Why Use Domain Models in Software Engineering?

- Abstracts reality
- Facilitates communication and understanding
- Reveals gaps in knowledge

- Promotes domain analysis, “what if…?”
- Helps elicit requirements
- Facilitates high-level design and redesign
Uses Goals, Softgoals, Tasks, Contributions, Actors, Actor Boundaries and Dependencies to explain the motivations and intentions behind socio-technical systems.
We can **Purchase PC Products** or **Obtain them from a Data Pirate**.

Should the **PC Product Provider Allow Peer-to-Peer Technology**?

How can we analyze the effects of these decisions?
i* Analysis

- i* models can be extremely complex.
- Analysis without a methodology or tools is difficult.
i* Qualitative, Interactive Evaluation

Procedure

• Give evaluation labels to initial elements.

• Procedure builds on evaluation in the NFR Framework.
Propagate labels using a combination of predefined rules and human judgment.
• The **PC User Obtains PC Products** somewhat **Affordably**, but the **PC Product Provider** has a conflict value for **Sell PC Products for Profit**.

• Is the model accurate? Is there a better alternative?

• Further iterations of modeling and analysis are needed.
Evaluation Implementation: OpenOME

Eclipse-based modeling tool, built on EMF and GMF

Trial Domain: Kids Help Phone

- Applied in a large case study involving Trusted Computing.
- Applied in a strategic requirements analysis project for the Canadian Youth Counseling Organization, Kids Help Phone.
Future Directions

• Inspired by Maiden et al.’s use of Satisfaction Arguments, systematically capture the rationale for evaluation decisions, including domain assumptions, in textual arguments attached to the model.

• Incorporating aspects of formal model checking in ways which are near “invisible” to the user, keeping the simplicity and usability of i* modeling constructs.

References


• J. Horkoff, An Evaluation Algorithm for the i* Framework, (Master’s Thesis), Department of Computer Science, University of Toronto, 2006.
