A CASE STUDY OF COMBINING i* FRAMEWORK AND THE Z NOTATION

Aneesh Krishna, Sergiy Vilkomir and Aditya Ghose
Decision Systems Lab, School of IT and Computer Science
University of Wollongong, NSW 2522, Australia

Keywords: Agent-Oriented Conceptual Modelling, i* framework, Requirements Engineering, Z notation.

Abstract: Agent-oriented conceptual modeling (AoCM) frameworks are gaining wider popularity in software engineering. In this paper, we are using AoCM framework i* and the Z notation together for requirements engineering (RE). Most formal techniques like Z are suitable for and designed to work in the later phases of RE and early design stages of system development. We argue that early requirements analysis is a very crucial phase of software development. Understanding the organisational environment, reasoning and rationale underlying requirements along with the goals and social dependencies of its stakeholders are important to model and build effective computing systems. The i* framework is one such language which addresses early stage RE issues cited above extremely well. It supports the modeling of social dependencies between agents with respect to tasks and goals both functional and non-functional. We have developed a methodology involving the combined use of i* and the Z notation for agent-oriented RE. In our approach we suggest to perform one-to-one mapping between i* framework and Z. At the first instance general i* model has been mapped into Z schemas, and then i* diagrams of the Emergency Flood Rescue Management Case Study are mapped into Z. Some steps explaining further information refinement with examples are also provided. Using Z specification schemas, we are in a position to express properties that are not restricted to the current state of the system, but also to its past and future history. The case study described in this paper is taken from one of the most important responsibilities of the emergency services agency, managing flood rescue and evacuation operations. By using this case study, we have tested the effectiveness of our methodology to a real-life application.

1 INTRODUCTION

The early phase of requirements engineering is the most vital phase in the software development process (Fuxman et al., 2001). In this phase understanding of the organisational environment, reasoning and rationale behind requirements captured is crucial to model and realize efficient computing systems. This phase is based on the interactions between users and the stakeholders.

Agent-oriented conceptual modeling framework i* was primarily designed to model the requirements gathered during the early-phase of requirements engineering. The framework is based on the notion of "distributed intentionality" (Yu and Mylopoulos, 1994; Yu, 1994). Intentional characteristics such as goals, beliefs, capabilities and commitments are assigned to actors in their organizational environment (Yu, 1997). The i* framework consists of two main modelling components: the Strategic Dependency (SD) Model and the Strategic Rationale (SR) Model. An SD model is used to represent strategic dependencies between actors. Actors in i* framework depend on each other to accomplish goals, perform tasks, and supply resources (called dependum). The framework provides a graphical notation to describe such situations. An actor (called depender) becomes "vulnerable" if the other actor/actors (called dependee), on which it depends, do not participate in the success of the dependency. Softgoal dependencies are comparable to goal dependencies and are commonly used to represent optimisation objectives. The concept of Softgoal is based on the Non-Functional Requirements (NFR) framework proposed by (Chung et al., 2000). The SR model provides an internal intentional characteristic of each actor/agent via task decomposition links, means-end links and the rationales behind them. The detailed information on i* framework is presented in (Yu, 1995).

Many formal techniques (Heitmeyer et al., 1996;

Krishna A., Vilkomir S. and Ghose A. (2004).
A CASE STUDY OF COMBINING i* FRAMEWORK AND THE Z NOTATION.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 192-200
DOI: 10.5220/0002645301920200

Bowen, 1996) have been proposed to address the later-phase of software development, which are important for specifying precise requirements, focusing on consistency and completeness issues etc. They are mostly designed to work in the design phase. We argue that formal methods lack features for modeling the rationale behind the various design decisions taken during the early phase of requirements analysis.

This paper describes a methodology for the combined use of i* framework (Yu, 1997) and the Z notation (Bert et al., 2003; Bowen, 1996; Spivey, 1992) for requirements engineering. This approach incorporates the advantages of i* framework for early-phase RE like modeling different alternatives of the desired system, easy requirements modifications etc and then continues specifying requirements in Z using advantages of mature formal method like powerful means of specifying precise requirements, better tool support, facility of verification etc. We can also claim that i* adds value to the Z formal method. Using Z specification schemas, we are in a position to express properties that are not restricted to the current state of the system, but also to its past and future history. The approach is based on describing a one-to-one mapping of the i* framework into the Z notation. This mapping is used to formalize the i* framework without incorporating any additional information. Further refinement is done to this process by using concealed additional information from i* framework and integrating in Z schemas. This requires further investigation but we have explained this refinement with the help of simple examples. Some approaches have been proposed (Fuxman et al., 2001; Wang and Lesprance, 2001) suggesting the use of i* framework for later phases of software development.

The remainder of this paper is structured as follows. Section 2 describes Emergency Flood Rescue Management Case Study. It also explains agent-oriented conceptual modelling concepts using i* notation along with a brief introduction to Z. Section 3 concentrates on explaining the mapping of i* models into Z schemas. Steps explaining further information refinement with examples are also discussed in this section. Section 4 discusses some lessons derived from the real-life conceptual modelling exercise. Section 5 presents concluding remarks.

2 CASE STUDY: FLOOD RESCUE MANAGEMENT

This paper presents a case study based on a collaborative project to build a comprehensive enterprise model for an emergency services agency (ESA). The case study concentrates on a key function of the ESA: managing flood rescue and evacuation operations.

The ESA is responsible for managing diverse emergency situations. The case study deals with an event, which can be described in many different ways, but from an ESA perspective the event is definitely a flood response operation. The timing of the emergency response is critical in these scenarios. The ESA is the agency chosen to deal with these kinds of situations since it has the expertise and appropriate resources to deal with the threat.

During the emergency situation described above, at ESA Emergency Coordination Centre is formed and the Coordinator (ECCC) heads it. The nominated Coordinator is the person in charge of the overall emergency coordination. The first action taken by the coordinator is to activate the Emergency plan partially or fully depending on the situation. The main function of the coordinator is to bring together elements of the organization together to ensure effective emergency management response and is primarily concerned with the systematic planning and application of resources (manpower and equipment). He is responsible for the acquisition of additional resources requested by different Field Control Centre Coordinator(FCCC). His other responsibilities are to collect and assess field information so that he can coordinate the rescue and evacuation operation in a more effective and efficient manner. Other significant activities are to analyze weather forecast supplied by weather bureau and forward the analyzed forecast to the concerned FCCC with his comments/observations.

Field Control Centre is a facility, where the FCCC is located, near the scene of an emergency to facilitate better control and management of the emergency. FCCC is primarily responsible for proficiently managing the rescue and evacuation operation in the flood affected area. He is also responsible for publicizing evacuation routes for the community, manages volunteers and available resources at his disposal in most optimal way.

The other people involved in the case study are Call Taking Supervisor/System, Volunteers/Emergency Workers, Community and Weather Bureau. Call Taking Supervisor/System is responsible for managing/handling calls from the affected people, classifying/prioritizing them and forwarding calls to concerned authorities for further action. Volunteers/Emergency Workers are very important actors in the whole emergency situation. They are trained in all aspects of rescue operation. They are proficient in general rescue, providing first aid, operating communication equipment, map reading and navigation, flood rescue boat operations, giving storm safety advice, provision of essentials to people cut off by flood waters etc. Community actors in our case study are people who are affected by the flood. They are the people who are living in the flood-prone area. They are concerned about many issues and would like to

know the answers of following questions:

- How deep the water could get in and around my property?
- Whether I might need to evacuate or could get cut off by flooding?
- Which are the safest evacuation routes?

The volunteers and the ESA provide the answers to these questions. Weather Bureau is responsible for providing weather forecast data that is crucial for the efficient planning of emergency operation. The other people involved in rescue and evacuation mission are dependent on weather bureau to provide forecast data at regular interval for the duration of emergency.

As we are aware that agent-oriented conceptual modelling notations are in their early years of development. Very few attempts have been made to deploy them in industry scale projects. The ESA is good testing ground for agent-oriented conceptual modelling deployment in the large-scale project. We believe that ESA offers some distinctive features like infrastructure that remains inactive for long periods of time but gets activated only during an emergency situation. These features make traditional conceptual modelling frameworks somewhat difficult to use in this domain, but this paves the way for using agent-oriented conceptual modelling techniques (a more detailed discussion of these issues is outside the scope of this paper, but is being reported elsewhere).

The case study shown in Figure 1 is used to illustrate the SD model (Yu, 1995) of managing flood rescue and evacuation operations by ESA. The modeling process begins with the identification of actors/agents involved in the flood rescue and evacuation operation and identifying mutual relationships between them. The Emergency Coordination Centre Coordinator (ECCC) agent depends on the Field Control Centre Coordinator (FCCC) agents to accomplish its goal Rescue People At Risk. Similarly the FCCC agent depends on the ECCC agent to achieve Coordination Support goal. The ECCC depends on the Call Taking Supervisor/System to provide Information About People At Risk, modeled as a goal dependency. Similarly, ECCC agent depends on Weather Bureau and Volunteers/Emergency Workers agents to achieve the goals Weather Forecast/Warnings and Evacuation and Rescue Mission respectively. The ECCC has a dependency on the Weather Bureau to provide Weather Data, modelled as a resource dependency. Similarly, ECCC has a dependency on the Volunteers/Emergency Workers and FCCC agents to provide Field Information and Acknowledgment of Emergency Notification, Local Information Update respectively, modelled as resource dependencies. The remaining dependencies may well be explained on the similar lines.

The SD model provides an external characterisation of an actor/agent in terms of two sets of dependencies: incoming dependencies (the agent acting as dependee) and outgoing dependencies (the agent acting as depender) (Yu, 2001). In SD model this is represented by showing the left half-arrow (pointing from right to left) to denote incoming dependency and the right half-arrow to denote outgoing dependency.

In the SR model a more detailed level of modeling is performed by looking "inside" the actors/agents to model internal relationships (Yu, 1995). The SR model is a graphical representation that captures some of the rationales involved in the rescue and evacuation setting. The SR model shown in Figures 2 and 3 elaborates on the relationships between the actors shown in SD model of Figure 1. There are two main types of links possible in SR model: means-ends links and task decomposition links.

For example, Volunteers/Emergency Workers has an internal task to Rescue People. This task can be performed by subtasks Prepare For Rescue, Map Reading and Navigation, Operate Rescue Boats, Communication Equipment Operation, Supply Essentials, Rescue/Evacuate People at Risk and goal Report Situation-which indicates that there can be different ways of achieving it (these are related to the parent task via task decomposition links). The Rescue/Evacuate People at Risk task is further decomposed into subtasks Provide First Aid and Conduct Emergency Drills. The softgoal Fast and Efficiently is also related to the Rescue People task via a task decomposition link. When a softgoal is a component in task decomposition, it serves as a quality goal for that task. Assessment of the local flood situation by Volunteers/Emergency Workers in the flood-affected region is crucial for the ECCC and FCCC agents to further plan their resources and strategies in an optimal way. This is represented as subgoal Report Situation in the SR model of Volunteers/Emergency Workers. This subgoal can be achieved by using any one of the shown subtasks, Radio/Mobile Phone or Satellite Phone. This is represented by a means-ends link. This is a Goal-Task Link (GTLink), the end here is specified as a goal Report Situation, and means is specified as a tasks (by using Radio/Mobile Phone or Satellite Phone). This task specifies the "how" through its decomposition into components.

3 i* - Z TRANSFORMATION

We believe that formal specification offers significant benefits to the developers of computer systems and it can play major role towards improving their quality. The Z notation is foremost amongst formal methods in use at present and is based on set theory and

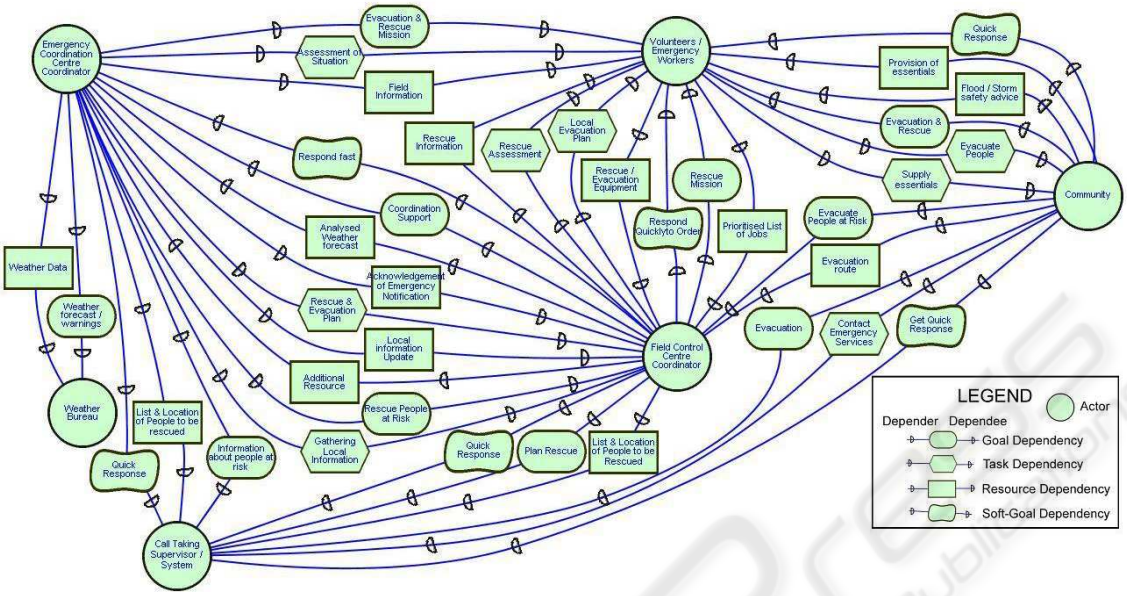

Figure 1: The Strategic Dependency Model

first order predicate logic (Spivey, 1992). A Z notation works by modelling the states that a system can take and the operations that cause changes in those states. A specification document written in Z consists of schemas. The schema is referred to by the name. It consists of declarations and constraining predicates.

Motivations and advantages behind the combined use of i* and Z are the following:

- We feel that the usefulness and effectiveness of i* can be increased manifold by using it with a formal specification language such as Z. Mapping rules provide a formal semantics to i* framework. Our view is that the Z formal notation and the i* modelling framework can function in a complementary and synergistic fashion and that a conceptual modelling methodology that supports their co-evolution is of interest.
- There is a need to map both SD and SR models into late phase requirements specifications; Z can be used effectively to realize these goals.
- We can formalize softgoals in Z, along with the task and goals. The *i** notation allows us to represent and reason with softgoals (representations of non-functional requirements or objectives).
- There is tool support available for Z and Z is a mature method that is widely used in arriving at complete specifications.
- For translating informal specifications provided in i* into Z, there is no need to add more details into the corresponding i* model. The mapping from i* models into Z schemas does not result in any information loss.
- Using Z specification schemas, we are in a position to express properties that are not restricted to the current state of the system, but also to its past and future history.

3.1 Representation in Z of General SD and SR Models

The basis of mapping specific i* models into Z is the representation in Z of general SD and SR models. Then existing information is added into general Z schemas to specify the specific i* models i.e., general representation in Z is used as a template. We have suggested and considered in detail Z representation of general SD and SR i* models in (Vilkomir et al., 2004), so here we are just showing the respective notations with the essential comments.

The source of names of actors *all_actors* and dependencies *all_depend* is given set *NAME*. Free types *STATE*, *TYPE*, *DEGREE* and *LINK_TYPE* describe correspondingly the possible states, types and degrees of dependencies and internal intentional elements and the types of links between internal intentional elements.

$[NAME]$

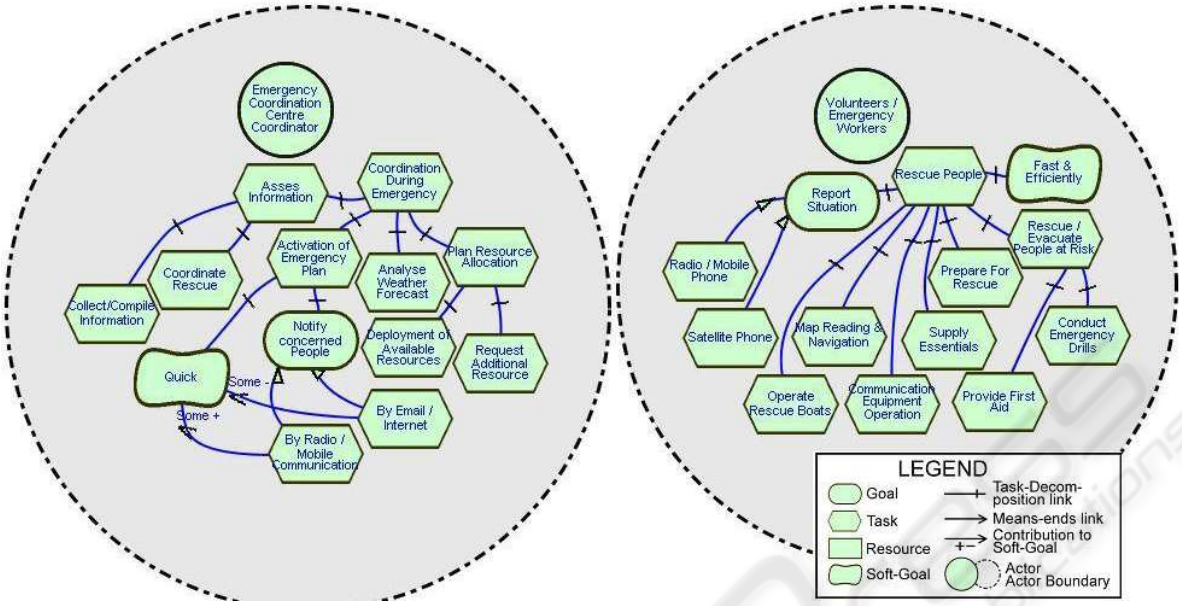

Figure 2: The Strategic Rationale Model

$$all\_actors, all\_depend : \mathbb{P}_1\ NAME$$

$$\begin{aligned} STATE ::= {} & inapplicable \mid unresolved \\ & \mid fulfilled \mid violated \mid satisficed \\ & \mid denied \mid undetermined \end{aligned}$$

$$\begin{aligned} TYPE ::= {} & goal \mid softgoal \mid task \mid resource \\ & \mid ISA \end{aligned}$$

$$DEGREE ::= open \mid committed \mid critical$$

$$\begin{aligned} LINK\_TYPE ::= {} & NA \mid task\_decomp \mid means\_ends \\ & \mid contrib \end{aligned}$$

The state of a SD model is a collection of states of all dependencies. The state of an actor is a collection of states of all internal intentional elements.

SD

$$SD\_state : NAME \pfun STATE$$

$$\mathrm{dom}\ SD\_state = all\_depend$$

Actor

$$actor\_name : NAME$$
$$actor\_element : \mathbb{P}_1\ NAME$$
$$actor\_state : NAME \pfun STATE$$

$$actor\_name \in all\_actors$$
$$\mathrm{dom}\ actor\_state = actor\_element$$

As a common pattern for SD dependencies and SR elements, the partial specification $\Phi Depend$ is used.

$\Phi Depend$

$$dependum : NAME$$
$$type : TYPE$$
$$degree : DEGREE$$
$$result! : STATE$$

$$result! \neq unresolved$$
$$result! = satisficed \lor result! = denied \lor result! = undetermined \Rightarrow type = softgoal$$

All these schemas allow creating *SDependency* schema to describe the general structure of a SD dependency.

SDependency

$$\Delta SD$$
$$\Phi Depend$$
$$depender, dependee : NAME$$

$$dependum \in all\_depend$$
$$depender \in all\_actors$$
$$dependee \in all\_actors$$
$$SD\_state' = SD\_state \oplus \{dependum \mapsto result!\}$$

To describe the general structure of internal elements in an actor, firstly it is necessary to create a Z schema, which reflects a structure of links between internal intentional elements.

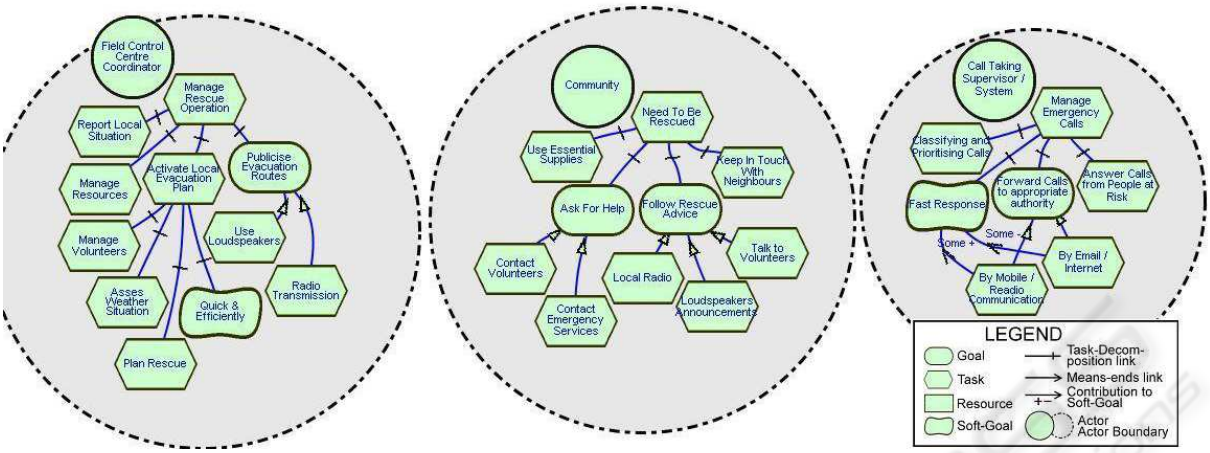

Figure 3: The Strategic Rationale Model

Link

$\Phi Depend$
$int\_components, ext\_components : \mathbb{P}\, NAME$
$contrib\_p, contrib\_m : \mathbb{P}\, NAME$
$link : LINK\_TYPE$

$link = task\_decomp \Rightarrow type = task$
$link = contrib \Rightarrow type = softgoal$
$contrib\_p \cup contrib\_m \neq \varnothing \Rightarrow link = contrib \wedge$
$\langle contrib\_p, contrib\_m \rangle$ partitions $int\_components$
$ext\_components \neq \varnothing \Rightarrow link = task\_decomp$
$link = NA \Leftrightarrow$
$int\_components \cup ext\_components = \varnothing$

Using *Link* schema as a component, now it is possible to describe the general structure of an intentional internal element in a general SR model.

AElement

$\Delta Actor$
Link

$dependum \in actor\_element$
$int\_components \subset actor\_element$
$ext\_components \subseteq all\_depend$
$actor\_name' = actor\_name$
$actor\_element' = actor\_element$
$actor\_state' =$
$actor\_state \oplus \{dependum \mapsto result!\}$

3.2 Mapping the Specific i* Model into Z

The first step of mapping SD model of managing flood rescue and evacuation operations is to specify names of all the actors and all external dependencies.

$coordinator, bureau, supervisor, worker,$
$field\_coord, community : NAME$
$weather, rescue\_plan, respond\_quickly,$
$evacuation, field\_info, evacuate\_people,$
$get\_quick\_response,$
$contact\_emerg\_services : NAME$

$all\_actors = \{coordinator, bureau, supervisor,$
$worker, field\_coord, community\}$
$all\_depend = \{weather, rescue\_plan, field\_info,$
$respond\_quickly, evacuation, evacuate\_people\}$

For simplicity, we are specifying only 8 out of 38 dependencies but for proper mapping it is necessary to specify all the names.

The second step is the creation of a Z schema for every dependency using *SDependency* schema as a basis. So for SD model of managing flood rescue and evacuation operations we need to create 38 schemas. Thus, the volume of using Z notation corresponds with the complexity of SD model. However, all Z schemas are of the same type, the process of their creation is very simple and can be easily automated. For simplicity, we are creating only 4 schemas for different types of dependencies below.

Weather
SDependency

$dependum = weather$
$depender = coordinator$
$dependee = bureau$
$type = resource$
$degree = committed$

RescuePlan
SDependency

$dependum = rescue\_plan$
$depender = coordinator$
$dependee = field\_coord$
$type = task$
$degree = committed$

RespondQuickly
SDependency

$dependum = respond\_quickly$
$depender = field\_coord$
$dependee = worker$
$type = softgoal$
$degree = committed$

Evacuation
SDependency

$dependum = evacuation$
$depender = community$
$dependee = supervisor$
$type = goal$
$degree = committed$

The first step of mapping SR model of managing flood rescue and evacuation operations is creating a Z schema for every actor using *Actor* schema as a basis. In these schemas we need to specify names of all the internal intentional elements. As an example, the schema for *supervisor* actor is as shown below:

Supervisor
Actor
$classifying\_calls, fast\_response, mobile, email,$
$forward\_calls, emergency\_calls,$
$answer\_calls : NAME$

$actor\_name = supervisor$
$actor\_element = \{classifying\_calls,$
$fast\_response, mobile, email, forward\_calls,$
$emergency\_calls, answer\_calls\}$

The second step is the creation of a Z schema for every internal intentional element using *AElement* schema as a basis. For example, we need to create 7 schemas for the internal elements of *supervisor* actor. To demonstrate this approach and show various possible situations, we are creating below three of them.

ClassifyingCalls
AElement
Supervisor

$dependum = classifying\_calls$
$type = task$
$degree = committed$
$int\_components = \varnothing$
$ext\_components = \varnothing$
$link = NA$

FastResponse
AElement
Supervisor

$dependum = fast\_response$
$type = softgoal$
$degree = committed$
$int\_components = \{mobile, email\}$
$contrib\_p = \{mobile\}$
$contrib\_m = \{email\}$
$ext\_components = \varnothing$
$link = contrib$

AnswerCalls
AElement
Supervisor

$dependum = answer\_calls$
$type = task$
$degree = committed$
$int\_components = \varnothing$
$ext\_components = \{get\_quick\_response,$
$contact\_emerg\_services\}$
$link = task\_decomp$

We have considered Z schemas represented above as part of one to one mapping of i* models (of the case study) into the Z notation. Using this approach, all the information from the i* models is reflected in Z. Further refinement is done to this process by using concealed additional information from i* framework and integrating it in Z schemas.

For example, consider again dependencies of *supervisor* actor. It is necessary to classify a call before forwarding it to the appropriate authority so dependency *classifying_calls* should be realized before

dependency *forward_calls*. This fact is not reflected in the i* diagrams but can be easily incorporated into Z. For this, it is necessary to include into the predicate part of *ForwardCalls* schema the following precondition: $SD\_state(classifying\_calls) = fulfilled$.

ForwardCalls

AElement
Supervisor

$dependum = forward\_calls$
$type = goal$
$degree = committed$
$int\_components = \{mobile, email\}$
$ext\_components = \{quick\_response\_1, list\_1,$
$plan, quick\_response\_2, list\_2,$
$contact\_emerg\_services\}$
$link = means\_ends$
$SD\_state(classifying\_calls) = fulfilled$

For dealing with systems which stipulate special timing requirements (like concurrent real-time reactive systems), it is possible to use special extensions of Z like Timed Communicating Object Z (TCOZ) (Mahony and Dong, 2000) which are designed for modelling real-time applications.

By applying our approach on the Emergency Flood Rescue Management Case Study, we are in a position to test the effectiveness of our methodology to real-life application.

4 ANALYSIS

Managing flood rescue and evacuation operation can be considered as a rich and challenging example for requirements engineering. As compared to conventional examples used in various research papers, it offers extraordinary combination of features found in real-life requirements engineering exercise. For example, the performance of the automated part (Call Taking Supervisor/System) of the combined system depends heavily on the satisfactory actions taken by the agents/actors in the environment.

This case study presents some methodological principles and lessons derived from a real-life conceptual modelling exercise:

Adding some structure to the modeling process

Early-phase RE activities have traditionally been done informally, beginning with stakeholder interviews and discussions on the existing systems and rationales. The motivation of early phase RE is to create a conceptual model of the domain, which is closer to the user view. We have proposed (Unni et al., 2003) to add some structure to this requirements gathering exercise by introducing the use of Requirements Capture Templates. These templates are to be filled out by the modeller whilst conducting interviews with the stakeholders. These forms have been designed to seek information specific to the needs of the underlying agent-oriented conceptual model that the modeller seeks to build. Consequently stakeholders are in a position to provide information for the conceptual modeling task, without being exposed to the complexities of understanding and using the conceptual modelling language. The design of RCT's makes it easy to systematically transform the details captured directly into AoCM.

Using existing domain knowledge

Many problem domains are well understood, with an existing body of knowledge, which a modeller can access. In (Unni et al., 2003) we have outlined an approach in which the pre-defined domain ontologies (specific to appropriate domain) can be used to generate elicitation triggers. We have also suggested how completeness and consistency testing of the conceptual models in relation to the domain ontology can lead to elicitation prompts for the modeller. We have proposed a methodology, which supports the co-evolution of conceptual models, ontologies and RCTs through the process of requirements elicitation and modelling.

Incremental creation of knowledge and model

The requirements gathering exercise began by conducting stakeholder interviews and discussions on the existing activities, systems and rationales. We agree that most of the RE activities are informal in nature. Modelling exercise benefited a lot as a result of observing practicing stakeholders and additional reading was undertaken to clarify these observations. We had constant interactions with the stakeholders during the requirements gathering as well as modeling exercise. Feedback from the stakeholders led to the development of conceptual model, which was closer to the users view. As a consequence the model was modified three times, reflecting upon the incremental creation of knowledge and model.

Observation

We believe that the modellers or systems analysts bring theoretical and technical knowledge to the RE process. The stakeholders bring comprehensive knowledge of their domain and experience of the problem situation. Neither has superior knowledge, rather they are both proficient in their own domains; it is the interaction between these two group of people that forms the basis for the problem solving required from the RE process. We feel that it is in this area that further research will bring advance understanding of the RE process.

5 CONCLUSION

This paper describes a methodology for the combined use of i* framework and the Z notation for requirements engineering. This approach incorporates the advantages of i* framework for the early phase and then continues specifying requirements in Z using advantages of mature formal methods. The approach is based on describing a one-to-one mapping of i* framework into the Z notation. This mapping is used to formalize i* framework without incorporating any additional information. Further refinement is done in this process by using concealed information from i* framework and incorporating it in the Z schemas. This requires further investigation but we have explained this refinement with the help of simple example. The case study described in this paper is taken from one of the most important and challenging responsibilities of the emergency services agency, managing flood rescue and evacuation operations. By using this case study, we have tested the effectiveness of our methodology to real-life application. Further we would like to add that the formalization of i* framework to Z can be automated.

REFERENCES

Bert, D., Bowen, J. P., King, S., and M. Walden, e. (2003). *Formal Specification and Development in Z and B, Third International Conference of B and Z Users (ZB2003), Turku, Finland, June 4-6, 2003, Proceedings. LNCS 2651*. Springer.

Bowen, J. P. (1996). *Formal Specification and Documentation Using Z: A Case Study Approach*. International Thomson Computer Press.

Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2000). *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers.

Fuxman, A., Pistore, M., Mylopoulos, J., and Traverso, P. (2001). Model checking early requirements specifications in tropos. In *Proceedings of Fifth IEEE International Symposium on Requirements Engineering, 27-31 August 2001*, pages 174–181, Toronto, Canada.

Heitmeyer, C., Jeffords, R., and Labow, B. (1996). Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology*, pages 231–261.

Mahony, B. and Dong, J. S. (2000). Timed communicating object z. *IEEE Transactions on Software Engineering*, 26(2):150–177.

Spivey, J. M. (1992). *The Z Notation: A Reference Manual*. Prentice Hall International Series in Computer Science, 2nd edition.

Unni, A., Krishna, A., Ghose, A. K., and Hyland, P. (2003). Practical early phase requirements engineering via agent-oriented conceptual modelling. In *To appear in Proceedings of ACIS-2003: The 2003 Australasian Conference on Information Systems, 26-28 November 2003*, Perth, Australia.

Vilkomir, S., Ghose, A. K., and Krishna, A. (2004). Combining agent-oriented conceptual modeling with formal methods. In *Submitted to Australian Software Engineering Conference (ASWEC-2004), 13-16 April 2004*, Melbourne, Australia.

Wang, X. and Lesprance, Y. (2001). Agent-oriented requirements engineering using congolog and i*. In *Proceedings of 3rd International Bi-Conference Workshop Agent-Oriented Information Systems (AOIS-2001)*, pages 59–78, Berlin, Germany.

Yu, E. (1994). Towards modelling strategic actor relationships for information systems development – with examples from business process reengineering. In *Proceedings of 4th Workshop on Information Technologies and Systems, 17-18 December 1994*, pages 21–28, Vancouver, B.C., Canada.

Yu, E. (1995). Modelling strategic relationships for process reengineering. Technical report, PhD Thesis. Dept. of Computer Science, University of Toronto.

Yu, E. (1997). Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of 3rd IEEE International Symposium on Requirements Engineering, 6-8 January 1997*, pages 226–235, Washington D.C., USA.

Yu, E. (2001). Agent orientation as a modelling paradigm. *Wirtschaftsinformatik*, 43(2):123–132.

Yu, E. and Mylopoulos, J. (1994). Understanding why in software process modelling, analysis, and design. In *Proceedings of 16th International Conference on Software Engineering, 16-21 May 1994*, pages 159–168, Sorrento, Italy.