Carnegie Mellon University

Software Process Definition – Spring 2002 – Final Project

Architecture Assessment

Process Documentation – version 1.0

Paulo Merson

– 05/08/2002 –

Table of Contents:

1. Introduction

Management of any process that is not described in terms of work products can only be done by mindreaders.
Parnas & Clements [PC86]

1.1 Business Context

The architecture assessment process is used by a *consulting company* specialized in development of enterprise, component-based, web applications. The client for this service typically has to build a complex software system and, in an early stage of the development, use the architecture assessment to validate the decisions being taken in regards to the architecture of the system. This assessment takes place when the functional requirements are already identified and the client has at least a primary view of the architecture.

World-class consultants go to the client's site and work close to the lead architect of the system to evaluate and evolve the architecture. Very often, non-functional aspects of the system, such as performance, security, availability, are not considered until final stages of the development process. Aware of that, this architecture assessment leverages the discussion of these non-functional requirements in the form of quality attributes.

1.2 Objectives

The architecture assessment process has the following objectives:

- Clarify quality attribute requirements for the system.
- Review the architecture of the system and its documentation to provide a solution that can meet functional and non-functional requirements.
- Document architectural decisions and rationale.
- Identify risks early in the life-cycle.
- Increase communication and cooperation among stakeholders.

For the consulting company, the architecture assessment has also the business goal of leveraging the opportunity within the client of contracting the company to follow-on services, typically extending the presence of consultants in the project as part of the architecture and development team.

2. Process Model

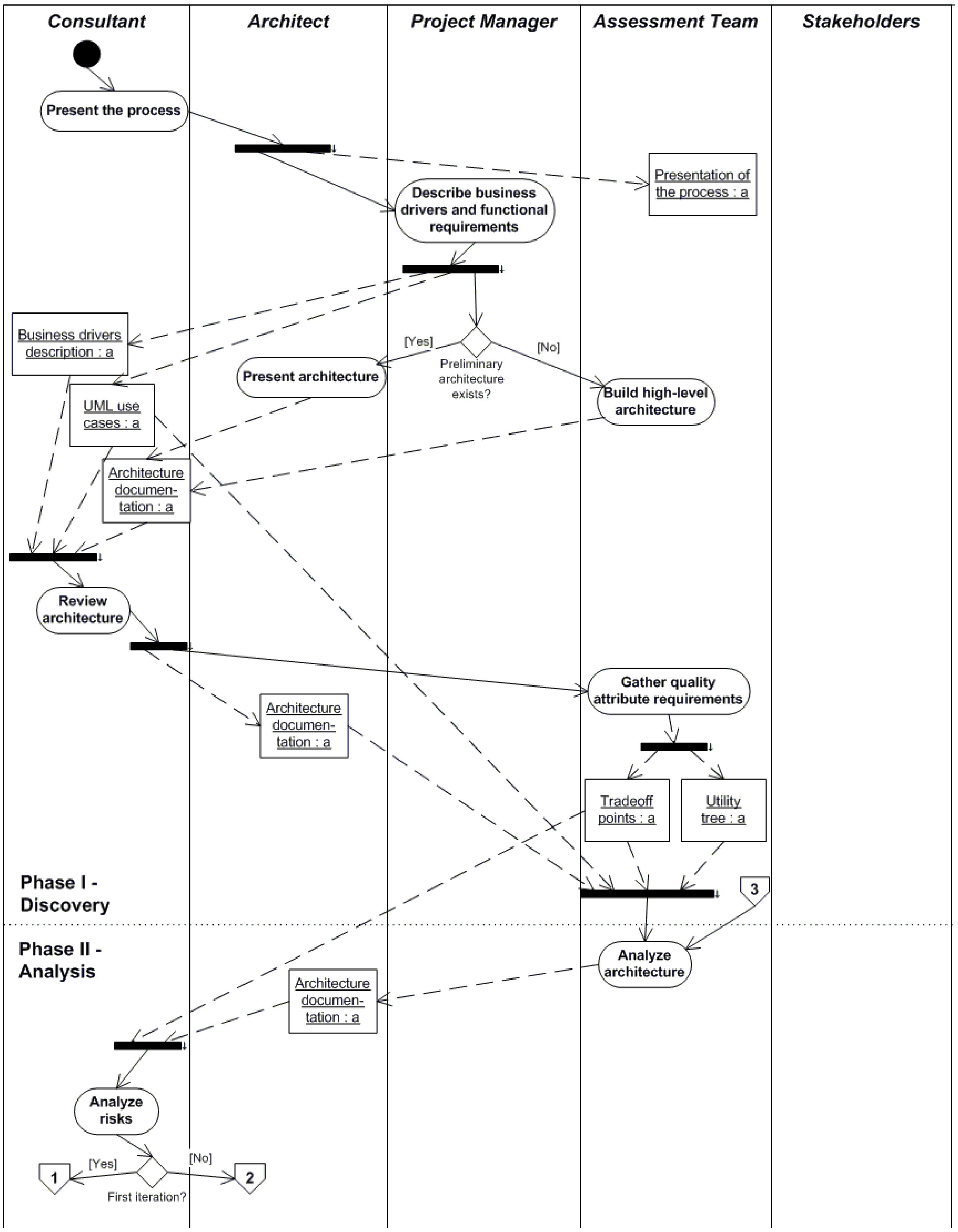
2.1 Activity Diagram

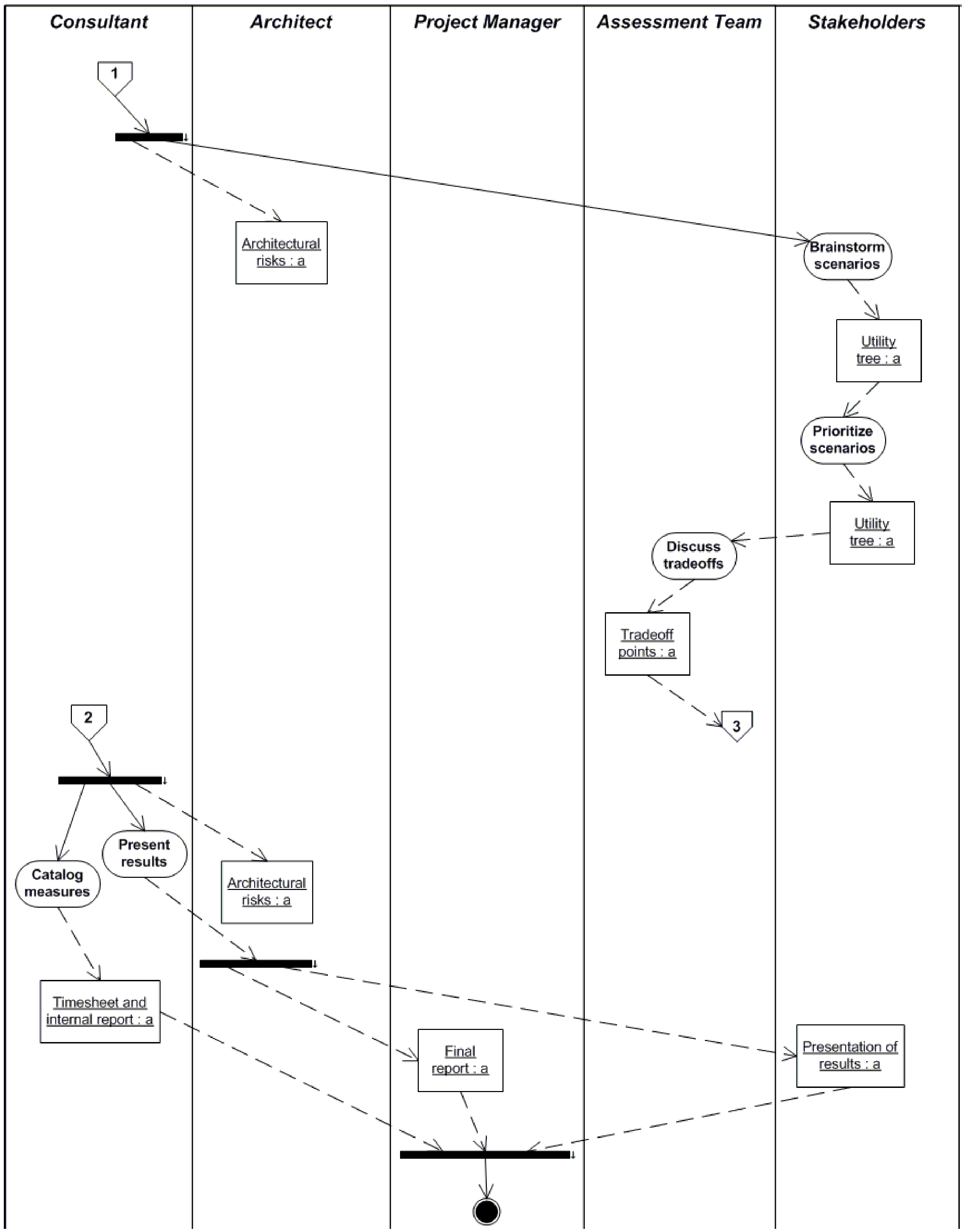
The process is graphically represented in the diagrams on pages 5 and 6. The notation utilized is described in Appendix III.

The process consists of two phases, Discovery and Analysis. Each phase consists of a sequence of activities and each activity consists of a sequence of steps. Phases and activities are represented graphically in the activity diagrams.

Section 3 of this document contains the textual description of the activities, agents and artifacts.

Besides the linearity suggested by the diagrams, in practice the process can recur briefly to an earlier step if minor corrections are needed in previously generated artifacts.


Consultant
Architect
Project Manager
Assessment Team
Stakeholders
Present the process
Presentation of the process : a
Describe business drivers and functional requirements
Business drivers description : a
UML use cases : a
[Yes]
[No]
Preliminary architecture exists?
Present architecture
Build high-level architecture
Architecture documentation : a
Review architecture
Gather quality attribute requirements
Architecture documentation : a
Tradeoff points : a
Utility tree : a
3
Phase I - Discovery
Phase II - Analysis
Analyze architecture
Architecture documentation : a
Analyze risks
[Yes]
[No]
1
2
First iteration?


Consultant
Architect
Project Manager
Assessment Team
Stakeholders
1
Architectural risks : a
Brainstorm scenarios
Utility tree : a
Prioritize scenarios
Utility tree : a
Discuss tradeoffs
Tradeoff points : a
3
2
Catalog measures
Present results
Architectural risks : a
Timesheet and internal report : a
Final report : a
Presentation of results : a

3. Process Definition

3.1 Agents

In the documentation of the activities, we see the following agents:

- ***Consultant***: This agent represents one or two technical consultants that will conduct the architecture assessment. They are senior architects and experts in the software platform that the client intends to use.
- ***Architect***: This agent represents the lead architect or the architecture team that is designing the system for the client. The architects are often employees within the client company, but can be outsourced professionals.
- ***Project manager***: This is the manager of the system development project and is often an employee in the client company.
- ***Assessment team***: This is the team that is performing the architecture assessment. This team basically consists of the consultant, the architect and the project manager. In some situations, the project manager may delegate the responsibility of participating in the assessment team to a senior developer in the project.
- ***Stakeholders***: This is the group of all stakeholders with some interest in the system. Usually they form a heterogeneous group, where we can find: the project manager, the architect, user representatives, manager (business sponsor), system administrator, integrator, tester, database administrator, customer/marketing representative. The consultant is also a member of this group.

3.2 Artifacts

Following is a description of all artifacts consumed or produced throughout the process.

- ***Presentation of the process***: This is a "Power Point" presentation that the consultant gives to describe the architecture assessment process. A template for this presentation is in Section A2.1.
- ***Business drivers description***: Written or verbally communicated list of non-functional factors and requirements of the system. Section A2.2 describes the items that should be present in this list.

- ***UML use cases***: Description of the functional requirements, i.e., the operations the system permits from a user's perspective. Ideally, should use UML use case diagrams along with respective textual descriptions.
- ***Architecture documentation***: This is a living document that evolves over the process. Initially, it contains a high-level view of the architecture, typically consisting of some box-and-line diagrams with no detailed documentation. At the end of the process, the architecture documentation should contain several diagrams, with an element catalog describing the components in each diagram. Depending on the system, different architectural views should be created: run-time view showing components in execution and their communication; code view showing the high-level structure of the source code to be built; deployment view, showing the different processors (machines) and communication protocols. If multiple architectural views are necessary, the documentation should contain the mapping from one view to the other.
  This documentation varies from system to system and this document does not define a template. Nevertheless, the consultant should follow the recommendations in [C+02].
- ***Tradeoff points***: This artifact is a table with tradeoff points and affected quality attributes. Tradeoff points are architectural decisions that affect positively some quality attributes and negatively others. For example, encryption of data transfers increases security but may decrease performance. This artifact is also a living document that is complemented over the process. Initially, it contains the tradeoffs based on quality attribute requirements discussed within the assessment team. At the end, it is completed with tradeoff points originated from the complete prioritized quality attributes, result of discussions with all stakeholders. The consultant should refer to [K+00] for more information about tradeoff points.
- ***Utility tree***: A utility tree is a graphical top-down structure used to organize the quality attribute requirements of the system. A utility tree has 4 levels:
  - The root is "utility".
  - Quality attributes (e.g. performance, availability, modifiability, security).
  - A sub-factor of a quality attribute. For example, performance can be broken down into "latency" and "throughput".
  - Quality attribute scenarios (or simply "scenarios") are the leaves of the tree. A scenario gives a concrete characterization of a quality of the system. For example, for performance/throughput, a scenario could be: "Execute the batch processing of file XYZ in less than 2 hours".

Besides the tree itself, this artifact contains a textual description of each scenario in the tree. As a living document, it will initially contain the assessment team view of the most important scenarios, and at the end of the process it will contain the prioritized scenarios in deliberation of interests of all stakeholders. The consultant should refer to [K+00] for a detailed description of utility trees and templates.

- ***Architectural risks***: This document is a list of risks and respective rationale. Risks can be:
    - Situations where an architectural decision was not taken yet. For example, it is not decided yet if the data store for audit logs will be a relational DBMS or a flat file. The absence of a decision represents a risk because it may affect other quality attributes in the future (e.g. performance, time-to-market).
    - Architectural decisions that are potentially problematic. It typically happens when it is not possible to determine beforehand if a specific part of the architecture will provide the expected results.
    - Non-technical issues, such as misalignment of management team and architecture team, insufficient communication between users and architecture team.
    - New technologies, tools, platforms. Very often, the main motivation to contract the architecture assessment is the fact that the system will be built using technologies that are new to the development team (e.g. the development team has been creating mainframe systems and now will develop a client-server application; or the team is used to creating Visual Basic client-server applications and now will develop a web-based multi-tier application using Java 2 Platform, Enterprise Edition – J2EE). In that case, the consultant must take the responsibility to evaluate how the new technologies fit in the project; he/she must discuss the limitations and emphasize the benefits.
- ***Final report***: This document is where the consultant consolidates the results of the architecture assessment. It may embody or just refer to the architecture documentation, tradeoff points, utility tree and architectural risks. Section A2.3 has a template for the final report.
- ***Presentation of results***: "Power Point" presentation to be given to all stakeholders involved in the assessment. It presents a summary of the findings and next steps for the overall development process. The presentation should be based on the contents of the final report (see Section A2.3).
- ***Timesheet and internal report***: These are internal documents of the consulting company, where the consultant will register the consulting hours, all activities and provide a report describing the progress of the process.

3.3 Activities of Phase I - Discovery

1. Present the process to client

The consultant meets with the project manager and the architect (this group forms the assessment team) to present the architecture assessment process and explain technical concepts involved.

Agent: Consultant.

Entry criteria: contract for the assessment is signed.

Inputs: none.

Steps:

1) Schedule date/time and location of the meeting with project manager, asking him/her to set the meeting with the lead architect and optionally other members of the architecture team. Inform that they should provide all existing architecture and use case documentation of the system.
2) Create the presentation following template in Section A2.1.
3) If this will be the first visit of the consultant to that client, make sure to bring the complete address, directions, contact names and phone numbers.
4) Give the presentation to the assessment team.

Exit criteria: presentation is finished.

Output: Presentation of the process to assessment team.

Purpose: Explain the architecture assessment process to the assessment team in order to set correct expectations, explain the results and leverage the knowledge of the other members of the assessment team in technical concepts that will be exercised during the process.

Estimated time: 3 hours.

2. Describe business drivers and functional requirements

Agent: Project manager.

Entry criteria: Presentation of the process is finished.

Inputs: none.

Steps:

1) Ideally, the project manager will give a presentation to the consultant; most likely the consultant will interview the project manager. Either way, the project manager will pass information about business and functional requirements.

2) For the business drivers, in order to guide the process, the project manager should be asked about all items in Section A2.2.
3) Functional requirements are explained to the consultant, which receives the documentation (usually in the form of UML use cases).

Exit criteria: The consultant will indicate when he/she has all necessary information.

Outputs:

- Business drivers description explained and documentation, if any, handed to consultant.
- UML use cases explained and documentation handed to consultant.

Purpose: Have the consultant aware of business drivers and functional requirements of the system. This knowledge will gradually increase, but it is necessary from the beginning for the consultant to reason about architectural decisions.

Estimated time: 1 hour.

3. Present architecture

Agent: Architect.

Entry criteria: Description of business drivers and functional requirements to consultant is finished, and there exists a documented version of the architecture.

Inputs: none.

Steps:

1) The architect will explain to the consultant the current version of the architecture. At this point, the consultant should not criticize or try to fix the architecture if it has defects; he/she should just try to get a deep understanding the architecture.
2) The level of detail presented by the architect in this step varies. Frequently, the consultant will need to specify to the architect the information that should be presented. At a minimum, the architect should be able to explain the following items:
    - Technical constraints of the execution environment of the system, such as hardware, OS, application server, database server, network connections.
    - External systems with which the system will interact.
    - High level architecture outline of the system.
3) The architect will hand all architecture documentation to the consultant.

Exit criteria: The architect finishes the explanation and hands the documentation to the consultant.

Output: Preliminary version of the architecture documentation presented and handed to consultant. (This documentation was produced previously to the start of the assessment process.)

Purpose: Very often, the client has a preliminary version of the architecture, which is normally used as the start point for the creation of the final (assessed) architecture. In this activity, the consultant can get acquainted with this preliminary architecture documentation.

Estimated time: 1.5 hours.

4. Build high-level architecture

Agent: Assessment team.

Entry criteria: Description of business drivers and functional requirements to consultant is finished, and there is no current documented version of the architecture.

Inputs: none.

Steps:

1) The consultant will guide the group in the creation of a preliminary high-level view of the architecture.
2) Using a white board, the consultant will try to delineate the components that should be the basis for the system. Interactively, the other members of the team will assist in the creation of the architecture. Sometimes, the architect has in mind a sketch of the architecture and this step should be used to materialize it.
3) The consultant takes note of the envisioned architecture.
4) Afterwards, the consultant individually creates the architecture diagram(s) and documentation using the appropriate tool (e.g. Microsoft Visio).
5) The consultant sends this document to the other members of the assessment team for review.

Exit criteria: The discussed architecture is documented and reviewed by the members of the assessment team.

Output: Preliminary version of the architecture documentation, as produced by the team and documented by the consultant.

Purpose: Create a start point architecture that will be gradually reviewed according to requirements and constraints to be discussed.

Estimated time: 3 hours for the meeting; 1 day for the consultant to write the documentation.

5. Review architecture

The consultant will now study the system requirements and the architecture documentation, checking if basic characteristics of good architecture documentation are being used.

Agent: Consultant.

Entry criteria: First round of meetings with the assessment team, project manager and architect is finished. The consultant received all documentation necessary to effectively begin the assessment.

Inputs:

- Business drivers description
- UML use cases.
- Architecture documentation (copy of the files so that the consultant can edit).

Steps:

1) The consultant reads all documentation looking for missing pieces or conceptual faults.
2) The architecture documentation is validated against recommendations in Appendix I. If necessary, the consultant complements the documentation or delineates in the document the points that need to be complemented by the architecture team. The goal is to fix just grave defects; the consultant should not spend much time here to validate in detail if the architecture is a good solution for the requirements. In following activities it will happen.
3) If the architecture documentation contemplates more than one candidate architecture for the system, all of them must be studied and verified against recommendations in Appendix I, unless a candidate architecture is notably faulty, in which case it can be discarded right away. Note: if a candidate architecture is discarded, it remains in the architecture documentation and the consultant writes a justification for the fact that it is not a good option.
4) The consultant must also review and evaluate if services provided by the target execution environment are being utilized correctly in the architecture. For example, if the system will run on a J2EE application server, the architecture should take advantage of provided services, such as authentication and authorization (security), pooled database connectivity and cluster of machines.
5) The consultant sends the reviewed architecture documentation to the other members of the assessment team.
6) The consultant calls the project manager to schedule a meeting with the assessment team, giving them a couple of days to review the new version of the documentation.

Exit criteria: The review is finished and an updated version of the documentation is sent to the assessment team.

Output: Preliminary version of the architecture documentation as originally produced by the architecture team and reviewed by the consultant.

Purpose: let the consultant understand the proposed architecture and fix major defects. The objective is not to analyze if the architecture will fulfill the requirements or not.

Estimated time: 1 week.

6. Gather quality attribute requirements

Agent: Assessment team.

Entry criteria: Consultant has delivered the first review of the architecture documentation.

Inputs: none.

Steps:

1) The consultant explains in more detail the concepts of quality attributes, scenarios, tradeoff points and utility tree. For details of these concepts, see [K+00] and [BCK98].
2) To guide the discussion, the consultant presents a list of quality attributes (e.g. performance, availability, security, modifiability, time-to-market), which can be extracted from [K+00], [BCK98] or other good source. There is a finite (but large) number of quality attributes or "ilities", and terminology and interpretation varies in the literature and among professionals. The consultant must follow a guideline to exhaust the quality attribute discussion. The objective of the list is to present a minimal but comprehensive list that the consultant must go through to check which ones are relevant for the system at hand. And the list avoids the need to remember all quality attributes – we don't want to find out two weeks later that we forgot to discuss security, for example.
3) For each quality attribute, the team discusses which scenarios are relevant to the system. To analyze if an architecture can meet quality requirements, those requirements need to be expressed in terms that are concrete and measurable or observable. The scenarios, characterized by a stimulus and quantifiable response, are the mechanism to express the quality attribute requirements. For example, instead of saying "this system is critical and has strong availability requirements", a scenario would define that "the system should detect and recover from hardware failures in 99.95% of the times". Once again, the consultant must refer to [K+00] and [BCK98] for a complete discussion of quality attribute scenarios.
4) Having all scenarios annotated, they are organized in a utility tree.
5) Then the group discusses which scenarios are conflicting to produce tradeoff points. Very often, a strong requirement for one quality attribute affects negatively another quality attribute (in situations where no other "technical" attribute is affected, at least the "business"attributes of time-to-market and cost are normally affected).
6) After the meeting, the consultant must write down the tradeoff point analysis and the utility tree with a textual description of the scenarios.
7) The documents are sent to the other members and another meeting is scheduled.

Exit criteria: All attributes were discussed, the utility tree was created and tradeoff points were listed

Outputs:

- Tradeoff points.
- First version of the utility tree along with a textual description of the scenarios.

Purpose: Identify the quality attributes that are important to the system and characterize them in an objective and measurable way.

Estimated time: 4 hours for meeting, 1.5 days for the consultant to consolidate the documentation.

3.4 Activities of Phase II - Analysis

7. Analyze architecture

This activity occurs twice in the process. The first iteration is after the discussion of quality attributes with the assessment team and the second iteration is after the discussion with all stakeholders.

Agent: Assessment team.

Entry criteria: A discussion of quality attribute requirements and tradeoffs – either within the assessment team or with all stakeholders – is finished.

Inputs:

- Architecture documentation.
- Tradeoff points.
- Utility tree.
- UML use cases.

Steps:

1) If in the first iteration, the consultant and architect will validate the architecture against the functional requirements (UML use cases). This step must produce a mapping of use cases to architectur components, as well as a revised version of the architecture. This step is critical to the architecture assessment, because it validates the completeness of the use case specifications and the architecture that will realize them.
2) If in iteration1 or 2, consultant and architect discuss how the architecture addresses each of the scenarios. For example: does the architecture ensure performance requirements dictated by scenario "the system must verify if the user has authorization to execute an operation it in less than 0.5 seconds"? This analysis is based on the experience of architect and consultant and should not be a deep formal analysis. The idea is to perform a back-of-the-envelope analysis trying to identify points in the architecture that can be problematic to fulfill the requirements. Note: if in iteration 2, only the scenarios with high importance are considered.
3) At this point, components and connectors are likely to be added or modified in different views of the architecture. It is possible that more than one candidate architecture be created for some parts of the system. Also, it is possible that new conclusions become necessary in the tradeoff points document.
4) If in iteration 2, a deeper analysis can be done for important quality attributes: the consultant must identify pieces of the architecture that impacts a quality attribute (sensitivity points), try to identify the architectural style is being used in that piece, and apply an analysis method that is suitable for that style and that quality attribute. Examples of analysis methods include: rate monotonic analysis for performance, Markov modeling for reliability in redundant systems, and queue theory for performance. This type of formal analysis is time-consuming and should be performed only in situations where the system has very strict requirements (e.g. a nuclear plant real time system).
5) After the meetings, the consultant will consolidate the conclusions in the architecture documentation and send the document to the other members.

Exit criteria: The analysis of the architecture is concluded.

Output: Architecture documentation reflecting decisions based on requirements of the system.

Purpose: Analyze if the proposed architecture is adequate and fulfills all functional requirements and high priority quality attribute requirements.

Estimated time: 1 day for meetings, a variable number of days for the consultant to consolidate the documentation (depends on the complexity of the analysis).

8. Analyze risks

This activity also occurs twice in the process, following the “analyze architecture” activity.

Agent: Consultant.

Entry criteria: A new version of the architecture documentation was produced reflecting functional and non-functional requirements.

Inputs:

- Architecture documentation.
- Tradeoff points.

Steps:

1) The consultant will study the tradeoff points and the architecture, trying to identify risks to the project at the architectural level.
2) The risks are described in a separate document. For each risk, the consultant should indicate actions to mitigate it, even if they will not be part of the plan. This task may require some technical research.
3) The consultant sends the document to the other members of the assessment team and schedule a meeting with all stakeholders.

Exit criteria: The analysis of architectural risks is complete

Output: Architectural risks.

Purpose: The analysis of the architecture is not definite and conclusive in all details. Many architectural decisions pose a risk due to the uncertainty of their effectiveness. This type of risk must be elicited.

Estimated time: from 1 to 3 days.

9. Brainstorm scenarios

Agent: Stakeholders.

Entry criteria: A first analysis of the architecture and risks is finished by the assessment team.

Inputs: none.

Steps:

1) The consultant gives a short presentation for all stakeholders describing the process. This presentation is a short version of the template in Section A2.1 (software architecture concepts are omitted).
2) The consultant reviews previously discussed scenarios showing the current version of the utility tree.
3) To facilitate the discussion, the consultant explains that there are three types of scenarios that can be considered in this step:
    - ***Use case scenarios***: typical way in which the system is expected to be used. Examples: “The user wants to be able to cancel the emission of a report after clicking the ‘print’ button”; “if an exception occurs in the server module, an e-mail is sent to administrator and requests are redirected to another processor”.
    - ***Growth scenarios***: these scenarios anticipate a change to the system that can be necessary in the near term. Examples: “The system must use a different relational DBMS”; “the number of users increases 100% and latency (response time) in user operations remains the same”.
    - ***Exploratory scenarios***: the objective of these scenarios is to expose the system to stress conditions and limit situations. Some of these conditions may not be realistic now but can occur in the future. Examples: “Migrate the system from Microsoft platform to Java 2, Enterprise Edition (J2EE)”; “provide 99.995% availability”.
4) Based on the existing scenarios, the group will try to enrich the utility tree with new scenarios in a brainstorm process (uncensored and creative generation of ideas, followed by a filter process to mine the best ideas).
5) In practice, during the brainstorm, the stakeholders indicate their concerns about the system in terms of quality attributes. The consultant tries to normalize a concern in the form of a scenario. For example: the system administrator says that “availability” is a concern. The consultant nails down this generic concern to specific scenarios, such as: “the system must detect and recover from a software failure to keep 99.8% of availability”. (A strong requirement like this might motivate the architectural decision of using a cluster of servers in a further analysis.) Commonly, each stakeholder will bring up concerns that are somehow related to his/her role in the project. Thus, the business sponsor of the project will call for time-to-market and cost, the user representative for usability, the administrator for security and maintainability, and so on.
6) All scenarios are annotated and the secretary of the meeting prints a version of the complete utility tree to everyone.
7) The consultant explains that after a short break the group will start the next activity – discuss priority of the scenarios.

Exit criteria: The group has exhausted all relevant ideas.

Output: Utility tree with new scenarios.

Purpose: This activity creates an opportunity for all stakeholders interested in the system to manifest their concerns in terms of requirements that are translated to tangible scenarios.

Estimated time: 3 hours.

10. Prioritize scenarios

Agent: Stakeholders.

Entry criteria: The brainstorm of scenarios is finished and a printed version of the utility is handed out to everyone in the meeting.

Input: Utility tree.

Steps:

1) First, the consultant and the architect go through all scenarios giving an approximation of complexity using a high/medium/low scale. Complexity is the level of perceived difficulty to implement the proper treatment for that scenario in the system. They should not go into technical details to explain why scenario XYZ is complex while another scenario is simple.
2) Then, each stakeholder is allocated a number of votes equal to 30% of the number of scenarios. For example, if there are 20 scenarios, each stakeholder gets 6 votes.
3) Before voting, some time is reserved for one person to advocate the importance of each scenario, if stakeholders are interested on that.
4) Then the stakeholders allocate their votes to scenarios in any way they want.
5) The votes are tallied and the scenarios are stratified in a high/medium/low scale for importance based on the number of votes.
6) Finished the voting process and the stakeholders meeting, the consultant will consolidate the results in a new version of the utility tree, where the low priority scenarios are removed. This new version is sent to other members of the assessment team and another meeting of this team is scheduled.

Exit criteria: All scenarios in the utility tree have a priority in terms of importance and complexity.

Output: Prioritized utility tree with high/medium/low classification for importance and complexity.

Purpose: The objective is to create a consensual filter of the most important scenarios, recognizing that not all of them can be implemented.

Estimated time: 2 hours.

11. Discuss tradeoffs

Agent: Assessment team.

Entry criteria: Prioritized utility tree is ready after discussion with all stakeholders.

Input: Utility tree.

Steps:

1) The assessment team goes again through each scenario to identify and discuss the tradeoffs of each one.

2) Tradeoff points are annotated by the consultant. Now with the prioritized scenarios, the tradeoffs will allow reasoned architectural decisions in the subsequent activity – analysis of the architecture.
3) The consultant again consolidates the results and sends them to the other members.

Exit criteria: Possible tradeoffs of all important scenarios are discussed.

Output: Tradeoff points reviewed.

Purpose: Identify conflicting scenarios and discuss the balance point between them to best respond to the stakeholders' expectations.

Estimated time: 3 hours.

12. Present results

Agent: Consultant.

Entry criteria: The final analysis of the architecture and risks is finished.

Inputs: none.

Steps:

1) The consultant first creates the final report following the template in Section A2.3. To produce the closing part of the report where there is a description of how the consulting company can assist the client in the next phases of the software project, the consultant must discuss with his/her manager.
2) The final report is sent to the project manager and a presentation is scheduled with all stakeholders. It is the project manager's responsibility to forward the final report to the other members of the assessment team and it is his/her choice to forward the report to all stakeholders.
3) The consultant will give a presentation to all stakeholders discussing the results of the architecture assessment.
4) If the client becomes interested in a continued assistance of the consulting company in the development of the system, the consultant indicates that interest to his/her manager, so that that possibility is discussed in the near future with the project manager.

Exit criteria: Final report delivered to project manager and presentation of results given to stakeholders.

Outputs:

- Final report.
- Presentation of results.

Purpose: Finish the process by wrapping up all deliverables in a consistent and organized way.

Estimated time: from 2 to 5 days to elaborate the final report and presentation; 1.5 hours for the presentation.

13. Catalog measures

Agent: Consultant.

Entry criteria: The final analysis of the architecture and risks is finished.

Inputs: none.

Steps:

1) The consultant consolidates in the timesheet the records of hours spent in each activity.
2) The consultant writes a report describing the progress of the process and how it fit in the process definition of the architecture assessment. He/she should indicate any suggestions and problems encountered.
3) These internal documents are submitted to the consultant's manager.
4) A copy of the report is submitted to the group responsible for the architecture assessment process definition in the consulting company.
5) The timesheet is sent to the finance department for billing purposes.

Exit criteria: Timesheet and internal report sent to manager.

Output: Timesheet and internal report.

Purpose: Evaluate the process for continuous process improvement and generate measures for history of productivity and billing purposes.

Estimated time: 2 days.

A1. Appendix I – Architecture Best Practices

The role of software architect requires years of experience in software development, good abstraction capabilities and deep knowledge in various technologies. This section has no ambition to give a roadmap to verify the correctness and consistence of an architecture. Each system may present a completely different architectural context and it is up to the architect to see the strengths and weaknesses of the solution being proposed in each case.

At a minimum, this section prescribes some items that must be present in the architecture documentation. To a detailed study on this subject, please refer to [K+02].

A1.1 Checklist of good architecture documentation

A good architecture documentation will follow as much as possible these recommendations:

- Definition of multiple views.
- Mapping of components from one view to the other.
- Multiple levels of abstraction (master-detail or hierarchical view of components).
- Diagrams must use consistent boxology. The same symbols should be used to represent the same types of things across multiple diagrams.
- Diagrams must contain legend identifying types of components and connectors.
- For each diagram, there must be an element catalog, i.e., a table with textual description for each component in the diagram. If necessary, the architect may include description of the connectors.
- If the system interacts with several external entities, a context diagram must be built.
- If the system is deployed on multiple machines (e.g. multi-tier web application that involves browser, http server, application server, database server, ERP server), then a deployment view must be built.
- The use of architectural styles should be identified in the documentation.
- Architectural decisions (accepted and rejected ones) must be documented with rationale.

A2. Appendix II – Templates

A2.1 Template for "Presentation of the process"

- Short description of the architecture assessment process, displaying the activity diagrams and talking about the activities in brief.
- Explain that besides functional requirements, quality attribute requirements may represent an equally difficult challenge to the development of a software system. Thinking of quality attributes early in the development process has paramount importance for the success of the project. Use the following quote:

  > *A system is motivated by a set of functional and quality goals. For example, if a telephone switch manufacturer is creating a new switch, that system must be able to route calls, generate tones, generate billing information and so forth. But if it is to be successful, it must do so within strict performance, availability, modifiability, and cost parameters. The architecture is the key to achieving – or failing to achieve – these goals.* [K+00]
- Short description of artifacts: architecture documentation, utility tree, tradeoff points, architectural risks.
- Concepts: software architecture concepts, components and connectors, architectural views, architectural styles.
- Definition of the agenda for Phase I of the assessment.

A2.2 Template for "Business drivers description"

The consultant must ask for information about the following non-technical requirements of the system:

- Business environment and history of the project.
- Stakeholders of the system.
- Time-to-market constraints.
- Qualities of competitors' products and market differentiators this system is pursuing.
- Description of the budget, resources (team formation) and schedule of the development project.

Other global aspects of the system should also be elicited here:

- Necessity of adherence to standards.

- Interoperation with other systems.
- Required hardware and software platform for execution.
- Software development platform (programming language, DBMS, application server, J2EE or Microsoft .NET, etc.).
- Skill set of the development team.
- Reuse of legacy code.
- Use of COTS products and components.

A2.3 Template for "Final report"

- Cover page with system and client names, title ("Architecture Assessment Final Report"), date.
- Executive summary containing most important findings in terms of architectural decisions and risks.
- Assessment team description.
- Name of stakeholders that participated in the process.
- Summary of activities performed.
- The architecture documentation or reference to external document.
- The utility tree, scenarios and tradeoff points, or reference to external document.
- Architectural risks analysis or reference to external document.
- Conclusions for the next steps in the overall development process, based on the assessment.
- How the "consulting company" can assist the client in the rest of the development process. (To produce this part of the report, the consultant must talk to his/her manager.)

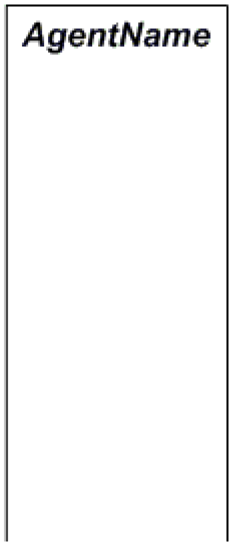
A3. Appendix III – Notation of Diagrams

The diagrams that describe process workflow use the notation of UML Activity Diagrams [RJB99]. The two major differences from the original UML notation are "off page reference" and the "phase boundary". The full notation and boxology is described in what follows.

A3.1 Entities of the Process

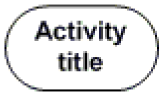
Agents

The agents are primitive entities of the process and are described in separate in the Process Definition Section. They are identified by "swimlanes" in the diagrams:



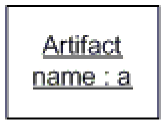
Activities

The activities are primitive entities of the process and are described in separate in the Process Definition Section. They are boxes with rounded ends in the diagrams:



Artifacts

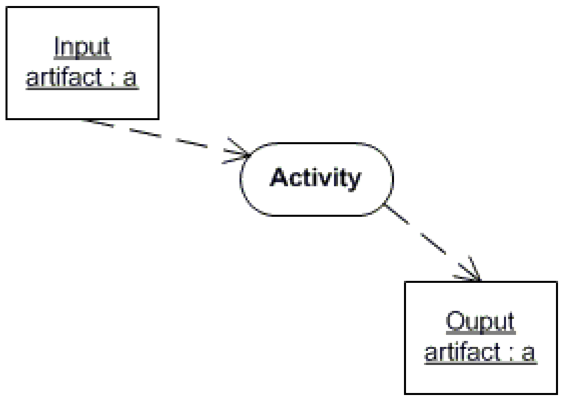
The input and output artifacts are primitive entities of the process and are described in separate in the Process Definition Section. In the diagrams, they are rectangles and the name of the artifact is underlined and followed by "<u>: a</u>", to indicate that it is an object of type "a" (artifact):



A3.2 Relationships Between Entities

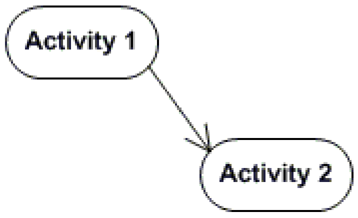
Input and output artifacts of an activity

A dashed arrow from an artifact to an activity indicates that it is an input, and from the activity to an artifact indicates it is an output:



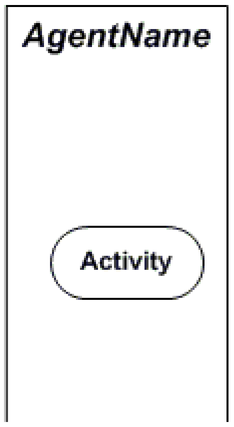
Activity flow of control

A solid arrow indicates that one activity follows the other in the process execution:



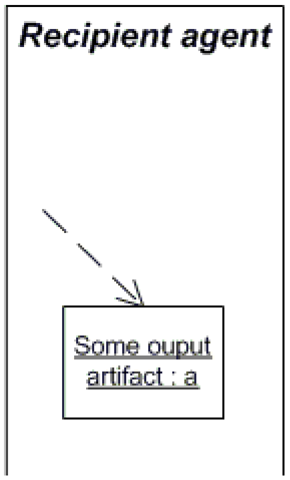
Agent responsible for an activity

The placement of an activity in the swimlane of an agent indicates that that agent is responsible for performing that activity:



Agent receives an output artifact

The placement of an output artifact in the swimlane of an agent represents that that agent is the consumer of that artifact. (Note: this semantic is not clearly defined in the UML notation.)

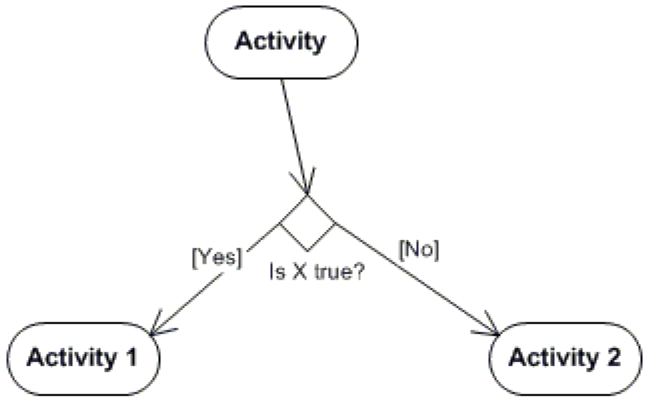


A3.3 Other Symbols

Conditional branch

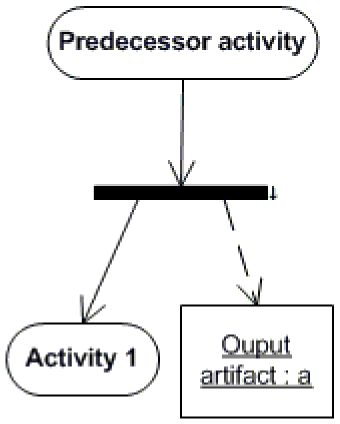
A conditional branch is represented by a diamond labeled with a conditional expression. The two or more exit arrows are flow of control solid arrows that go to activities (or synch bars with an exit arrow to an activity).

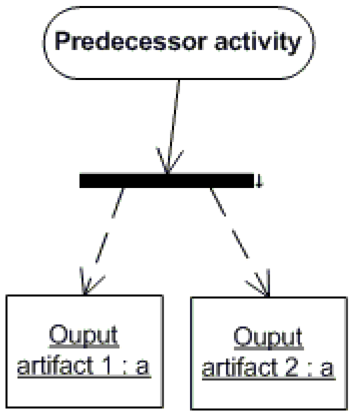
These arrows are labeled with the so called “guard condition”. Not all of the exits are executed (the diamond is an “or”):

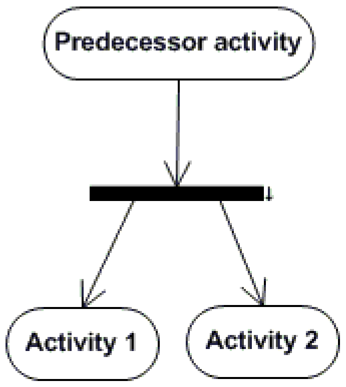


Synch bars

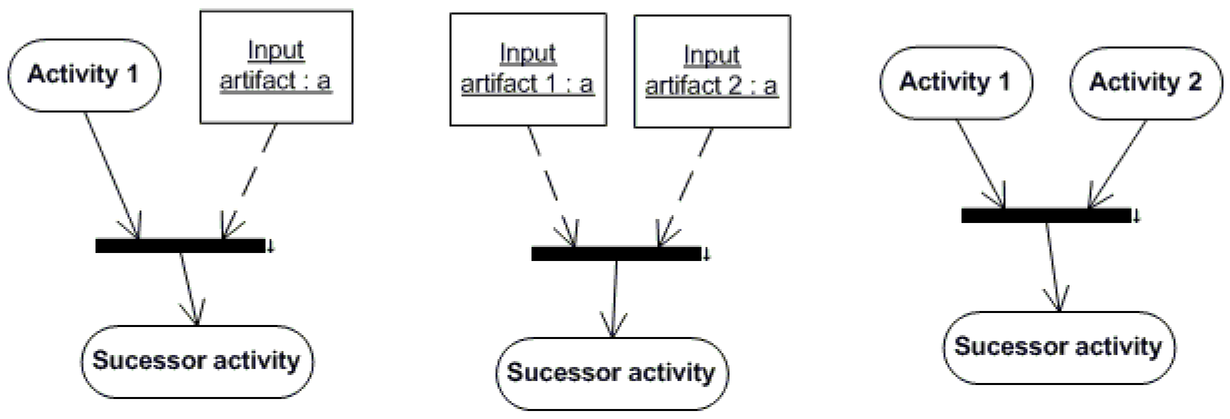
A “synch bar” is used when there are two or more arrows exiting from an activity – this is a *fork* synch bar. The outgoing arrows of a fork synch bar can be flow of control (solid arrow) and/or output artifact flow (dashed arrow). If there are two or more flow-of-control arrows exiting the bar, the two or more successor activities may be executed in parallel, and all of them are executed (the fork bar is an “and”). Examples:





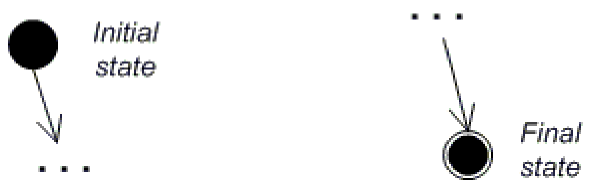


The other type of synch bar is used when there are two or more arrows entering an activity – this is a *join* synch bar. The incoming arrows in a join synch bar can be flow of control (solid arrow) and/or input artifacts flow (dashed arrow). The single exit arrow is an activity that is only executed when all previous activities are finished and input artifacts are available. Examples:



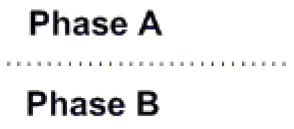
Initial and final states

These are “pseudo-activities” that are used to indicate the beginning and termination of the process:



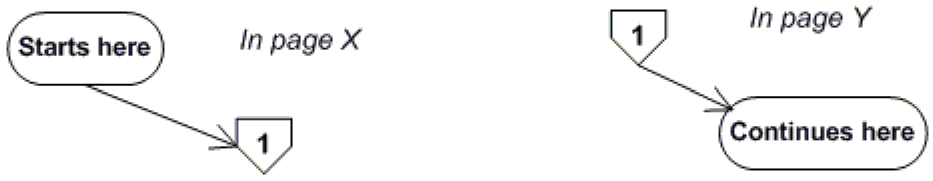
Phase boundary

A dotted line is used to represent graphically the separation of activities in phases of the process. The phase boundary representation is not part of the UML activity diagram notation. The process model is represented in three hierarchical levels: phases, activities, steps. The steps within a process are not represented graphically; they are documented in the activity textual description instead.



Off page reference

This symbol is used to link diagrams in separate pages. It does not contain any semantic; it is just a graphical simplification. This symbol is not part of the UML activity diagram notation. The off page reference symbol with an incoming arrow indicates that the actual pointed entity is in a separate page that is identified by the same symbol with the same numeric label.



References

[BCK98] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice. Addison-Wesley. 1998.

[C+02] Clement et. al. Documenting Software Architectures: Views and Beyond. Addison-Wesley 2002.

[K+00] Kazman, Klein, Clements. ATAM: Method for Architecture Evaluation. Software Engineering Institute Technical Report, CMU/SEI-2000-TR-004, August 2000.

[PC86] David Parnas & Paul Clements. A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering, v. SE-12 n. 2, pp. 251-257, 1986.

[RJB99] J. Rumbaugh, I. Jacobson, G. Booch. The Unified Modeling Language Reference Manual. Addison Wesley, 1999.