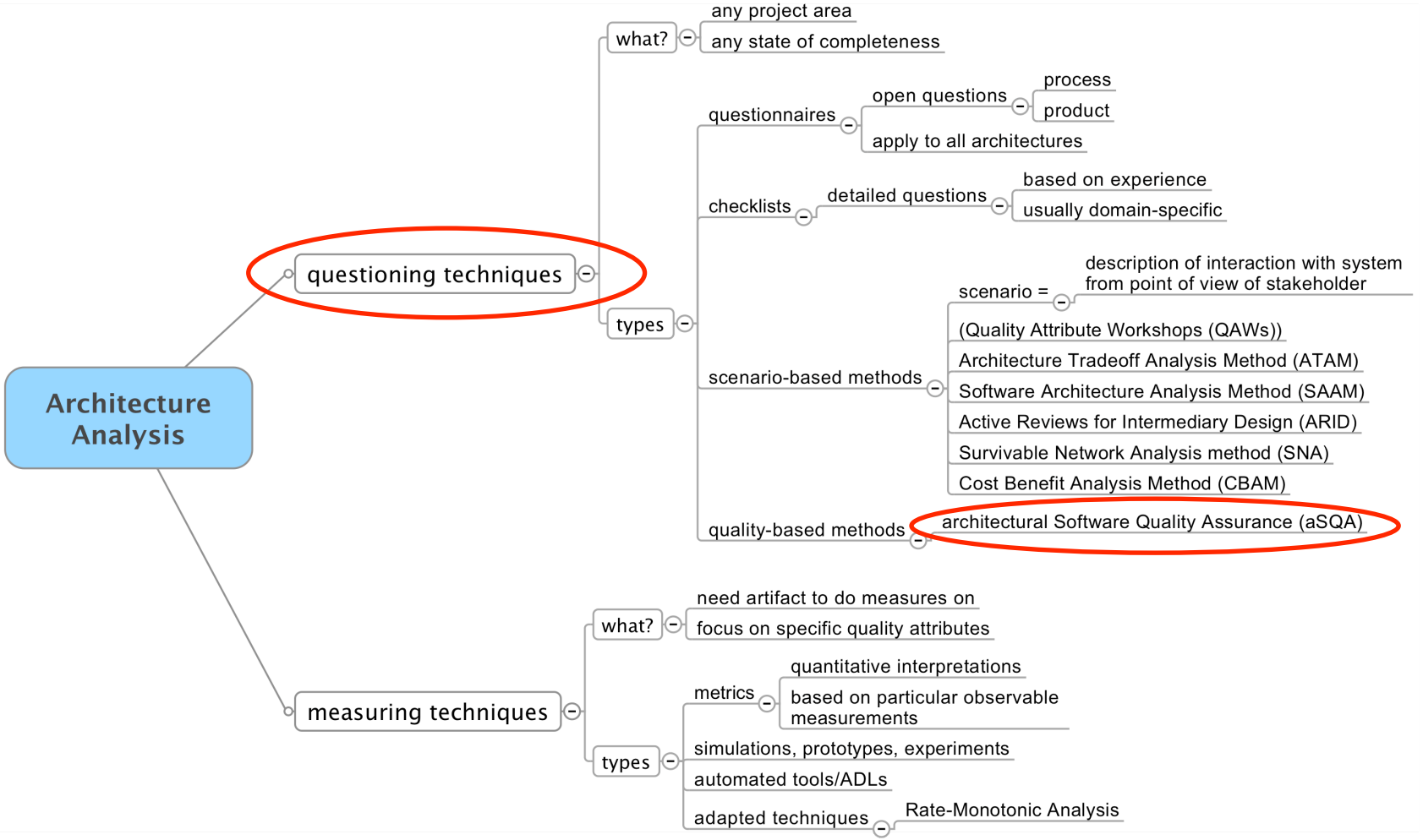
AARHUS UNIVERSITET

Architecture Evaluation

Architectural Software Quality Assurance (aSQA)

Overview


Architecture Analysis
questioning techniques
what?
any project area
any state of completeness
types
questionnaires
open questions
process
product
apply to all architectures
checklists
detailed questions
based on experience
usually domain-specific
scenario-based methods
scenario =
description of interaction with system from point of view of stakeholder
(Quality Attribute Workshops (QAWs))
Architecture Tradeoff Analysis Method (ATAM)
Software Architecture Analysis Method (SAAM)
Active Reviews for Intermediary Design (ARID)
Survivable Network Analysis method (SNA)
Cost Benefit Analysis Method (CBAM)
quality-based methods
architectural Software Quality Assurance (aSQA)
measuring techniques
what?
need artifact to do measures on
focus on specific quality attributes
types
metrics
quantitative interpretations
based on particular observable measurements
simulations, prototypes, experiments
automated tools/ADLs
adapted techniques
Rate-Monotonic Analysis

Motivation

How to continuously asses health of architecture and implementation?

- Architecture may change frequently
- Evaluation methods may be expensive to apply

Architecture Software Quality Assurance (aSQA)

AARHUS UNIVERSITET

Architecture- and quality attribute-driven approach to architectural evaluation

- Divide architecture into component
- Choose quality attribute framework, e.g., from [Bass et al., 2003]

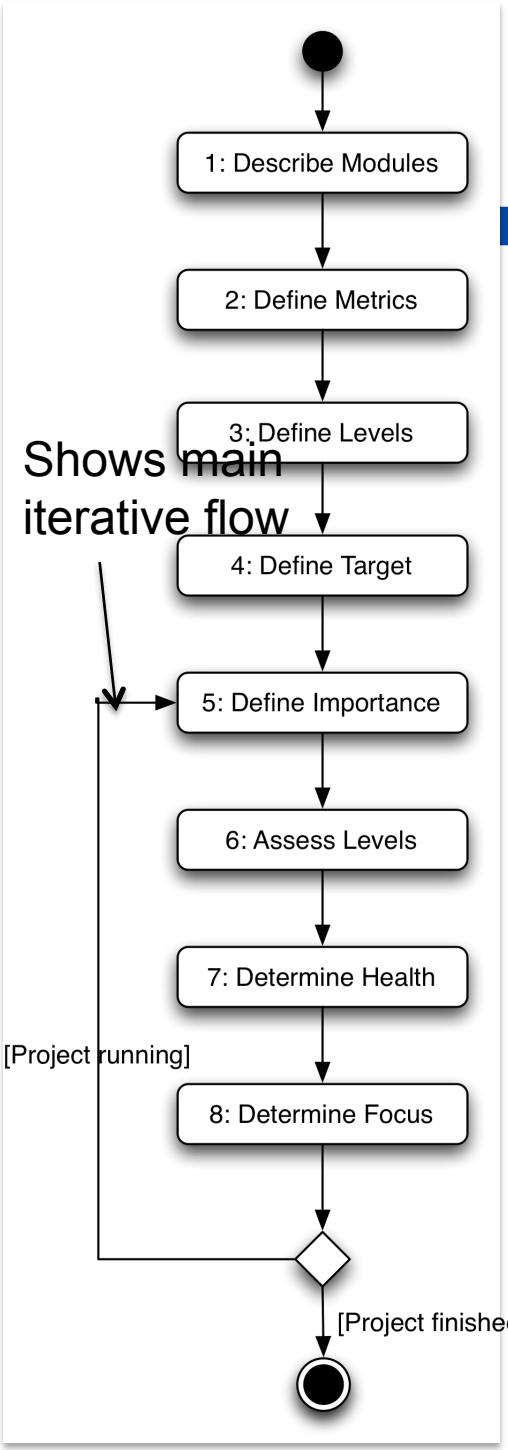
Assess each component

- Current, goal, and health level
- Combine to system-level assessment

Warning

- Research in progress ☺!

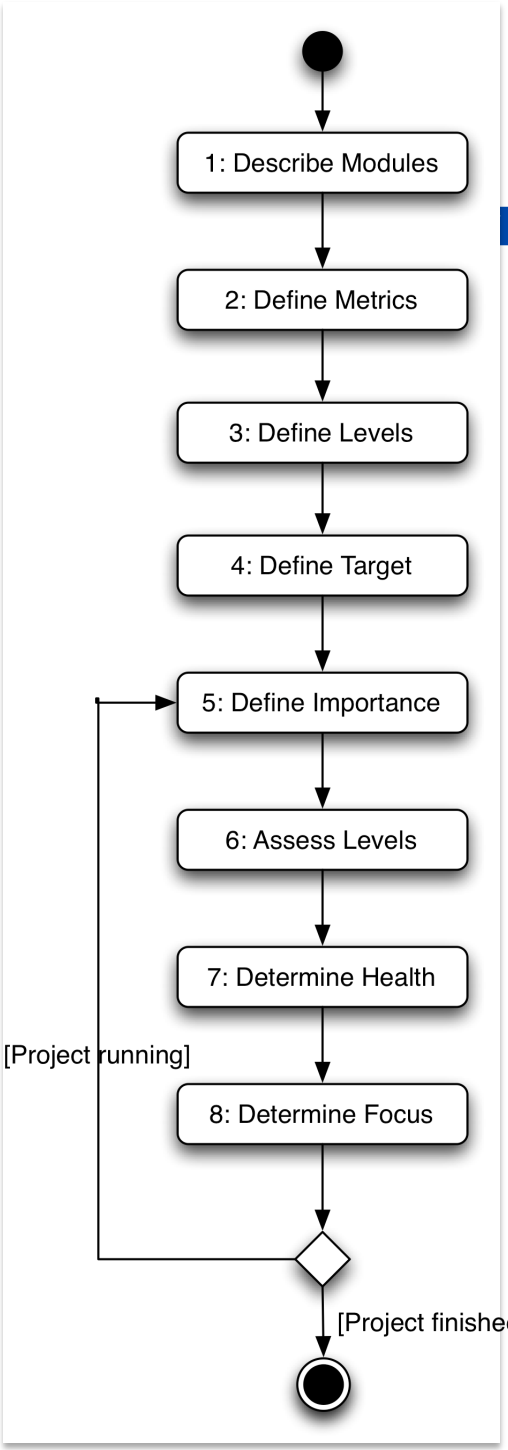
aSQA Steps



ERSITET

1. Describe Components
   - Describe the component structure of the system
   - May be used for work assignment
2. Define Metrics
   - Given a quality framework, what is to be measured for each module for each attribute?
   - May be scenario-based
3. Define Levels
   - How do measurements (made using metrics) map to levels?
4. Define Target
   - Define which level must be reached for each component and attribute

aSQA Steps



ERSITET

5. Define Importance

- Set the importance level of each attribute for each component
- Important in prioritization

6. Assess Levels

- Use metric to determine current levels

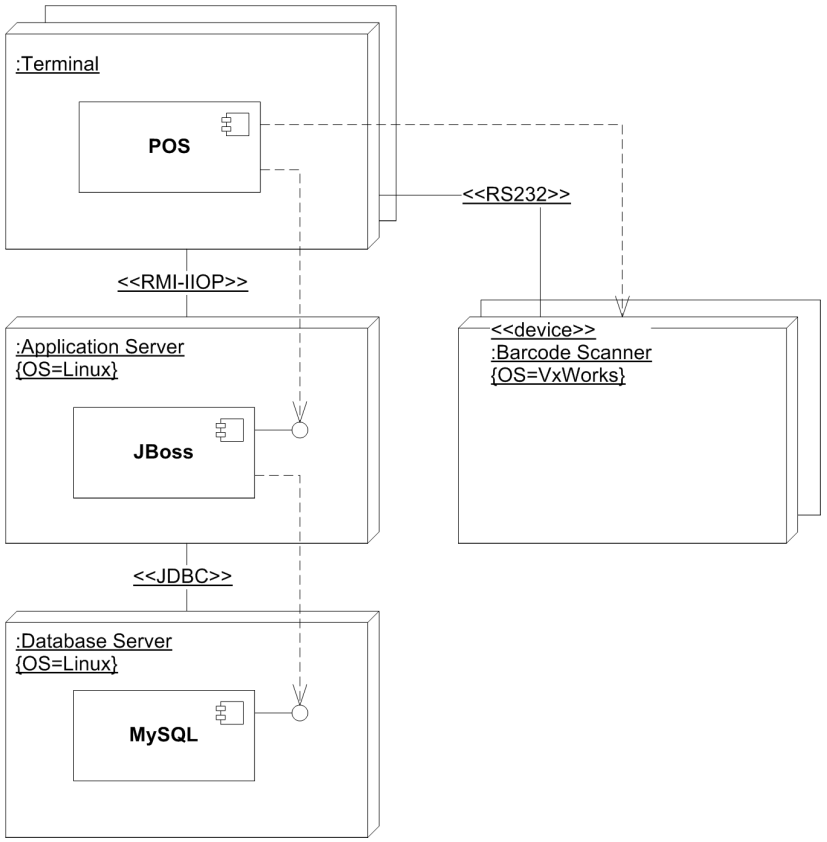
7. Determine Health

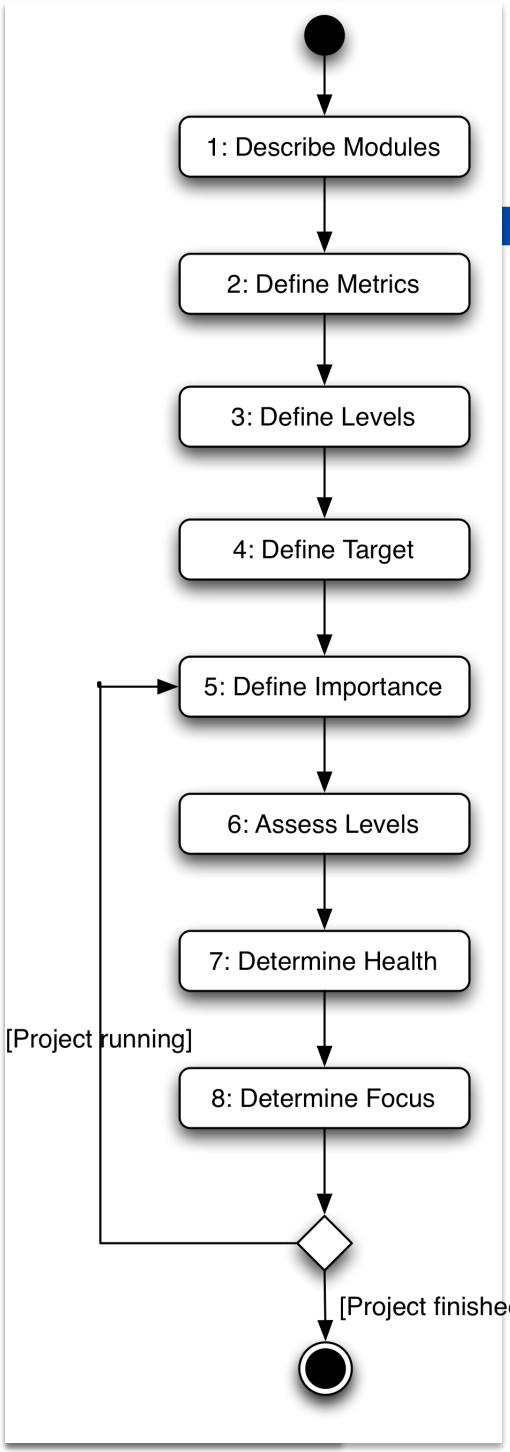
- Combine current and target levels to get a health level

8. Determine Focus

- Using importance, decide what focus is for further work

Example: POS Again

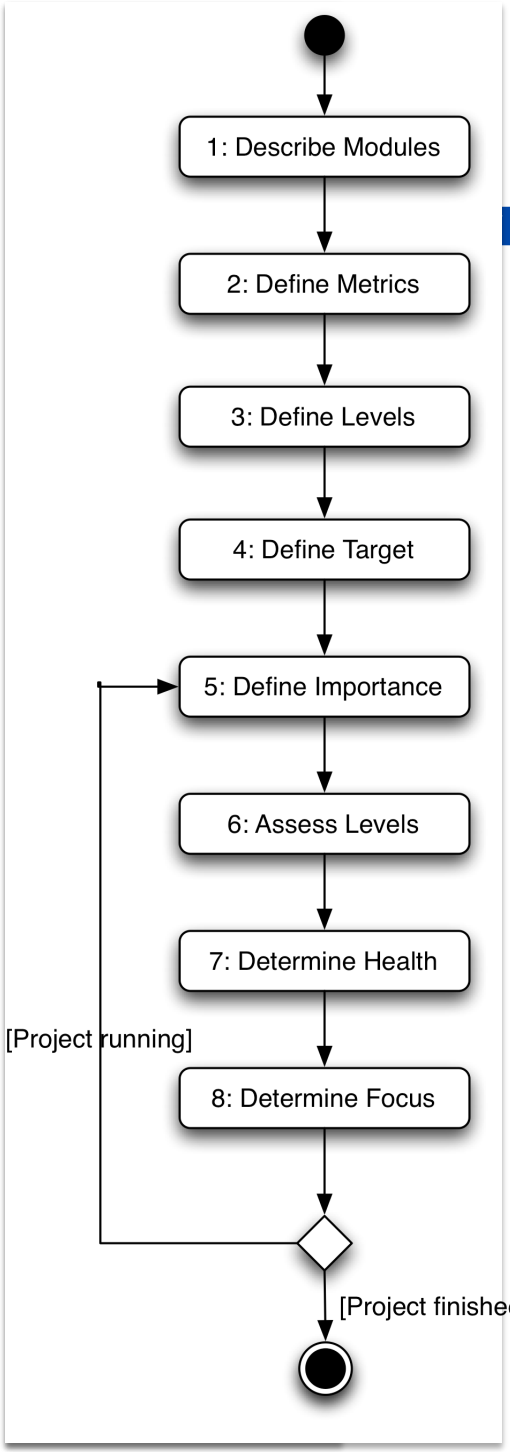




1. Describe Components

ERSITET

	Terminal	Scanner	App Server
Availability			
Performance			
Modifiability			
Testability			
Security			
Usability			

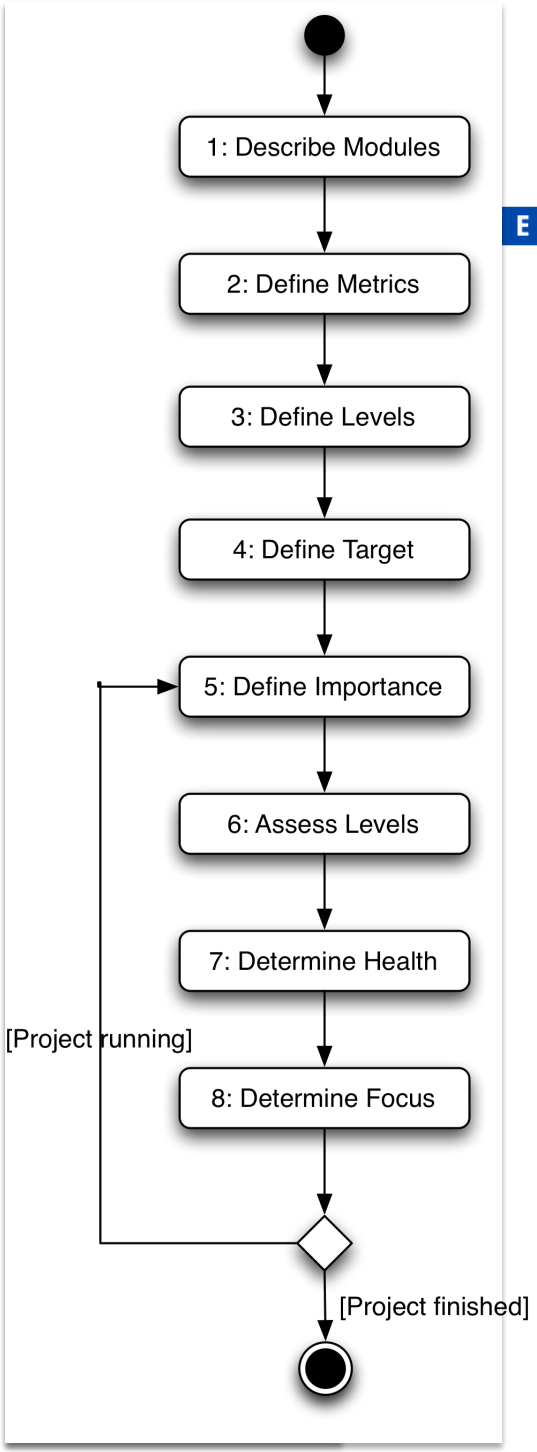


2. Define Metrics

ERSITET

Main types

- Scenario-based
- Measurement-based
- Judgment



3. Define Levels

ERSITET

Problem

- Want to compare across attributes
- Need a common scale

Define ordinal scale

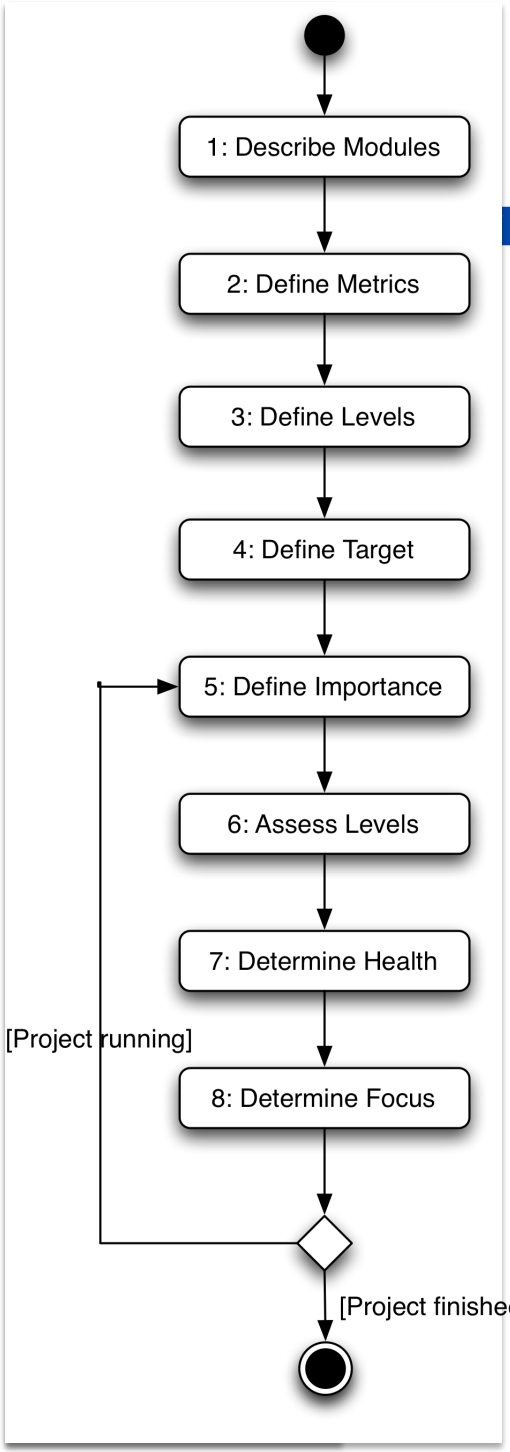
- 1, 2, 3, 4, 5

Generic levels

- Level 1: Unacceptable
  - Important stakeholders find system unacceptable because of quality level of the attribute in question
- Level 3
  - No relevant stakeholder find system unacceptable because of quality level of the attribute in question
- Level 5: Excellent
  - All relevant stakeholder are highly satisfied by the quality level of the attribute in question

Map metrics to levels

- E.g., all scenarios fulfilled completely = 5
- E.g., no scenarios fulfilled = 1

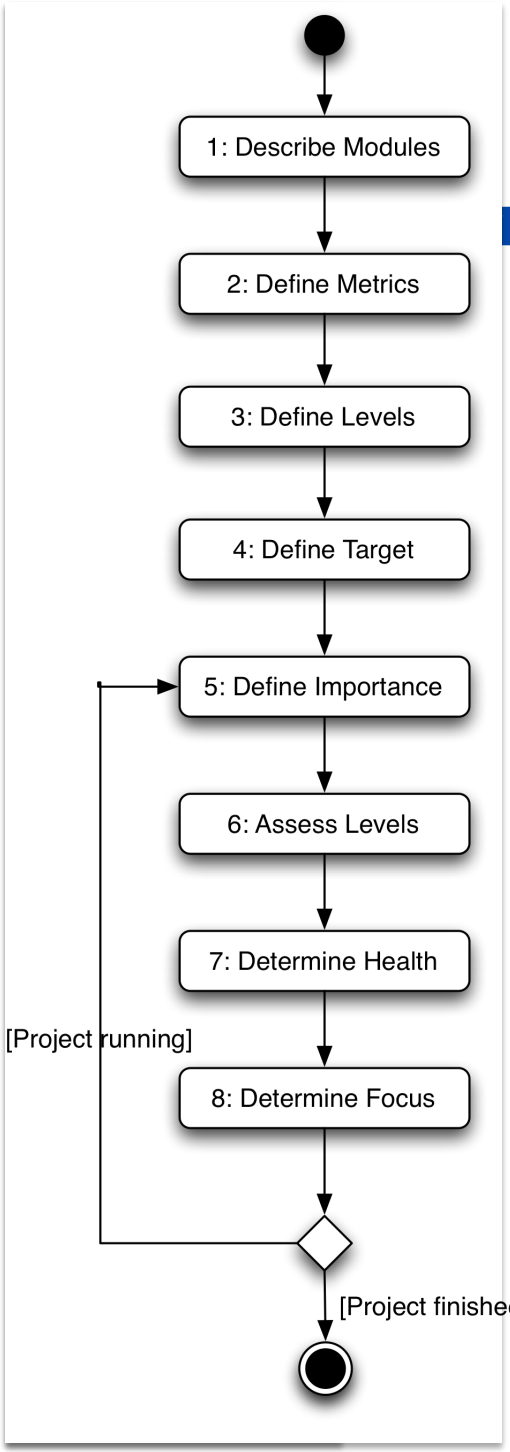


4. Define Target

ERSITET

For each component/attribute determine target level

Target Levels			
	Terminal	**Scanner**	**App Server**
Availability	4	3	4
Performance	5	5	1
Modifiability	4	3	5
Testability	4	3	5
Security	4	3	5
Usability	5	5	3

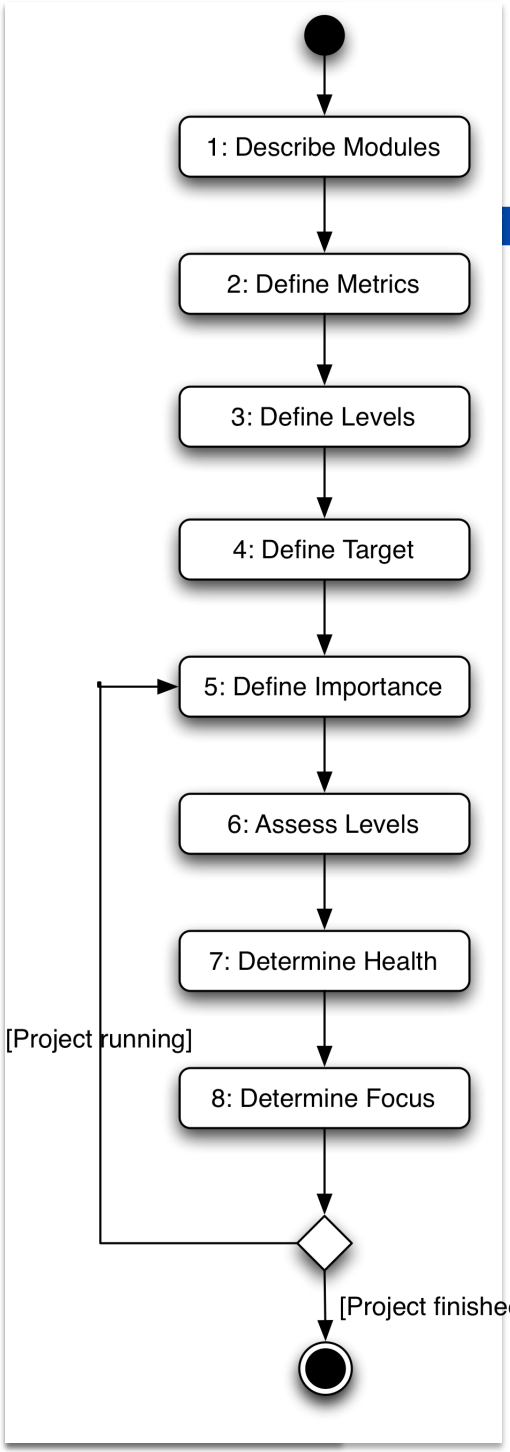


5. Define Importance

ERSITET

For each component/attribute determine importance level

Importance Levels			
	Terminal	**Scanner**	**App Server**
Availability	1	1	1
Performance	5	5	5
Modifiability	2	2	2
Testability	2	2	2
Security	2	2	3
Usability	2	2	2



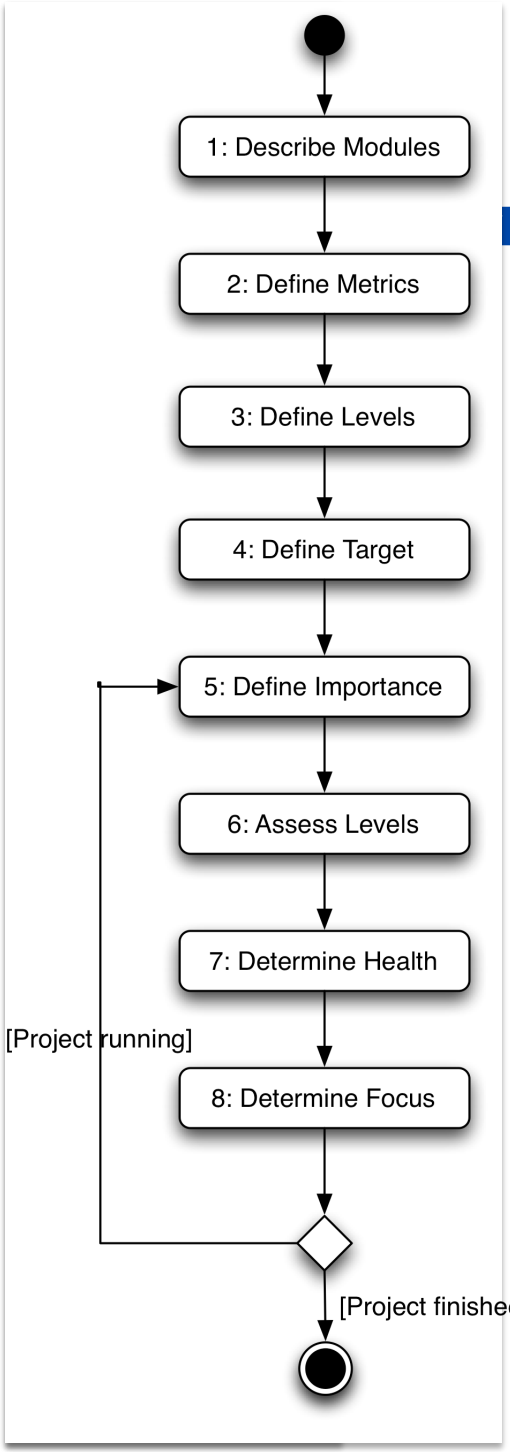
6. Assess Levels

ERSITET

Use metrics for each component/attribute in order to determine current level

- May be delegated to architect/lead developer for each component

Current Levels			
	Terminal	**Scanner**	**App Server**
Availability	4	3	4
Performance	2	3	2
Modifiability	4	4	5
Testability	4	4	4
Security	4	3	2
Usability	1	2	3



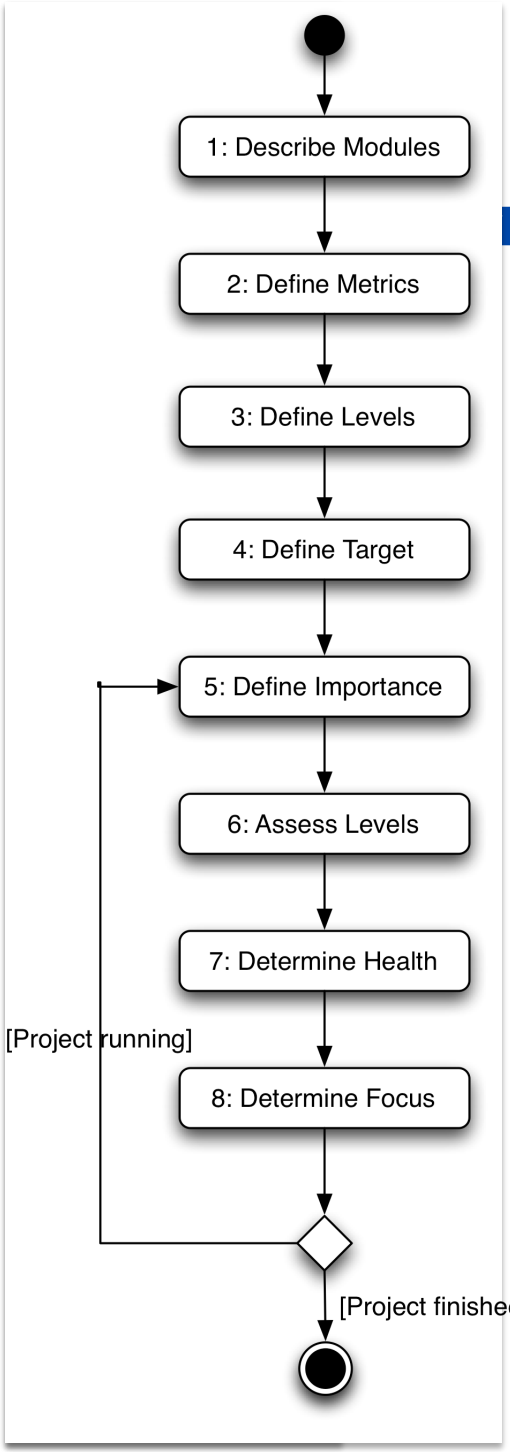
7. Determine Health

ERSITET

Use target and current levels to determine health

– health = 5 – max(0, (target – current))

Health Levels			
	Terminal	**Scanner**	**App Server**
Availability	5	5	5
Performance	3	2	4
Modifiability	5	5	5
Testability	5	5	5
Security	5	5	3
Usability	1	2	3



8. Determine Focus

ERSITET

Use health and importance to determine focus

- E.g., focus = min(5, (6 – health)*importance)

Prioritize focus for qualities

- E.g., App server: performance, security, usability

Health Levels			
	Terminal	**Scanner**	**App Server**
Availability	5	5	5
Performance	3	2	4
Modifiability	5	5	5
Testability	5	5	5
Security	5	5	3
Usability	1	2	3

Results

Continuous assessment of quality level

- E.g., every month or as necessary
- May have low overhead depending on metrics and tool support

Health as well as focus on an overview level

- Useful as a tool for communicating with management
- Useful for prioritization of development effort

Summary

AARHUS UNIVERSITET

Quality attribute-centric approach to monitoring quality

- Operates on quality attributes
- Operates at an architectural level

Requirements

- System can be divided into components that can be measured for each quality attribute
- Metrics are defined for quality attributes

Result

- An assessment of current health and input to focus for further work

Developed by Systematic Software Engineering

- More information
  - http://www.daimi.au.dk/ATiSA/material/2007-11-15%20Software%20Quality%20in%20Practice.pdf