

COE691: Software Requirements and Specifications

Dr. Rasha Kashef
Dept. of Electrical, Computer Engineering, and Biomedical Engineering

RYERSON UNIVERSITY

Everyone Makes a Mark

Last Week

- **Requirements Classifications and Organizations**
- **Requirements Conflicts**
- **Requirements Prioritization and Negotiation**

Week 6 :Agenda

- User Requirements Notation (URN)
- Goal Modelling Tools
- *Goal-oriented Requirements Language* (GRL)
  - Notations
  - Examples
  - Strategies and Evaluations

Recall:

- What is a goal?
  - Representation of stakeholder objectives
- What is a goal model?
  - Hierarchical arrangement of goals – Demonstrates relationships between goals
- Examples of goals:
  - Camera sensor must have 180 degree field of view – Radar sensor is always on
  - All sensors must provide reliable data
- Examples of non-goals:
  - Camera software implemented in C
  - Radar housing painted red

Recall:

- Goal hierarchy
  - Goals can be decomposed from high-level objectives to low-level requirements
  - Each goal refined with sub-goals that define how it can be satisfied
  - Leaf-level goals are considered to be requirements
- Functional goals
  - "Hard" goals – Functions that system will perform
  - Well-defined criteria for satisfaction
    - E.g., vehicle always stays within lane markings
- Non-functional goals
  - "Soft" goals – Desired system qualities
  - Hard to define and quantify – Reliability – Quality
    - E.g., automatic stop is not jarring to passenger

Goal exercise

- Identify the goals in the following paragraph: Company X is designing a new autonomous vehicle. Their autonomous vehicle system comprises at least two sensors: a camera and a radar. Both the camera and radar are responsible for sensing objects at a minimum distance of 10 meters. These sensors can communicate to a CPU via a secure CAN bus, at which point the CPU parses the incoming data. For safety purposes, at least one sensor must be active at all times.

Goal exercise

- Identify the goals in the following paragraph: Company X is designing a new autonomous vehicle. Their autonomous vehicle system comprises at least two sensors: a camera and a radar. Both the camera and radar are responsible for sensing objects at a minimum distance of 10 meters. These sensors can communicate to a CPU via a secure CAN bus, at which point the CPU parses the incoming data. For safety purposes, at least one sensor must be active at all times.

Principles

- Goals are objectives which a system should achieve through cooperation of actors in the intended software and in the environment.
- Goal modeling is especially useful in the early phases of a project.
- Projects may consider how the intended system meets organizational goals, why the system is needed and how the stakeholders' interests may be addressed.

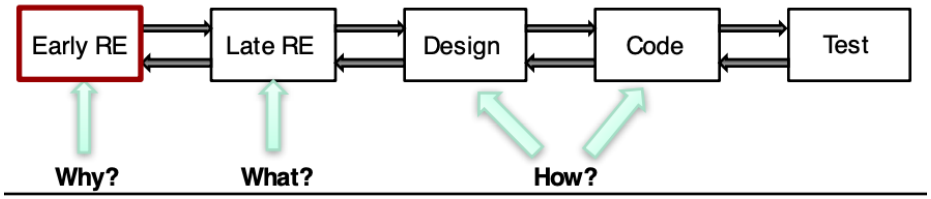
A Goal Model

- Expresses the relationships between a system and its environment (i.e. not only on what the system is supposed to do, but why).
- The understanding thus gives, of the reasons why a system is needed, in its context, is useful because "systems are increasingly used to fundamentally change business processes rather than to automate long-established practices".
- Clarifies requirements : Specifying goals leads to asking "why", "how" and "how else".
- Stakeholders' requirements are often revealed in this process, with less risk of either missing requirements, or of over-specifying (asking for things that are not needed).

A Goal Model

- Allows large goals to be analyzed into small, realizable goals.
- Deals with conflicts :
  - goal modeling can identify and help to resolve trade-offs between cost, performance, flexibility, security and other goals.
  - It can reveal divergent interests between stakeholders.
  - It can identify conflicts because meeting one goal can interfere with meeting other goals.
- Enables requirement completeness to be measured: requirements can be considered complete if they fulfil all the goals in the goal model.
- Connects requirements to design: for example, the i* "Non-Functional Requirements (NFR) framework" uses goals to guide the design process.

Goal Models



- Why do we use goal models?
  - Provide rationale for requirements
  - Identify stable information in system objectives
  - Guide requirements elaboration / elicitation
  - Provide visual depiction of relationships and dependencies between objectives
- When to use goal models
  - Early in requirements engineering process
    - Identify problems
    - Explore solutions and alternatives
    - Performed prior to UML modeling
    - Continually refine goal model as new requirements or obstacles surface

Notations

- There are several notations in use for goal models in software development, including:
- i* (pronounced "eye-star") and a variant, GRL
- KAOS
- UML Use Case diagram
- Other notations have been proposed by researchers, while the Goal Structuring Notation (GSN) and GRL are sometimes used to make safety cases to satisfy the regulator in safety-related industries.

Goal Modeling in i* , GRU

- The i* goal modeling notation provides two kinds of diagram:
  - "Strategic Dependency" (SD), defining relationships between roles in terms of specific goals that one role depends on the other role to provide.
  - "Strategic Rationale" (SR), analyzing the goals identified on the SD model into subsidiary goals and tasks.
- i* shows each role (an actor, agent or position) as a large circle containing the goals, tasks, and resources which that role owns.
- Ownership in i* means that the role desires the satisfaction of its goals, either for its own benefit or for the benefit of some other role.
- Goals may be accompanied by "obstacles" (negative goals) to be surmounted.
- Non-functional goals can be modeled as "soft goals" in i*: they are diagrammed as clouds or indented ovals.

Goal Modeling in i* , GRU

- The i* Framework shares many concepts with GRL. However, it also contains many types of actors (e.g., agents, roles, and positions) and associations (e.g., generalization, instantiation, covers, occupies, is part of, and plays) that GRL does not differentiate.
- For the links that are defined in both languages, there are more restrictions on their usage in i* than in GRL.
- i* also distinguishes between strategic design models (where the focus is on actors and dependencies) and strategic rationale models (where the internal concerns of the actors are defined, hence providing a rationale for the dependencies).
- GRL, on the other hand, brings in strategies, a combination of qualitative and quantitative contributions, actor evaluations, generic URN links, and metadata. GRL also makes no distinction between strategic and rationale models, although both views can be expressed in different diagrams of a same GRL model.
- The OpenOME tool enables the creation and analysis of i* models.24 In the i* framework, an interactive (semi-automated), forward propagation algorithm with qualitative values.

Goal Modeling in KAOS

- **KAOS**, is a goal-oriented software requirements capturing approach in requirements engineering. It is a specific Goal modeling method; another is i*.
- It allows for requirements to be calculated from goal diagrams.
- KAOS stands for *Knowledge Acquisition in automated specification* or *Keep All Objectives Satisfied*.
- The University of Oregon and the University of Louvain (Belgium) designed the KAOS methodology in 1990 by Axel van Lamsweerde and others.
- It is now widely taught worldwide at the university level for capturing software requirements.
- The KAOS goal modeling notation provides a way of defining goals and obstacles, underpinned by a formal (mathematical) method of analysis

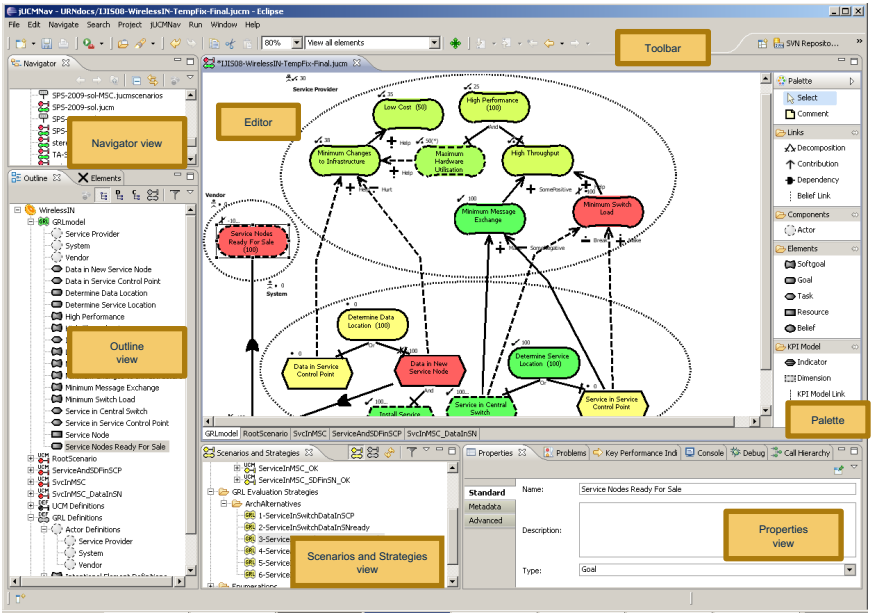
Goal Modeling in UML

- UML's use case diagram provides a simple goal modeling notation.
- The bubbles name functional goals, so a Use case diagram forms a simple functions-only goal model, use cases cover only the behavioral requirements.
- Roles are shown as actors (stickmen on the diagram), linked to the use cases in which they take part.
- The use cases are drawn as elliptical bubbles, representing desired behavioral goals.

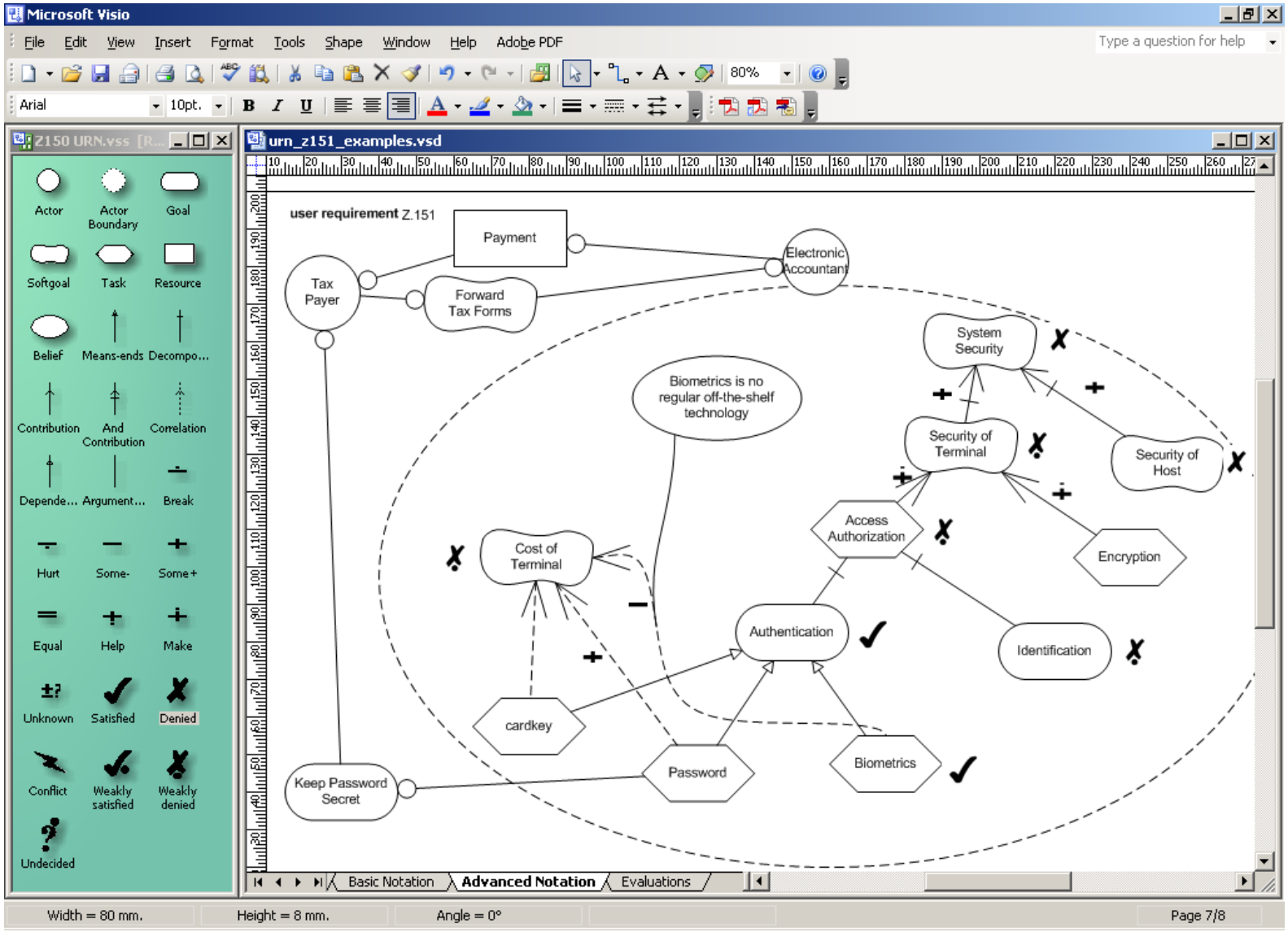
Goal Modelling Tools

jUCMNav (Eclipse Plug-in)

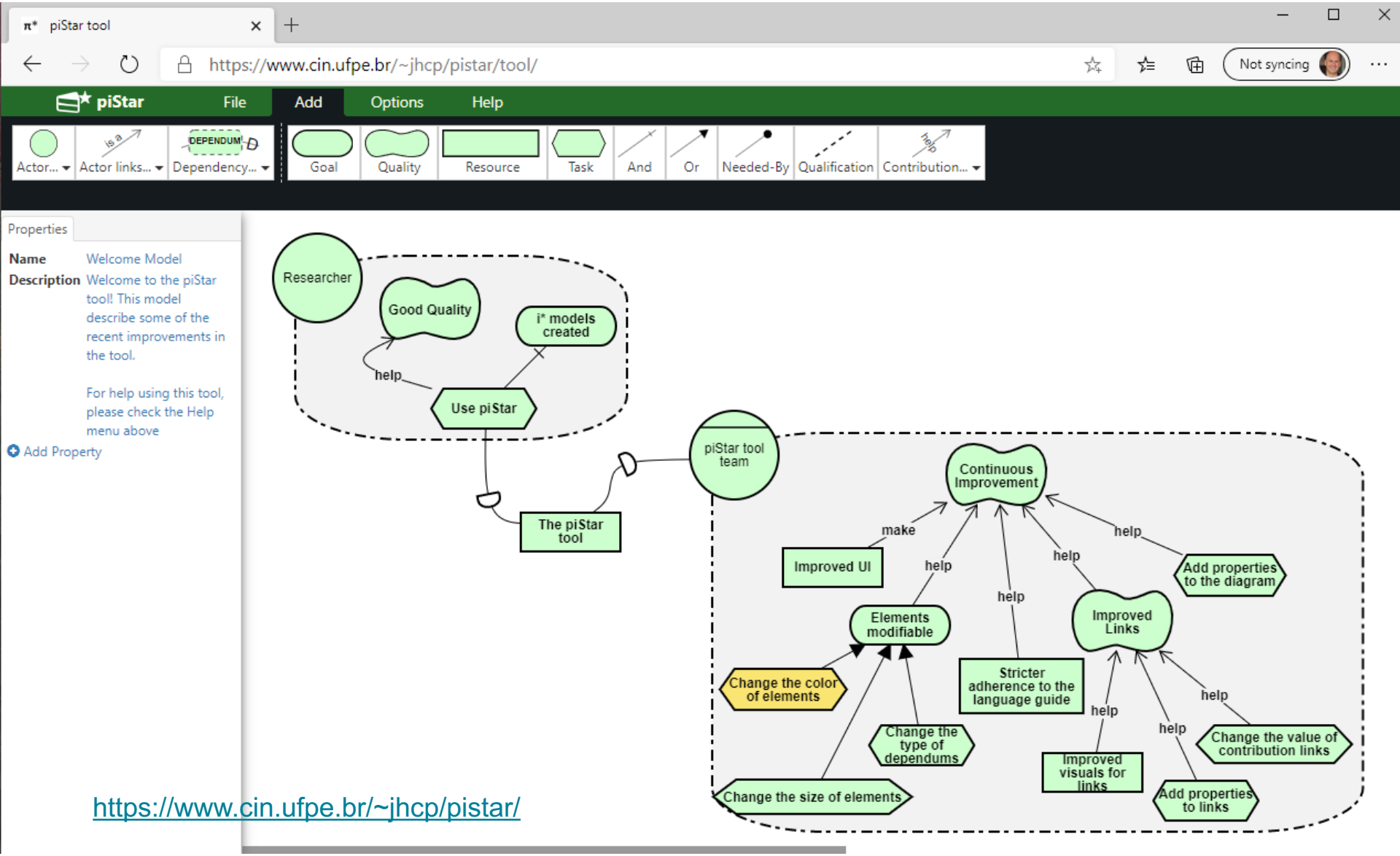
- Features for GRL
  - 7 GRL evaluation algorithms, with color highlight
  - One model, multiple diagrams
  - References to actors and intentional elements
    - Drag&drop from outline or via properties
  - Auto-layout
  - Catalogues
    - For exporting/importing/reusing common models
  - Export graphics (.bmp, .gif, .jpg)
  - Export strategy evaluations (.csv)
  - URN links (for integration with UCMs)
  - Export to DOORS



Tool Support – SanDriLa (Plug-in for Visio)

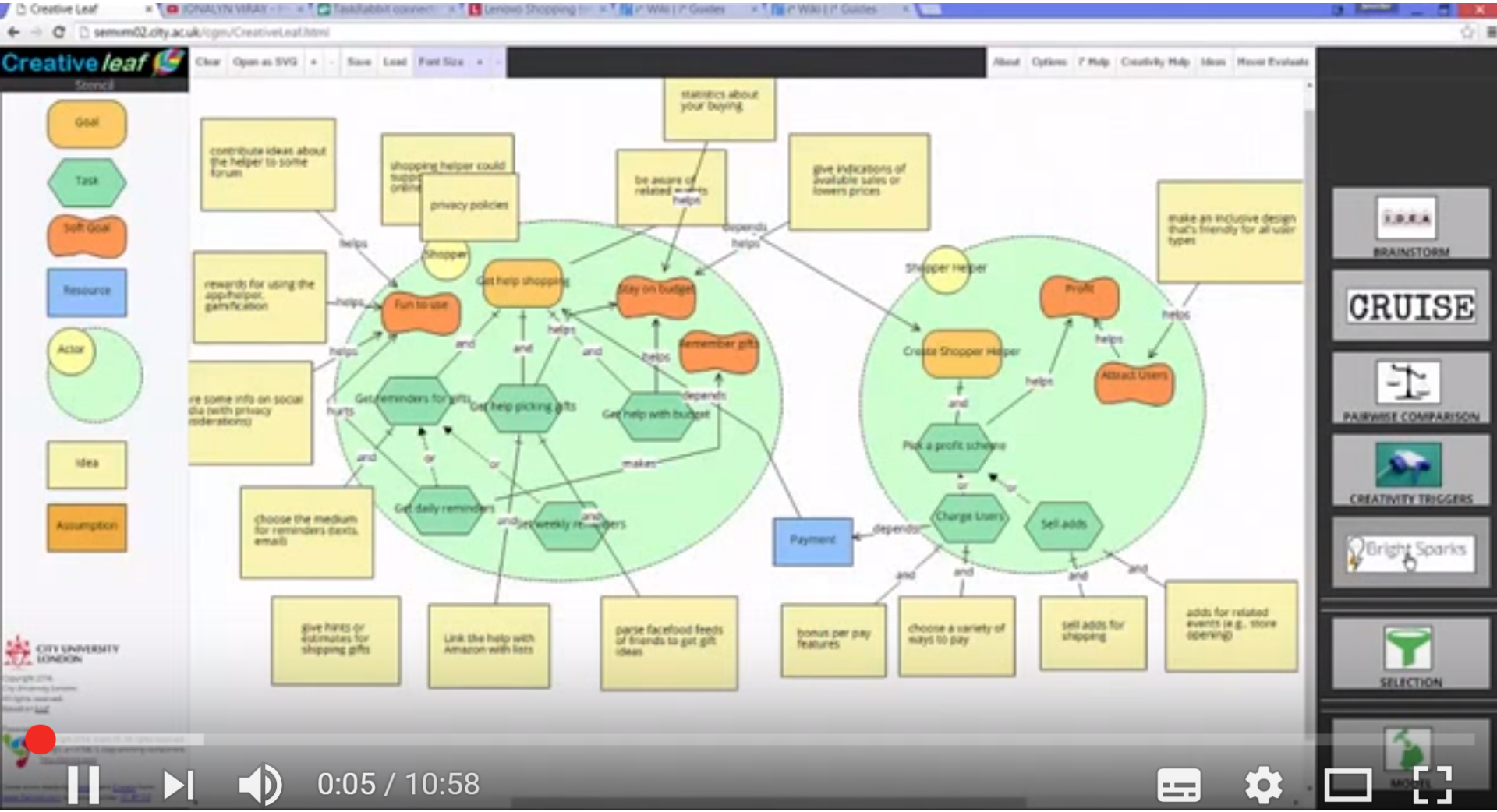


Web Tool – piStar



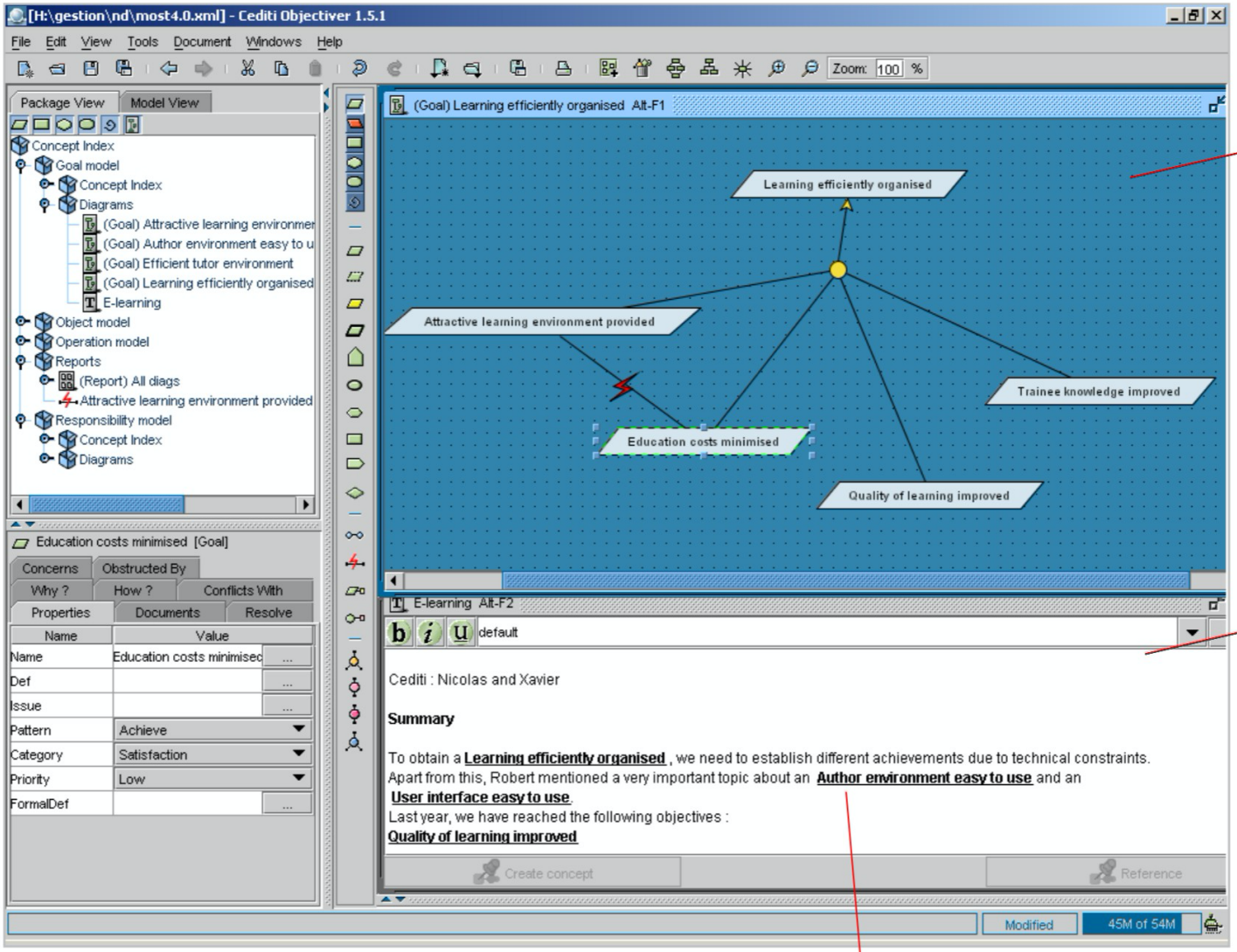
https://www.cin.ufpe.br/~jhcp/pistar/

Web Tool – Creative Leaf



- Online: http://semvm02.city.ac.uk/cgm/
- Videos: https://www.youtube.com/channel/UCjU8aqp-93qP8DnbDKXQpMQ

Commercial Tool: Objectiver (for KAOS)



Indexes

Diagram viewer

Property viewer

Text viewer

http://www.objectiver.com/

Concept references are hyperlinks.

Goal-oriented Requirements Language (GRL)

User Requirements Notation (URN)

- User Requirements Notation (URN)
  - Focus on early stages of development with **goals** and **scenarios**
- Combined use of two complementary notations:
  - **Goal-oriented Requirement Language** (GRL)
    - for goals and non-functional requirements
    - http://www.cs.toronto.edu/km/GRL/
  - **Use Case Maps** (UCM)
    - for operational requirements and architectures
    - http://www.UseCaseMaps.org/
    - http://www.UseCaseMaps.org/pub

Goal-oriented Requirements Language (GRL)

- **Goal-oriented Requirements Language** (GRL), an i*-based modeling language used in systems development, is designed to support goal-oriented modeling and reasoning about requirements especially the non-functional requirements
- Application and Research Areas
  - Transformations to Message Sequence Charts and test goals
  - Architectural Evaluations
  - Performance engineering
  - Business process modelling and management
  - Requirements management and policy compliance
  - Pattern formalization
  - Reverse engineering
  - Aspect-oriented requirements engineering

Why GRL?

- Goals are an important driver for requirements elaboration
- GRL expresses and clarifies tentative, ill-defined and ambiguous requirements
  - Supports argumentation, negotiation, conflict detection & resolution, and decisions
  - Captures decision rationale and criteria (documentation!)
- GRL identifies alternatives (requirements, architectures, means…)
- GRL provides traceability from strategic objectives to technical requirements
- GRL allows reuse of stable higher-level goals when the system evolves
- Nothing quite like this in UML…

GRL in a Nutshell

- Goal-oriented Requirement Language
  - graphical notation
  - connects requirements to business objectives
  - allows reasoning about (non-functional) requirements
  - has its roots in i* and the NFR framework
- GRL models the "why" aspect
  - objectives, alternatives, as well as decision rationales
  - little or no operational details
- Supports goal and trade-off analysis and evaluations

UCMs in a Nutshell

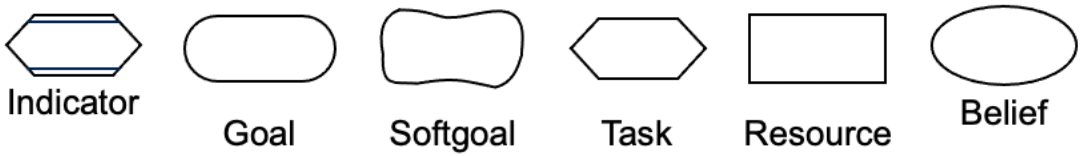
- Use Case Maps
  - graphical scenario notation
  - causal relationships between responsibilities
  - scenario elements may (optionally) be allocated to components
- UCMs model the "what" aspects
  - functional requirements as scenarios
  - integration and reusability of scenarios
  - guidance for architecture and detailed behaviour
- Performance analysis, conflict detection, transformations

GRL - UCM Relationship

- Goal-based approach
  - Focuses on answering “why” questions
  - Functional and non-functional requirements
- Scenario-based approach
  - Focuses on answering “what” questions
- Goals are *operationalized* into tasks and tasks are elaborated in (mapped to) UCM scenarios
  - Focus on answering “how” questions
- Enables completeness and consistency analysis
- User-defined links for requirements management
  - Any GRL element can be linked to any UCM element

Goal-oriented Requirements Language (GRL)

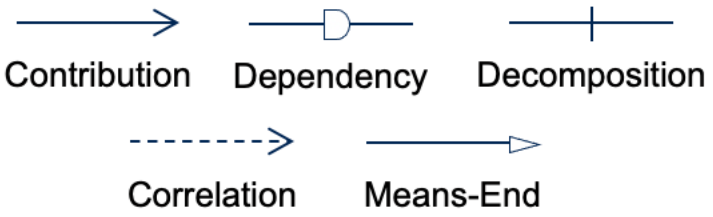
- Goal-oriented Requirements Language (GRL) allows to express conflict between goals and helps to make decisions that resolve conflicts. There are three main categories of concepts in GRL:
  - Intentional elements,
  - Relationships, and
  - Actors
- They are called for intentional because they are used in models that primarily concerned with answering "why" question of requirements (for ex. why certain choices for behavior or structure were made, what alternatives exist and what is the reason for choosing of certain alternative.)



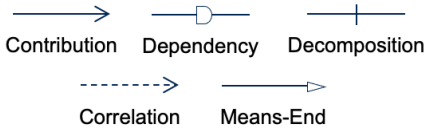
Intentional elements

- Intentional elements are: goal, soft goal, task, belief and resource.
- Goal is condition or situation that can be achieved or not. Goal is used to define the functional requirements of the system. In GRL notation goal is represented by a rounded rectangle with the goal name inside.
- Task is used to represent different ways of how to accomplish goal. In GRL notation task is represented by hexagon with the task name inside.
  - A task is a proposed solution that can achieve a goal and contribute to softgoals/qualities
- Softgoal is used to define non-functional requirements. It's usually a quality attribute of one of the intentional elements. In GRL notation softgoal is represented by irregular curvilinear shape with the softgoal name inside.
- Resource is a physical or informational object that is available for use in the task. Resource is represented in GRL as a rectangle.
- Belief is used to represent assumptions and relevant conditions. This construct is represented as ellipse in GRL notation.
- An indicator transforms a measure to a satisfaction level

Relationships(Links)

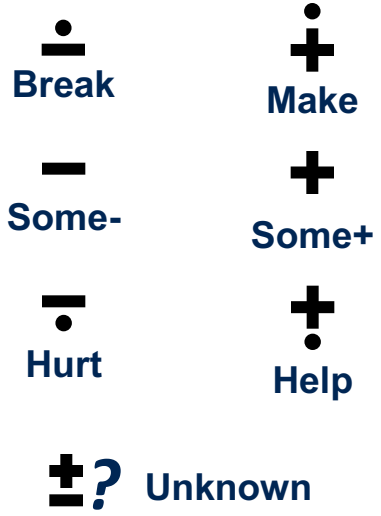


- **Relationships are: means-ends, decomposition, contribution, correlation and dependency.**
- Means-ends relationship shows how the goal can be achieved. For example, it can be used to connect task to a goal.
- Decomposition relationship is used to show the sub-components of a task.
- Contribution relationship describes how one element influence another one. Often a weighted means-ends relationship for brevity. +ve and -ve contribution allows for defeasible reasoning by way of Defenders and Defeaters.
- Correlation relationship describes side effects of existence of one element to others.
- Dependency relationship describe interdependences between agents.



Relationship Notations

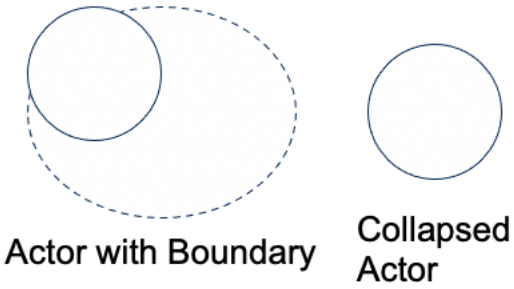
GRL Contributions Types: (qualitative)



- Contribution Link
  - A contribution describes desired impact or side effects (positive or negative)
  - Qualitative (symbols) or quantitative (numbers) contribution types are used for these links
- Decomposition Link
  - Defines what an intentional element needs to be satisfied
    - AND
    - OR
    - XOR
- Dependency Link (source → target)
  - The source of the dependency cannot be more satisfied than its target

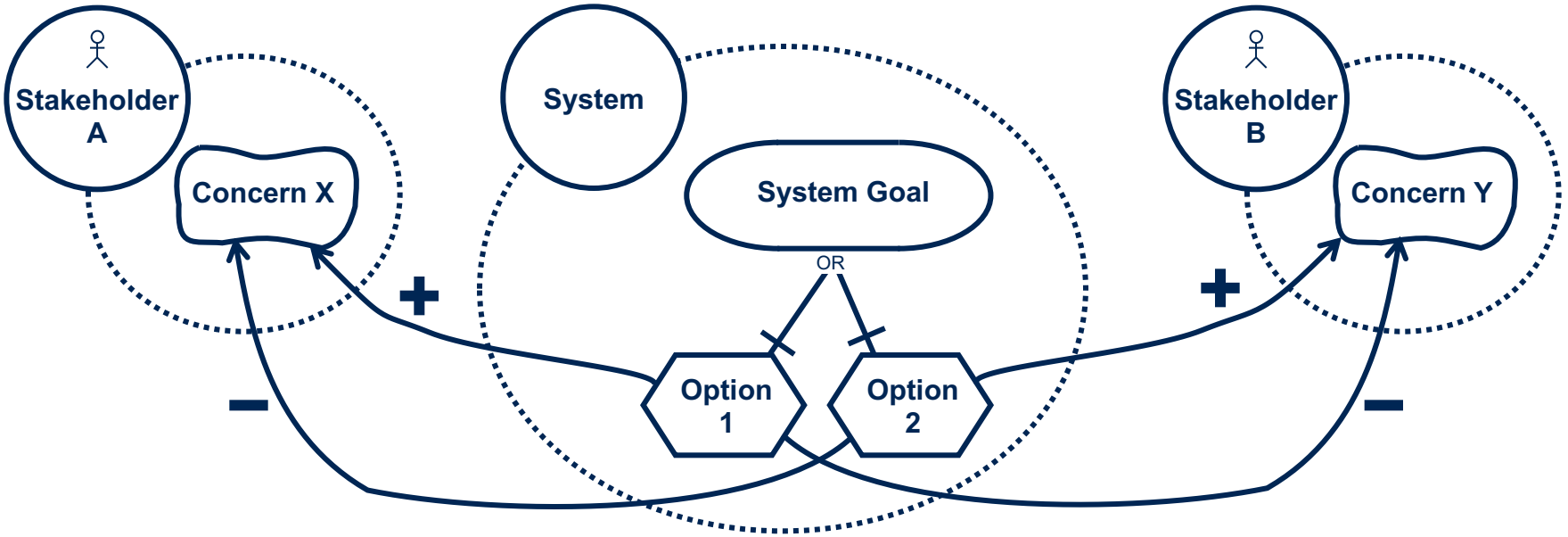
Actors

- Actor is an active object that carries out actions to achieve the goal. In GRL notation actor is represented as a circle with the actor name inside.
- Agent is a concrete actor, such as a human individual or machine.
- Role can be taken to be an behavioral aspect assigned to either an Actor or an Agent.



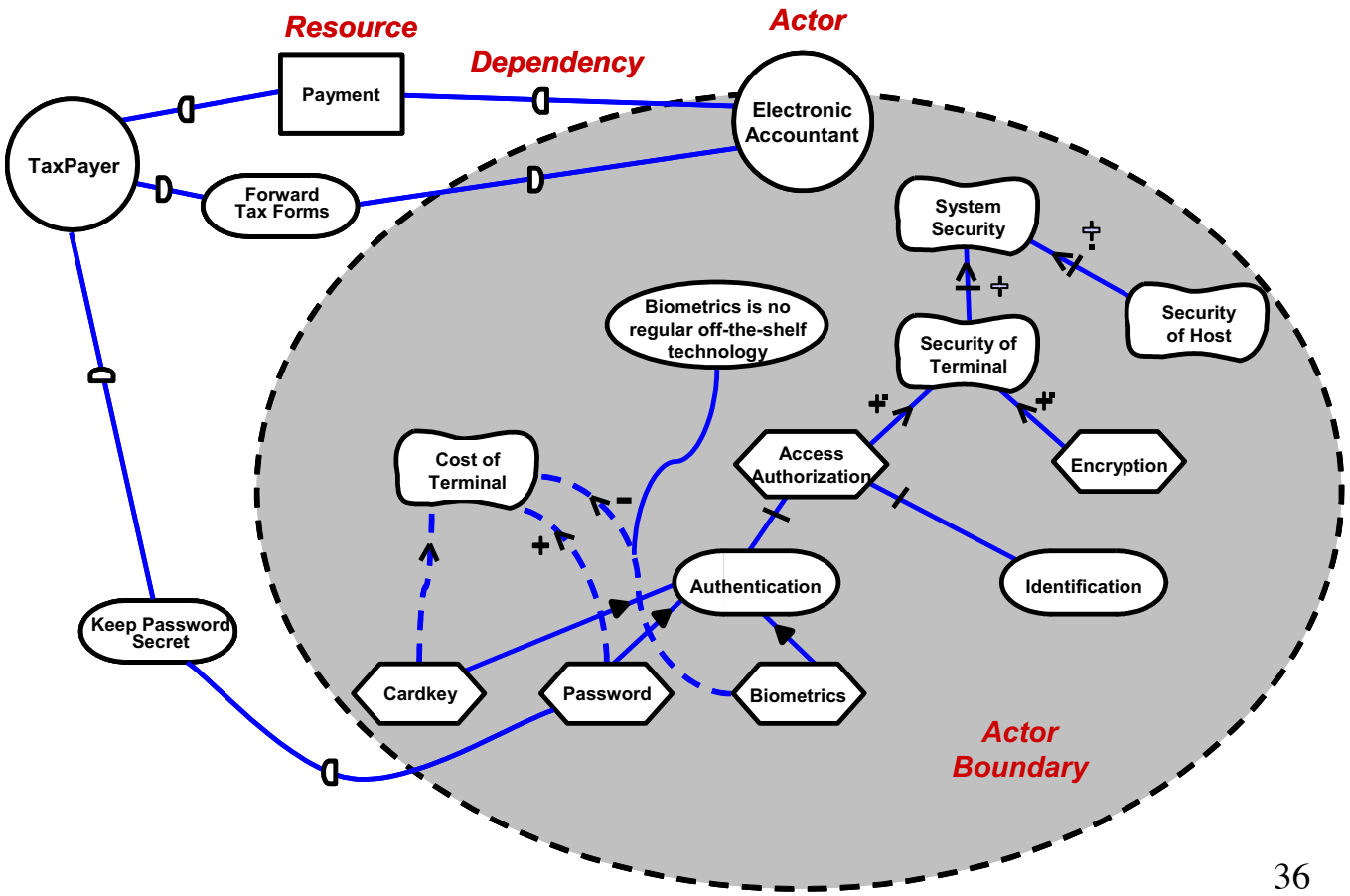
Recurring Pattern in Goal Modelling

- The system (actor) has several functional goals, with various alternative ways of performing them (shown with tasks)
- There are several stakeholders (actors) involved, with their own concerns, often non-functional (captured with softgoals)
- There are known contributions and side-effects from the potential solutions to the softgoals

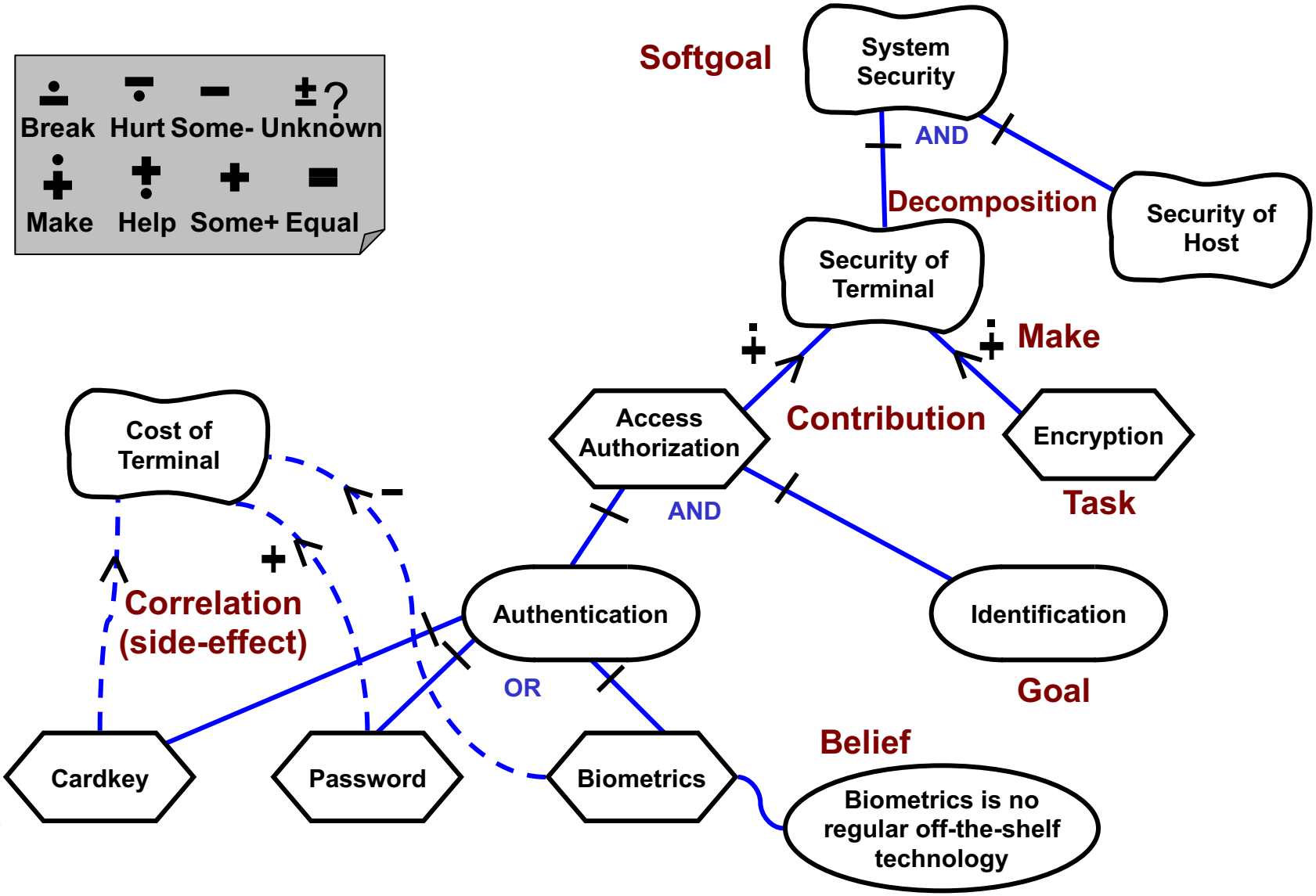


Advanced GRL Notation

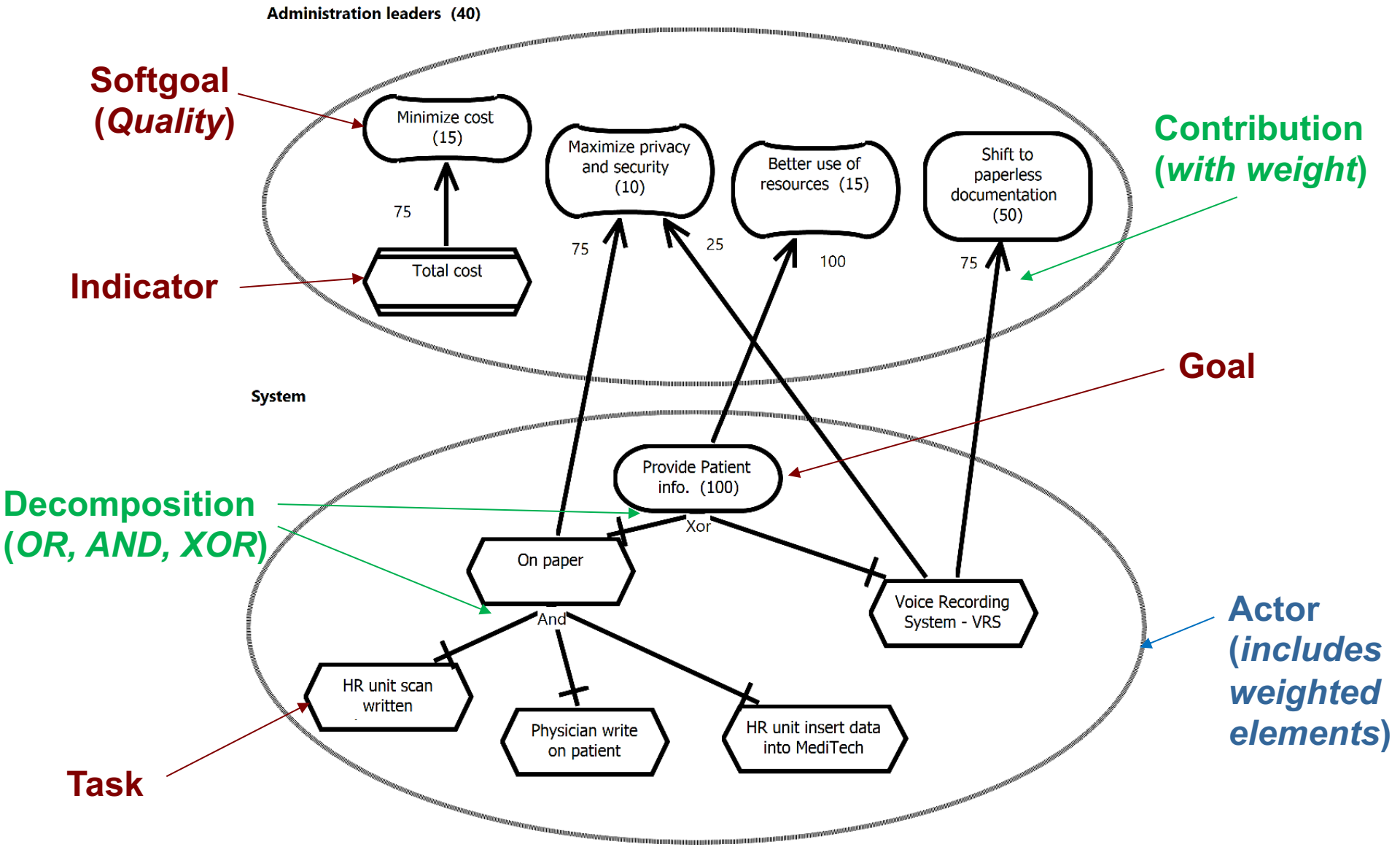
- GRL graphs can be allocated to *actors*
- *Dependencies* can be defined between actors, together with intermediate *resources*.



Basic GRL Notation

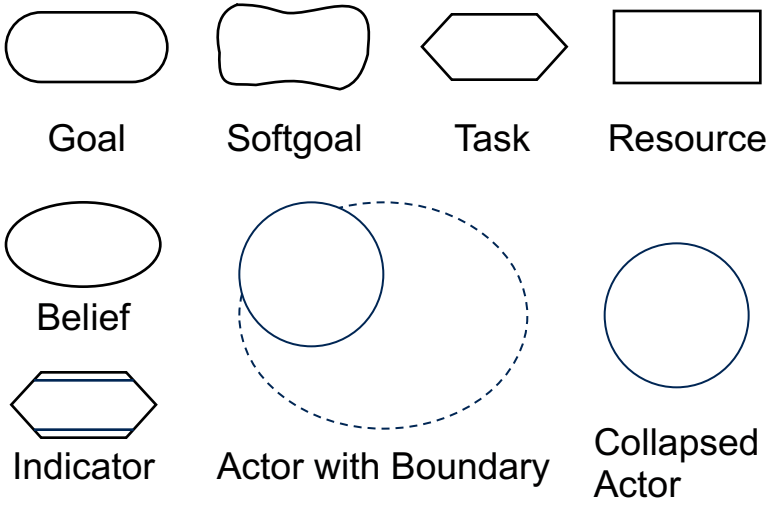


Example: Voice Recording System

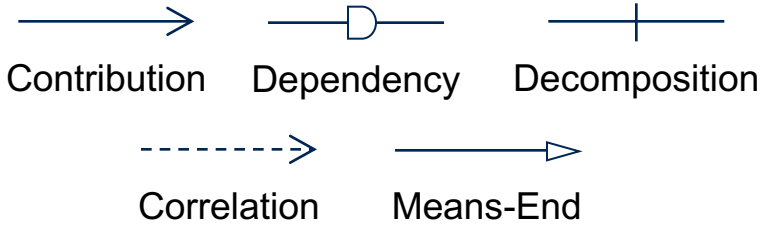


VRS = Voice Recording System

GRL Notation

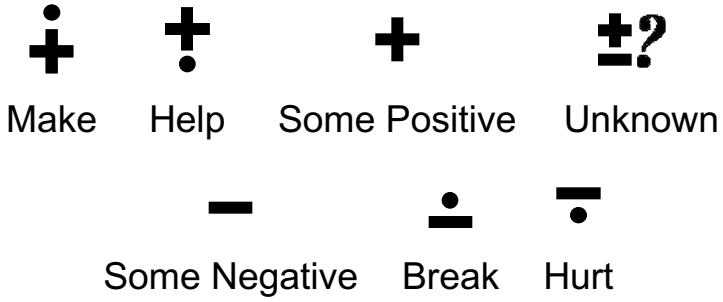


(a) GRL Elements

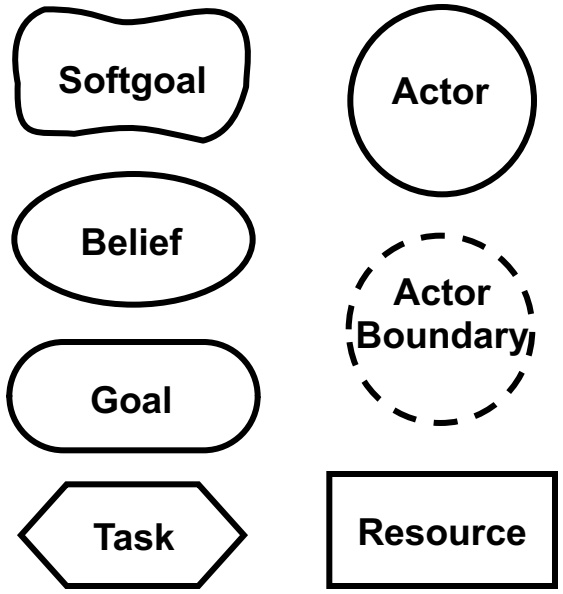


(b) GRL Links

(c) GRL Satisfaction Levels

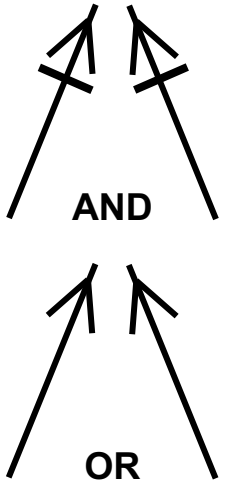


(d) GRL Contributions Types

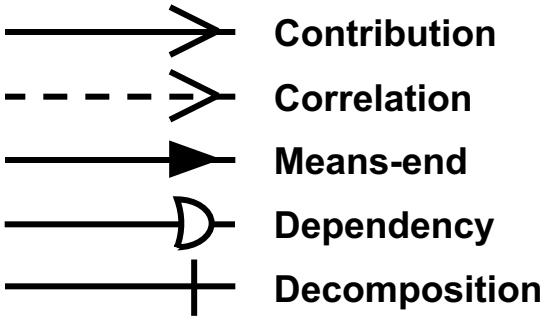


(a) GRL Elements

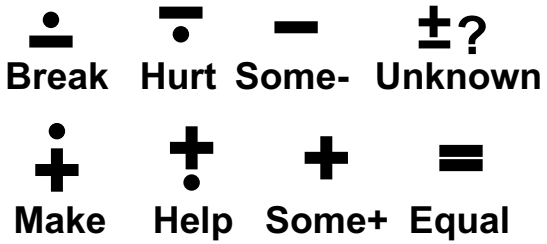
(b) GRL Satisfaction Levels



(c) Link Composition



(d) GRL Links

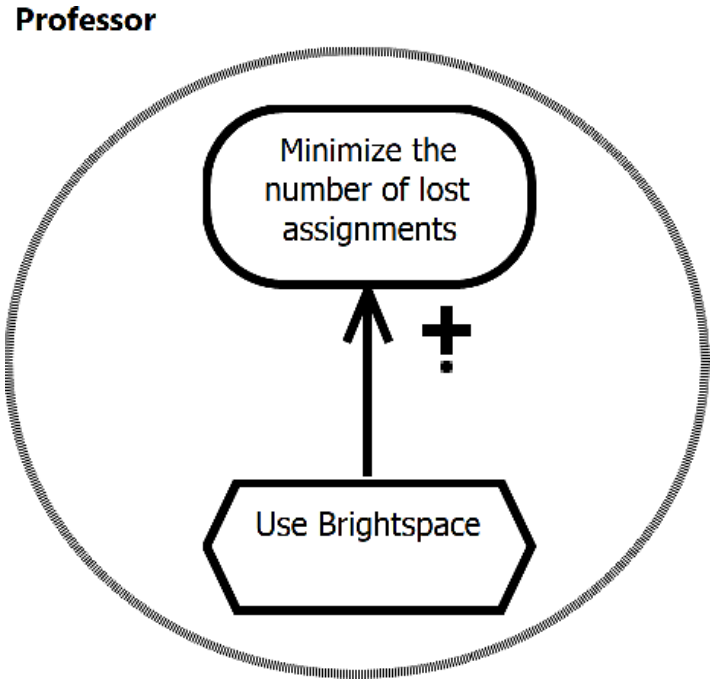


(e) GRL Contributions Types

User Stories and Goal models

- Pattern: *As an [Actor], I want to [Task] in order to [Contribution] achieve [Goal]*
- This user story identifies the task of an actor, a contribution level, and a contribution from the task to a goal (inside or outside the actor)

- *As a professor, I want to use Brightspace in order to help minimize the number of lost assignments.*




Goal Model
Analysis

GRL – Strategies and Evaluation Mechanism

- GRL 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 separately from the GRL graphs
  - Intentional elements tagged with (*) in jUCMNav
  - Strategies can be compared with each other for trade-off analysis

- In order to analyze the goal model and compare solutions with each other, jUCMNav's customizable evaluation mechanism executes the strategies
  - Propagating satisfaction levels to the other elements and to actors 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)

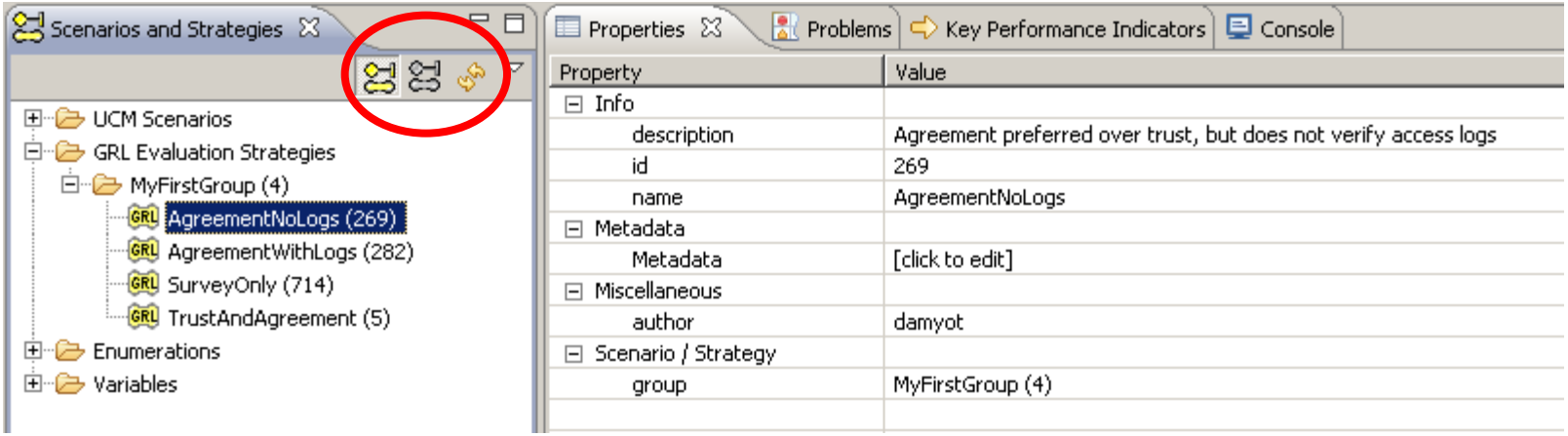
Evaluations with GRL

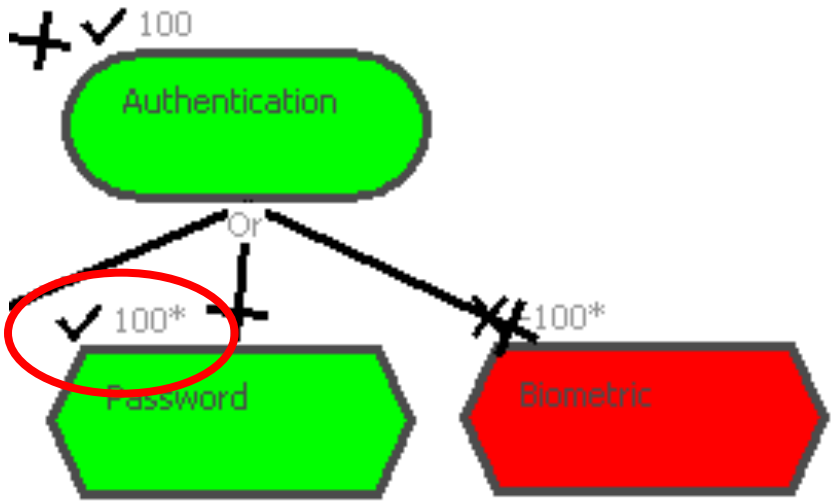
- *Evaluations* of GRL graphs show the impact of qualitative decisions on high level softgoals
- *Propagation* is usually bottom-up
- Fuzzy evaluation of *satisfaction level*
- Takes into consideration four parameters:
  - Degrees of satisfaction of children (satisficed, denied, …), from a strategy
  - Composition operators (AND, OR), Contributions and correlations (+/-, sufficient or not), and Dependencies
  - Importance defined for intentional elements and actors
- More complete than simple pros/cons tables or criteria evaluation matrices
- One could use numerical values and functions instead of qualitative values (see jUCMNav tool)

GRL Strategies

- User defined sets of initial evaluations
- Propagated to the rest of the model
- Numerical interpretation of satisfaction levels
- Implemented using the strategies view
- Visual coloured feedback
- Cycles permitted
- Evaluation of the impact of strategies on the operational and architectural aspects, using URN links

Strategies in jUCMNav





A star (*) indicates an initial value part of a given strategy. All the others are evaluated through a propagation algorithm.

GRL Evaluation – Qualitative or Quantitative Approach

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

Satisfaction Levels (Quantitative Approach)

Evaluation between -100 and 100.

E = -100 → Denied

-100 < E < 0 → Weakly Denied

E = 0 → Undecided

0 < E < 100 → Weakly Satisficed

100 → Satisficed

Numerical Evaluation: Contributions

- For each contribution, convert the contribution level to the corresponding factor

Make = 100

Help = 25

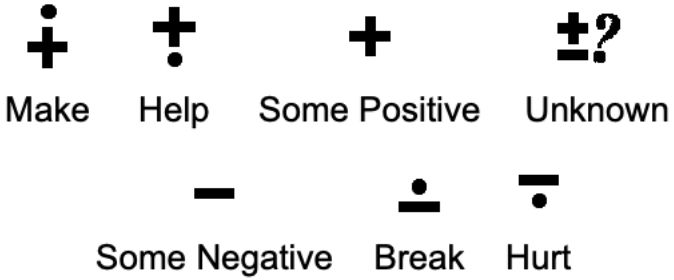
Some Positive = 75

Unknown = 0

Some Negative = -75

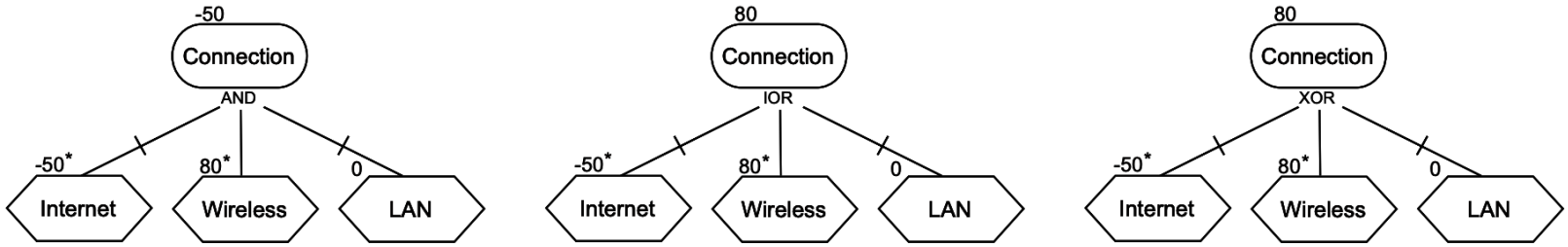
Hurt = -25

Break = -100



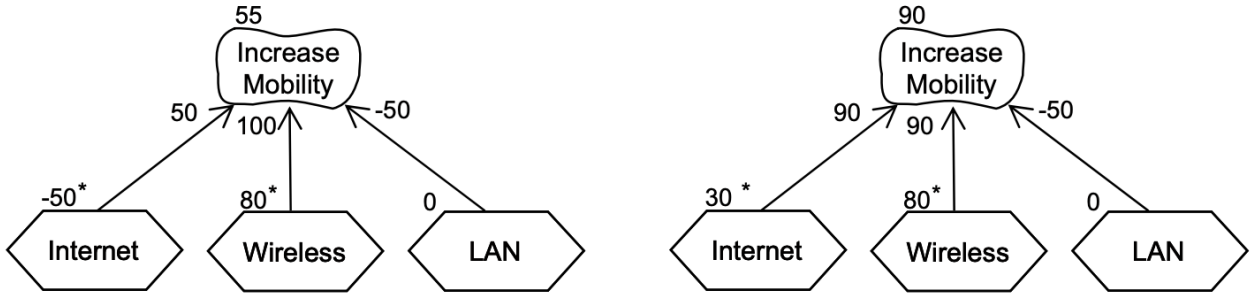
Satisfaction Levels (Quantitative Approach) Decompositions and Contributions

- Minimum for AND, maximum for OR



(a) AND decomposition (b) IOR decomposition (c) XOR decomposition

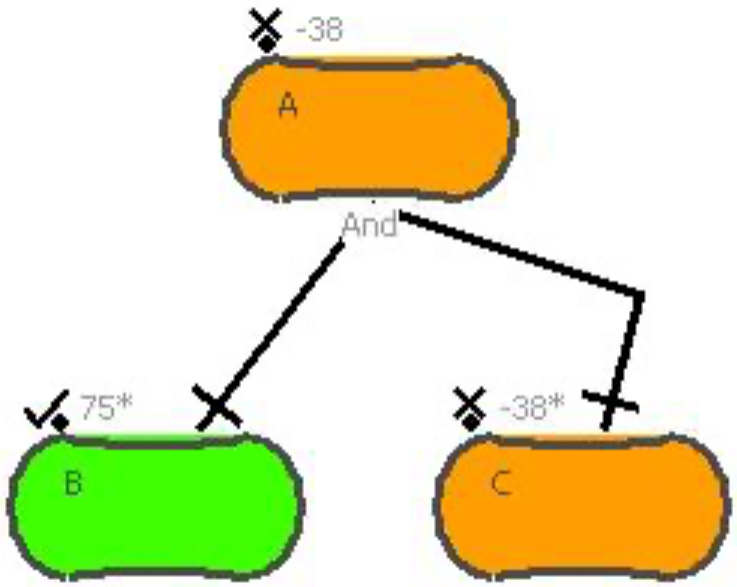
- Contributions are additive but normalized and take a tolerance into account



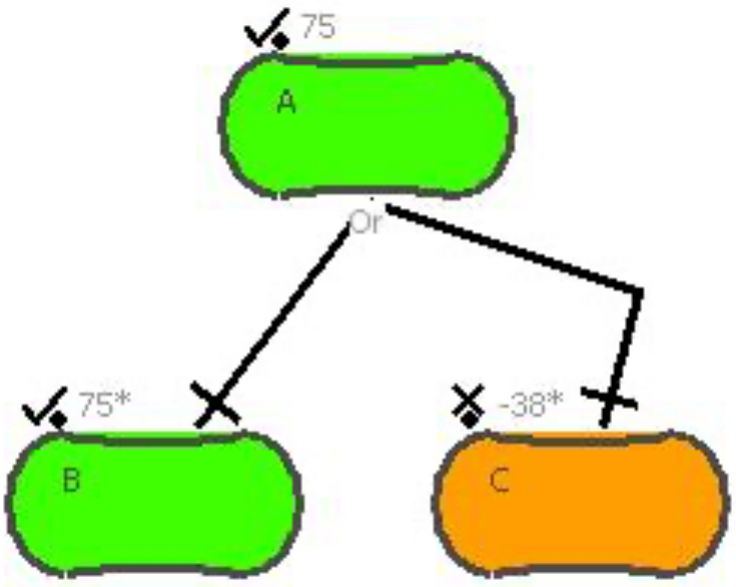
(a) Contributions (b) Contributions with a tolerance of 10

Numerical Evaluation: Decompositions

Minimum for AND, maximum for OR

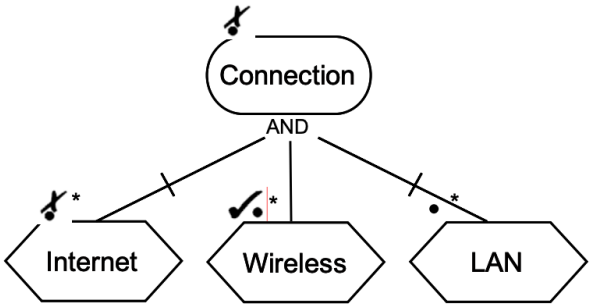


And Decomposition

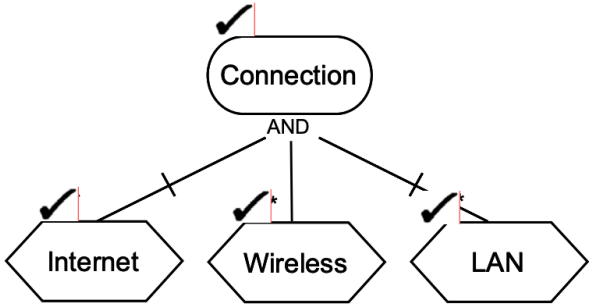


Or Decomposition

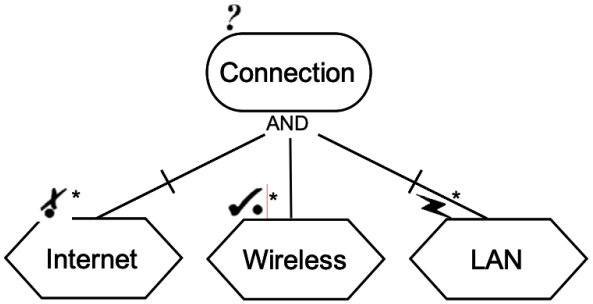
AND Decomposition: Satisfaction Levels (Qualitative Approach)



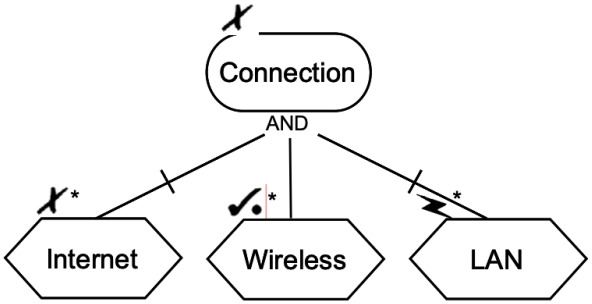
(a) Miminum is WeaklyDenied



(b) Miminum is Satisfied



(c) Minimum is Conflict: Undecided is propagated

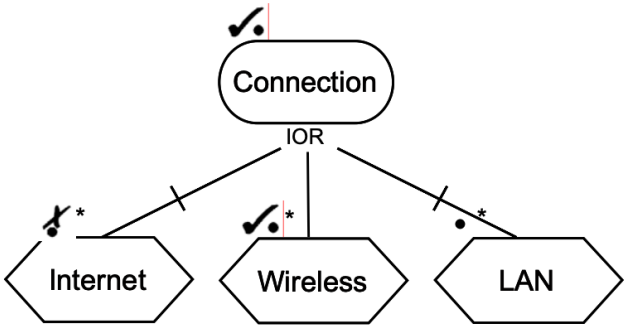


(d) Miminum is Denied, even if Conflict is present

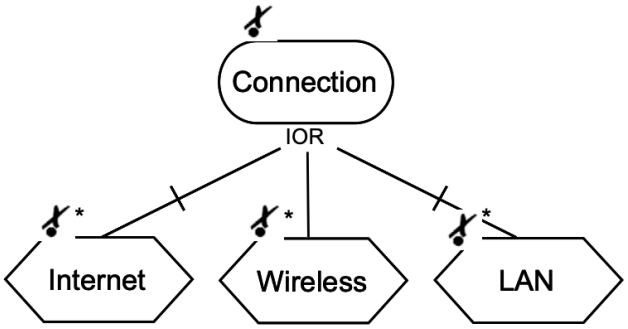
GRL Satisfaction Levels: (qualitative)

- Satisfied
- Weakly Satisfied
- Unknown
- Weakly Denied
- Denied
- Conflict
- None

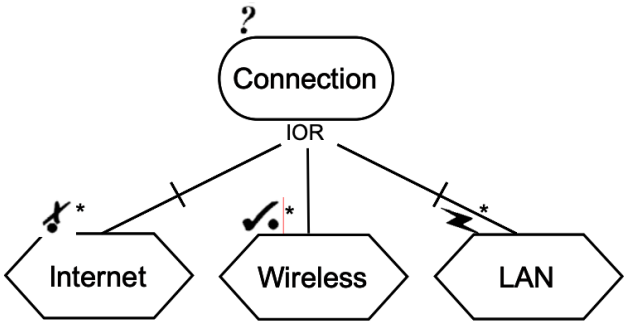
OR Decomposition: Satisfaction Levels (Qualitative Approach)



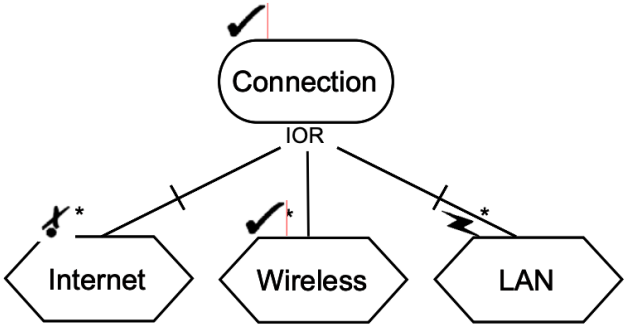
(a) Maximum is WeaklySatisfied



(b) Maximum is Denied



(c) Maximum is Conflict: Undecided is propagated



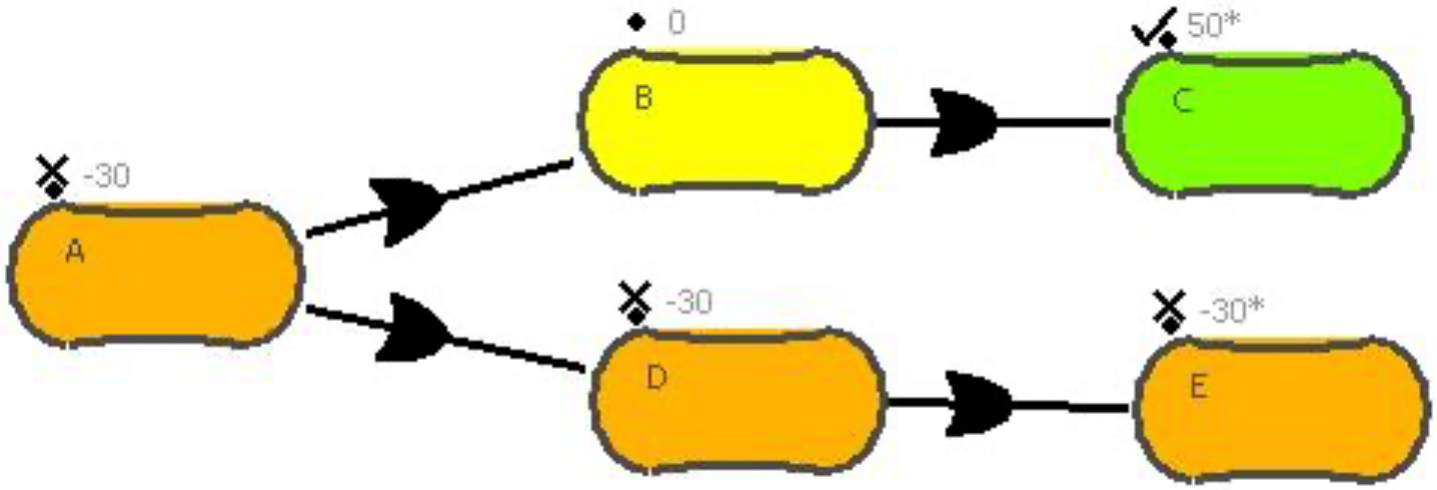
(d) Maximum is Satisfied, even if Conflict is present

GRL Satisfaction Levels: (qualitative)

- Satisfied
- Weakly Satisfied
- Unknown
- Weakly Denied
- Denied
- Conflict
- None

Numerical Evaluation: Dependencies

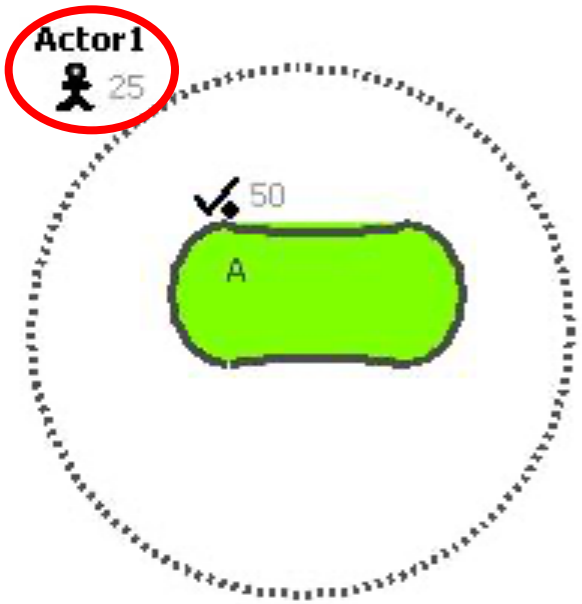
Intentional element evaluation is set to the minimal value in the set of dependees evaluation or it current evaluation

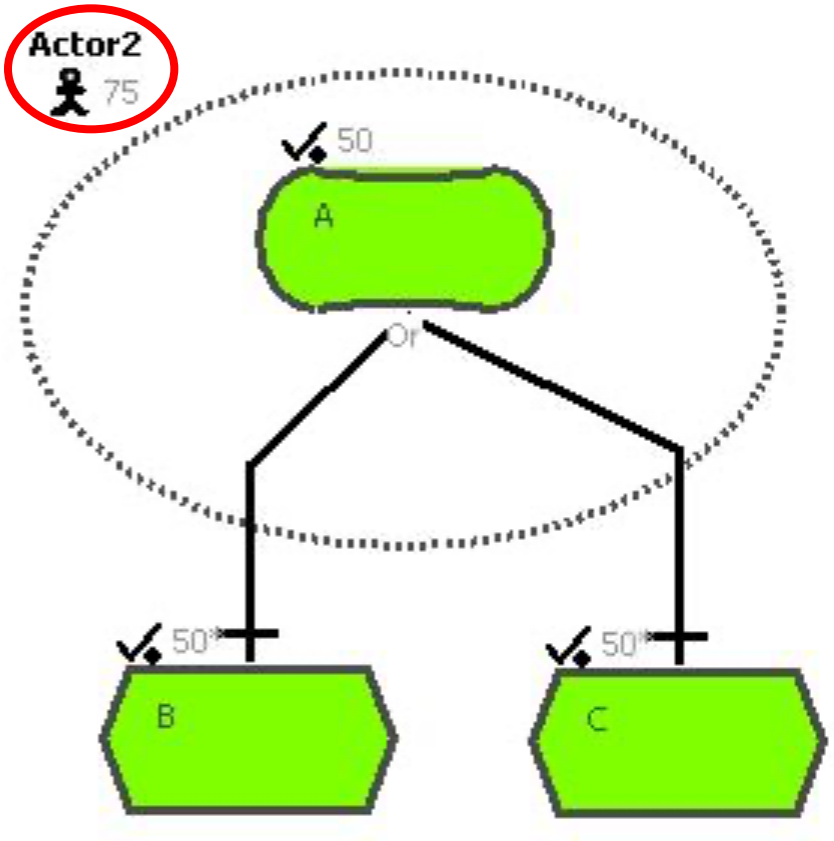


Actor Evaluation 

- Evaluations deal with negotiations between stakeholders
- Actor evaluations help analyzing and comparing the satisfaction levels of each actor based on the selected strategy
- Computed from **importance level** of intentional element references bound to actors

Numerical Evaluation: Actor Evaluation

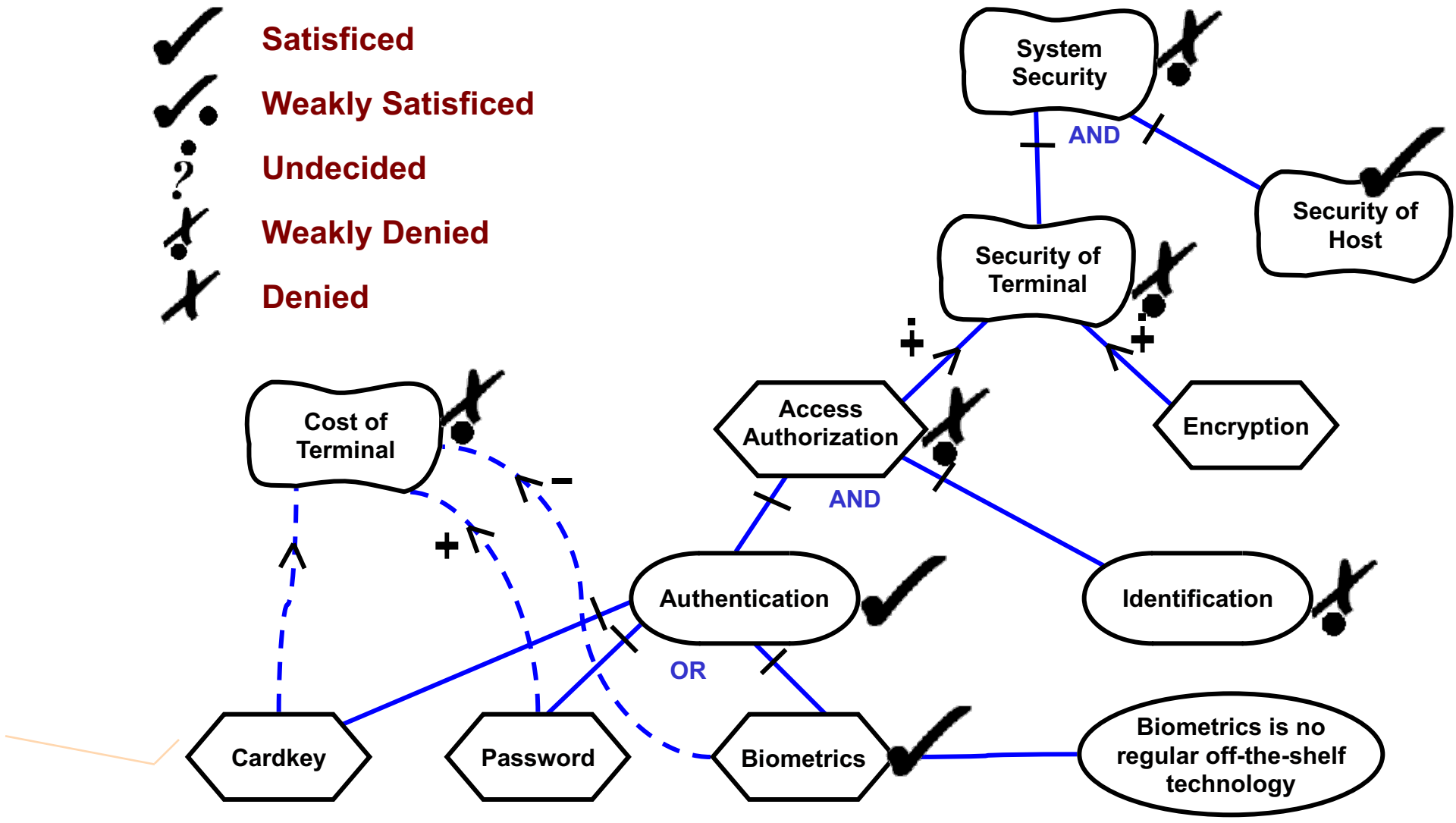




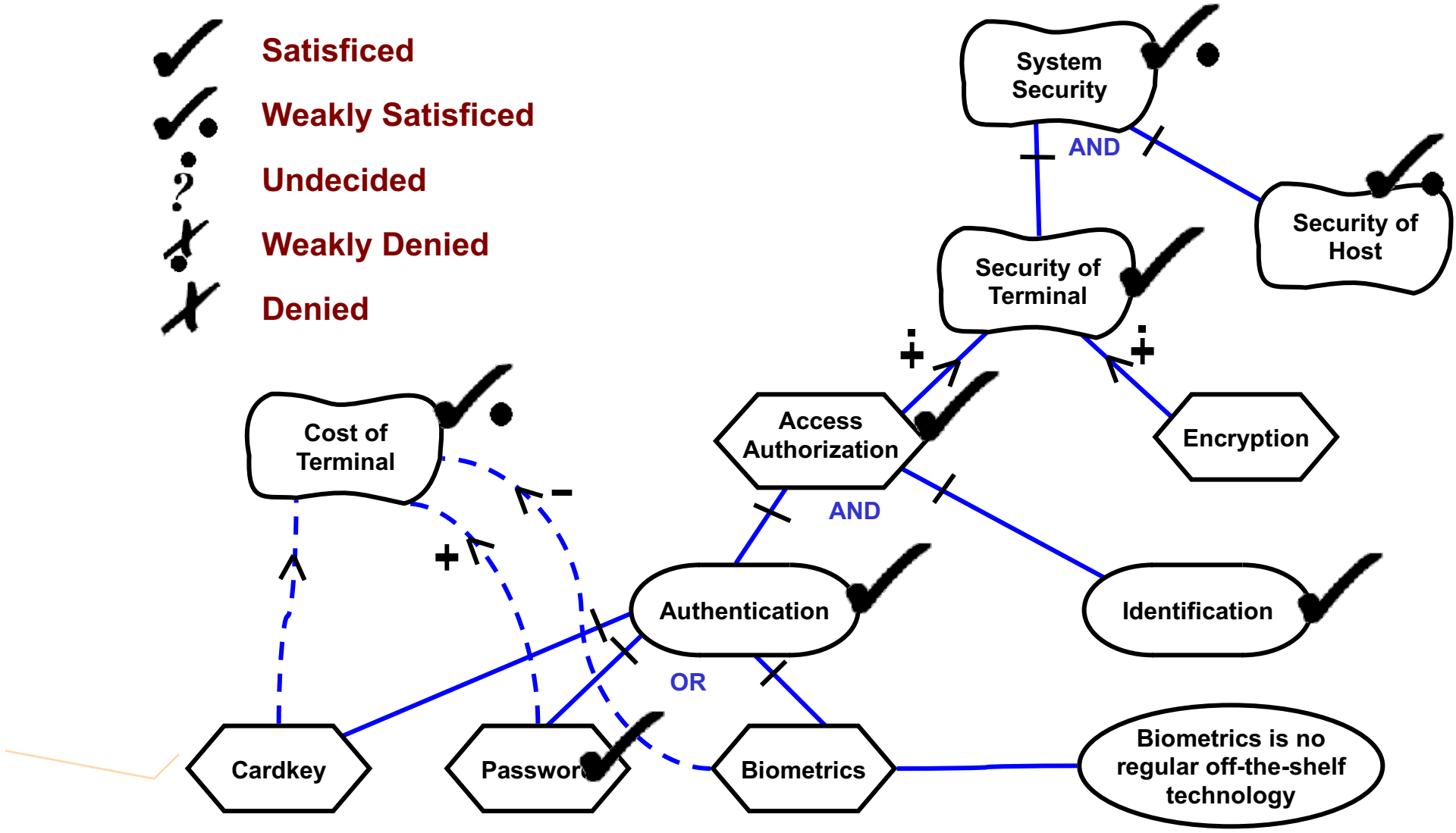
Priority = Low
Criticality = None

Priority = None
Criticality = High

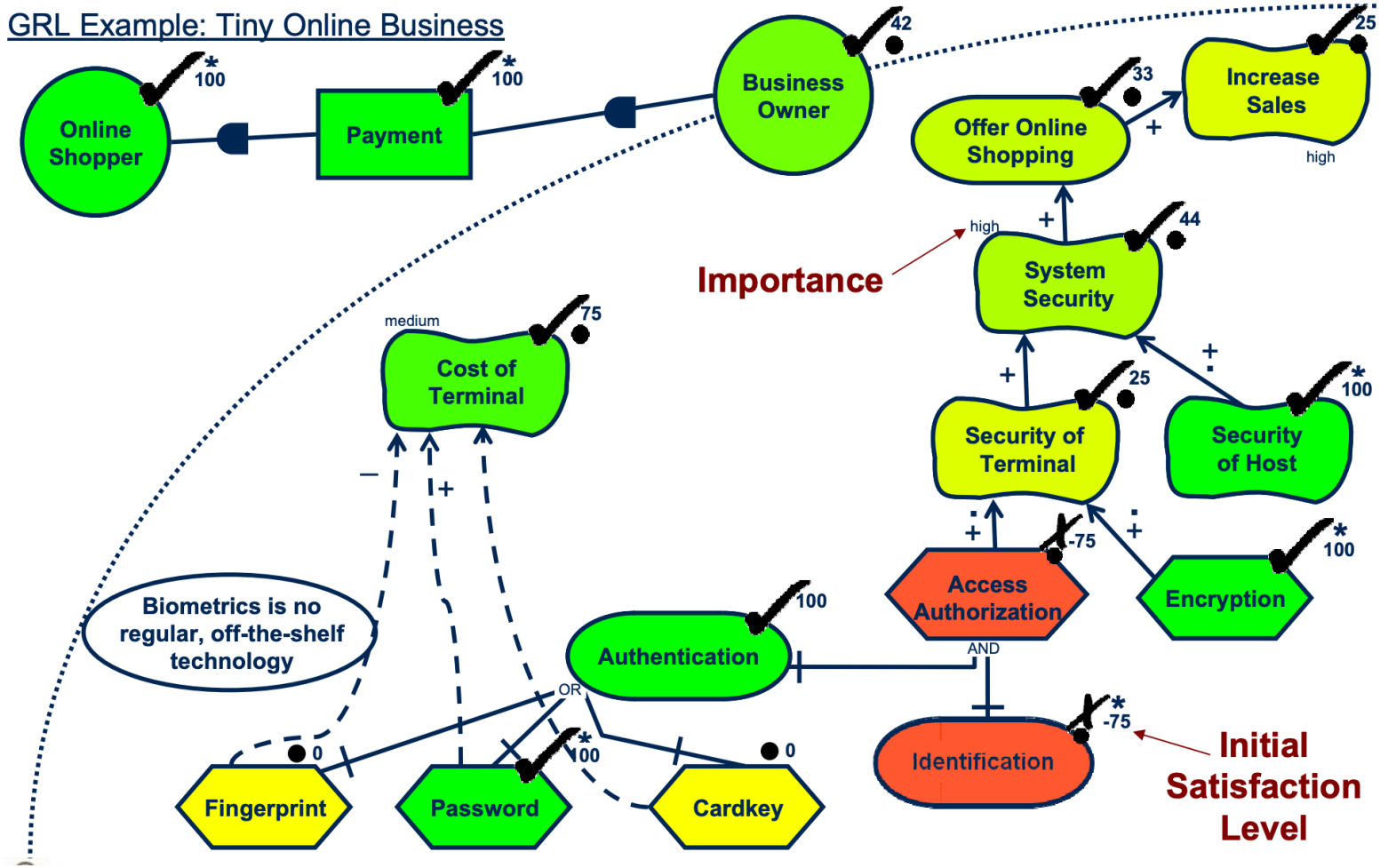
Example 1: Evaluations with GRL (strategy 1): Qualitative



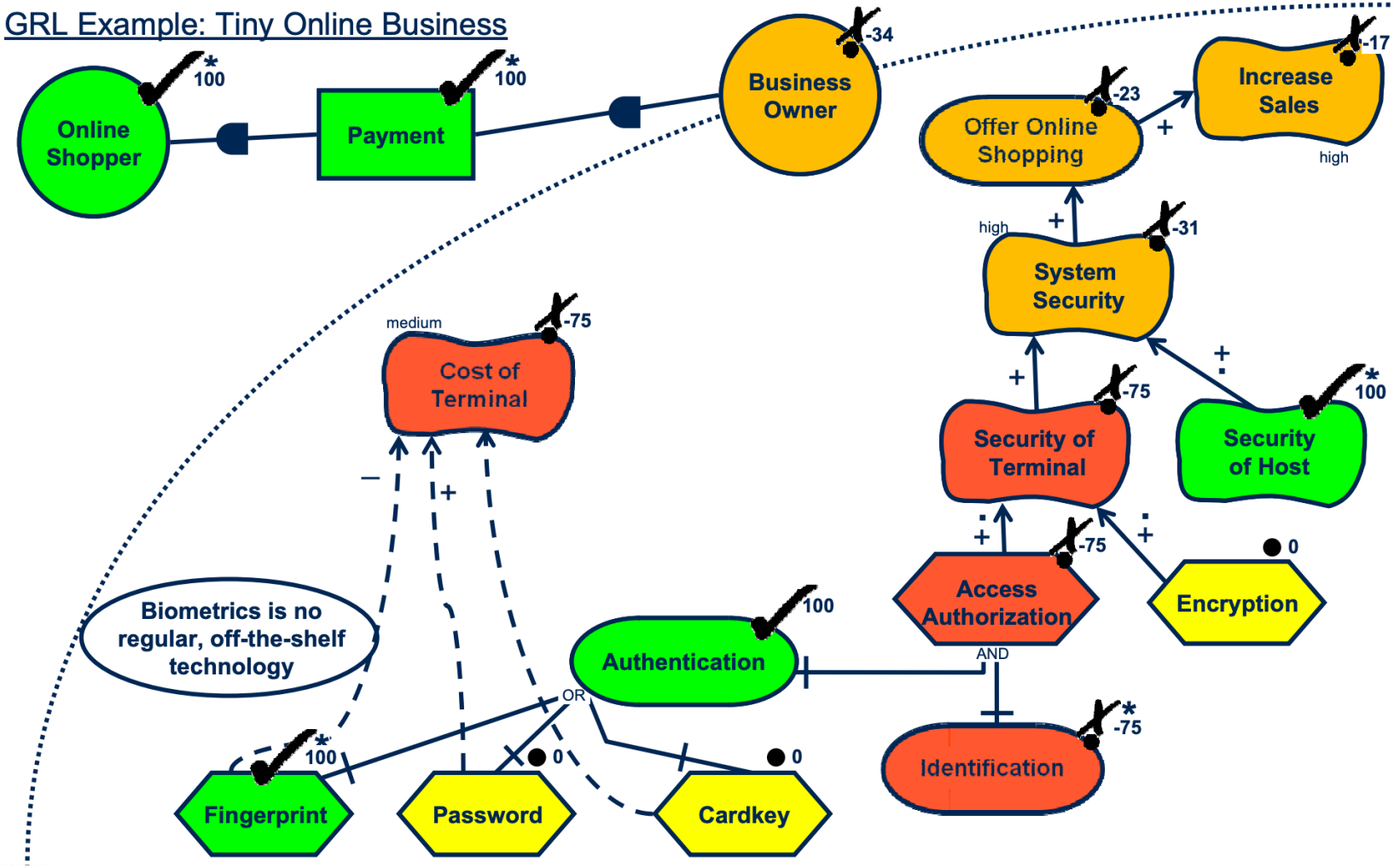
Example1: Evaluations with GRL (strategy 2): Qualitative



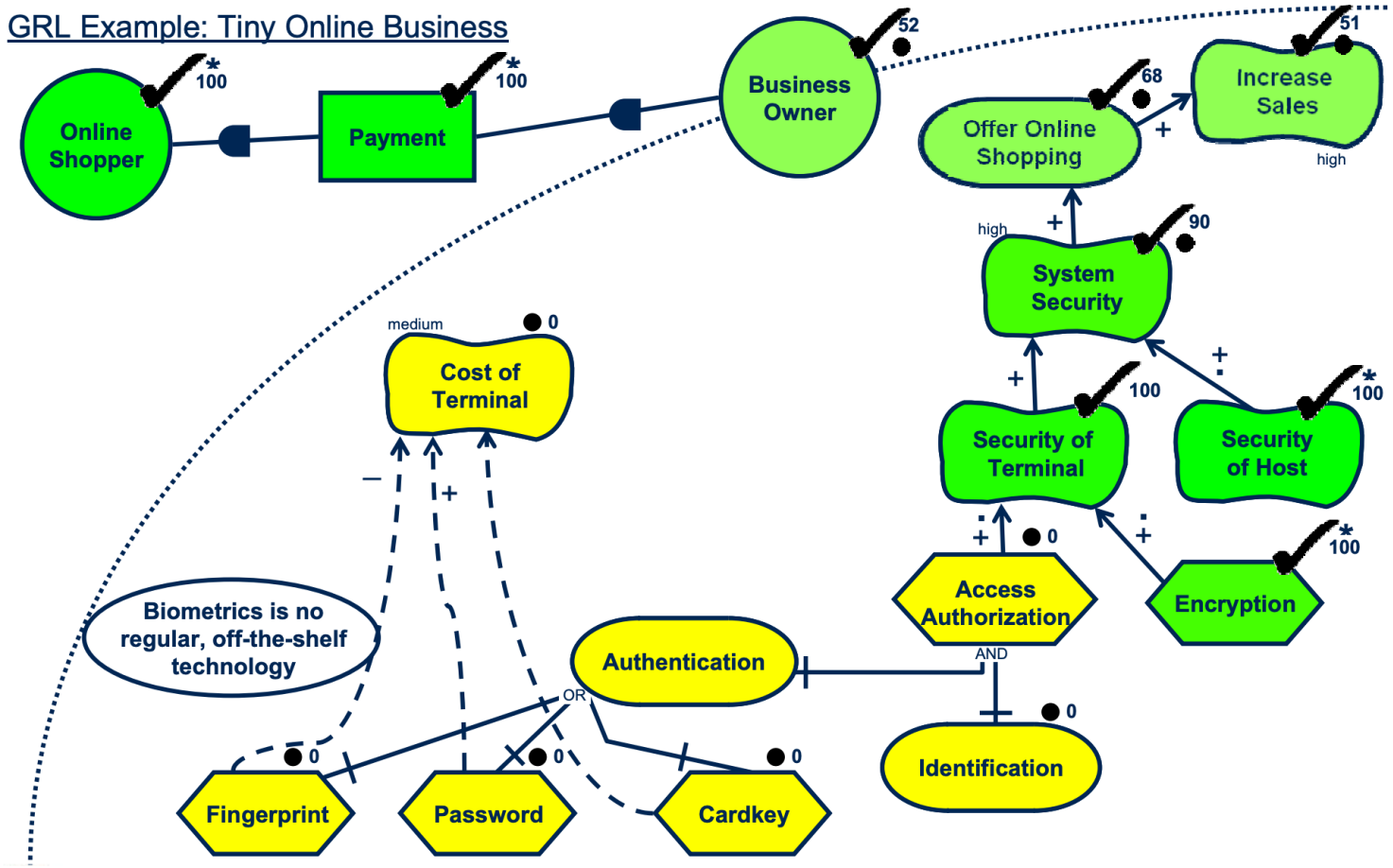
Example 2: Evaluations with GRL (strategy 1): Quantitative



Example 2: Evaluations with GRL (strategy 2): Quantitative



Example 2: Evaluations with GRL (strategy 3): Quantitative



Summary

- URN
  - Allows engineers to specify or discover requirements for proposed and evolving systems, and review such requirements for correctness and completeness.
  - Combines goals and scenarios
  - Helps bridging the gap between informal and formal concepts, and between requirements models and design models
  - Big benefits for little modeling investment, even when used informally
- GRL
  - For incomplete, tentative, (non-functional) requirements
  - Capture goals, objectives, alternatives and rationales
- UCM
  - For operational requirements and architectures
  - Enables analysis and transformations
  - Architectural alternatives and dynamic systems