

# Requirements Models at Design- and Runtime

John Mylopoulos  
University of Trento

ER@BR at RE 2013  
Rio de Janeiro, Brasil  
July 16, 2013



[Chidi Okoye]

# Abstract

We review the history of requirements models going back to the seminal works of Douglas Ross on SADT. We also discuss the main ingredients of modeling languages in general, and requirements modeling languages in particular, including the set of primitive concepts they are founded on (their ontology), the language they offer for building models, and their semantics, defined in terms of an entailment relation over models. Finally, we sketch some of the desirable features (. . . “requirements”) of design-time and runtime requirements models and draw conclusions about their similarities and differences.

# Acknowledgements

This presentation is based on on-going joint work with Alex Borgida (Rutgers University), Fabiano Dalpiaz (University of Toronto), Jennifer Horkoff (University of Trento), Vitor Souza (UFES) and others on Adaptive Software Systems.

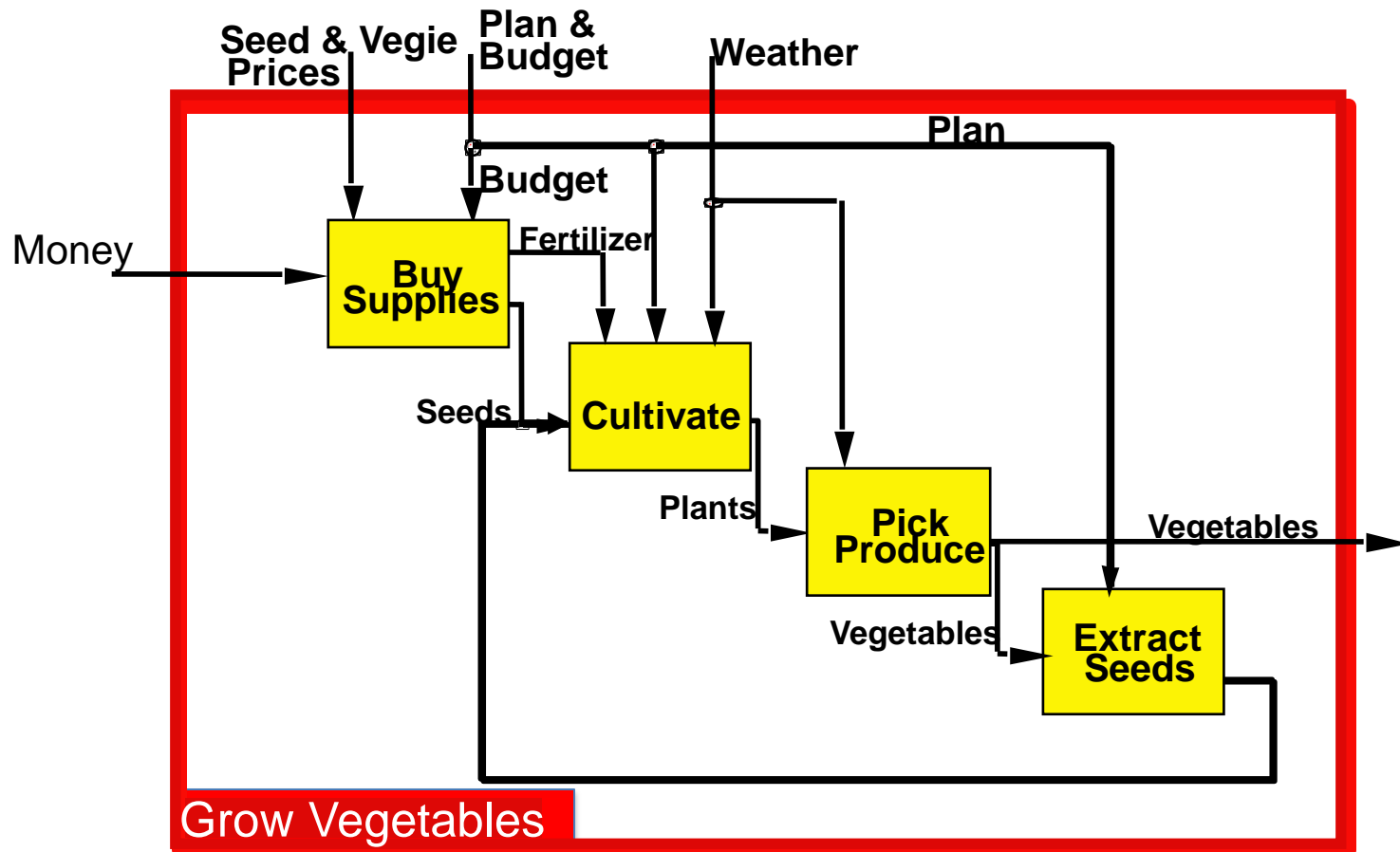


# Requirements Engineering (RE)

- 🌈 Concerned with the elicitation, analysis and refinement of stakeholder requirements in order to produce a specification for a system-to-be.
- 🌈 Founded on seminal works by Douglas Ross, Michael Jackson and others in the mid-70s.
- 🌈 Unique research area within CS because its task is not to solve problems, but rather to *define* ones.
- 🌈 Interesting area because stakeholder (“early”) requirements are necessarily vague, informal, self-contradictory, and more (... in short, "scruffy"), but they are requirements none-the-less.

# Origins

🌈 Requirements are activities/functions the system-to-be will perform within its operational environment (Douglas Ross, c.1977).



# The Requirements problem

🌈 In its original formulation [Jackson95], a requirements problem consists of finding a *specification*  $S$  for a given set of *requirements*  $R$  and *indicative environment properties*  $E$  such that

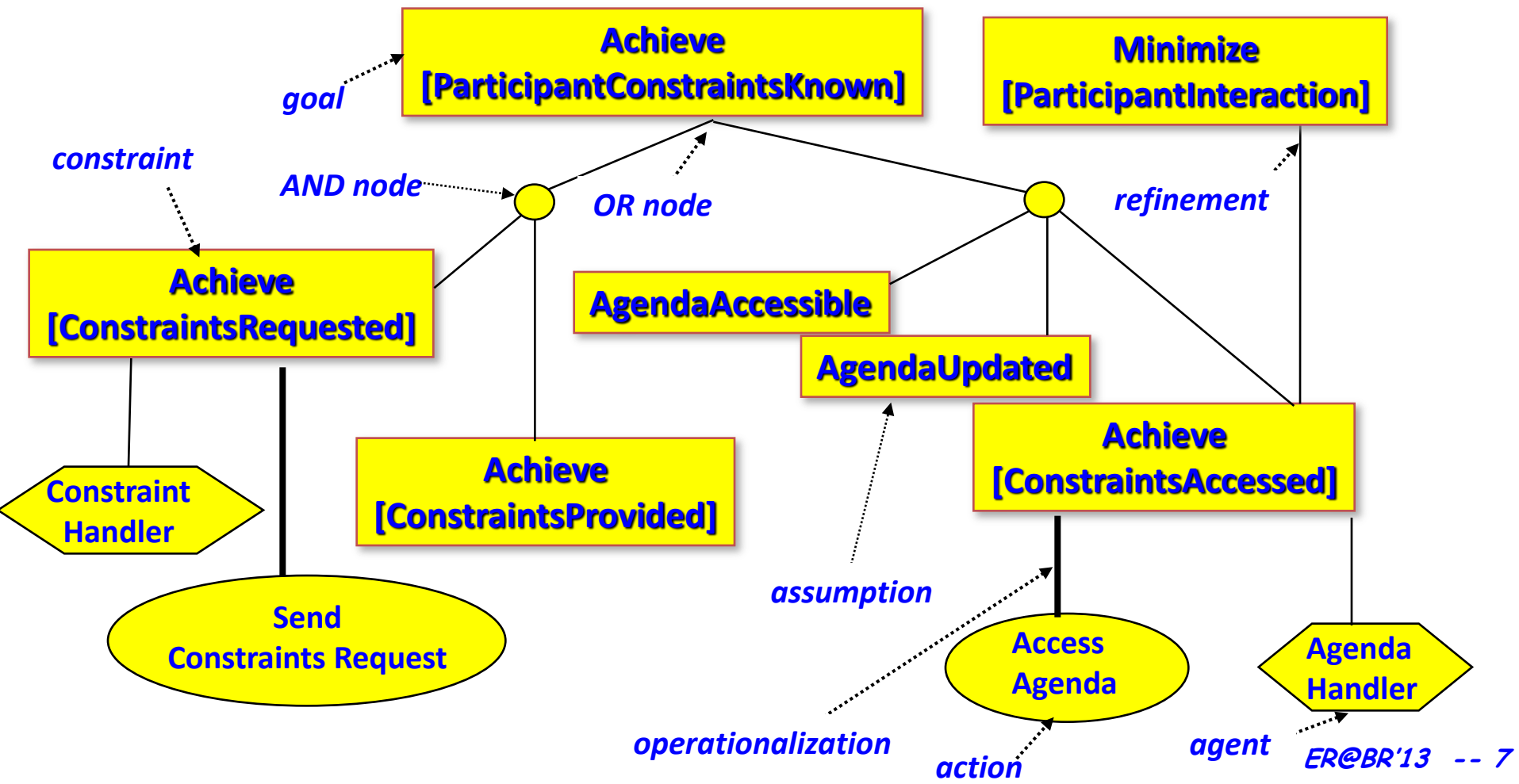
$$E, S \models R$$

meaning: “... satisfaction of the requirements can be deduced from satisfaction of the specification, together with the environment properties...” [Jackson95]

🌈 Solution through refinement (as in program refinement): Start with requirements and keep refining them to eliminate mention of non-executable elements.

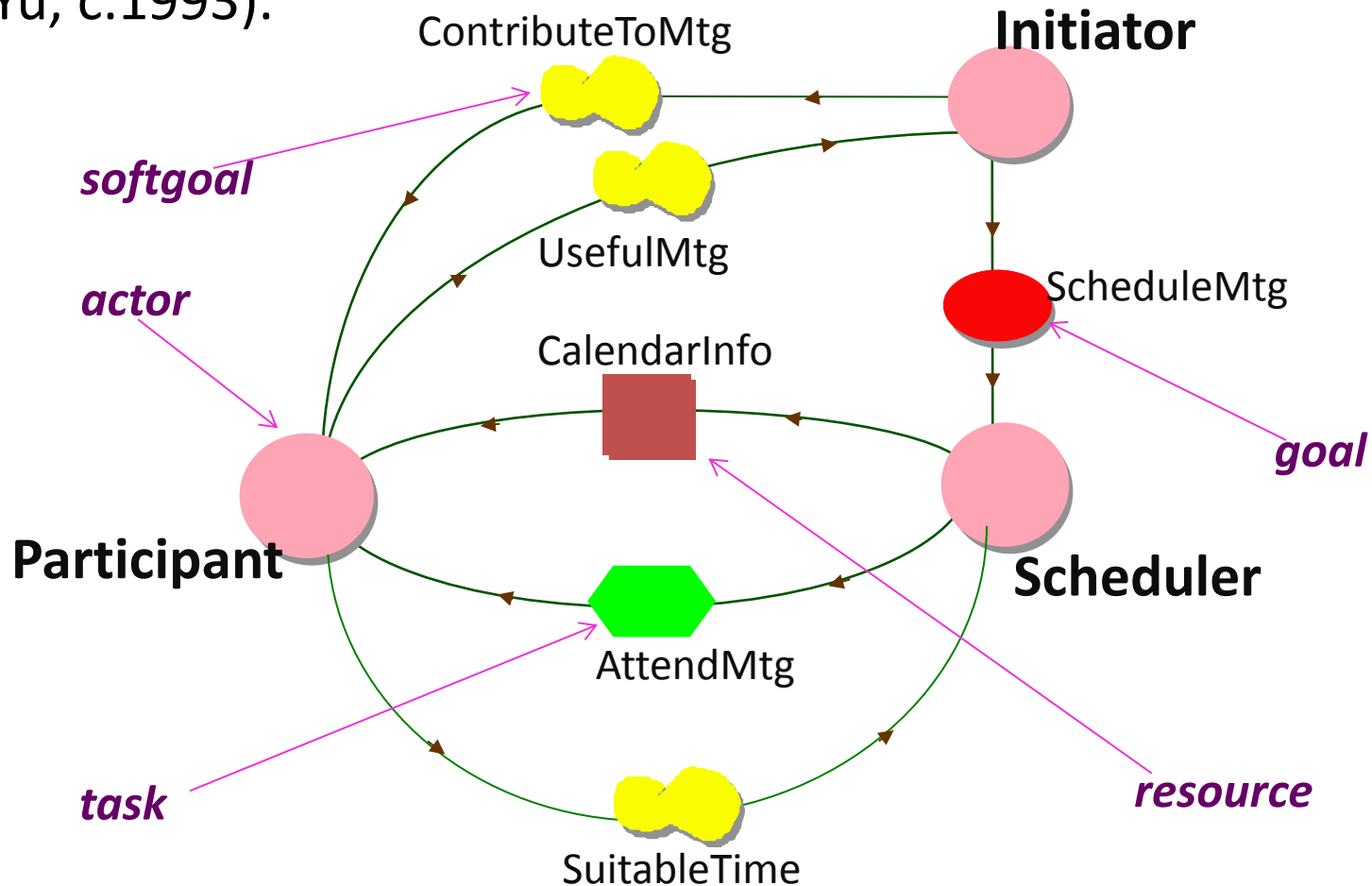
# Requirements as goals

Requirements are now goals and (requirements) problem solving amounts to incremental AND/OR goal refinement (Axel van Lamsweerde, c.1993).



# Stakeholders as Actors

Models now include stakeholders, represented as actors, who have goals and rely on others for their fulfillment (Eric Yu, c.1993).





# Interesting ideas ...

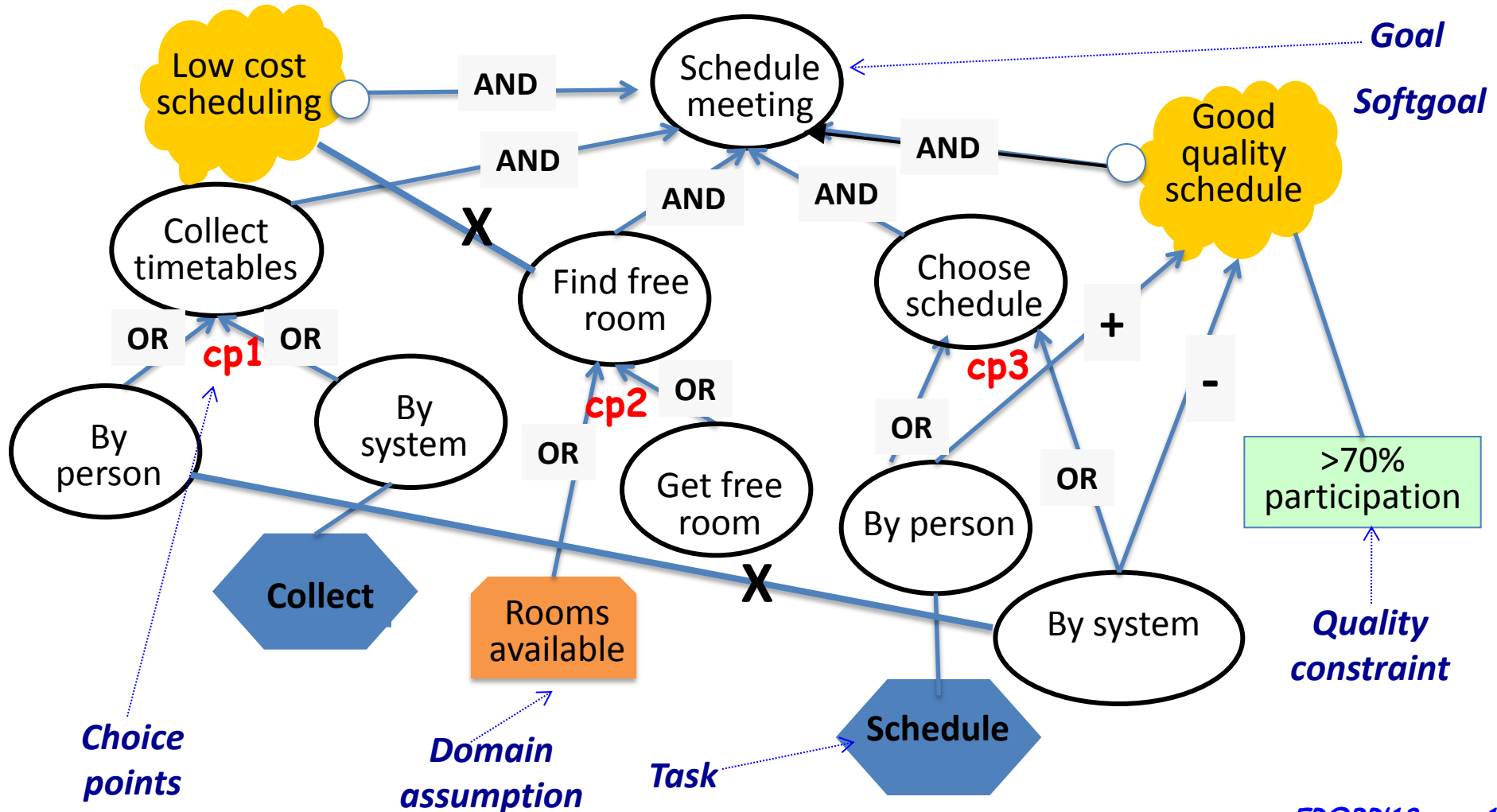
- 🌈 Requirements derived from models of the **domain** (Ross).
- 🌈 Requirements and **specifications** are different things, though logically related (Jackson).
- 🌈 Requirements as **goals** stakeholders want (vanLamsweerde).
- 🌈 The requirements problem is a **social** problem (Yu).
- 🌈 The requirements problem is solved through problem **refinement** (all), and this refinement has many forms: activity decomposition (Ross), abductive inference (Jackson), goal refinement (vanLamsweerde), social delegation (Yu).
- 🌈 With goal models and refinement, you are not exploring a design, but rather a design **space**.

# Lessons learned

- 🌈 Requirements modelling languages (RMLs) should be informality-tolerant, therefore not terribly expressive, e.g., propositional languages will do.
- 🌈 RMLs should be inconsistency-tolerant.
- 🌈 RMLs should be logics with a well defined semantics, e.g., an entailment relation ( $\models$ ).
- 🌈 RMLs are (... should be) triples, consisting of an ontology of primitive concepts, a language for making statements about requirements and the domain, and an entailment relation.
- 🌈 The ontology chosen for a RML impacts greatly elicitation, modelling and analysis.

# Goal models circa 2013

Goals can be mandatory/nice-to-have, can have priorities [Jureta08], probabilities [Letier04], utility [Liaskos13], ...



# Reasoning with design-time goal models

- What-if: Assuming that some goals have been fulfilled/denied, infer the status of the rest of the goal model.
- Satisfiability: Is there a specification for a given goal model?
- What-if reasoning can be handled with simple label propagation algorithms, satisfiability requires a SAT solver [Sebastiani04].
- Reasoning with preferences, probabilities and utilities requires more than a SAT solver, e.g., AI planners [Liaskos10], SMT solvers, ...

# What do these models tell us?

- They give us alternative specifications (sets of functions, qualities and assumptions) for fulfilling requirements.
- If someone wants a design that fulfills requirements in multiple ways (e.g., product families, flexible business processes, adaptive software systems) then our solution and implementation should encompass multiple specifications, not just one.
- These are *design-time* goal models, of no apparent use during runtime and/or evolution.



# Adaptive Software Systems

- 🌈 Software systems increasingly operate within volatile environments where the one constant is uncertainty: cyber-physical systems, socio-technical systems, ...
- 🌈 In response, there has been growing interest in adaptive software that monitors its own performance and the environment, and adapts if its requirements fail.

👉 *Need to monitor requirements, but how?*

- 🌈 Two approaches: (a) Monitor design artifacts (code, architecture, business process) and draw conclusions about requirements, (b) monitor requirements.

*... We opt for the latter, of course!*

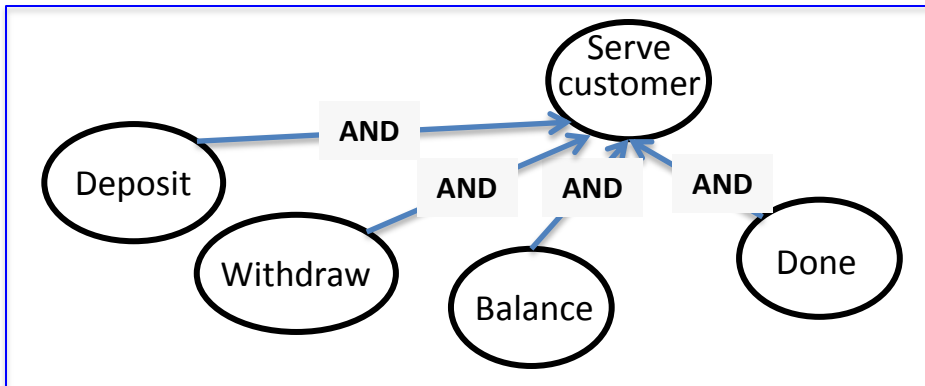
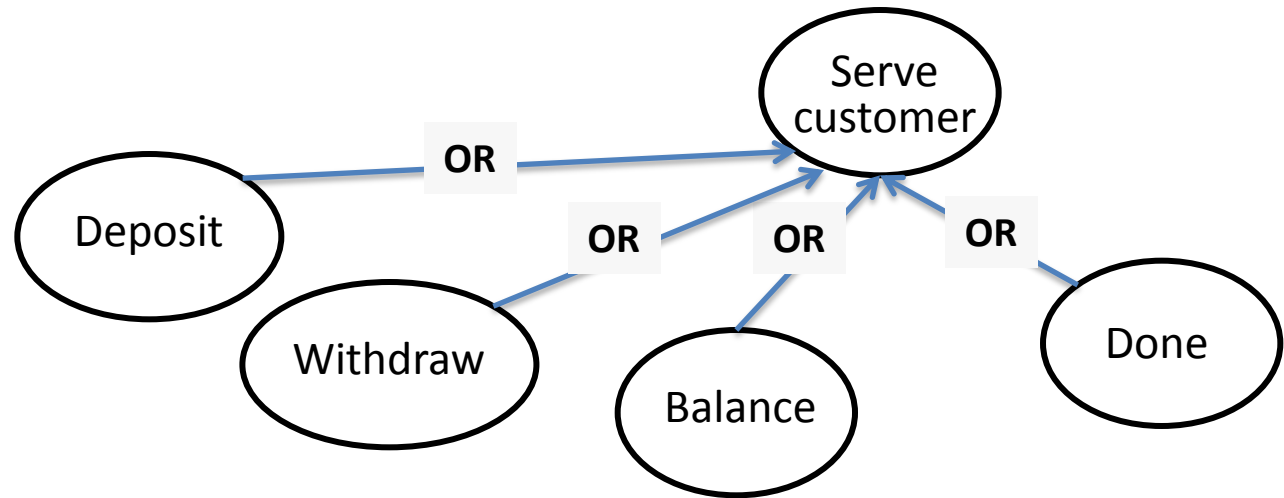
# Requirements for adaptation

- Awareness requirements [Souza11]: Impose constraints on the success/failure of other requirements  
e.g., “Requirement R should not fail for more than 5% of the time for any 1-month period”
- Evolution requirements [Souza12]: When certain conditions apply, specify changes to other requirements  
e.g., “If requirement R fails 3 times in a row, drop it (it is no longer a requirement).”



# Runtime models – motivation

- Yiqiao Wang [Wang07] used goal models to support monitoring and diagnosis for adaptive systems.
- Excerpt from her ATM example goal model

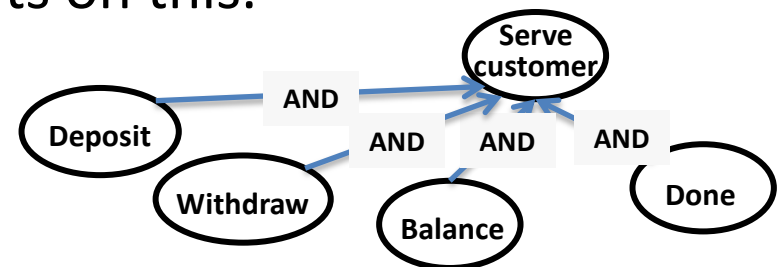


Wait a minute ...  
Why not ...??



# Design-time vs runtime models

- Design-time models are intended to help us capture *required functionality* for the system-to-be.
- Runtime models are intended to help us *monitor behaviour* of the system and take corrective action, if necessary.
- We know how to (formally) reason with design-time models.
- How do we reason with runtime ones? For example, if we know that an instance of D, satisfied, is followed by 2 instances of W, both satisfied, and two instances of B, one satisfied, the other pursued, what can we infer about their parent instance of SC?
- See [Morandini09] for early results on this.



# Towards Runtime Models

- We augment goal models, so that they additionally capture:
  - ✓ Behaviour – possible sequences of sub-goals for fulfilling a goal;
  - ✓ State – possible states of a goal instance; current state of each goal instance;
  - ✓ History – the state history of all instances of a goal



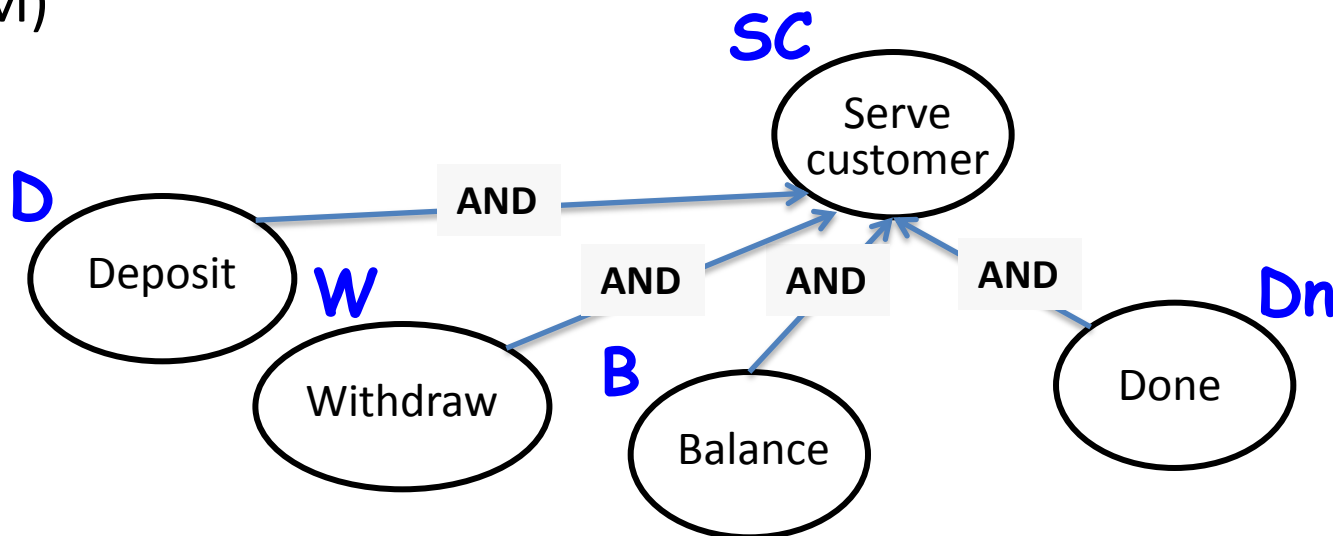
# Behaviour

- Defined by annotating every non-leaf goal with a shuffle regular expression, e.g.,

$$\text{annot}(SC) = (D \mid W \mid B)^+ ; D_n$$

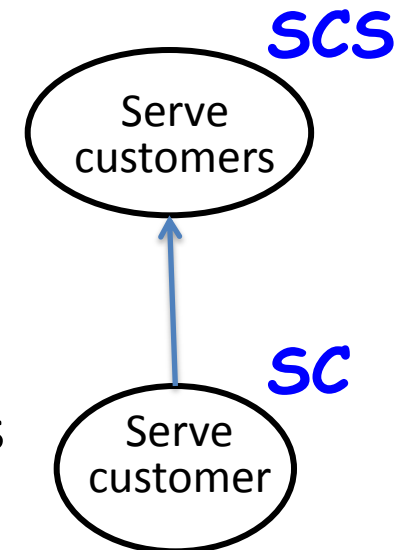
... or ...  $((D \mid W)^+ \# B^+); D_n$

- The result of such annotations is a behavioural goal model (BGM)



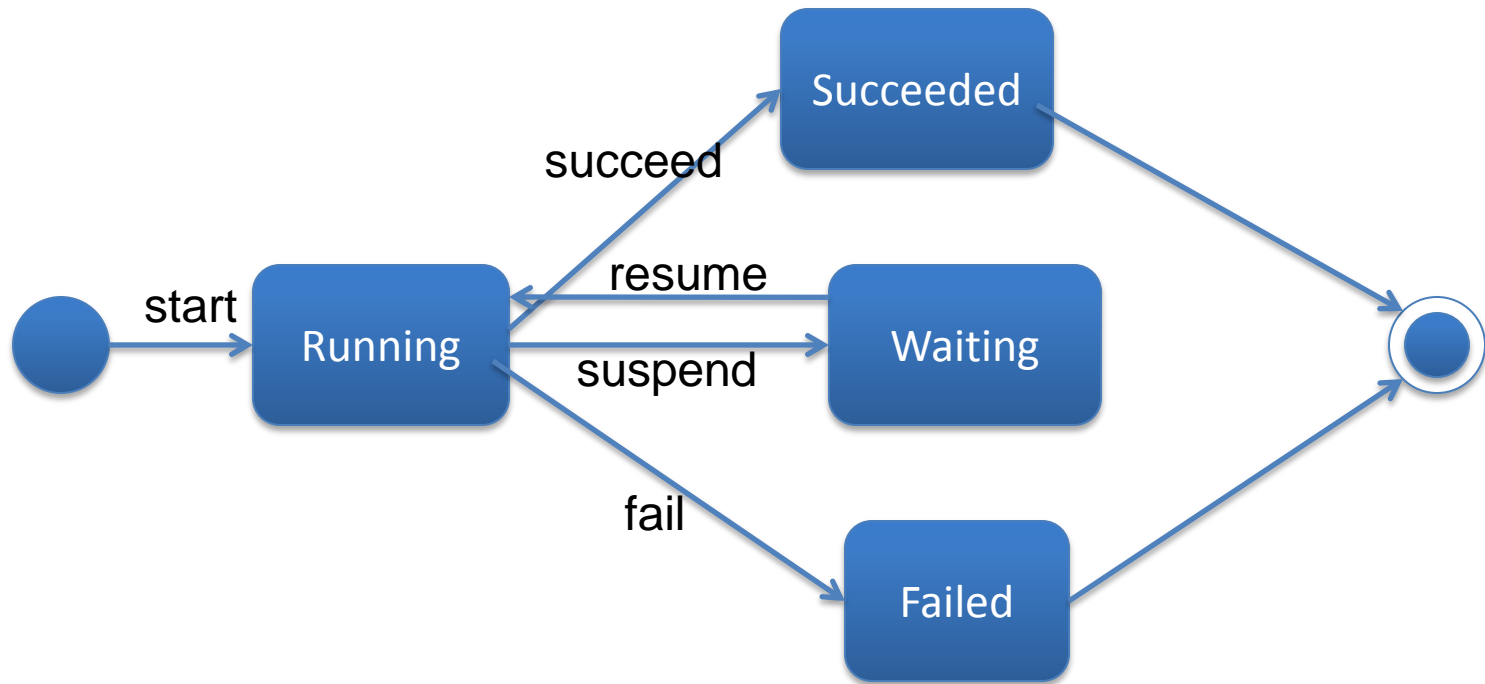
# Behaviour – Shuffle

- Means exactly what you see ...
- If  $w1 = abb$  and  $w2 = acbb$ , then  $w1 \# w2$  consists of strings like  $abcbbb$ ,  $acbbbb$ , ... lot's of them!
- More interestingly, shuffle closure ...  
 $w^\# = w \mid w\#w \mid w\#w\#w \mid \dots$   
amounts to unbounded concurrency
- For example,  $SCS = SC^\#$
- Recognition for shuffle regular expressions is PTIME



# State

- We can use FSMs, such as the following one (for goals).
- Every goal instance can be in one of these states. ...

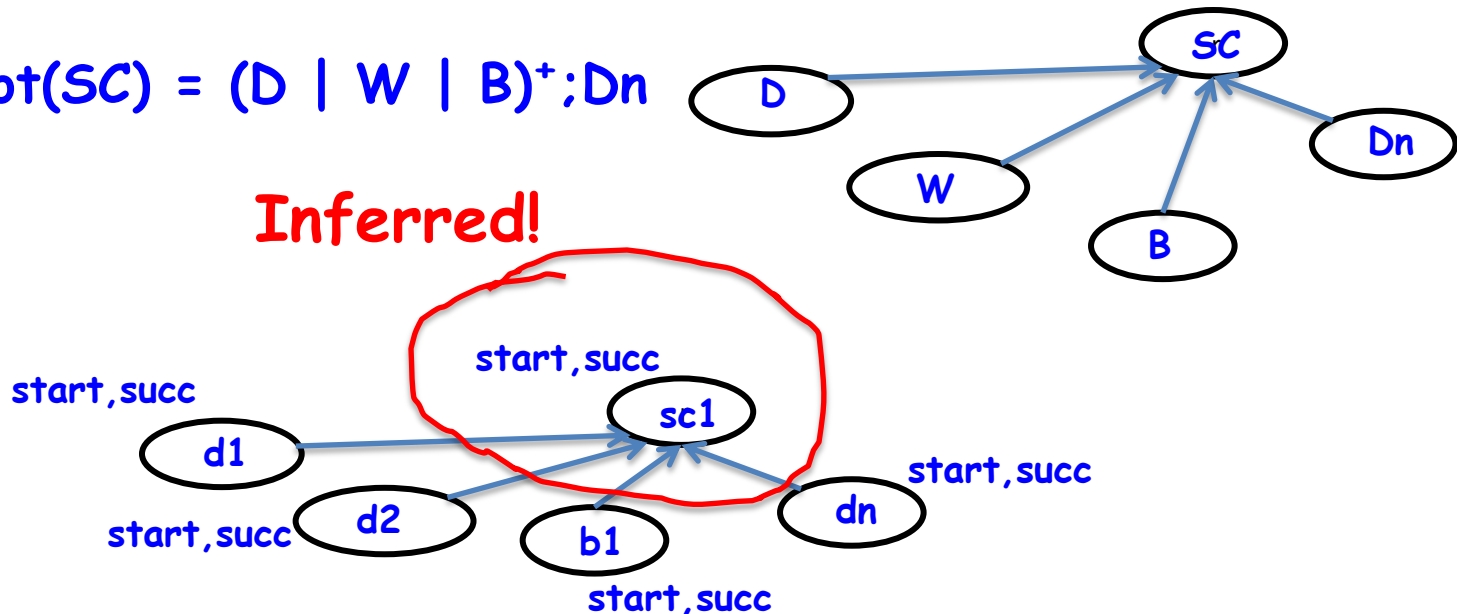


# History

At runtime, a system *trace* is generated marking state transitions for *leaf* goal instances

$d1.start, d1.succ, d2.start, \dots, dn.start, dn.succ$

$annot(SC) = (D \mid W \mid B)^+; Dn$



# Reasoning with runtime goal models

- 🌈 Recognition: Given a trace and a BGM, determine if the trace is legal.
- 🌈 RGI construction: Given a trace and a BGM, infer the states of non-observable goal instances and construct a corresponding goal instance model.
- 🌈 Diagnosis: Assume a class of possible failures; given a trace and a BGM, determine if there is a failure; if so, determine all possible root causes.



# Summary

- Unlike their design cousins, runtime requirements models need to capture behaviour, state and history.
- Reasoning with such models is founded on recognition problems for formal languages and automata, rather than satisfiability in logics.
- The ever-growing demand for flexibility, adaptability, customizability, evolvability, etc. dictates the use of requirements models both at design time, runtime and throughout the lifecycle of a software system.





## References

- 🌈 [Borgida13] Borgida A., Dalpiaz F., Horkoff J., Mylopoulos J., “Requirements Models for Design- and Runtime”, 2<sup>nd</sup> ICSE Workshop on Models in Software Engineering (MISE’13), San Francisco, May 2013.
- 🌈 [Dalpiaz13] Dalpiaz F., Borgida A., Horkoff J., Mylopoulos J., “Runtime Goal Models”, 7<sup>th</sup> IEEE International Conference on Research Challenges in Information Science (RCIS’13), Paris, May 2013.
- 🌈 [Dardenne93] Dardenne, A., van Lamsweerde, A. and Fickas, S., “Goal-Directed Requirements Acquisition”, in The Science of Computer Programming 20, 1993.
- 🌈 [Jackson95] Jackson M., Zave P., “Deriving Specifications from Requirements: An Example”, 17<sup>th</sup> International Conference on Software Engineering (ICSE’95).
- 🌈 [Jureta10] Jureta, I., Borgida, A., Ernst, N., Mylopoulos, J., “Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences and Inconsistency Handling”, 19<sup>th</sup> Int. IEEE Conference on Requirements Engineering (RE’10), Sydney Australia, Sept. 2010.
- 🌈 [Letier04] Letier E., van Lamsweerde A., “Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering”, 12th International Symposium on the Foundation of Software Engineering FSE-04, pages 53–62, Newport Beach CA, November 2004.

## References (cont'd)

- [Liaskos10] Liaskos, S., McIlraith, S., Sohrabi, S., Mylopoulos, J., “Integrating Preferences into Goal Models for Requirements Engineering”, 19<sup>th</sup> International IEEE Conference on Requirements Engineering (RE’10), Sydney Australia, September 2010.
- [Liaskos13] Liaskos S., Khan S., Soutchanski M., Lapouchnian A., and Mylopoulos J., “Modeling and Reasoning About Uncertainty in Goal Models”, (submitted for publication)
- [Morandini09] Morandini M., Penserini L. and Perini A., “Operational Semantics of Goal Models in Adaptive Agents,” 8th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS’09), 129–136.
- [Ross77] Ross, D. T., and Schoman, “Structured Analysis: A Language for Communicating Ideas,” *IEEE Transactions on Software Engineering* 3(1), Special Issue on Requirements Analysis, January 1977, 16-34.
- [Sebastiani04] Sebastiani R., Giorgini P., Mylopoulos J., “Simple and Minimum-Cost Satisfiability for Goal Models”, 16<sup>th</sup> International Conference on Advanced Information Systems Engineering (CAiSE’04), Riga, June 2004, Springer-Verlag LNCS 2003, 20-35.
- [Souza11] Silva Souza V., Lapouchnian A., Robinson W., Mylopoulos J., “Awareness Requirements for Adaptive Systems”, 6<sup>th</sup> ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS’11), Waikiki Honolulu, May 2011.

## References (cont'd)

- 🌈 [Souza11a] Silva Souza V., Lapouchnian A., Mylopoulos J., “System Identification for Adaptive Software Systems: A Requirements Engineering Perspective”, 30<sup>th</sup> International Conference on Conceptual Modelling (ER’11), Brussels, November 2011, 346-361.
- 🌈 [Souza12] Silva Souza, V., Lapouchnian A., Angelopoulos K., Mylopoulos J., “Requirements-Driven Software Evolution”, *Computer Science – Research and Development (CSR)*, Springer-Verlag, October 2012 (online version).
- 🌈 [Souza12a] Souza V., Lapouchnian A., Mylopoulos J., “Requirements-Driven Qualitative Adaptation”, 20<sup>th</sup> International Conference on Cooperative Information Systems (CoopIS’12), Rome, September 2012, Springer-Verlag LNCS 7566, 342-361.
- 🌈 [Wang07] Wang, Y., McIlraith, S., Yu, Y., Mylopoulos, J., “An Automated Approach for Monitoring and Diagnosing Requirements”, 22<sup>nd</sup> IEEE/ACM International Conference on Automated Software Engineering (ASE’07), Atlanta, November 2007, 293-302.
- 🌈 [Yu93] Yu Eric, “Modelling Organizations for Information Systems Requirements Engineering”, First IEEE International Symposium on Requirements Engineering (ISRE’93), San Jose, January 1993.
- 🌈 [Chidi Okoye] <http://www.modernartimages.com/expressionsofdance3.htm>