16 October, 2013

Introduction to the User Requirements Notation (URN)

Gunter Mussbacher

ECE, McGill University, Canada ◄► gunter.mussbacher@mcgill.ca

Motivation

vision

Model business goals, stakeholders' priorities, alternative solutions, rationale, and **decisions**

Model & test **use cases**; investigate high level architecture; transform to more detailed models

more detailed models

As this tutorial builds on a long history of other tutorials, a special acknowledgement is due to Daniel Amyot (University of Ottawa, Canada)

Table of Contents

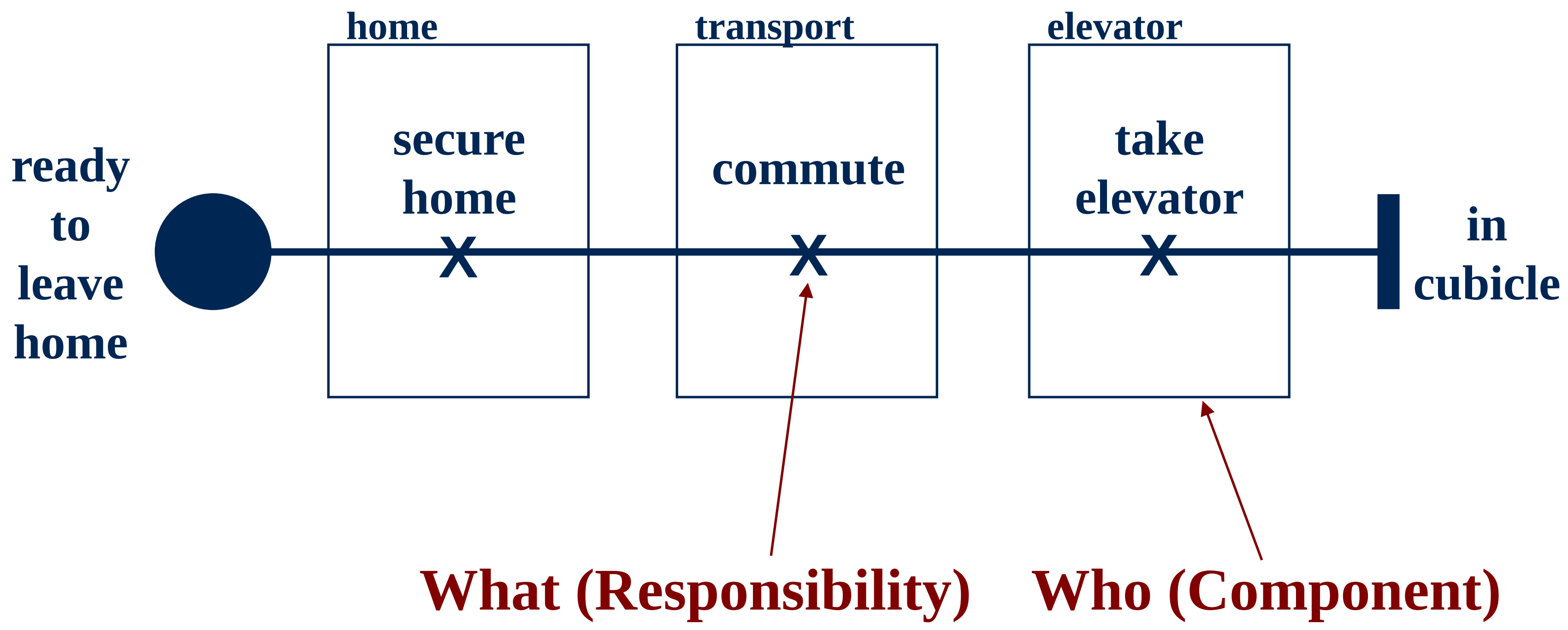
A Simple Problem: What is the "best way" for you to commute?

Problem Description

- Need to commute to work 230 days of the year
- Need to spend considerable amounts of time and money on the commute
- Various solutions for commuting are available
  - Various types of public transport
  - Car (own car or colleague's car)
- What is the "best" solution for you to choose and why?

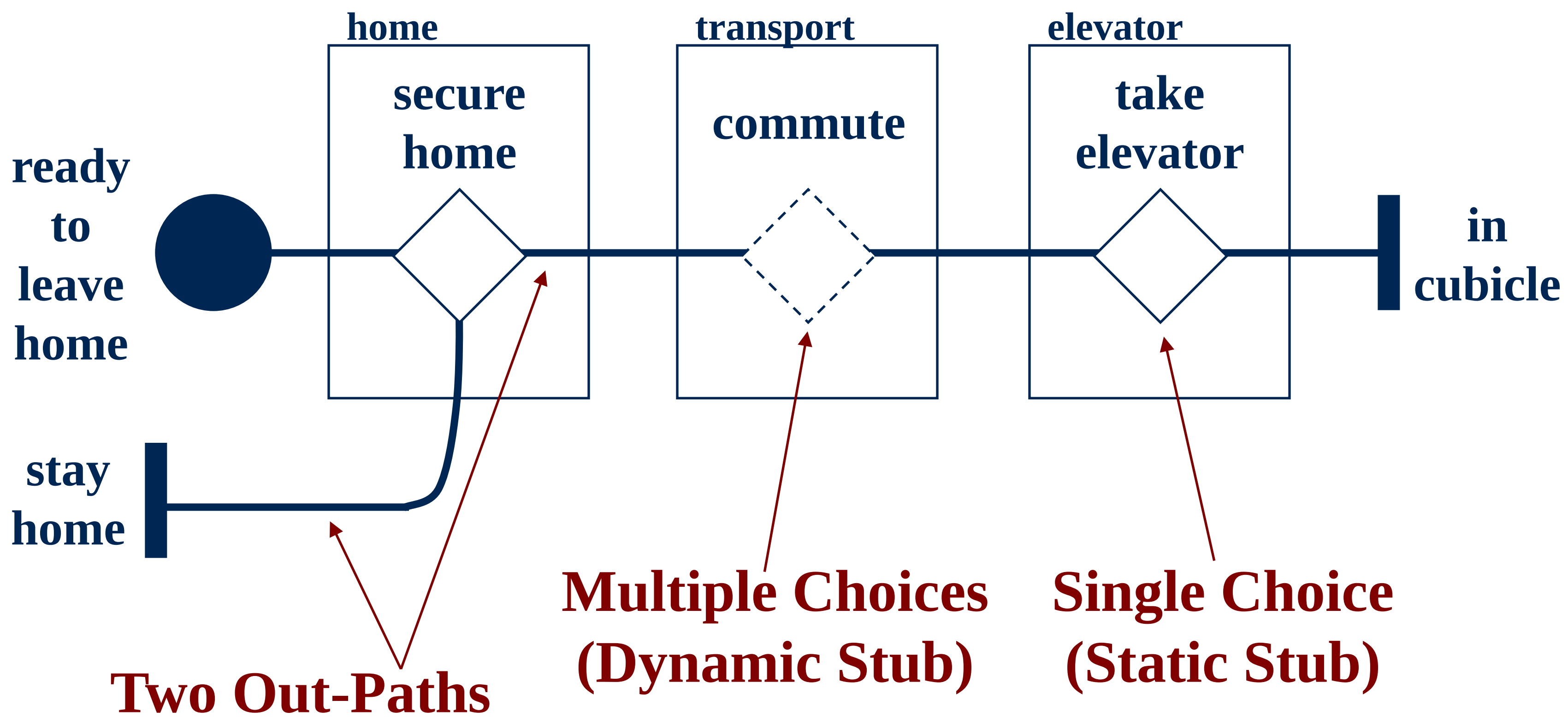
Basic Modeling of Solution

Example: Commuting

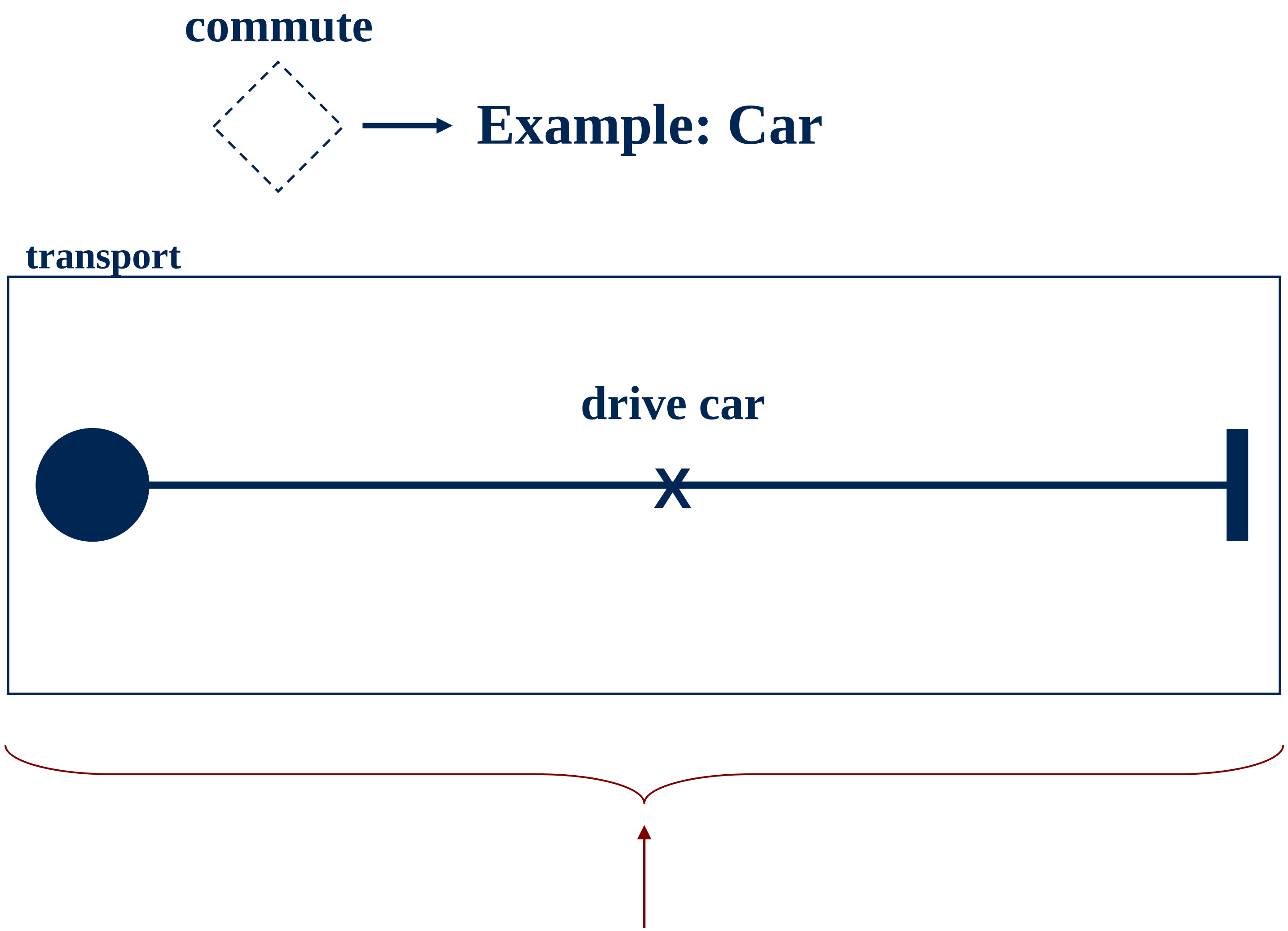


Hierarchical Structuring (1/3)

Example: Commuting

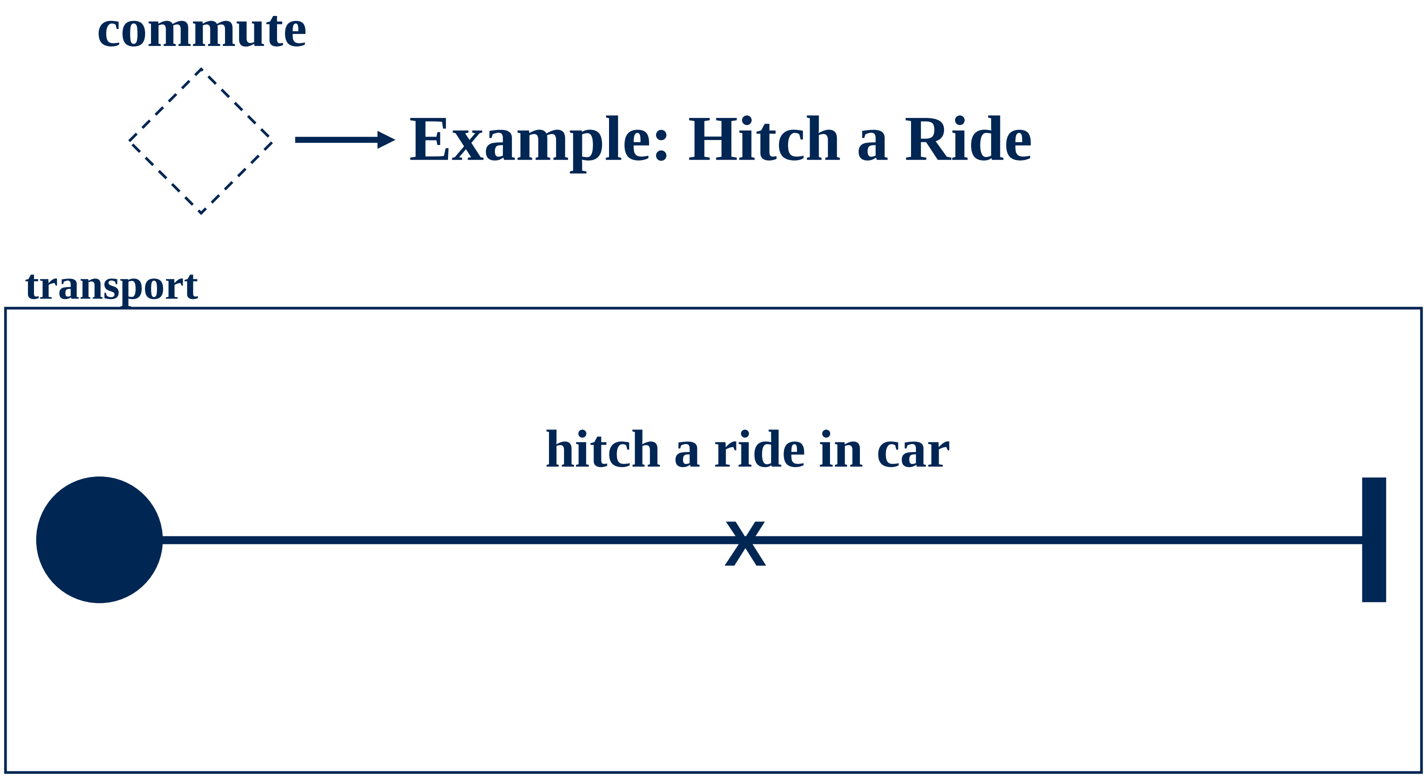


Hierarchical Structuring (2/3)

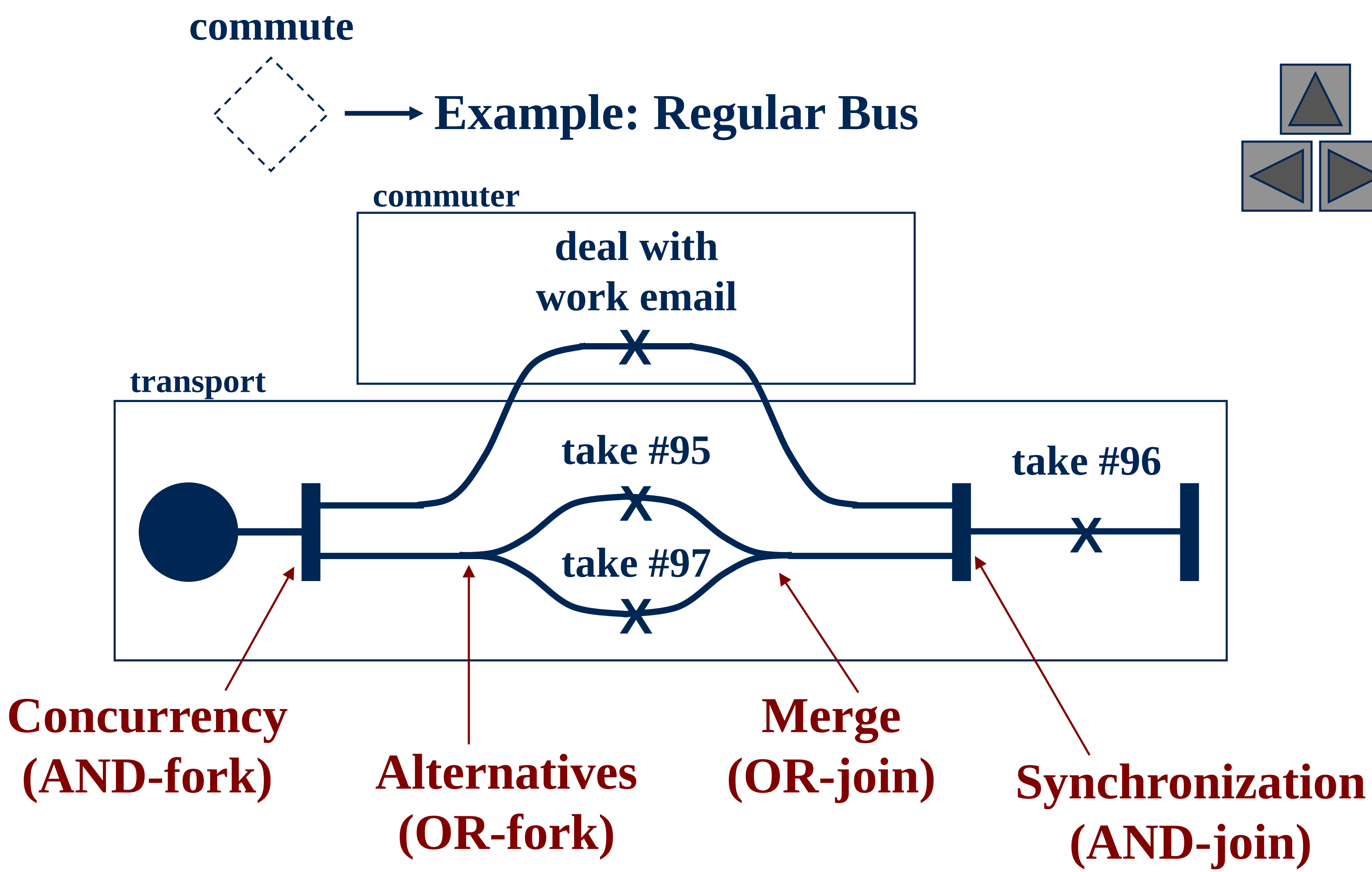


Plug-in Map (plugged into stub)

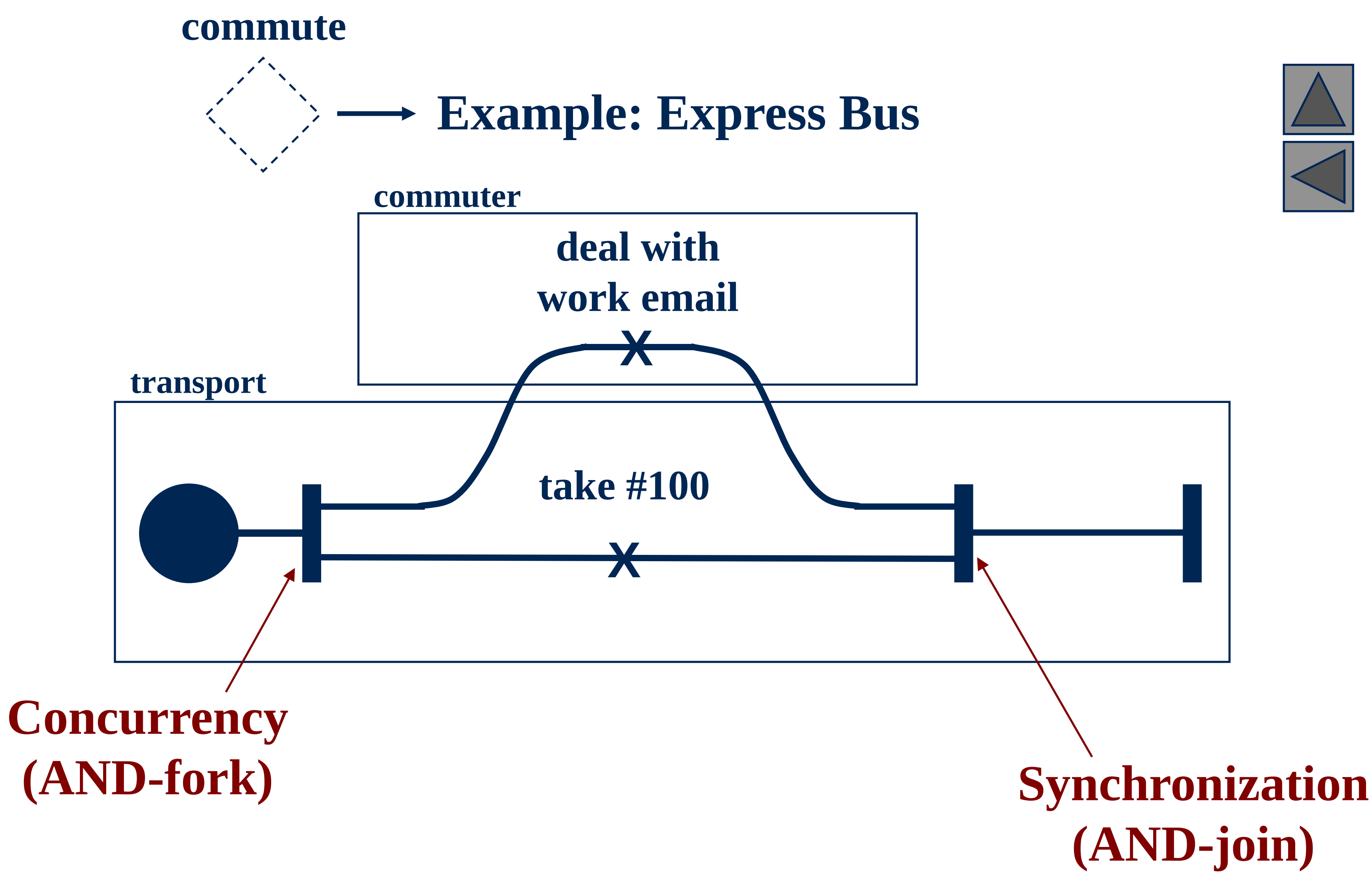
Hierarchical Structuring (3/3)



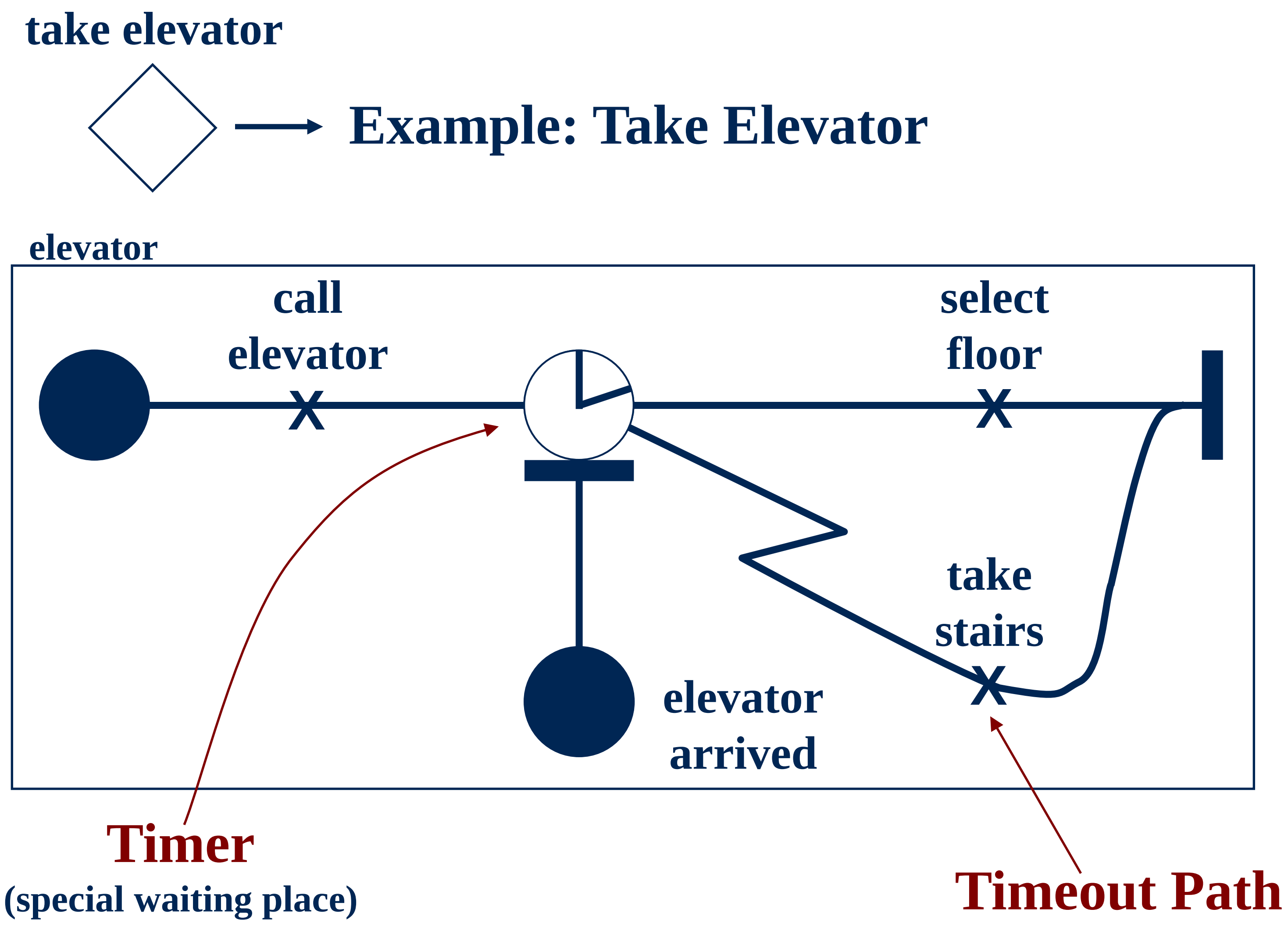
Alternatives and Concurrency (1/2)



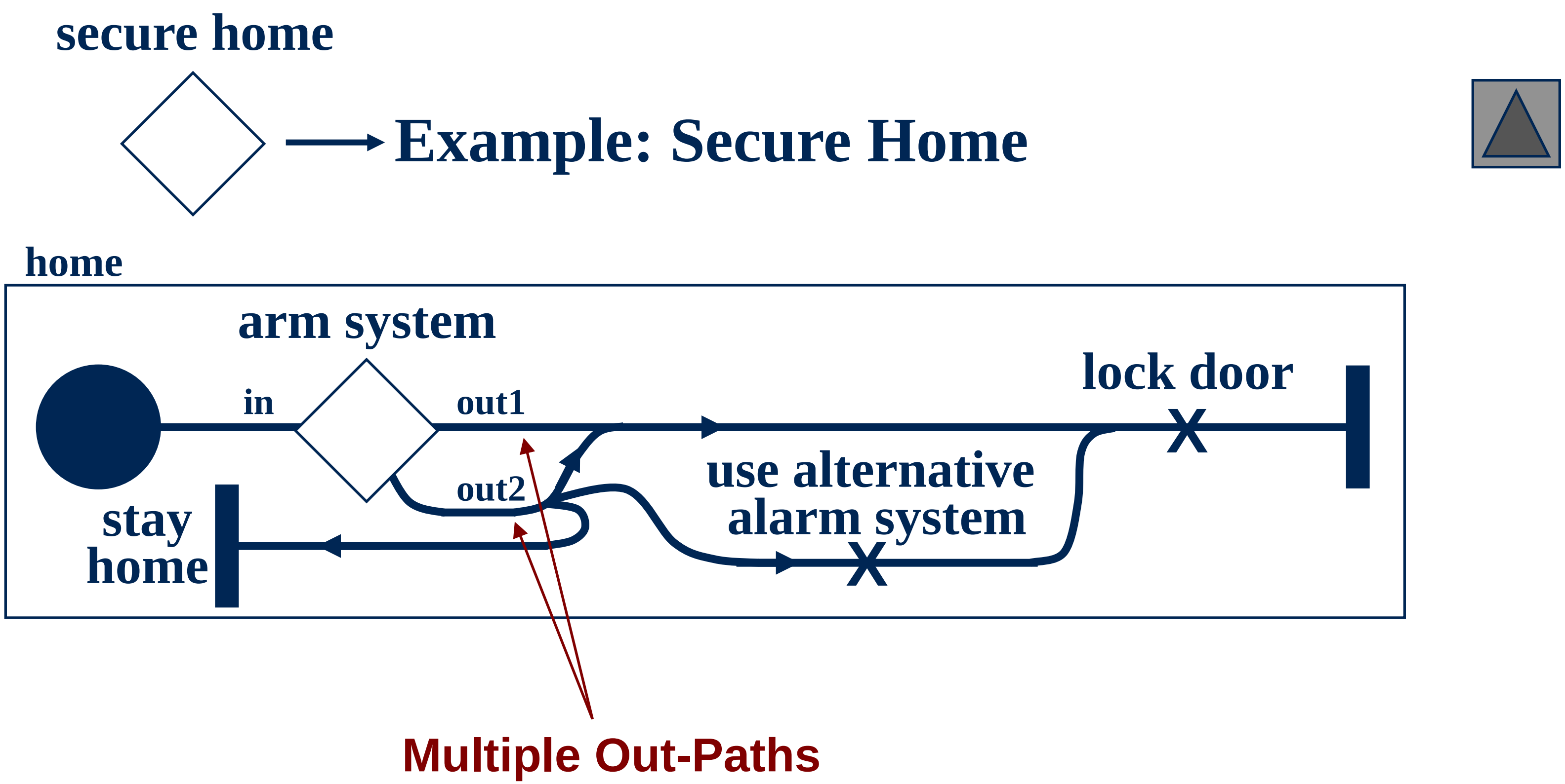
Alternatives and Concurrency (2/2)



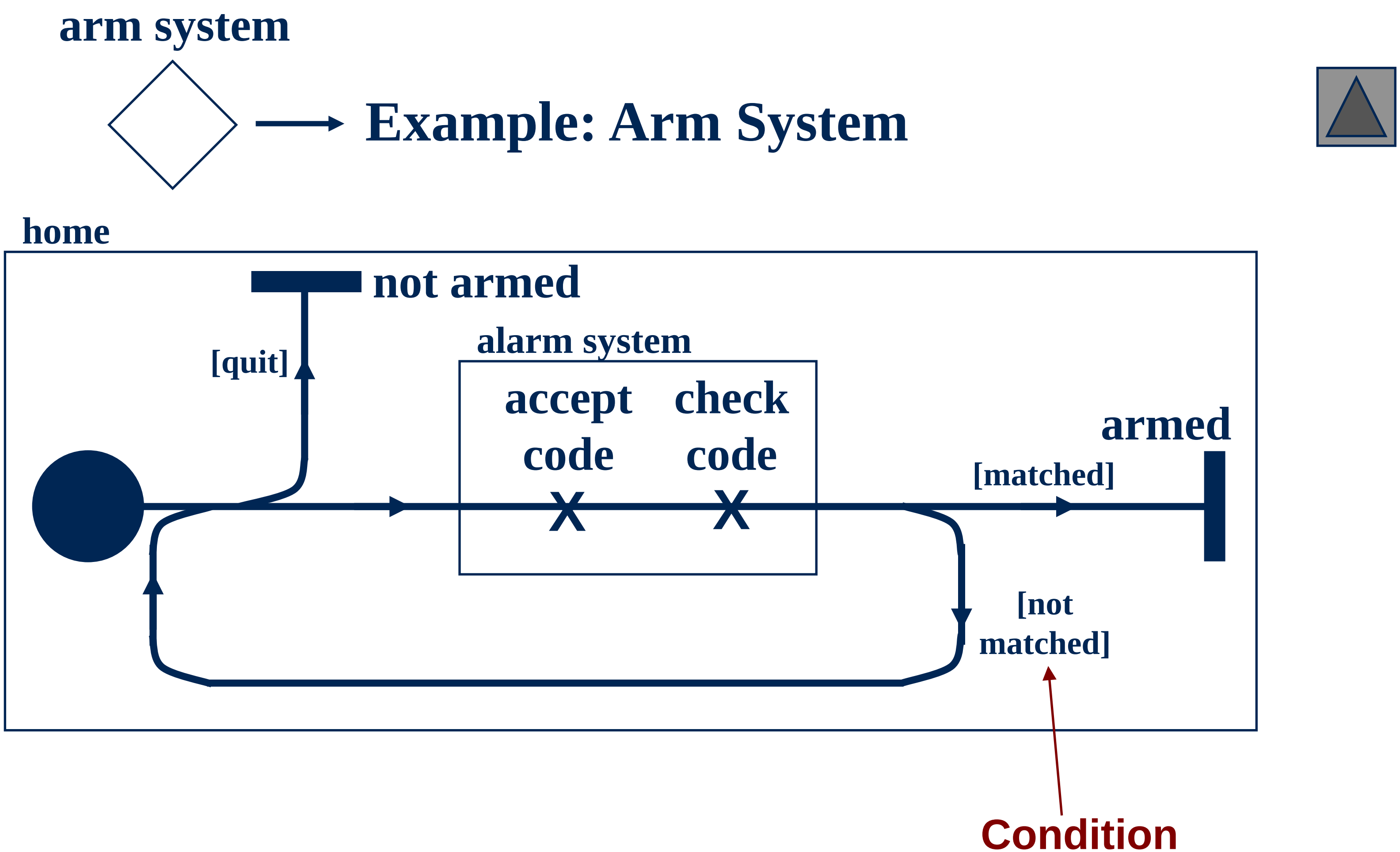
Waiting Place / Timer



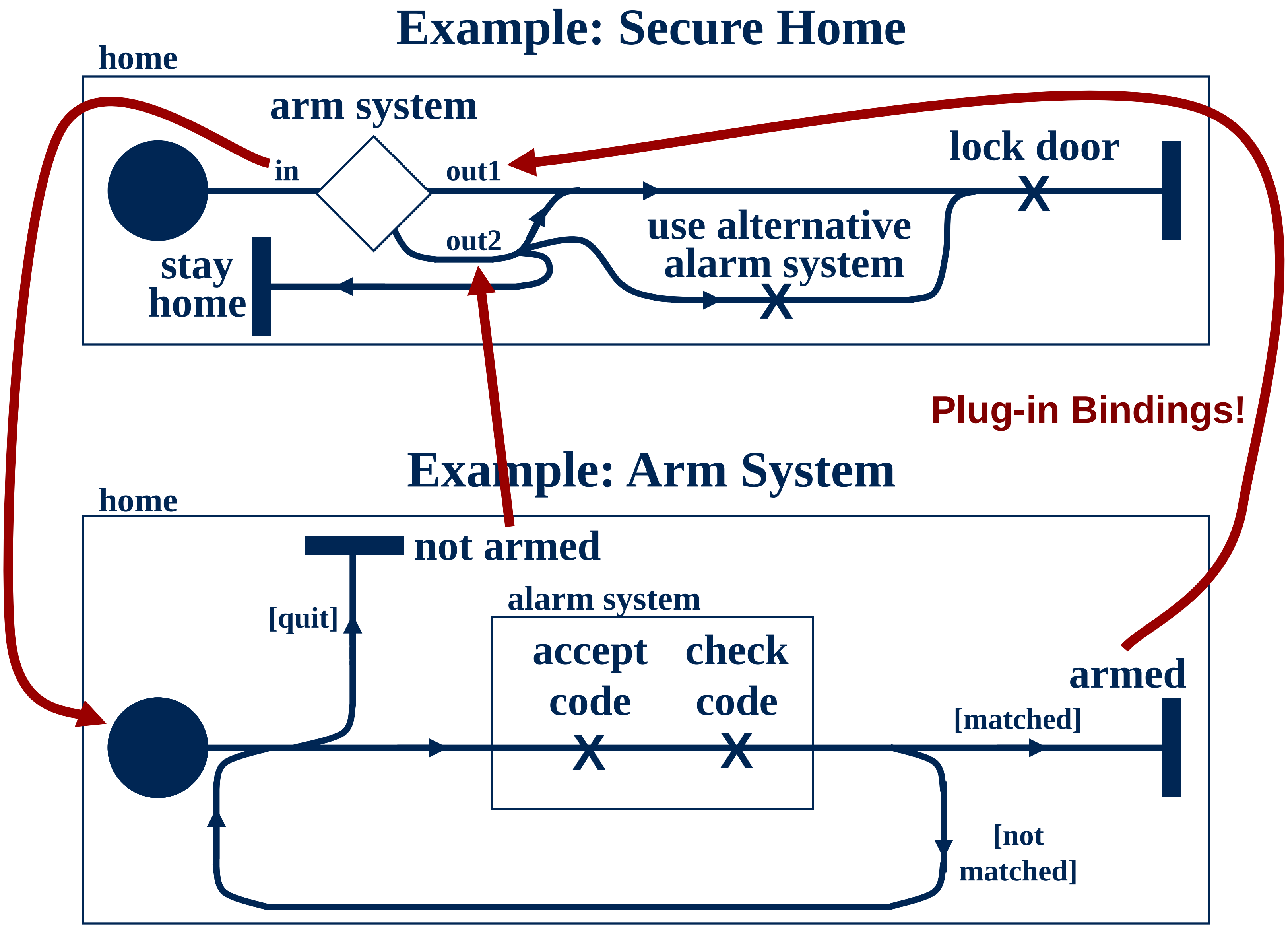
Multiple Results



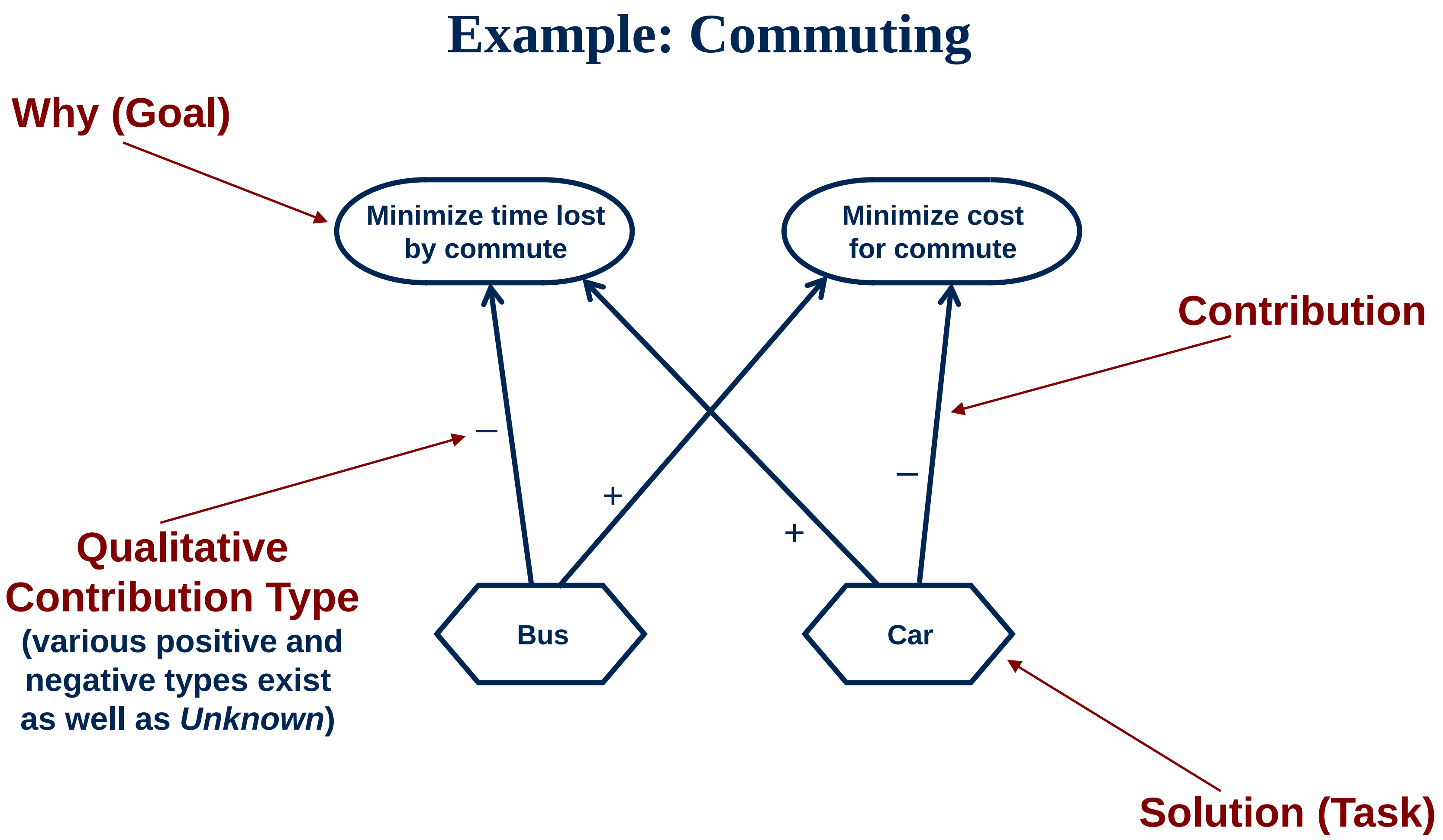
Condition


arm system
Example: Arm System
home
not armed
[quit]
alarm system
accept code
check code
armed
[matched]
[not matched]
Condition

Continuations between Hierarchical Levels

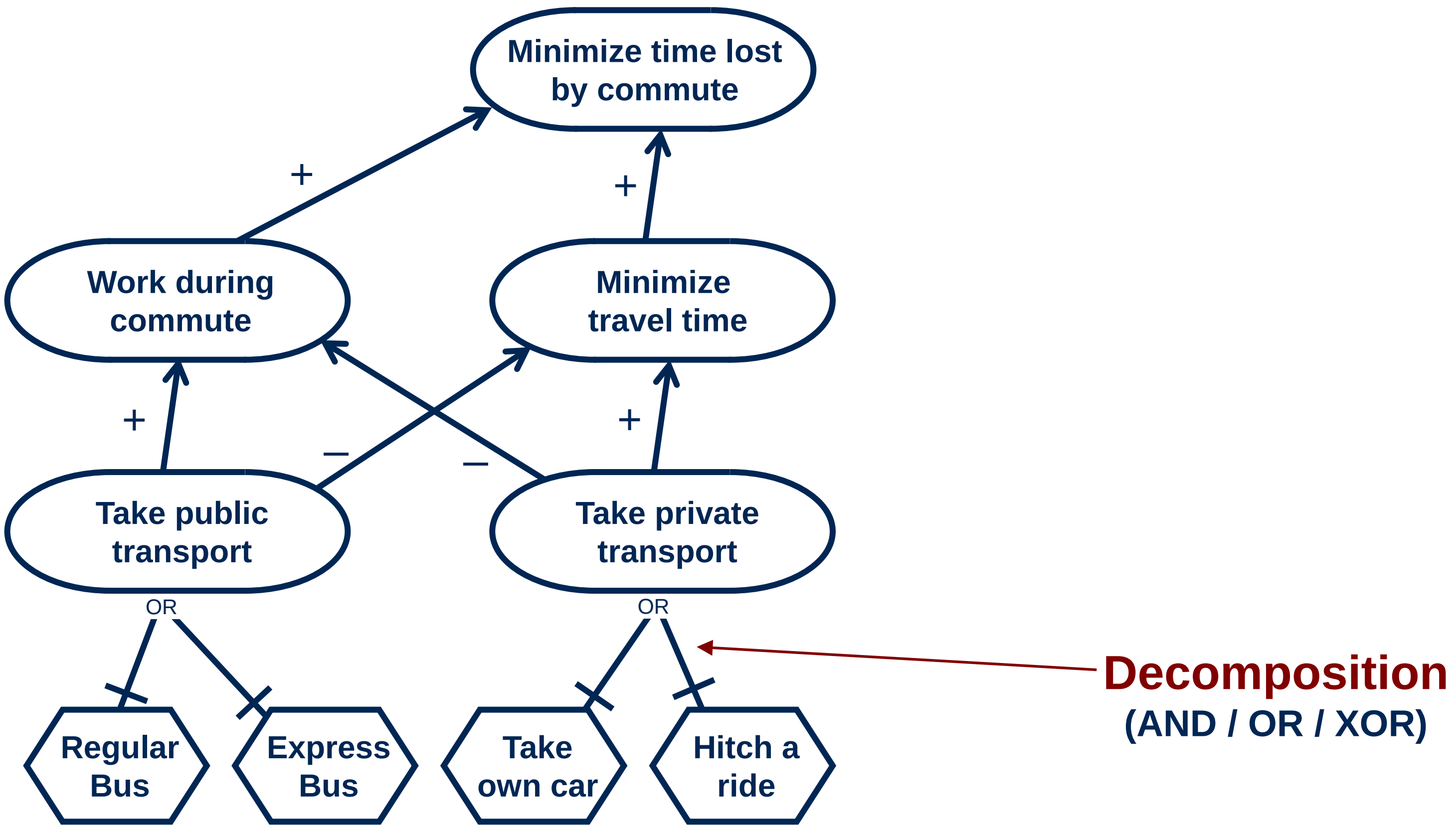


Basic Modeling of Reasons



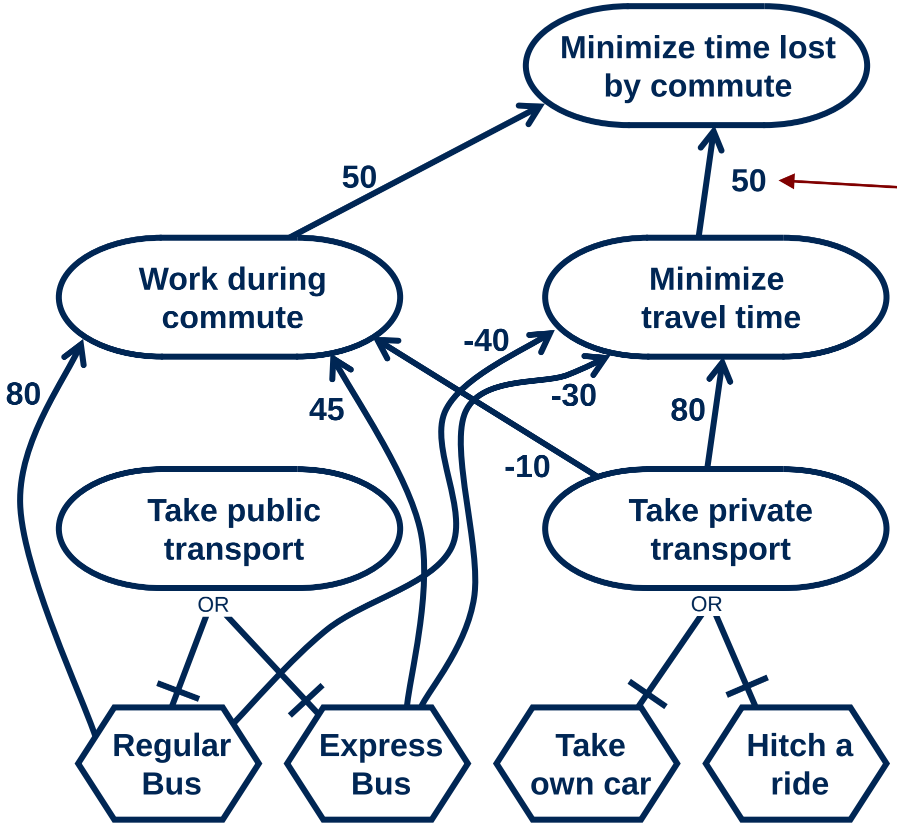
Decomposition

Example: Minimize time lost by commute (refinement 1)



Quantitative Contribution (1/2)

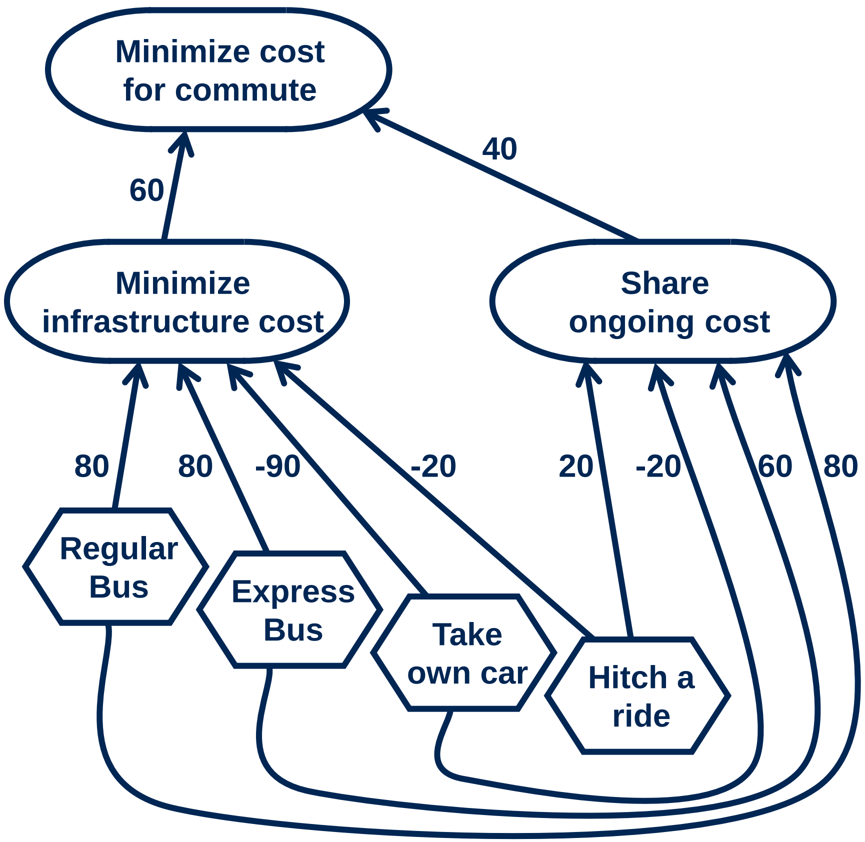
Example: Minimize time lost by commute (refinement 2)



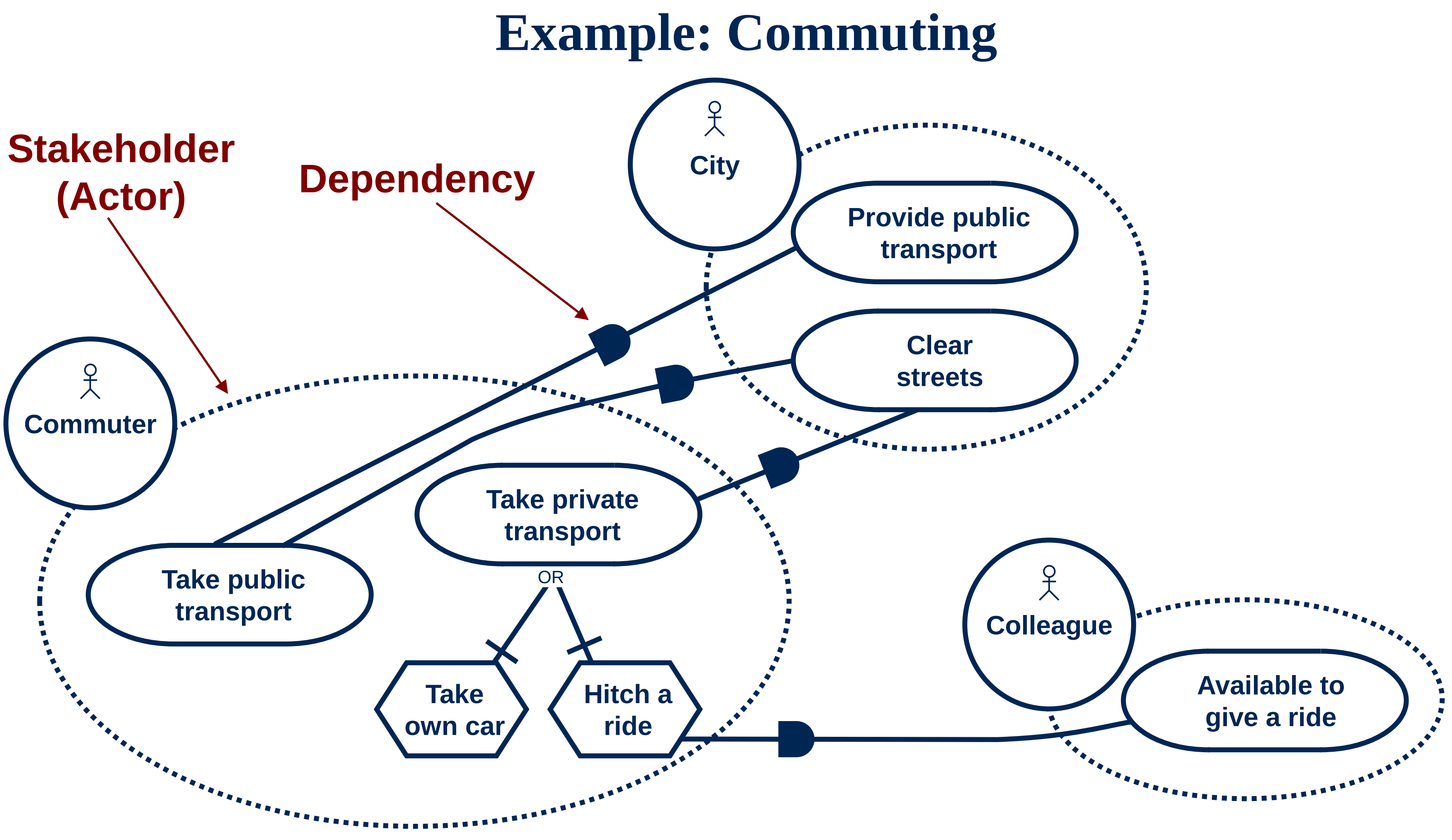
Quantitative Contribution Type
[-100, 100]

Quantitative Contribution (2/2)

Example: Minimize cost for commute (refinement 1)

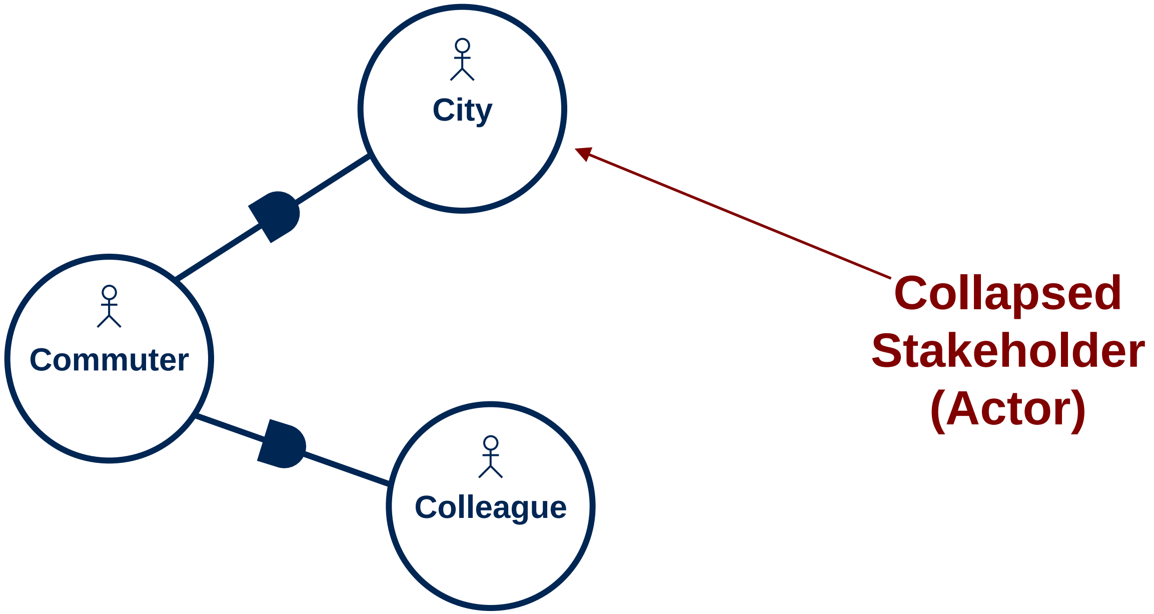


Stakeholders and Dependencies (1/2)



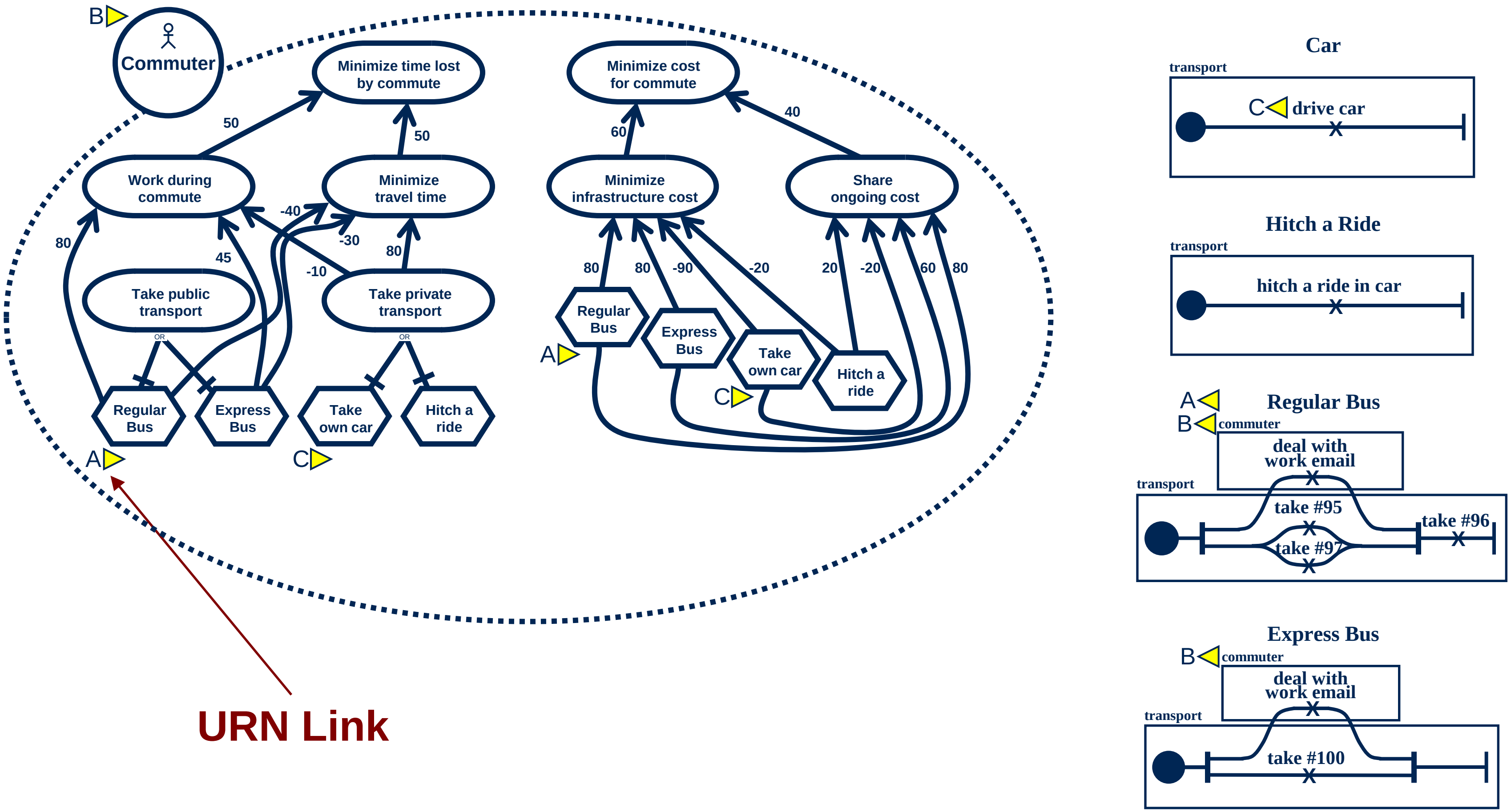
Stakeholders and Dependencies (2/2)

Example: Commuting



Links

Example: Commuting



Overview of the User Requirements Notation (URN)

User Requirements Notation: History (1/2)

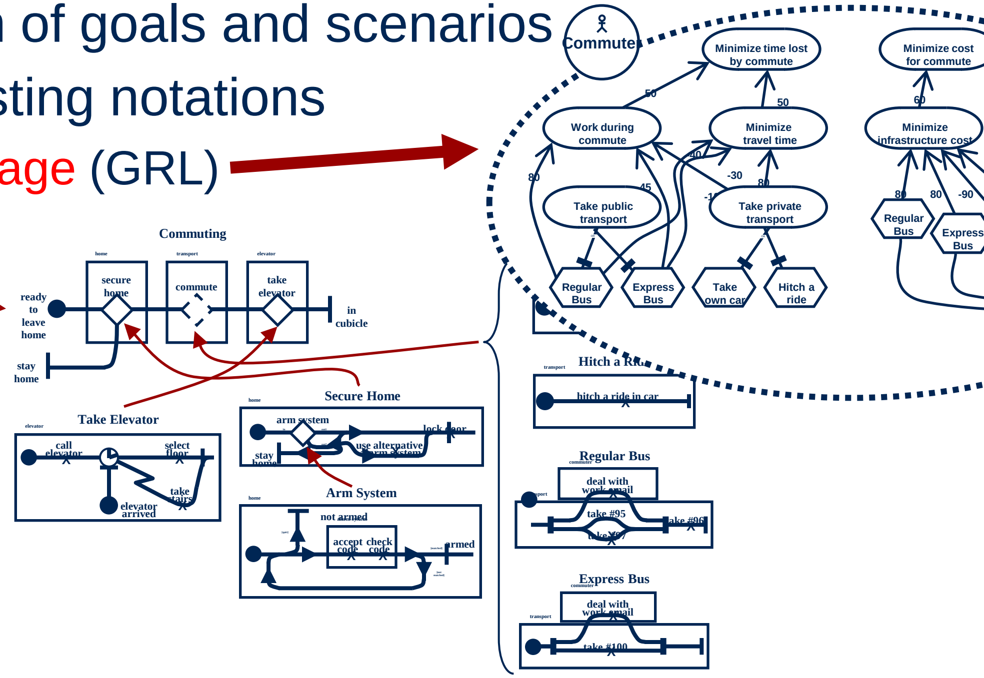
- In the 1990's
  - Work on program slices, timethreads, and Use Case Maps
    - Buhr, Woodside, Vigder, Casselman, Amyot... (Carleton University/University of Ottawa)
  - Work on the Non-Functional Requirements (NFR) Framework
    - Chung, Mylopoulos, Nixon, Yu... (University of Toronto)
  - Work on the i* Framework
    - Yu, Mylopoulos... (University of Toronto)
  - Industrial research projects and standardization projects
    - Visser, Hodges, Monkewich... (Nortel)
    - Gray, Pinard, Mankovski... (Mitel)
- 1999
  - Nortel proposes to standardize Use Case Maps at ITU-T
  - Launching of UseCaseMaps.org
- 2000–2002
  - Mitel gets involved and suggests adding i* to UCMs
  - Canadian proposal for the User Requirements Notation

User Requirements Notation: History (2/2)

- 2000–2002 (cont'd)
  - Integration of subsets of i* and NFR into GRL
    - Yu and Liu (University of Toronto)
  - ITU-T Rapporteurs
    - Hodges (2000), Cameron (2001), Amyot (2002-2008), Reed (2008-)
- 2003: ITU-T Recommendation Z.150 (02/03)
  - User Requirements Notation (URN) – Language requirements and framework
    - Editor: D. Amyot
- 2005
  - First release of jUCMNav URN tool
  - Launching of the Wiki for the URN Virtual Library (www.usecasemaps.org/pub)
- 2008: ITU-T Recommendation Z.151 (11/08)
  - User Requirements Notation (URN) – Language definition
    - Co-editors: D. Amyot, G. Mussbacher
- 2012: ITU-T Corrigendum to Z.151 and Z.151 v2
    - Co-editors: D. Amyot, G. Mussbacher

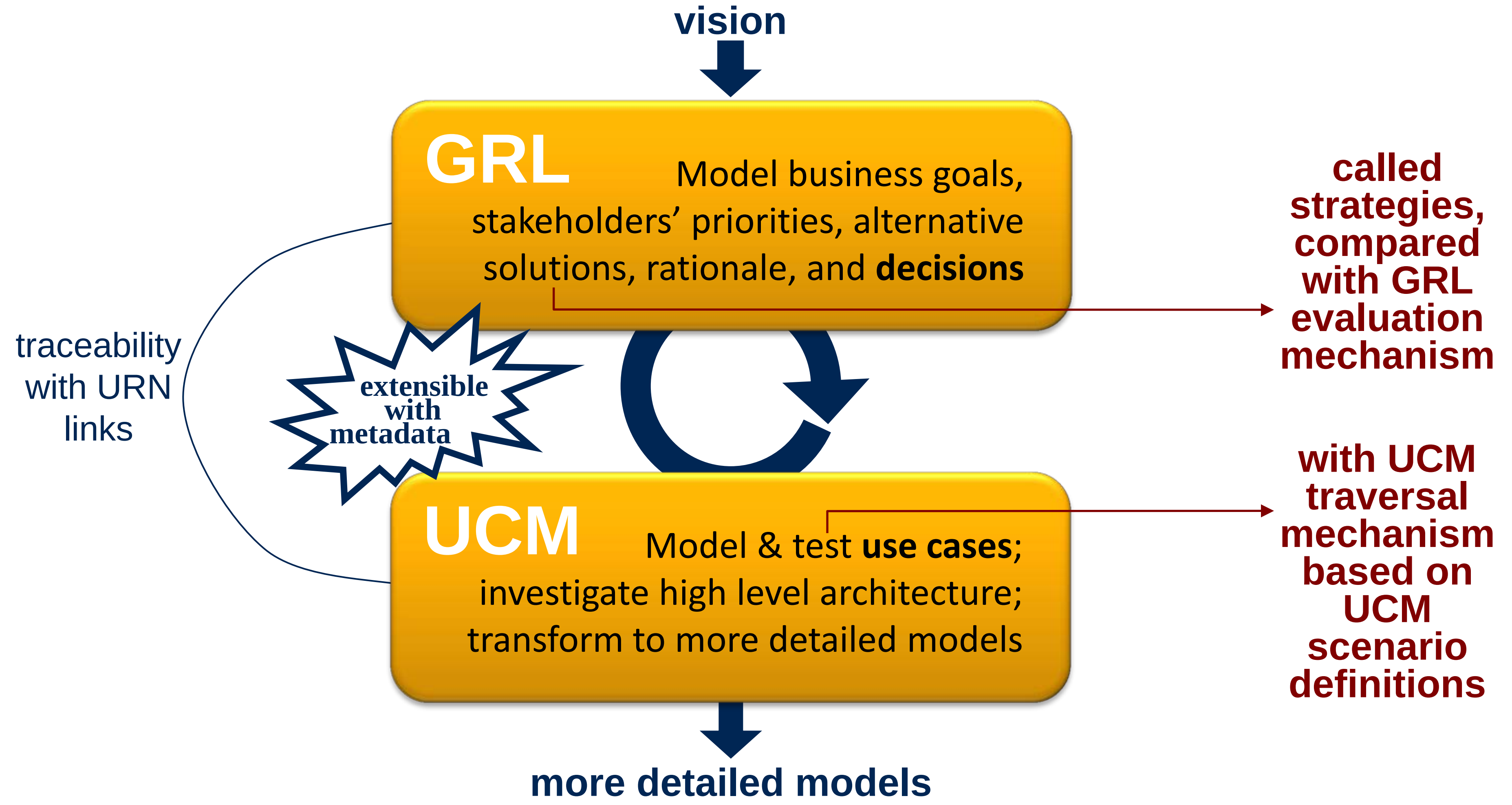
User Requirements Notation: Overview (1/2)

- URN is a semi-formal, lightweight graphical language for modeling and analyzing requirements in the form of goals and scenarios
- Formalizes and integrates two existing notations
  - Goal-oriented Requirement Language (GRL)
  - Use Case Maps (UCMs)
- Support for the elicitation, analysis, specification, and validation of requirements



- Allows systems/software/requirements engineers to discover and specify requirements for a proposed or an evolving system, and analyse such requirements for correctness and completeness
- URN models can be used to specify and analyze various types of reactive systems, business processes and goals of organizations, and telecommunications standards

User Requirements Notation: Overview (2/2)



- A GRL / UCM model visually communicates business objectives and constraints / high-level functional requirements to all stakeholders

URN Tool: jUCMNav – Juice Up Your Modeling!

- URN editor & analysis tool,
  Eclipse plugin,
  open source project
- GRL,
  Strategies, Evaluation
- UCMs,
  Scenario Definition &
  Execution, Test Suite
- Support for AoURN / CORE
- Support for BPM
- MSC Generation
- Export to DOORS / CSM

Pronounced: juicy – em – nav

http://jucmnav.softwareengineering.ca

ITU-T Z.151: URN – Language Definition

- URN is the first and currently only standard which explicitly addresses goals (non-functional requirements with GRL) in addition to scenarios (functional requirements with UCMs) in a graphical way in one unified language
  - International Telecommunication Union (ITU-T Z.150 series)
  - ITU-T Z.150 (02/03):
    User Requirements Notation (URN) - Language requirements and framework
  - ITU-T Z.151 (11/08):
    User requirements notation (URN) - Language definition
- Part of the ITU family of languages: SDL, MSC, TTCN-3, ASN.1…
- Definition of URN in Recommendation Z.151 (approved November 2008)
  - Modeling elements of notation and their meaning, analysis capabilities, interchange format

ITU-T Z.150: http://www.itu.int/rec/T-REC-Z.150/en ITU-T Z.151: http://www.itu.int/rec/T-REC-Z.151/en

Why Use Case Maps?

- Bridge the modeling gap between abstract and informal descriptions useful at early stages of system development and more formal and concrete descriptions useful at later stages of system development
- Use Case Maps integrate many scenarios and allow reasoning about potential undesirable interactions
- Provide ability to model dynamic systems where scenarios and structures may change at run-time
  - E-business applications, Web services, business processes and workflows
  - Distributed systems based on agents, reactive systems
- Effective learning tool for people unfamiliar with the domain
- May be transformed (e.g., into MSC/sequence diagrams, performance models, test cases)

Use Case Maps: Summary

- Model scenario concepts
  - Mainly for operational requirements, functional requirements, and business processes
  - For reasoning about scenario interactions, performance, and architecture

- Use Case Maps provide …
  - Visual description of behavior superimposed over entities (from stakeholders and users to software architecture to hardware)
  - Easy graphical manipulation of scenario descriptions
  - Single scenario view
  - Combined system view
  - Enhanced consistency and completeness
  - Connections to goal models
  - Smooth transition to design models (e.g., message sequence charts)
  - Connections to performance models and testing models

Why the Goal-oriented Requirement Language?

- Goals become an important driver for requirements elaboration. Yet, stakeholders goals and objectives are complex and will conflict…
- GRL expresses and clarifies tentative, ill-defined, and ambiguous requirements
  - Supports argumentation, negotiation, conflict detection & resolution, and in general decisions
  - Captures decision rationale and criteria (documentation!)
- GRL identifies alternative requirements and alternative system boundaries
- GRL provides clear traceability from strategic objectives to technical requirements
- GRL allows reuse of stable higher-level goals when the system evolves

- **Nothing like this in UML or BPMN or in other standard languages…**

Goal-oriented Requirement Language: Summary

- The Goal-oriented Requirement Language is based on …
  - i* (concepts / syntax)
  - NFR Framework (evaluation mechanism)

- Model goals and other intentional concepts – mainly for non-functional requirements, quality attributes, rationale documentation, and reasoning about alternatives and tradeoffs

- The Goal-oriented Requirement Language is used to …
  - Visually describe business goals, stakeholders' priorities, alternative solutions, rationale, and decisions
  - Decompose high-level goals into alternative solutions called tasks (this process is called operationalization)
  - Model positive and negative influences of goals and tasks on each other
  - Capture dependencies between actors (i.e., stakeholders)

Modeling Requirements with URN

- For requirements modeling, we need to answer the W5 questions
  - *Where, What, Who, When,* and *Why*
- Goal-oriented Requirement Language (GRL)
  - Business or system goals and rationales (**Why**)
  - Solutions/Tasks (What)
  - Stakeholders/Actors (Who and Where)
- Use Case Maps (UCMs)
  - Responsibilities (What)
  - Components (Who and Where)
  - Scenarios and causal sequences (**When**)
- GRL & UCMs
  - Link processes to business goals with URN links (▷), for traceability, completeness, alignment, compliance, what if scenarios, and evolution

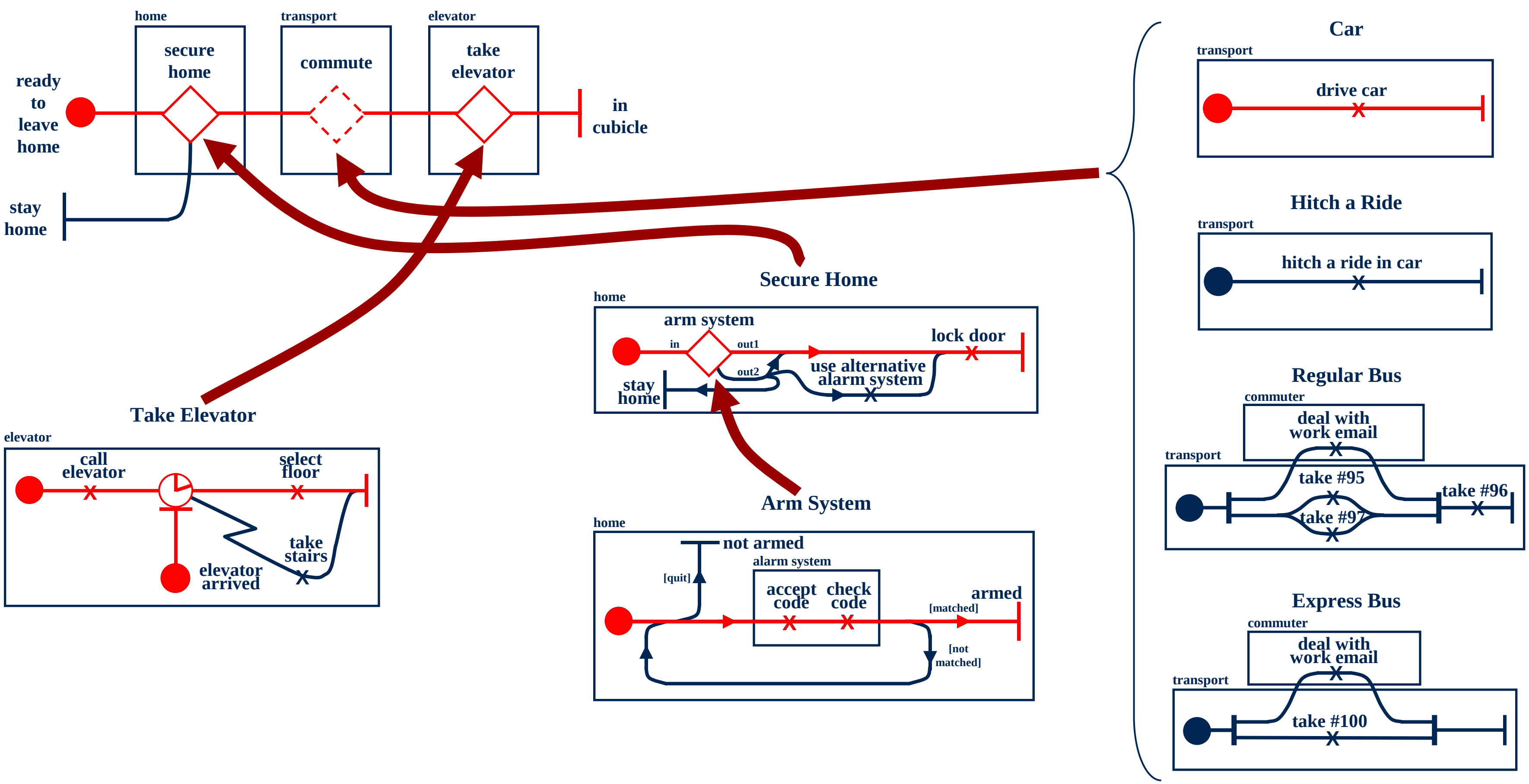
Analysis of the Simple Problem

Scenario Execution (1/3)

Scenario Definition "my own car, home armed, elevator":

ready to leave home; matched; Car; elevator arrived → armed; in cubicle

Example: Commuting

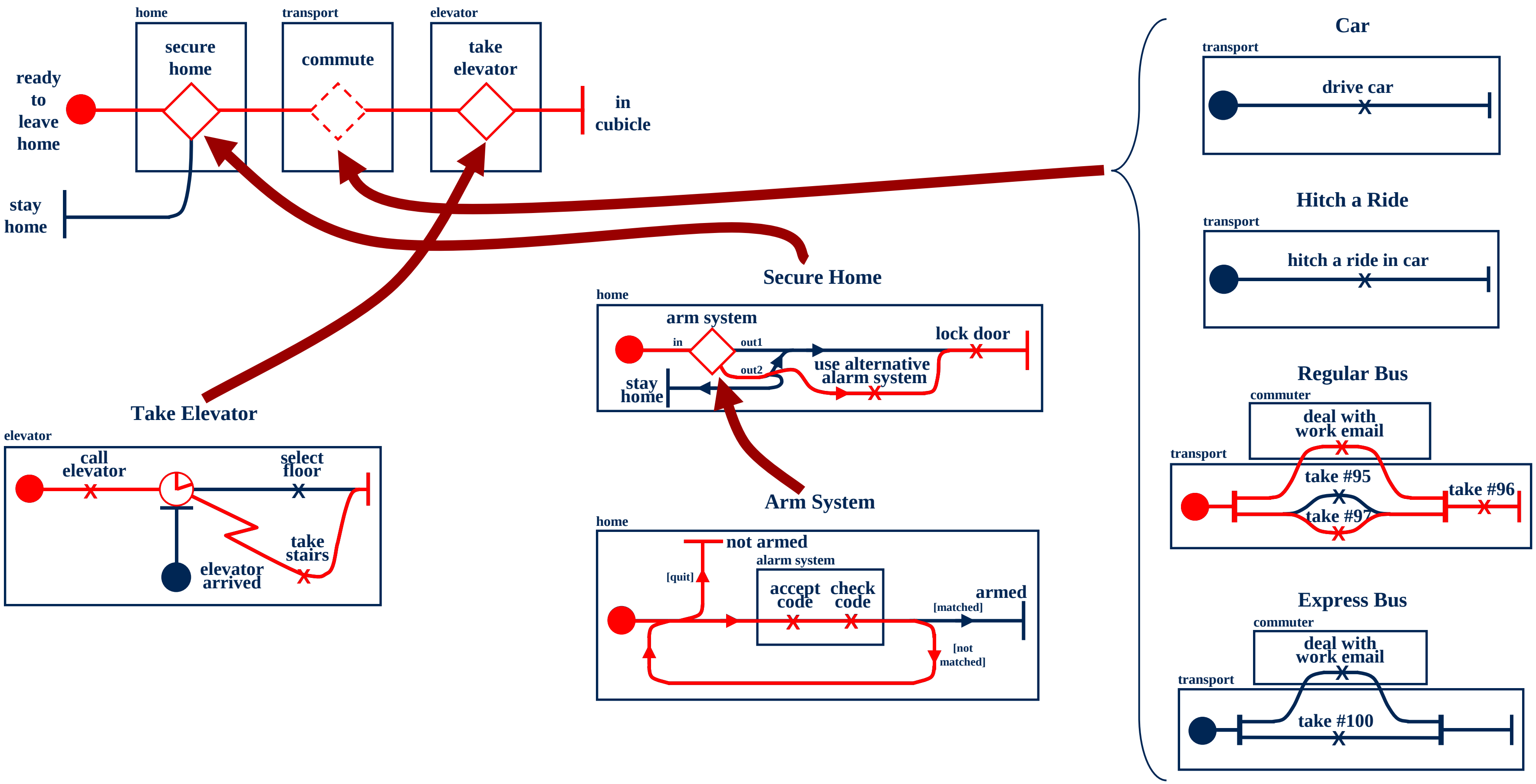


Scenario Execution (2/3)

Scenario Definition "regular bus #97, alternative alarm, stairs":

ready to leave home; not matched; quit; alternative alarm; Regular Bus; #97; elevator does not arrive → not armed; in cubicle

Example: Commuting

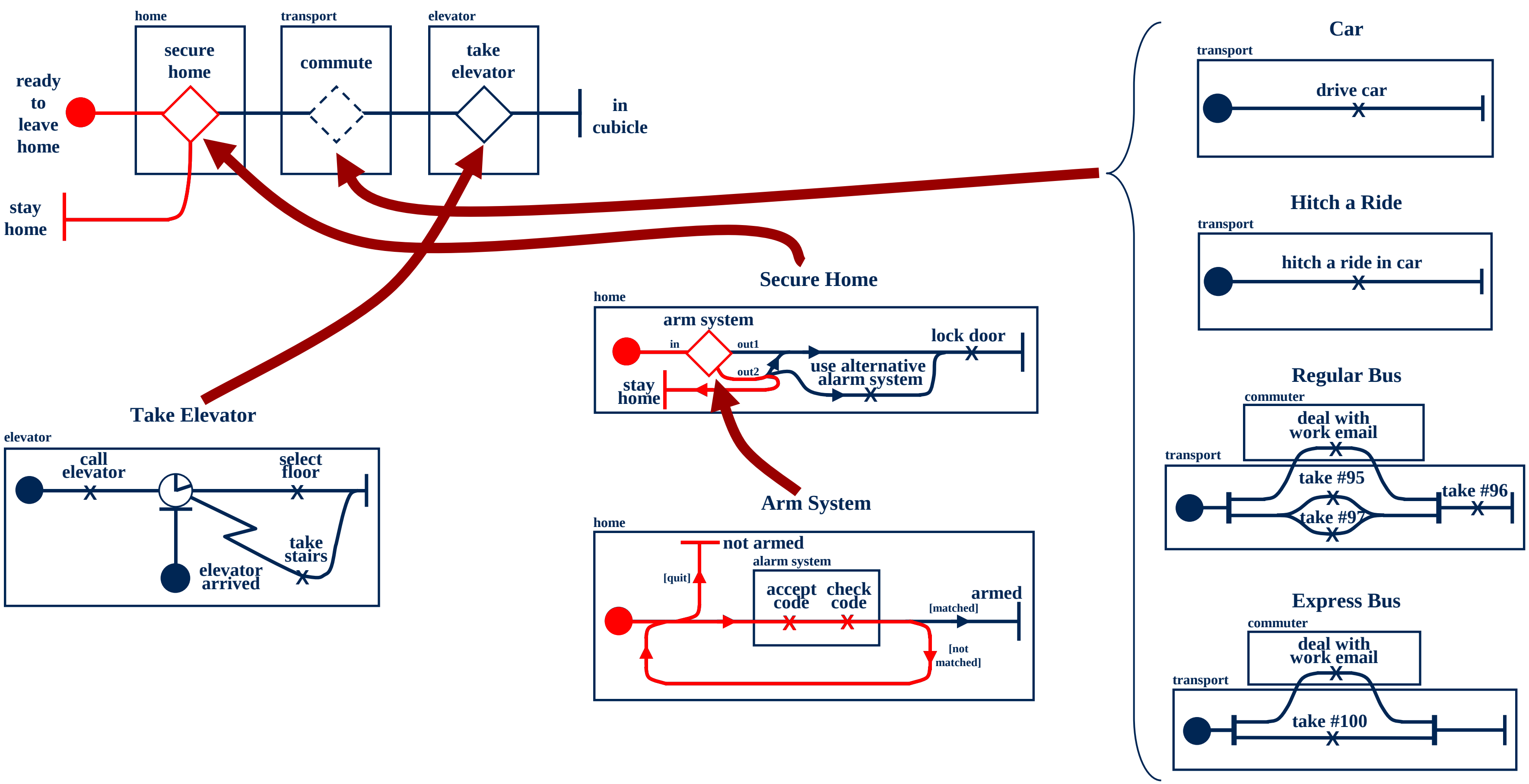


Scenario Execution (3/3)

Scenario Definition “stay home”:

ready to leave home; not matched; quit; stay home → not armed; stay home

Example: Commuting



Strategy Execution (1/7)

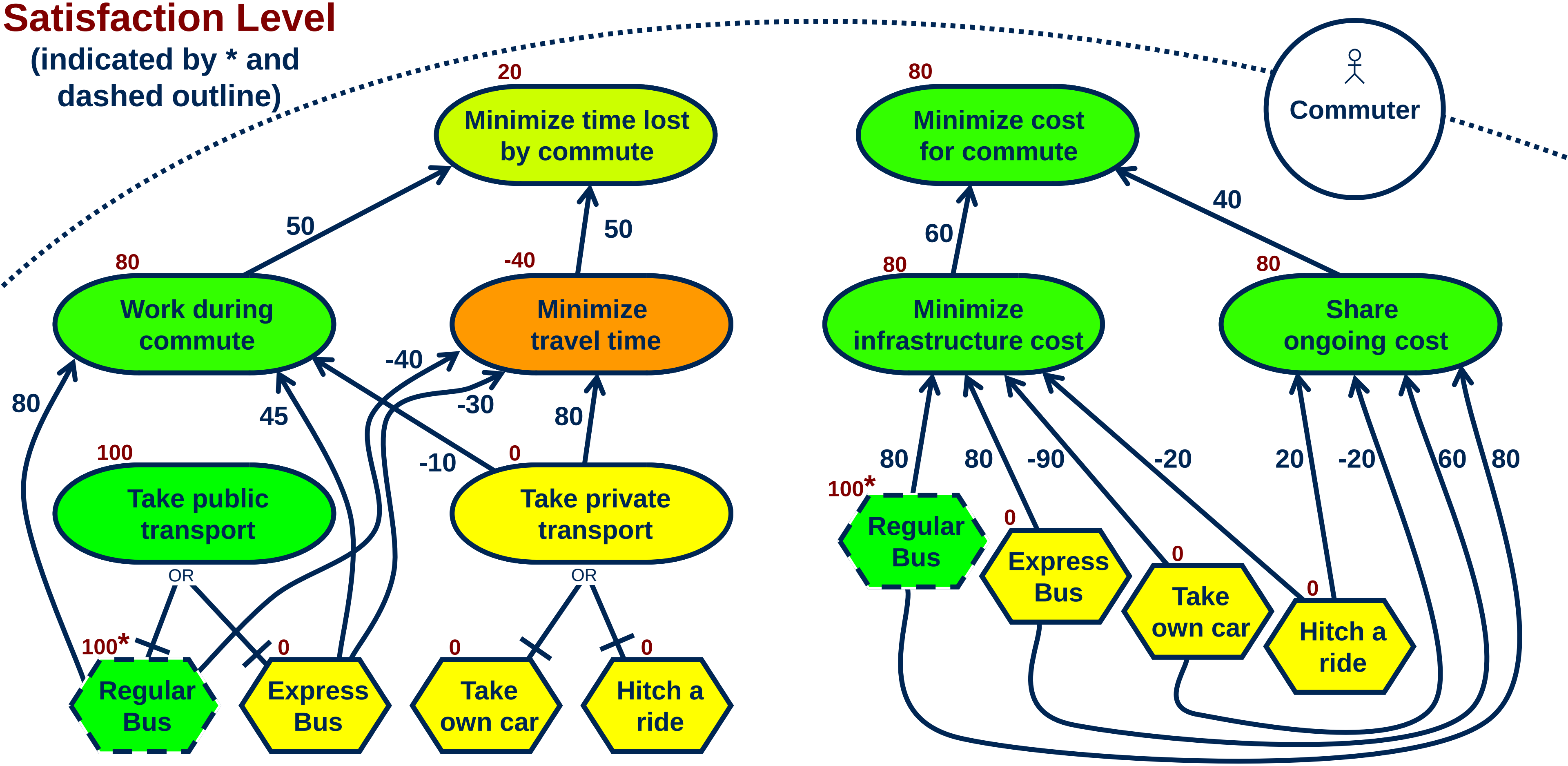
Strategy "Regular Bus":

Regular Bus = 100

Initial Satisfaction Level

(indicated by * and dashed outline)

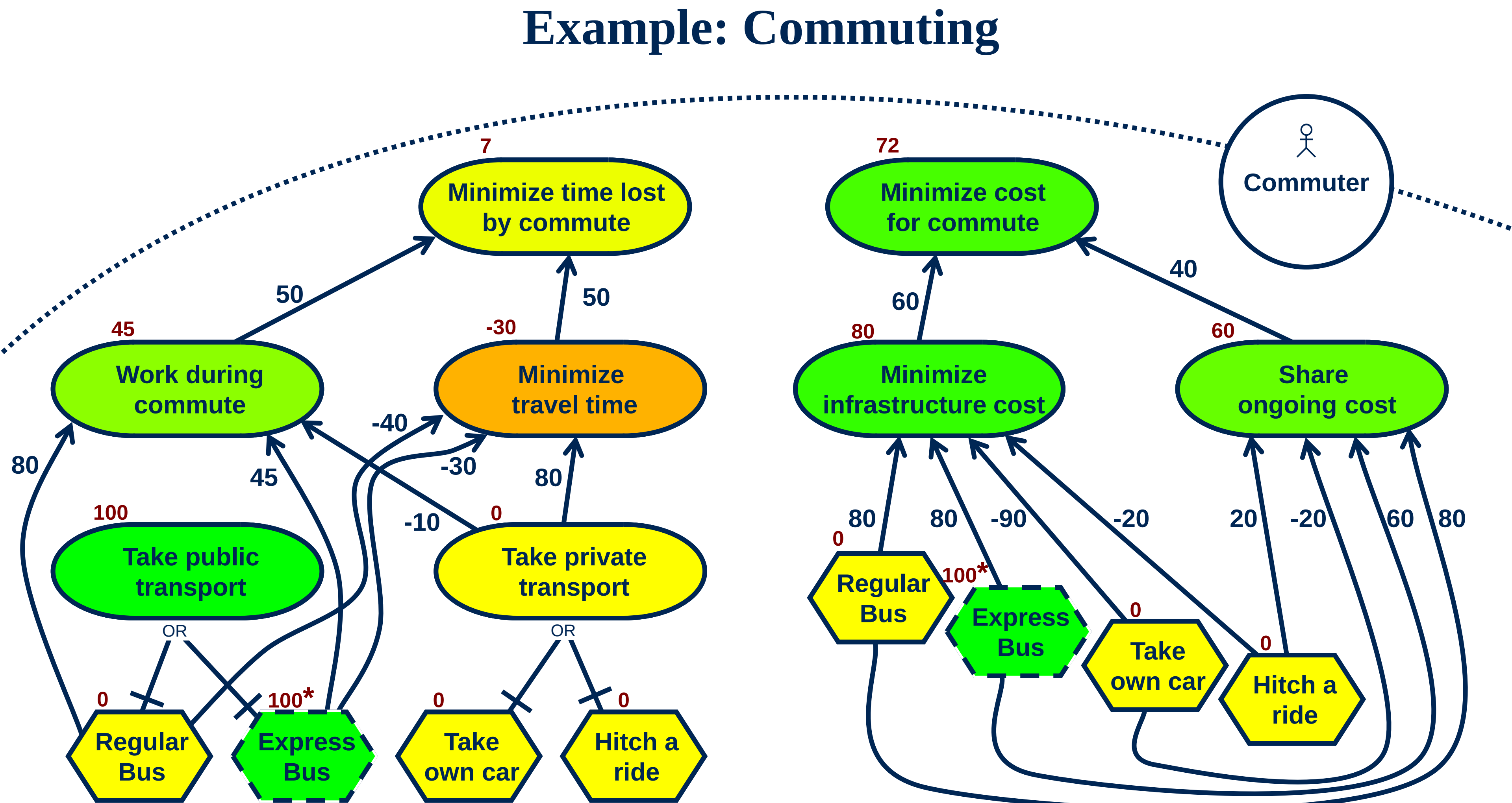
Example: Commuting



Strategy Execution (2/7)

Strategy "Express Bus":

Express Bus = 100

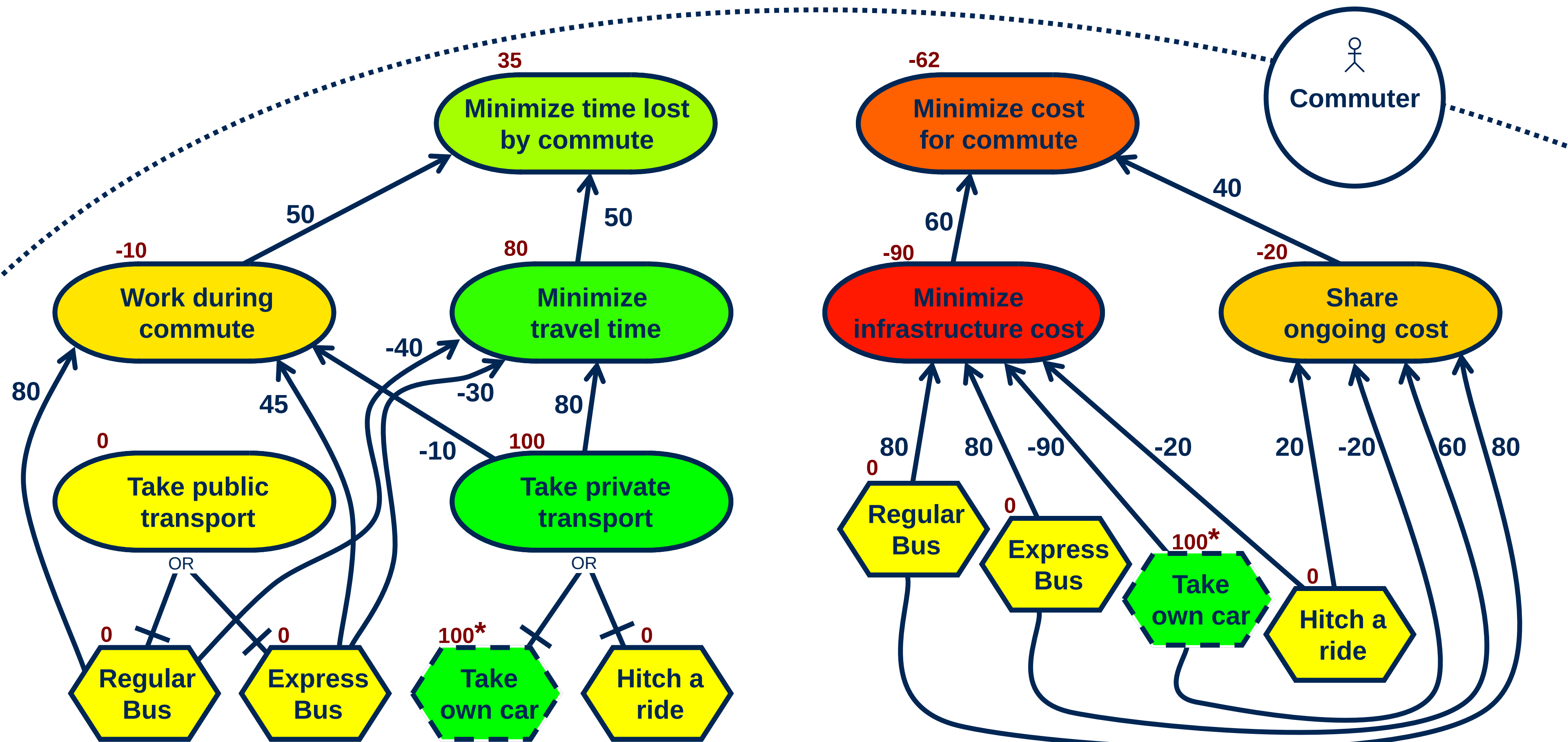


Strategy Execution (3/7)

Strategy "Take own car":

Take own car = **100**

Example: Commuting

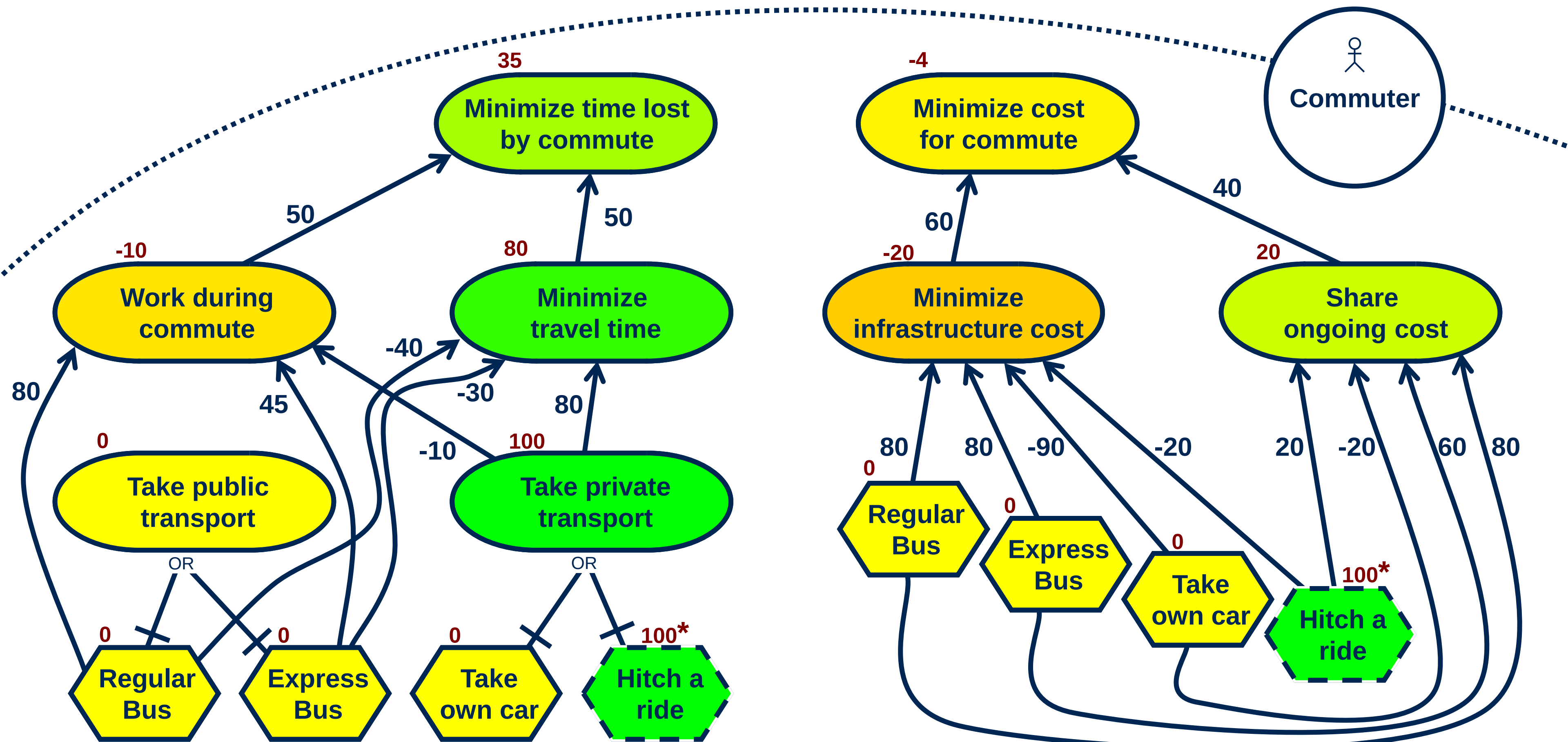


Strategy Execution (4/7)

Strategy "Hitch a ride":

Hitch a ride = 100

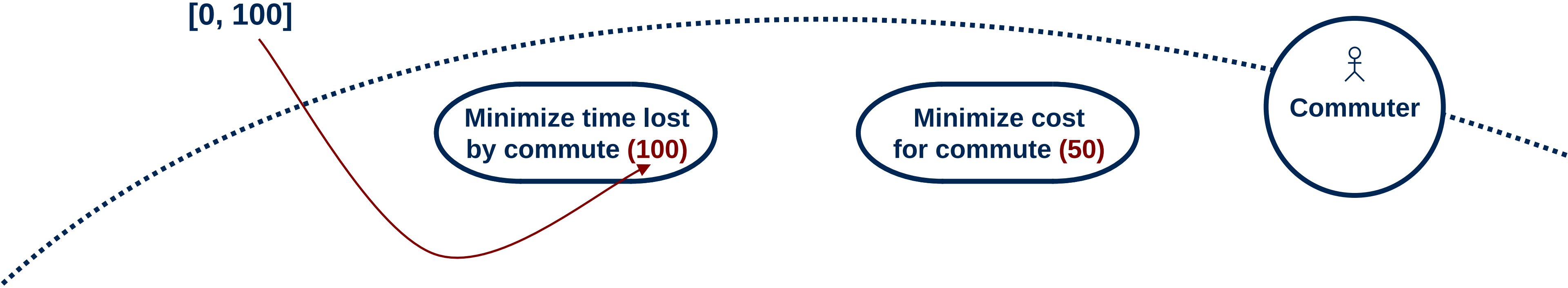
Example: Commuting



Strategy Execution (5/7)

Importance
[high, medium, low, none]
[0, 100]

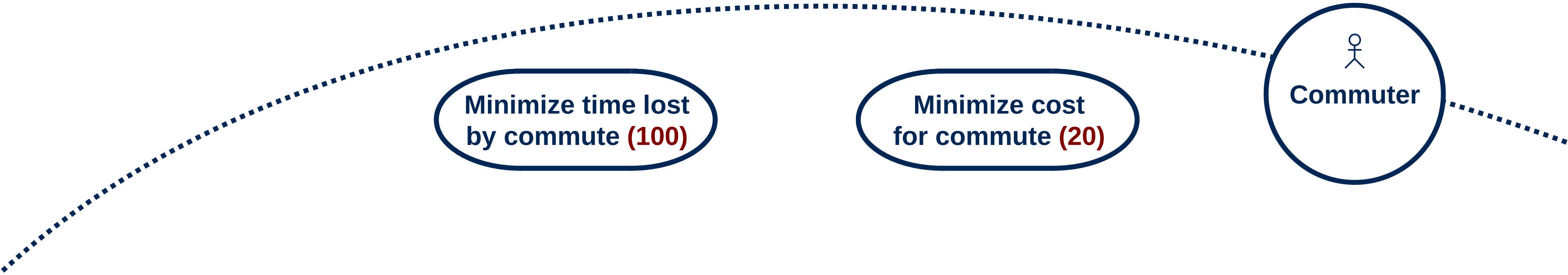
Example: Commuting



Strategy “Regular Bus”:	20	80	Commuter 40	Rank 1
Strategy “Express Bus”:	7	72	Commuter 28	Rank 2
Strategy “Take own car”:	35	-62	Commuter 2	Rank 4
Strategy “Hitch a ride”:	35	-4	Commuter 22	Rank 3

Strategy Execution (6/7)

Example: Commuting



Strategy "Regular Bus":	20	80	Commuter 30	Rank 1
Strategy "Express Bus":	7	72	Commuter 17	Rank 4
Strategy "Take own car":	35	-62	Commuter 18	Rank 3
Strategy "Hitch a ride":	35	-4	Commuter 28	Rank 2

Strategy Execution (7/7)

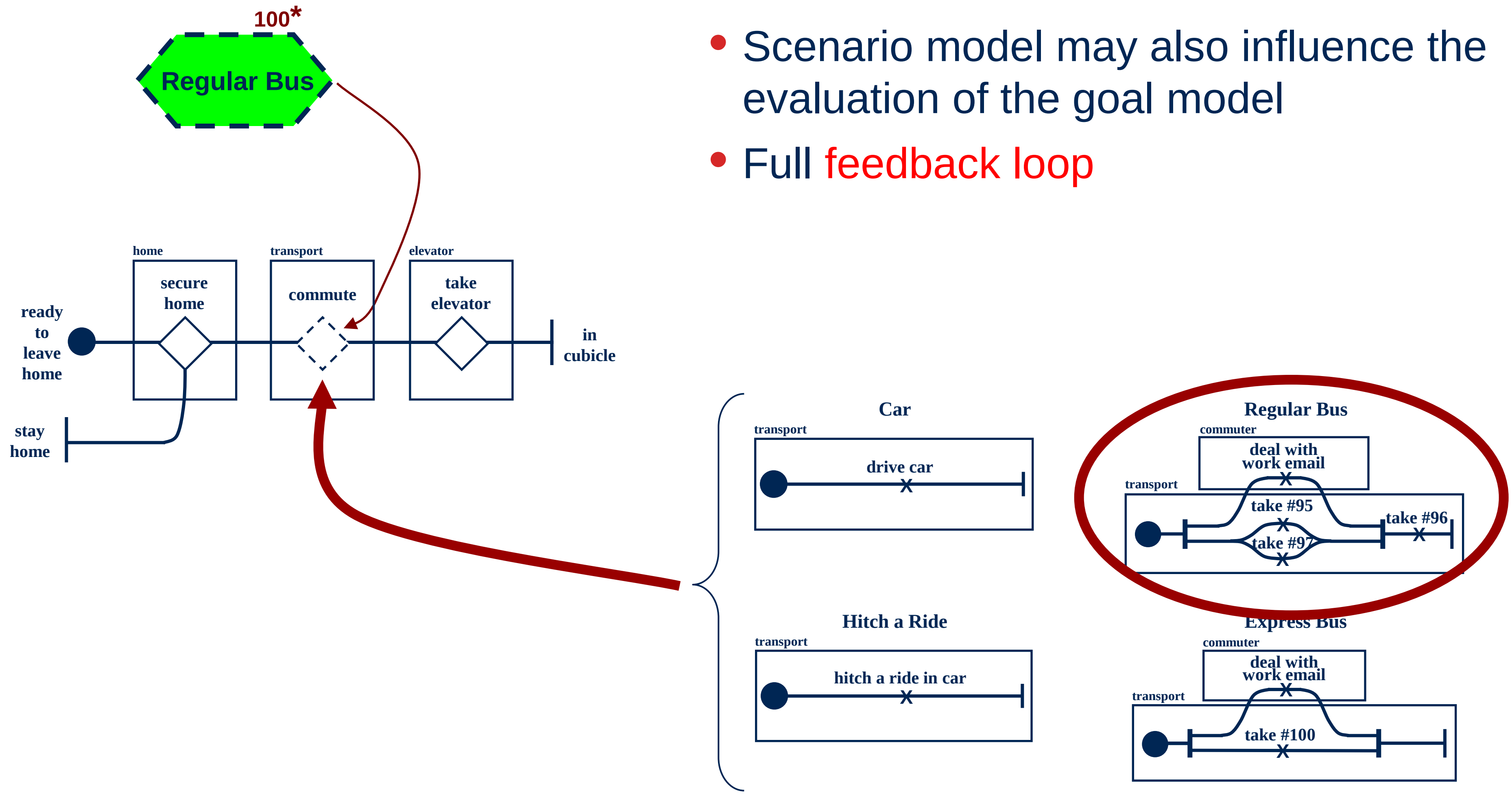
Example: Commuting

Strategy “Regular Bus”:	20	80	Commuter 71	Rank 1
Strategy “Express Bus”:	7	72	Commuter 62	Rank 2
Strategy “Take own car”:	35	-62	Commuter -47	Rank 4
Strategy “Hitch a ride”:	35	-4	Commuter 1	Rank 3

Integration of Goal and Scenario Models

Example: Commuting

- Choices in the goal model (i.e., the chosen strategy) may influence the scenario model
- Scenario model may also influence the evaluation of the goal model
- Full feedback loop



Overview of Analysis with the User Requirements Notation (URN)

Scenario Traversal Mechanism

- A scenario describes one path through the model (only one alternative at any choice point is taken)
  - Set of initial values for the variables used in conditions and responsibilities
  - Start points triggered, end points reached
  - Possibly pre/post conditions
- A traversal mechanism interprets the model given scenario description(s)
  - Requires the use of the data model in choice points (forks, dynamic stubs, timers, conditions) and responsibilities
- Extraction of individual scenarios (highlight, transformations)
  - Learning tool – allows focus on key scenarios
- Groups of scenarios can be run together (i.e., a test suite for regression testing)

Strategies and Evaluation Mechanism (1/3)

- The goal model allows a particular configuration of intentional elements to be defined in a strategy (i.e., one possible solution)
  - Captures the initial, user-defined satisfaction levels for these elements
  - Strategies can be compared with each other for trade-off analyses
- In order to analyze the goal model and compare solutions with each other, a customizable evaluation mechanism executes the strategies
  - Propagating levels to the other elements and to stakeholders shows impact of proposed solution on high level goals for each stakeholder
  - Propagation starts at user-defined satisfaction levels of intentional elements (usually bottom-up)
  - Takes into consideration
    - Initial satisfaction levels of intentional elements
    - Links and contribution types
    - Importance defined for intentional elements
  - Qualitative or quantitative interpretation

Strategies and Evaluation Mechanism (2/3)

- Bottom-up analysis
  - Typically propagates satisfaction values of low-level tasks (i.e., selected solutions) to those of high-level stakeholder goals

- Top-down analysis
  - Searches for the optimal result taking the structure of the goal model and the relationships between nodes in the goal model into account
  - Can be formulated as a planning problem
  - Is akin to a constraint solving approach

Strategies and Evaluation Mechanism (3/3)

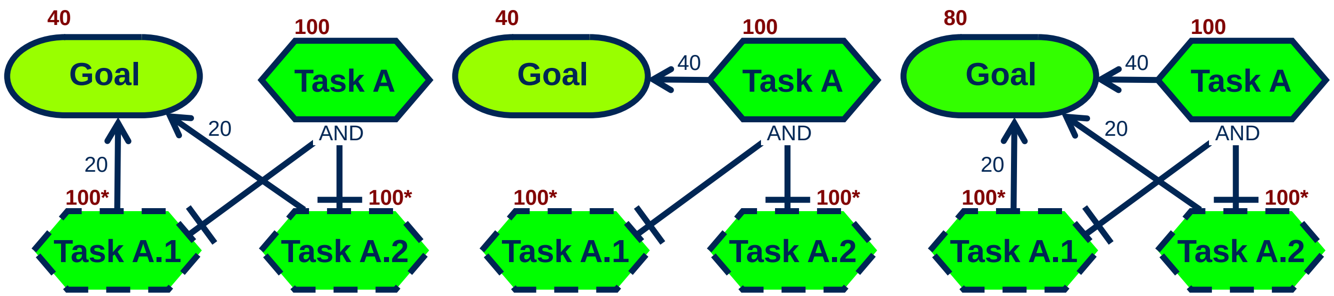
- Quantitative Approach
  - Contribution types: [-100, 100]
  - Importance: [0, 100]
  - Quantitative satisfaction levels: [-100, 100]
- Qualitative Approach
  - Contribution types: from Make to Break
  - Importance: High, Medium, Low, or None
  - Qualitative satisfaction levels
- Mixed (Hybrid) Approach also possible
  - Qualitative contribution types
  - Quantitative importance
  - Quantitative satisfaction levels

Satisfaction Levels:
(qualitative)

Strategy Execution – Anomalies

- Contributions can be shown for any element at any hierarchical level

- Using contributions at several levels has implications in terms of the evaluation mechanism
  - Not necessarily forbidden, but one has to be aware of the ramifications

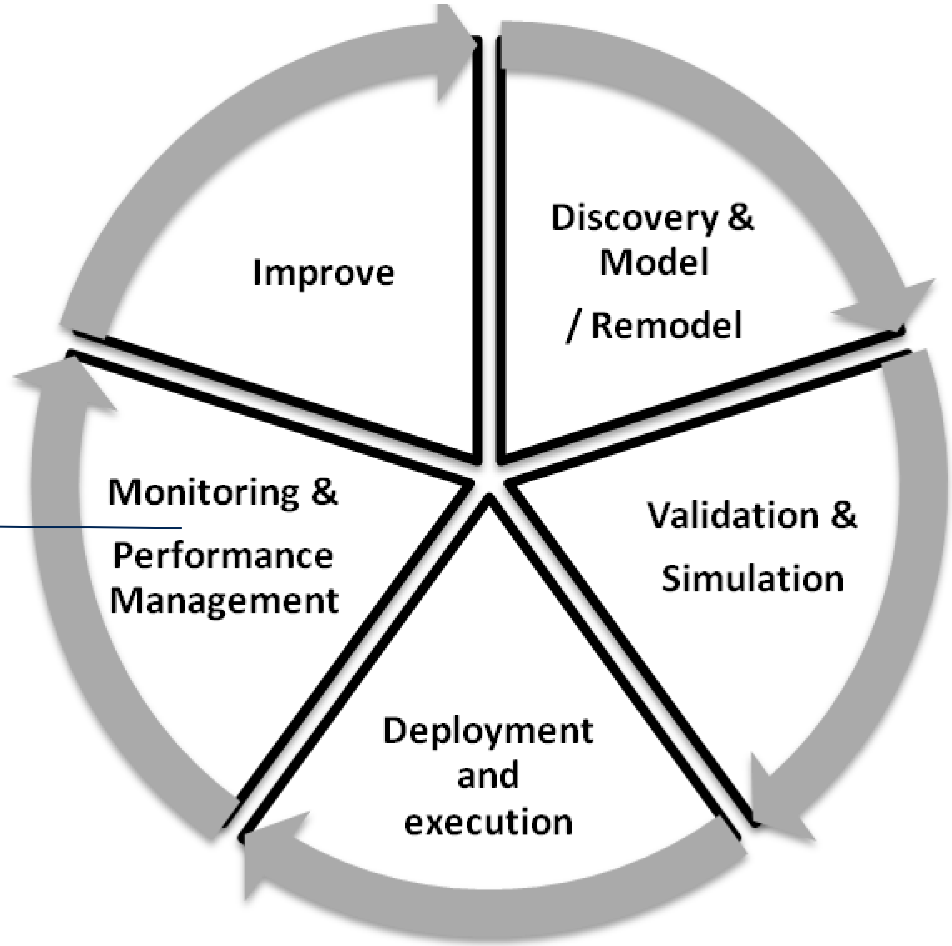


- May be annoying while the goal model is being built
  - Start with high-level contributions
  - Refine → May need to add contributions at lower level and therefore may have to remove contributions at the higher level
  - Or vice versa

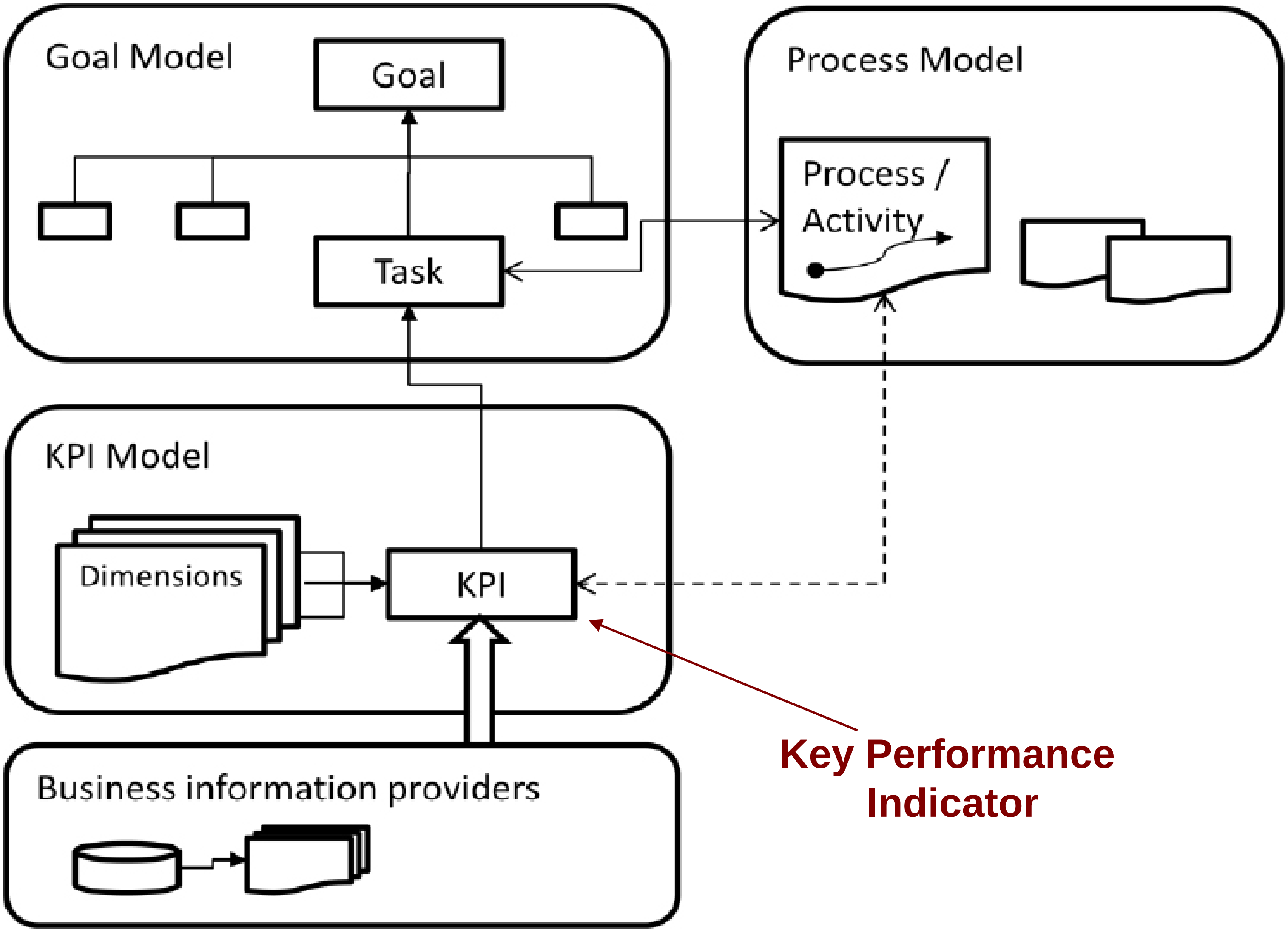
Key Performance Indicators (KPIs)

Business Process Management

Goals are not enough... Need a way to measure functional and non-functional properties in terms of the domain units.

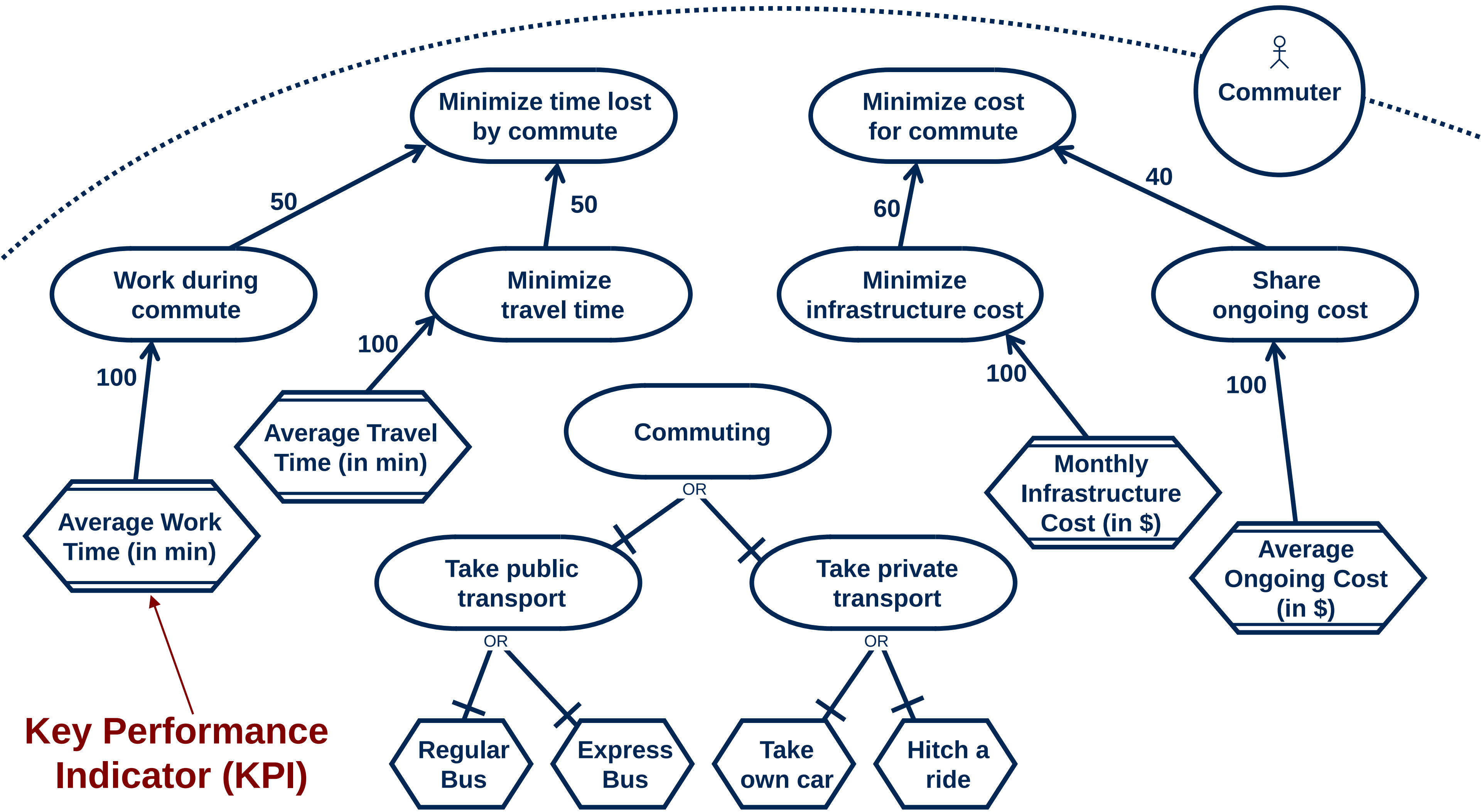


Evaluations Involving Key Performance Indicators



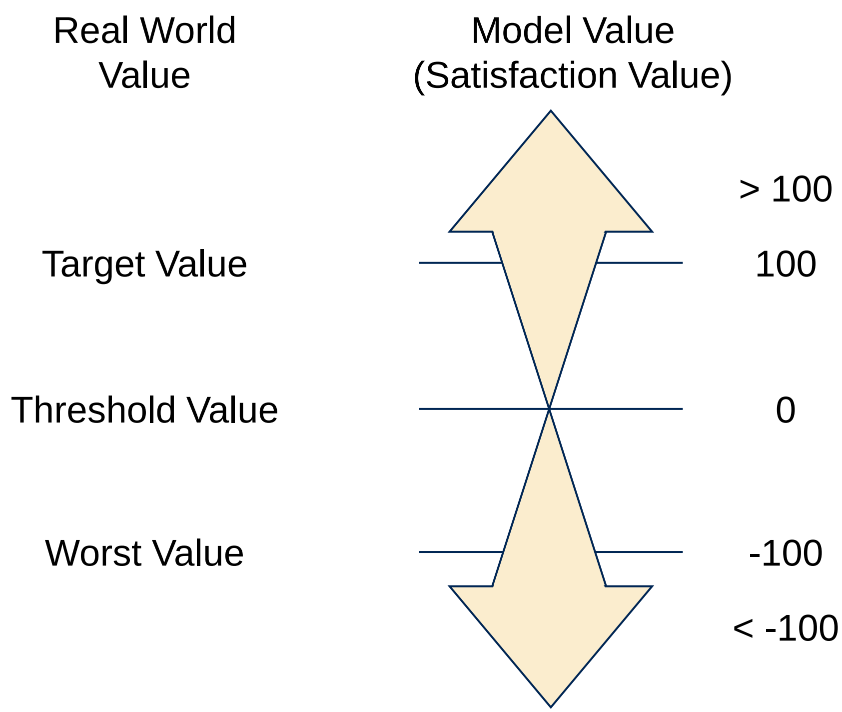
Key Performance Indicators

Example: Commuting



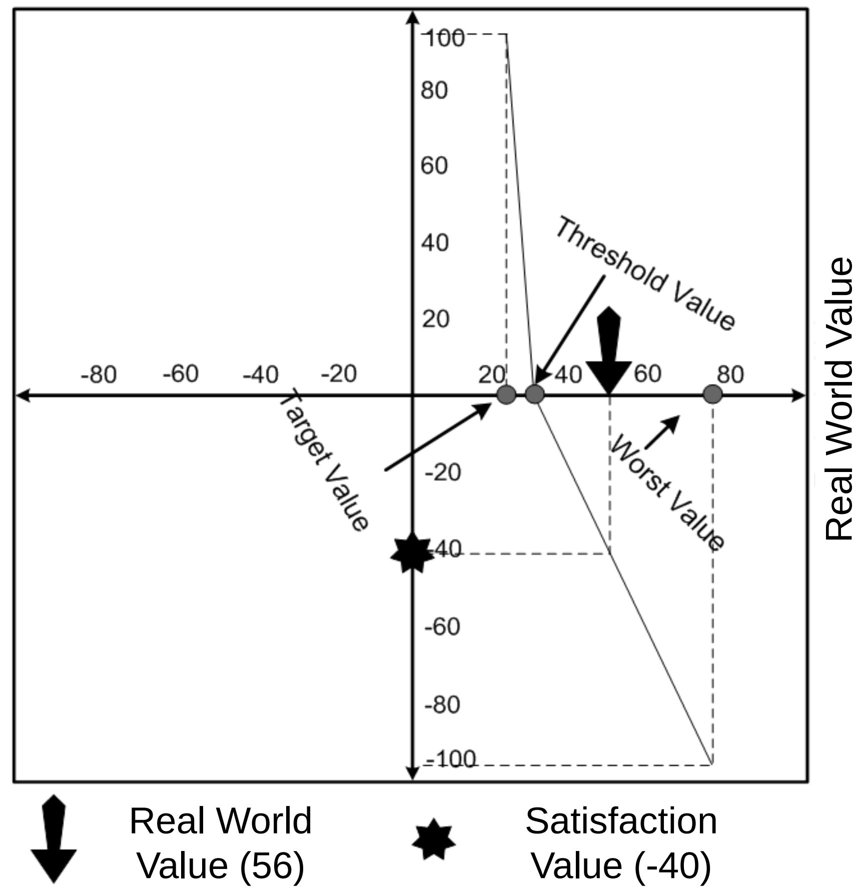
Key Performance Indicator (KPI)

From Real World Values to Model Values (1/2)

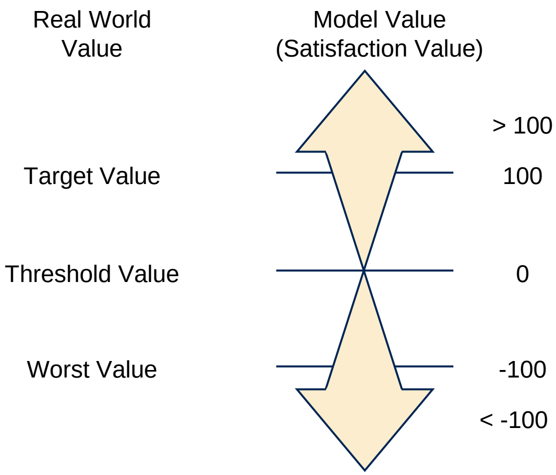


Target Value (20), Threshold Value (40), Worst Value (80)

Satisfaction Value



From Real World Values to Model Values (1/2)



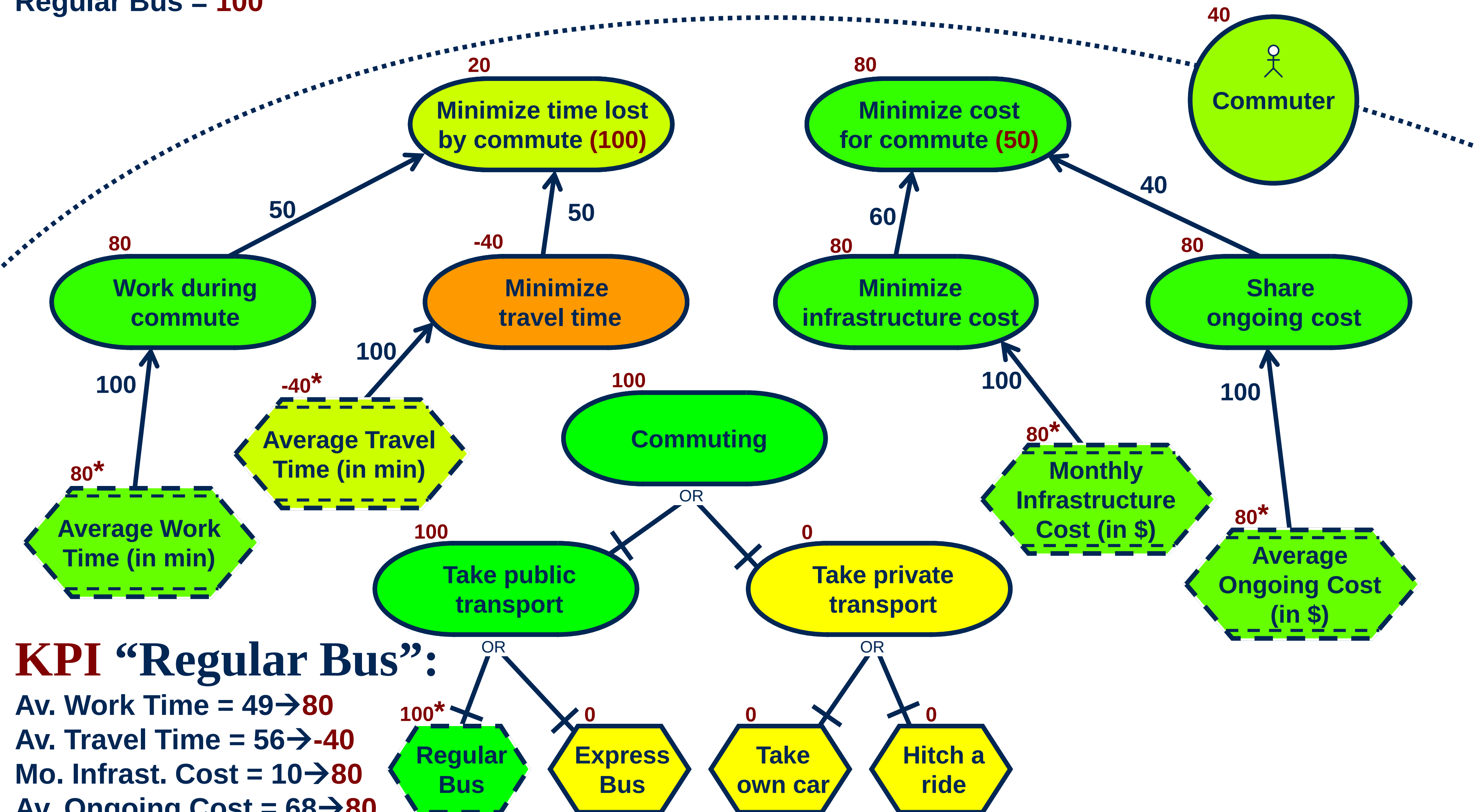
KPI	Regular Bus	Express Bus	Take own car	Hitch a ride
Average Work Time (in min) Target Value (60) Threshold Value (5) Worst Value (0)	49 → 80	29.75 → 45	4.5 → -10	4.5 → -10
Average Travel Time (in min) Target Value (20) Threshold Value (40) Worst Value (80)	56 → -40	52 → -30	24 → 80	24 → 80
Monthly Infrastructure Cost (in $) Target Value (0) Threshold Value (50) Worst Value (500)	10 → 80	10 → 80	455 → -90	140 → -20
Average Ongoing Cost (in $) Target Value (60) Threshold Value (100) Worst Value (200)	68 → 80	76 → 60	120 → -20	92 → 20

Strategy Execution with KPIs (1/2)

Strategy "Regular Bus":

Regular Bus = 100

Example: Commuting



KPI "Regular Bus":

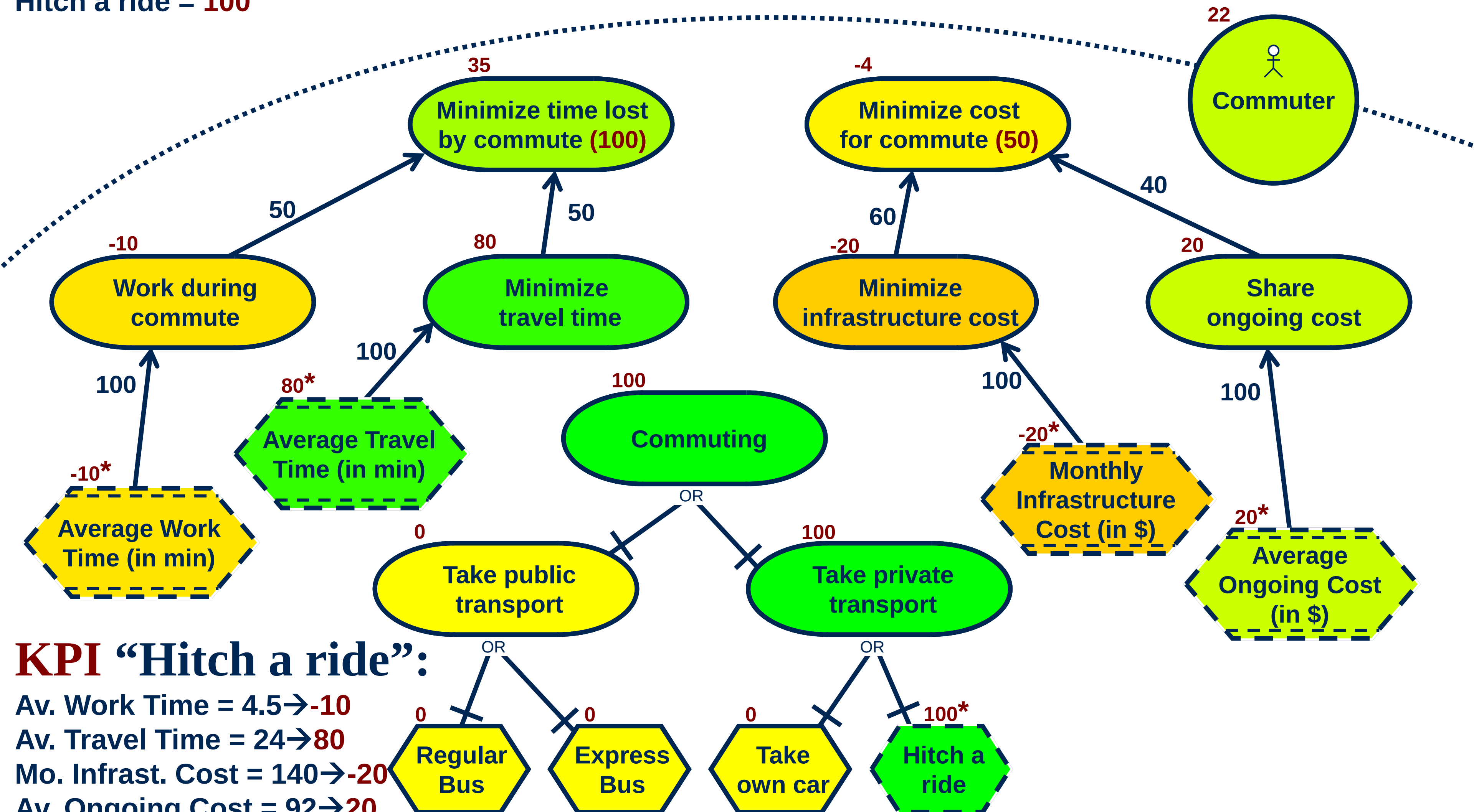
Av. Work Time = 49→80
Av. Travel Time = 56→-40
Mo. Infrast. Cost = 10→80
Av. Ongoing Cost = 68→80

Strategy Execution with KPIs (2/2)

Strategy "Hitch a ride":

Hitch a ride = 100

Example: Commuting



KPI "Hitch a ride":

Av. Work Time = 4.5→-10
Av. Travel Time = 24→80
Mo. Infrast. Cost = 140→-20
Av. Ongoing Cost = 92→20

Conclusion and References

Conclusion

- The User Requirements Notation (URN) is an ITU-T standard
- URN is a competitive notation for requirements engineering activities including business process modeling and analysis
- Modeling with the Goal-oriented Requirement Language (GRL)
  - Focuses on answering "why" questions
  - Intentions, business goals, functional / non-functional requirements, rationales
  - Key Performance Indicators (KPIs) are a must in a business environment to measure and monitor processes, compliances, and non-functional properties
- Modeling with Use Case Maps (UCMs)
  - Focuses on answering "what" and "when" questions
  - Scenarios, business processes, services, architectures

- Enables the elicitation/specification of systems, business processes and goals, standards, and products, as well as their analysis/validation from various angles
- Tool support

More Advanced URN Modeling Concepts

- Both GRL and UCMs have more advanced modeling concepts not covered in this course
- URN
  - Data model in support of GRL evaluations and UCM scenario definitions
  - Extensibility through profiling mechanism (metadata, URN links, and OCL)
- GRL
  - More types of intentional elements and links
  - More evaluation algorithms
  - KPI conversion and aggregation
- UCMs
  - More types of components
  - More advanced ways of binding elements on plug-in maps to stubs
  - More types of path elements that allow various workflow patterns to be modeled concisely
  - Extensions for failure and exception handling

References

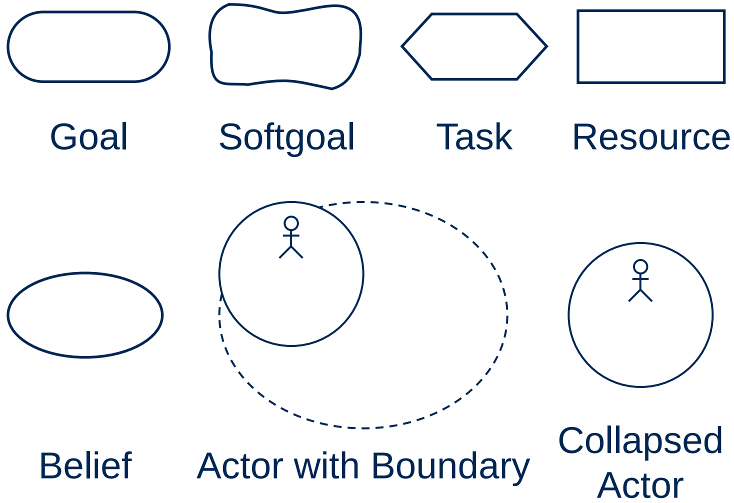
General

- URN Virtual Library (~350 entries), http://www.UseCaseMaps.org/pub/
- URN tool: jUCMNav, University of Ottawa, http://jucmnav.softwareengineering.ca/jucmnav/
- ITU-T, Recommendation Z.151 (11/08): User Requirements Notation (URN) - Language Definition, Geneva, Switzerland, approved November 2008.
- URN website: http://www.usecasemaps.org/urn

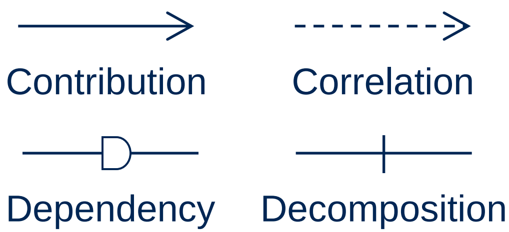
Overview of URN

- Amyot, D. and Mussbacher, G.: “User Requirements Notation: The First Ten Years, The Next Ten Years”. Invited paper, *Journal of Software (JSW)*, Academy Publisher, 6(5):747-768 (May 2011)
- Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E.: “Evaluating Goal Models within the Goal-oriented Requirement Language”. *International Journal of Intelligent Systems (IJIS)*, Wiley, 25(8):841–877 (2010)
- Amyot, D., and Mussbacher, G.: “Development of Telecommunications Standards and Services with the User Requirements Notation”. *Joint ITU-T and SDL Forum Society Workshop on "ITU System Design Languages"*, Geneva, Switzerland (September 2008)

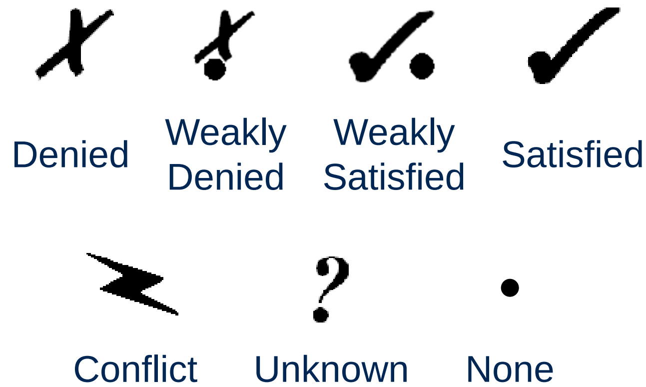
Appendix: Notation Overview – GRL



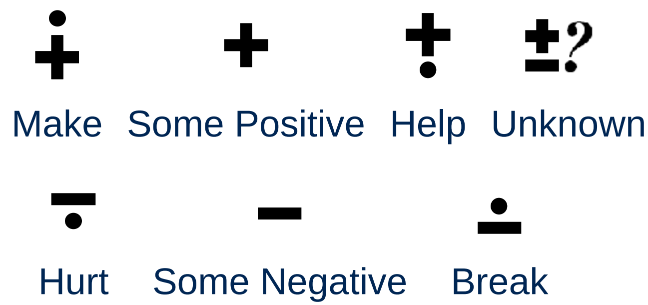
(a) GRL Elements



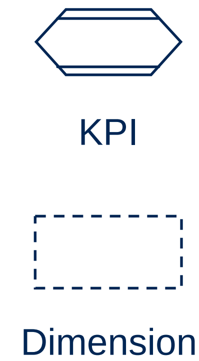
(b) GRL Links



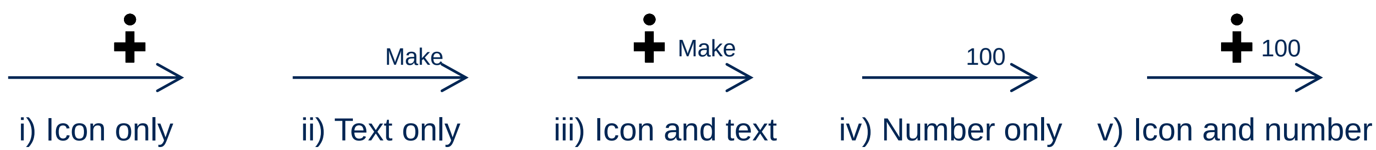
(c) GRL Satisfaction Levels



(d) GRL Contributions Types



(f) Performance Modeling



(e) Representations of Qualitative and Quantitative Contributions

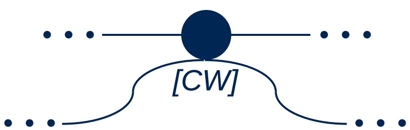
Appendix: Notation Overview – UCMs (Behavior)

Path with Start Point with Precondition CS and End Point with Postcondition CE

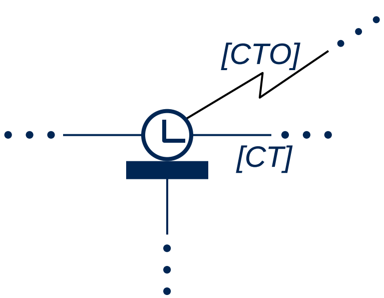
Responsibility

Empty Point

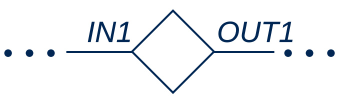
Direction Arrow



Waiting Place with Condition and Asynchronous Trigger

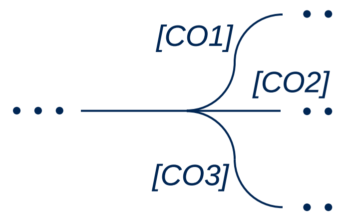


Timer with Timeout Path, Conditions, and Synchronous Release

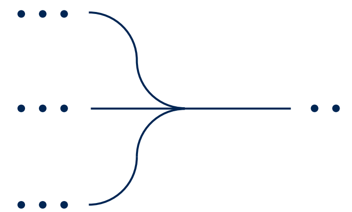


Static Stub with In-Path ID and Out-Path ID

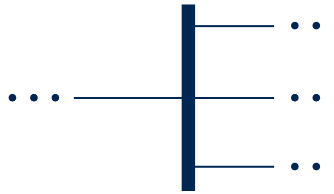
Dynamic Stub with In-Path ID and Out-Path ID



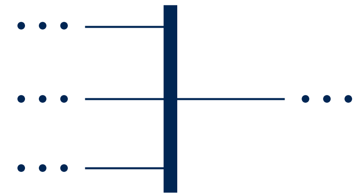
Or-Fork with Conditions



Or-Join



And-Fork



And-Join

Synchronizing Stub with In-Path ID, Out-Path ID, and Synchronization Threshold

Blocking Stub with In-Path ID, Out-Path ID, Synchronization Threshold, and Replication Indicator

Appendix: Notation Overview – UCMs (Structure)

Components: