External Variability of Software: Classification and Ontological Foundations

ER'2011

Iris Reinhartz-Berger
Department of Infromation Systems
University of Haifa, Israel

Arnon Sturm
Department of Information Systems Engineering
Ben-Gurion University of the Negev, Israel

Yair Wand
Sauder School of Business
University of British Columbia, Canada

Agenda

- Variability
  - What is it?
  - Why classify it?
  - Existing classifications & limitations
- The suggested classification framework
  - Premises & foundations
  - Applications vs. Systems
  - Comparison of Applications and External Variability Classes
  - Possible uses of the framework
- Summary & Future Work

What is Variability?

- *Variability* is the ability of a reusable artifact to be efficiently extended, changed, customized, or configured for use in a particular context.
- Variability is of particular importance in different engineering fields
  - In *software product line engineering*, for example, variability entails assumptions about how *members* of a *family* may *differ* from each other.

Why Classify Variability?

- Variability classification can serve for:
  - Better understanding of families of software products
  - Better analysis of the implications of using reusable artifacts in various contexts
  - Developing generic components that can be easily adapted to different software products
  - Better understanding of the expressiveness of specification languages

Existing Classifications of Variability

- Pohl, Böckle, and van der Linden (2005):
  - ***External (or essential) variability*** – variability that is visible to customers
    - Functionality, System Environment, Quality, Integration in Business Processes, Information/Data, …
  - ***Internal (or technical) variability*** – variability that is hidden from customers
    - IT-Infrastructure, Binding Time, Implementation, …

Existing Classifications of Variability

- Both types are important to the success of developing software artifacts. But:
- External variability deserves special attention:
  - It is visible to stakeholders
  - It relates to requirements defined at early development stages
    - where errors or inaccuracies are relatively inexpensive and easy to detect and correct.

Limitations in Existing Classifications

- Variability in general and external variability in particular are mostly analyzed either from ***practical points of view*** or in terms of ***taxonomies and metamodels*** .
- The usability of these works for ***analyzing*** the similarity and differences between complete applications or software products is limited, as they are primarily designed for ***specification purposes***.

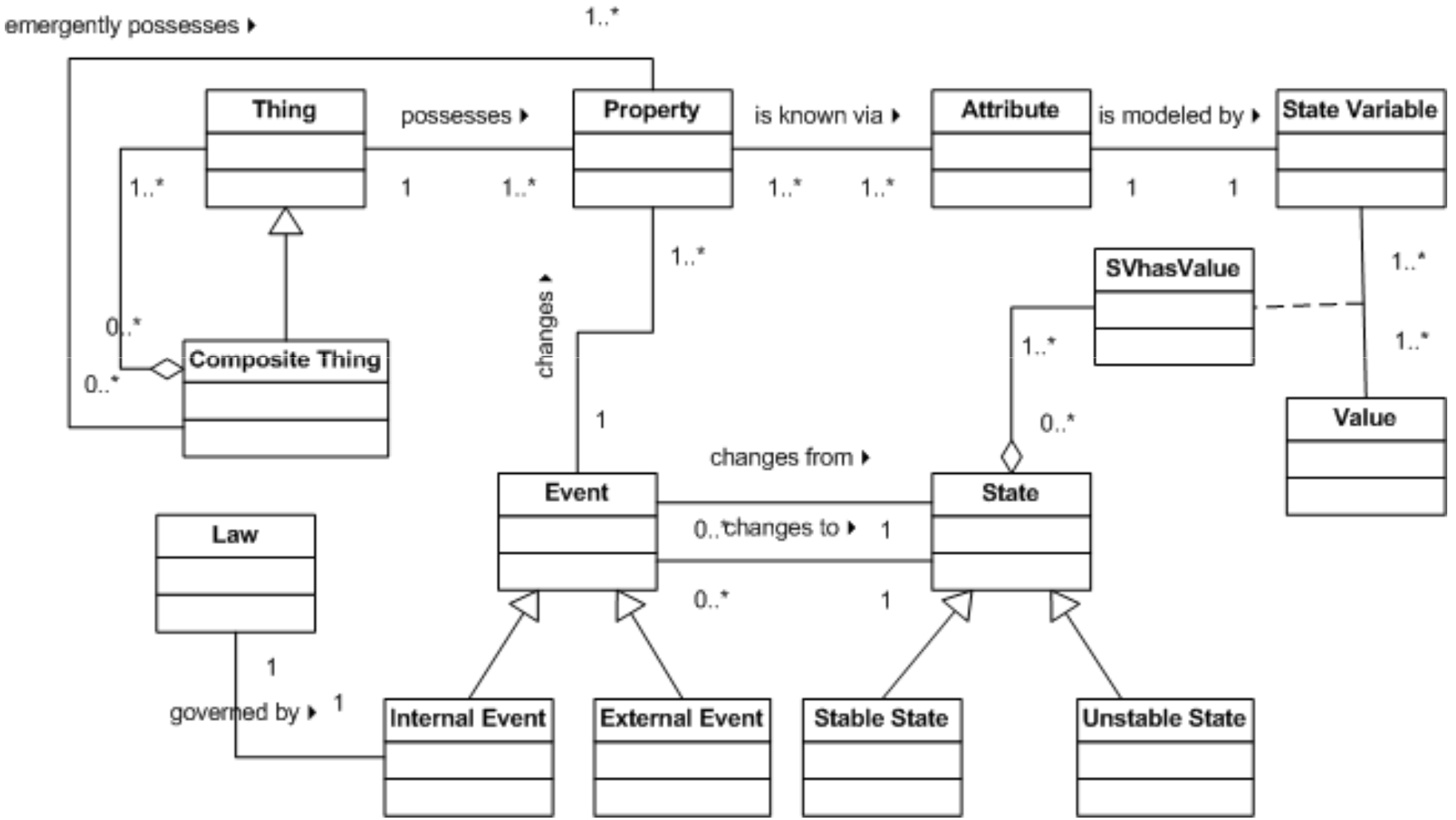
The Suggested Classification Framework

- The main premise that underlies our approach is:
  - From a user's point of view, what mainly matters is ***the behavior of the implemented system***.
- Thus, an application is defined as a set of ***intended changes in a given domain*** (= ***application behaviors***).
  - We will focus on application behaviors as specified in functional requirements.
  - We will not refer to realization and implementation
    - these aspects are usually less relevant to the customers (users) of the resulting system.

The Suggested Classification Framework

- To formalize behaviors we use an ontological model
  - We use concepts from Bunge's ontological work.
    - It places an emphasis on the notion of systems
    - It defines the concepts of states and events in an axiomatic, well-formalized way
    - It has been adapted to conceptual modeling in the context of systems analysis and design

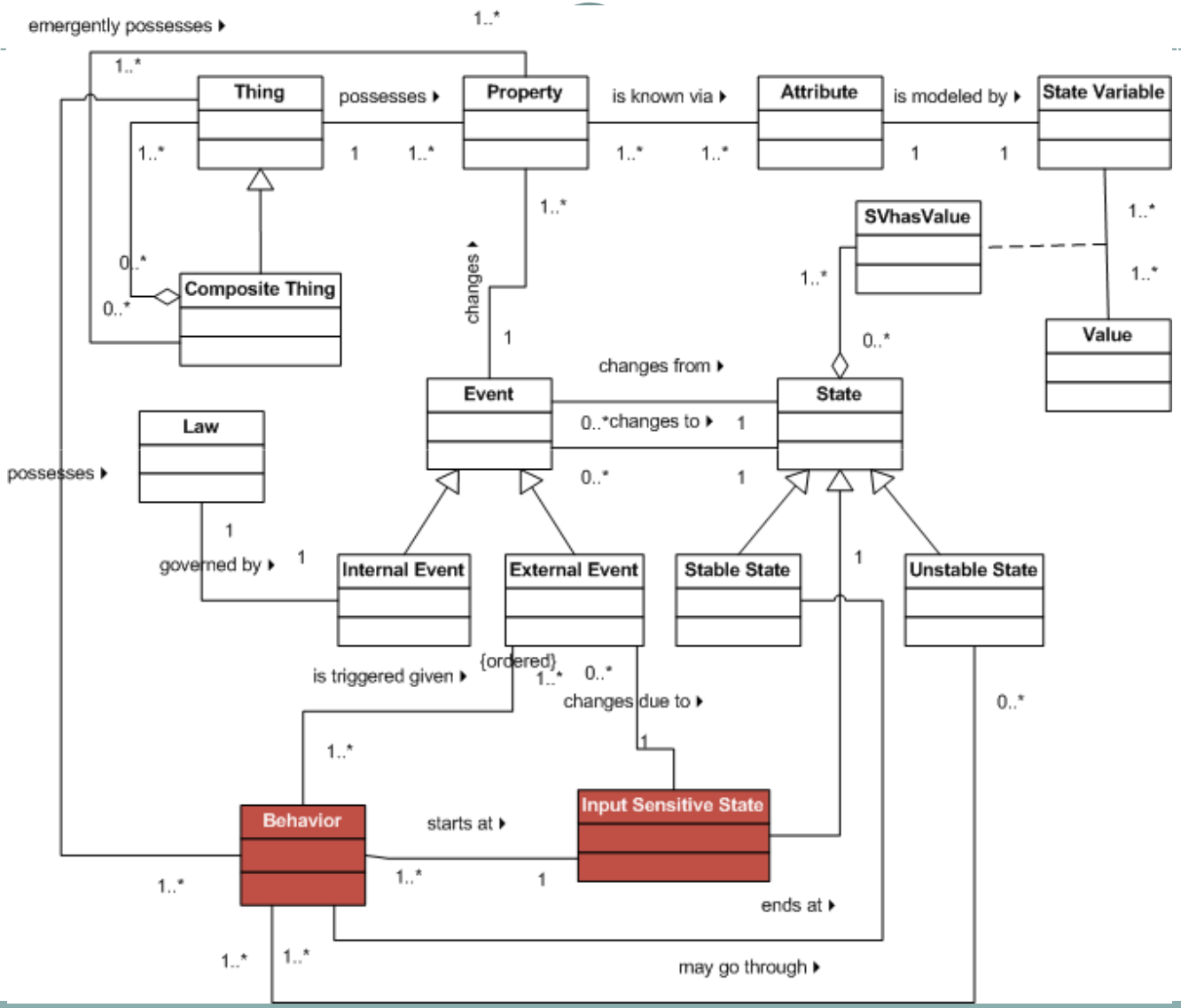
Bunge's Terminology



Additional Terms

- An ***input sensitive state*** is a state that can change due to an occurrence in the environment of the thing, i.e., due to an external event.
- Given an input sensitive state s_1 and a sequence of external events $<e_i>$, *a* ***behavior*** is a triplet (s_1, $<e_i>$, s^*)
  - s^* is the first stable state the thing reaches from state s_1 when the sequence of external events $<e_i>$ occurs.
  - s_1 is termed the ***initial state*** of the behavior and s^* – the ***final state*** of the behavior.

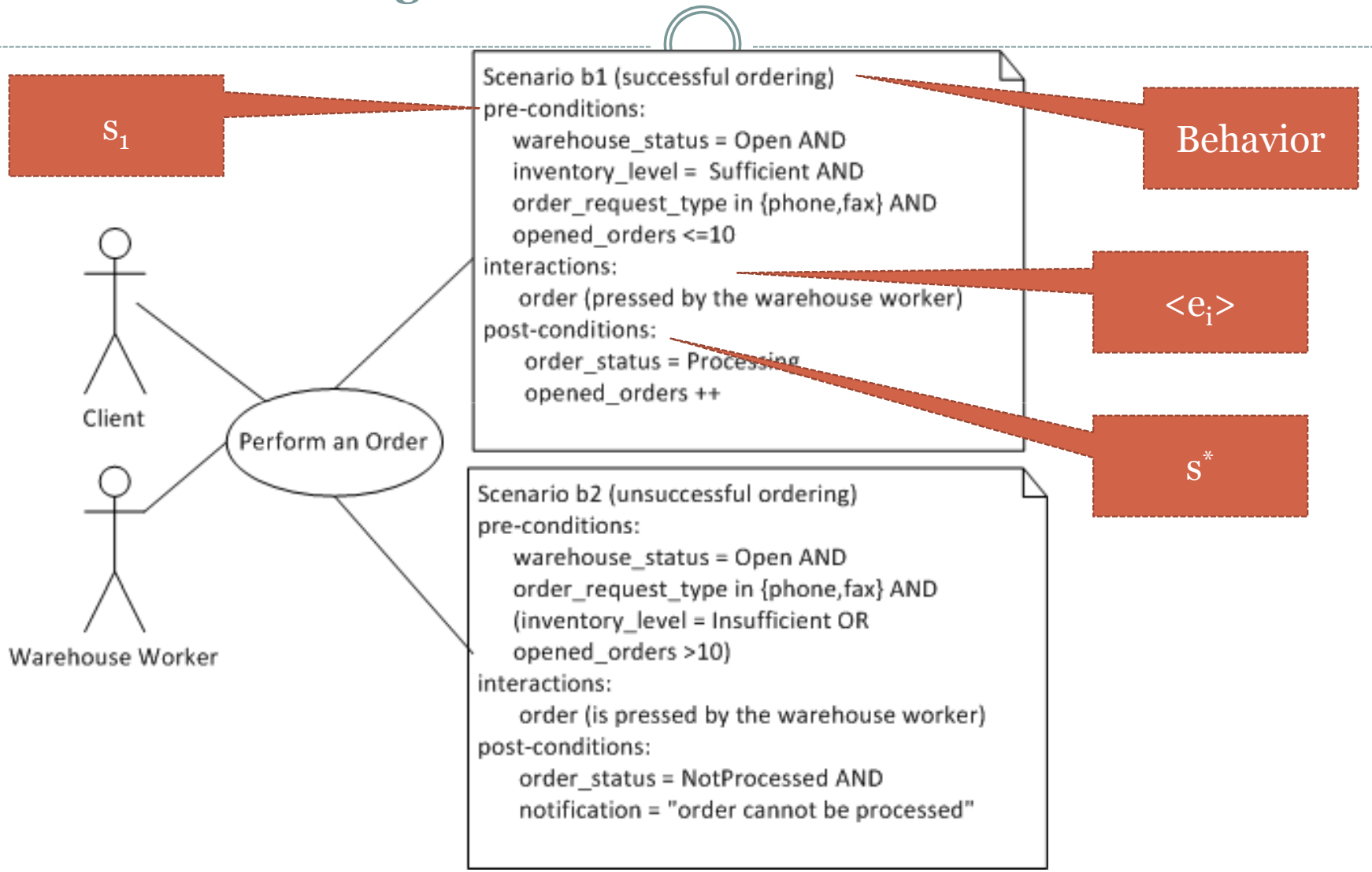
Additional Terms



Applications vs. Systems

- An application domain ("***domain***"):
  - a set of things where the states or the behaviors of these things are of interest for some stakeholders
    - Example: the warehouse domain
- An ***application over a domain***:
  - a sub-set of behaviors of interest in the domain
    - Examples: order processing, billing, and order delivery in warehouses
- A ***system***:
  - a way to realize and implement behaviors
  - the actual composition of components to create such behaviors.
    - Examples: Different systems for order processing, different systems for billing, and different systems for order delivery in warehouses

Representation of Applications
with Use Case Diagrams



Comparison of Applications

- A_1 and A_2, which are defined over the same domain D, are ***state-comparable*** if and only if their sets of state variables are not disjoint.
- States s and t of two applications *are* ***equivalent with respect to a set of state variables*** common to the two applications (X) if and only if they have the same values for all these state variables.
- We define similarity of external events in terms of ***equality***:
  - external events are defined as occurrences in the domain
  - the two compared applications are defined over the same domain

External Variability Classes
in application behaviors

#	s_1	<e_i>	s^*	Class Name	Examples	Explanation of the Examples
1.	$\overset{X}{\sim}$	=	$\overset{X}{\sim}$	Completely similar behaviors	$ready \xrightarrow{client\ orders} \mathbf{succeeded}$ $ready \xrightarrow{client\ orders} \mathbf{handled}$	View: orders are handled*. Result: The behaviors are completely similar.
2.	$\overset{X}{\sim}$	≠	$\overset{X}{\sim}$	Similar cases and responses, different interactions	$ready \xrightarrow{client\ orders\ by\ phone} succeeded$ $ready \xrightarrow{client\ orders\ by\ fax} \xrightarrow{technician\ fixes} succeeded$	View: orders are successfully processed. Result: The order is processed even though the events are different.
3.	$\overset{X}{\sim}$	=	$\overset{X}{\nsim}$	Similar triggers, different responses	$ready \xrightarrow{client\ orders} \mathbf{succeeded}$ $ready \xrightarrow{client\ orders} \mathbf{failed}$	View: the warehouse is ready to receive orders. Result: Although having the same start and interactions, the behaviors end differently.
4.	$\overset{X}{\sim}$	≠	$\overset{X}{\nsim}$	Similar cases, different behaviors	$ready \xrightarrow{client\ orders\ by\ phone} \mathbf{succeeded}$ $ready \xrightarrow{client\ orders\ by\ fax} \xrightarrow{technician\ cancels} \mathbf{failed}$	View: the warehouse is ready to receive orders. Result: Behaviors start similarly, but different sets of events yield different responses.

External Variability Classes
in application behaviors

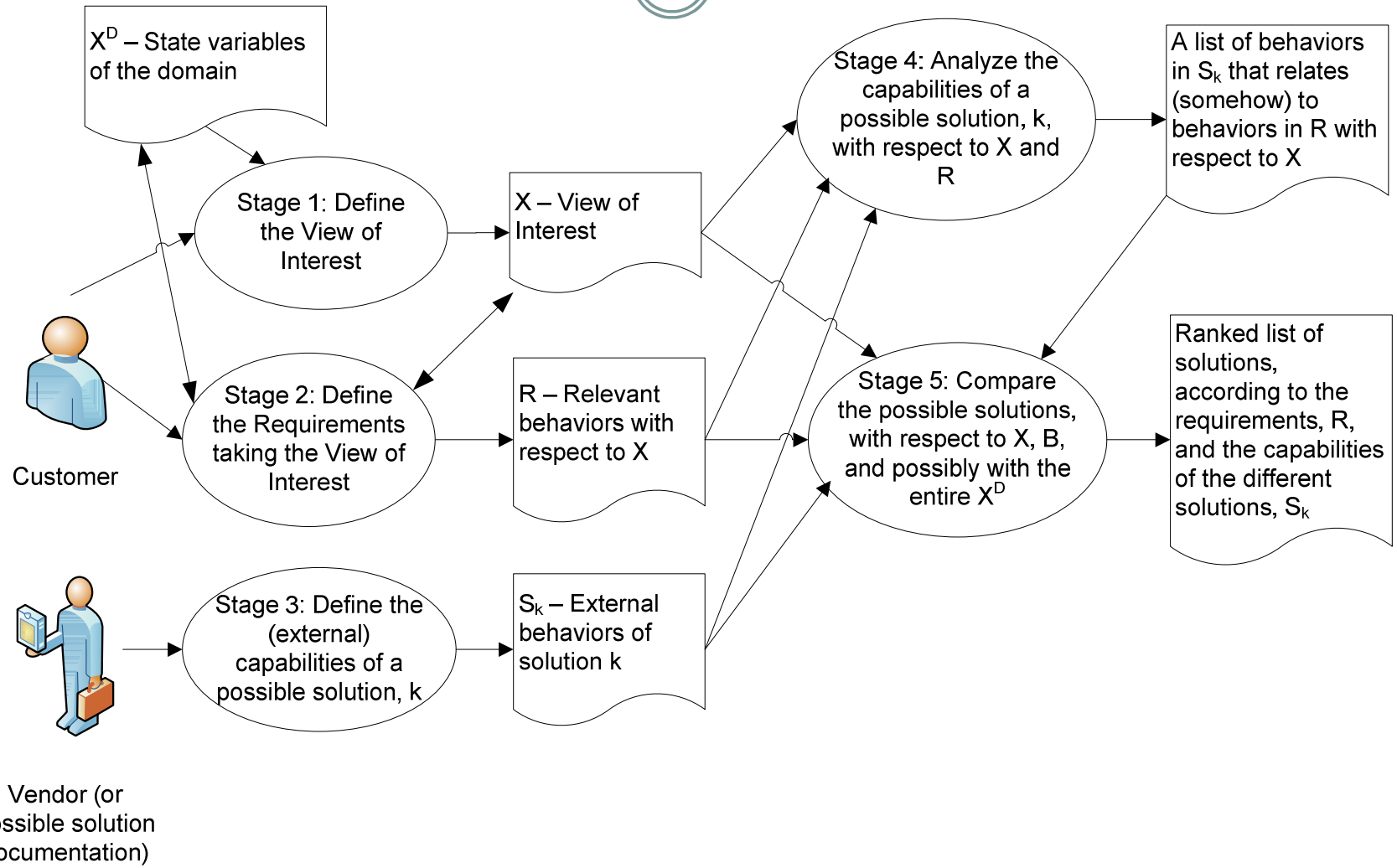
#	s_1	$\langle e_i \rangle$	s^*	Class Name	Examples	Explanation of the Examples
5.	$\overset{X}{\sim}$	=	$\overset{X}{\sim}$	Different cases, similar behaviors	$opened \xrightarrow{client\ orders} succeeded$ $closed \xrightarrow{client\ orders} handled$	View: orders are handled. Result: Behaviors end similarly with the same set of events occurring at different initial states
6.	$\overset{X}{\sim}$	≠	$\overset{X}{\sim}$	Different triggers, similar responses	$opened \xrightarrow{client\ orders} succeeded$ $closed \xrightarrow{client\ orders} \xrightarrow{client\ cancels} handled$	View: orders are handled. Result: Behaviors end similarly with different sets of events occurring at different initial states.
7.	$\overset{X}{\sim}$	=	$\overset{X}{\sim}$	Different cases and responses, similar interactions	$opened \xrightarrow{client\ orders} succeeded$ $closed \xrightarrow{client\ orders} failed$	View: the warehouse receives client orders. Result: Behaviors react to the same set of events that arrives at different initial states and yields different responses.
8.	$\overset{X}{\sim}$	≠	$\overset{X}{\sim}$	Completely different behaviors	$opened \xrightarrow{client\ orders} succeeded$ $closed \xrightarrow{client\ orders} \xrightarrow{client\ cancels} failed$	View: the warehouse handles ordering processes. Result: The behaviors react to different sets of events that arrive at different initial states and yield different responses.

The Degree of Similarity
between Applications

- The similarity of applications ***with respect to a behavior b in A1***, takes into consideration:
  - *The view of interest X*
  - *The variability class c*
  - The behaviors in A_2
- The ***similarity between applications*** *takes into consideration:*
  - *The view of interest X*
  - *The weights of the different variability classes*
  - The *weights of the different* behaviors in the two applications

Possible Usage

Conducting Systematic Feasibility Studies



Summary

- The prevalence of variability in various engineering fields indicates that it is important to further understand and formalize it for software-based systems.
- Presently, in software engineering, formalization of variability usually reflects technical or ad-hoc considerations.
- To formally define and analyze external variability which is tied to functional requirements, we suggest a framework based on an ontological model of application domain in terms of states and events.
  - The approach enables us to operationalize the notion of a stakeholder view and to reflect the degree of similarity among applications.
  - **Fewer differences exist => less effort is needed to adjust a system to the requirements.**

Future Work

- Test usability by creating practical case studies
- Test usefulness by conducting empirical work, such as:
  - Experimenting with the guidance provided by the framework to identify similarities and differences between applications;
  - Comparing the measures of similarity calculated analytically to answers provided by domain experts as to how willing they would be to substitute one application for the other;
  - Comparing the average similarity value to application adaptation effort as assessed by experts.
- Add uses of the approach
  - Such as the evaluation of variability modeling methods
  - Practical metrics to recommend adaption of alternatives (perhaps supported by an interactive software tool)

Future Work

- Extend the framework to include additional behavioral considerations and (some) aspects of internal variability, reflecting structural and non-behavioral characteristics
  - times and order of external events
  - variability related to quality
  - security aspects
  - …

Thank you for your attention. Any Questions?

ER'2011

Iris Reinhartz-Berger iris@is.haifa.ac.il
Arnon Sturm sturm@bgu.ac.il
Yair Wand yair.wand@ubc.ca