Modeling Ideator using Tropos

Syed Hamza Javed

Msc.Ac Candidate, University of Toronto
Toronto, Canada M5S 2E5

Abstract
The aim of this paper is to highlight the experience of modeling Ideator, a collaborative mind map that will be used in creative ideation, using the Tropos modeling language. Tropos is built on top of and extends the i* framework.

1. Introduction and Motivation

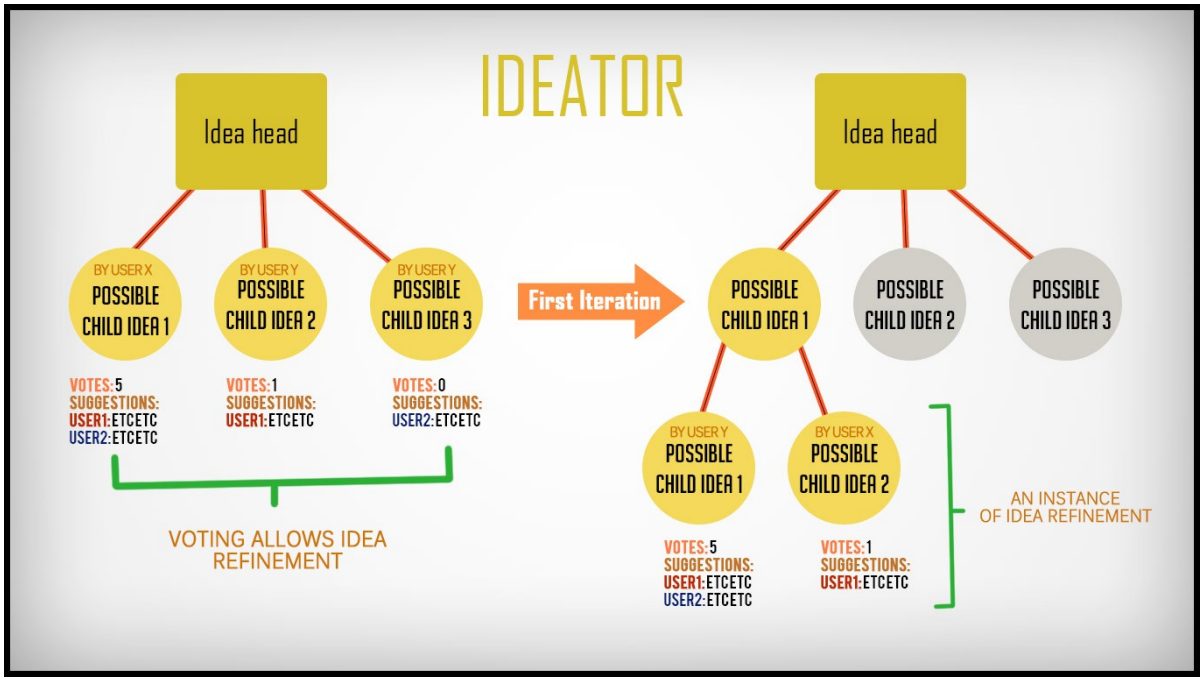
In this paper we explore the use of i* and Tropos to model a collaborative mind map, dubbed Ideator.
I* is a framework that allows developers to model information systems in terms of their heterogeneous actors, each with their own goals and intentions [1]. Tropos extends that framework and allows the user to go beyond just the intentional modeling and allows the developer to model and then implement the system in that environment. There are five key stages to the process, namely early requirements, late requirements, architectural design, detailed design and implementation [2]. We will closely look at the first four, and discuss the potential of the fifth.
We start by describing what Ideator is in section 1.1. Next we move on to defining I* and Tropos in a bit more detail in 2.1 and 2.2. In 2.3 we estabilish some Key terms used throughout the paper. Section 3 talks about tool selection and in section 4, having selected the tool, we talk about the experience of modeling with Tropos. We present the conclusion and discuss future work in section 5.

The key reason for using Tropos to model Ideator lies in the intrinsic collaborative and social nature of Ideator itself. The belief was that using Tropos will not only be fitting for modeling Ideator, but also help explicate some of the hidden aspects that other modeling tools would not have surfaced.

So what is Ideator?

1.1 – Ideator:

Ideator is a web-based tool to aid teams in work collaboration and idea organization with a focus on face-to-face team meeting settings led by a facilitator. The problem it seeks to solve is the management of data during team-based ideation processes.



It does by providing a collaborative mind map on which a team can ideate using their own computers or devices. It starts by taking an input from the meeting facilitator (here-after known as the Head of the meeting). This input is the main idea or the theme of the ideation, which has been provided to the Head by the client.

Once the theme has been posted the ideators (people participating in the ideation session) start thinking about the idea theme and start adding child ideas to it. At this time other ideators will view and review the posts by the people who have posted and then vote and comment on the ideas. The ideas that receive votes above a certain threshold will be retained, while the others will be dismissed.

The purpose of this is to refine the ideas and counter the cluttering of space with undesirable ideas.

2. Tropos and i*

2.1 – I* framework

I* modeling intends to bring the concepts of sociality into the system engineering process. It does this by introducing certain social ideas and dependencies into the modeling process and forces the developer to shift away from the mechanistic view of flows and activities and focus on “what each playing actor wants”, “How they plan to achieve it” and “Who do they depend on to achieve it”. [1]

2.2 - Tropos:

Tropos is an Agent Oriented modeling language that adopts the i* framework. It allows for the analysis and design of the entire software software development process. The idea is to model the System-to-be, all the actors that will contribute to this system along with the internal actors that will provide the required functionalities to the external actors.
The models go through several phases, getting more detailed and refined with each iteration. [3]

There are 5 major stages in the Tropos development process:

Early requirements:
This is the first step of requirements engineering. Here the requirements engineer must first identify all the stakeholders in the given domain and then model there stakeholders as actors and agents in way which highlights how these actors depend on each other for various goals, tasks or resources. Once these dependencies are clearly modeled the engineer can see the "WHY" behind the systems functionalities. We start this phase by making the Strategic Dependency diagram which models the the dependencies between the actors. Next we expand onto this diagram by exploring each actors inner rationale based on his internal goals and tasks. This is the Strategic Rationale model.

Late requirements:
In the late requirements the model created earlier is further extended and a new actor is added to this model. This actor represents the system and defines the relations and dependencies with all the other actors. This extended model gives us the functional and non-functional requirements of the system-to-be.

Architectural Design:
Now that a system-to-be has been modeled, this system can be further decomposed into sub-systems that are usually represented using sub-actors. These sub-actors can be mapped to software agents, each of which can have specific capabilities. In doing so a global Architecture for the system emerges.

Detailed Design:
In this phase the the capabilities and interactions between the agents is specified usually in form of UML activity and sequence diagrams.

Implementation:
In this final phase the models are implemented via a mapping between the detailed design models and the implementation platform that has been selected.

2.3 - Key Terms

Actor:
An Intentional entity. Can be an Agent, a Role or a Position(human or software)

Goal:
A strategic interest of an actor . Can be a hard-goal or a soft-goal. A hard-goal is something that is more definable, where as a soft-goal does not have such a clear cut definition.

Task (or Plan):
A set series of actions that, when executed work to achieve a goal.

Resource:
A Physical or informational entity

Social Dependency:
An actors dependence on another actor to accomplish a goal, execute a task, or deliver a resource

Capability:
An actors ability to execute a plan or series of tasks to fulfill a goal.

3.1 - Tool Selection:

Selecting a tool for Tropos was not an easy task and I had to go through a few to figure out which one I was most comfortably with.

The selection criteria took into account several factors:

- Ease of use of the tool
- Its graphical interface
- Dynamism and stability of tool
- Documentation and support available on-line
- Consistency with rules of Tropos

3.2 - Comparison of tools based on these criteria:

Tool name	**Ease of use**	**Graphical modeling interface**	**Dynamism and stability**	**Documentation available on-line**	**Consistency with rules of Tropos**
SecTro (Secure Tropos)	The tool looked fairly easy to use and had a nice interface.	The modeling interface was good. Color coding for different objects was pretty good	This is where the software lacked a bit. The diagrams weren't stable and would get messed up very easily. Most annoying of all – it had no undo feature	There was one available document for support that was quite detailed for the first half of the development phase, but not so much for the second half, which included Architectural design, detailed design and implementation	Maybe due to lack of stabiliy, the sofware would sometimes fail to live up to the rules of Tropos, throwing an error when I tried to make tropos-legal connections. Also it was aimed more at Secure Tropos – which is an extension to regular Tropos, which I wanted to use.

OpenOme	Tool was generally easy to use up till the late requirements phase. After that it got a bit complicated and the documentation didn't help much.	The interface was nice and clean. No color coding however. But it allowed the bending of arrows, which could be leveraged to make cleaner designs.	As it was a plugin for eclipse, it was a lot more stable than SecTro.	There wasn't much detailed documentation available.	Was just a general i* tool, and apparently didn't support Tropos development.
TAOM4E	Again, a plugin for eclipse. Had a clean interface and was easy to use for all the levels. Each level had a clear demarcation and the process followed in a natural way.	The graphical modeling interface was also pretty good. It was well color coded and intuitive to use.	This was the most stable and dynamic of all three. User could drag and drop from previous stages and copy paste as well, making it easy to use.	There was a lot of reference material available on their website, as well as external references in papers about tropos	In general it seemed compliant with Tropos, however there was one inconsistency. It wouldn't allow tasks to be decomposed into anything except other task, whereas tropos allows tasks to be decomposed into goals and resources as well.

With these criteria in mind I selected **TAOM4E** for the modeling process. However, not before trying out the others and spending quite some time on the other two. SecTro especially, with its deceptive smooth looks had me working on it for quite some time, until I eventually decided it wasn't doing a very stable job.

4. Working with TAOM4E

(Tool for Agent Oriented visual Modeling for the Eclipse platform)

4.1 - Early Requirements:

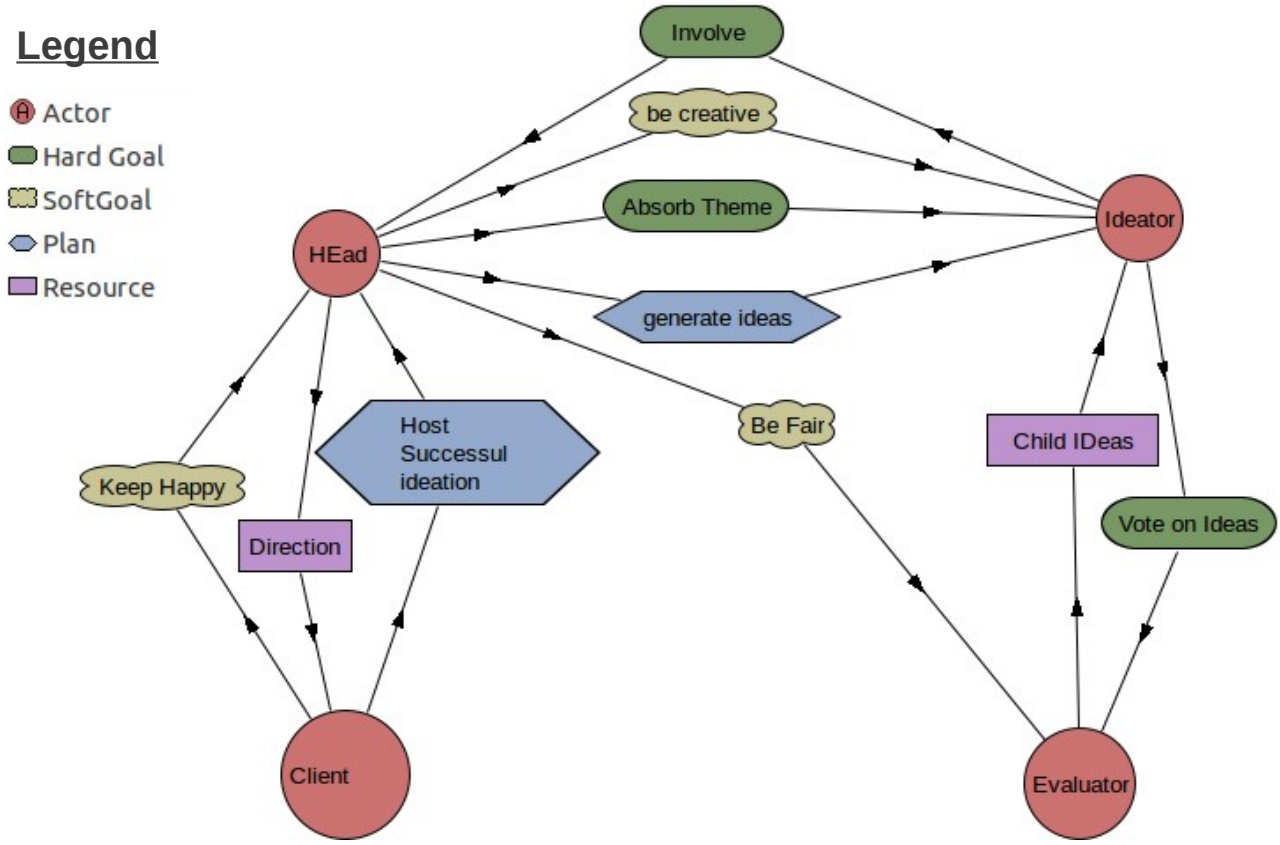
Having selected the tool for Tropos I started modeling the early requirements. The key activity in this phase is to identify all the major intentional actors and stakeholders and model the dependencies between them. [4]

The Key Stakeholders that I identified for Ideator are as follows:

The Head: The facilitator of the meeting. He is in-charge of posting the idea theme to the software and moderating the process.
The Ideator: The persons who will add ideas to the main theme and extend the idea in various directions.

The Evaluator: These are also ideators, taking on the role of evaluating the child ideas posted by ideators.
The Client: The client, for whom the ideation is being done. He will provide the main direction to the Head, who will then post it on the system.

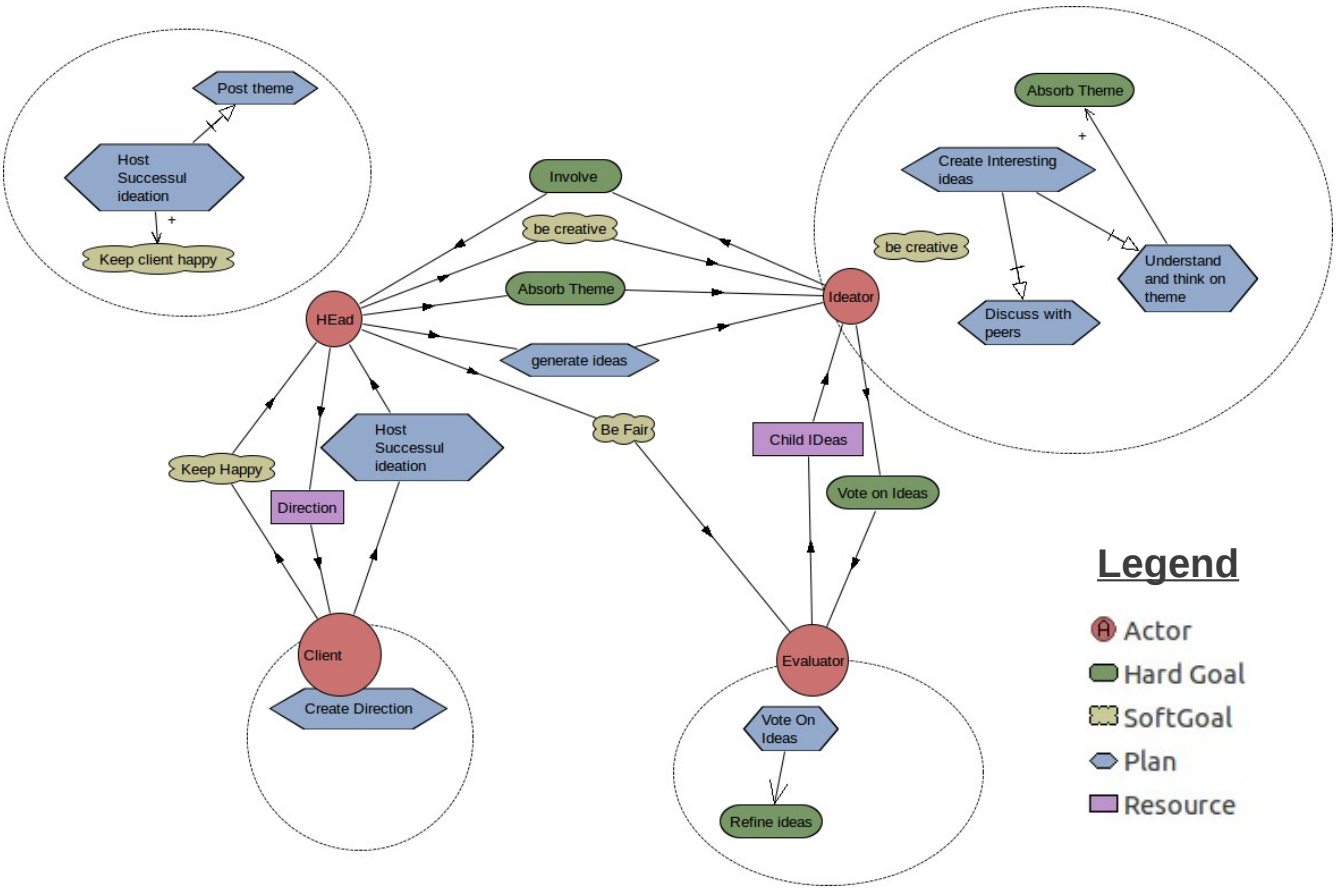
Once the actors have been identified, the dependencies between these actors have to be identified. The resulting model is the Strategic Dependency model.



Some of the Dependencies between actors are as follows:

- The Head depends on the Ideator on the goal to Absorb and understand the theme
- The Ideator depends on the Head on the goal to to involve everyone in the ideation.
- The Ideator depends on the evaluator on the task to vote on the ideas he creates, while the evaluator as a result depends on the Ideator for the resource 'child ideas' so he can vote on them.
- The Client depends on the Head on the soft goal of keeping him happy.

Once a basic Dependency Model has been created it is further expanded by looking at each actors inner intentions and behaviors. This results in the Strategic Rationale Model.



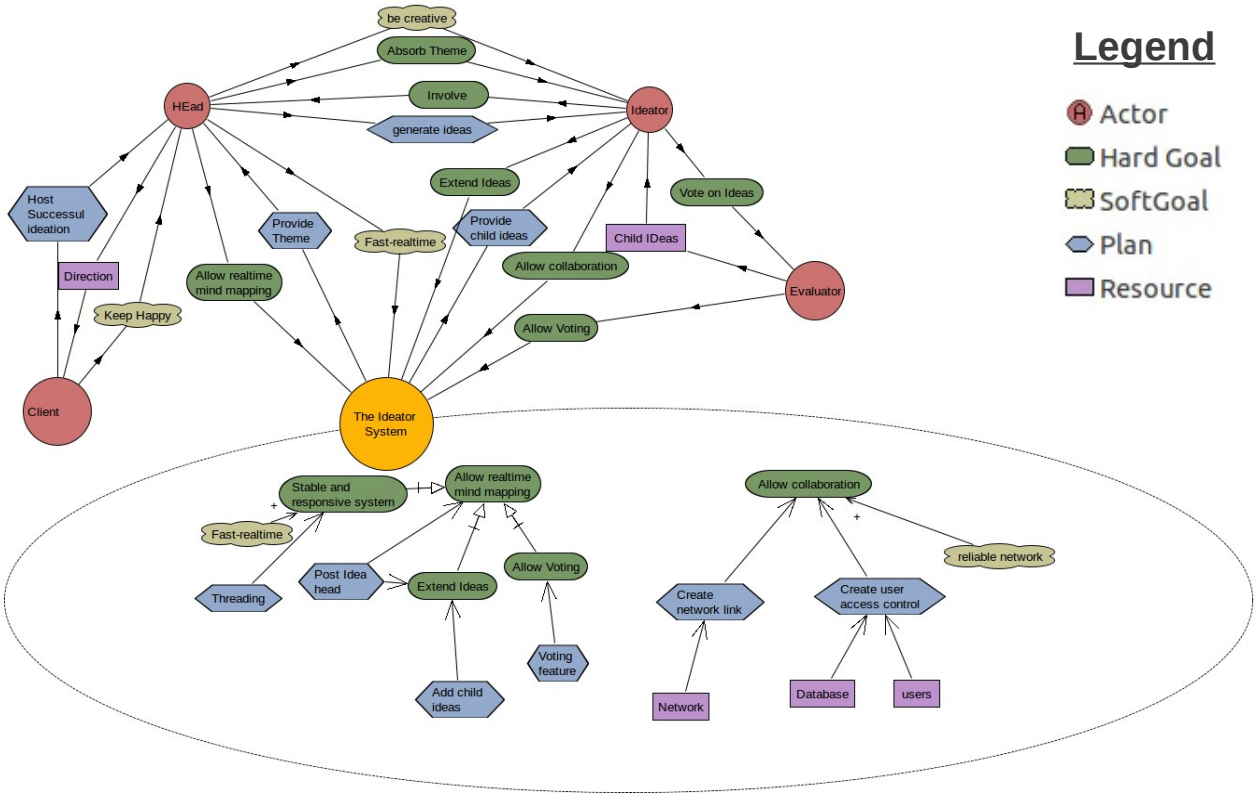
The SR model gives us the intentional description of the individual actors and stakeholders, giving us the rationales behind their actions.
This gives us a way to analyze the the intentional relationships, such as means-ends relationships of various goals and tasks that are internal to the actor.

Some of the Internal Rationales are as follows:
The Ideator wants to generate interesting ideas – this is a task that can decomposed into tasks such as discussing ideas with peers and understanding the theme and applying some creative methodologies to it. The latter also contributes positively to the goal of absorbing the theme, something the Head depends on the Ideator to do.
Another example is that of the evaluator, who votes on ideas by Ideators and this is a means-to-the-goal of refining ideas.

4.2 - Late Requirements:

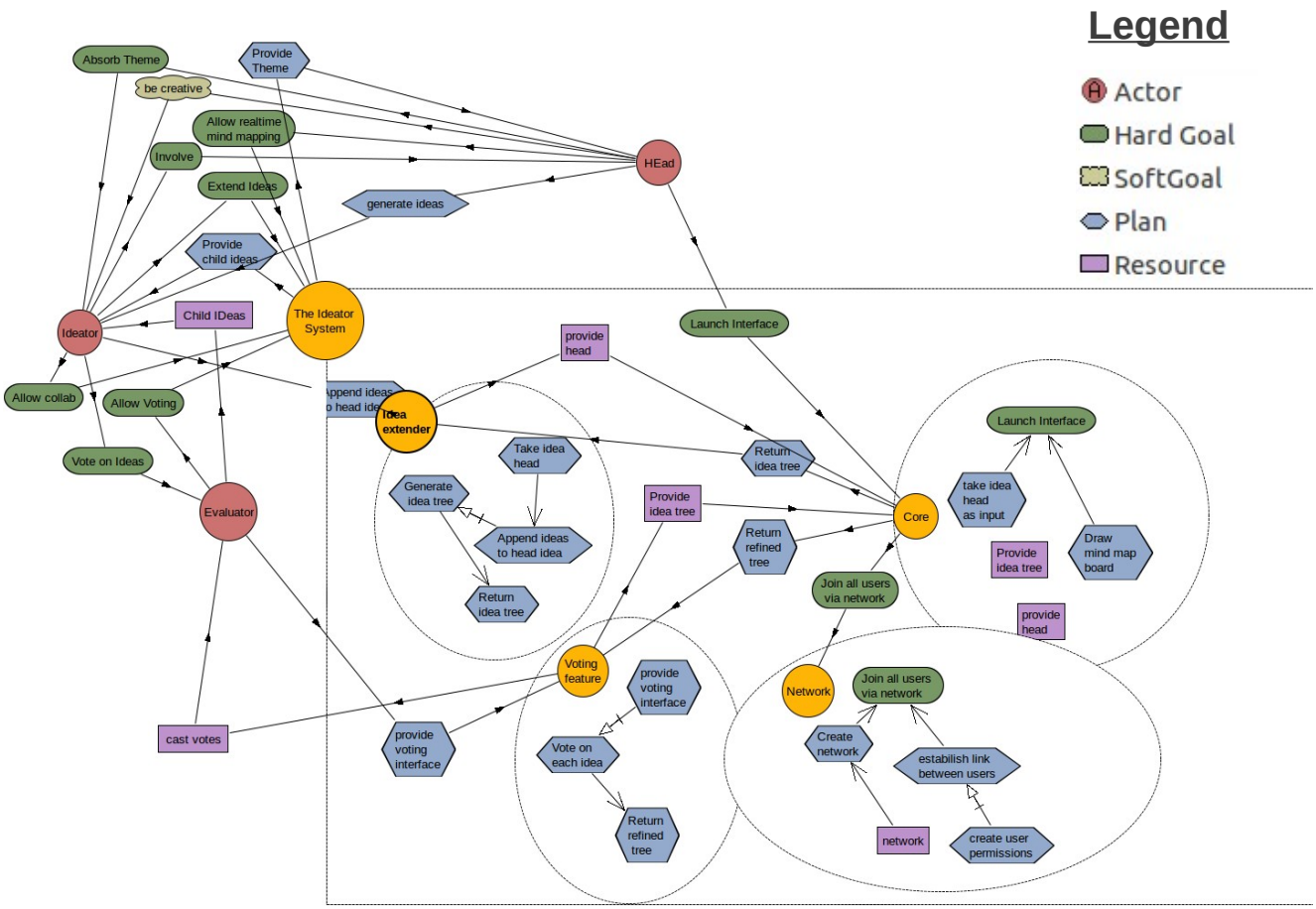
Once the Strategic Rationale Model is developed we can move on to the Late Requirements phase, where the system actor is added to the model. The system is modeled like any other actor participating in the environment. First the dependencies between the different actors and the system are modeled identifying system's functional and non-functional requirements . These dependencies can run both ways, i.e. the actors can depend on the system for goals, tasks and resources and vice versa.

Once these dependencies have been modeled, the system actor is expanded and its various internal goals and tasks are identified. These goals and tasks are then further decomposed and means-ends and contribution analysis is performed.



4.3 - Architectural Design:

Now that the system actor and its internal representation has been modeled, we can further expand it and model it in terms of sub-actors.
The dependencies and relationships between these sub-actors and the external actors are also modeled.
These sub-actors are delegated the subgoals of the system. The system's goals are further analyzed. Each of the sub-actors has internal roles and goals which are then modeled.

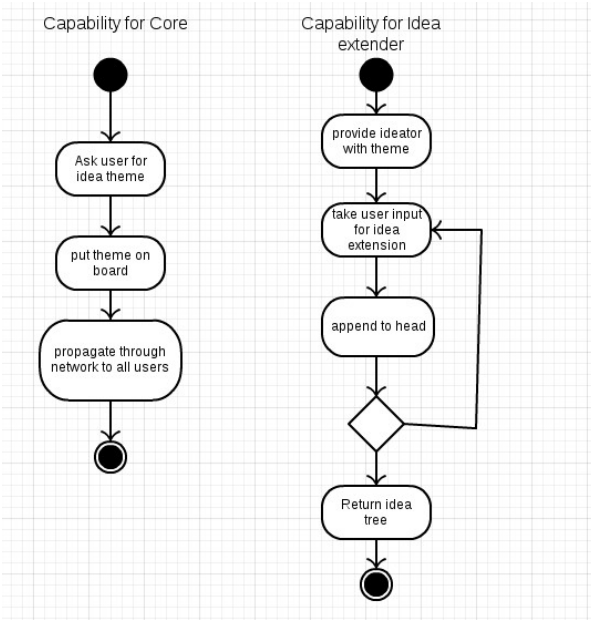


The capabilities for each of these goals and related tasks are then specified in terms of means-ends analysis.

In the table we can see some examples of capabilities. Lets take Core-c1 as an example. This is a capability of the agent Core. This specific capability tell us that first the core must Draw the mind map board, which is where the mind map will be held which is a means to the end of launching the interface for the users to work with.

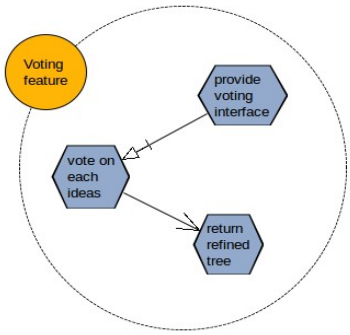
Agent	Capability	Means_End(goal,plan)
Core	core-c1	Draw mind map board, Launch Interface
Core	core-c2	take idea head as input, Launch Interface
Core	core-c3	Add child ideas to head, Extend ideas
Voting feature	Voting-c1	provide voting interface, Provide voting interface
The Ideator System	Id-c5	Create network link, Allow collaboration
The Ideator System	id-c6	Create user access control, Allow collaboration
Network	Net-c1	Create network, Join all users via network
Network	Net-c2	estabilish link between users, Join all users via netv

4.4 - Detailed Design:



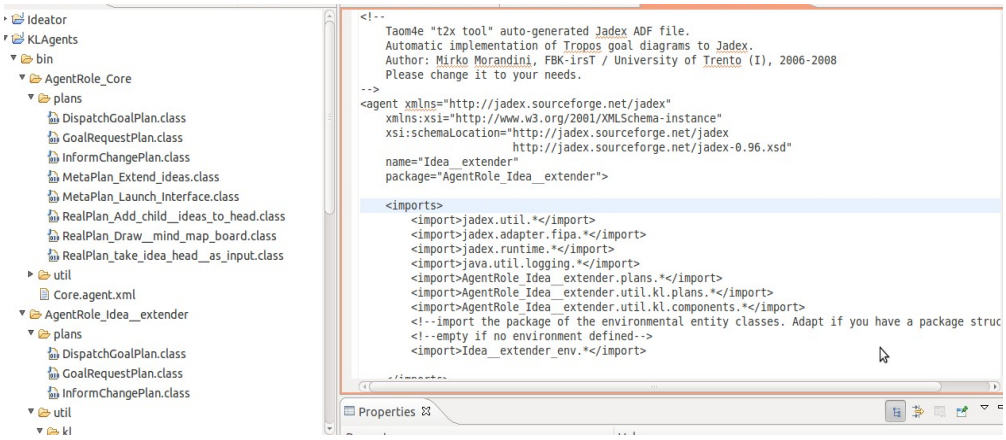
Once we the basic capabilities have been realized, we can then model capability diagrams using UML activity diagrams. To do so in eclipse the UML 2.0 plug-in must be installed. However when using t2x with TAOM4E, making capability diagrams is not imperative.

TAOM4E instead allows you to zoom into and independently look at each sub-actor and their tasks and goals and refine these goal models to the tee. Once you do this TAOM4E allows you to generate that agents code.



TAOM4E goes on to use the t2x tool and Jadex (a Java and XML composite) to generate code for the agents and their capabilities defined so far.

4.5 – Implementation:



TAOM4E uses t2x, a plugin which generates Agent-oriented code (based on Belief-Desire-Intention architecture) by mapping goal models to the code. [5]

T2x analyses the goals and plans created by TAOM4E and generates Agent Definition Files which can then be run on the Jadex platform, that comes with TAOM4E. Furthermore, Java skeleton files are created which layout the agents reasoning mechanisms to execute the correct plans to achieve desired goals.

The focus of this paper was to look at the modeling process up to detailed design with little focus on the implementation aspect. I have given a quick overview of how the code generation actually works

and what tools and models it uses. I also generated some skeleton goal and plan files for select agents.

The issues I ran into when trying to generate and execute code were mostly related to lack of unified documentation. All Tropos and TAOM4E documentations would give support up till the architectural design phase. There was very little explanation on how to actually generate the code and execute it. The information that was available was sparse and relied heavily on inference in some cases.

5. Conclusion

5.1 – Summary:

Over the course of the paper we discussed Ideator being modeled in the Tropos modeling language using the TAOM4E tool. We started with gathering and modeling the early requirements by identifying the stakeholders and modeling their dependencies. We further expanded the intentionality of these stakeholders and looked more closely at their rationales for their actions.
Then we moved onto the late requirements and we modeled the system and its intentions as an actor. Once the system and its intentions take shape we moved into the architectural design phase and modeled the internal goals and plans of the system as being handled by separate internal sub-actors. We also define each of these actors' capabilities, which are then further modeled in the detailed design as UML activity diagrams. At this stage the implementation platform is also taken into consideration and then these models can be implemented as Jadex agents using the t2x tool.
Over all the experience was an interesting one, since Ideator very nicely fits into the domain of social collaboration. It was interesting to see visually how the different actors would act in such a system and actually helped model some of the aspects that were not so apparent before.

5.2 – Limitations:

The work presented in this paper came with a few limitations.
First, i* and Tropos were completely new to me and I had no prior training with it, which caused me to having to do the initial model more than a few times. I also had no clear idea on which tool was the best and had to spend some time on the tool selection process. Since the aim of the paper was to highlight the process and experience of using Tropos, I have no other experience to compare it and therefore I can't draw comparisons and effectively comment on various aspects of the tool I used.

5.3 - Future Work:

The next step would be to carefully study the design phase and model the capabilities and agent plans more thoroughly so that they can be implemented in Jadex using t2x. There is still the need to revisit the architectural design and make it more comprehensive by adding further actors that will contribute to the system.

References:

1. Yu, Eric. Social modeling and i*. Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos, 2009.
2. B. Davide, P. Anna, S. Angelo, M. Haralambos, The Tropos visual modeling language. A MOF 1.4 compliant meta-model.
3. Bertolini, Delpero, Novikau, Orler, Penserini, Perini, Susi, Tomasi , A Tropos Model-Driven Development Environment , ITC-IRST, Via Sommarive 18, I-38050, Trento, Italy
4. M. Mirko ,N. Duy, P. Loris, P. Anna, S. Angelo: Tropos modeling, code generation and testing with the Taom4E tool *iStar*, Vol. 766CEUR-WS.org (2011) , p. 172-174.
5. M. Mirko – http://selab.fbk.eu/morandini/home.html, FBK-Irst Trento / Universita' di Trento