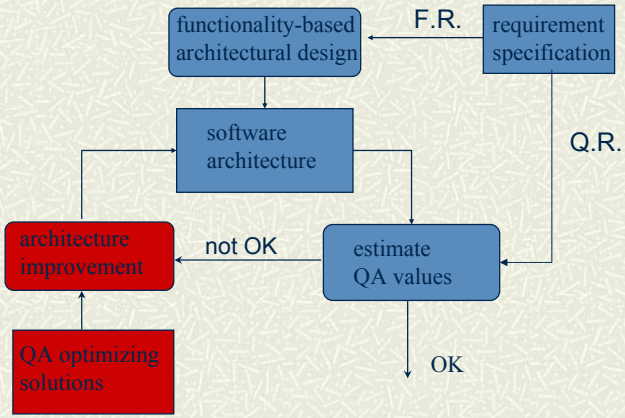
Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Architecture Design Method



Elaborating usability for architecture consideration

- Usability Pattern
  - A technique or mechanism that can be applied to the design of the architecture of a software system in order to address a need identified by a usability property at the requirements stage
  - The aim is to take what was previously very much the "art" of designing usable software and turn it into a repeatable engineering process
  - Their goal is to capture design experience in a form that can be effectively reused by software designers in order to improve the usability of their software

Usability Patterns

- Different languages
- Different acces methods
- Alert
- Status Indication
- Shortcuts (key and tasks)
- Form/Field Validation
- Undo
- Context sensitive help
- Wizard
- Standard help
- Tour
- Workflow Model
- History Logging
- Provision of Views
- User profile
- Cancel
- Multi-Tasking
- Commands aggregation
- Action for multiple objects
- Reuse information

Relationship Properties/Patterns

ACCESSIBILITY (internationalisation)	Different languages
CONSISTENCY, ACCESSIBILITY (multichannel, disabilities)	Different access methods
FEEDBACK	Alert
ERROR MANAGEMENT, FEEDBACK	Status indication
EXPLICIT USER CONTROL, ADAPTABILITY (user expertise)	Shortcuts (key and tasks)
ERROR MANAGEMENT (error prevention)	Form/field validation
ERROR MANAGEMENT (error correction),	Undo
GUIDANCE, ERROR MANAGEMENT	Context-sensitive help
GUIDANCE, ERROR MANAGEMENT	Wizard
GUIDANCE, ERROR MANAGEMENT	Standard help
GUIDANCE, ERROR MANAGEMENT	Tour
MINIMISE COGNITIVE LOAD, ADAPTABILITY, ERROR MGMNT (error prevention)	Workflow model
ERROR MANAGEMENT (error correction)	History logging
GUIDANCE, ERROR MANAGEMENT (error prevention)	Provision of views
ADAPTABILITY (user preferences)	User profile
ERROR MANAGEMENT, EXPLICIT USER CONTROL	Cancel
EXPLICIT USER CONTROL	Multi-tasking
MINIMISE COGNITIVE LOAD, ERROR MANAGEMENT (error prevention)	Commands aggregation
EXPLICIT USER CONTROL	Action for multiple objects
MINIMISE COGNITIVE LOAD, ERROR MANAGEMENT (error prevention)	Reuse information

Usability patterns: state of the art

- Usability pattern [Perzel, 99]:
  - "a description of solutions that improve usability attributes"
- User interface patterns[Casaday, 97] or interaction design patterns [Tidwell, 98]

Usability patterns: state of the art

- Feedback pattern [Wellie,00] :
  - "Provide a valid indication of progress. Progress is typically the time remaining until completion, the number of units processed or the percentage of work done. Progress can be shown using a widget such as a progress bar. The progress bar must have a label stating the relative progress or the unit in which it is measured"
- Progress Indication pattern [STATUS,02] :
  - " …. Components to be added to a software architecture and the relations among such components in order to provide such mechanism …."

Usability patterns: state of the art

- Usability scenario [Bass, 00]:
  - "A scenario describes an interaction that some stakeholder (e.g. user, developer, system administrator) has with the system under consideration from a usability viewpoint"

Usability patterns: state of the art

- Usability scenarios [Bass, 00]:

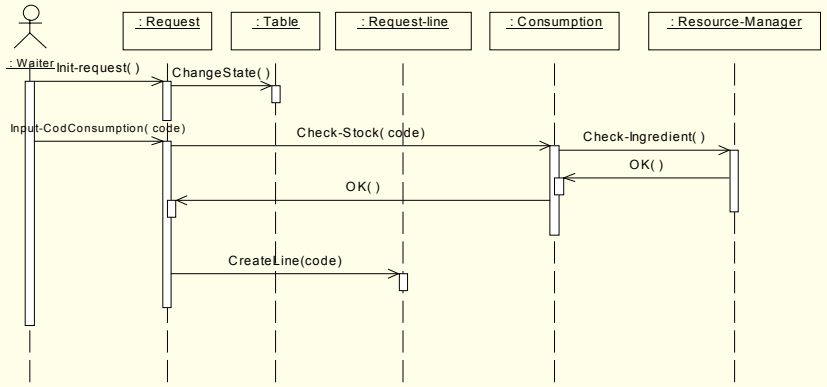
• Account for human needs and capability when interacting • Keep coherence through multiple views • Define upgrades similar to previous ones • Support international use • Predict task duration • Verify resources before beginning an operation • Present system state • Check for errors • Undo • Minimize user recovery work due to systems errors • Provide alternative secure mechanisms • Provide good Help • Novice interfaces for user in unfamiliar contexts	• Maintain device independence • Allow searching by different criteria • Make views accessible • Provide reasonable set of views • Cancel • Use applications concurrently • Allow quick switch back and forth between different tasks • Aggregate commands • Aggregate data • Provide test points for evaluation • Reuse information • Design easily modifiable interfaces • Quick navigation into a view

Usability patterns description

- Pattern Name
- Problem
- Design Solution
- Usability Benefits
- Usability Rationale
- Consequences
- Related Usability Patterns
- Example

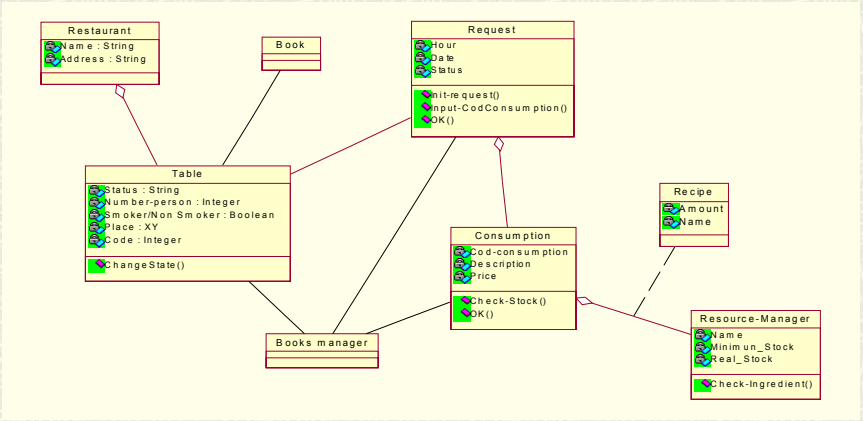
Getting design solutions

Interaction Diagram Without Usability Pattern *Status Indication*



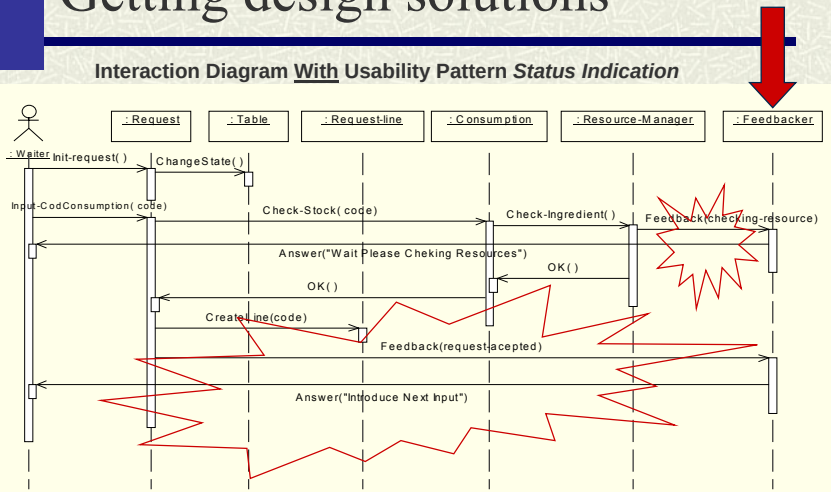
Getting design solutions

Class Diagram Without Usability Pattern *Status Indication*



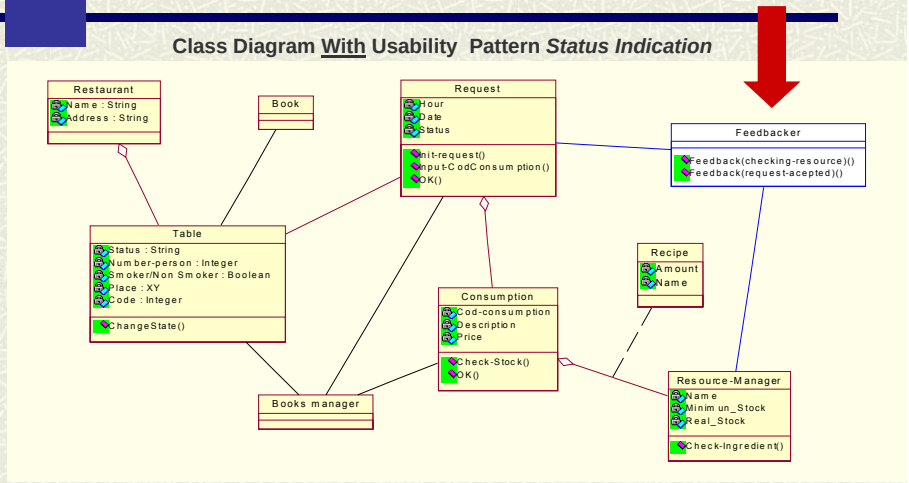
Getting design solutions

Interaction Diagram With Usability Pattern *Status Indication*



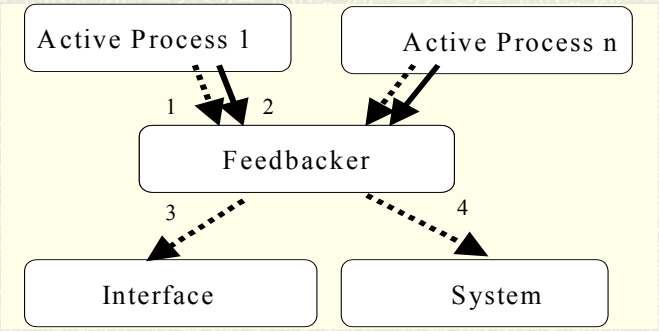
Getting design solutions

Class Diagram With Usability Pattern *Status Indication*



Getting design solutions

Abstracting Design Solution



Usability patterns catalogue

We provide tutorial participants with STATUS usability patterns catalogue

EXERCISE (if we have time)

1. We provide you with some requirements
2. Participants develope models
3. We modify requirements for usability improvement
4. Participants modify their models using usability patterns

Conclusion

- Exploring usability for SA
- Usability patterns: state of the art
- Usability pattern description
- Architectural design solution

Conclusion

Natalia Juristo
Professor of Software Engineering
Technical University of Madrid, Spain
natalia@fi.upm.es
http://www.ls.fi.upm.es/udis/miembros/natalia

Jan Bosch
Professor of Software Engineering
University of Groningen, Netherlands
Jan.Bosch@cs.rug.nl
http://www.cs.rug.nl/~bosch

Overview

- Tutorial introduction (*Jan*)
- Introduction to usability and STATUS project (*Natalia*)
- Usability attributes affected by software architecture (*Natalia*)
- Specifying usability (*Jan*)
- Assessing SA for usability (*Jan*)
- Improving SA for usability – usability patterns (*Natalia*)
- Conclusion (*Jan*)

Tutorial Contents

- Introduction
- Engineering Architectural Design
- Basic Usability Concepts
- Elaborating Usability for Architecture
  - Usability Properties
  - Usability Patterns
- Assesing Usability in Architecture
- Designing for Usability

Tutorial Area

- Software architecture has, during the last decade, become an area of intense research in the software engineering community
- Architectural analysis of quality attributes aims to bring forward critical decisions that affect different quality attributes to the architectural design phase
- Usability is a quality attribute that can be supported by architecture

Tutorial Sources

- European Research Project:
  STATUS (Software Architecture that supports Usability)
- Duration:
  12/01-12/04
- Partners:
  - Technical University of Madrid (Spain)
  - University of Groningen (Netherlands)
  - Imperial College (England)
  - IHG (Spain)
  - LogicDis (Greece)

Tutorial Goals

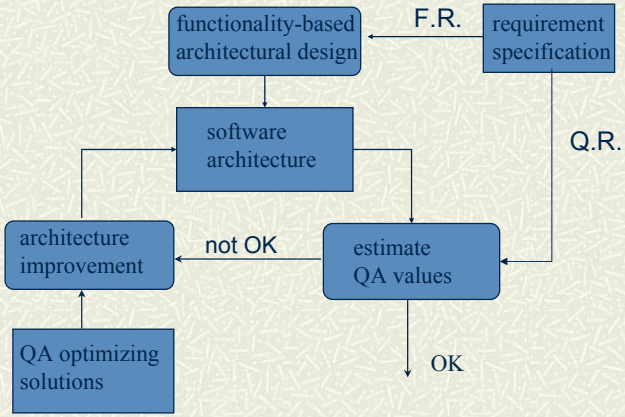
- Usability: the major omission of developers
- Importance of usability
- Basic usability concepts
- Predict usability in the architecture
- Improve design for usability

Why design for usability?

1. Q.A. are restricted by the software architecture → S.A. restricts usability
2. Design decisions in the beginning of the design process are the hardest to revoke

- Need for an explicit design process -

Architecture Design Method

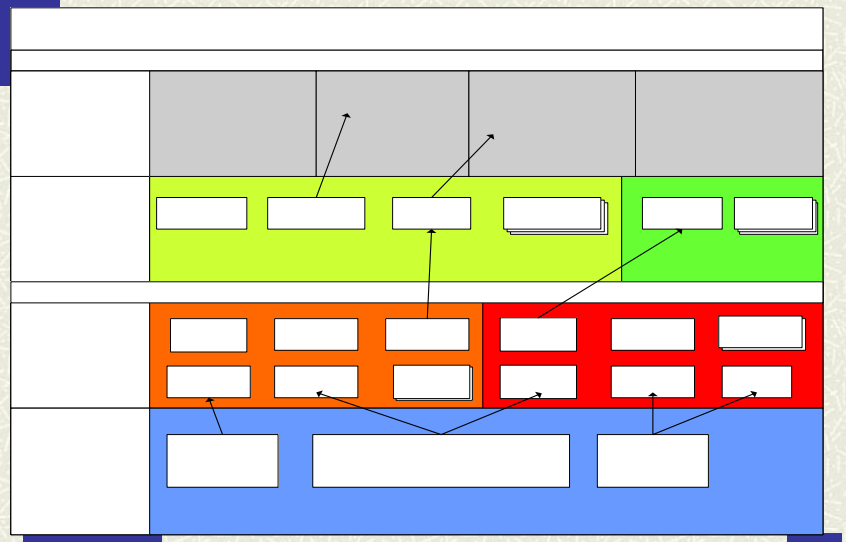


Being able to design for usability

Design process depends on:

- **Determine** when the software design process is finished. (requires assessment techniques)
- Development/identification of design solutions that **improve** usability.

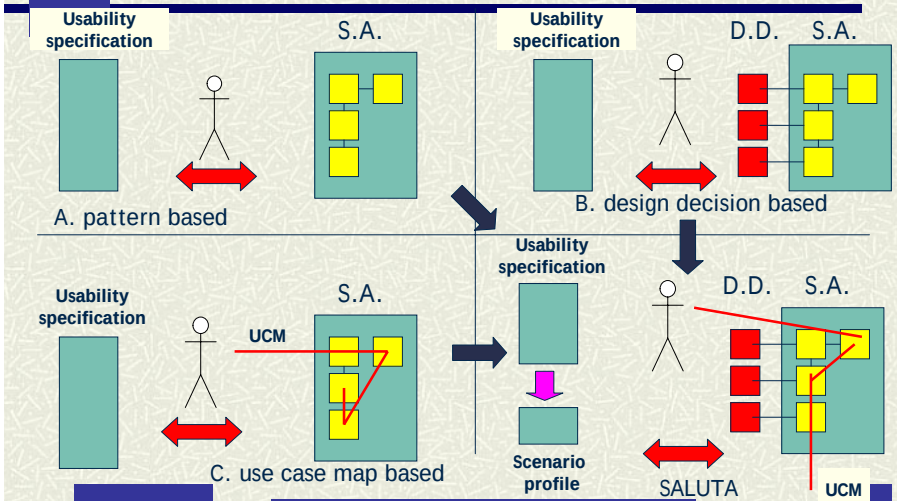
Layered model of usability



ISO

- effectiveness
- efficiency
- learn ability
- retention
- flexibility

Scenario based assessment



Usability patterns

- Progress Indication
- Alerts
- Status Indication
- Undo
- Form or Field Validation
- Preview
- User Profile
- User Modes
- Shortcuts
- Context-Sensitive Help
- Wizard
- Selection Indication
- Cancel
- Macros

Conclusion

- Tutorial introduction
- Introduction to usability and STATUS project
- Usability attributes affected by software architecture
- Specifying usability
- Assessing SA for usability
- Improving SA for usability – usability patterns
- Conclusion