

# EXPRESSING AND ANALYSING GOAL MODELS IN DESIGN STRUCTURE MATRICES

Ralf Laue

Chair of Applied Telematics and E-Business, University of Leipzig, Germany

## ABSTRACT

Goal-oriented models describe the actors within a complex system, dependencies between system elements and organizational goals. Large models in goal-oriented graphic languages like *i\** are very difficult to comprehend. In this paper, we propose to model the relations within a design structure matrix which is easier to read. We show how existing analysis methods can be used for such matrices.

*Keywords: Goal modelling, i\*, design structure matrix*

## 1 INTRODUCTION

Goal-oriented models are frequently used in the early stages of system development, in particular for requirements engineering. Such models describe the intentions, capabilities, requirements and collaboration of actors that are involved in a system. They help to understand the goals of human actors that influence the functional and non-functional requirements of a system and the dependencies between social actors. Also, they help to identify alternatives to achieve organizational goals.

Nowadays, the *i\** language (Yu et al., 2011) is one of the most adopted frameworks in the requirements engineering community. The *i\**-based User Requirement Notation (URN) has been adapted as recommendation Z.150 by the International Telecommunication Union.

*i\** is a graphical language that aims to depict the influence that a design decision has on the achievement of various goals. *i\** models of complex systems can have a few hundred elements which are connected with different kinds of arcs. In (Matulevicius, 2008), it has been stated that *i\** goal models have one problem – “they quickly become difficult to comprehend”. Franch (2010) complains that “since the model is a monolithic unit ..., the reader has more difficulties than ever to comprehend the full meaning of the system modelled”.

This statement is illustrated by Figure 1 which shows an excerpt from a real-life *i\** model.

This paper suggests the use of design structure matrices (DSM) as a non-graphical way to model all the relations that can be found in *i\** models.

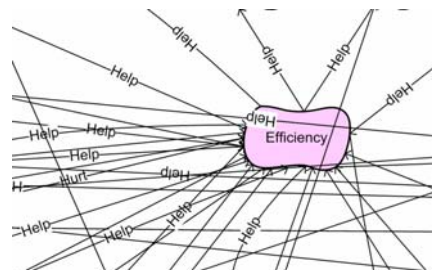


Figure 1. Relations between elements become very difficult to read (from Horkoff, 2006)

## 2 THE MODELLING LANGUAGE *i\**

This section provides a very short overview of the main concepts of *i\**. More details can be found in Yu (1995) and Yu et al. (2011).



## 2.1 Actors

Actors (depicted by a circle) in an i\* model can be human actors (“claims manager”), groups of human actors (“accounting department”) or non-human actors such as computer systems (“central database”). Human and non-human actors have in common that they are able to execute tasks and to provide resources to other actors. The i\* framework includes two kinds of models:

The Strategic Dependency (SD) model includes actors and the dependencies between them: One actor (dependor) has a strategic dependency to another actor (dependee) if the dependor can achieve its goals only if the dependee provides a resource, completes a task or achieves a goal.

The Strategic Rationale (SR) model describes **how** an actor can achieve the goals.

## 2.2 Goals and softgoals

Goals (symbol: ) inside an actor can be used to model an intentional desire of the actor. Goals are precise non-functional requirements such as “Products be sold online”. Softgoals (symbol: ) are used for quality attributes or non-functional requirements such as “High availability”.

## 2.3 Tasks

Tasks (depicted by a hexagon) inside an actor model activities that can be performed by this actor. They can also be used for modelling design decisions, for example “use a failover system”.

## 2.4 Resources

Resources (depicted as rectangle) are physical or informational entities that can be provided by an actor. Examples are “Confirmation letter” or “Internet access”.

### 2.4 Relations between actors, goals, softgoals, tasks and resources

Several kinds of relations between the model elements can be expressed in an i\* diagram:

- **Dependencies between actors:** In i\*, there are four possibilities to model a strategic dependency between actors: The dependor can depend on the dependee to perform an task (Task dependency), for the availability of a resource (Resource dependency) or to fulfill a goal or subgoal (Goal / Subgoal dependency).
- **Task decomposition:** A task can be decomposed into a sub-task, a sub-goal, a resource and a goal. This means that the task is broken into several sub-tasks and other prerequisites that must all be fulfilled in order to complete the task.
- **Means-end links** can be used to show different ways to achieve a goal. They show a relationship between an end (the goal to achieve) and the means (tasks) for attaining it. Usually, there is more than one way to achieve a goal, i.e. means-end links have to be regarded as alternatives. For example, a goal “transaction authentication number must be known to the banking customer” can be achieved either by the task “sent TAN by SMS” or “Use an one-time TAN generator”.
- **Contribution links:** Any modelling element can influence a softgoal positively or negatively. For example, the availability of a resource “Failover System” as well as the execution of the task “Perform Data Backup” can have a positive contribution to the softgoal “System Availability”. On the other hand, the availability of the resource “Failover System” can have a negative contribution to the softgoal “Keep IT Costs Low”.

Quantitative information about the kind of contribution is assigned to a contribution link from element X to softgoal S. These labels can be “make” (if X is executed/provided/fulfilled, softgoal S will be met), “some+”, “help” (both mean that if X is executed/provided/fulfilled, this has a positive contribution on softgoal S), “some-”, “hurt” (modelling a negative contribution), “break” (if X is executed/provided/fulfilled, softgoal cannot be met) and “unknown”.

Figure 2 shows an example of a very small i\* model with two actors: The user's goal “have up-to date information available” depends on the data resource that is provided by the software application if it meets its goal “provide running system”. For doing so, three design decisions on the selection of a programming language can be made, all of them have different effects (contributions) on the softgoals “Short Response Times” and “Platform Independence”. For the sake of simplicity, the model does not contain task-decomposition links.



- **for a contribution link:** Contribution links have one of the labels “make”, “some+” “help”, “some-” “hurt”, “break” and “unknown”. In accordance with the meanings of these labels, we mark the corresponding matrix element as follows:
  - 1 for a “make” contribution (if the element shown in the row heading is executed/ provided/ fulfilled, the softgoal in the column heading will always be fulfilled),
  - 0 for a “break” contribution (the softgoal in the column heading cannot be fulfilled),
  - + for a “some+” or “help” contribution (showing the positive influence),
  - for a “some-” or “hurt” contribution (showing the negative influence),
  - ? for an “unknown” contribution.
- If there is no edge between model elements, the according matrix element is left blank.

Table 1. The i\* model shown in Figure 2 as DSM

	Task “Impl. Using Assembler” (App.)	Task “Impl. Using C” (App.)	Task “Implement Using Java” (App.)	Goal “Provide Running System” (App.)	Goal “Have up-to-date information available” (User)	Softgoal “Short Response Times” (App.)	Softgoal “Platform Independence” (App.)
Task “Impl. Using Assembler” (App.)				∨		+	0
Task “Impl. Using C” (App.)				∨		+	+
Task “Impl. Using Java” (App.)				∨		-	+
Goal “Provide Running System” (App.)					Data		
Goal “Have up-to-date information available” (User)							
Softgoal “Short Response Times” (App.)					Response Time		
Softgoal “Platform Independence” (App.)							

#### 4 ANALYSING A GOAL MODEL EXPRESSED AS A DESIGN STRUCTURE MATRIX

In this section we show that the analysis of a goal model expressed as a DSM can be made at least as easy as the analysis of a graphical i\* model – by avoiding the notational overhead of the latter.

##### 4.1 Forward and backward i\* analysis

For analysing an i\* model, qualitative satisfaction labels can be assigned to elements. For example, in Figure 2, the design decision to use assembler language would be modelled by attaching a “satisfied” label to the corresponding task and a “not satisfied” label to both other tasks.

Forward analysis allows to answer the question “how effective is an alternative with respect to goals in the model?” (Horkoff & Yu, 2010). Those labels are forwarded according to propagation rules in order to reason about the satisfaction of goals. For example, a goal that has means-ends links to several tasks will get the label “satisfied” if at least one of those tasks has such a label. The full set of propagation rules is described in Horkoff and Yu (2010). When the model is represented in tabular form, this propagation algorithm can be applied very easily: The satisfaction labels can be written both in the row and column headers. The DSM follows the principle “row influences column”. From the fact that a row header for row x has a satisfaction label, it can be concluded that this label has to be considered in the forward analysis for all those elements y where element (x,y) is not empty. On the other hand, it

is easy to see whether a decision can be made on the satisfaction of element  $y$  – when all  $x$  with  $(x,y) \neq \text{empty}$  already have a satisfaction label assigned. Both manual as automated analysis can be done in the same way as for the graphical model (which is a result of the fact that both models contain the same information). The same statement is true for backwards analysis which deals with questions like “Is this goal achievable?”, “If so, how?” and “If not, why?” (Horkoff & Yu, 2010).

## 4.2 Correctness

It has been reported that  $i^*$  models (or to be more precise: models that claim to be  $i^*$  models) in practice often have incorrect syntax (Horkoff, 2008). Often, the wrong kinds of links are used between the elements. For example, the decomposition link that is allowed only to decompose a task could be used wrongly to decompose a goal into a subgoal. Of course, automatic syntax checking can be applied when a tool is used for the creation of the model (Amyot & Yan, 2008), but syntax errors in a model drawn on a sheet of paper are difficult to spot.

The situation is much better if the goal model expressed as a DSM, because such a DSM contains 5 5 sub-matrices for each combination of the element types. For example, in the sub-matrix that shows the edges from a task to a task, only task decomposition links and dependency links are allowed. Table 1 shows the entries that are allowed in each of the sub-matrices. It is easy to check whether the DSM complies to the syntactical rules of  $i^*$ .

It is also possible to detect forbidden cycles like: softgoal A has a positive “make” contribution to softgoal B, while B has a negative “break” contribution to A (Gebala & Eppinger, 1991).

## 4.3 Structural analysis

Eben and Lindemann (2010) discuss several structural criterions for analysing the relations between requirements that are given in a matrix-based description. The same criterions can be used for analysing the relations between intentional elements in an  $i^*$  model. For example, the product of the number of incoming and outgoing arcs into/from a node can be a measure for how much this element affects and is affected other elements. Elements belonging to a subset of highly interconnected elements (which do not have to be inside the same actor!) are likely to be associated to the same class of (sub)goals, etc.

## 4.4 Component selection

One application of  $i^*$  models deals with the question how to define a good architecture for systems consisting of multiple, interdependent software components. In this case, the actors in the  $i^*$  model are software components, and the  $i^*$  model shows the dependencies between them.

(Franch & Maiden, 2003) give some useful heuristics for this purpose. For example, they point out that “the flow of information from one component, shown using resource dependencies, exposes data to potential security breaches”. From this it follows that components should be packaged into “self-contained” modules such that dependencies between the modules should be minimised.

This minimisation can be achieved effectively by DSM clustering as discussed by McCord and Eppinger (1993) and Pimpler and Eppinger (1994). So far, the corresponding solutions for graphical  $i^*$  models have been described at an ad-hoc level only (Franch & Maiden, 2003). The fact that the DSM contains non-numerical values in our notation does not lead to problems, as these values can be easily transformed into numbers before applying the algorithms.

## 5 CONCLUSION

Moody et al. (2010) point out that the current  $i^*$  notation lacks effective complexity management mechanisms and suggest to introduce decomposition mechanisms to improve readability. Alencar et al. (2010) introduce an extension to the metamodel of the  $i^*$  language that allows various views on a model, but current  $i^*$  modelling tools still lack support for complexity management.

The advantage of using DSM lies in the fact that such mechanisms come almost for free, given the filtering possibilities of current spreadsheet programs. Even without filtering, dependencies between the system elements are much easier to spot (Figure 1 would correspond to a row in the DSM), and the notation (like the symbol  $\rightarrow$  for means-ends links) is intuitive with respect to the propagation rules. To sum up, DSM provide the same access to analysis algorithms as their graphical counterparts while the readability of large models is improved considerably.

## REFERENCES

- Alencar, F., Castro, J., Lucena, M., Santos, E. Silva, C. Arafujo, J. & Moreira, A. (2010). Towards Modular i\* Models. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC)*, Sierre, Switzerland, March 22-26, 2010 (pp. 292-297).
- Amyot, D. & Yan, B. (2008). Flexible Verification of User-Defined Semantic Constraints in Modelling Tools. In: *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research* (pp. 81-95).
- Eben, K.G.M. & Lindemann, U. Structural Analysis of Requirements – Interpretation of Structural Criteria. In: *Proceedings of the 12<sup>th</sup> International Dependency and Structure Modelling Conference*, 2010.
- Franch, X. & Maiden, N.A.M. (2003). Modelling Component Dependencies to Inform Their Selection. In: *Proceedings of the Second International Conference on COTS-Based Software Systems*, London: Springer.
- Franch, X. (2010). Fostering the Adoption of i\* by Practitioners: Some Challenges and Research Directions. In S. Nurcan, C. Salinesi, C. Souveyet & J. Ralyté (Eds.), *Intentional Perspectives on Information Systems Engineering*. Berlin: Springer.
- Gebala, D.A. & Eppinger, S.D. (1991). Methods for Analyzing Design Procedures. In: *Proceedings of 3rd International ASME Conference on Design Theory and Methodology* (pp. 227-233).
- Horkoff, J. (2006). *Using i\* Models for Evaluation*, Master Thesis, Department of Computer Science, University of Toronto.
- Horkoff, J., Elahi, G., Abdulhadi, S. & Yu, E. (2008). Reflective Analysis of the Syntax and Semantics of the i\* Framework. In: *Advances in Conceptual Modeling – Challenges and Opportunities, Proceedings of ER 2008 Workshops*, Barcelona Spain, October 20-23, 2008 (pp. 249-260).
- Horkoff, J & Yu, E. (2010). Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. In: *Conceptual Modeling – Proceedings of ER 2010, 29th International Conference on Conceptual Modeling*, Vancouver, BC, Canada, Lecture Notes in Computer Science, Vol. 6412, Springer (pp. 59-75)
- Kreimeyer, M., Braun, S., Gürtler, M. & Lindemann, U. (2009). Extending Multiple Domain Matrices to Allow for the Modeling of Boolean Operators in Process Models. In: *Proceedings of International Conference on Engineering Design*, Stanford, August 2009.
- Matulevicius, R. (2008). Improving the Syntax and Semantics of Goal Modelling Languages. In: *Proceedings of the 3rd International i\* Workshop*, Recife, Brazil, February 11-12, 2008 (pp. 75-78)
- McCord, K. & Eppinger, S. (1993). *Managing the Integration Problem in Concurrent Engineering*. Massachusetts Institute of Technology Sloan School of Management Working Paper 3594-93-MSA.
- Moody, D.L., Heymans, P. & Matulevicius, R. (2010): Visual Syntax Does Matter: Improving the Cognitive Effectiveness of the i\* Visual Notation. *Requir. Eng.*, 15(2), 141-175.
- Pimmler, Thomas U. & Eppinger, Steven D. (1994). Integration Analysis of Product Decompositions. In: *Proceedings of the ASME Sixth International Conference on Design Theory and Methodology*, Minneapolis, MN, September 1994.
- Yu, E. (1995). *Modelling Strategic Relationships for Process Reengineering*. PhD Dissertation, University of Toronto.
- Yu, E., Giorgini, P., Maiden, N. & Mylopoulos, J. (2011). *Social Modeling for Requirements Engineering*. The MIT Press.

Contact: R. Laue  
University of Leipzig  
Chair of Applied Telematics and E-Business  
Klostergasse 3  
04109 Leipzig  
Germany  
e-mail: laue@ebus.informatik.uni-leipzig.de

# Expressing and Analysing Goal Models in Domain Structure Matrices

Ralf Laue

Chair of Applied Telematics and E-Business,  
University of Leipzig, Germany



## Index

- Early Requirements
- i\* Modelling Concepts
- Simple Example
  - in i\* Notation
  - in DSM Notation
- DSM Analysis for Answering Common Questions
- Analysing Structural Criteria (Case Study: e-News System)
- Using DSM Clustering for Component Selection
- Summary



## Early Requirements

- Goals of human actors influence the functional and non-functional requirements of a system.
- Understanding the goals of various actors prevents from solving the wrong problem.
- Often, there are several alternatives to achieve organizational goals. Different alternatives have different advantages and disadvantages.
- i\* models intend to show these dependencies and to allow an analysis of the influence on alternative design decisions.



## i\* Modelling Language

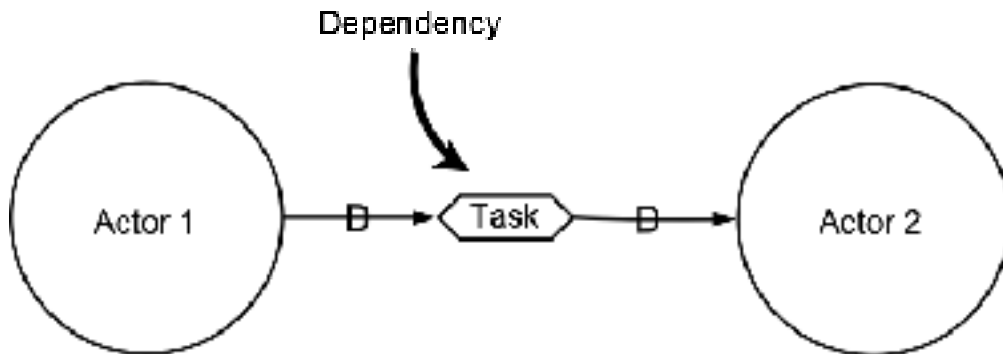
- i\* language: proposed by Eric Yu (1997)
- One of the most adopted frameworks in the requirements engineering community.
- ITU-T recommendation Z.151
- Describes:
  - Dependency relations between actors
  - How actors achieve their goal





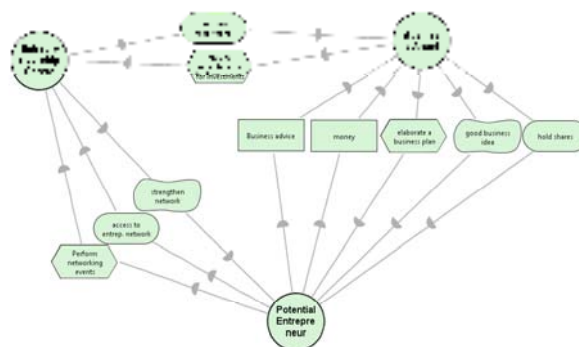
## Actors

- What are my goals?
- How they can be achieved?
- What are my abilities?
- Who do I depend on?



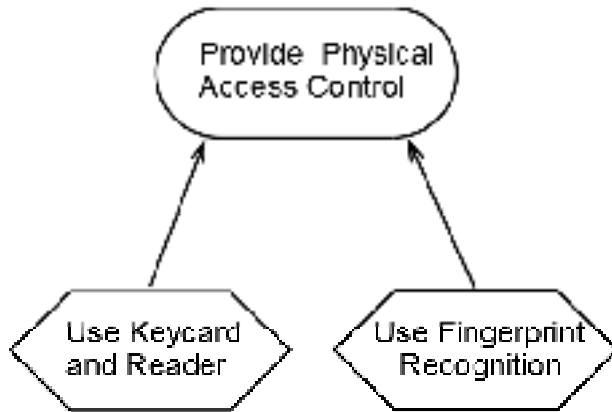
## SD-Model: Dependencies Between Actors

- Actor depend on each other...
- to achieve a goal
- to perform a task
- to get access to a resource
- --> Three types of dependencies
- SD Model



### SR Model: Inside an Actor

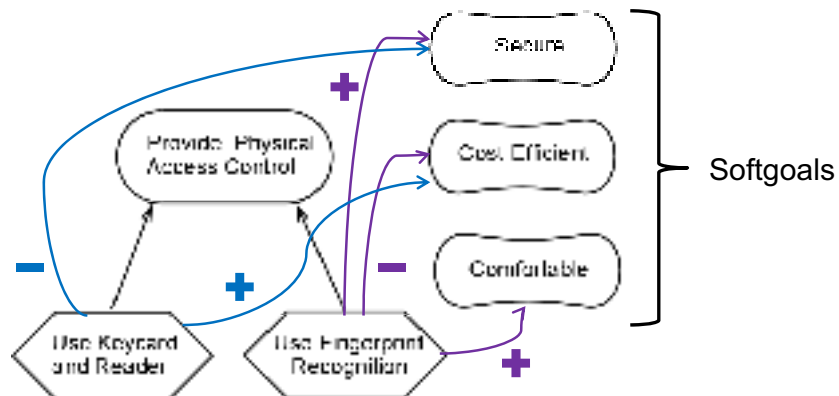
- What are the means for achieving a goal...
- Which tasks have to be performed?
- Which resources are needed?
- How are tasks decomposed into subtasks? etc.



Alternative options to achieve a goal

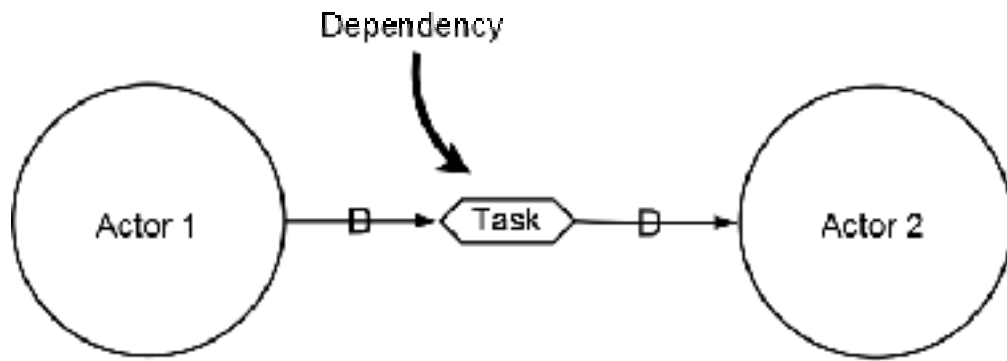


### What is the Best Way to Achieve a Goal?





INVEST ON VISUALIZATION



	Actor 1	Actor 2
Actor 1		Task
Actor 2		

depends on...

Type of Dependency



INVEST ON VISUALIZATION

	Task "Use Keycard and Reader"	Task "Use Fingerprint Recognition"	Goal "Provide Access Control"	Softgoal "Secure"	Softgoal "Cost Efficient"	Softgoal "Comfortable"
Task "Use Keycard and Reader"			✓	-	+	
Task "Use Fingerprint Recognition"			✓	+	-	-
Goal "Provide Access Control"						
Softgoal "Secure"						
Softgoal "Cost Efficient"						
Softgoal "Comfortable"						

Different kinds of vertices → different kinds of matrix elements



### Semantical Correctness

	tasks	resources	goals	subgoals	actors
tasks	$\wedge$ , dependency	dependency	$\vee$ , dependency	0, 1, +, -, ?, dependency	dependency
resources	$\wedge$ , dependency	dependency	$\vee$ , dependency	0, 1, +, -, ?, dependency	dependency
goals	$\wedge$ , dependency	dependency	$\vee$ , dependency	0, 1, +, -, ?, dependency	dependency
softgoals	$\wedge$ , dependency	dependency	$\vee$ , dependency	0, 1, +, -, ?, dependency	dependency
actors	dependency	dependency	dependency	dependency	dependency



	<b>Goal "Provide Access Control"</b>
Task "Use Keycard and Reader"	$\vee$
Task "Use Fingerprint Recognition"	$\vee$

Filtering: How can the goal "Provide Access Control" be achieved?



INVEST ON VISUALIZATION		
	Softgoal "Secure"	Soft- goal "Cost Effi- cient"
Task "Use Keycard and Reader"	-	+

Filtering: How are the softgoals affected by "Use Keycard and Reader"?



INVEST ON VISUALIZATION		
	Softgoal "Secure"	Soft- goal "Cost Effi- cient"
Task "Use Keycard and Reader"	-	+
Task "Use Fingerprint Recognition"	+	-

Analysis, for example...

- Looking for (soft)goal conflicts (see above)
- Transitive propagation ("What happens if...")
- Loop analysis

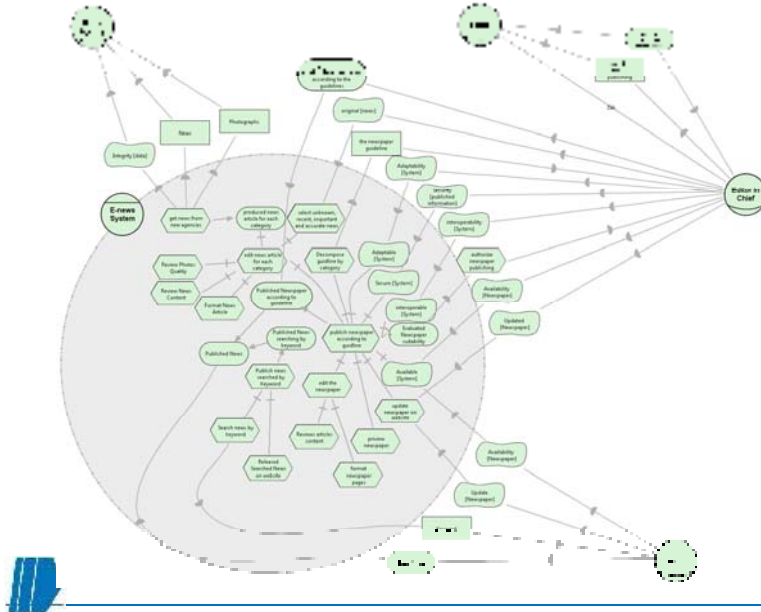
- Can be easier with spreadsheet software than with graphical i\* modelling tools

- Concept of "user programming": Spreadsheet software allows to generate own analysis queries quickly.



### Case Study: E-News System

Source: C.T.L.L. Silva: Separating Crosscutting Concerns in Agent Oriented Detailed Design: The Social Patterns Case, Doctoral Thesis, Universidade Federal de Pernambuco, 2007, p. 158



Dependencies between:

- E-News System and News Agency
- E-News System and User
- E-News System and Editor

DSM with 29 rows/columns and 38 non-empty matrix elements



### Structural Criteria

- Eben and Lindemann (2010): structural criteria for analysing the relations between requirements
- **Which elements have the largest influence on other elements?**

*Number of incoming to outgoing relations*

publish newspaper according to guideline 9

edit news article for each category 6

...

Review Photos Quality 0

- **Which (soft)goals are structurally similar to each other?**

For example we find: Soft-goal “adaptability” can be seen as structurally similar to the soft-goal “interoperability”:

They support the same goals by the same stakeholders and depend on the same elements.



## Component Selection

- One special usecase of  $i^*$  models is concerned with defining an architecture for systems that consist of multiple, independent software components.
- Actors: software components, the  $i^*$  model shows the dependencies between them.
- Heuristics: The flow of information from one component (modelled as ressource dependencies) exposes data to potential security breaches. Conclusion: Such dependencies should be minimised.
- When the usual graphical models are used, additional software would be necessary. Anyway, defining the architecture is still a semi-manual tasks.
- DSM clustering provides a simple approach.



## Summary

- $i^*$  models (as any other models that can be regarded as directed graphs) can be expressed as DSM.
- Using filtering techniques, matrix-based description can be easier to understand and to analyse than graphical models (in particular, for large models)
- DSM analysis techniques like clustering can be used.
- Structural analysis has commonalities with Eben and Lindemann (2010)

