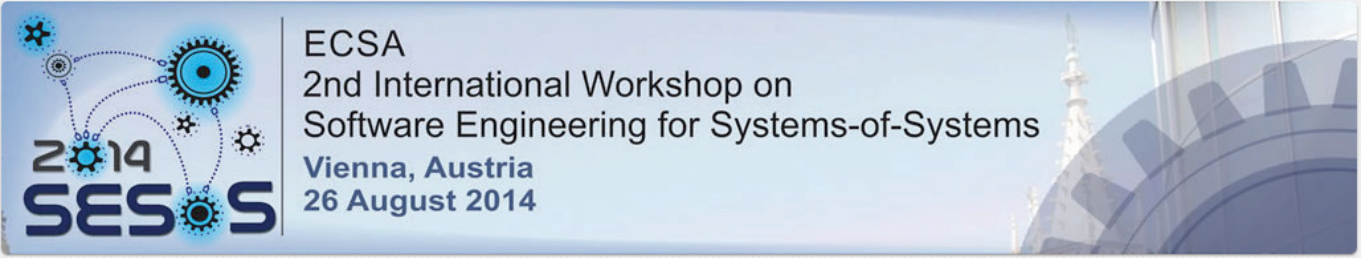


Probabilistic Macro-Architectural Decision Framework

Plamen Petrov, University of Illinois at Chicago
Robert L. Nord, Carnegie Mellon University
Ugo Buy, University of Illinois at Chicago

Presented at the 2nd International Workshop on
Software Engineering for Systems-of-Systems (SoS) at
8th European Conference on Software Architecture (ECSA 2014)

August 26, 2014
Vienna, Austria

Large number of software development projects fail or at least significantly underperform even though their architecture was designed using generally accepted methods and state-of-the-art best practices

The Problem

- ✓ Identified as recurring problem
- ✓ Significant business impact
  - ➢ No obvious issues identified with respect to requirements or architecture
  - ➢ Software developed using best practices

- ❑ Existing architectural methods couldn't solve the problem
  - ▪ Inappropriate for environment in which developed and operated
  - ▪ Incompatible organizationally, financially, culturally, strategically, operationally

Examples - Motivation

Examples:

1. Service Oriented Architecture (SOA)(1) chosen for clinical performance management system

   - **Issue**:
     - Took much longer to design and develop
     - Too costly
     - Slow to maintain and evolve
     - Quality issues became unsustainable
     - Design eventually abandoned
   - **Assessment**:
     - Software architecture and design followed standard practices
     - Root cause - misalignment of software architecture and contextual environment
       - ✓ Funding model would not support a SOA system
       - ✓ Development team lacking right skills and experience
       - ✓ Business stakeholders did not support investment for SOA

(1) http://msdn.microsoft.com/en-us/library/ee658117.aspx
Defined as: "… expose and consume functionality as a service using contracts and messages"

Examples - Motivation

Examples (cont.):

2. Open source libraries chosen for user interface and database access for mission critical transactional business system

- **Issue**:
  - Too many defects using libraries
  - Slow long to fix defects
  - Too costly to maintain
  - Quality issues became unsustainable
  - System was removed and redeveloped
- **Assessment**:
  - Software architecture and design followed standard practices
  - Root cause - misalignment with organizational and funding environment
    - ✓ Funding model did not justify open source risks
    - ✓ Development team and maintenance team experience & skills misaligned
    - ✓ Cultural bias for low risk tolerance to quality issues

Examples - Motivation

Examples (cont.):

3. Chosen Domain Specific Language (DSL) instead of configuration files for system customization of a survey collection and management system

- **Issue**:
  - User dissatisfaction – not easy to use
  - Slow and hard to make modifications
  - Extra staff to operate the system
  - Too costly to operate
  - Unsustainable quality issues with enhancements
  - System re-architected and redeveloped
- **Assessment**:
  - Software architecture and design followed standard practices
  - Root cause - misalignment with organizational and funding environment
    - ✓ Funding model did not support extra complexity
    - ✓ Development team and maintenance team experience & skills misaligned
    - ✓ Cultural and skill misalignment with business user preferences

Numerous Software Architecture Analysis and Design Methods exist:

- Rational 4+1, Siemens 4 Views, SEI Views and Beyond, SEI Attribute Driven Design (ADD), Philips BAPO, etc.

Architectural Drivers.[1][2]

- **Functional requirements**: what the system must do as functions
- **Quality attributes**: qualification of how the system should perform the functions (e.g. how fast, how reliable, how secure, etc.)
- **Constraints**: design decision already made that the architect cannot change (e.g. system must use Java, must use Oracle DB, etc.)

State-of-the-art: ***quality attributes*** are by far the most important drivers.

➢ **Business goals** – recently introduced - *Pedigreed Attribute eLicitation Method (PALM)* [1][3]

"some higher purpose that can be described in terms of added value"

✓ Insufficient – do not capture all architectural drivers

✓ We introduce the concept of **contextual factors** - *constraints* and *considerations*

(1) L. Bass, Software architecture in practice, 3rd ed. Upper Saddle River, NJ: Addison-Wesley, 2013.

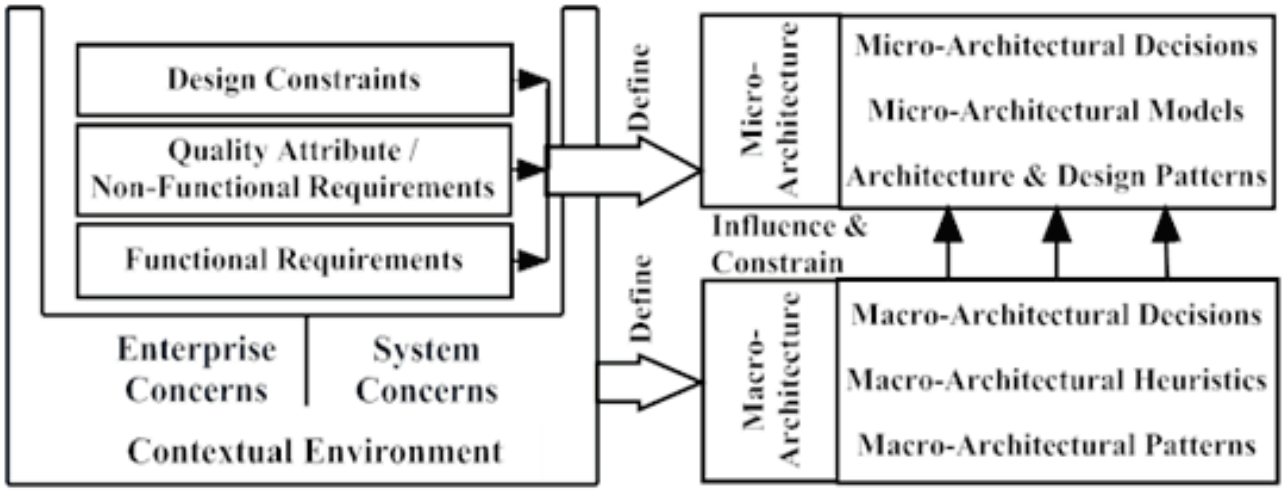
(2) N. Rozanski, E. Woods, Software Systems Architecture, 2nd Edition, Pearson Education, 2012

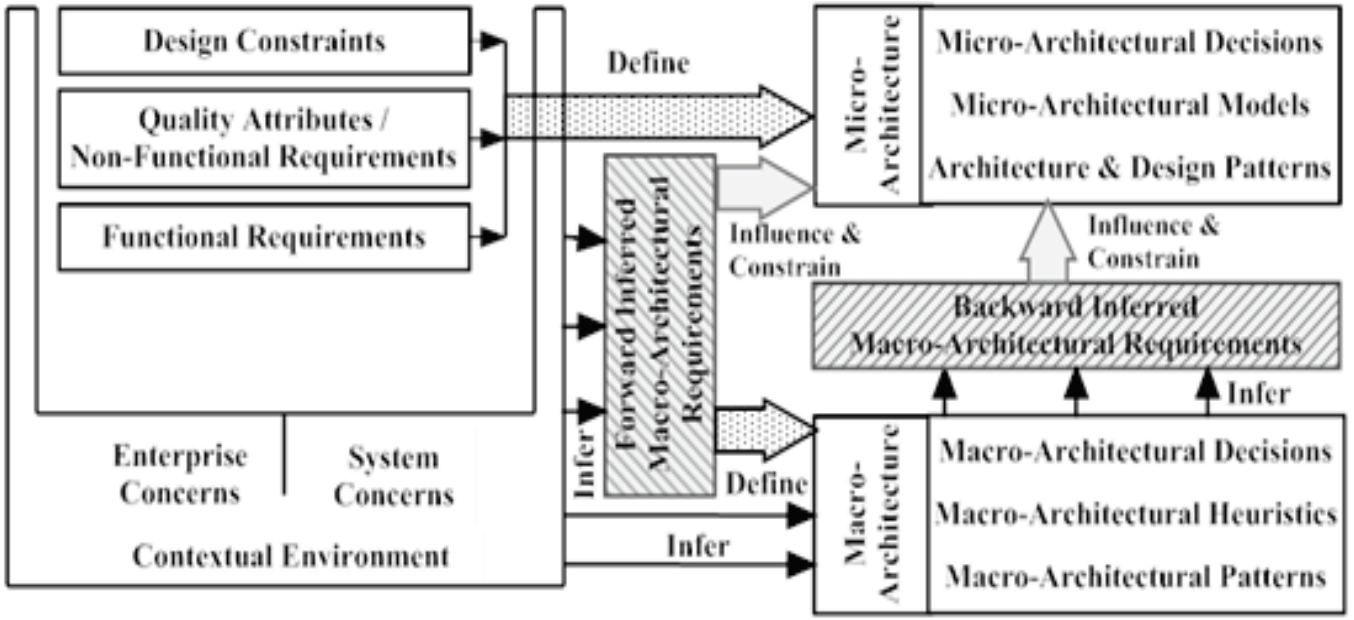
(3) P. Clements et. al, Relating Business Goals to Architecturally Significant Requirements for Software Systems, CMU/SEI-2010-TN-018, 2010

Approach

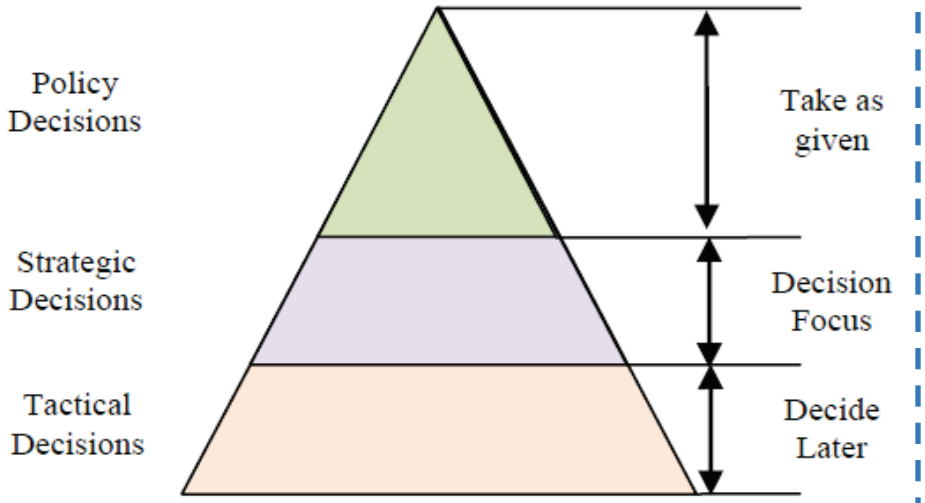
- Importance of contextual factors and their formal use as architectural drivers (funding models, organizational structure, team expertise, cultural biases, business and operational focus)
- Bayesian Networks-based modeling formalism and decision framework for contextual factors
- Macro-architectural level - address contextual factors early in lifecycle (before rest of software architecture work)
- Bayesian Network-based decision support tool (several cases studies examples)

Multilevel Context-Aware Architecture

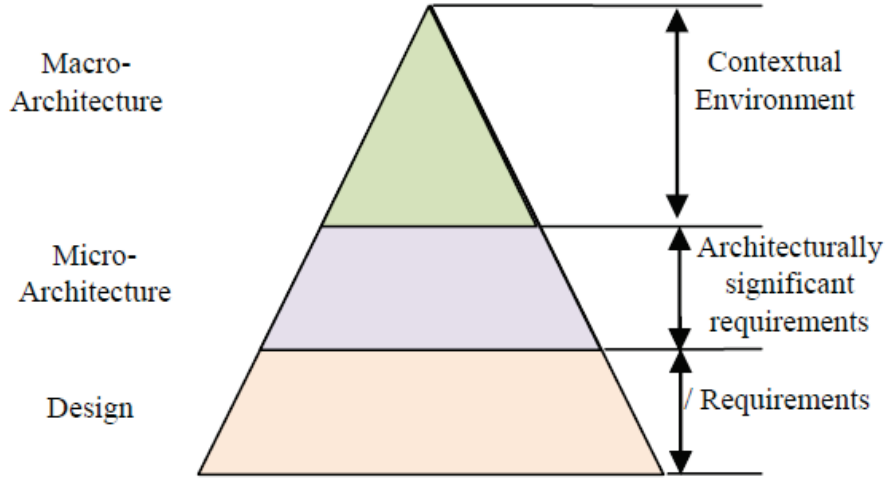



Design Constraints
Quality Attributes / Non-Functional Requirements
Functional Requirements
Enterprise Concerns
System Concerns
Contextual Environment
Define
Infer
Forward Inferred Macro-Architectural Requirements
Influence & Constrain
Define
Infer
Micro-Architecture
Micro-Architectural Decisions
Micro-Architectural Models
Architecture & Design Patterns
Influence & Constrain
Backward Inferred Macro-Architectural Requirements
Infer
Macro-Architecture
Macro-Architectural Decisions
Macro-Architectural Heuristics
Macro-Architectural Patterns

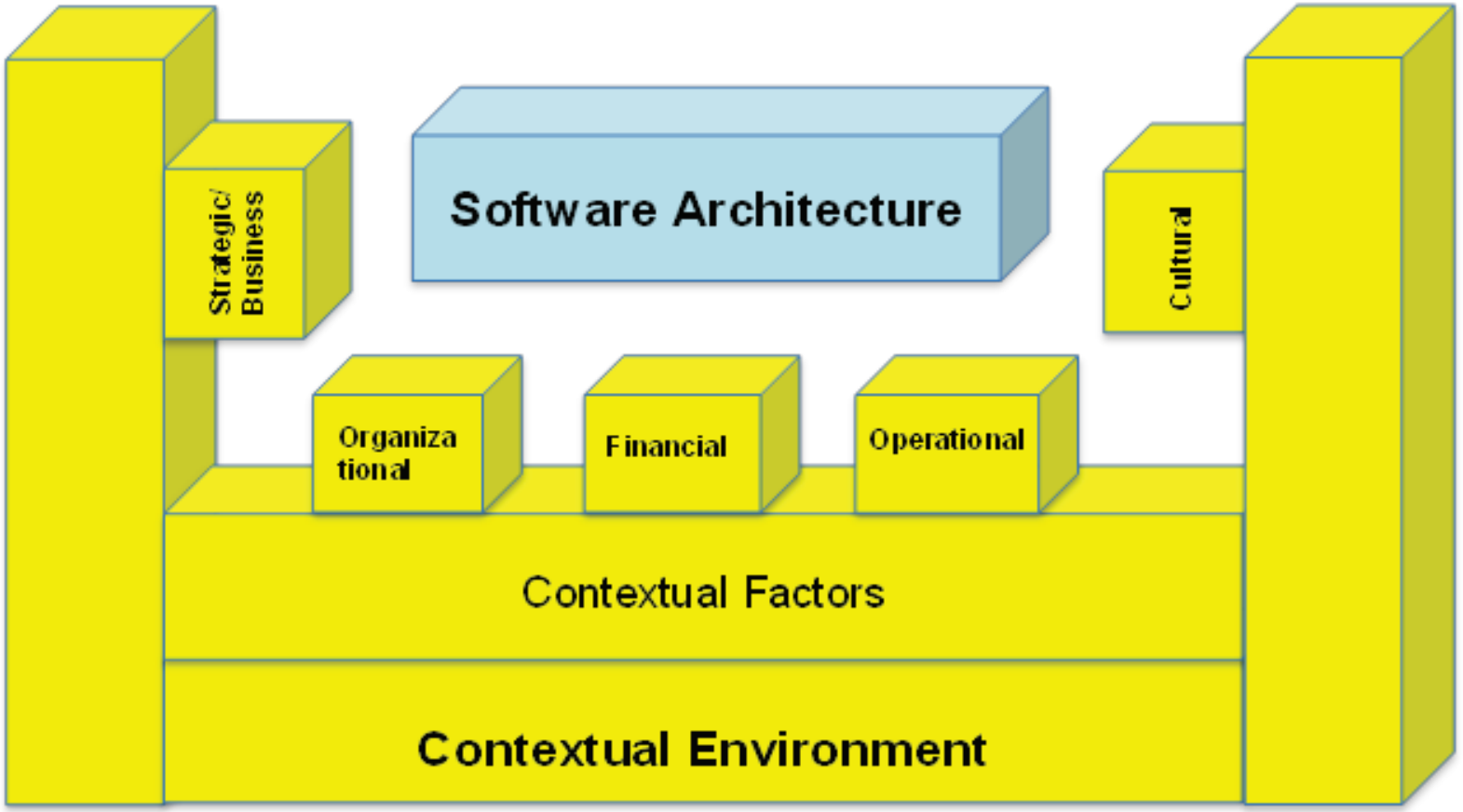
Macro-Architecture as a Decision Analysis Problem



Standard Decision Analysis Model



Software Architecture Adaptation of Decision Analysis Model


Strategic/ Business
Software Architecture
Cultural
Organiza tional
Financial
Operational
Contextual Factors
Contextual Environment

Contextual Factors Definition

- We define **contextual factors** as strategic/business, organizational, financial, operational, and cultural features of the surrounding environment within which the software system is designed.

- The collection of the contextual factors constitutes the **contextual environment** of the system.

  - ✓ Contextual factors are typically not specified explicitly as functional requirements, non-functional requirements or business goals.
  - ✓ Software architecture researchers and practitioners did not consider traditionally these factors as architecturally significant
  - ✓ They have direct or indirect effect on the software architecture and as a consequence on the success or failure of the software system.

Taxonomy / dimensions of the contextual environment:

- Strategic/business
- Organizational
- Financial
- Operational
- Cultural

Contextual Factors Examples

- Strategic/business

  - Target market type
    - ✓ High volume commodity
    - ✓ Low volume niche

  Architectural impact:
    - ✓ High volume commodity
      - ❖ Emphasizes
        - maintainability
      - ❖ Deemphasizes
        - customization

    - ✓ Low volume niche
      - ❖ Emphasizes
        - customization
      - ❖ Deemphasizes
        - maintainability

Contextual Factor examples (cont.):

- Organizational
  - Development Staff Sourcing Model
    - ✓ On staff employees
    - ✓ Contracted augmentation staff
    - ✓ Outsourced consulting
  - Development Staff Skill Level
    - ✓ Experienced
    - ✓ Inexperienced
  - Development Staff Experience Focus
    - ✓ Large scale new software development
    - ✓ Existing software maintenance
    - ✓ Packaged software integration

Contextual Factor examples (cont.):

- Financial
  - Software system funding model
    - ✓ Enterprise-wide pre-allocated
    - ✓ Line-of-business charge back
  - Funding Source
    - ✓ Product marketing profit center
    - ✓ Department cost center
  - Funding horizon
    - ✓ Specific short term feature funding
    - ✓ Multi-year multi-release funding
  - Funding relative priority
    - ✓ Minimize cost as primary consideration
    - ✓ Deliver as specified with cost as secondary consideration

Contextual Factors Examples

Contextual Factor examples (cont.):

- Operational
  - Deployment sourcing
    - ✓ Internal data center
    - ✓ Outsourced data center
  - Software maintenance
    - ✓ Internal team
    - ✓ Outsourced team
  - Hardware diversity support
    - ✓ Single target platform
    - ✓ Few target platforms
    - ✓ Numerous target platforms
  - Deployment instances
    - ✓ Single system deployment
    - ✓ Few instances deployment
    - ✓ Large number of customer deployments

Contextual Factor examples (cont.):

- Cultural
  - Risk profile
    - ✓ Encourages risk taking
    - ✓ Tolerates risk
    - ✓ Risk averse
  - Cost profile
    - ✓ Deliver on projected budget is highest priority
    - ✓ Deliver with aggressive low cost highest priority
  - Timeline profile
    - ✓ Deliver on projected timeline is highest priority
    - ✓ Deliver with aggressive short timeline is highest priority
  - Feature delivery profile
    - ✓ Deliver on projected features is highest priority
    - ✓ Deliver with aggressive number of features is highest priority
  - Quality profile
    - ✓ Deliver on projected quality metrics is highest priority
    - ✓ Deliver with ability to remediate quality issues is highest priority

Contextual Constraints and Considerations

Two types of contextual factors - contextual **constraints** or contextual **considerations**.

- **Contextual constraints** are factors taken for given — externally controlled and non-negotiable with a very high probability that they will not change. We consider the probability that they will not change to be 1.

- **Contextual considerations** are external factors outside of the architect's immediate influence, however with uncertainty happening. Influenced by other factors. Prior probability less than 1 and elicited from subject matter experts and historical data.

- **Macro-architectural preferences** are significant architectural decisions that can be determined from contextual factors with certain level of probability.
  - ✓ Reflect likelihood of success of a software architecture if that decision is made given the specific contextual environment.
  - ✓ Incorporate desirability of the architectural outcome given objective architectural measures and subjective stakeholder utilities.

- Necessarily causal relationship between contextual factors and macro-architectural preferences.

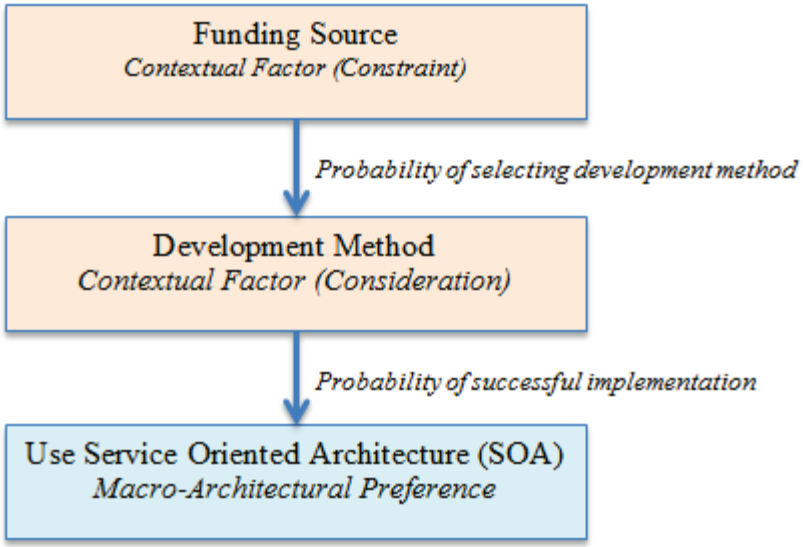
- Named “macro-architectural preferences” and not “decision” to indicate their inherent uncertainty and subjectivity.

Macro-Architectural Preferences - Examples

- Service Oriented Architecture (SOA) as architectural style(1)
  - ✓ Use SOA
  - ✓ Do Not Use SOA
- Concurrent Processing Architecture Style
  - ✓ Multi-threaded with Light Weight Threads (LWT)
  - ✓ Single-threaded with event loop dispatcher
  - ✓ Multi-process forking server
- Reusable libraries
  - ✓ Embed open source libraries
  - ✓ Embed Commercial Off The Shelf (COTS) libraries
  - ✓ Internally developed libraries
- System customization architecture
  - ✓ Configuration file
  - ✓ Domain Specific Language (DSL)
  - ✓ Source code distribution and customization
  - ✓ Custom code exits

(1) http://msdn.microsoft.com/en-us/library/ee658117.aspx
Defined as: "… expose and consume functionality as a service using contracts and messages"

Connecting Contextual Constraints, Contextual Considerations and Macro-Architectural Preferences in a Probabilistic Graphical Model



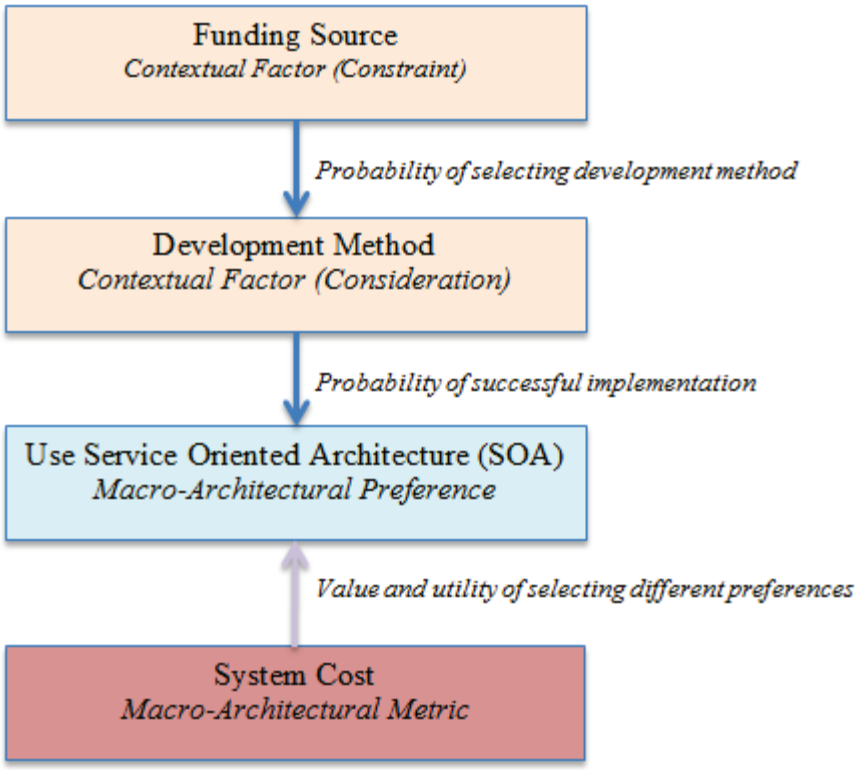
Macro-architectural metrics – quantify the desirability of the architectural outcome given objective measures and subjective stakeholder utilities.

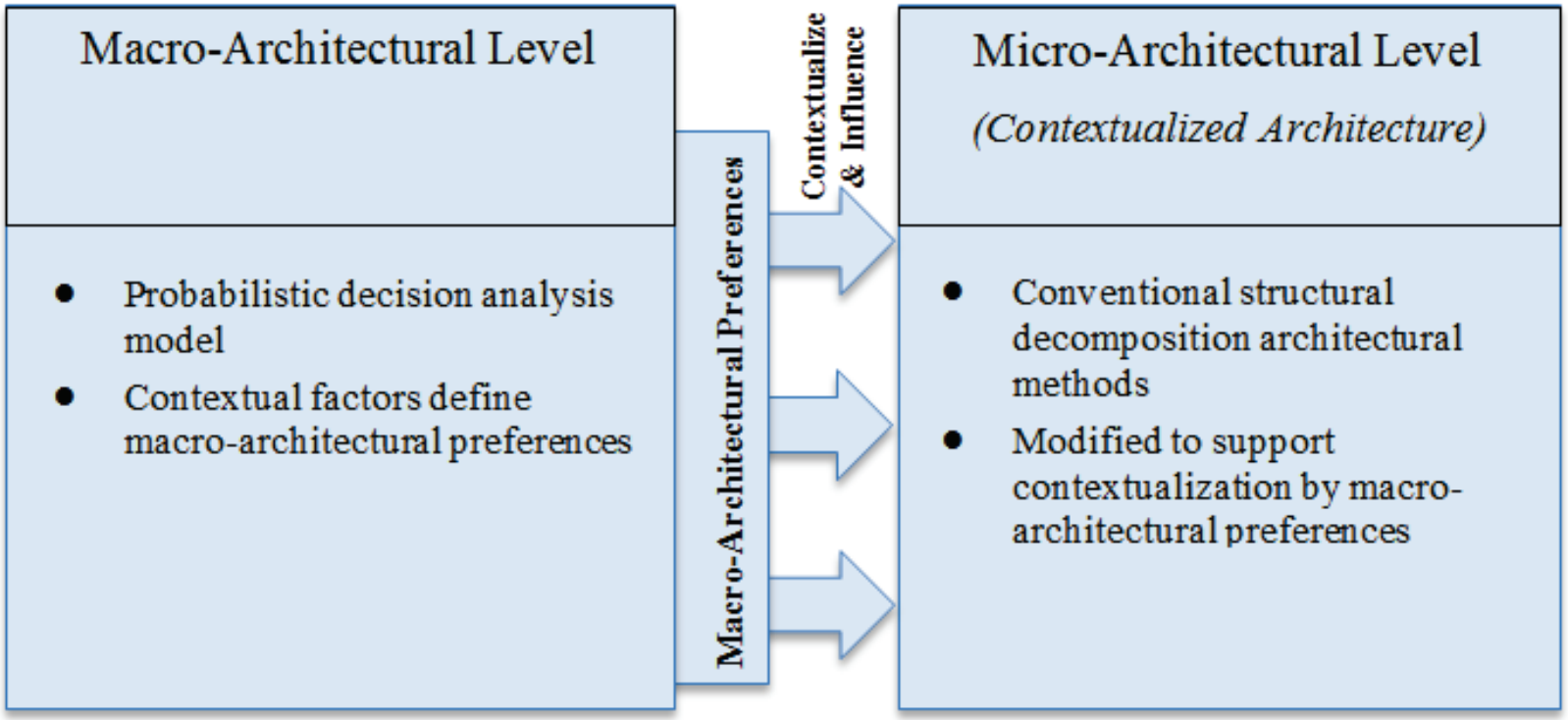
Comprehensive:

- objective "hard" measures that quantify the quality of the architecture, and
- subjective "soft" utilities that capture the desirability of architectural outcomes from the perspective of the key stakeholders.

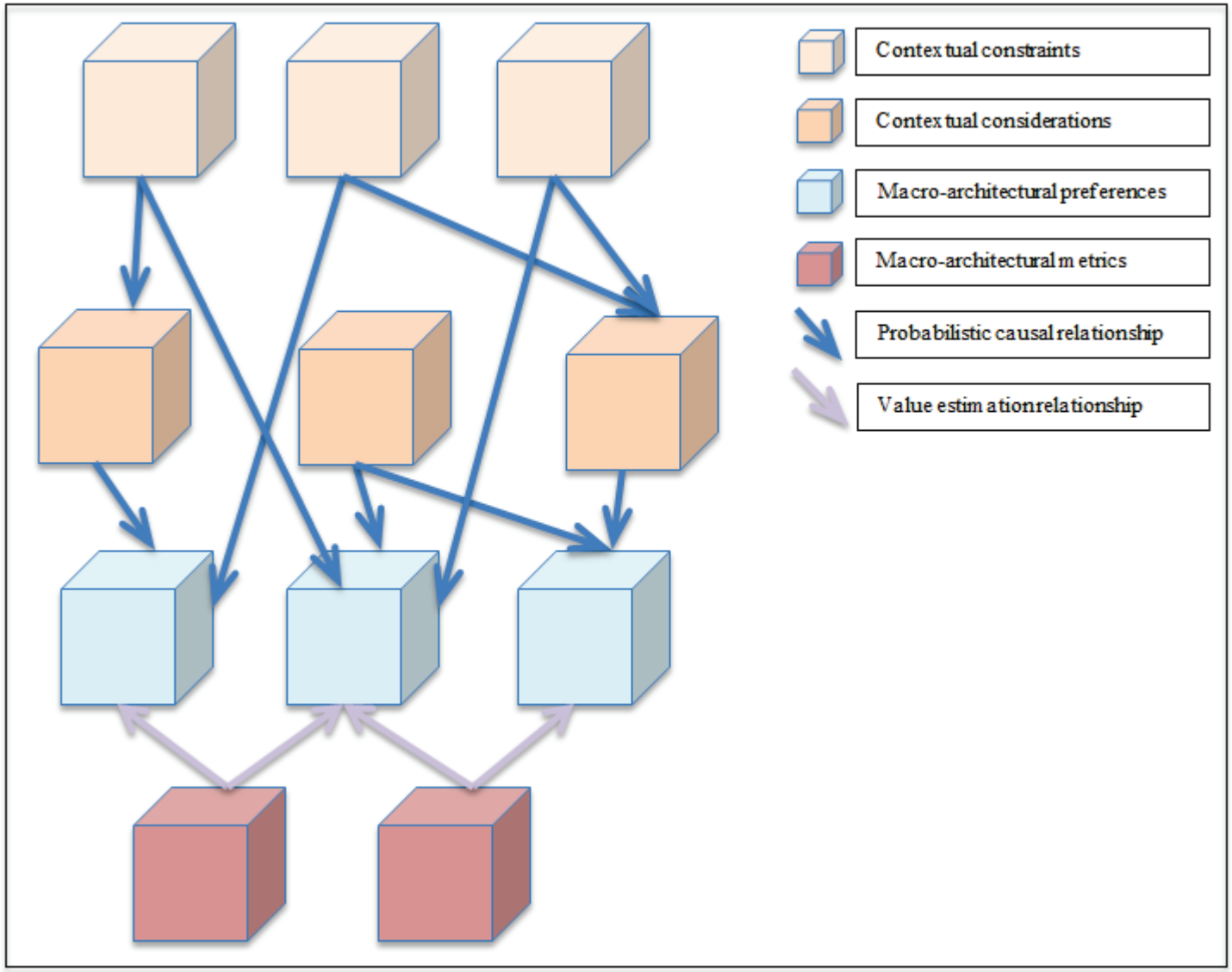
- Categorized into two types:
  - ✓ objective measures
  - ✓ subjective utilities

Connecting Contextual Constraints, Contextual Considerations, Macro-Architectural Preferences and Macro-Architectural Preference in a Probabilistic Graphical Model




Macro-Architectural Level
Probabilistic decision analysis model
Contextual factors define macro-architectural preferences
Macro-Architectural Preferences
Contextualize & Influence
Micro-Architectural Level
(Contextualized Architecture)
Conventional structural decomposition architectural methods
Modified to support contextualization by macro-architectural preferences

Macro-Architecture as Probabilistic Graphical Model



Bayesian Belief and Decision Networks

- Bayesian networks (BNs) are graphical models for reasoning under uncertainty
- Invented by Judea Pearl at UCLA in the 1980's
- Nodes represent variables (discrete or continuous) and arcs represent direct connections between them.
- Direct connections are often causal connections.
- BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available.[1][2]

(1) K. Korb et al, Bayesian Artificial Intelligence, 2nd ed., CRC Press. 2011
(2) D. Koller et al, Probabilistic Graphical Models, MIT Press, 2000

- The structure, or topology, of the network captures qualitative relationships between variables.
- Two nodes should be connected directly if one affects or causes the other, with the arc indicating the direction of the effect.
- Once the topology of the BN is specified - quantify the relationships between connected nodes by specifying a conditional probability distribution for each node.
- Takes the form of a conditional probability table (CPT) [1][2]

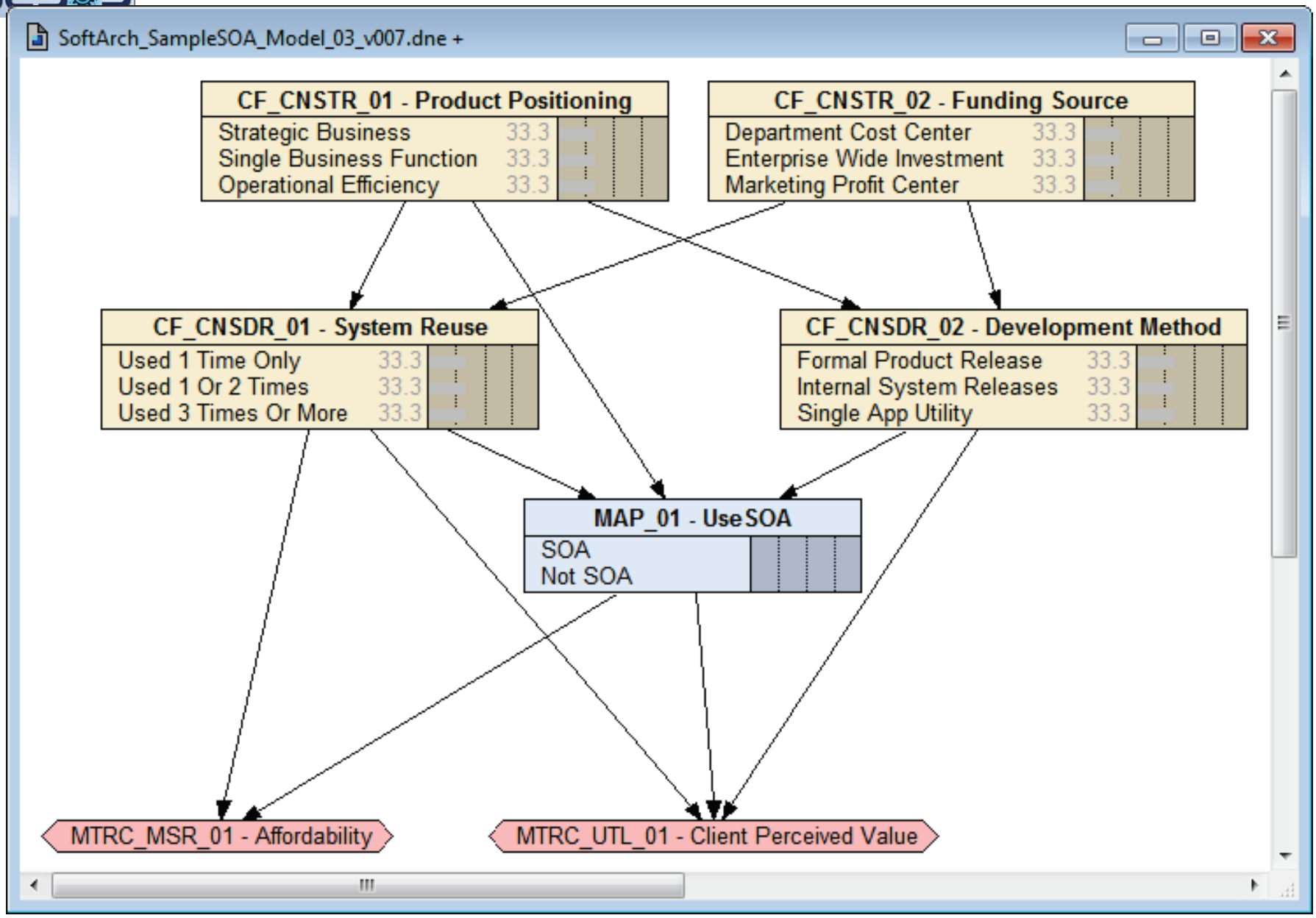
(1) K. Korb et al, Bayesian Artificial Intelligence, 2nd ed., CRC Press. 2011
(2) D. Koller et al, Probabilistic Graphical Models, MIT Press, 2000

Bayesian Belief and Decision Networks

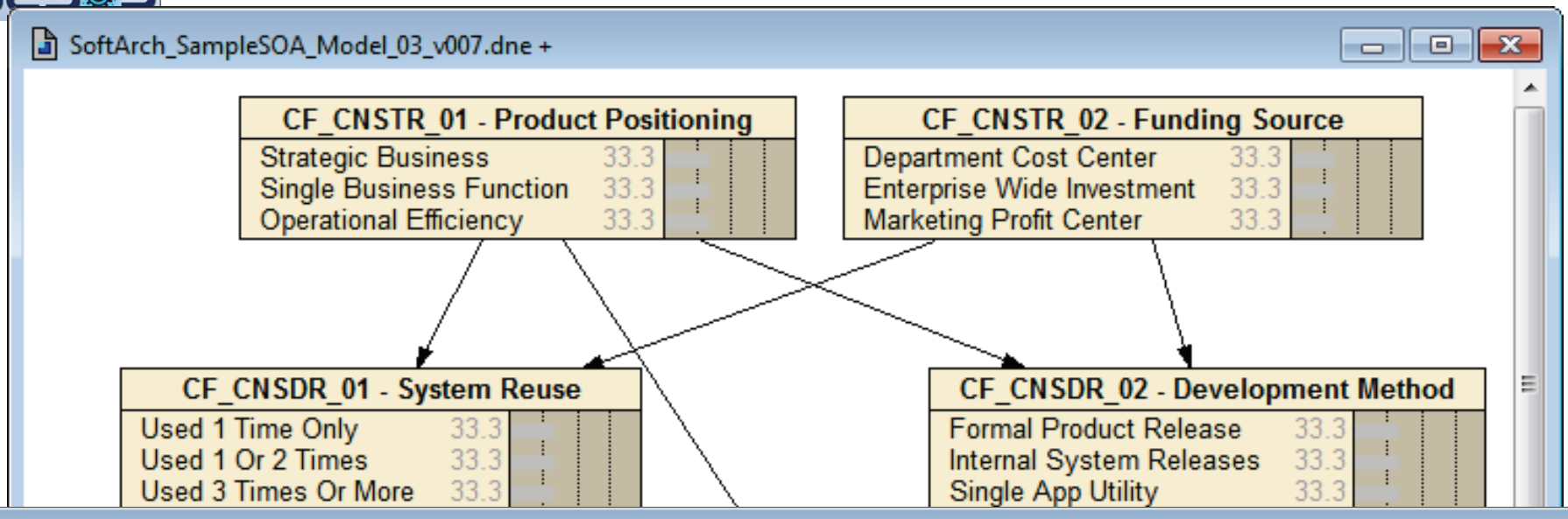
- Markov property of Bayesian Networks - there are no direct dependencies in the system being modeled which are not already explicitly shown via arcs.
- Markov blanket of a node - consists of the node's parents, its children, and its children's parents.
- The joint distribution of the variables in the Markov blanket of a node is sufficient knowledge for calculating the distribution of the node.
- Bayesian network allow us to reason about the domain.
- When we observe the value of some variable - we condition upon the new information.
- The process of conditioning (also called inference or belief updating) is performed via a "flow of information" through the network. (1)(2)

(1) K. Korb et al, Bayesian Artificial Intelligence, 2nd ed., CRC Press. 2011
(2) D. Koller et al, Probabilistic Graphical Models, MIT Press, 2000

Bayesian Decision Network - Example



Bayesian Decision Network - Example



SysReuse Table (in Bayes net SoftArch_Decide_Model_01_v001)

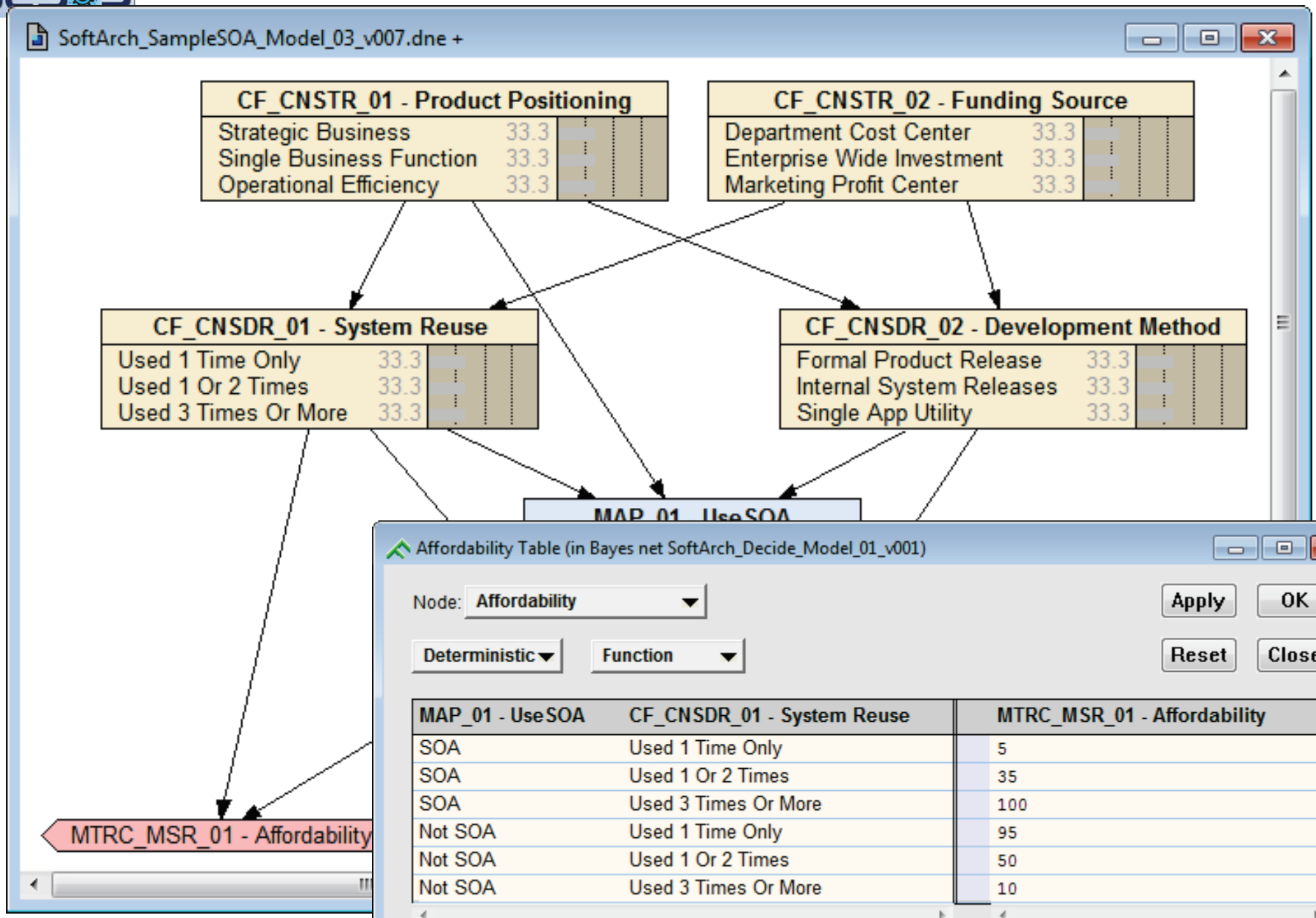
Node: SysReuse

Apply | OK | Reset | Close

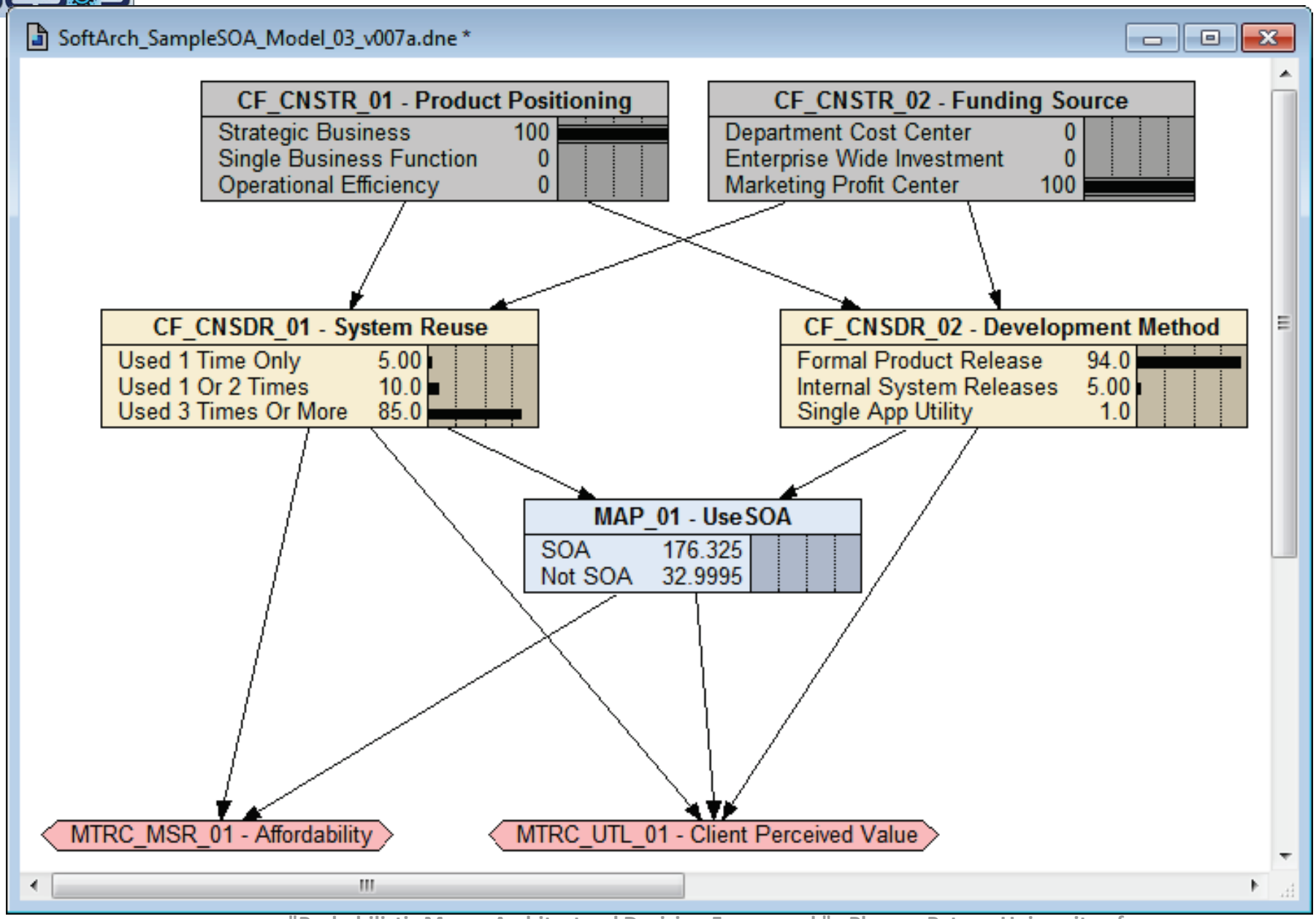
Chance | % Probability

CF_CNSTR_01 - Product Positio...	CF_CNSTR_02 - Funding Source	Used 1 Time Only	Used 1 Or 2 Times	Used 3 Times Or More
Strategic Business	Department Cost Center	30	30	40
Strategic Business	Enterprise Wide Investment	10	20	70
Strategic Business	Marketing Profit Center	5	10	85
Single Business Function	Department Cost Center	79	20	1
Single Business Function	Enterprise Wide Investment	30	25	45
Single Business Function	Marketing Profit Center	75	20	5
Operational Efficiency	Department Cost Center	85	10	5
Operational Efficiency	Enterprise Wide Investment	55	20	25
Operational Efficiency	Marketing Profit Center	75	15	10

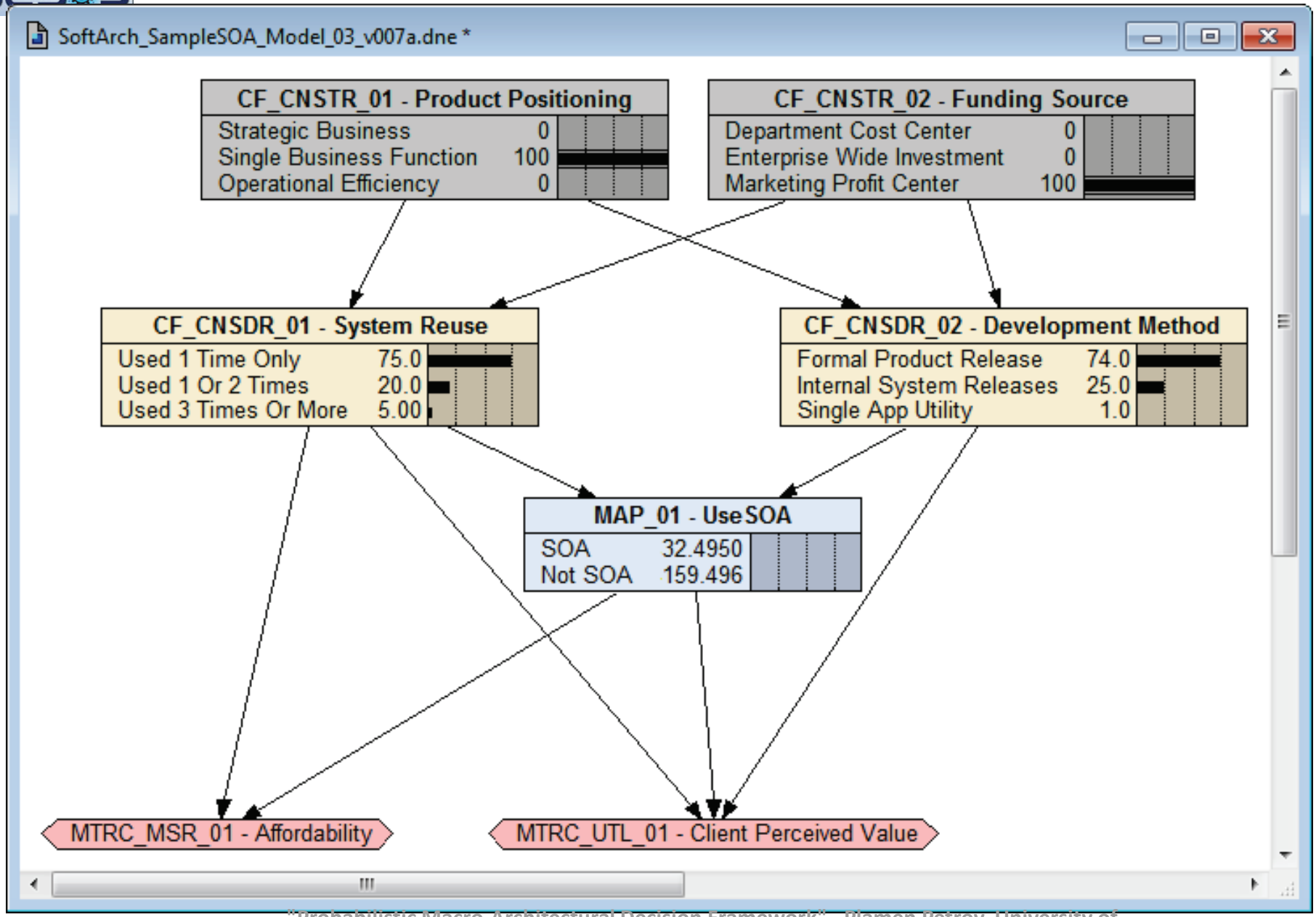
Bayesian Decision Network - Example



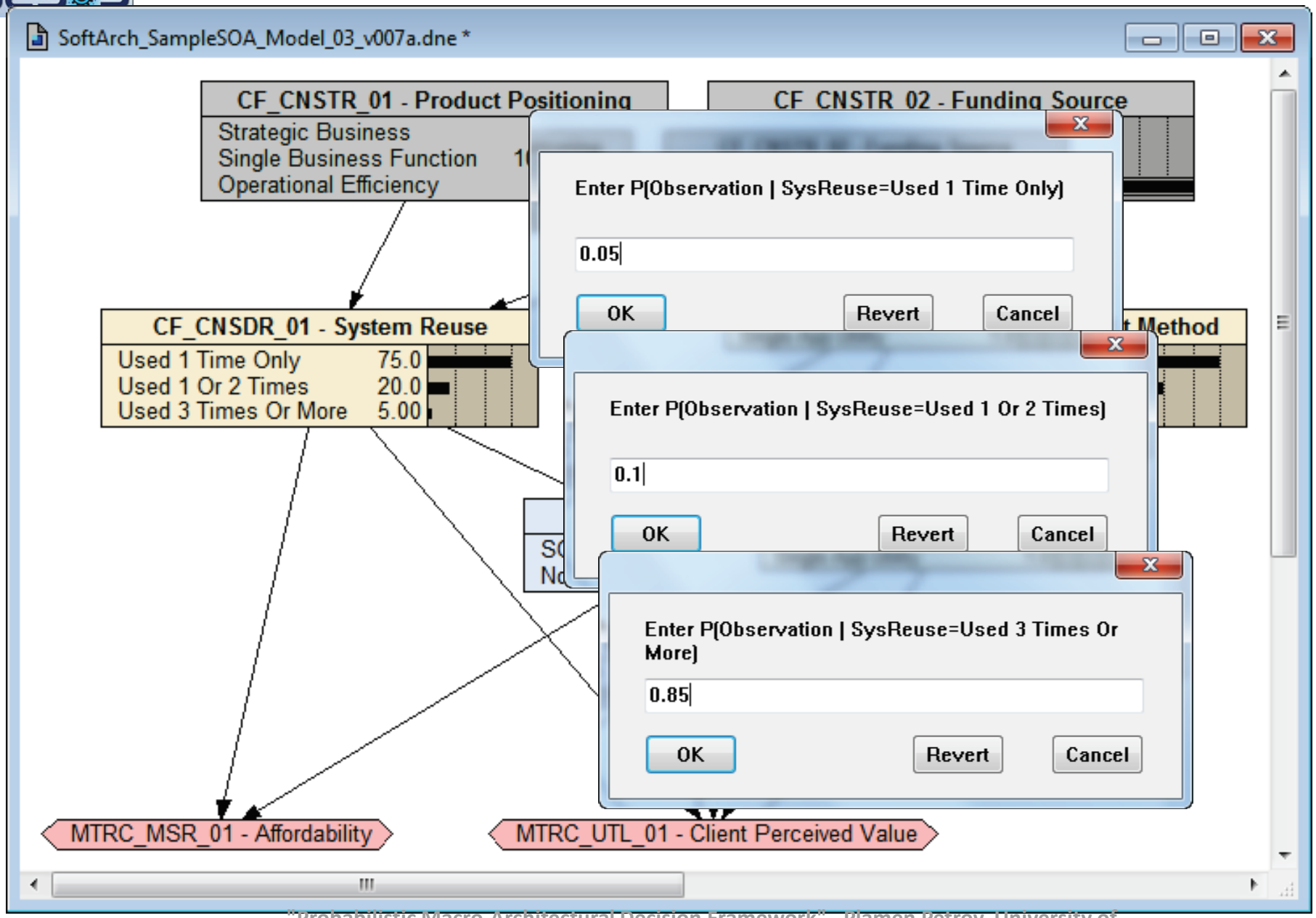
Bayesian Decision Network - Example



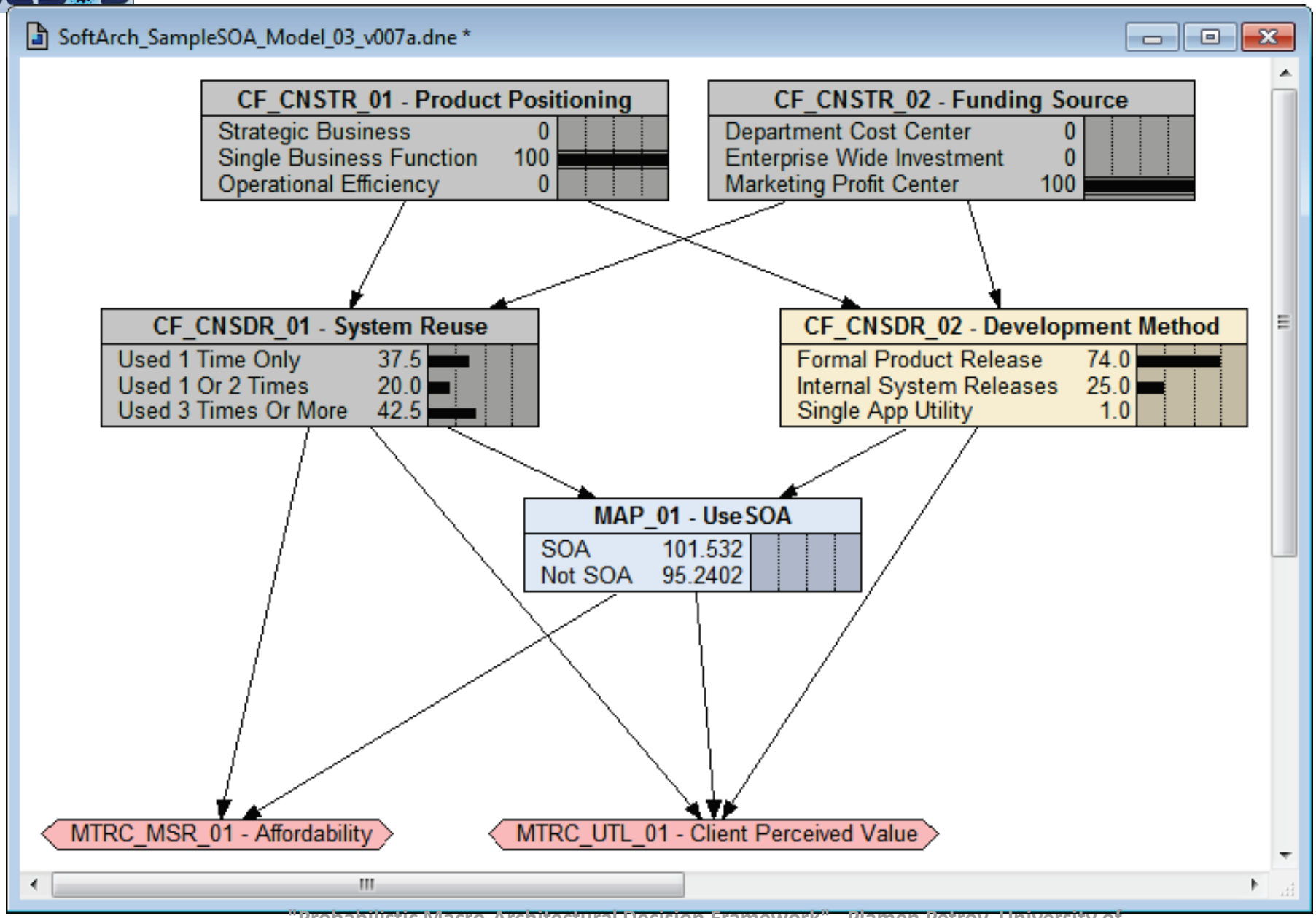
Bayesian Decision Network - Example



Bayesian Decision Network - Example



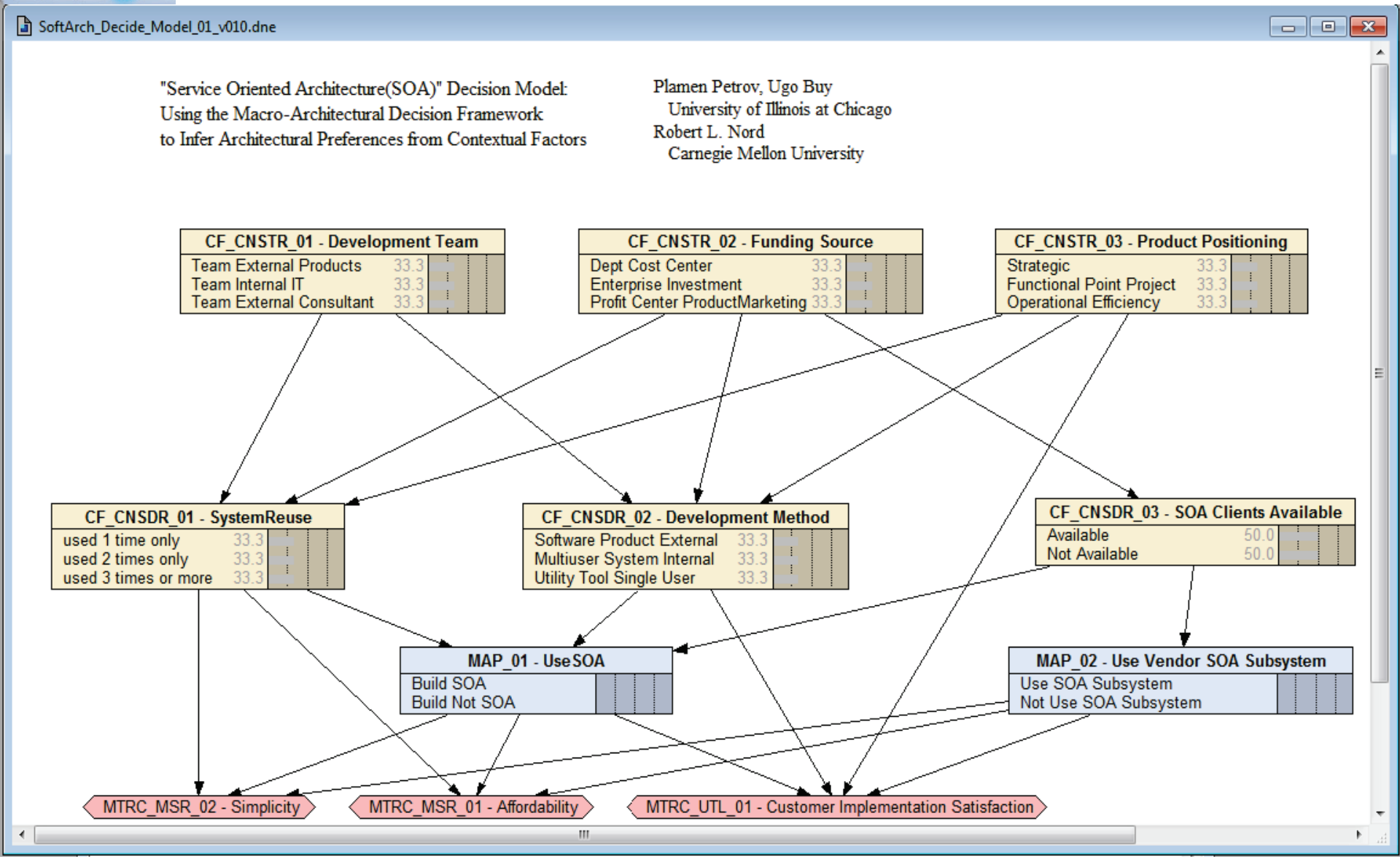
Bayesian Decision Network - Example



Case Study – Service Oriented Architecture (SOA)

- CF_CNSTR_01 – Development Experience and Skillset
- CF_CNSTR_02 – Project Funding Source
- CF_CNSTR_03 – Product Positioning
- CF_CNSDR_01 – System Reuse
- CF_CNSDR_02 – Software Development Method
- CF_CNSDR_03 – 3rd party SOA enabled subsystems
- MAP_01 – Use Service Oriented Architecture (SOA)
- MAP_02 – Use 3rd party SOA enabled subsystem
- MTRC_MSR_01 – Affordability
- MTRC_MSR_02 – Simplicity
- MTRC_UTL_01 – Customer Satisfaction

Case Study – Service Oriented Architecture (SOA)



Case Study – Service Oriented Architecture (SOA)

SysReuse Table (in Bayes net SoftArch_Decide_Model_01_v001)

Node: SysReuse

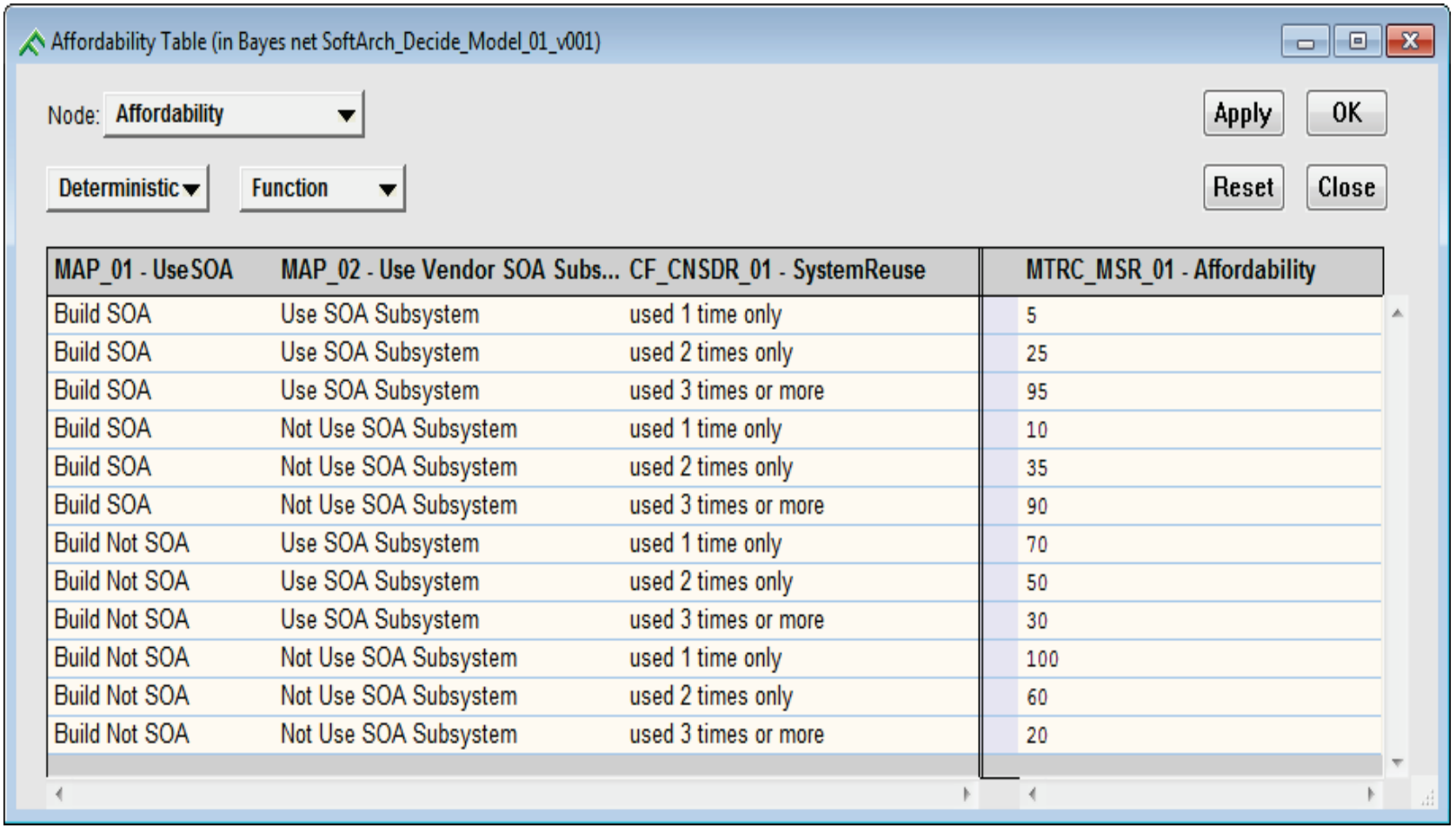
Apply OK

Chance % Probability

Reset Close

CF_CNSTR_02 - Funding Source	CF_CNSTR_03 - Product Positio...	CF_CNSTR_01 - Development T...	used 1 time only	used 2 times only	used 3 times or more
Dept Cost Center	Strategic	Team External Products	84	10	6
Dept Cost Center	Strategic	Team Internal IT	90	8	2
Dept Cost Center	Strategic	Team External Consultant	89	9	2
Dept Cost Center	Functional Point Project	Team External Products	86	10	4
Dept Cost Center	Functional Point Project	Team Internal IT	90	8	2
Dept Cost Center	Functional Point Project	Team External Consultant	88	9	3
Dept Cost Center	Operational Efficiency	Team External Products	91	6	3
Dept Cost Center	Operational Efficiency	Team Internal IT	95	4	1
Dept Cost Center	Operational Efficiency	Team External Consultant	95	3	2
Enterprise Investment	Strategic	Team External Products	7	19	74
Enterprise Investment	Strategic	Team Internal IT	45	25	30
Enterprise Investment	Strategic	Team External Consultant	46	26	28
Enterprise Investment	Functional Point Project	Team External Products	44	22	34
Enterprise Investment	Functional Point Project	Team Internal IT	73	18	9
Enterprise Investment	Functional Point Project	Team External Consultant	78	16	6
Enterprise Investment	Operational Efficiency	Team External Products	65	25	10
Enterprise Investment	Operational Efficiency	Team Internal IT	77	16	7
Enterprise Investment	Operational Efficiency	Team External Consultant	66	23	11
Profit Center ProductMarketing	Strategic	Team External Products	4	6	90
Profit Center ProductMarketing	Strategic	Team Internal IT	60	15	25
Profit Center ProductMarketing	Strategic	Team External Consultant	7	18	75
Profit Center ProductMarketing	Functional Point Project	Team External Products	79	12	9
Profit Center ProductMarketing	Functional Point Project	Team Internal IT	90	4	6
Profit Center ProductMarketing	Functional Point Project	Team External Consultant	87	5	8
Profit Center ProductMarketing	Operational Efficiency	Team External Products	81	11	8
Profit Center ProductMarketing	Operational Efficiency	Team Internal IT	86	9	5
Profit Center ProductMarketing	Operational Efficiency	Team External Consultant	81	10	9

Case Study – Service Oriented Architecture (SOA)



Affordability Table (in Bayes net SoftArch_Decide_Model_01_v001)

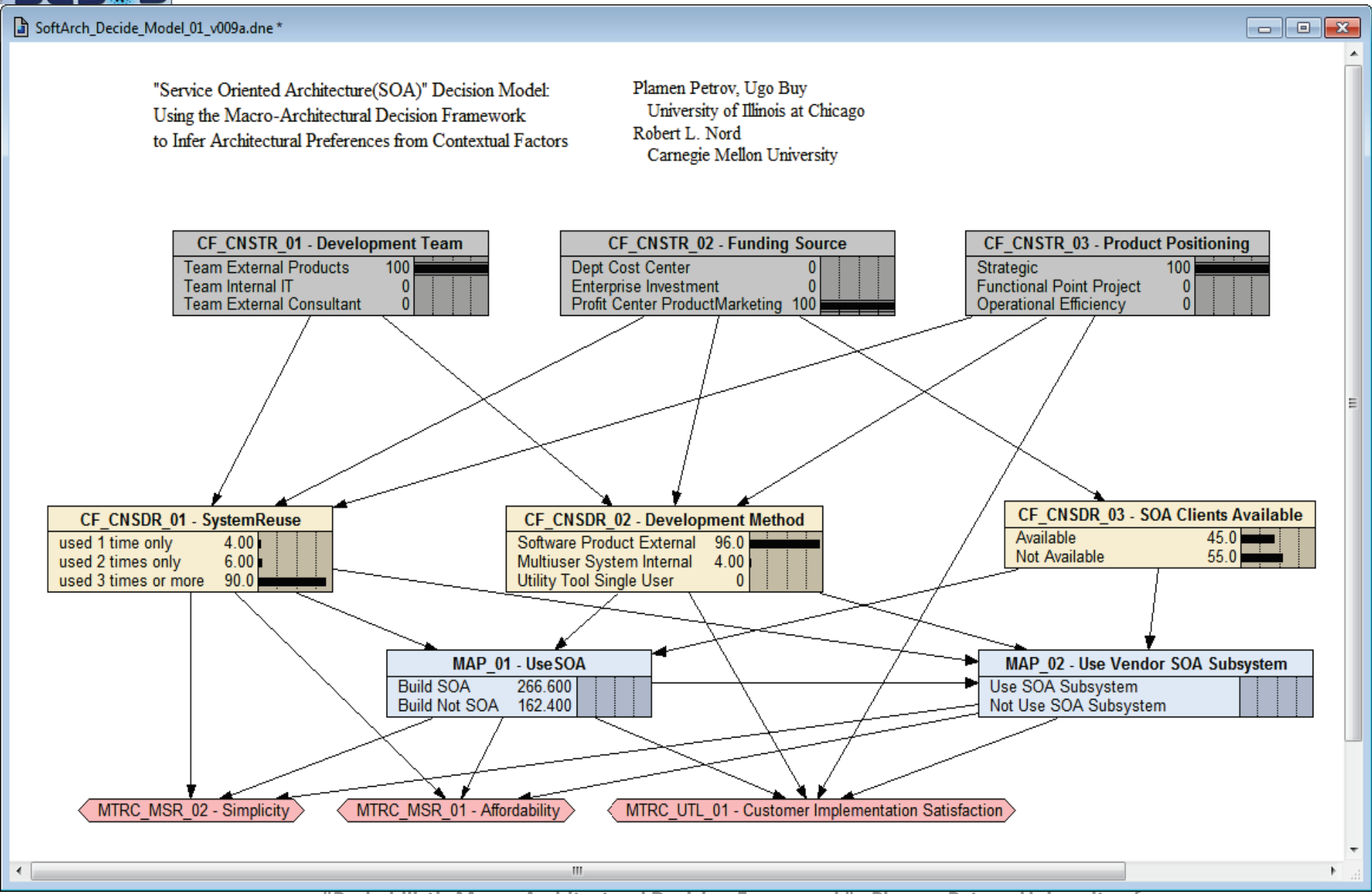
Node: Affordability

Apply | OK | Reset | Close

Deterministic | Function

MAP_01 - UseSOA	MAP_02 - Use Vendor SOA Subs...	CF_CNSDR_01 - SystemReuse	MTRC_MSR_01 - Affordability
Build SOA	Use SOA Subsystem	used 1 time only	5
Build SOA	Use SOA Subsystem	used 2 times only	25
Build SOA	Use SOA Subsystem	used 3 times or more	95
Build SOA	Not Use SOA Subsystem	used 1 time only	10
Build SOA	Not Use SOA Subsystem	used 2 times only	35
Build SOA	Not Use SOA Subsystem	used 3 times or more	90
Build Not SOA	Use SOA Subsystem	used 1 time only	70
Build Not SOA	Use SOA Subsystem	used 2 times only	50
Build Not SOA	Use SOA Subsystem	used 3 times or more	30
Build Not SOA	Not Use SOA Subsystem	used 1 time only	100
Build Not SOA	Not Use SOA Subsystem	used 2 times only	60
Build Not SOA	Not Use SOA Subsystem	used 3 times or more	20

Case Study – Service Oriented Architecture (SOA)

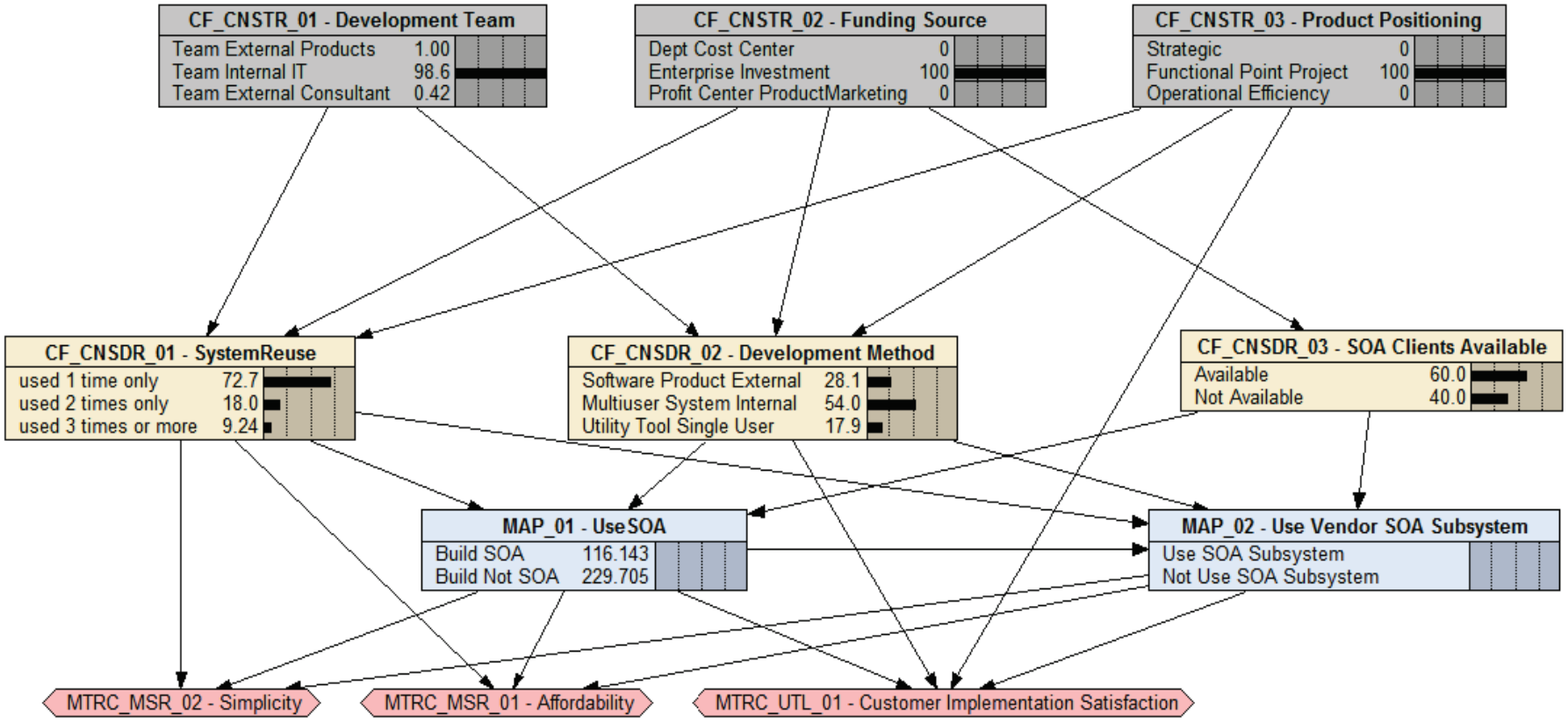


Case Study – Service Oriented Architecture (SOA)

SoftArch_Decide_Model_01_v009a.dne *

"Service Oriented Architecture(SOA)" Decision Model:
Using the Macro-Architectural Decision Framework
to Infer Architectural Preferences from Contextual Factors

Plamen Petrov, Ugo Buy
University of Illinois at Chicago
Robert L. Nord
Carnegie Mellon University

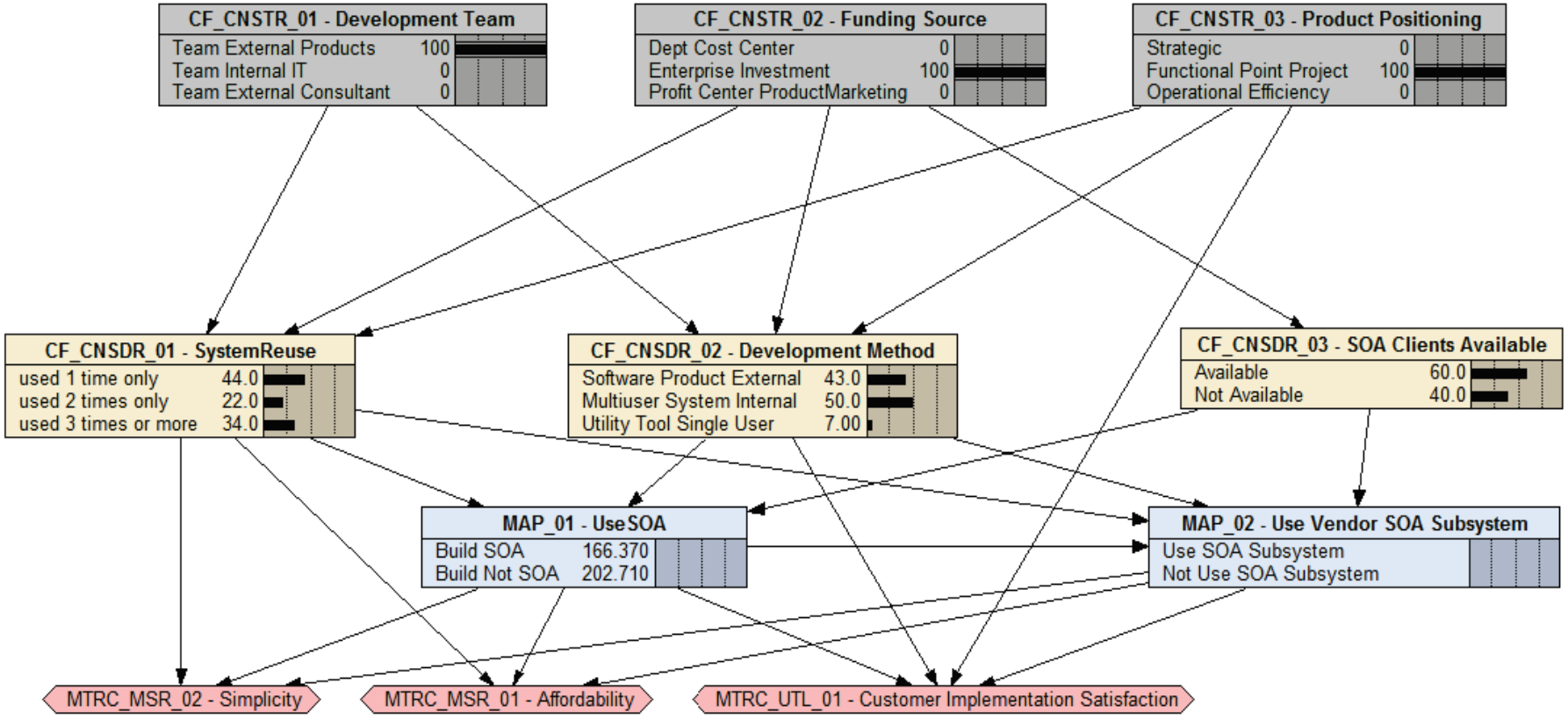


Case Study – Service Oriented Architecture (SOA)

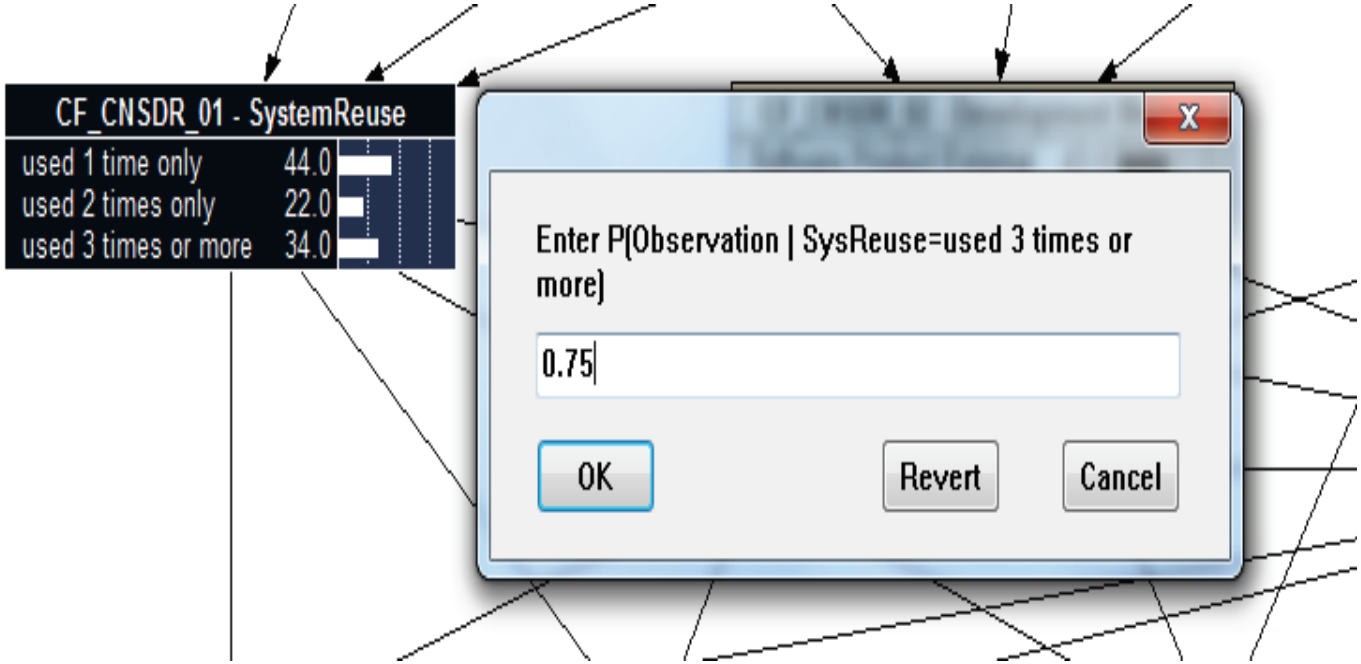
SoftArch_Decide_Model_01_v009a.dne *

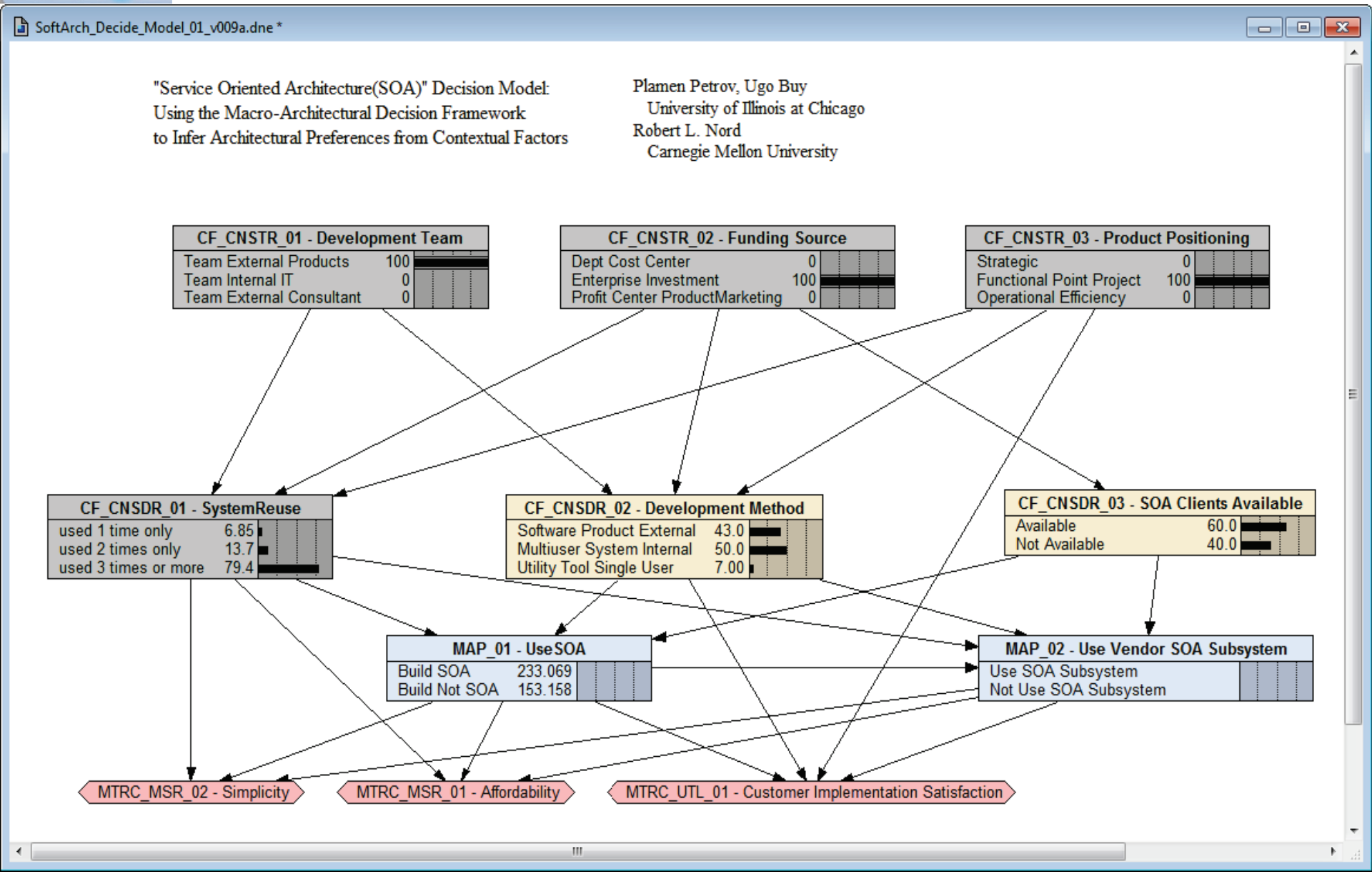
"Service Oriented Architecture(SOA)" Decision Model:
Using the Macro-Architectural Decision Framework
to Infer Architectural Preferences from Contextual Factors

Plamen Petrov, Ugo Buy
University of Illinois at Chicago
Robert L. Nord
Carnegie Mellon University

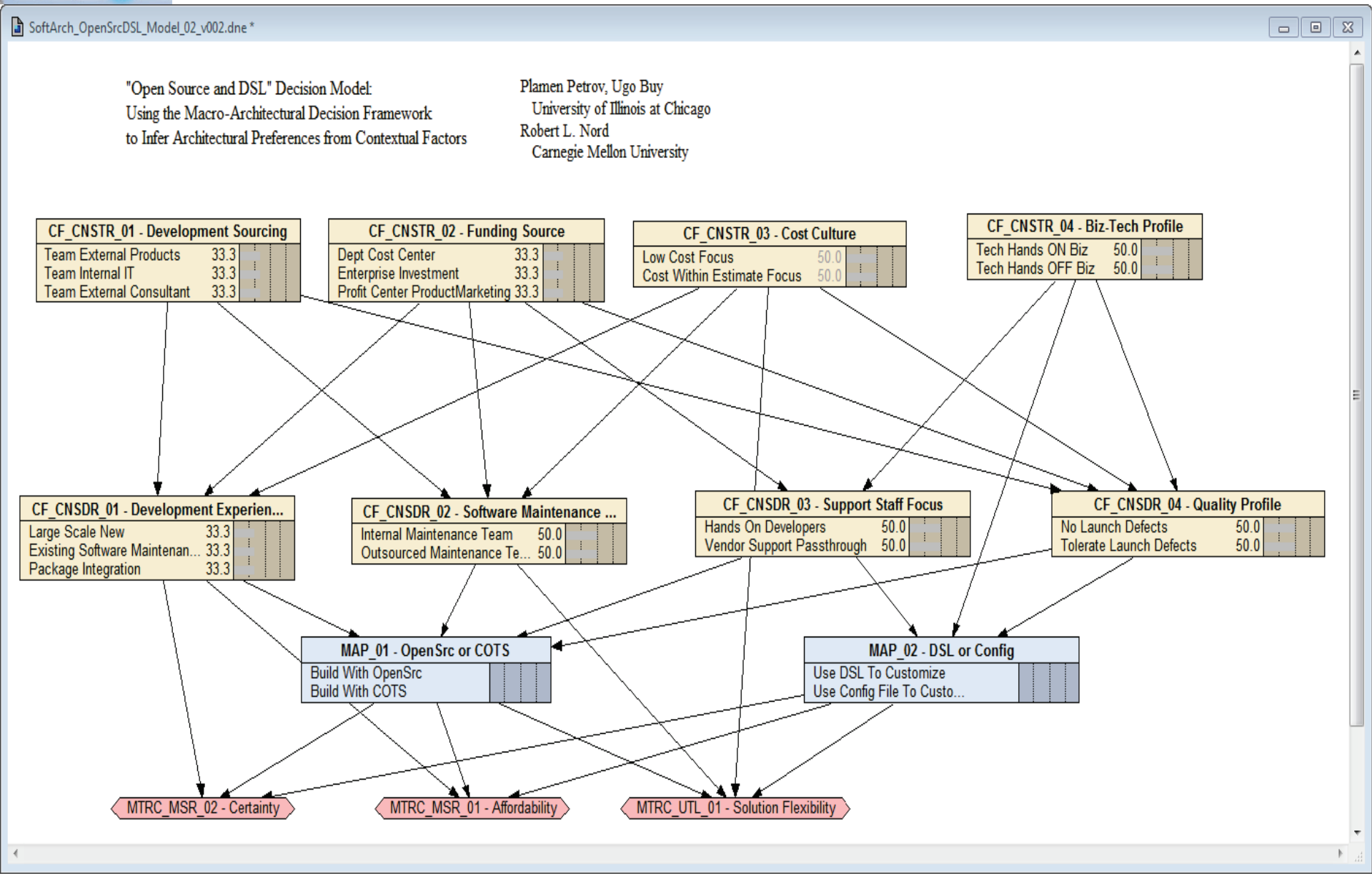


Case Study – Service Oriented Architecture (SOA)




SoftArch_Decide_Model_01_v009a.dne *
"Service Oriented Architecture(SOA)" Decision Model: Using the Macro-Architectural Decision Framework to Infer Architectural Preferences from Contextual Factors
Plamen Petrov, Ugo Buy
University of Illinois at Chicago
Robert L. Nord
Carnegie Mellon University
CF_CNSTR_01 - Development Team
Team External Products 100
Team Internal IT 0
Team External Consultant 0
CF_CNSTR_02 - Funding Source
Dept Cost Center 0
Enterprise Investment 100
Profit Center ProductMarketing 0
CF_CNSTR_03 - Product Positioning
Strategic 0
Functional Point Project 100
Operational Efficiency 0
CF_CNSDR_01 - SystemReuse
used 1 time only 6.85
used 2 times only 13.7
used 3 times or more 79.4
CF_CNSDR_02 - Development Method
Software Product External 43.0
Multiuser System Internal 50.0
Utility Tool Single User 7.00
CF_CNSDR_03 - SOA Clients Available
Available 60.0
Not Available 40.0
MAP_01 - UseSOA
Build SOA 233.069
Build Not SOA 153.158
MAP_02 - Use Vendor SOA Subsystem
Use SOA Subsystem
Not Use SOA Subsystem
MTRC_MSR_02 - Simplicity
MTRC_MSR_01 - Affordability
MTRC_UTL_01 - Customer Implementation Satisfaction

Case Study: "DSL-Config file" and "Open Source–COTS"



Conclusion and Future Research

- We argue that the method and framework that we introduced significantly contributes to improving the quality of software architecture practices and the software development outcomes.
- Our results to date confirm the power and relevance of the framework
- Intend on continuing our research in this area and suggest the following research problems as extensions:
  - ✓ Automate the process of discovery of new factors, preferences and their relationship through data mining and machine learning techniques.
  - ✓ Expand the catalogue of contextual factors, macro-architectural preferences and causal dependencies

Nice Alignment

The 8th European Conference on Software Architecture (ECSA)
University of Vienna, Austria, from August 25 to 29, 2014

Keynote Speaker: Hans van Vliet: **Architecting = Decision Making**
Professor in Software Engineering at the VU University Amsterdam, The Netherlands

http://ecsa2014.cs.univie.ac.at/program/keynote-speakers/

" … Do experienced architects make better decisions than novice architects? Can the architecting process be rational, or is it affected by the same irrationalities one sees in everyday decision making? Can we discover when design decisions are biased? If so, how and when. …
… I will sketch the evolution of our thinking of what constitutes software architecture, and the kind of research questions that arise if we view architecting as decision making. "

Related Publications

[1] P. Petrov and U. Buy, “A Systemic Methodology for Software Architecture Analysis and Design,” in 2011 Eighth International Conference on Information Technology: New Generations, Las Vegas, 2011.

[2] P. Petrov, U. Buy, and R. L. Nord, “The Need for a Multilevel Context-Aware Software Architecture Analysis and Design Method with Enterprise and System Architecture Concerns as First Class Entities,” in 2011 Ninth Working Conference on Software Architecture, Boulder, CO USA, 2011.

[3] P. Petrov, U. Buy, and R. L. Nord, “Enhancing the software architecture analysis and design process with inferred macro-architectural requirements,” 1st Intl. Workshop on the Twin Peaks of Requirements and Architecture at the 20th IEEE Intl. Requirements Engineering Conference, 2012, pp. 20–26.

Thank You

Plamen Petrov

ppetro2@uic.edu