Requirements Engineering

Lecture 5

Prof. Dr. Alexander Pretschner

Chair of Software & Systems Engineering
TUM Department of Informatics
Technical University of Munich

Orientation

Recap:

- Identifying Stakeholders
- Elicitation Techniques

Coming up:

→ How do I capture goals in a model and which techniques are available?

→ Which interactions exist between target models and further contents of the RE?

→ How do I use target models in the analysis?

→ What difficulties arise during target modeling?

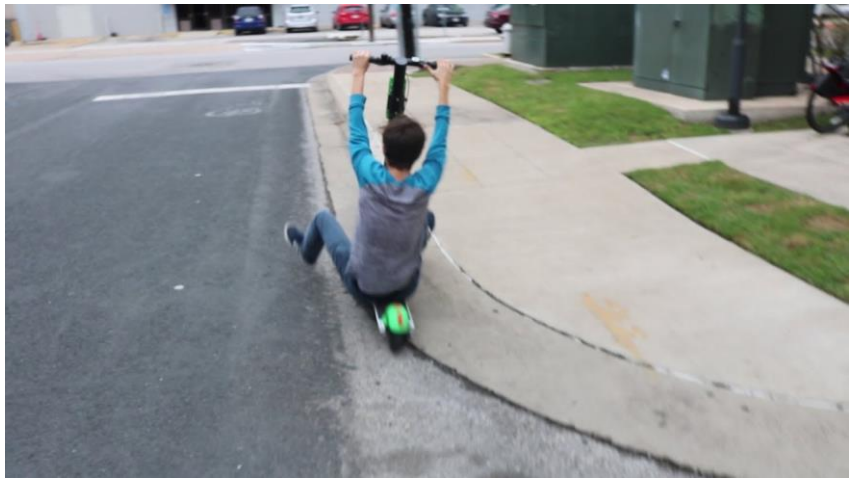
The RE problem

We want to design a rent-a-scooter system, what do we need?

- Scooters and an app to process payment and unlock the scooters!
- Users ride a scooter from one destination to another.
- A system to locate and pick-up scooters!
- Scooters need to be cheap, but durable.
- Make money!

https://www.heraldnet.com/news/lime-scooters-are-unofficially-temporarily-back-in-everett/

Scooters in the Real World



Requirements seem simple

… until you try to write them down.

- Project goals are unclear (just make money or improve transportation?).
- Stakeholders' priorities differ (Users don't want to return scooters).
- People make unspoken assumptions (Users are nice).
- People interpret meanings differently (A riverbed is technically a destination).
- Requirements often can't be verified (What is durable enough? fireproof?).

- => Goal Orientation can help to structure the process and makes assumptions, dependencies and conflicts explicit.

(Alexander, 2011)

*ORE to the rescue

To solve this problem, different school of thought exist:

- Stakeholder-oriented RE, or SORE (the soft systems methodology)
- Goal-oriented RE, or GORE (i*, KAOS, and so on)
- Scenario-oriented RE, or ScORE (use cases, user stories, and so on)
- Priority-oriented RE, or PORE (business cases, cost-benefit analyses, triage, and so on)
- Context/event-oriented RE, or CORE (context diagrams showing interfaces and events)

Goals

- A **goal** is an **objective** the system under consideration should achieve.
- Goals may be formulated at different **levels of abstraction**, ranging from high-level, strategic concerns to low-level, technical concerns.
- Goals also cover different **types of concerns**: functional concerns associated with the services to be provided, and non-functional concerns associated with quality of service -- such as safety, security, accuracy, performance, and so forth.
- Goals require the **cooperation** of different agents (e.g., humans, software, devices). In contrast to requirements, a goal may in general require the **cooperation** of a hybrid combination of multiple agents to achieve it.

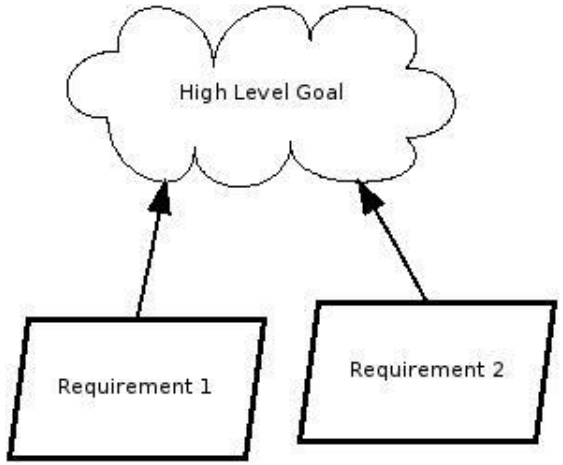
GORE

GORE focuses on activities that **precede** the formulation of systems requirements.
It aims to answer the question **why** a system is needed in the first place.
Purpose => Goal => Requirement
GORE views the system-to-be and its environment as a collection of ***agents***.
*Examples: humans, devices, software; in general, they are active components that have a **choice of behavior***

In GORE agents are responsible for achieving goals.
A ***requirement*** is a goal that is the responsibility of a single software agent.
[Idea: Goals are achieved by the cooperation/interplay of different functional units ("agent"), not one unit alone. But then "performance" in a distributed system is a "goal" because performance is achieved not by one but by all agents together, and hence cannot be a requirement. Um.]
An ***assumption*** is a goal delegated to a single agent *in the environment*.
Assumptions cannot be enforced by the system-to-be!

Benefits of GORE

Broader perspective than traditional RE.
Allows for **traceability** of requirements and their rationale.
Goals provide a precise criterion for sufficient completeness of a requirements specification.
A **goal refinement tree** provides traceability links from high-level strategic objectives to low-level technical requirements. These trees are the core idea behind GORE.
Goal modeling provides a natural mechanism for structuring complex requirements documents.
Goal models provide an excellent way to communicate requirements to customers.



Pros and cons of competing element-oriented requirements (xORE) methods.

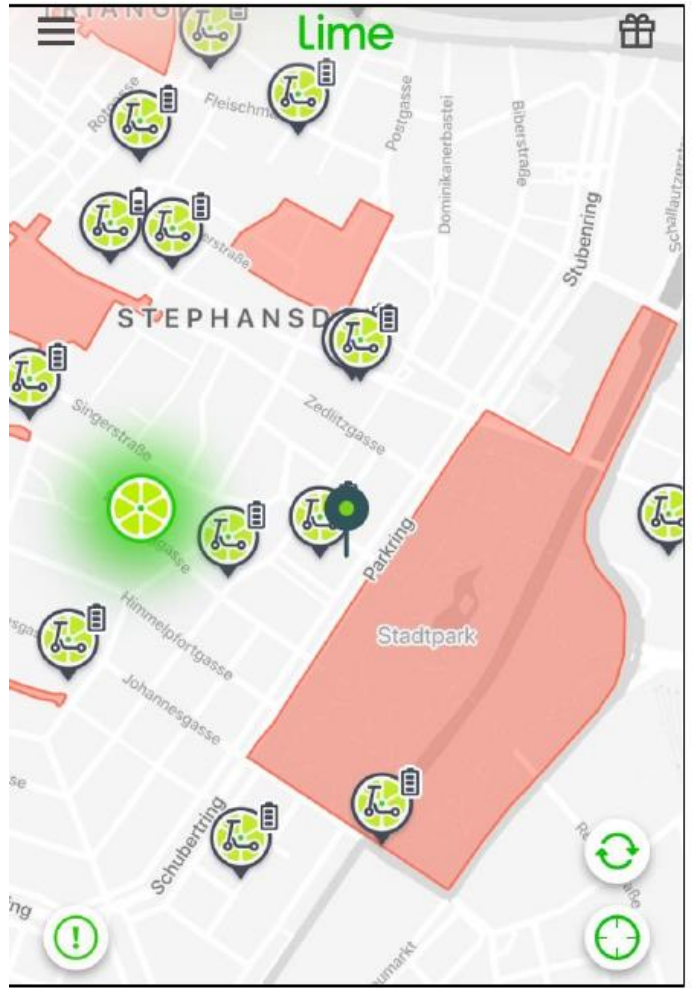
Element used in method	Advantages	Disadvantages	Examples of method (with sources)
Stakeholders (SORE) (See Lecture 4)	Identifies political, economic, social, and cultural forces that drive design	No precise, verifiable description of system requirements; unsuitable for contracts	Soft systems methodology (P. Checkland, *Soft Systems Methodology in Action*, Wiley, 1999)
Goals (GORE)	Says what stakeholders want; expresses both functions and desired qualities ("soft goals"); identifies goal conflicts and trade-offs	Hard to express timing relationships (scenarios) between goals; danger of premature design choices	KAOS (http://lamswww.epfl.ch/reference/goal); i* (www.cs.toronto.edu/km/istar/)
Context, interfaces, events (CORE)	Defines system boundaries explicitly; identifies events at interfaces, how to handle them	Assumes context well-defined: doesn't attend to soft systems issues or conflicting stakeholder goals	Event-driven methods (J.Z. Lavi and J. Kudish, *Systems Modeling & Requirements Specification Using ECSAM*, Dorset House, 2005)
Scenarios (ScORE) (See Lecture 7)	Says how design will deliver results to human operators; identifies procedures for normal and maintenance operations, as well as training and manuals	Overemphasis on product behavior; risk of ignoring nonfunctional requirements, nonoperational stakeholders; danger of premature user interaction design	Use cases (A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2001); agile/user stories (M. Cohn, *User Stories Applied for Agile Software Development*, Addison-Wesley, 2004)
Priorities (PORE)	Shows which design features are needed most or would be most profitable; reduces waste	Assumes requirements are independent, can be compared financially; may ignore context, purpose, qualities, constraints, nonfinancial beneficiaries	Business case; benefit-cost analysis; value engineering; triage (A.M. Davis, *Just Enough Requirements Management*, Dorset House, 2005)

Alexander, I. (2010).

In the scooter scenario

One goal might be a “good distribution of scooters throughout the city”.
This will require the cooperation of

- Users: incentivize taking scooters from fringe areas to central locations.
- Scooter tracking system: Identify areas and times where demand might surge (e.g., a big event; people getting to work).
- Scooter pick-up personnel: schedule pick-ups and charging efficiently.



Moran, M. E., Laa, B., & Emberger, G. (2020)

Assumption vs. Requirement

- A goal under responsibility of a single agent, i.e., active components such as humans, devices or software, in the system-to-be becomes a requirement whereas a goal under responsibility of a single agent in the environment of the software-to-be becomes an assumption.

- Unlike requirements, assumptions cannot be enforced by the software-to-be; they will hopefully be satisfied thanks to organizational norms and regulations, physical laws, etc.

- In our example: only *identifying* surge of demand is a requirement! The behavior of the users and the pick-up personal are beyond our direct control and can only be assumptions.

Advantages of goals

- Achieve completeness: A requirements specification is *sufficiently complete*, if we can show that all the goals can be achieved from the specification and properties of the domain. (cf. GQM)
- Avoid irrelevant requirements
- Explain requirements to stakeholders by linking them to goals.
- Goals can be refined; so we can structure complex requirements.
- They allow us to resolve conflicts between multiple viewpoints by making them explicit and take a decision.

- Make it possible to design and implement excitement factors for the user or customer (→)

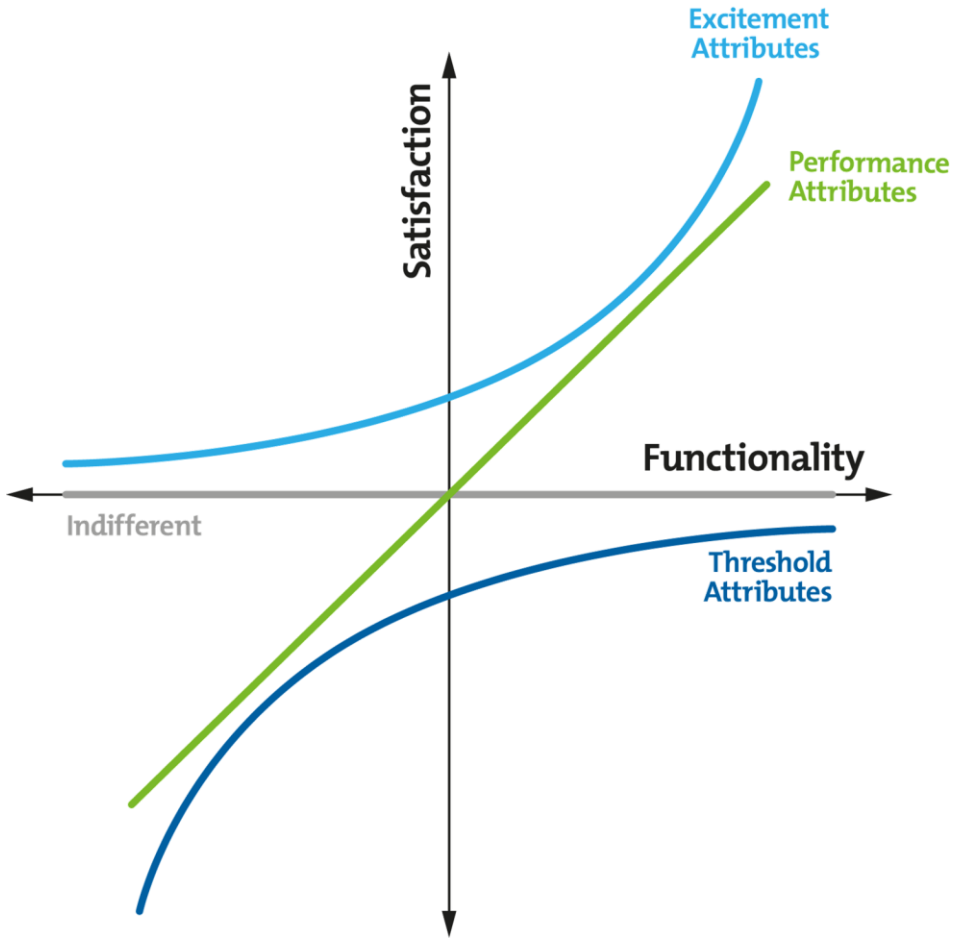
Goal of knowing the Goals

Kano-Model:

Threshold Attributes: Basic functionalities that the system must be able to provide. The customer is disappointed if they are missing.

Performance Attributes: They increase the customer's satisfaction but are not absolutely necessary

Excitement Attributes: Unexpected benefits contributing to the customer's goal, even if not perfectly implemented. Excitement attributes can only be implemented **if the customer's actual goal is understood!**



Hotel: clean / daily change of towels / chocolate on the pillow

Figure: https://www.mindtools.com/pages/article/newCT_97.htm

Where are goals coming from?

- Observations
- Explicitly stated by stakeholders.
- Analyzing the current system.
- Elicited from preliminary documents (The system *shall…)*
- *Design Thinking*

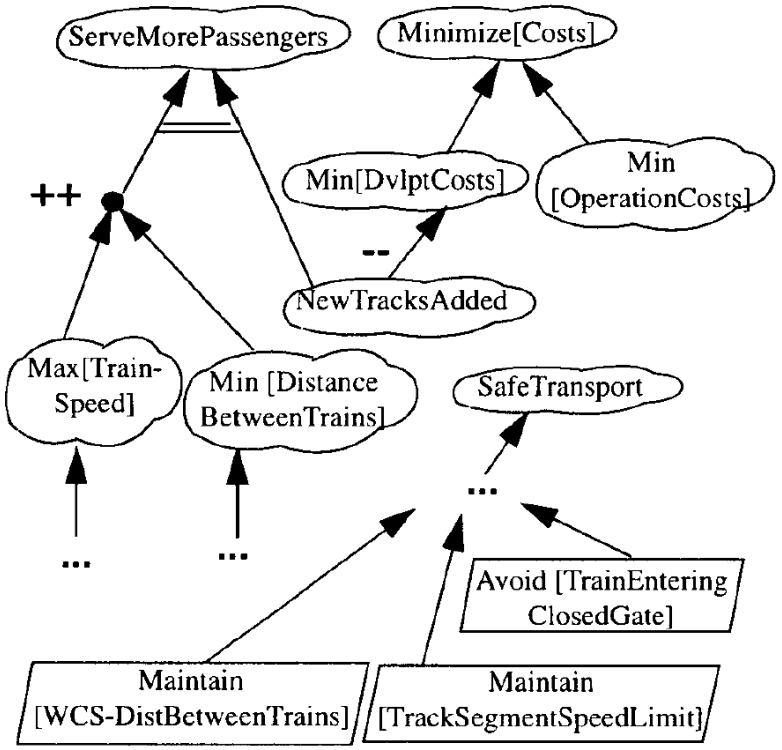
- Once we have an initial set of goals, new goals can be identified by refinement or abstraction (The system has users -> User data shall be kept safe and secure).
- Goal might often only be identified very late in the RE process, when we think about technical details; so GORE is NOT just top-down!

Modeling Goals

- **Functional** goals
  - e.g, 98% of users should have a scooter within walking-distance.
  - Can be divided into satisfaction and information goals. The first satisfy agent requests (e.g. locate a scooter), the second keep agent informed about object states (e.g. where are scooters in my vicinity?).
- **Non-functional** goals
  - e.g., The UI should be accessible for visually impaired users (e.g., red/greed blind, far-sighted people) who can use a scooter, but might have problems with a badly designed app.
- **Accuracy** goals ensure that the state of the software corresponds to the object in the environment (e.g., if the scooter app indicates a scooter is at some place, the scooter is actually there), Performance goals, Security goals (CIA triad).
- **Soft** goals (no clear-cut measure) vs. hard goals (can be verified)

Links between goals

- Goals might **positively** or **negatively support** each other
- Goal can have **sub-goals**
- **AND/OR** graphs
- Here clouds are so called “**soft goals**”, i.e., goals whose satisfaction cannot be established in a clear-cut sense; they are defined in contrast to **hard goals** whose satisfaction can be established through verification.
- We can develop these models **top-down or bottom up**.
- This model allows us to **trace** the **rational** for specific requirements.

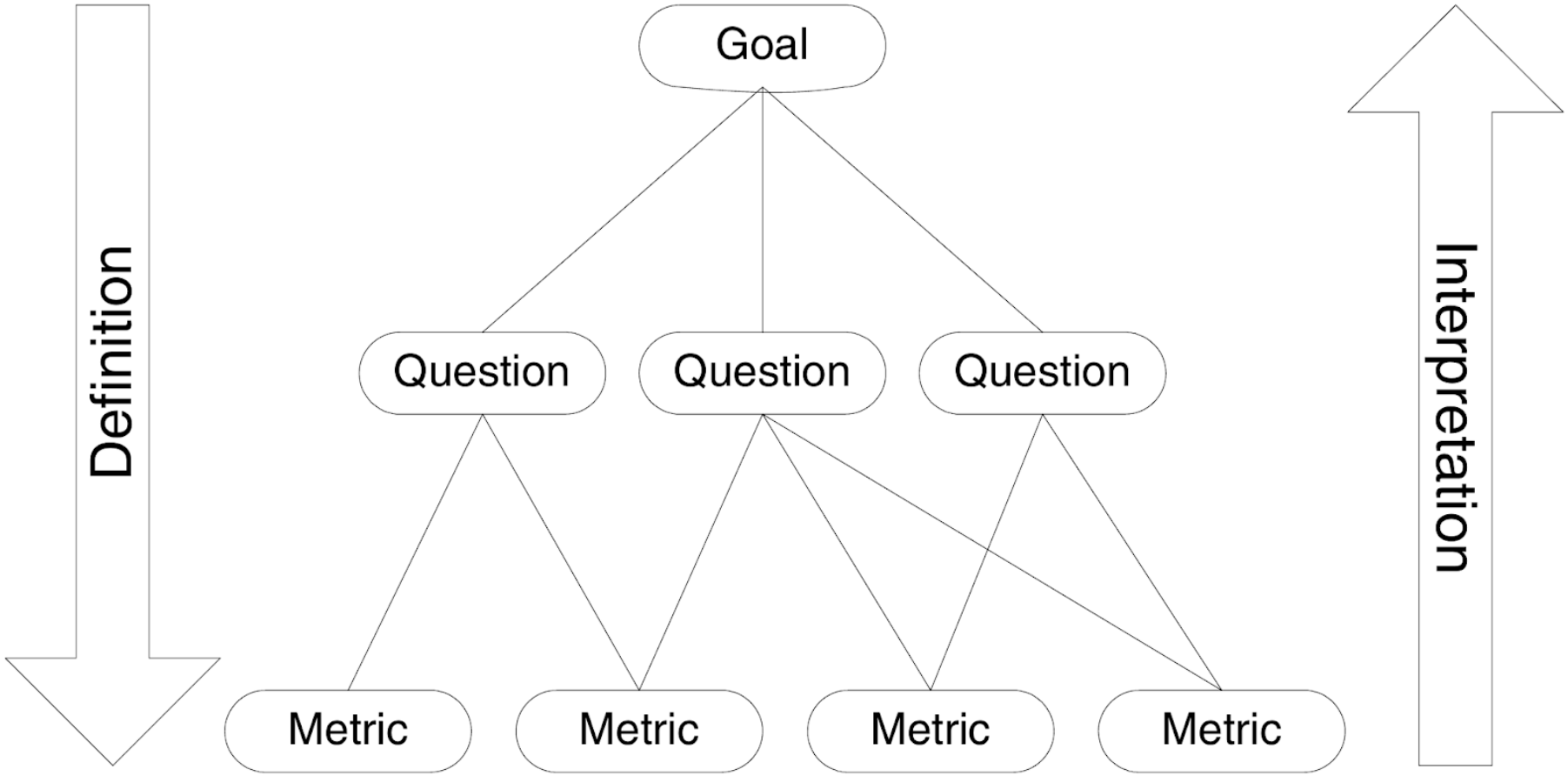


Lamsweerde (2001)

Goal Question Metric (GQM)

- Assumption: **to measure, you need a goal. To see if a goal is satisfied, you need to measure.**
- **Conceptual level (GOAL):** A goal is defined for an object, for a variety of reasons with respect to various models of quality, from various points of view, relative to a particular environment.
- **Operational level (QUESTION):** A set of questions is used to characterize the way *the assessment/achievement of a specific goal* is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.
- **Quantitative level (METRIC):** A set of data is associated with every question in order to answer it in a quantitative way. Metrics can be objective or subjective.

GQM


Definition
Goal
Question
Question
Question
Metric
Metric
Metric
Metric
Interpretation

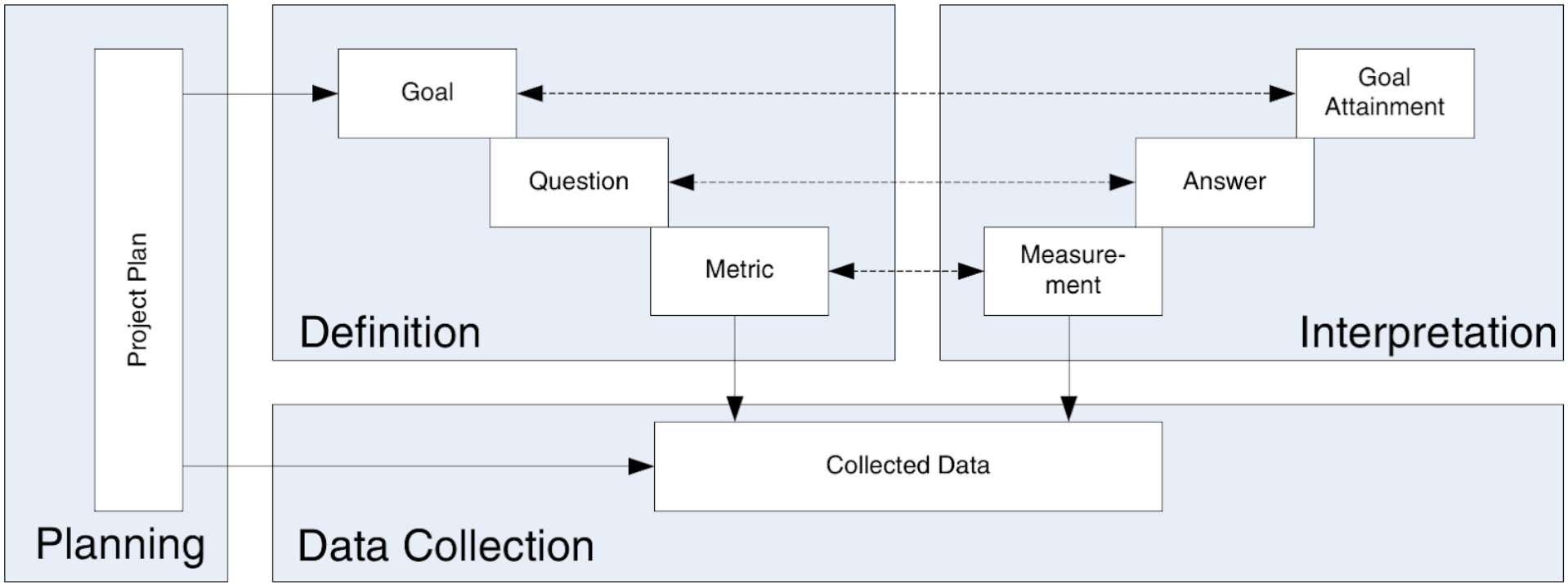
GQM Example

Goal	Purpose	Improve
	Issue	the reliability
	Object	of product X
	Viewpoint	for the user
	Context	within organisation Y
Question	Q1	What is the probability of failure on demand for function Z?
Metric	M1	POFOD
Question	Q2	How erroneous does function Z behave?
Metric	M2	MTTF
Metric	M3	MTTR
...	...	...

Example for GQM

Goal	**Purpose** **Issue** **Object** **Viewpoint** **Context**	**Improve** **the availability** **of the scooter** **for the user** **when picking up a scooter.**
Question Metric	Q1 M1	How long do the brakes last? Mean-Time-To-Failure: 100km (~2 years)
Question Metric Metric	Q2 M2 M3	How resistant is the frame of the scooter? Probability of Failure on Demand: 5% Mean-Time-To-Repair: 5 days
Question Metric	Q3 M4	How often are scooters vandalized? Probability: 5% (10% on a weekend)

Measuring Goals - Goal Question Metric (GQM)



Measuring Goals

- **Planning** Phase: decide upon an area of improvement
  - e.g., number of requests served.
- **Definition** Phase: define measurement goals
  - e.g., Latency at peak load of the scooter app.
- **Data Collection** Phase: gather data on the goal
  - e.g., extend the app with analytics.
- **Interpretation** Phase: Process data to get measurement results and determine answers to the questions
  - e.g., parse the data and get concrete profiled data.

Keep All Objects Satisfied (KAOS)

Problem: How should we now **express and model goals** and their **connection to requirements**?

KAOS is one GORE methodology that supports **formal analysis**. It allows us to **visualize goals** and their **dependencies** and offers us formal semantics to **detect conflicts.**

Think of it as **UML for requirements**.

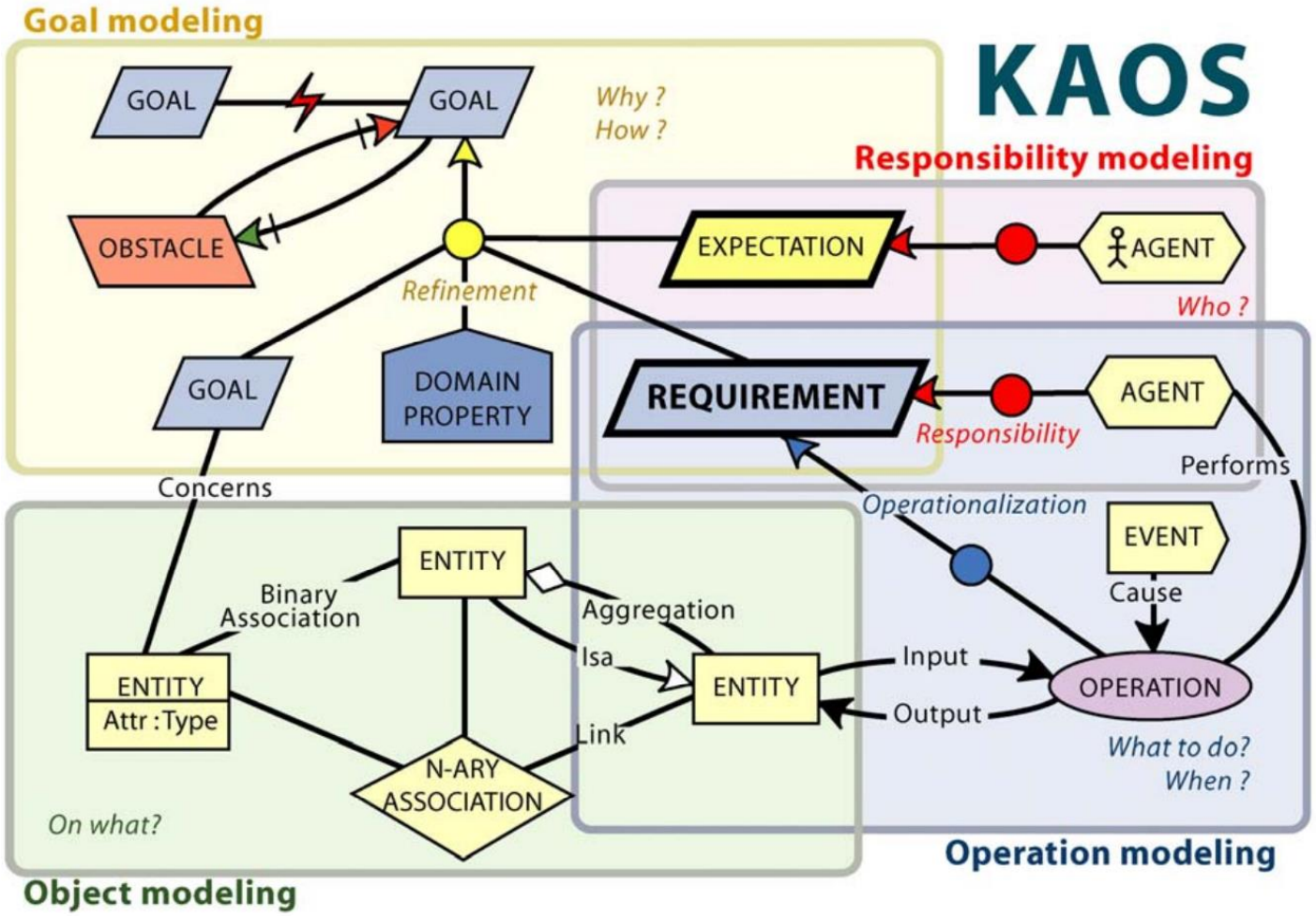
Alternatives:

- The NFR framework focuses on non-functional requirements
- i*/Tropos is an agent-oriented modeling framework.
- GBRAM (Goal-Based Requirements Analysis Method) emphasizes the initial identification and abstraction of goals from various sources of information.

KAOS ontology

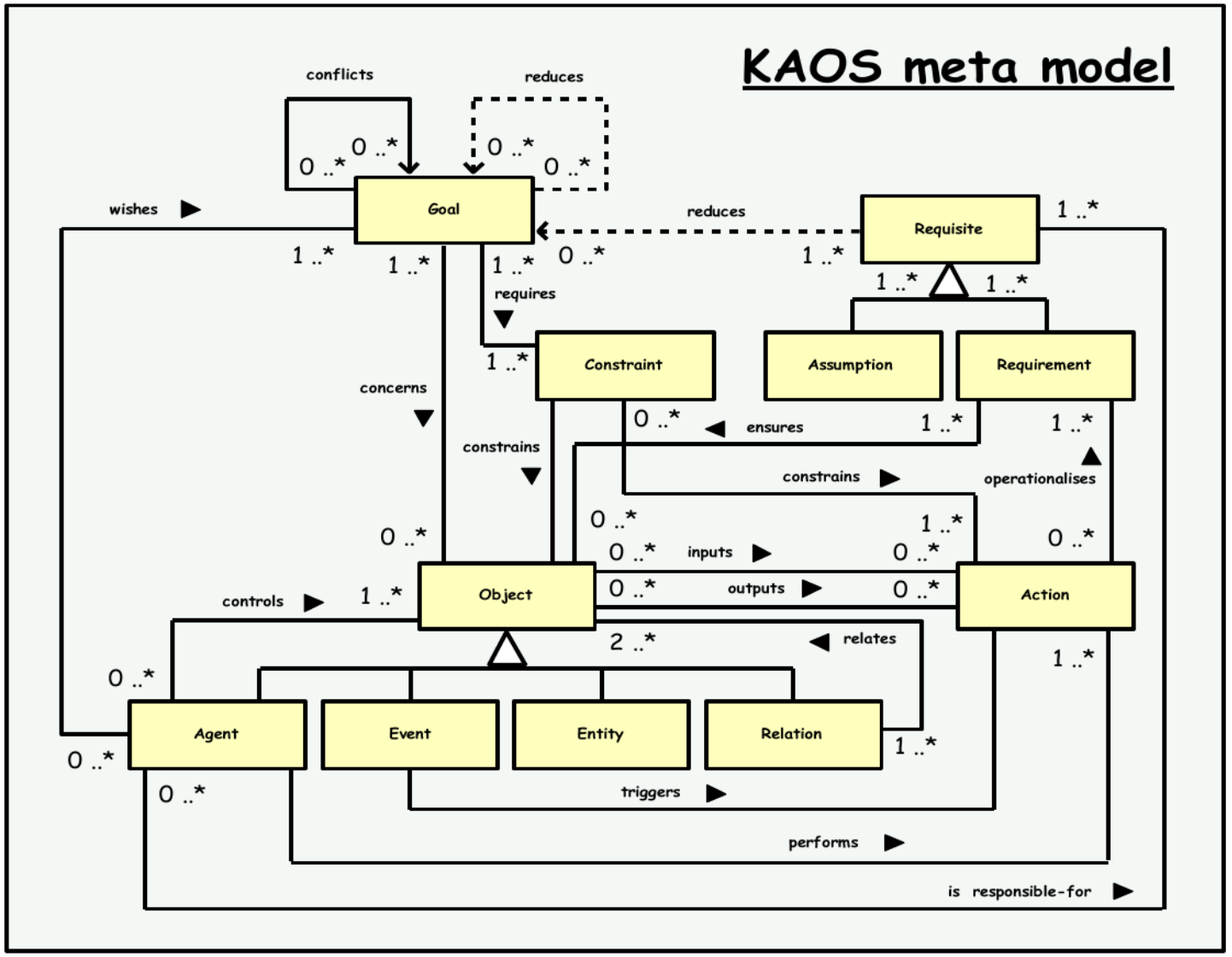
- **Objects** are things of interest in the composite system whose instances may evolve from state to state. Objects can be entities, relationships, or events.
- **Operations** are input-output relations over objects. Operation applications define state transitions. Operations are declared by signatures over objects and have pre-, post-, and trigger conditions.
- **Agents** are objects that acts as a processor for operations (e.g.: humans, devices, software, etc.) Agents perform operations that are allocated to them.
- **Goals** are “prescriptive statement[s] of intent about some system whose satisfaction in general requires the cooperation of some of the agents forming that system”.
  Goals are organized in **AND/OR refinement-abstraction hierarchies.**

Formalizing GORE



http://www.objectiver.com/fileadmin/download/documents/generalleaflet.pdf

KAOS Meta Model



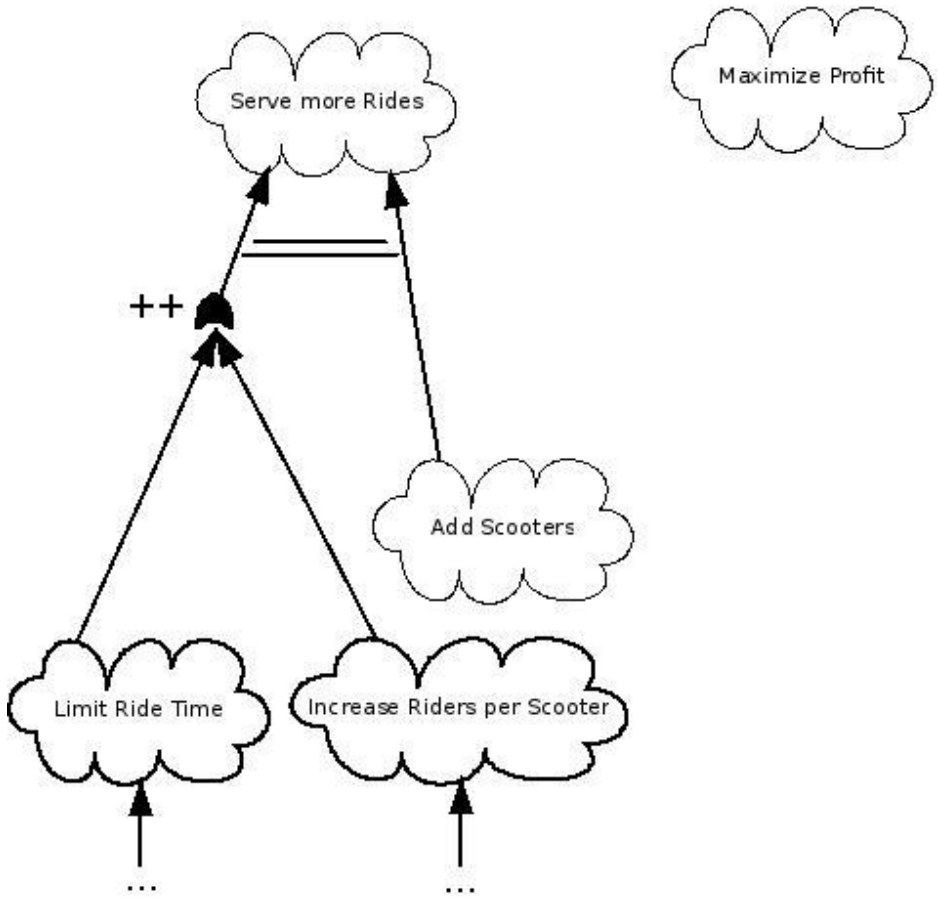
Werneck et al. (2009)

Two goals

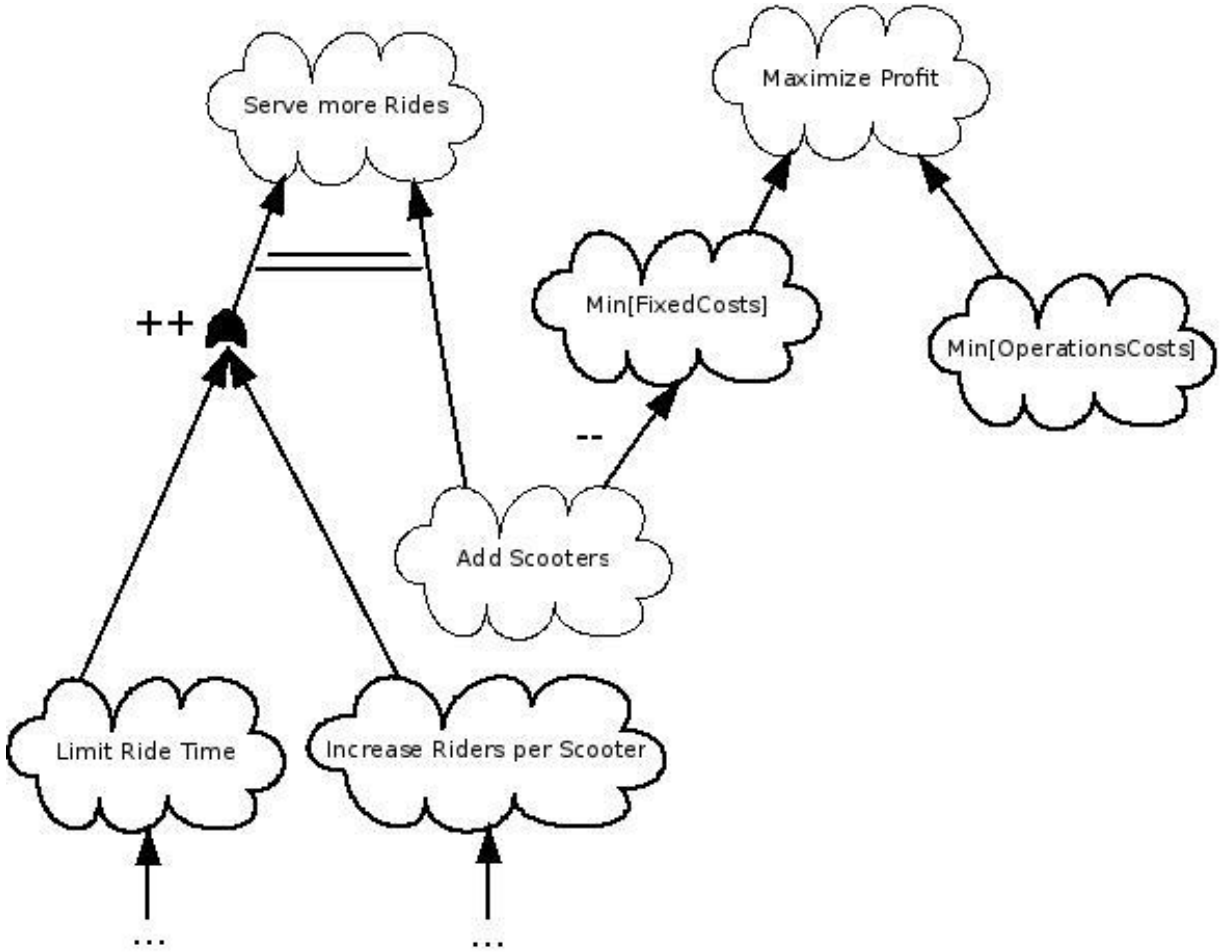
Two goals

Conquering a market may mean not to make profit for a long time!

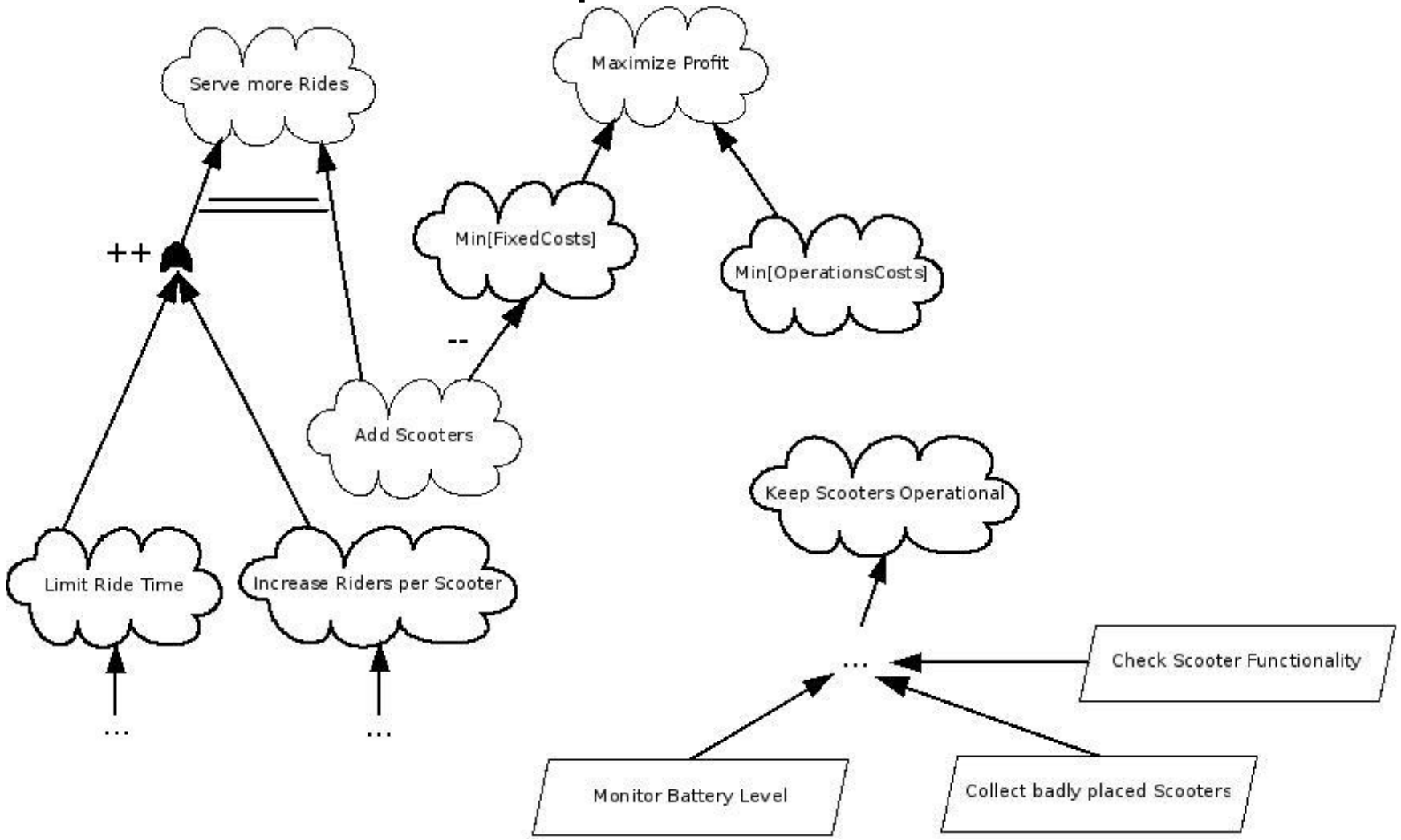
Different ways to increase the number of riders



Adding scooters, however, might hurt profit

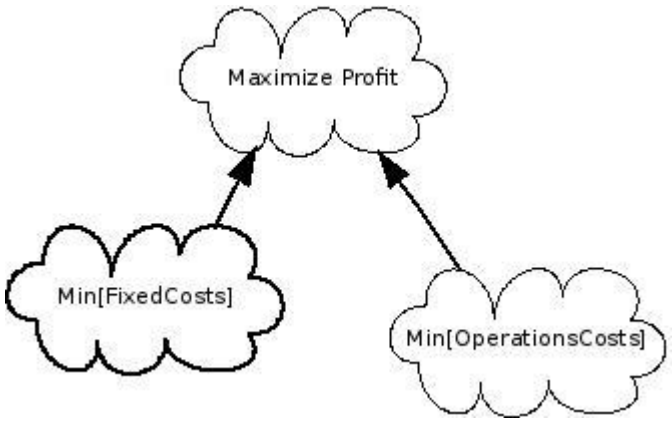


Another goal and some hard requirements



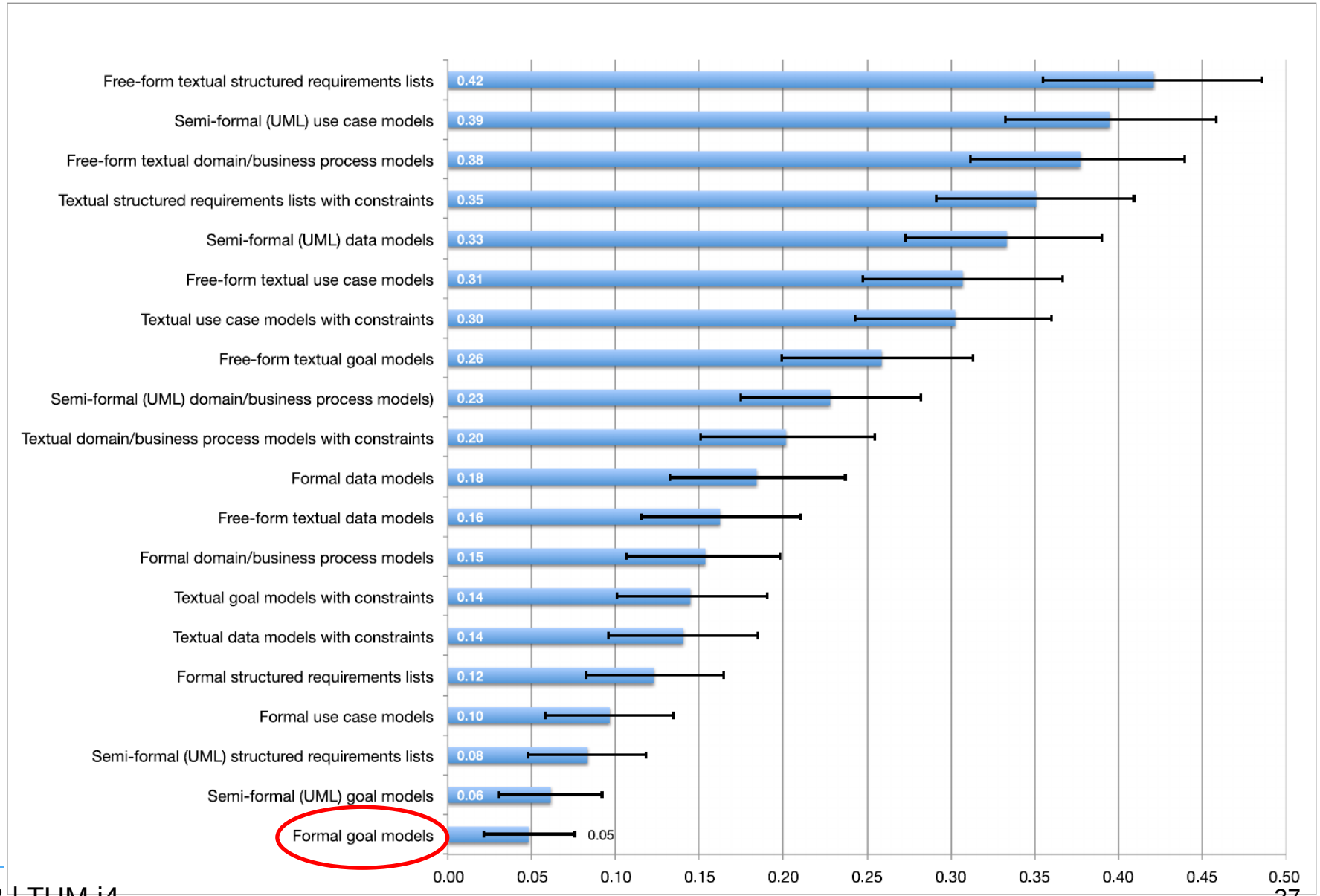
How to resolve Conflicts?

- Goal: conquer market (serve more rides) and make profit
- Minimizing the fixed cost might mean to buy cheaper scooters.
- However, operations costs of more expensive scooters might be lower.
- Some scooters might be easier to maintain, because you have people in the company that already know how to work with them.
- => and many more possible conflicts!



- The goal model can show that there is a conflict, i.e., **make it explicit!**
- The model **on its own cannot resolve it**. Here we need to talk to stakeholders etc.

*ORE is, however, not used in practice ...



Wagner et al (2019)

Summary

- **Goals** are a very useful concept to **structure** requirements and **trace** their origin.
- The **Kano Method** helps us generate goals.
- **GORE** approaches **formalize** this and aid us in making **conflicts** and dependencies **explicit.**
- **GQMs** helps link goals with means to measure whether they are reached or not.
- **Goals as concretizations/formalizations of needs/problems IMO very convincing**
- Conceptually hence very useful; in practice, **GORE** approaches rarely used.

Outline and Outlook

Terms and Definitions
Core activities
Quality models for Requirements
Engineering Models
Stakeholders and Elicitation Techniques

Goals and Goal-oriented RE

Non-functional requirements
Functional requirements
Formalization
Agile Processes
Requirements Management and Quality Assurance
Trends in Research

Further reading

Alexander, I. (2010). GORE, SORE, or what?. *IEEE software*, *28*(1), 8-10.

Basili V, Caldiera G & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, 528-532.

Heaven, W., & Finkelstein, A. (2004). UML profile to support requirements engineering with KAOS. *IEE Proceedings-Software*, *151*(1), 10-27.

Koziolek, H. (2008). Goal, question, metric. In *Dependability metrics* (pp. 39-42). Springer, Berlin, Heidelberg.

Lapouchnian, A. (2005). Goal-oriented requirements engineering: An overview of the current research. *University of Toronto*, *32*.

Van Lamsweerde, A. (2001, August). Goal-oriented requirements engineering: A guided tour. In *Proceedings fifth ieee international symposium on requirements engineering* (pp. 249-262). IEEE.

Van Lamsweerde, A. (2004, May). Elaborating security requirements by construction of intentional anti-models. In *Proceedings. 26th International Conference on Software Engineering* (pp. 148-157). IEEE.

Werneck, V. M. B., Oliveira, A. D. P. A., & do Prado Leite, J. C. S. (2009, July). Comparing GORE Frameworks: i-star and KAOS. In *WER*.

www.objectiver.com

Moran, M. E., Laa, B., & Emberger, G. (2020). Six scooter operators, six maps: Spatial coverage and regulation of micromobility in Vienna, Austria. *Case Studies on Transport Policy*, *8*(2), 658-671.

Wagner, S., Fernández, D. M., Felderer, M.,et al. (2019). Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *28*(2), 1-48.

https://en.wikipedia.org/wiki/Kano_model