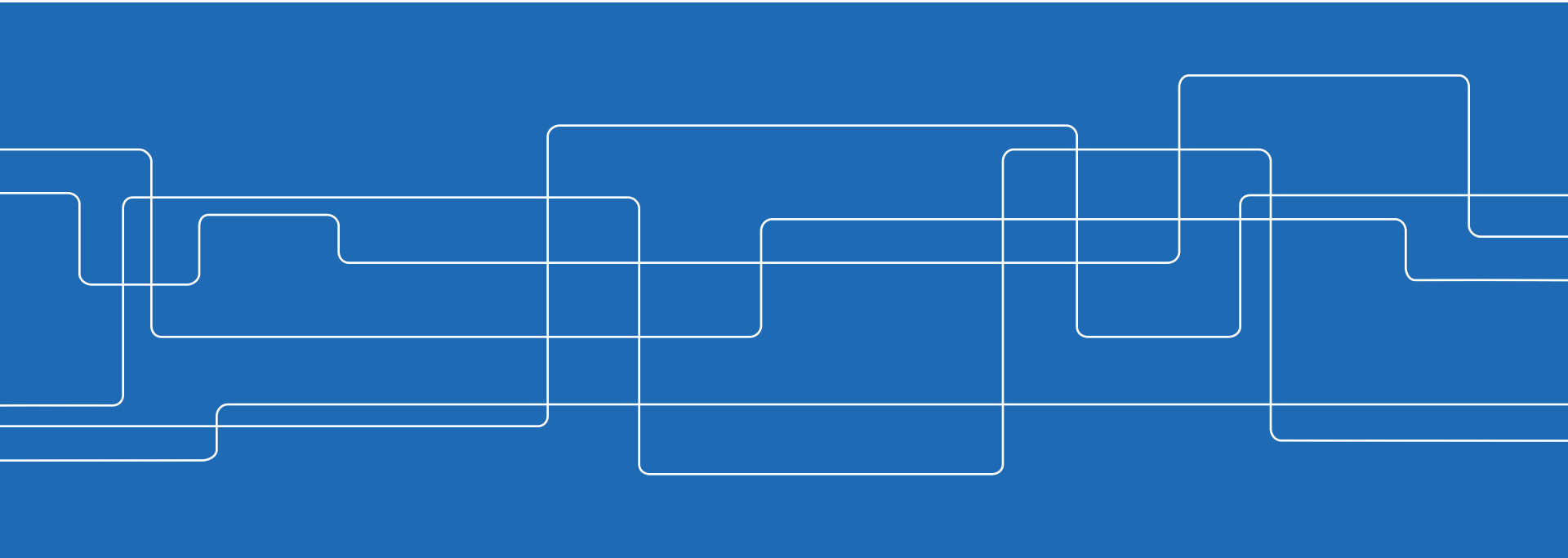
Search-Based Design of Large Software Systems-of-Systems

Robert Lagerström, Pontus Johnson, and Mathias Ekstedt
KTH Royal Institute of Technology



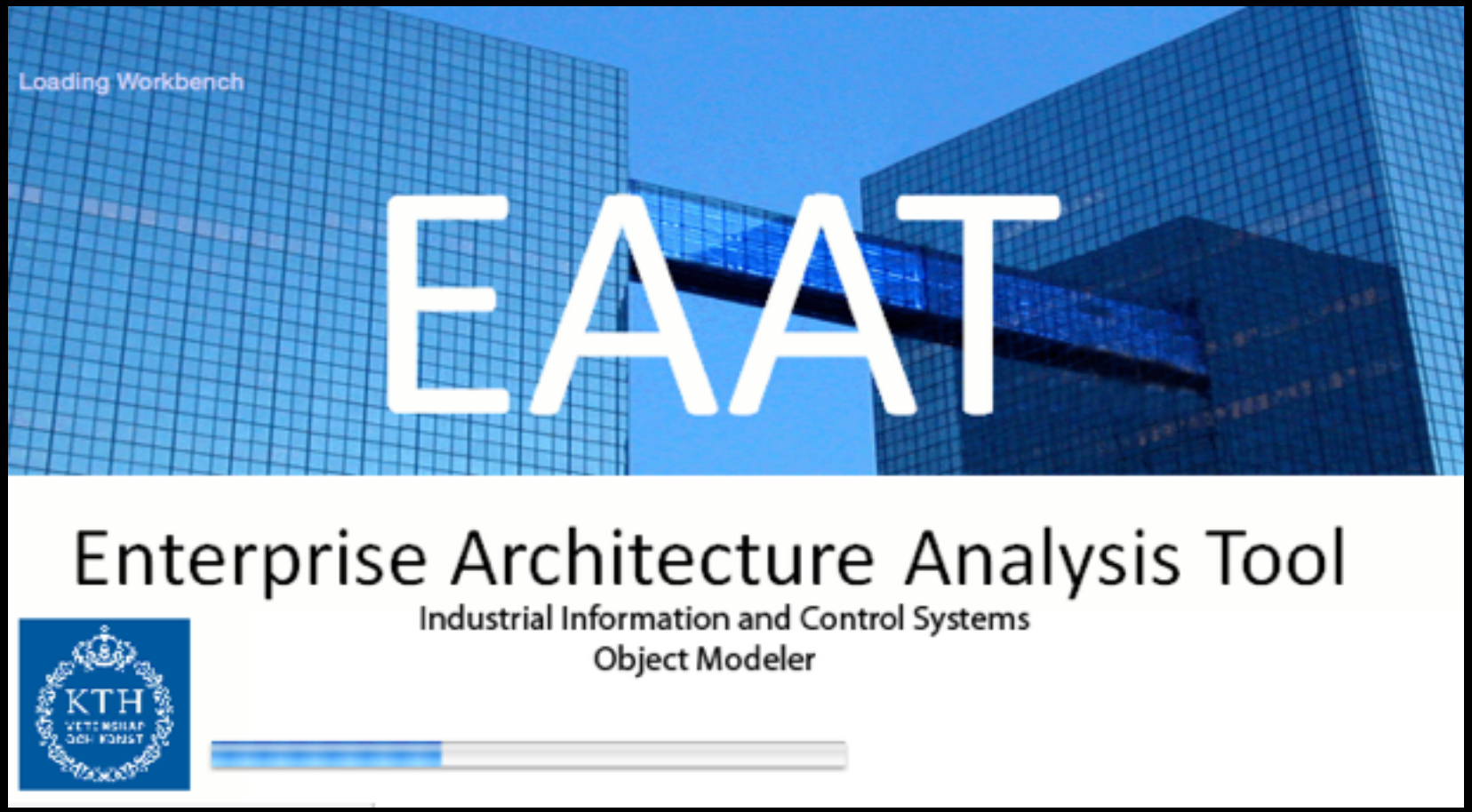
Outline

Today:
Introduction to a systems-of-systems modeling and analysis tool

Search-based software engineering

Tomorrow:
Search-based design of systems-of-system?

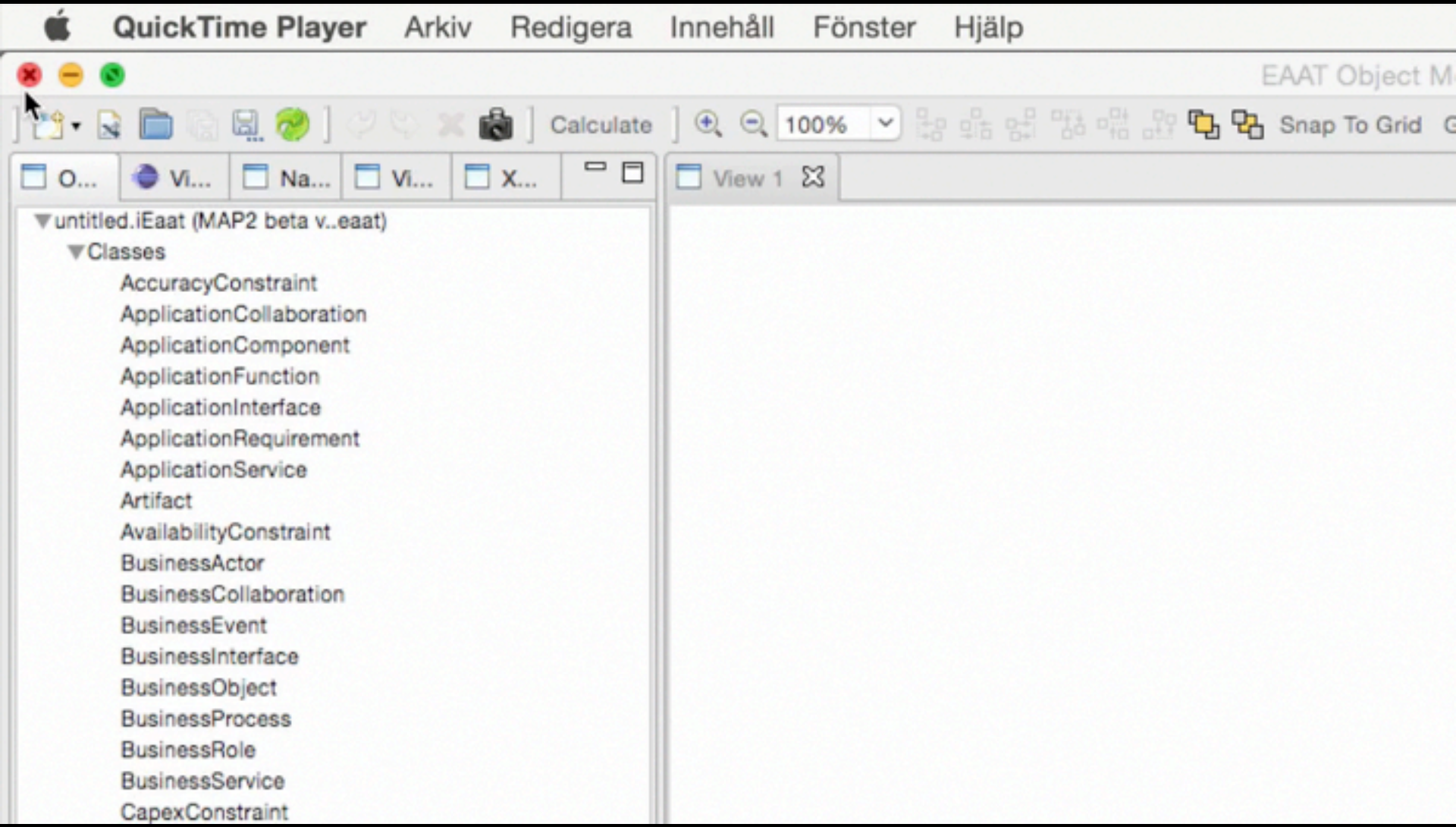
Our EA tool



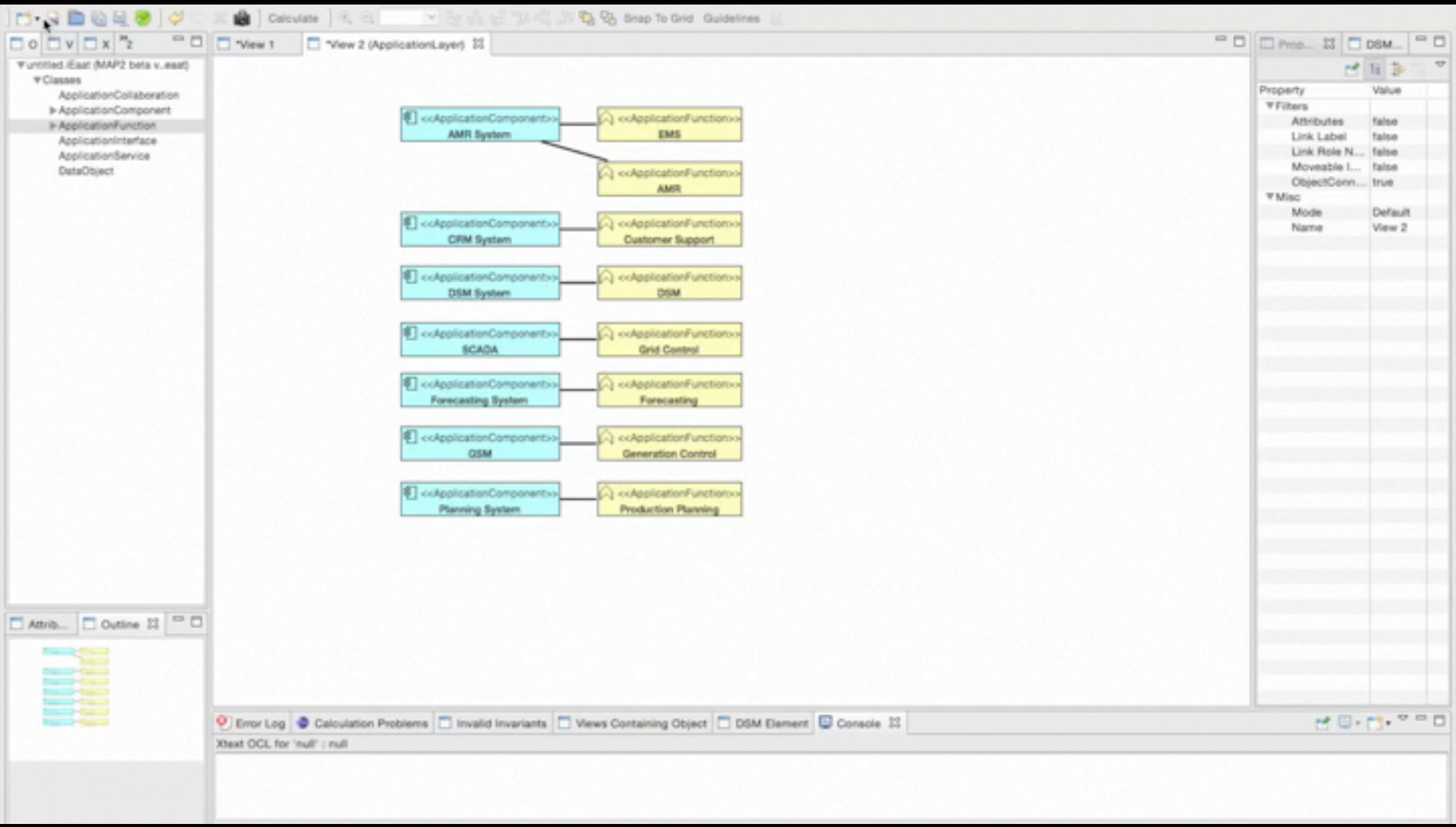
A demo…

(Download at: www.ics.kth.se/eaat)

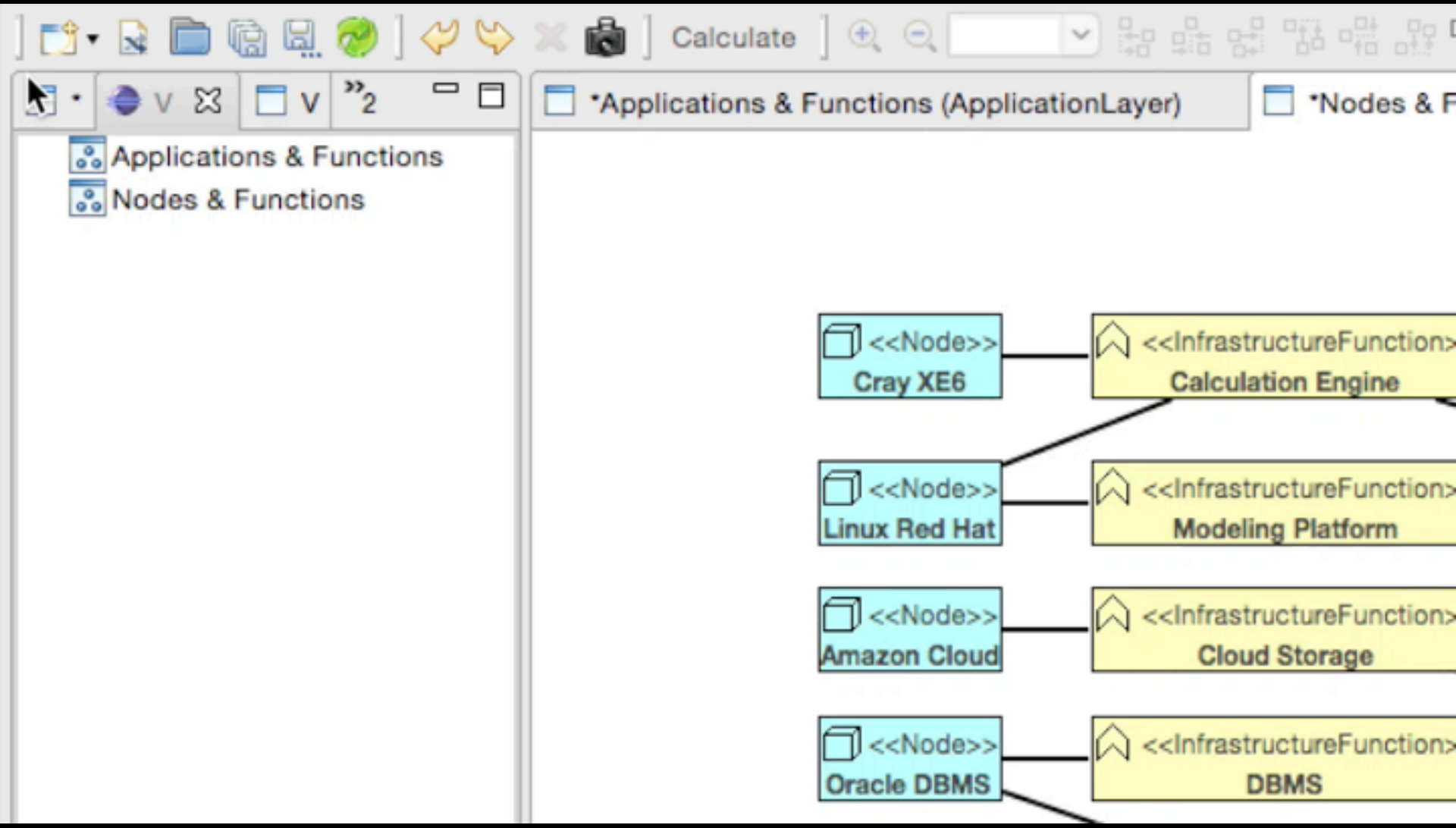
Adding Applications



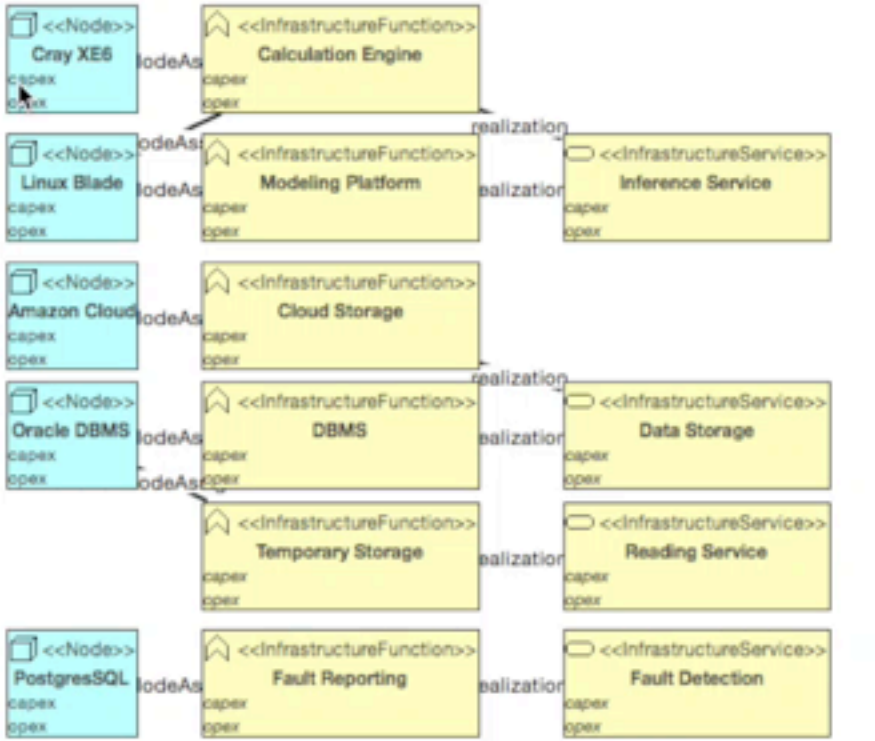
Adding Infrastructure



Cost View

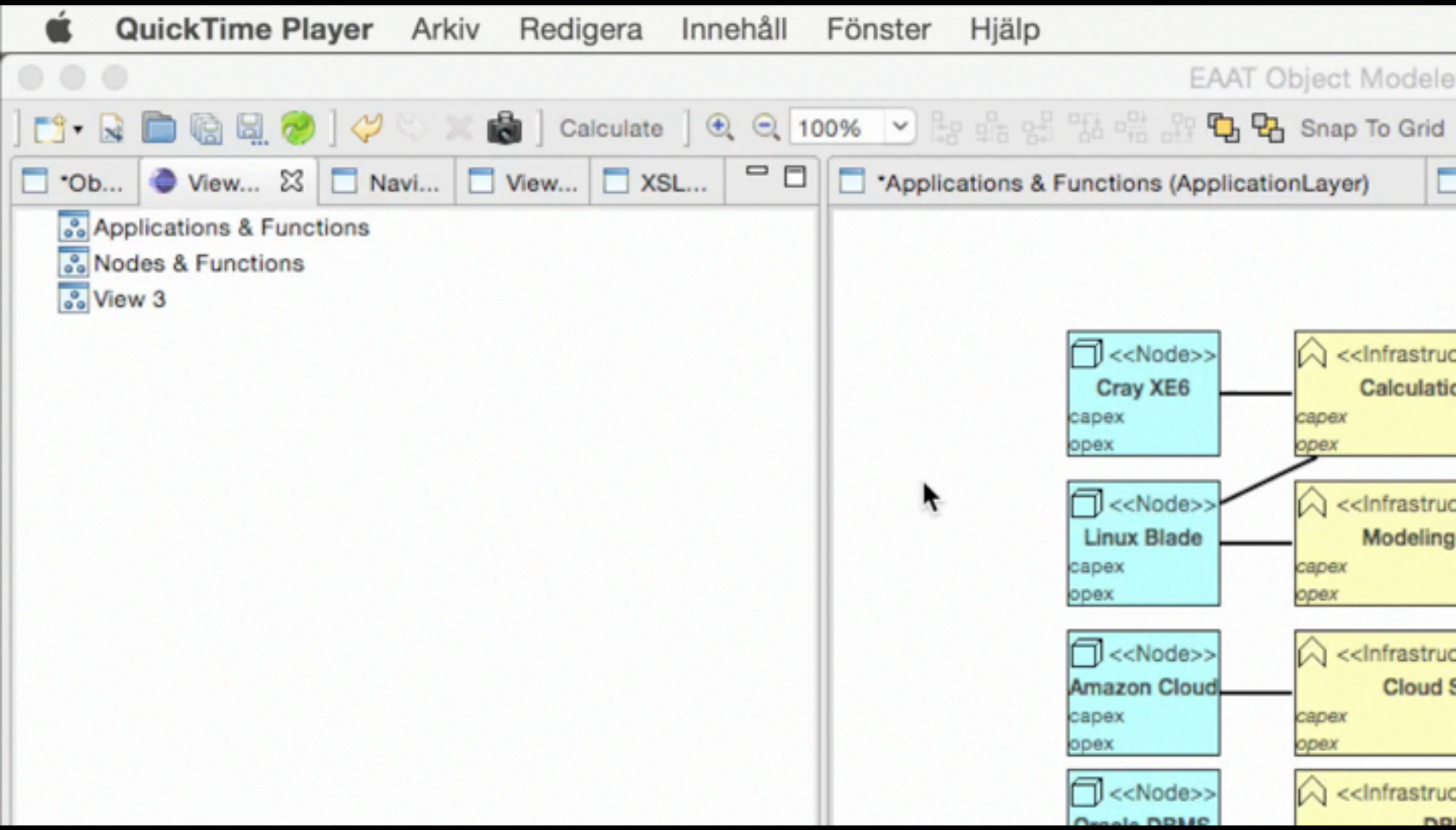

Calculate
Applications & Functions
Nodes & Functions
*Applications & Functions (ApplicationLayer)
*Nodes & F
<<Node>>
Cray XE6
<<InfrastructureFunction>
Calculation Engine
<<Node>>
Linux Red Hat
<<InfrastructureFunction>
Modeling Platform
<<Node>>
Amazon Cloud
<<InfrastructureFunction>
Cloud Storage
<<Node>>
Oracle DBMS
<<InfrastructureFunction>
DBMS

Adding Costs

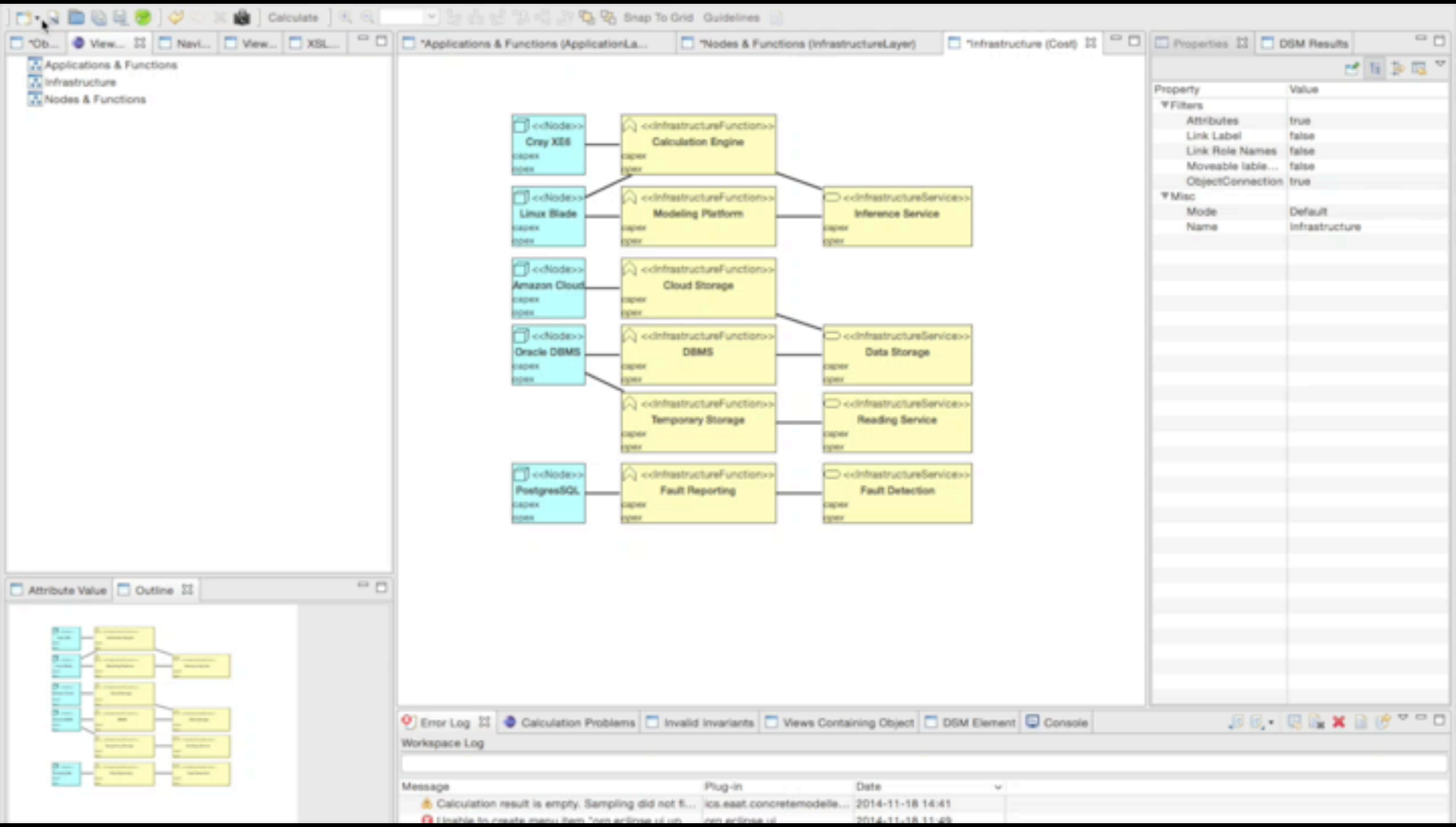


Property	Value
Filters	
Attributes	true
Link Label	true
Link Role Names	false
Moveable lable…	false
ObjectConnection	true
Misc	

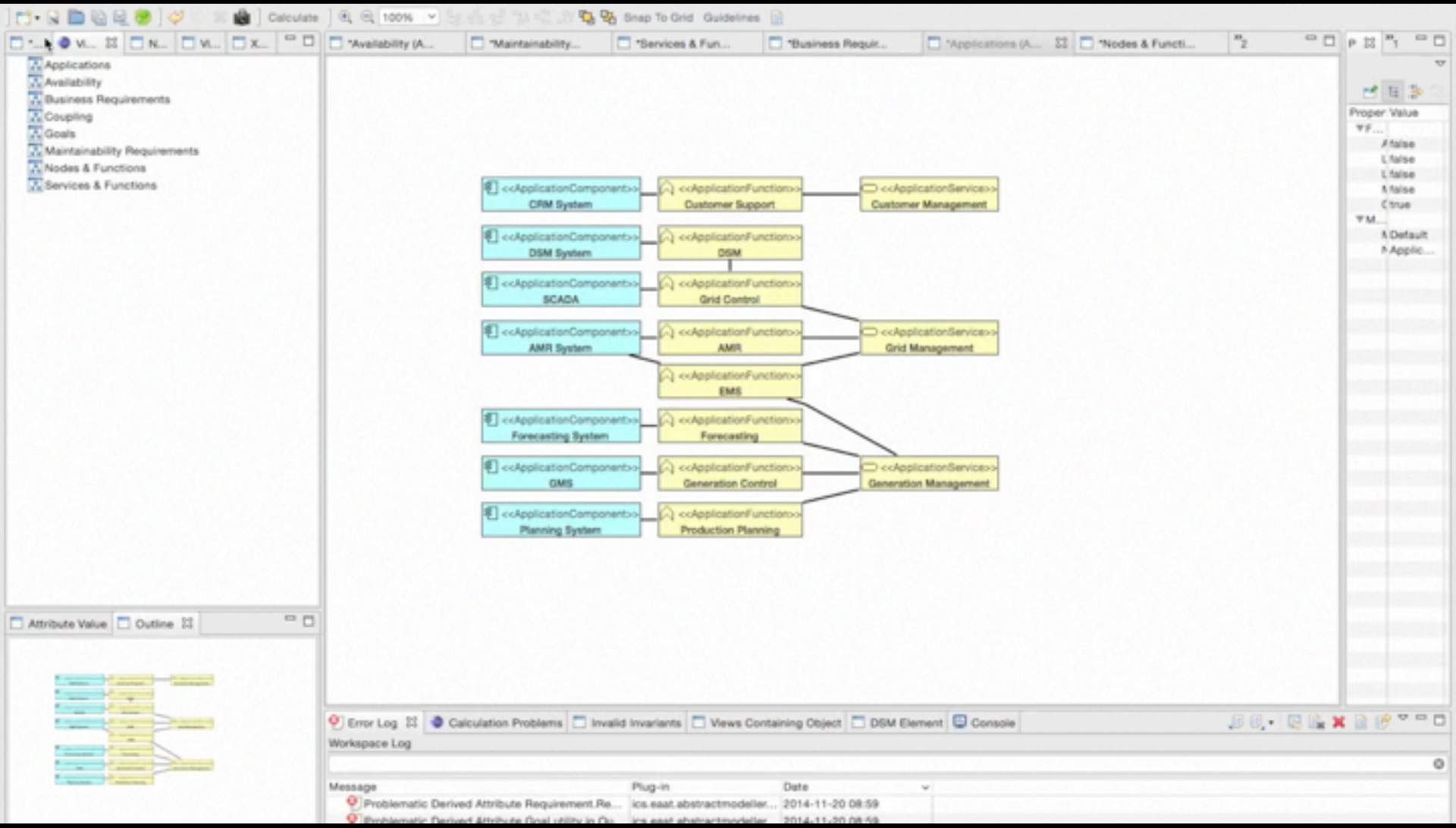
Calculating Infrastructure Costs



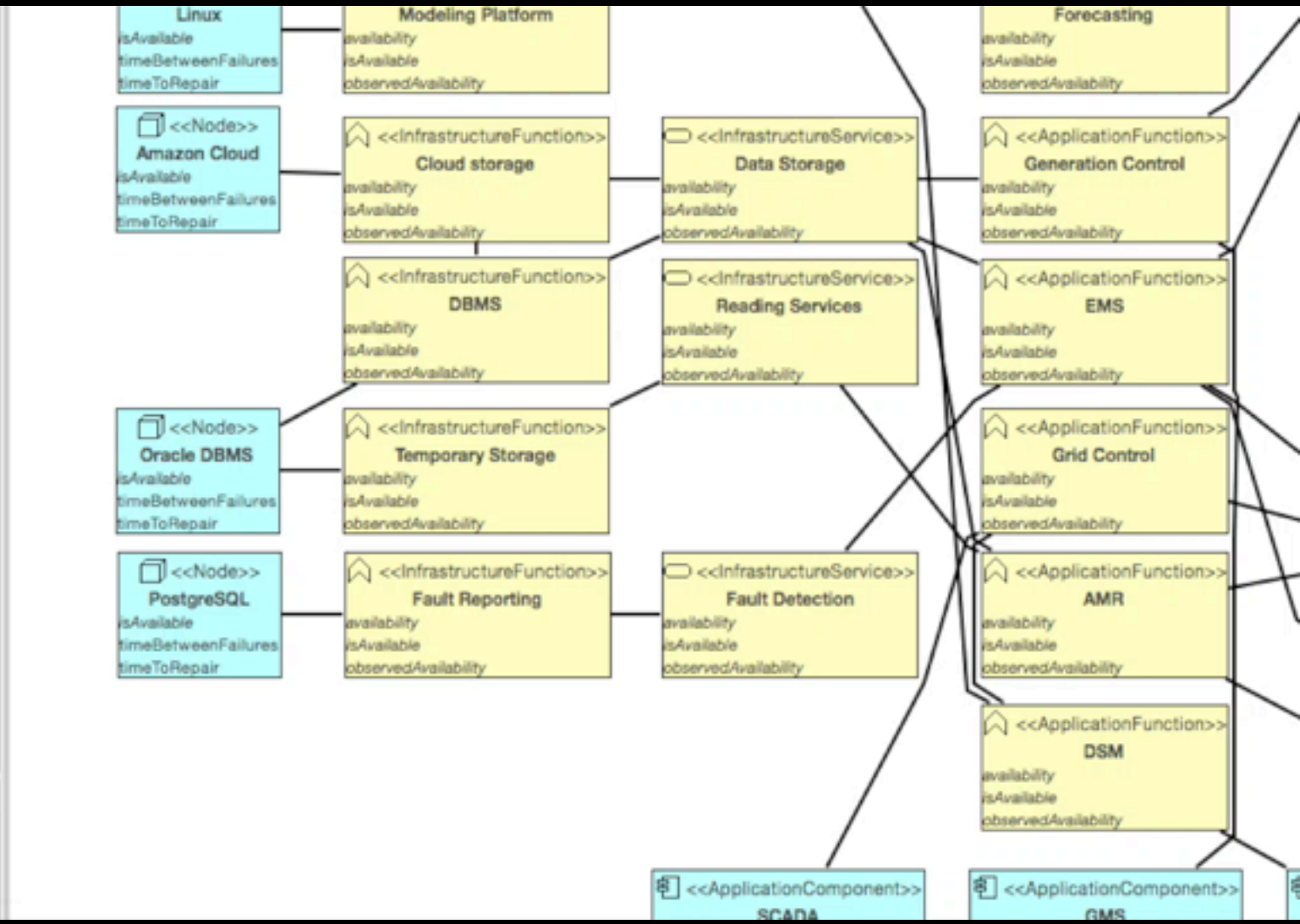
Calculating Application Costs



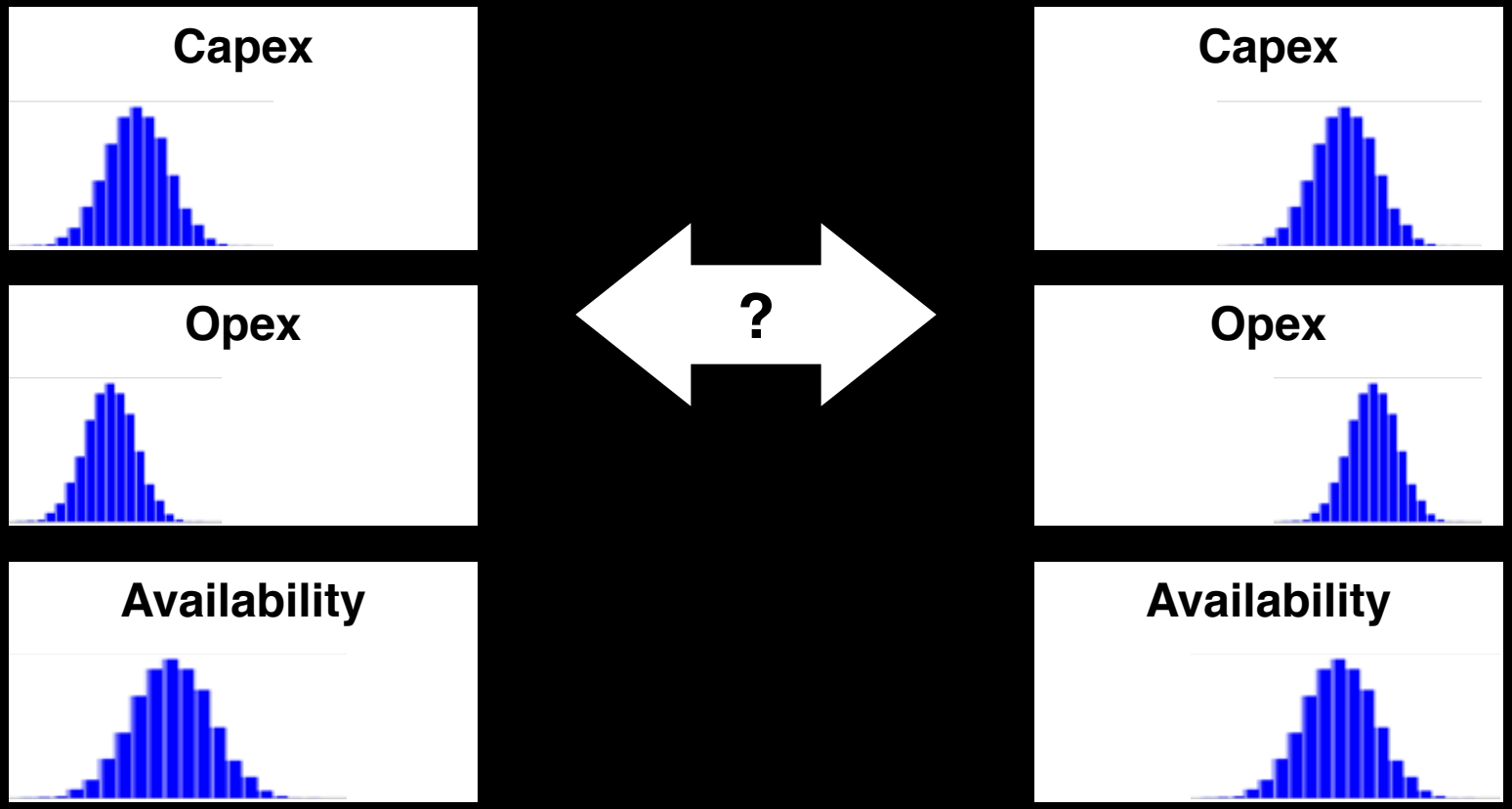
Calculating Availability



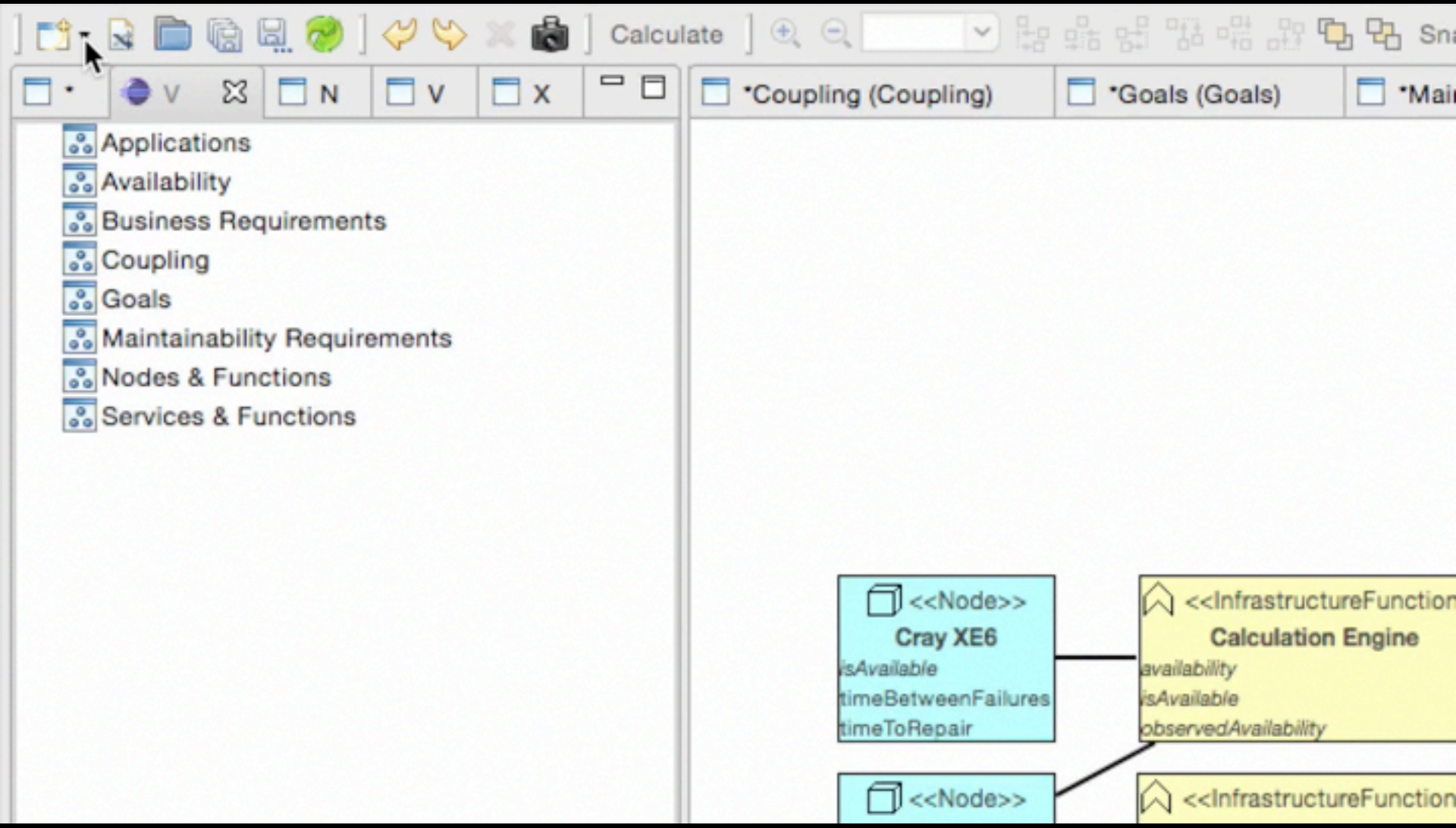
Introducing Redundancy

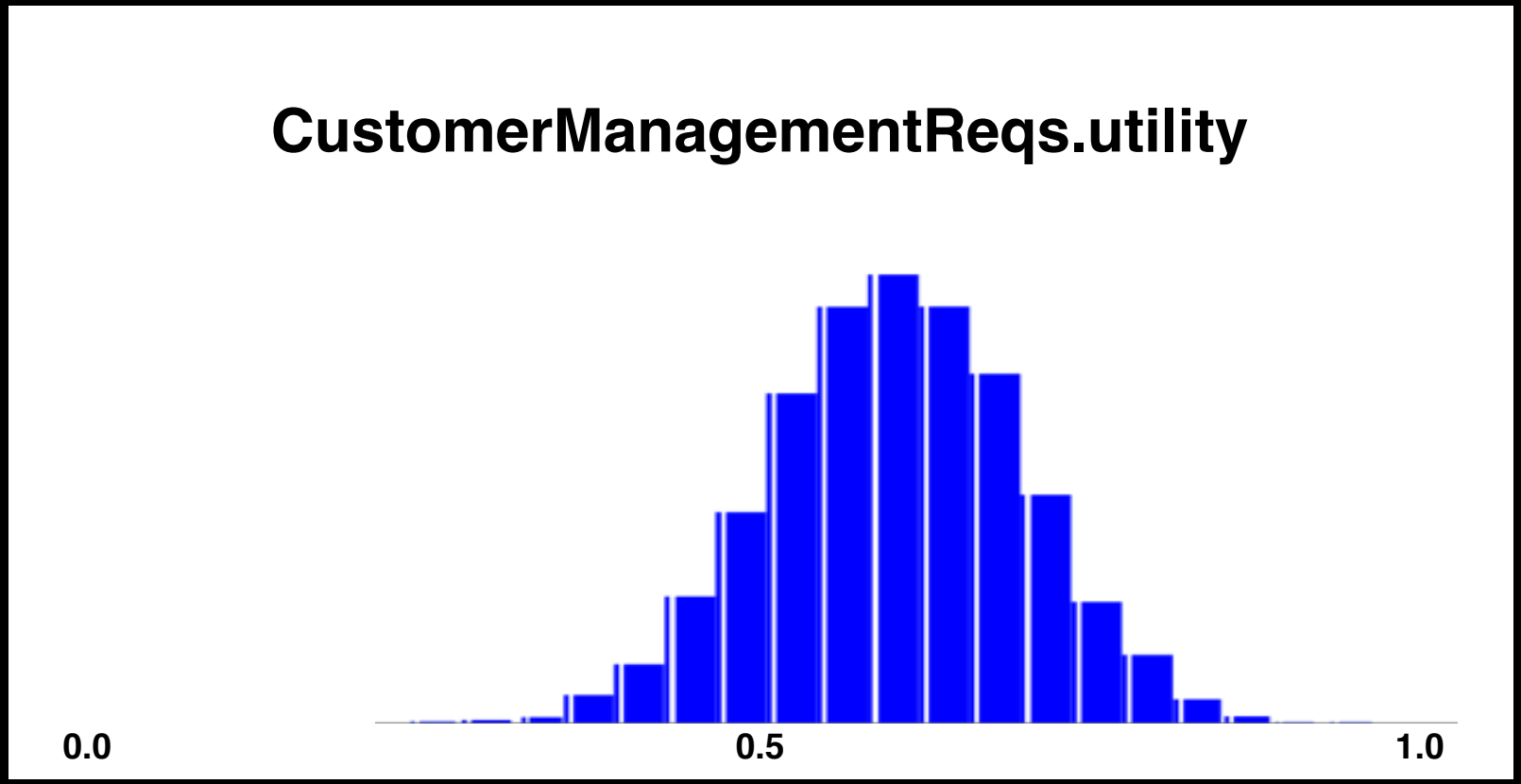


Design Trade-offs

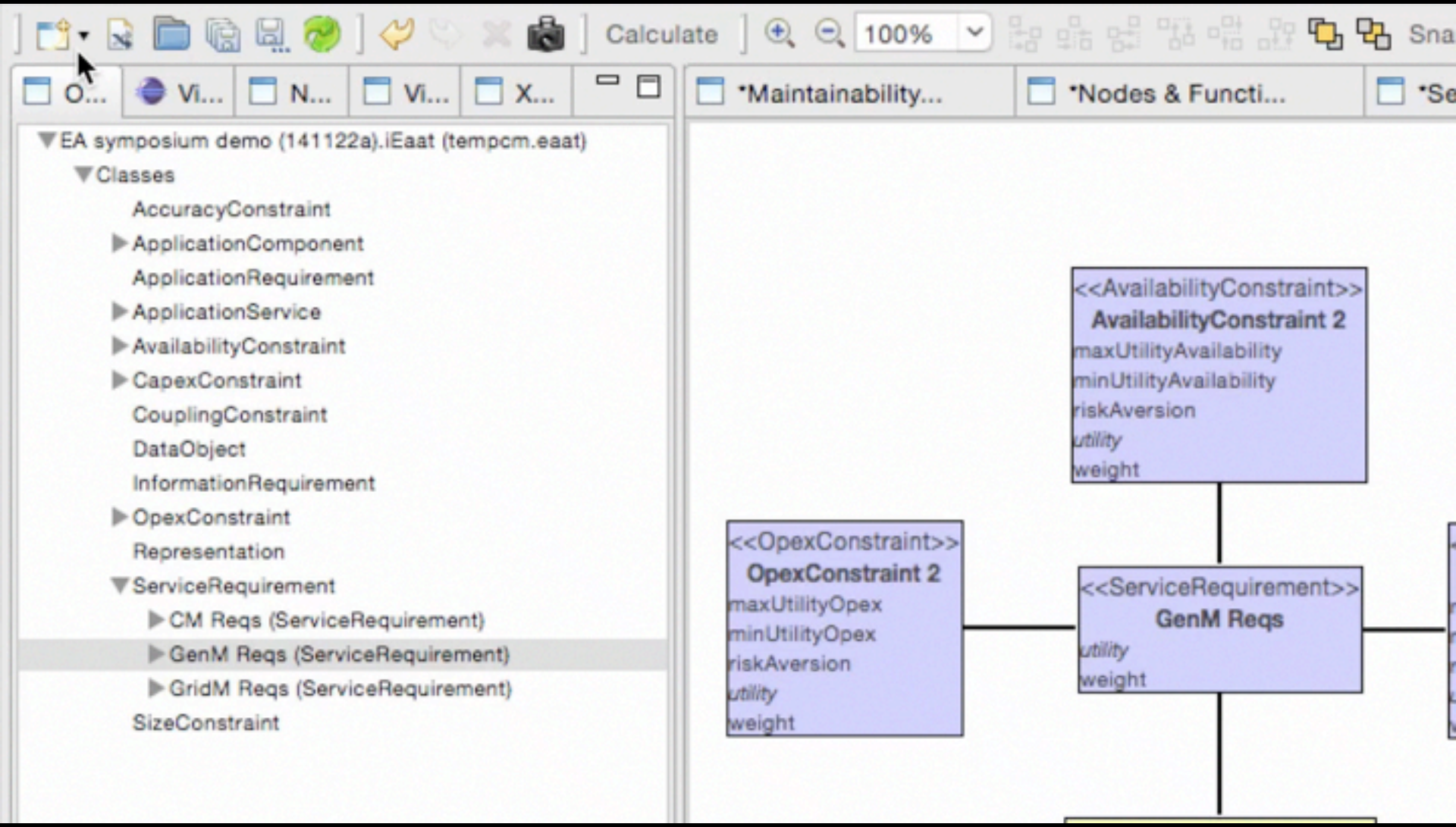


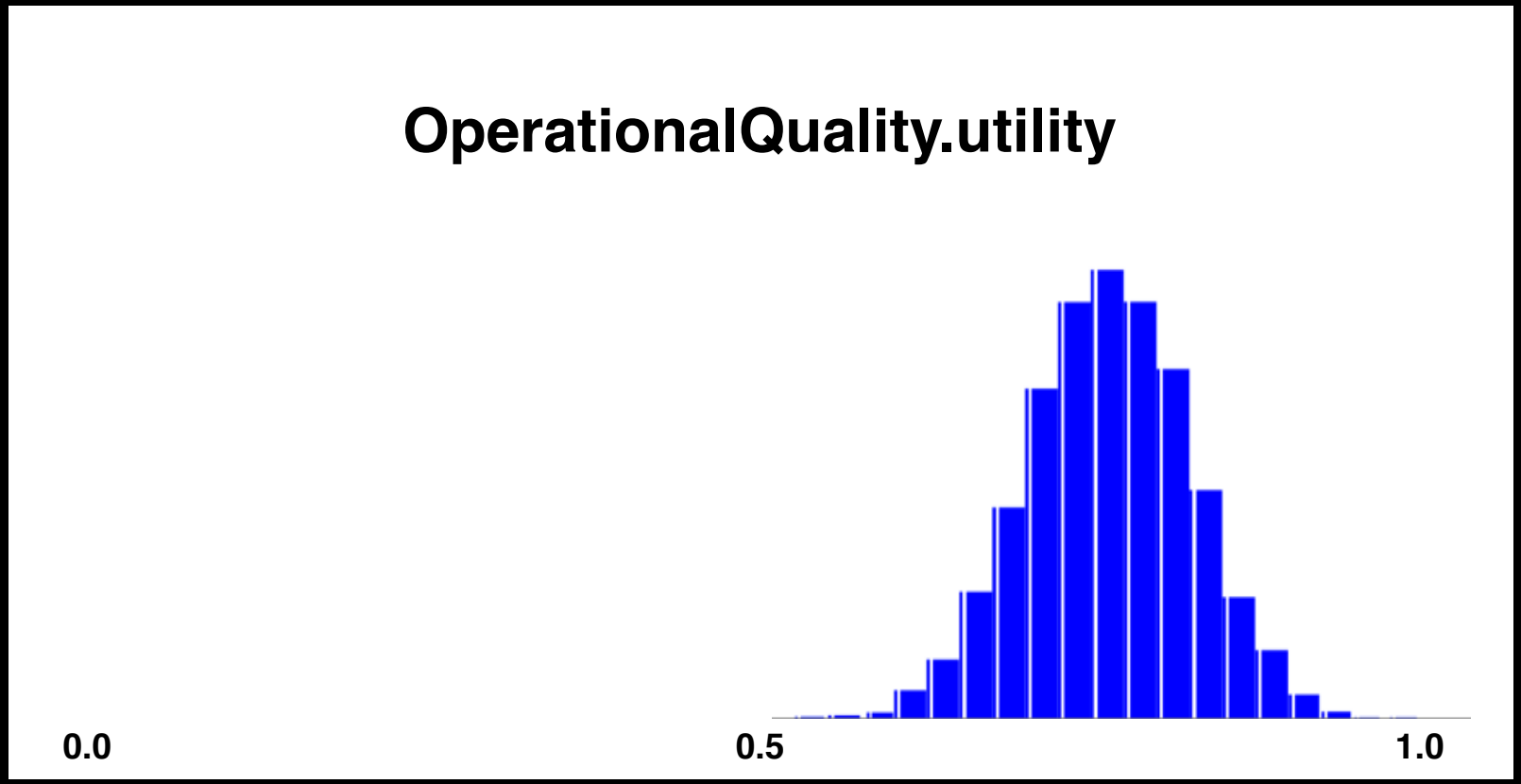
Specifying Requirements



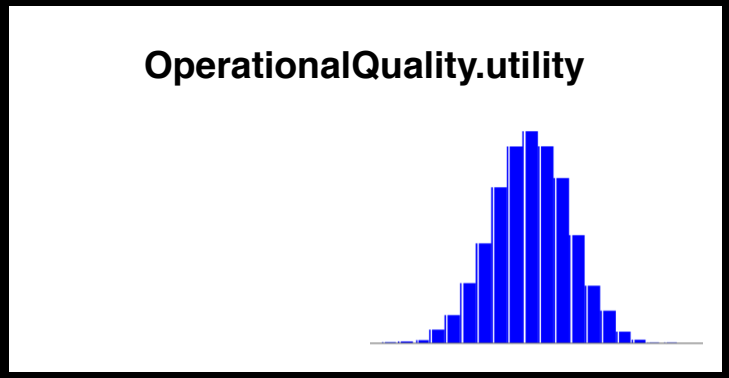

CustomerManagementReqs.utility
0.0
0.5
1.0

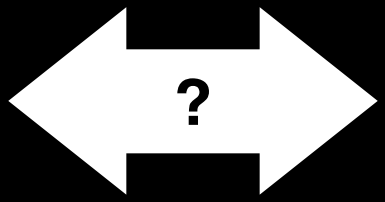
Prioritizing Requirements

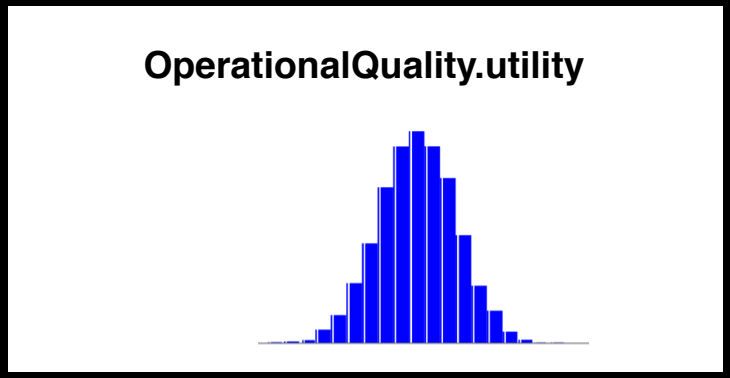


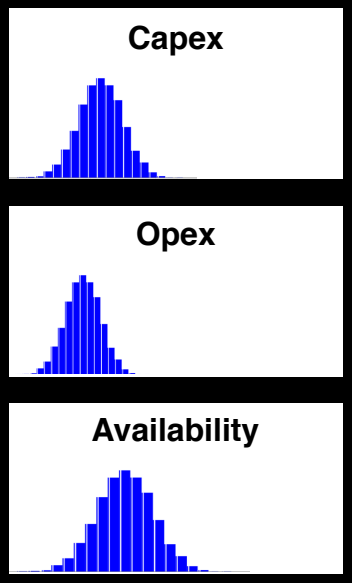

OperationalQuality.utility
0.0
0.5
1.0

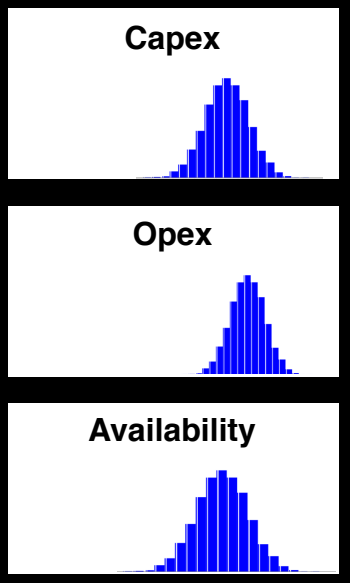
Design Trade-off Aggregation











Current quality attribute coverage

Requirements (utility)
Application service Availability
Application service cost
Data accuracy
Application component coupling
Application size

System interoperability

Information security

Search-Based Software Engineering

Optimization techniques:

Hill climbing
Genetic algorithms
Simulated Annealing
Particle swarm optimization
Ant colony optimization

Search-based software engineering

Mark Harman[1,*], Bryan F. Jones[2,1]

Abstract

This paper claims that a new field of software engineering research and practice is emerging: search-based software engineering. The paper argues that software engineering is ideal for the application of metaheuristic search techniques, such as genetic algorithms, simulated annealing and tabu search. Such search-based techniques could provide solutions to the difficult problems of balancing competing (and some times inconsistent) constraints and may suggest ways of finding acceptable solutions in situations where perfect solutions are either theoretically impossible or practically infeasible.

In order to develop the field of search-based software engineering, a reformulation of classic software engineering problems as search problems is required. The paper briefly sets out key ingredients for successful reformulation and evaluation criteria for search-based software engineering. © 2001 Elsevier Science B.V. All rights reserved.

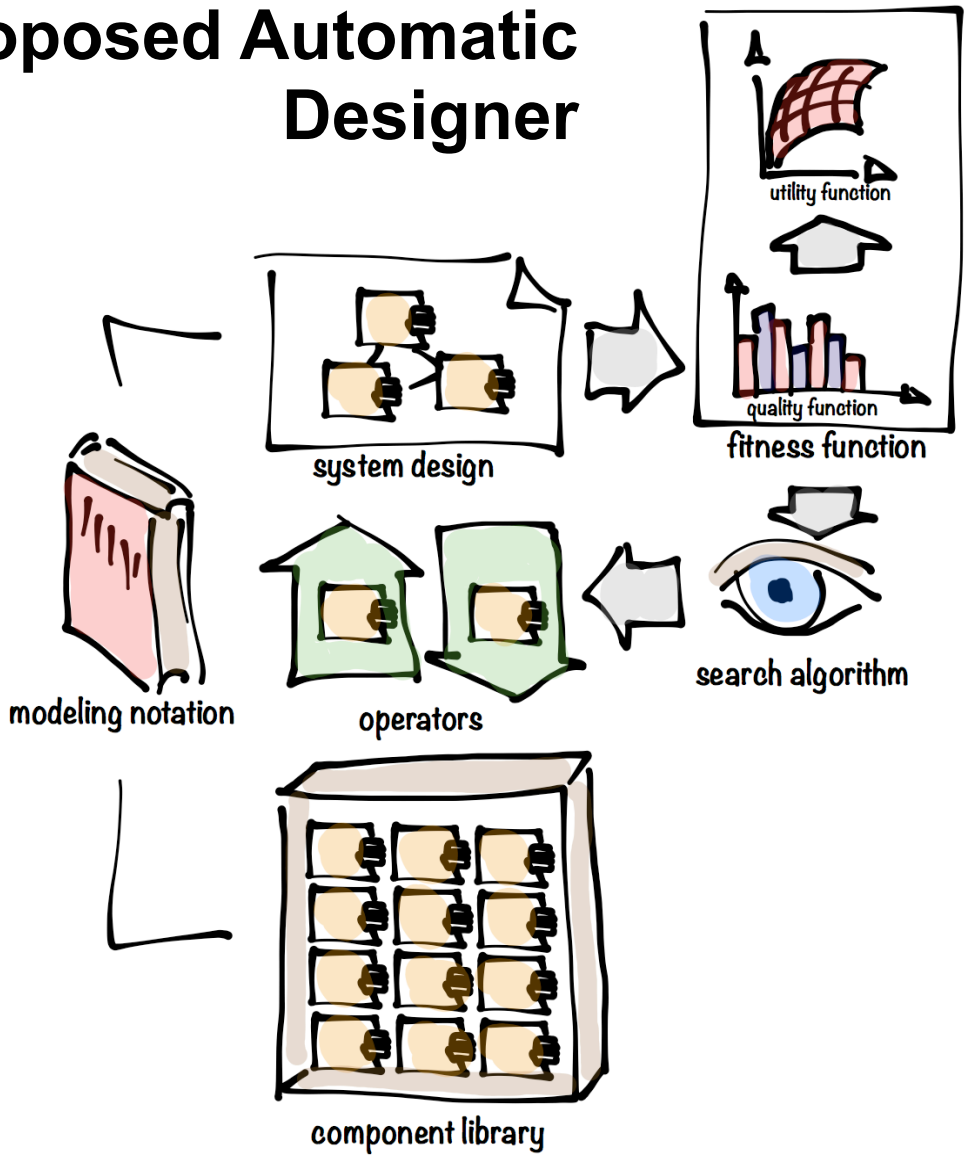
1. Introduction

Mark Harman, Bryan F. Jones,
Search-based software engineering,
Information and Software Technology, 2001

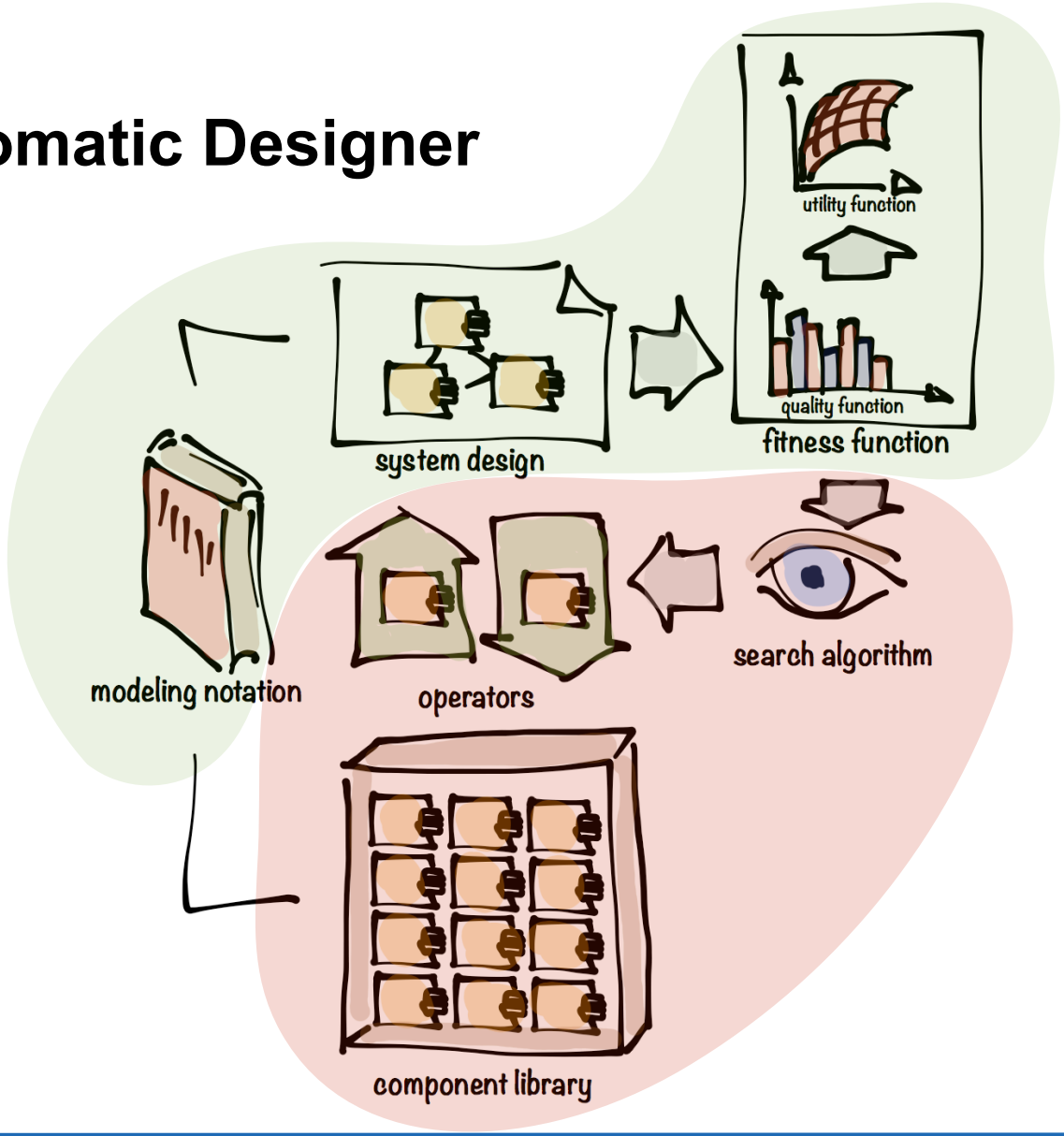
Definition of a (search) problem

- The **initial state** that the agent starts in.
- A description of the possible **actions** available to the agent.
- A description of what each action does; **the transition model**.
- The **fitness function/goal test** is the characterization of what is considered to be a good solution.
- A **path cost** function that assigns a numeric cost to each path.

Proposed Automatic Designer



Automatic Designer



Joint Utility Function