Thanks. Daniel has introduced the agent and goal-oriented approach more from the agent side. Now I’ll talk about the goal side of the approach. So basically, our paper is about how the goal-oriented approach can be used together with a scenario-based approach in requirements engineering and architectural design.
Motivations of this work includes:

(1) we want to strengthen the connections between requirements and architecture.

(2) we want to overcome the deficiencies and limitations of goals and scenarios when they are used in isolation.

The work is more directly motivated by a submission to ITU study group 10 on URN- user requirement notation.

This notation is composed of two languages, one is a GRL -a goal oriented requirement language, the other is UCM- a scenario-based high level architecture description language.

Coupling the two languages make it possible to visualize the refinement processes from abstract descriptions to increasingly concrete design alternatives, to detect new requirements during design, and to explore new alternatives and to make tradeoffs among them.
First, I am going to briefly introduce the concepts in GRL and UCM, then go over an example modelling process.
GRL is a language supporting goal and agent oriented modelling and reasoning about requirements, especially for dealing with NFRs.

There are three categories of concepts in GRL graphical notation: 

- intentional elements
  - goal, task, resource, softgoal, belief
- intentional links
  - means-ends, decomposition, contribution, correlation, dependency
- actors
This is a GRL model made up of a single goal structure.

The goal structure can be divided into three sections. (Point to the graph) The first section models the goal refinement process. Here a high-level abstract softgoal is refined into several lower-level concrete softgoals. The second section models the operationalization process. Here goals are linked with alternative means of achieving them. The third section shows the side effects of alternatives, and designer’s argumentations about this model.

Softgoals are used to describe NFRs, tasks are used to represent the particular operationalizations of softgoals, goals are used to refer to functional requirements. Beliefs record the designer’s rationales and argumentations.
Goal structures may also be distributed amongst actors. In this case, they are connected by dependency links across actor boundaries. As we are focusing more on the goal side, I won’t explain this model in detail.
UCM Introduction (Use Case Maps)

- Scenarios describing causal relationships between responsibilities
- UCM scenarios can be allocated to abstract components
- Useful to describe features visually

UCM is a graphical notation for high-level architectural design. Scenarios in UCM vary from the scenarios in RE literature in their forms, purposes and usage. UCM is esp. good for telecommunication domain.

UCM Scenarios are paths illustrating causal relationships among responsibilities.

A scenario path is a wiggly line connecting start points, responsibilities and end points. Responsibilities are crosses representing actions, tasks or functions to be performed. Start points are the preconditions of the scenarios, end points are the postconditions of the scenarios.

UCM may also provide an integrated view of behavior and structure by allocating scenario paths to abstract components.

Components are entities or objects composing the system. If a responsibility is inside a component, it means that the component is responsible for performing the action.

The example shows a simple scenario. A user A attempts to call user B through the switching system. She first sends a connection request to the system. This request causes the switching system to check whether a valid number is dialed, then verify whether the called party is idle or busy, if he is idle, some status will be updated, and a ring signal will be activated on B’s side.
UCM also has mechanisms to represent more complex scenarios. For example, forks and joins are used when there are alternatives and shared segments in the path. Plug-ins and stubs can be used to represent sub-maps and their containers.
We suggest that goal modelling and scenario modelling proceed in parallel. The two processes interact when there are new requirements or design decisions discovered.
After the original information about the intended system is provided, the two processes may start respectively.

In the goal modelling side, the functional requirements on a mobile switch center is represented as a goal, “narrowband and wideband voice, data and image services be provided”. The two NFRs identified are softgoal “maximize call capacity” and softgoal “minimize cost”.

In the other side, the essential scenario implementing the function is also defined.
By binding responsibilities to components, the existing TDMA architecture is shown in this model. (Pointing to the bound use case map,) where the decoder of voice coder is located in the base station.

This is the first design alternative considered, we use a task node in the GRL model to refer to it. As we evaluate this architecture, we find that two new NFRs need to be considered: high voice quality and performance. These newly detected softgoals are added into the GRL model and further refined. Then the contributions of current architecture to these requirements are evaluated in the goal model.
As the result of the evaluation is that the requirements can not be sufficiently satisfied by this architecture, new solutions need to be explored. One possibility is to change the binding of voice decoder from base station to the switch center. This time most of the NFRs are satisfied by the new architecture, although it may have some negative side-effects to system complexity. If this architecture is granted, the modelling process of both sides may stop. Otherwise, other alternatives need to be considered.
This modelling process is iterative and incremental. Both top-down and bottom-up. The models in the given example is quite simple, alternatives of architectures are not limited to different ways of responsibilities binding. Other complicated changes of run-time behaviors and component structures can also be analyzed with these two languages.
**Related Works**

- **RE**
  - Van Lamsweerde and Willemet’s work on using scenarios for requirement (goal) elicitation
  - Rolland’s CREWS-L’Écritoire Approach

- **Architecture**
  - Kazman’s Software Architecture Analysis Method
  - Krutchten’s 4+1 model of software architecture

In RE literature, the combined use of goals and scenarios is primarily for eliciting and validating of software requirements.

Van Lamsweerde and Willemet’s work explored the process of inferring formal specifications of goals and requirements from scenario description. Rolland’s approach also focus on the elicitation of functional requirements and goals.

In architecture area, Kazman’s Software architecture analysis method is a scenario method for evaluating architecture. The scenarios are more a way of testing rather than a model of an architecture. It is more an after the fact evaluation than a pre design evaluation according to requirements.

Krutchen’s 4+1 model first address the concerns of various stakeholders separately with 4 views. And then use a scenario view to link them together.
Conclusion and Future Works

- This preliminary effort shows that:
  - Goal-orientation and Scenario-orientation complement to each other in both RE and architecture design
  - NFRs are inevitably the criteria of architectures tradeoffs
- Future directions
  - Tighter coupling of goals and scenarios in notation level
  - Better guidance and tool support on the modelling process
  - Improve the flexibility of GRL: GRL+UML, GRL+X
More Information?

- GRL web site:
  http://www.cs.toronto.edu/km/GRL

- UCM web site:
  http://www.usecasemaps.org