CAiSE 2014

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Identifying modularity improvement opportunities in goal-oriented requirements models

Catarina Gralha, Miguel Goulão, **João Araújo**

CITI, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal
acg.almeida@campus.fct.unl.pt
{mgoul, joao.araujo}@fct.unl.pt

Roadmap


Introduction
*i** framework
Metrics
Evaluation
Conclusions

Introduction

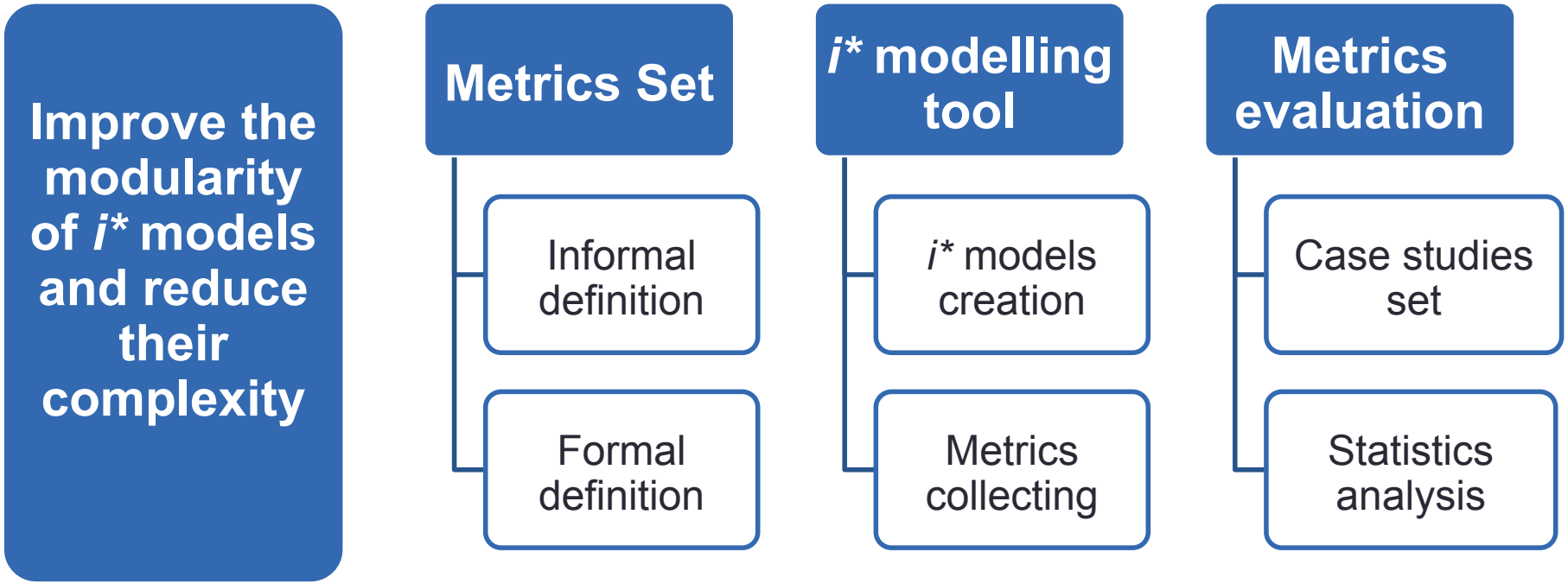
Introduction

- Goal-oriented Requirements Engineering (GORE)
  - Great impact and importance in the Requirements Engineering community
  - Provide expressive model elements for requirements elicitation and analysis
  - ***i****, KAOS, GRL
- The models can quickly become very complex
- Manage the accidental complexity of the models is a challenge
- Identify refactoring opportunities to improve the modularity of those models, and consequently reduce their complexity

Objectives

- To provide a tool supported metrics suite, targeted to the measurement and analysis of the complexity of *i** models, for identifying modularity refactoring opportunities
- The identification of such opportunities can be useful during development, where a better modularization can lead to a sounder distribution of responsibilities among the system components
  - If performed in a timely fashion, this is likely to contribute to relevant costs savings through the reduction of the model's accidental complexity
  - Refactoring opportunities identification is also an asset in the context of preventive maintenance, as a facilitator for future requirements changes

The Approach



About the metrics suite

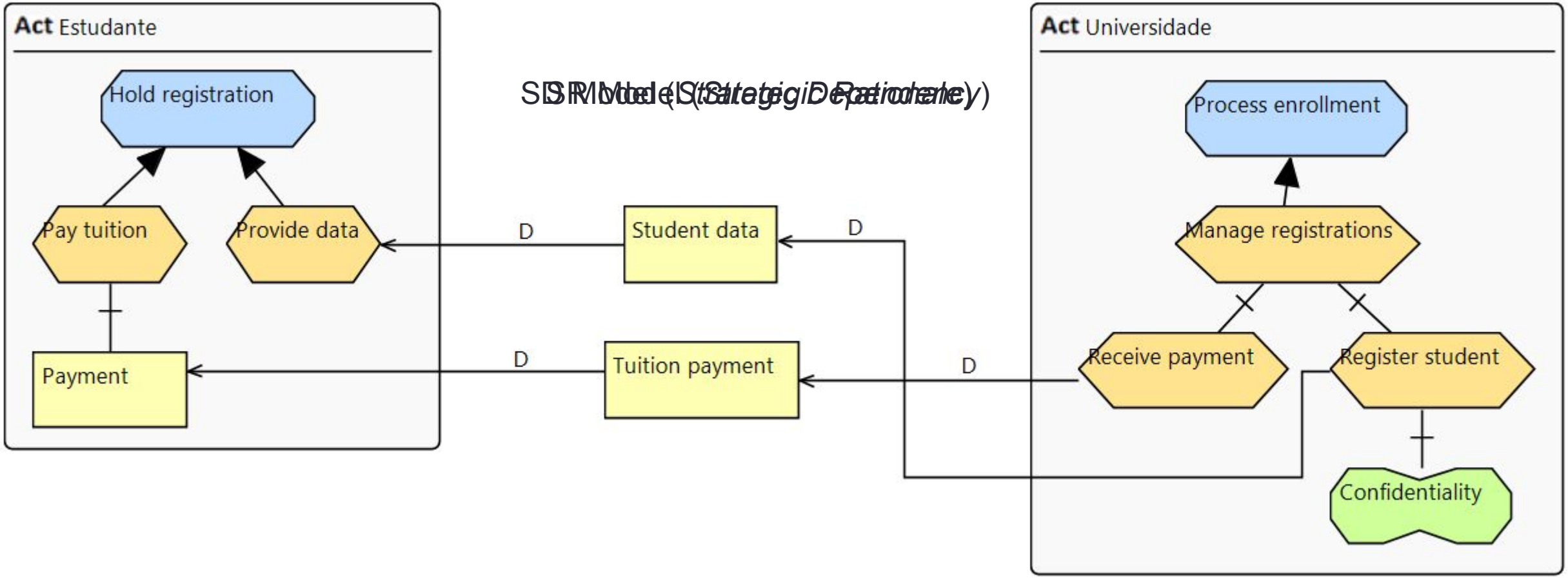
- The metrics suite is integrated in an eclipse-based *i** editor, so that metrics can be computed during the requirements modelling process, whenever the requirements engineer requests them
- The metrics are defined using the Object Constraint Language (OCL) upon the *i** metamodel
- This makes our metrics set easily extensible, as improving the metrics set can be done by adding new OCL metrics definitions
- Actor's boundaries are a key mechanism in the metrics suite proposed here. Our goal is to use these metrics to leverage the modularity of *i** models.

*i** Framework

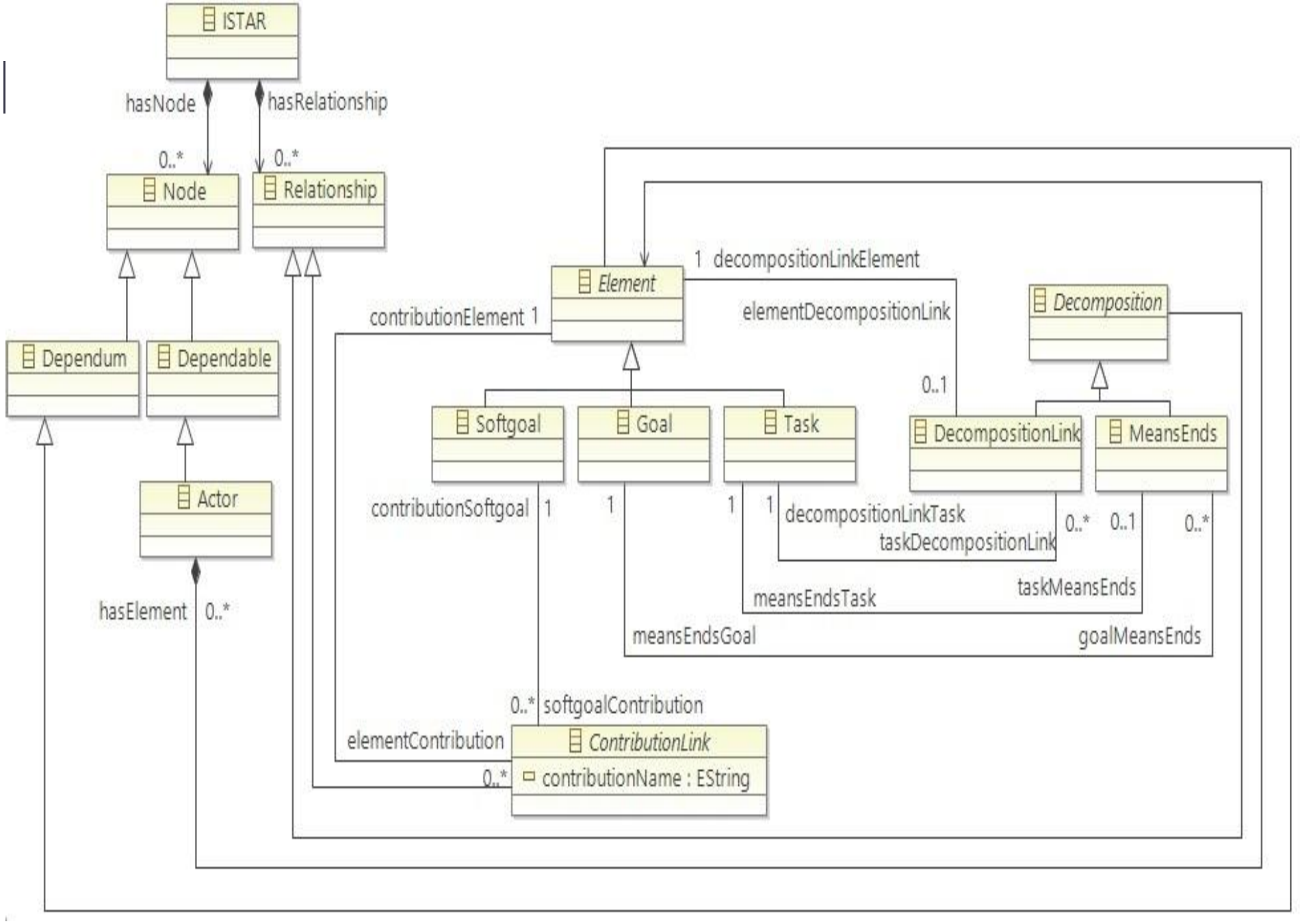
*i** Framework

Approach focused on the system stakeholders and in their relations

Developed for modelling and reasoning about organizational environments

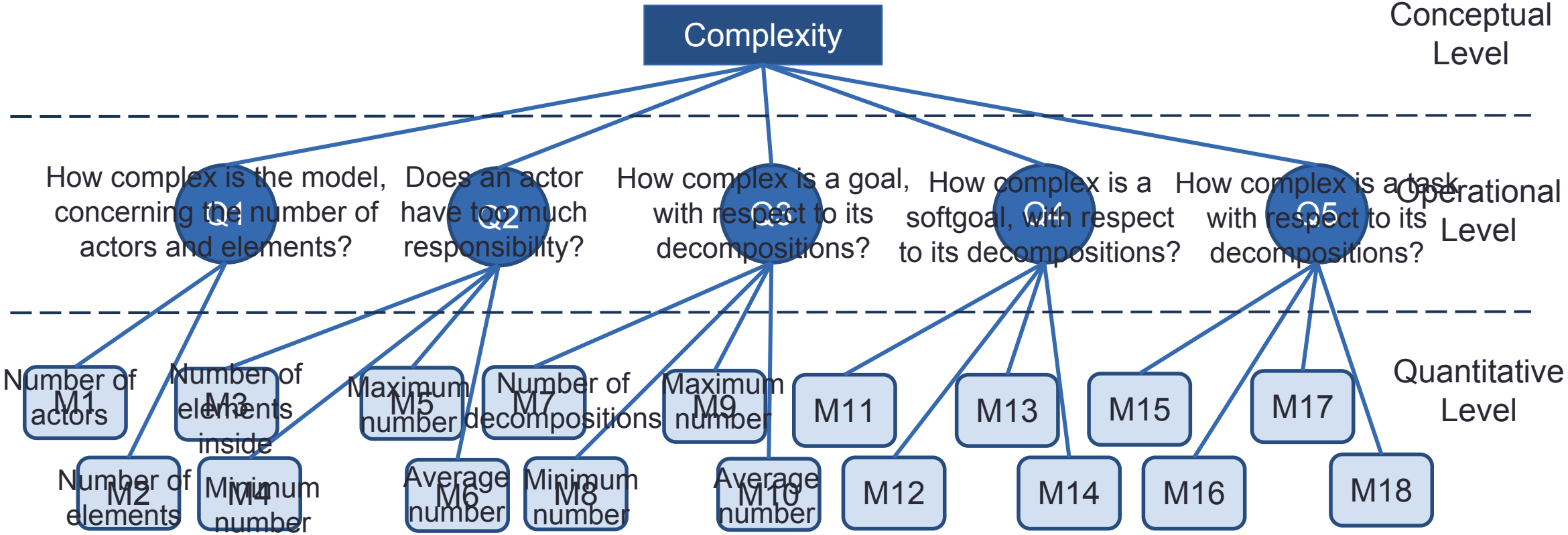


i* Metamodel


ISTAR
hasNode
hasRelationship
0..*
0..*
Node
Relationship
Dependum
Dependable
Actor
hasElement
0..*
Element
1 decompositionLinkElement
contributionElement 1
elementDecompositionLink
0..1
Decomposition
Softgoal
Goal
Task
DecompositionLink
MeansEnds
contributionSoftgoal
1
1
1
1
decompositionLinkTask
taskDecompositionLink
0..*
0..1
0..*
meansEndsTask
taskMeansEnds
meansEndsGoal
goalMeansEnds
0..*
softgoalContribution
elementContribution
ContributionLink
0..*
contributionName : EString

Metrics

Complexity Metrics Definition



Metrics Definition for Q1

Q1 - How complex is the model, concerning the number of actors and elements?	
Name	**NAct** – ***N**umber of **Act**ors*
Informal definition	Number of actors in the SD/SR model

Formal definition	`context ISTAR`
	`def:NAct():Integer = self.hasNode ->`
	`select(n:Node | n.oclIsKindOf(Actor)) -> size()`

Name	**NElem** – ***N**umber of **Elem**ents*
Informal definition	Number of elements in the SD/SR model
Formal definition	`context ISTAR` `def:NElem():Integer = self.NEOAB() + self.NEIAB()`
Requires	**NEOAB** – *Number of Elements Outside Actors' Boundary* **NEIAB** – *Number of Elements Inside Actors' Boundary*

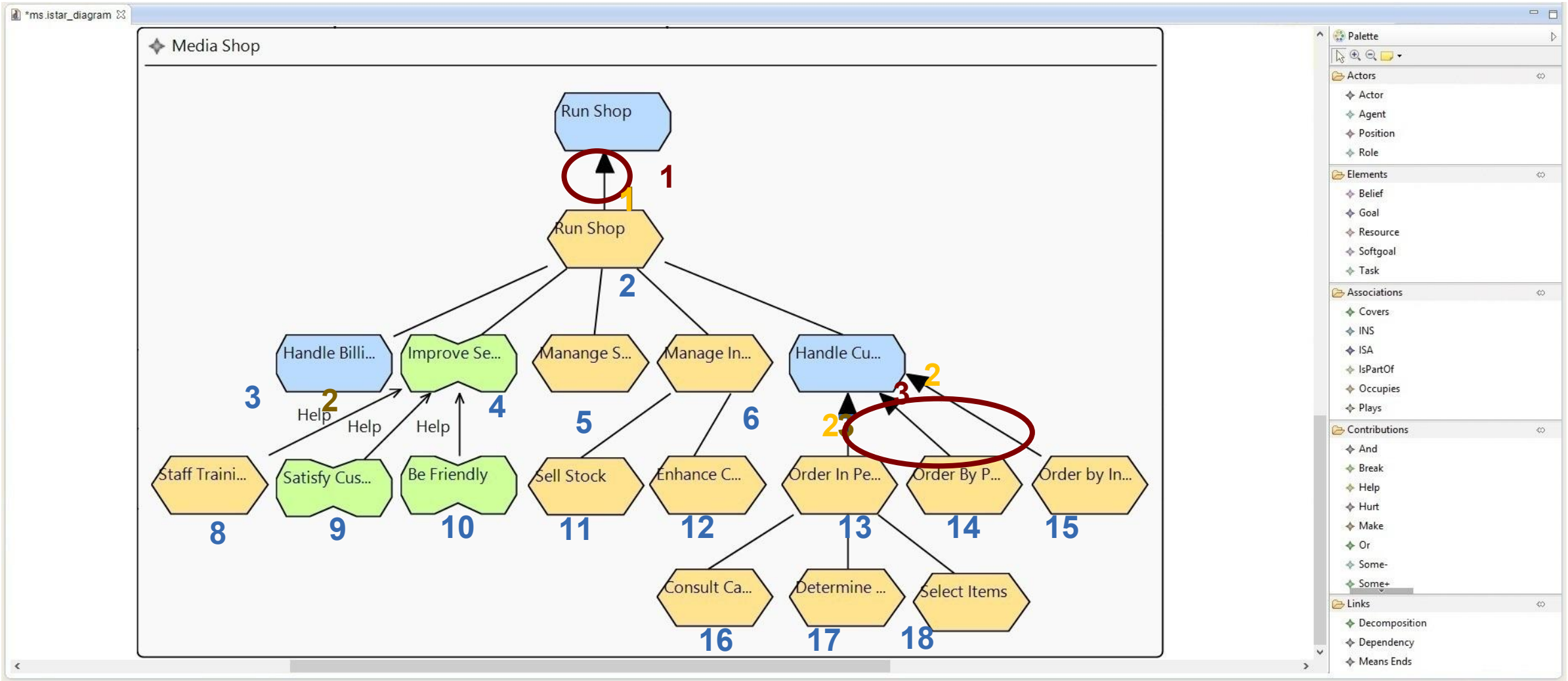
Some Metrics Definition for Q2

Q2 - *Does an actor have too much responsibility in the model?*	
Name	**NEA** – ***N**umber of **E**lements of an **A**ctor*
Informal definition	Number of elements inside an actor's boundary in the SR model
Formal definition	`context Actor` `def:NEA():Integer = self.hasElement -> select(e : Element | e.oclIsKindOf(Element)) -> size()`
Name	**AvgNEA – Av**erage **N**umber of **E**lements of an **A**ctor
Informal definition	Average number of elements inside an actor's boundary in the SR model

Formal definition	`context ISTAR`
	`pre:self.NAct() > 0`
	
	`context ISTAR`
	`def:AvgNEA():Real = self.NEA() / self.NAct()`

Requires	**NEA –** *Number of Elements of an Actor* **NAct –** *Number of Actors*

Tool Prototype

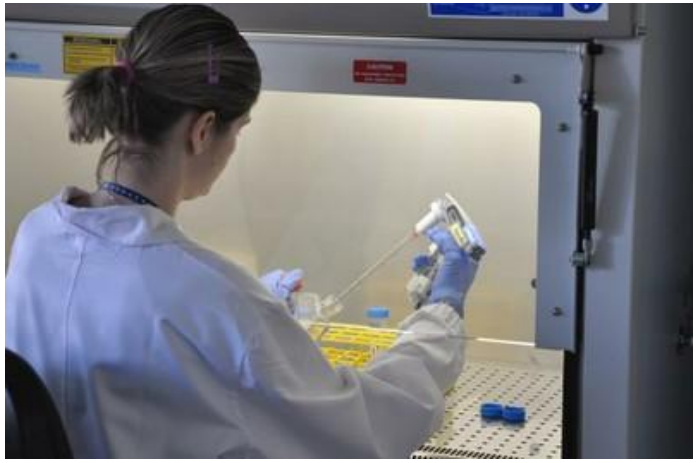


NEI	NGI	NDGI	NGWDI	MinNDGI	MaxNDGI	AvgNDGI	NSI	NDSI	NSWDI	MinNDSI	MaxNDSI	AvgNDSI	NTI	NDTI	NTWDI	MinNDTI	MaxNDTI	AvgNDTI
18	3	4	2	1	3	4/2	3	3	1	3	3	3	12	10	3	2	5	3,333

Evaluation

Case Studies

HC: Health Care



HPA: Health Protection Agency

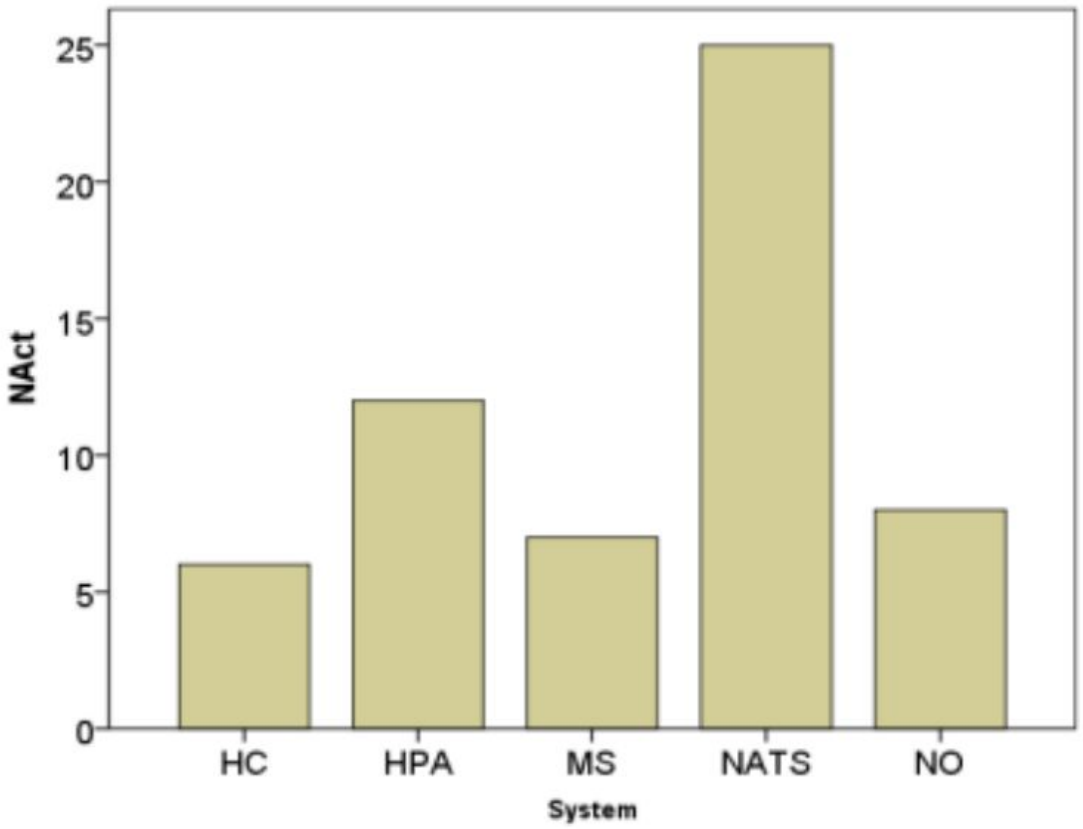
MD: Media Shop

NATS: National Air Traffic Services

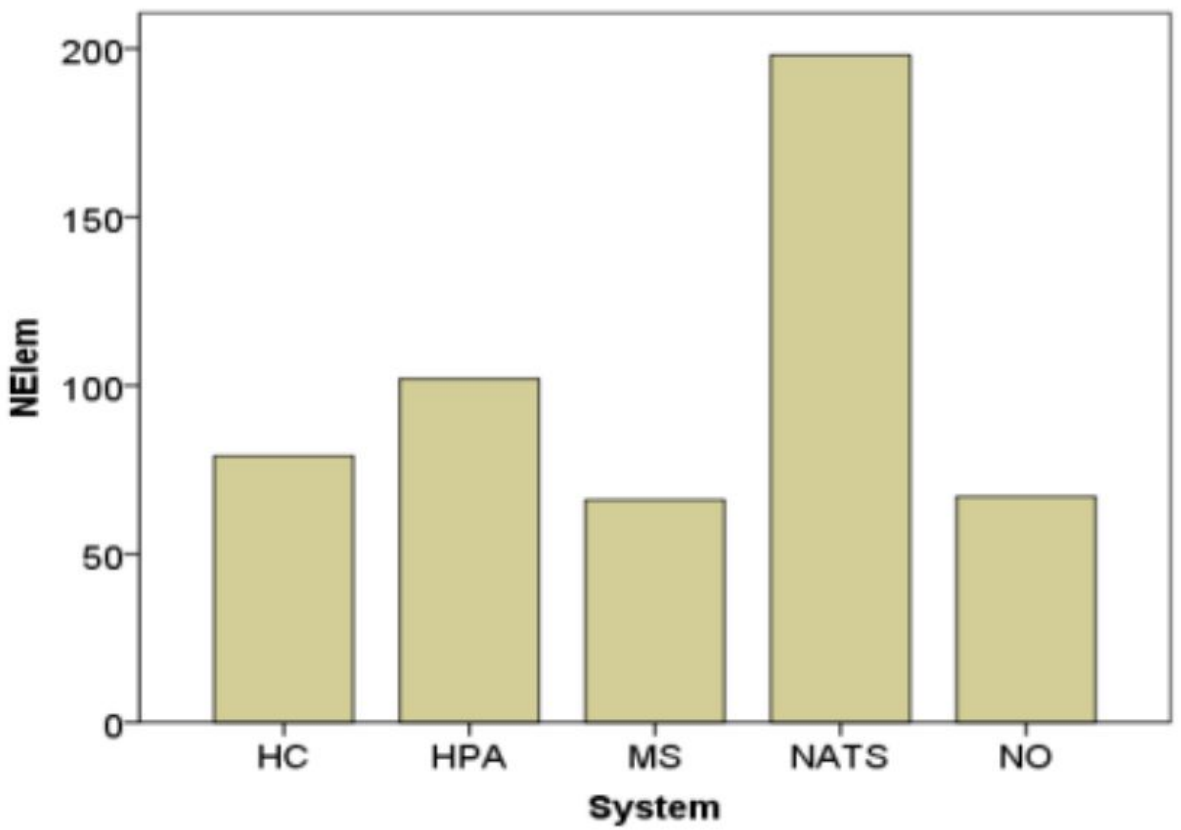
NO: Newspaper Office

Number of Actors and Elements in the System

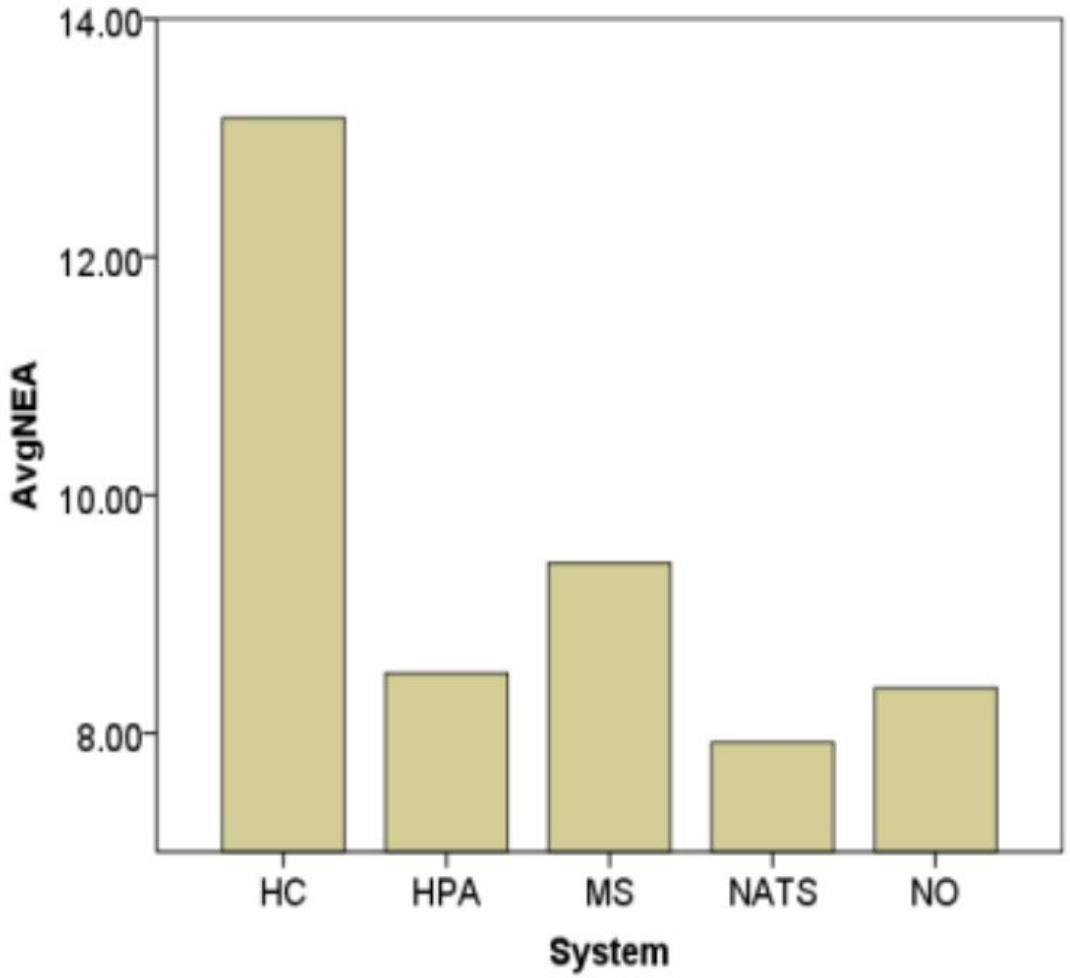
Number of Actors



Number of Elements

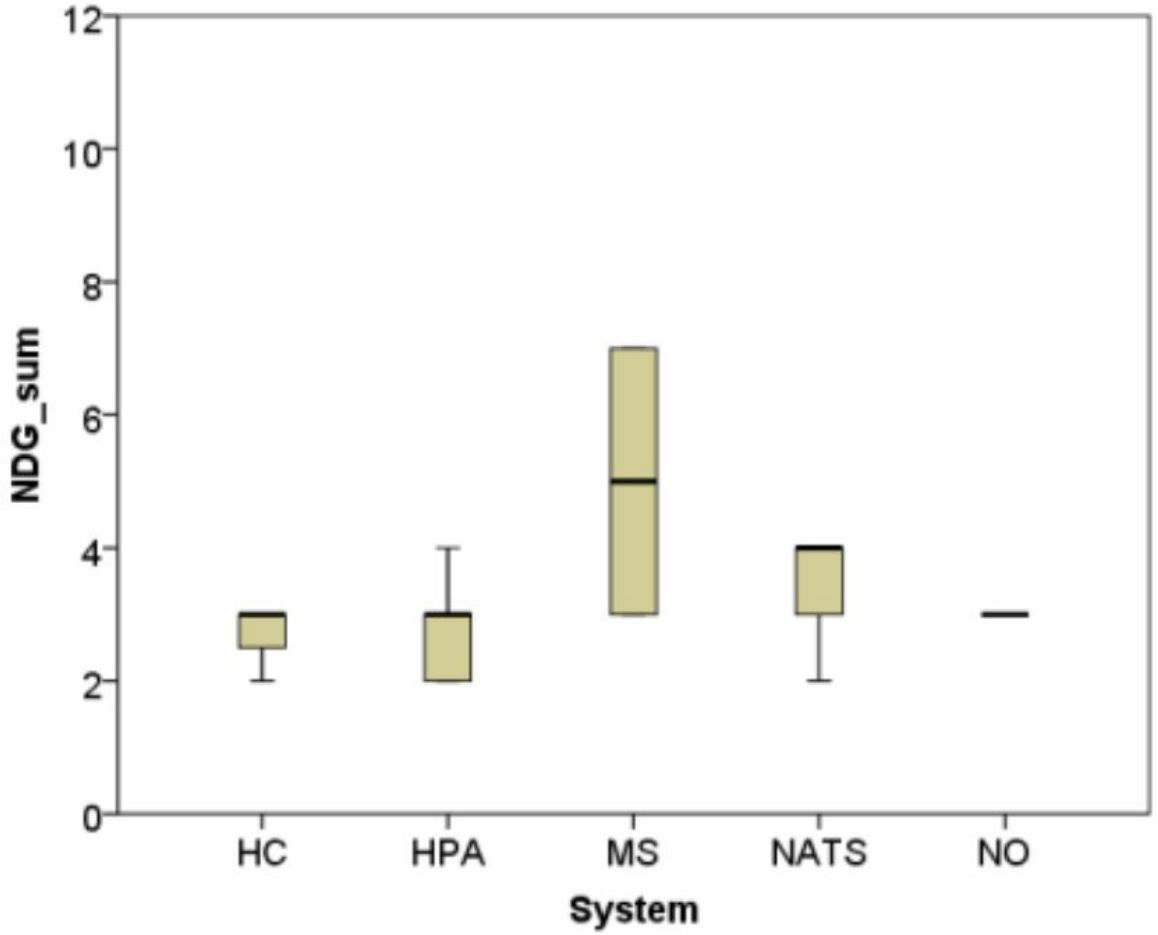


Goal Decomposition per Actor

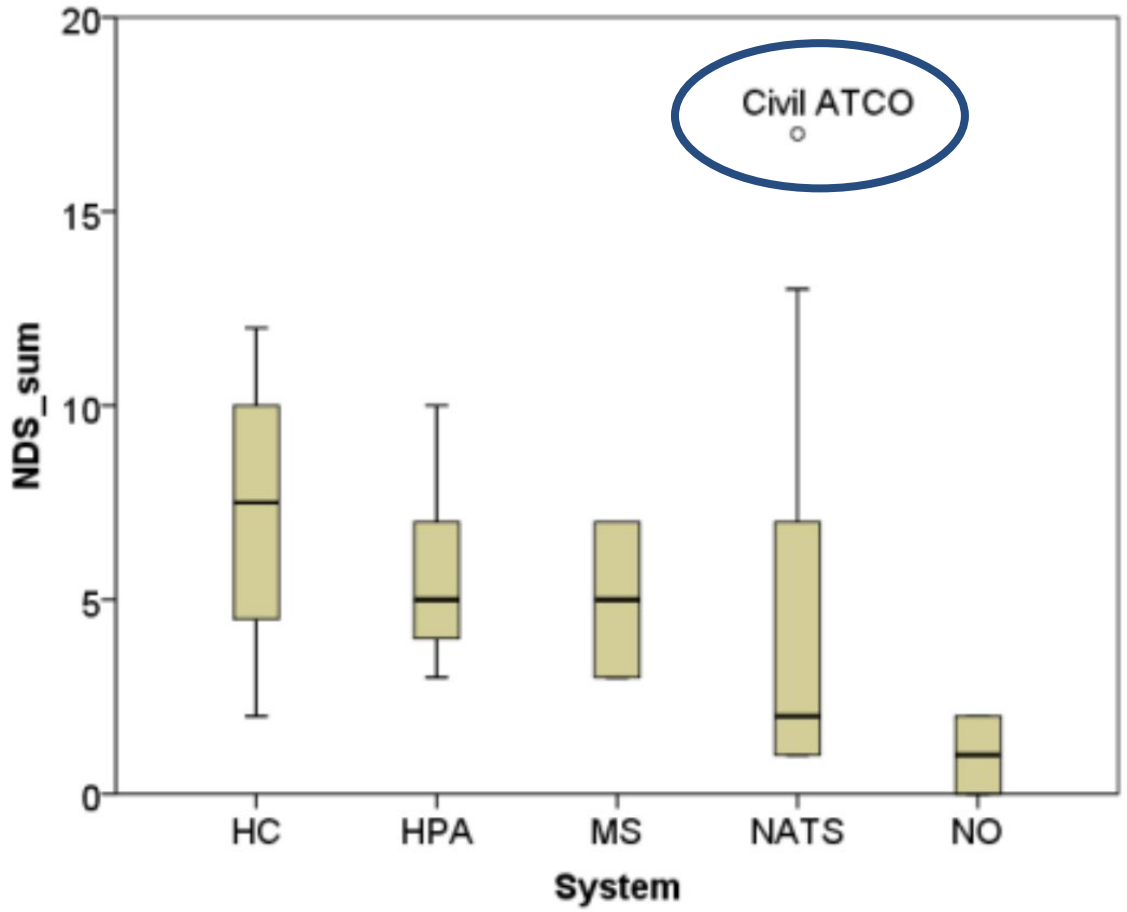


HC has a higher ratio than all the other systems, which have very similar ratios.
This may suggest that HC could be an interesting candidate for refactoring.
In contrast, we note that the most complex system, in terms of size, has the lowest element/actor density, suggesting a good overall modularity

Number of Decompositions of a Goal

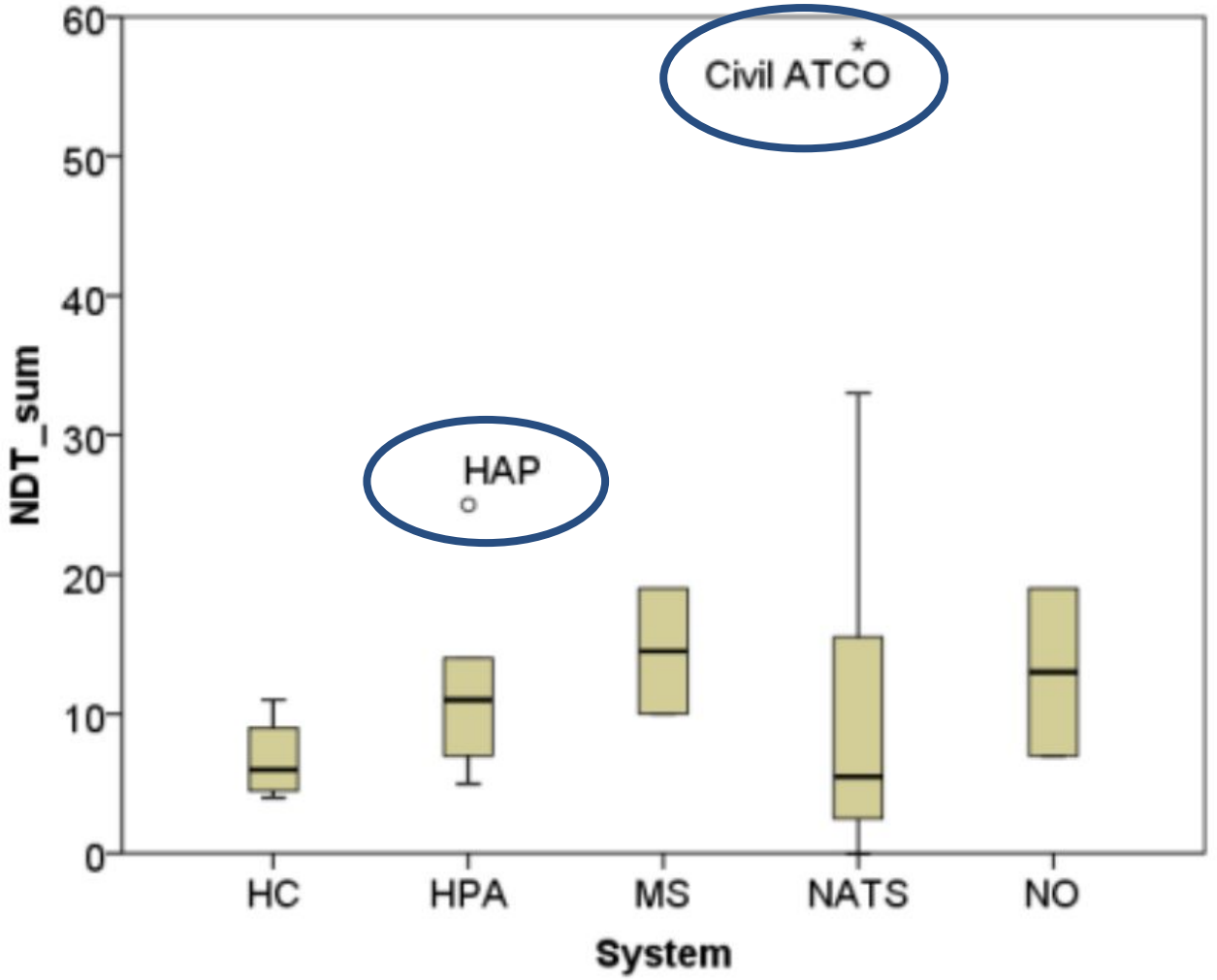


Number of Decompositions of a Softgoal



Civil ATCO may have too many responsibilities. A typical refactoring would be to decompose the actor into *sub-actors*

Number of Decompositions of a Task



It may also be the case that the requirements engineer may over-decompose these goals, softgoals, or tasks, by following a functional decomposition strategy, leading to poor modularity. This is similar to the *functional decomposition* anti-pattern

Conclusions and Future Work

Conclusions

- The questions allow evaluating the complexity of a model as a whole
- Evaluating complexity at early stages allows avoiding eventual extra costs
  - during the later stages of software development
  - during software maintenance and evolution
- The realization that the modularity of a requirements model can be improved can trigger requirements refactoring opportunities
- The results of these metrics reveal a pattern of usage in goal modelling concerning modularity of those models

Future Work

- Replicate this evaluation with other *i** models
- Extend the metrics set to cover other model quality attributes
- Identify thresholds for suggesting merging and decomposing model elements
- Conduct an experiment with requirements engineers
  - Assess the extent to which those thresholds are correlated with an increased difficulty in *i** model comprehension
- Define and apply refactoring patterns for GORE models
- Implement different views and analyse each view separately

Thank you.

Questions?