Specifying and Analyzing Early Requirements: Some Experimental Results

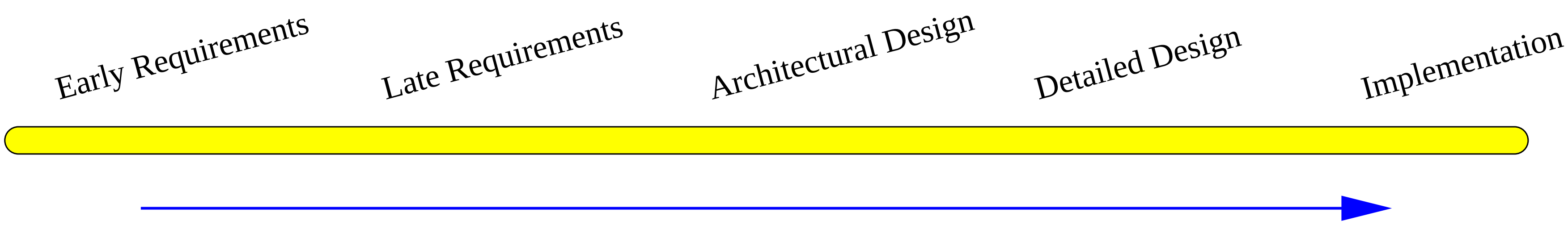
Marco Roveri
joint work with A. Fuxman, L. Liu, J. Mylopoulos and M. Pistore

roveri@irst.itc.it

ITC-irst, Via Sommarive 18, 38050 Povo, Trento, Italy

The Formal Tropos Project

- The *Tropos* project aims to the development of and *Agent-Oriented* software engineering methodology, the *Tropos Software Development Process*, supported by a variety of analysis tools.



- The *Formal Tropos* project aims to an effective integration and harmonization of Formal Methods in the Tropos Software Development Process. It builds on. . .
  - *i**, a framework for modeling social settings, based on the notions of actors, goals, dependencies...
  - KAOS, a goal-oriented requirements framework that provides a rich temporal specification language.
  - NuSMV, a (symbolic) model checker initially developed for the verification of hardware systems.

Model Checking Early Requirements [RE01]

- Formal Methods (FM) are usually applied in advanced stages of the development process, and their application in Early Requirements is by no means trivial:
  - FM amounts to validate an implementation against requirements;
  - FM require a detailed description of the behavior of the system;
  - FM concepts are not appropriate for Early Requirements.
- Formal Methods, and in particular *Model Checking* cannot be used to prove correctness of the specification.
- However they can. . .
  - show misunderstandings and omissions in the requirements that might not be evident in an informal setting;
  - assist the requirements elicitation by helping in the interaction with the stakeholders;
  - add expressive power to the requirements specification formalism;
  - enable proof of correctness in advanced development phases.

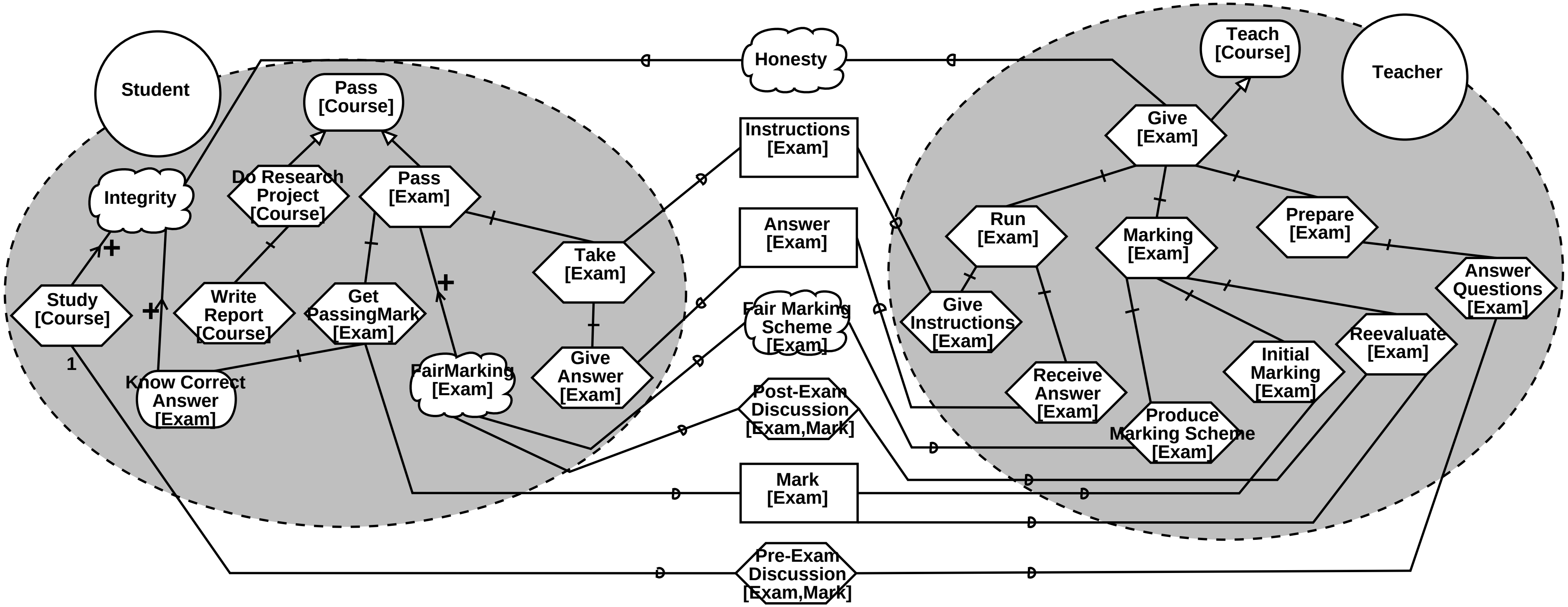
Outline

- The original contribution.
- The methodology.
- The T-TOOL
- The experimental analysis.
- Conclusions and Future Work.

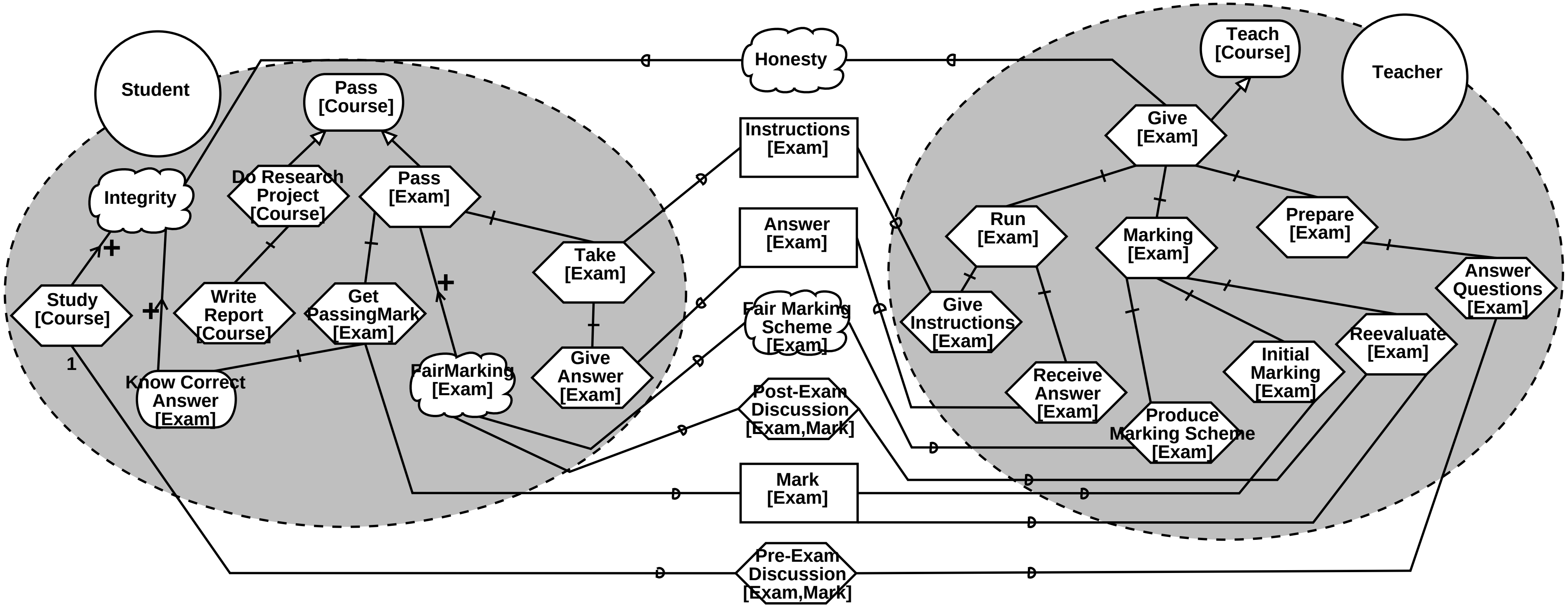
Our contribution

- In this paper we focus on applying model checking in early requirements analysis.
- We build on the results in [RE01].
- The original contribution:
  - Enriched the *i** notation (e.g., Prior-to links, cardinality constraints)
  - Heuristic rules to automatically extract a Formal Tropos model from the enriched *i** model
  - A methodology to use the most effective model checking techniques for the analysis of FT specifications
  - A tool supporting the methodology (T-TOOL).
  - Experimental evidence of the effectiveness of the approach.

The course-exam management case study

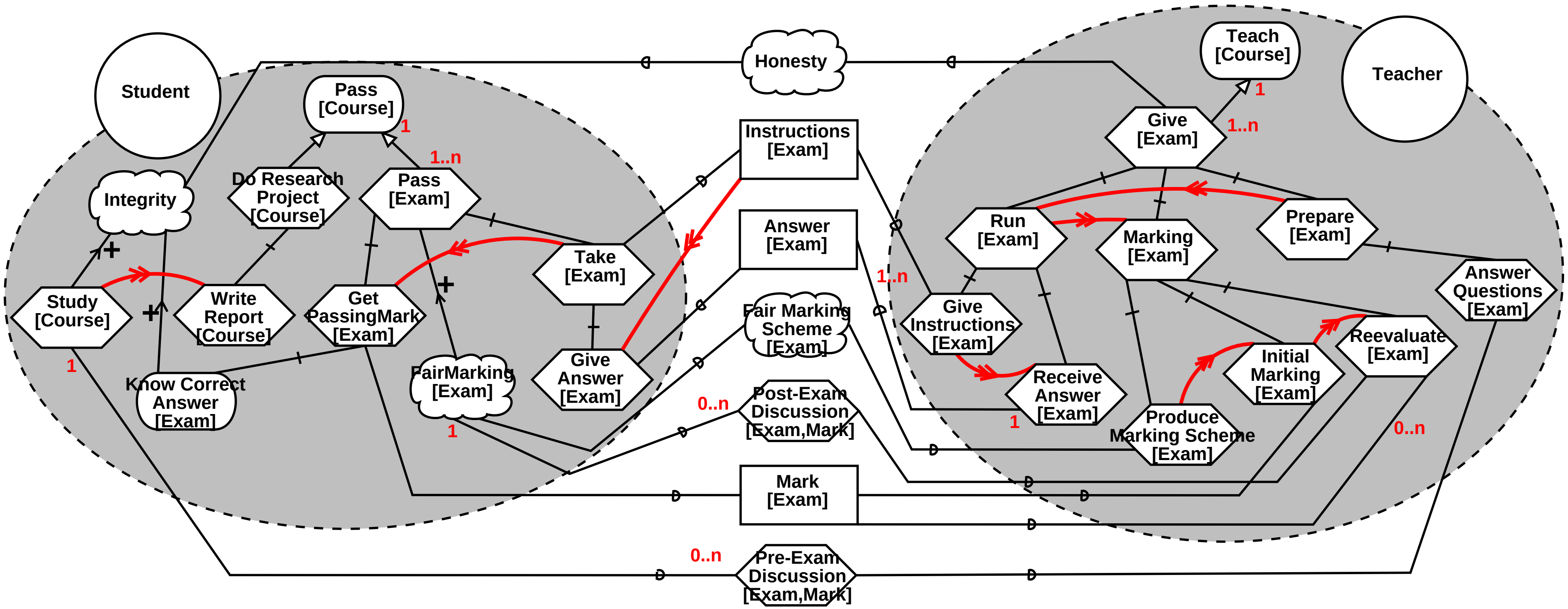


The course-exam management case study



- there are different instances of actors, goals, dependencies, and relations among these instances
- strategic dependencies have a temporal evolution (they arise, they are fulfilled, there is an order,...)

The course-exam management case study



- there are different instances of actors, goals, dependencies, and relations among these instances
- strategic dependencies have a temporal evolution (they arise, they are fulfilled, there is an order,...)

The course-exam management case study (II)

Entity **Course**

Entity **Exam**
  Attribute constant **course** : **Course**

Actor **Student**

Goal **PassCourse**
  Mode achieve
  Actor **Student**
  Attribute constant **course** : **Course**

Actor **Teacher**

Task **GiveExam**
  Mode achieve
  Actor **Teacher**
  Attribute constant **exam** : **Exam**

Resource Dependency **Answer**
  Mode achieve
  Depender **Teacher**
  Dependee **Student**
  Attribute constant **exam** : **Exam**

Resource Dependency **Mark**
  Mode achieve
  Depender **Student**
  Dependee **Teacher**
  Attribute constant **exam** : **Exam**
    passed : boolean

Softgoal **Integrity**
  Mode maintain
  Actor **Student**

- FT emphasis is in modeling the “strategic” aspects of the evolution elements.
- FT focus is on the **creation** and **fulfillment** central moments of elements.
- FT allows the designer:
  - to specify different modalities for the fulfillment of elements.
  - to specify temporal constraints on the creation and fulfillment of elements.

The course-exam management case study (III)

Goal **PassCourse**

Mode achieve

Actor **Student**

Attribute constant **course** : **Course**

Fulfillment definition

/* OR decomposition */

$((\exists\ \mathbf{e} : \mathbf{Exam}\ (\mathbf{e.course} = \mathbf{course}) \wedge$

$\forall\ \mathbf{e} : \mathbf{Exam}\ (\ \mathbf{e.course} = \mathbf{course} \rightarrow$

$(\exists\ \mathbf{p} : \mathbf{PassExam}\ (\ \mathbf{p.exam} = \mathbf{e} \wedge \mathbf{p.pass\_course} = \text{self} \wedge \text{Fulfilled}\ (\mathbf{p})))))$

$\vee\ (\exists\ \mathbf{r} : \mathbf{DoResearchProject}\ (\mathbf{r.pass\_course} = \text{self} \wedge \text{Fulfilled}\ (\mathbf{r}))))$

/* cardinality constraint */

$\wedge\ (\neg\exists\ \mathbf{p} : \mathbf{PassCourse}\ ((\mathbf{p} \neq \text{self}) \wedge (\mathbf{p}.\text{actor} = \text{actor}) \wedge (\mathbf{p.course} = \mathbf{course}) \wedge \text{Fulfilled}\ (\mathbf{p})))$

Formal Analysis of Early Requirements

Once a "satisfactory" Formal Tropos model of the requirements is available we can perform the following formal analysis:

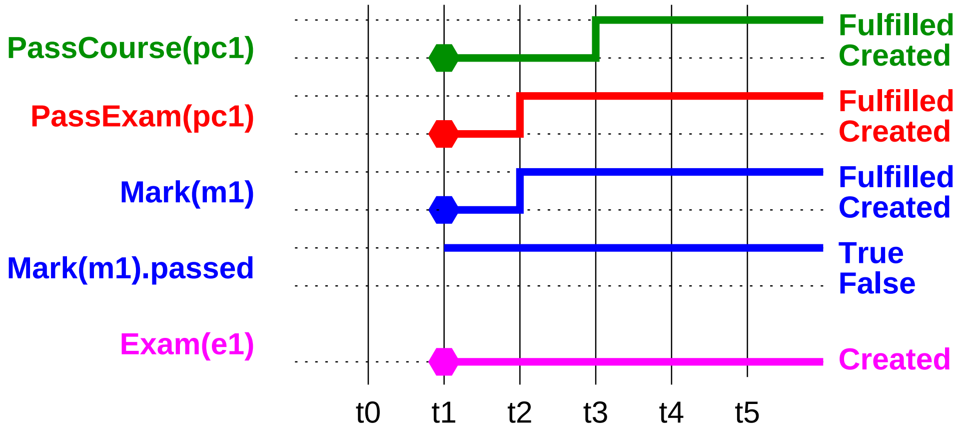
- consistency check: "the specification admits valid scenarios"
- possibility check: "there is *some* scenario for the model that respects certain possibility properties"
- assertion validation: "*all* scenarios for the model respect certain assertion properties"
- animation: the user can interactively explore valid scenarios of the model:
  - gives immediate feedback on the effects of the constraints;
  - makes it possible to catch trivial errors;
  - is an effective way of communicating with the stakeholder.

Possibility Checks in Formal Tropos

- A **possibility**:
  - describes *expected*, *valid* scenarios of the specification;
  - is used to guarantee that the specification does not rule out any wanted execution of the system.

    Global Possibility $\exists$ p : **PassCourse** (Fulfilled (**p**))

- The result of the verification is:
  - A witness scenario if the possibility is verified.



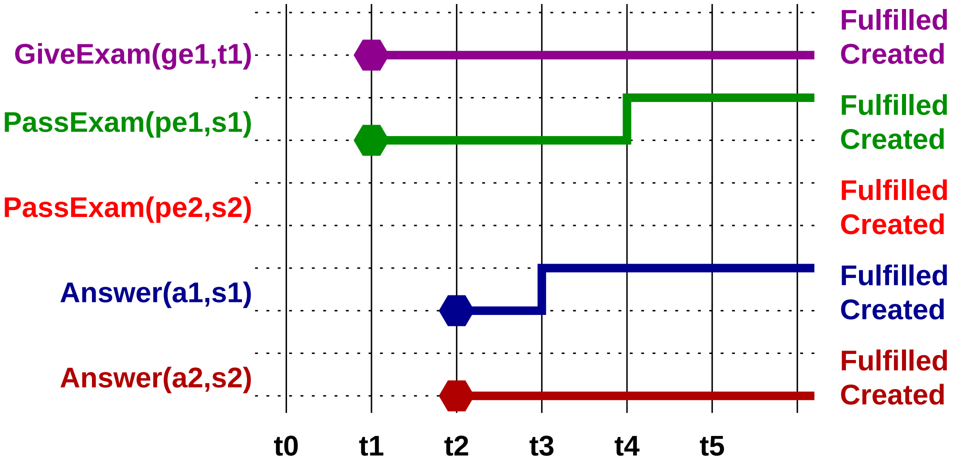
  - A negative answer if there is no scenario satisfying the possibility.

Assertion Validation in Formal Tropos

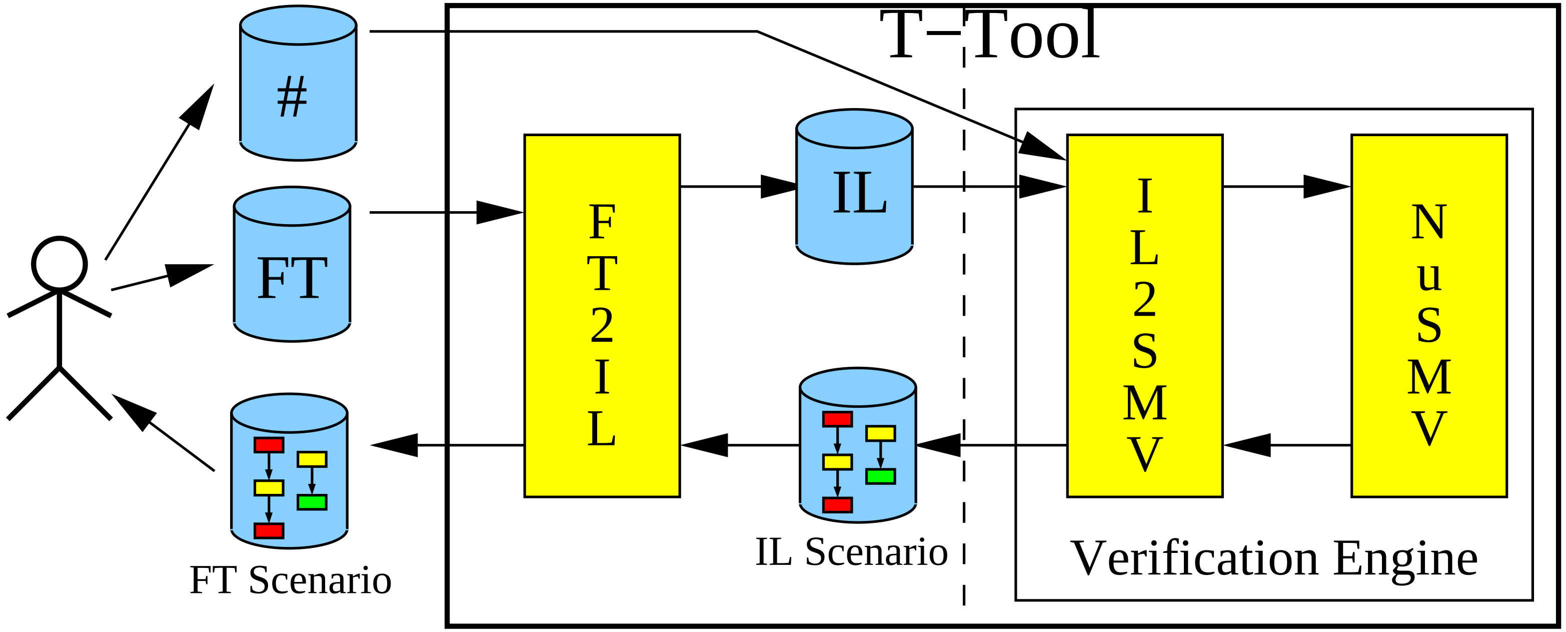
- An **assertion**:
  - describes *expected* conditions for all the valid scenarios;
  - is used to guarantee that the specification does not allow for unwanted scenarios.

    Global Assertion $\forall$ **a** : **Answer** (F $\exists$ **pe** : **PassExam** (**pe.exam** = **a.exam** $\wedge$ **pe**.actor = **a**.dependee))

- The result of the verification is:
  - A positive answer if the property is satisfied,
  - A counter-example scenario if the assertion is not satisfied.



The Supporting Tool: T-TOOL



The Model Checking Verification Engine

- It is based on the NuSMV symbolic model checker.
- NuSMV adopts symbolic model checking algorithms based on:
  - Binary Decision Diagrams (BDDs):
    - performs an exhaustive traversal of the model by considering all the possible behaviors in a compact way;
    - because of the exhaustiveness they are complete;
    - very expensive for large models.
  - Propositional Satisfiability (SAT), known as Bounded Model Checking (BMC).
    - Looks for a trace of given length that satisfies/falsifies a property;
    - more efficient than BDD for traces of reasonable length;
    - complete up to the considered length; a longer trace could falsify the property.
- NuSMV has been extended to allow for the verification of FT specifications:
  - An IL2SMV module to interface the IL with the NuSMV system;
  - SAT based BMC has been extended to deal with past operators;
  - An improved flexible interactive animator.

Which Verification Technique for What?

- Possibility (Consistency) checks amounts to identify a witness scenario for a given property.
  - These kind of properties appear to be more amenable to SAT based BMC techniques.
  - The length of the witnesses is usually reasonable (≤ 10).
- Assertion validation amounts to check whether all the admissible behaviors satisfy a certain property.
  - SAT based BMC can provide a quite immediate feedback on the truth of the considered property up to a reasonable length.
  - If SAT BMC does not point out flaws, then we can proceed with BDD based Model Checking to possibly confirm the result.
  - Model often too big to be efficiently handled by BDD based symbolic model checking techniques.

Which Verification Technique for What? (II)

- The problem: while verifying assertions, BDD based exhaustive techniques very often blow up when the system is too large.
- The solution: use of "standard" reduction techniques, e.g. abstraction techniques.
  - The general assertion validation problem: $\bigwedge_{i \in I} C_i \Rightarrow \varphi$
  - If we consider a $J \subseteq I$ if $\bigwedge_{i \in J} C_i \Rightarrow \varphi$ then $\bigwedge_{i \in I} C_i \Rightarrow \varphi$
  - If we fail with $J \subseteq I$ we need to choose another L such that $J \subset L \subseteq I$ and iterate.
    - SAT based BMC can give an immediate feedback on the truth of the reduced model, thus suggesting refinement of the constraints considered.
    - BDD based MC then will guarantee the truth.
    - Open to different verification strategies (BMC and BDD in parallel, the first that produce a result stops the other, . . .).

Experimental Results

- Following the devised methodology we conducted several iterations of experiments.
- At each iteration an FT specification was validated by consistency checks, and possibility and assertions verifications on different upper bounds of number of class instances.
  - Whenever a bug was found the FT specification was corrected and the approach iterated.
- Iterations ended when all the checks in the FT specification were successful, i.e. we had a “reasonable” specification.

Experimental Results (II)

Possibility Checks						
	1 instance		**1..2 instances**		**2 instances**	
	BMC	**BDD**	**BMC**	**BDD**	**BMC**	**BDD**
P1	Valid[3] 9.4sec / 29Mb	Valid[3] 1786sec / 64Mb	Valid[3] 55.7sec / 77Mb	Undecided T.O.	Valid[3] 860sec / 295Mb	Undecided M.O.
P2	Valid[3] 9.3sec / 29Mb	Valid[3] 1719sec / 63Mb	Valid[3] 55.6sec / 77Mb	Undecided T.O.	Valid[3] 842sec / 295Mb	Undecided M.O.
P3	Valid[4] 14.2sec / 38Mb	Valid[5] 1979sec / 64Mb	Valid[4] 94.9sec / 96Mb	Undecided T.O.	Valid[4] 1629sec / 375Mb	Undecided M.O.
P4	Undecided[10] 105sec / 84Mb	Invalid 1626sec / 64Mb	Undecided[10] 2143sec / 237Mb	Undecided T.O.	Undecided[4] T.O	Undecided M.O.

Experimental Results (III)

Assertion Checks						
	1 instance			**1..2 instances**		
	BMC	**BDD**	**BDD-reduced**	**BMC**	**BDD**	**BDD-reduced**
A1	NoBug[10] 100sec / 83Mb	Valid 1298sec / 64Mb	Valid 0.3sec / 2Mb	NoBug[10] 1086sec / 237Mb	Undecided T.O.	Valid 30.8sec / 4.2Mb
A2	NoBug[10] 111sec / 84Mb	Valid 1295sec / 64Mb	Valid 44sec / 17Mb	Invalid[3] 57.6sec / 77Mb	Undecided T.O.	Invalid[7] 757sec / 100Mb
A3	NoBug[10] 107sec / 83Mb	Valid 2110sec / 64Mb	Valid 2.5sec / 4Mb	NoBug[10] 2837sec / 234Mb	Undecided T.O.	Undecided T.O.
A4	NoBug[10] 114sec / 83Mb	Valid 1297sec / 63Mb	Valid 0.1sec / 2Mb	NoBug[9] T.O.	Undecided T.O.	Undecided T.O.

Analysis of results

For our case study the devised methodology…

- was effective in producing a FT specification of good quality;
- lead to a better understanding of the domain revealing tricky aspects (e.g. …)
- validation techniques provided by the T-TOOL verification engine were useful in detecting bugs, while animation was effective in early phases to point out trivial bugs.
- SAT based BMC techniques were very effective in answering to consistency and possibility checking.
- SAT based BMC is very effective in providing a confidence on the truth of assertions, thus preventing spending much effort in applying BDD based verification.
- Abstraction techniques are very promising, but need to be automated, defining heuristics to extract initial set of constraints and to refine them in case of verification failure.

Future Work

- Devise techniques guaranteeing correctness of an FT specification regardless of qualifications.
- Automate the verification process for assertions
  - developing techniques for choosing initial set of constraints and to refine them when reduced verification fails;
  - heuristics to automatically alternate phases where the tool tries to prove validity of a model, with phases where it looks for bugs.
- Improve model generation by exploiting possible symmetries in the specification.
- Develop a GUI allowing the user to write FT specifications and to inspect scenarios produced by T-TOOL as animation of the *i** diagrams.
- Extend the methodology to the further phases of the Tropos Software Development Process.

References

[RE01] *Model Checking Early Requirements Specifications in Tropos*. A. Fuxman, M. Pistore, J. Mylopoulos and P. Traverso. IEEE Int. Symposium on Requirements Engineering. 2001.

[JRE03] *Specifying and Analyzing Early Requirements: Some Experimental Results*. A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri and P. Traverso. Submitted to to Journal of Requirements Engineering. 2003.

[T-TOOL] http://sra.itc.it/tools/t-tool

[Tropos] http://dit.unitn.it/tropos

[NUSMV] http://nusmv.irst.itc.it