



The disappearing boundary between development-time and run-time

Carlo Ghezzi
Politecnico di Milano
Deep-SE Group @ DEI



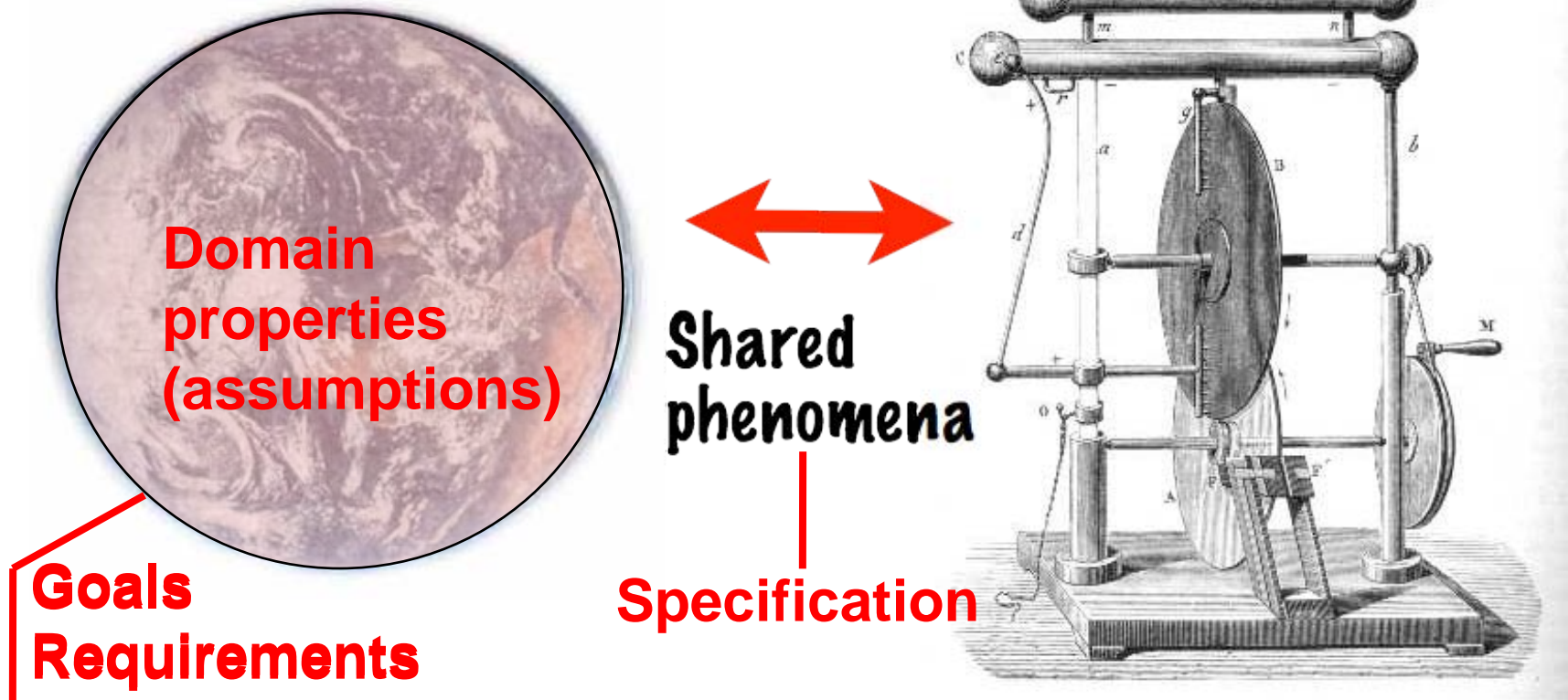
Our journey

- **Motivations**
 - Open-world challenges
 - Adaptation and evolution
- **Thesis**
 - Modelling and verification should they extend to runtime
- **Focus**
 - Non-functional quantitative requirements
- **Zoom** into a current research effort
- **Challenges and future work**

The *machine* and the *world*

World (the environment)

Machine





Dependability arguments

- If we have a formal representation for
 - R = requirements
 - S = specification
 - D = domain assumptionsit is necessary to prove that
 - $S \text{ and } D \text{ entail } R$
- Domain assumptions bridge the gap between requirements and specifications





The role of (formal) models

- The formal representations of D and S are often given in terms of models (e.g., state machines)
- Dependability arguments are based on proofs that the models satisfy R
- For example, model checking may be used at design time to assess dependability



A world of change

- Changes in **goals/requirements**
 - Business level
 - User: skills, profiles (preferences, role, ...)
- Changes in **domain**
 - Physical context
 - space, time, ...
 - Computational context
 - external components



Multiple ownership

- Systems increasingly built out of parts (**services**) that are developed, maintained, and even operated by independent parties
- No single stakeholder oversees and controls all parts
- Parts may change over time in an unannounced manner
- Yet by assembling the whole we commit to achieving a certain goal
 - We may even subscribe a *contract (SLA)*



Real-world case (1)

- **Networked enterprises**
 - Business integration infrastructures via Web services are becoming common
- A marketplace for Web services is being created
 - New services created (and possibly exposed) by composing other services
- Networks must be reconfigured to respond to rapidly changing requirements (and changes in business world)



Real-world case (2)

- **Pervasive/ubiquitous systems**
 - Situational changes mainly due to mobility
 - new devices/components encountered/discovered dynamically
 - Interactions/collaborations established dynamically
- Further adaptations due to resource constraints
 - E.g., power consumption
 - Physical conditions (heat, humidity, light, ...)



Change revisited

- Change recognized as a crucial problem since the 1970's (see work by M. Lehman)
- What is new here
 - The unprecedented degree of change
 - The request that software responds to changes in a **self-managed** manner (continuously running systems)



Changes may affect dependability

- Changes may concern R and/or D
 - hereafter we focus on self-managed reactions to changes in D **adaptation**
 - we ignore reactions to changes in R **evolution**
- We can decompose D as $D = D_f \wedge D_c$
 - D_f is the fixed/stable part
 - D_c is the changeable part
- We need to **identify D_c** and automatically **change S**
 - change detection**
 - self reaction**



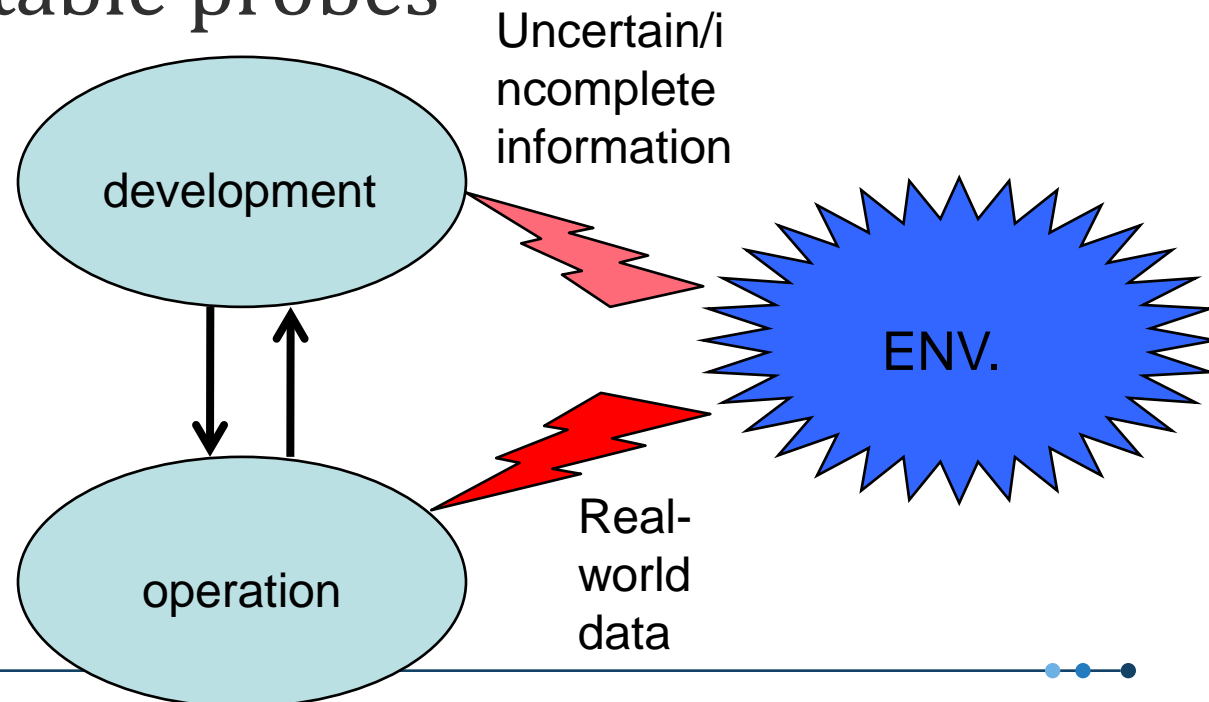
Change detection

- We need to get real data from the world through (abstract) sensors; e.g., by activating suitable probes

– *monitor*

- Transform data into information

– *learn*

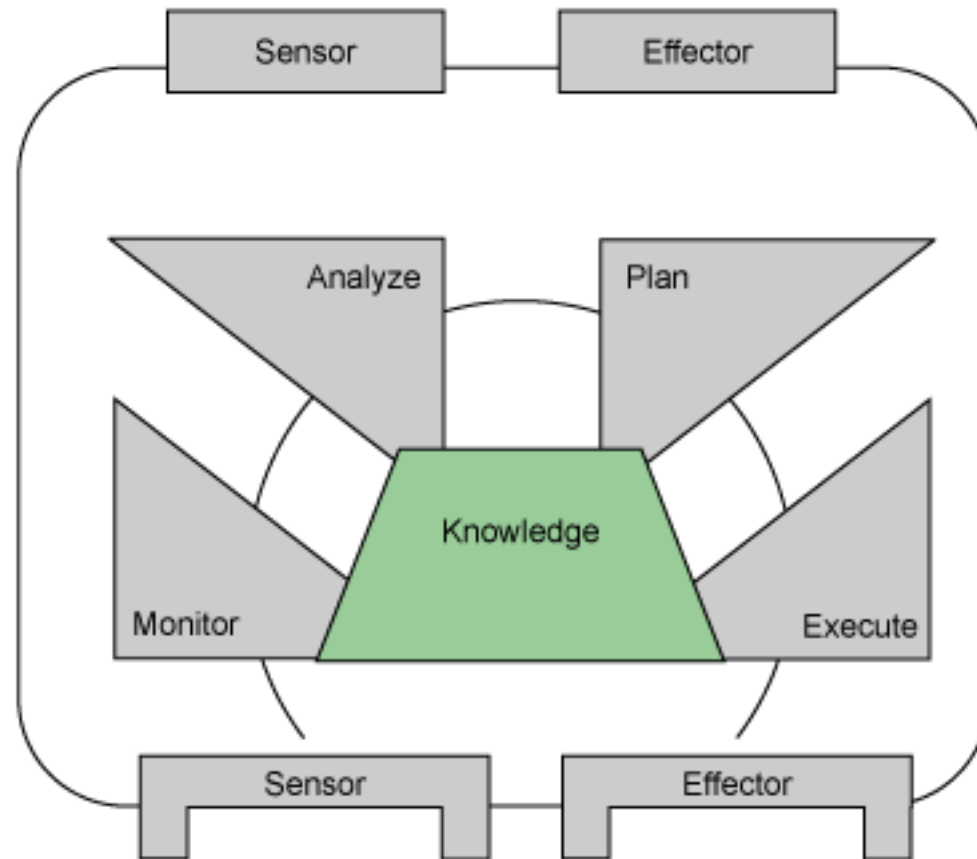




Self reaction

- Models of D and S kept alive at run time
- Model of D updated (see change detection)
- Continuous verification to check that R is satisfied a machine that operates according to S in an environment that behaves as D
- Self-reaction strategies triggered by failure of dependability argument
- **Development-time/run-time boundary vanishes**

Autonomic computing context (IBM)

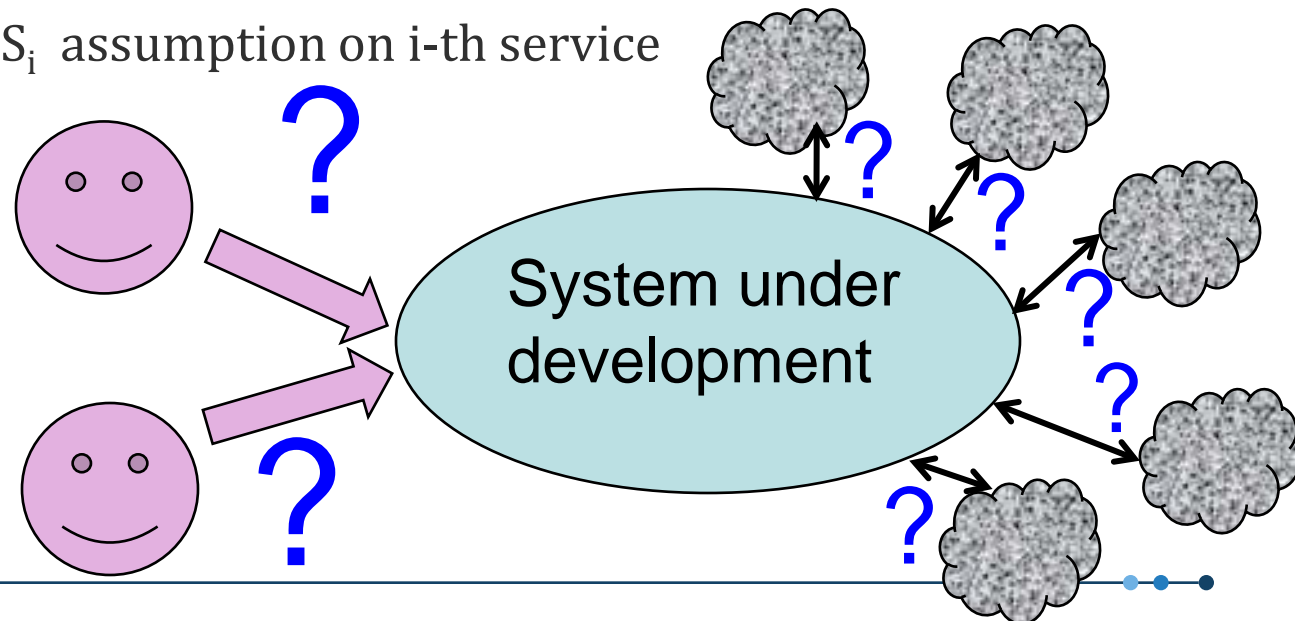




Our focus

- We focus on non-functional properties
 - reliability, performance
- Quantitatively stated in probabilistic terms
- $D_c = D_u \wedge D_s$
 - D_u = usage profile
 - $D_s = S_1 \wedge \dots \wedge S_n$ S_i assumption on i-th service

?
Hard to estimate at
design time +
very likely to change
at run time



Our approach

The KAMI system



We build on three key pillars

Modeling
Verification

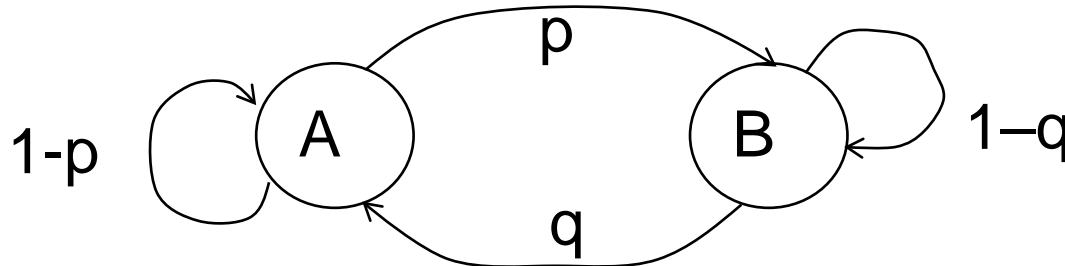


- Markovian models
 - DTMCs
 - CTMCs
- Model checking
 - PRISM (MRMC)
- On going work on using Queuing Networks
- *Open environment should allow adding tools to the workbench*



A detour: DTMCs

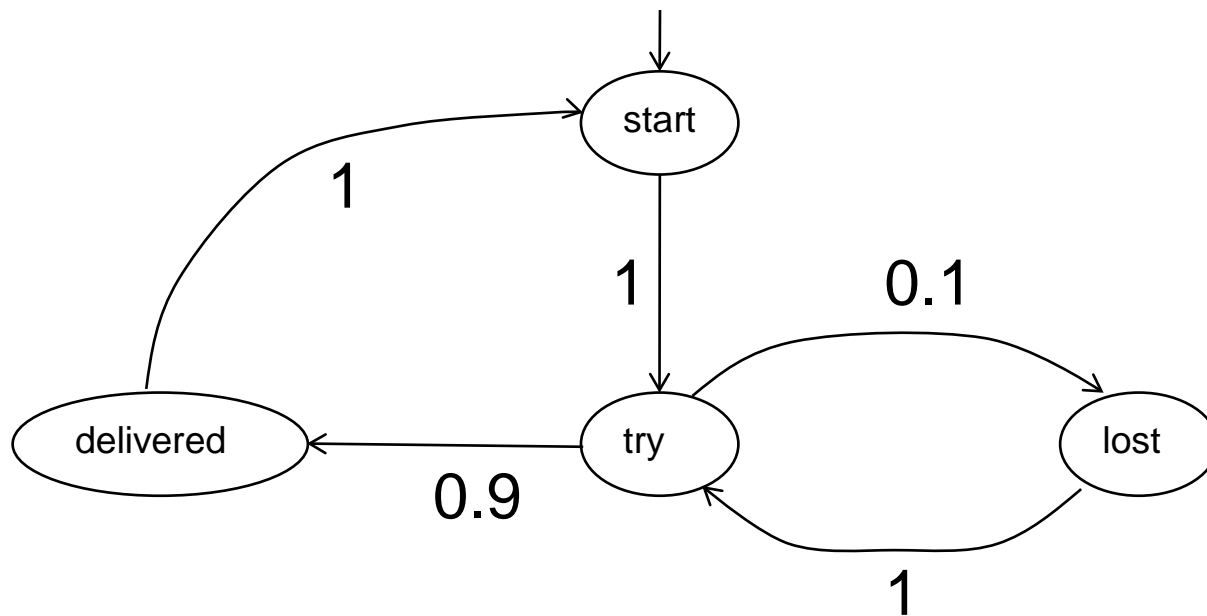
- A finite-state machine where transitions are labelled with probabilities
 - the sum of probabilities associated with transitions exiting each state is 1
- At every time slot a transition is chosen randomly based on **current** state (a coin is flipped at every time slot)





An example

A simple communication protocol operating with a channel



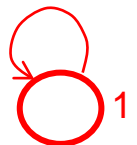
	S	D	T	L
S	0	0	1	0
D	1	0	0	0
T	0	0.9	0	0.1
L	0	0	1	0

matrix representation



PCTL

- Probabilistic extension of CTL
- In a **state**, instead of existential and universal quantifiers over paths we can state $P_{\approx p} [\varphi]$, where p is a probability value and \approx is $<$, $>$, \leq , \geq
 - e.g.: $P_{<0.2} [\varphi]$ means that the probability for the set of paths (leaving the state) to satisfy φ is less than 0.2
- In addition, **path** formulas also include step-bounded until
 - $\phi_1 U^{\leq k} \phi_2$
- An example of a reliability statement
 - $P_{>0.8} [\diamond(\text{system state} = \text{success})]$ ← **absorbing state**



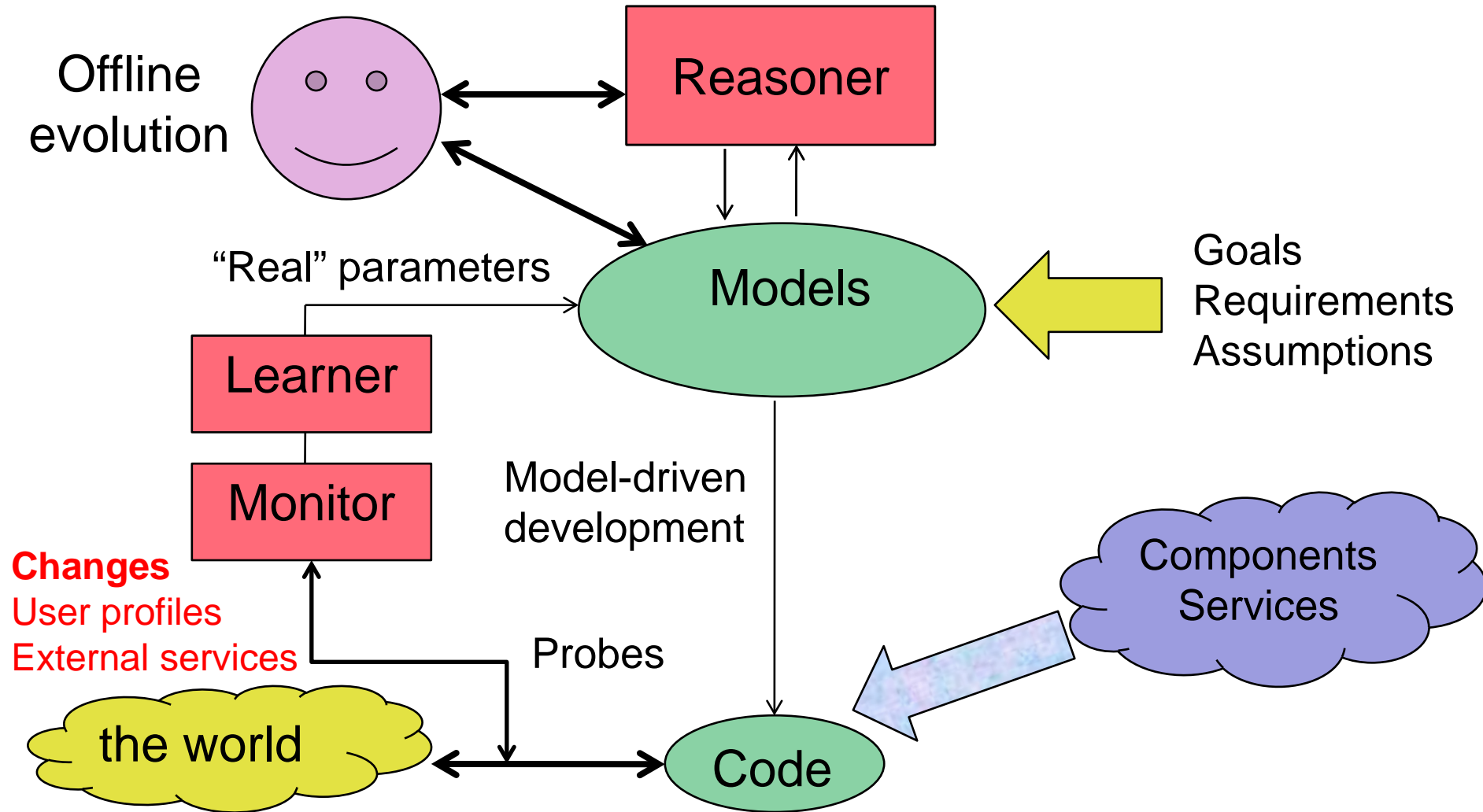


Further side remarks on models

- Why focus on formal models in an ephemeral world? Isn't this a contradiction?
 - see anti-model attitude of “agile” methods
- **Dependability**
 - Models are needed to support systematic reasoning in presence of uncertainty
- **Rapid development**
 - Implementations may be derived by transformation

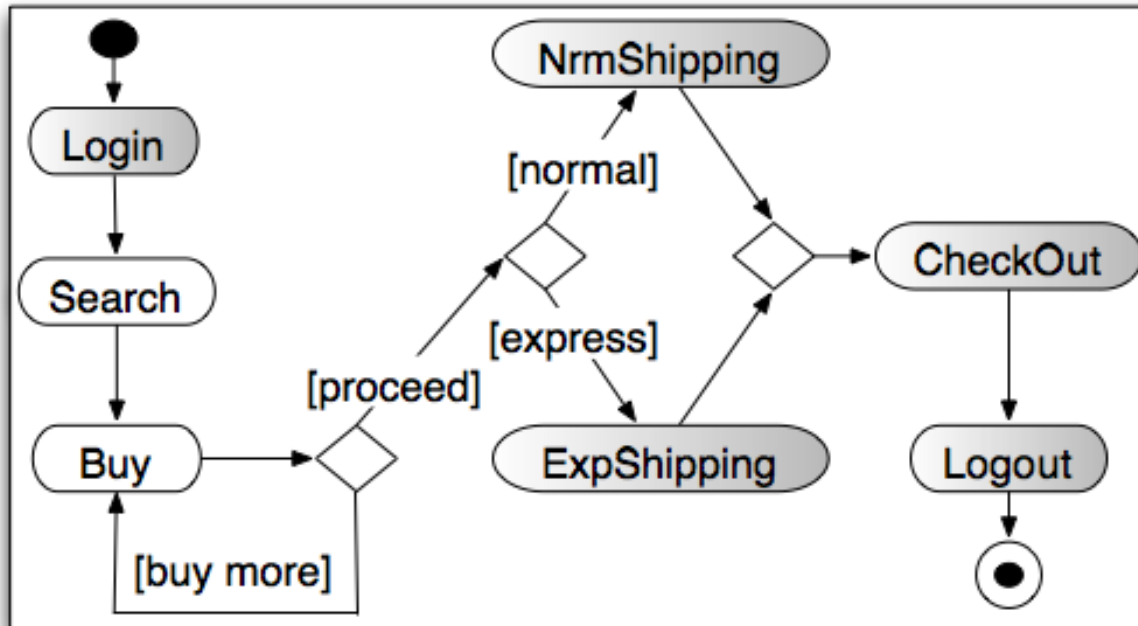


KAMI's conceptual model





KAMI in action: e-commerce service composition



FACT: Users classified as BigSpender or SmallSpender (SS), based on their usage profile.

3 probabilistic requirements:

R1: "Probability of success is > 0.8 "

R2: "Probability of a ExpShipping failure for a user recognized as BigSpender < 0.035 "

R3: "Probability of an authentication failure is less then < 0.06 "



Assumptions

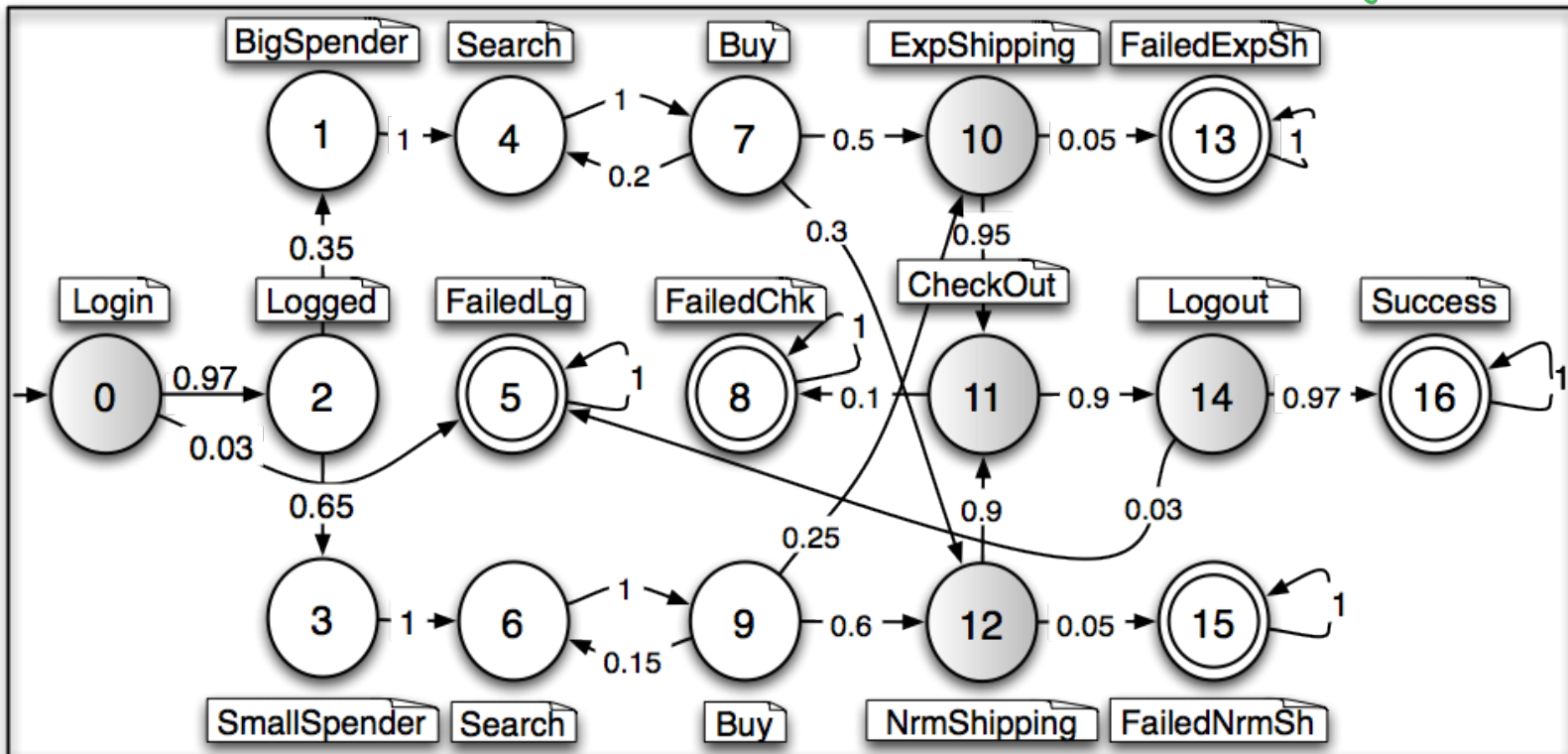
User profile domain knowledge

$D_{u,n}$	Description	Value
$D_{u,1}$	$P(\text{User is a BS})$	0.35
$D_{u,2}$	$P(\text{BS chooses express shipping})$	0.5
$D_{u,3}$	$P(\text{SS chooses express shipping})$	0.25
$D_{u,4}$	$P(\text{BS searches again after a buy operation})$	0.2
$D_{u,5}$	$P(\text{SS searches again after a buy operation})$	0.15

External service assumptions (reliability)

$D_{s,n}$	Description	Value
$D_{s,1}$	$P(\text{Login})$	0.03
$D_{s,2}$	$P(\text{Logout})$	0.03
$D_{s,3}$	$P(\text{NrmShipping})$	0.05
$D_{s,4}$	$P(\text{ExpShipping})$	0.05
$D_{s,5}$	$P(\text{CheckOut})$	0.1

DTMC model



Property check via model checking

R1: "Probability of success is > 0.8 " **0.84**

R2: "Probability of a ExpShipping failure for a user recognized as BigSpender < 0.035 " **0.031**

R3: "Probability of an authentication failure is less then < 0.06 " **0.056**



What happens at run time?

- We monitor the actual behavior
- A statistical (Bayesian) approach estimates the updated DTMC matrix (posterior) given run time traces and prior transitions
- Boils down to the following updating rule

$$m_{i,j}^{(N_i)} = \frac{c_i^{(0)}}{c_i^{(0)} + N_i} \times m_{i,j}^{(0)} + \frac{N_i}{c_i^{(0)} + N_i} \times \frac{\sum_{h=1}^d N_{i,j}^{(h)}}{N_i}$$

A-priori Knowledge

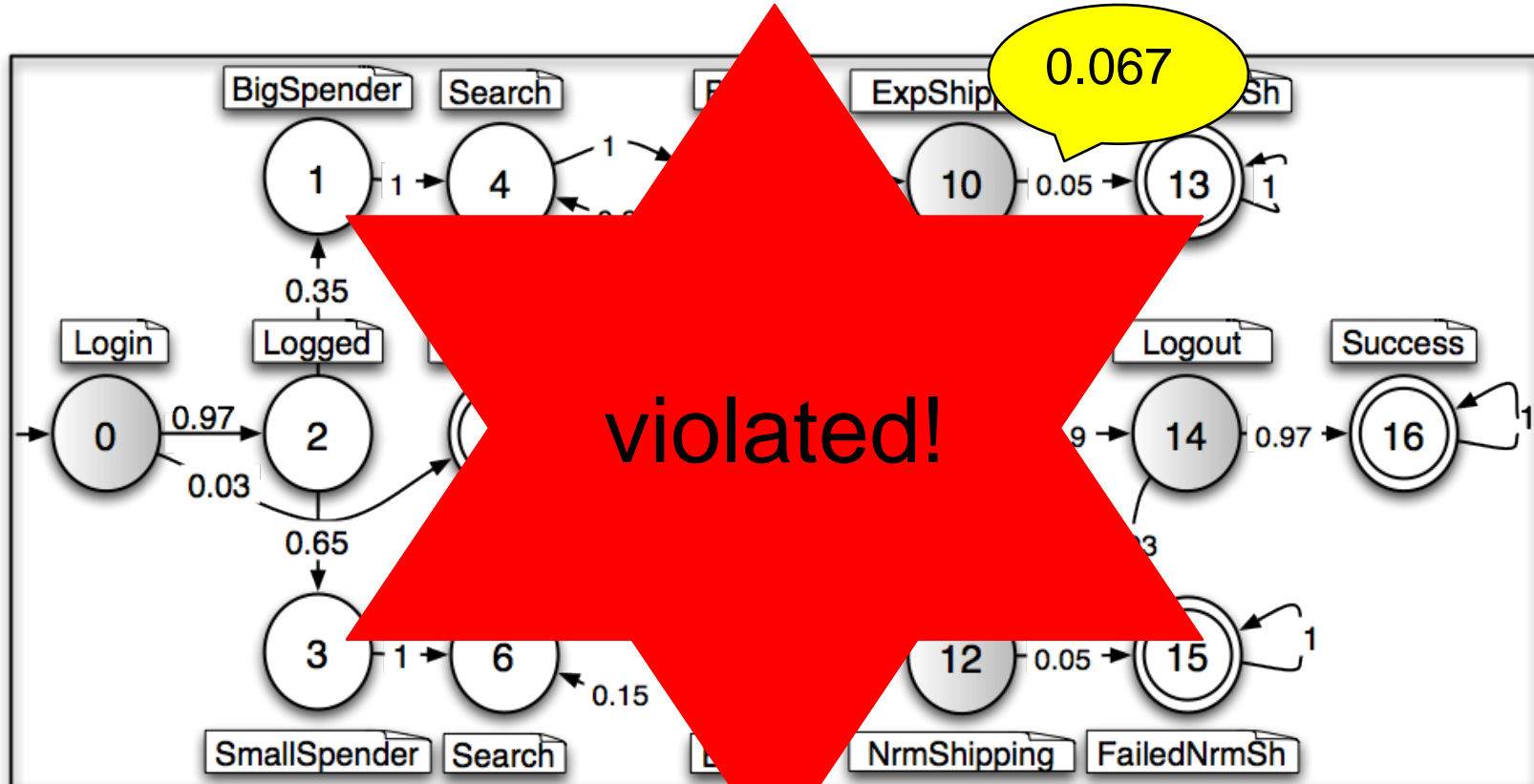
A-posteriori Knowledge



Why is this useful?

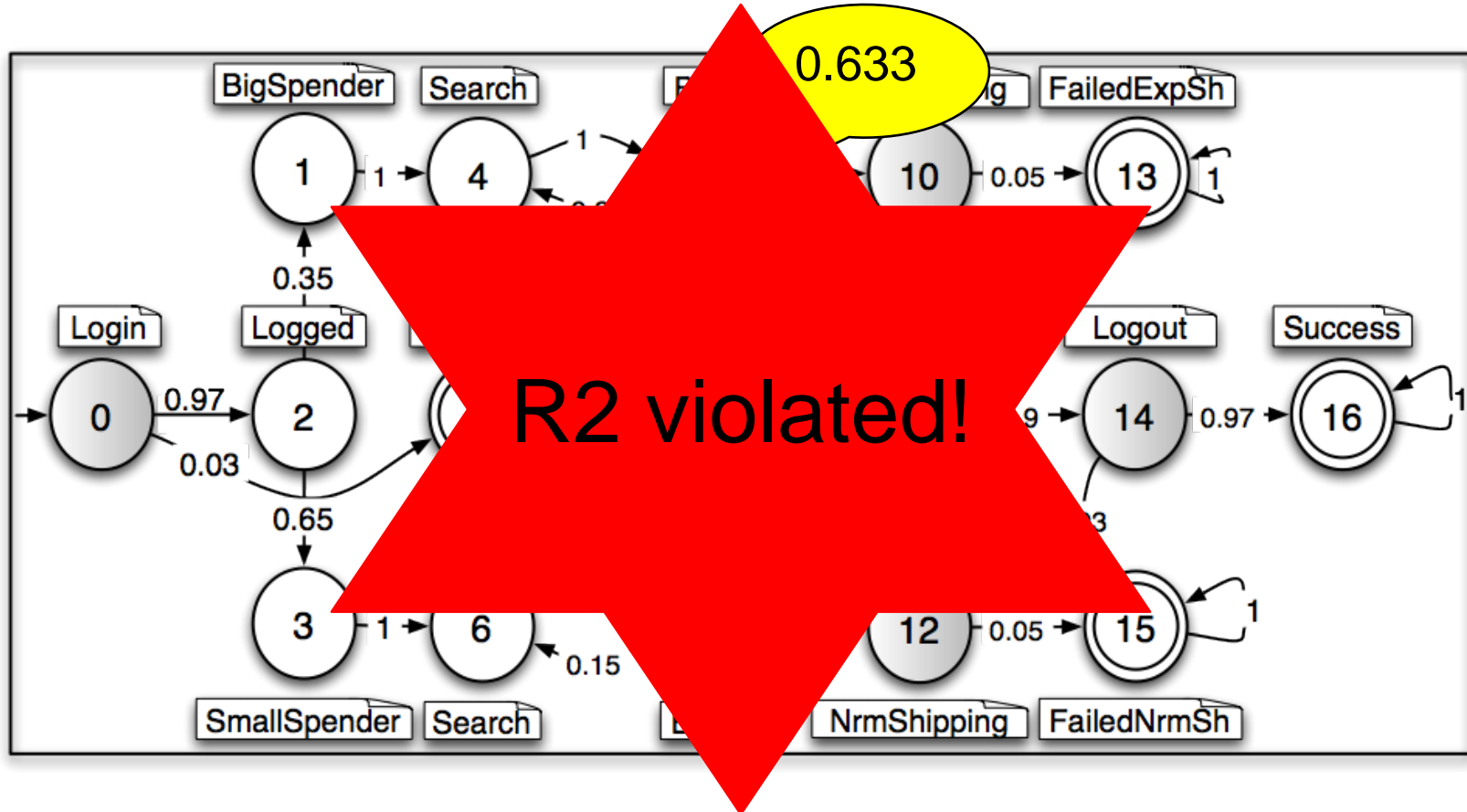
- **Fault**
 - Machine or environment do not behave as expected
- **Failure**
 - Experienced violation of requirement
- Assume that a fault is detected (due to environment).
3 cases are possible
 - All Reqs still valid
 - **OK, but contract violated**
 - Some Req violated + violation experienced in real world
 - **Failure detection**
 - Some Req violated, but violation not experience yet
 - **Failure prediction**

In our example



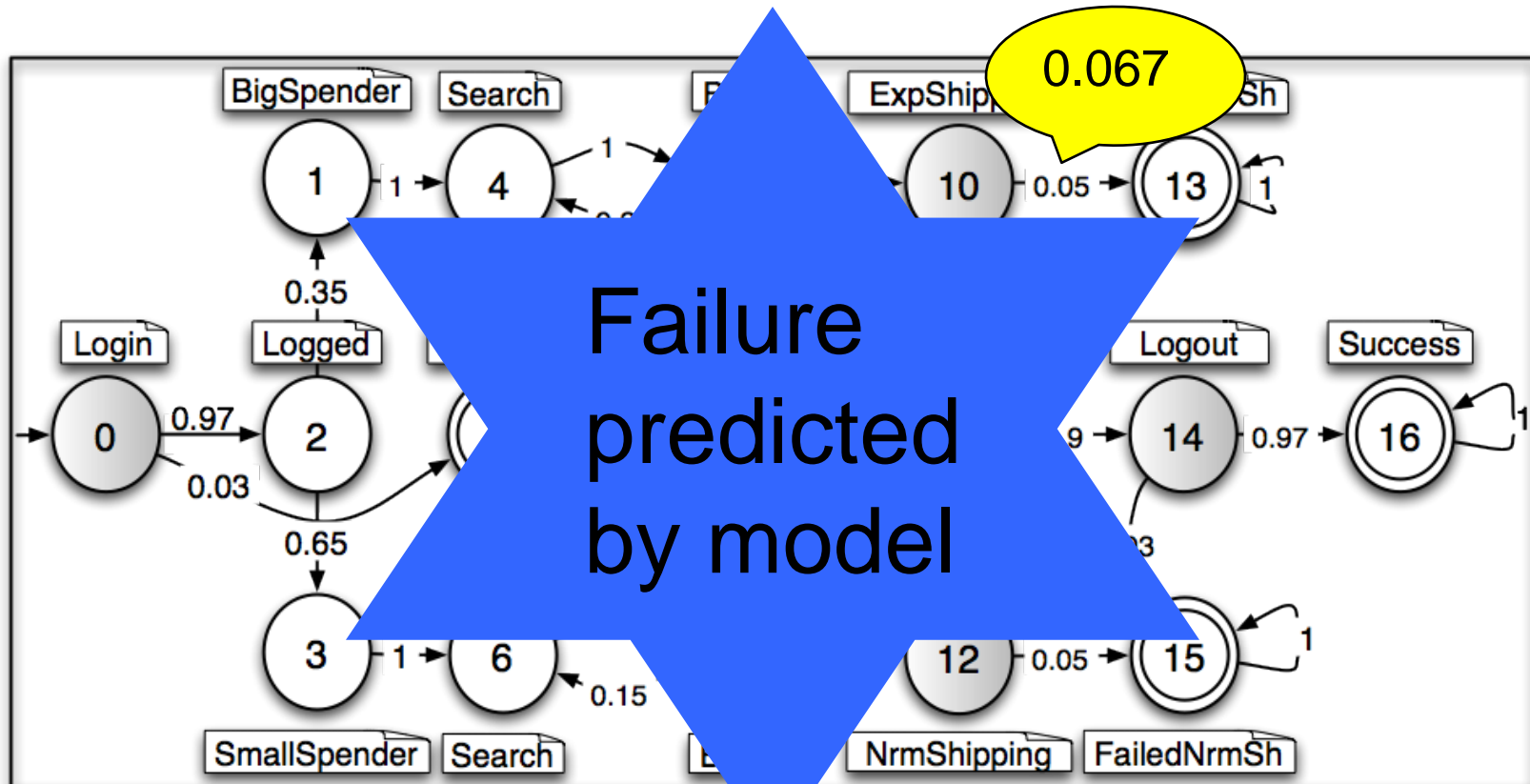
R12: "Probability of an ExpShipping failure for a user categorized as Failure probability
BigSpender < 0.035"

In our example



Similarly, suppose we detect a change in user profile

In our example



Suppose that execution traces that lead to updating the failure probability of ExpShipping are those involving small spenders

BigSpender < 0.035"



Conclusions

- Modern software systems increasingly live in highly dynamic environments and behave in a situational manner
- Design decision are based on quantitative data and are subject to uncertainty
- Boundary between development time and run time vanishes
- Models should be kept alive at run time and should be adapted to changes in the environment
- Detected changes may trigger model-driven adaptation of the implementation
 - Self managed
 - Human-driven, off-line

Bridging between communities



- To support self-adaptation, systems should be able to introspect about their behavior and the world's behavior
- This requires empowering the (often neglected) run-time environment
- Empirical researchers can provide methods and techniques to digging into run-time data and interpreting them

On-going work



We just scratched the surface, much remains to be done

1. Where do requirements come from? How are they elicited?
How do we move from requirements to models?
2. How can a change-point be detected?
3. How can we devise strategies for self-adaptation?
4. Which architectures, middleware, languages are supportive of dynamic change and adaptation?
5. Can we find common realistic case-studies and empirical assessments?
6. How can analysis be done in real time? Incremental analysis techniques?
7. Analysis of partial systems? Inference of specifications?



More details?

L. Baresi, C. Ghezzi, L. Mottola, "Loupe: Verifying Publish-Subscribe Architectures with a Magnifying Lens", IEEE TSE, in press.

I. Epifani, C. Ghezzi, R. Mirandola, G. Tamburrelli, "Model Evolution by Run-Time Parameter Adaptation", ICSE 2009, Vancouver, Canada, May 2009.

C. Ghezzi, G. Tamburrelli, "Reasoning on Non Functional Requirements for Integrated Services", RE 2009, Atlanta, 31 August-4 September 2009.

C. Ghezzi, V. Panzica La Manna, Alfredo Motta, G. Tamburrelli, "QoS Driven Dynamic Binding in-the-many", QoSA 2010, Prague, June 23-25, 2010.

I. Epifani, C. Ghezzi, G. Tamburrelli, "Changepoint Detection for Black-Box Services", FSE 2010, Santa Fe, November 2010.

A. Filieri, C. Ghezzi, G. Tamburrelli, "Run-Time Efficient Probabilistic Model Checking", submitted for publication.

Thanks to the group: these and many others.....





Acknowledgement

- This work is supported by an Advanced Grant of the European Research Council, Programme IDEAS-ERC, Project 227977---SMScom.

The end



questions?

