Activity 10-2: Scenarios and Profiles

Why?

Architectural evaluation is difficult because a program's architecture is adequate only if a program embodying the architecture meets its requirements, but an architecture must be judged before such a program exists. Scenarios provide a basis for evaluating architectures before writing any code.

Learning Objectives

- Make a utility tree, form profiles, and write scenarios
- Use scenarios to evaluate architectural alternatives

Success Criteria

- Be able to construct a good utility tree
- Be able to write a scenario description
- Be able to evaluate the ability of architectural alternatives to support scenarios

Resources

ISED section 10.2

Vocabulary

Quality attribute, scenario, scenario description, profile, utility, utility tree, scenario walkthrough

Plan

1. Review *ISED* section 10.2 individually.
2. Answer the Key Questions individually, and then evaluate the answers as a team.
3. Do the Exercise as a team, and check your answer with the instructor.
4. Do the Problems (based on the Case Study) and Assessment as a team.
5. Turn in the Problems and Assessment as a team deliverable.

Key Questions

1. What are quality attributes?
2. What are scenarios and profiles?
3. What is utility tree and how is it constructed?
4. How are scenarios used to evaluate architectural alternatives?

Exercise

ISED Chapter 10 exercise 12

Case Study

The *Madison Design Tool* (MDT) is a CASE tool supporting software engineering design. It must support an end-to-end engineering design process that includes architectural and detailed design. It must generate a Software Architecture Document (SAD), a Detailed Design Document (DDD), and a Design Document consisting of the SAD and DDD.

MDT must run as a web service using a data repository on the client's computer. In other words, the MDT tool must reside at a website and must run in web clients' browsers or special client programs, but the data it manipulates must all reside on the clients' machines.

The format of the data in the repository must be XML.

The MDT must provide editing tools for each of the following: UML Use Case Diagrams, Use Case Descriptions, Box-and-Line Drawings, Textual Interface Specifications, UML Class Diagrams, UML Sequence Diagrams, UML State Diagrams, Operation Specifications, Textual Design Rationale, Decision Matrices, Glossary.

MDT must provide configuration control to the level of individual design models. All versions and revisions of design models and design specifications must be kept in the repository.

MDT must provide a linking mechanism between model elements to support traceability and other connections between model elements. All links must be maintained in the repository.

MDT must generate Java code from design models. Generated code will not be kept in the repository.

The MDT must never lose data saved to the repository or corrupt the repository.

The MDT must be at least as reliable as major object-oriented CASE tools currently on the market.

The MDT is an experimental tool so it must accommodate high rates of change.

The MDT may support simplified versions of notations.

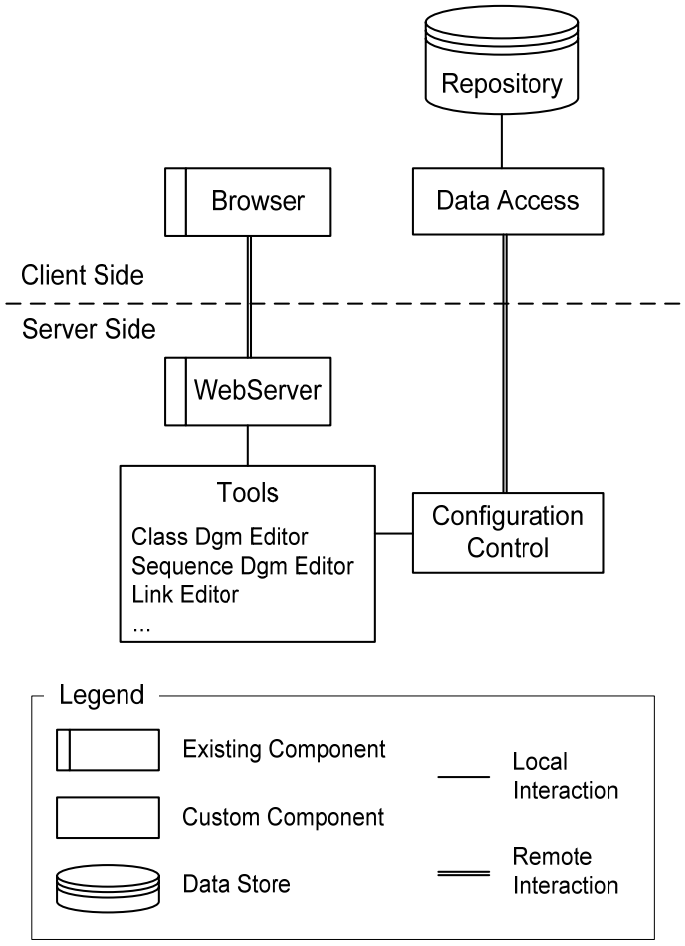
The MDT may provide minimal editing, checking, and support features—in other words, solid basic functioning is much more important than lots of advanced features that do not work reliably.

Problems (Deliverable)

Reminder: Operational quality attributes include reliability, performance, availability, and security. Development quality attributes include maintainability and reusability.

1. Make a list of quality attributes important for the MDT program.

2. Construct a utility tree for MDT using the process described in *ISED* section 10.2. Record and turn in (a) an initial list of scenarios for each profile, (b) a rationalized list of scenarios for each profile, with weights, and (c) the final utility tree.
3. Choose a scenario from your utility tree and write its description.
4. If you have time, walk through your scenario using the candidate architecture below and rate its success in supporting the scenario on a five point scale (five is best).



Assessment (Deliverable)

1. How could you improve your team's performance next time?
2. How could the instructor improve this activity?