SwSE

Re-constructing system behaviour from log mining

Governing Adaptive Unplanned System of Systems (GAUSS)

Kick-off meeting

UNIBZ

Barbara Russo

unibz Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Research Gaols and Scope

- System level. Behavioural analysis within eSoS
  - Identify boundary conditions in eSoS (WP5 - online monitoring and data mining)
  - Pattern recognition in log sequences (WP9 - root cause analysis)
- Code level. Feature back-porting
  - Identify code that smells, but worked (WP7 - recovery planning)

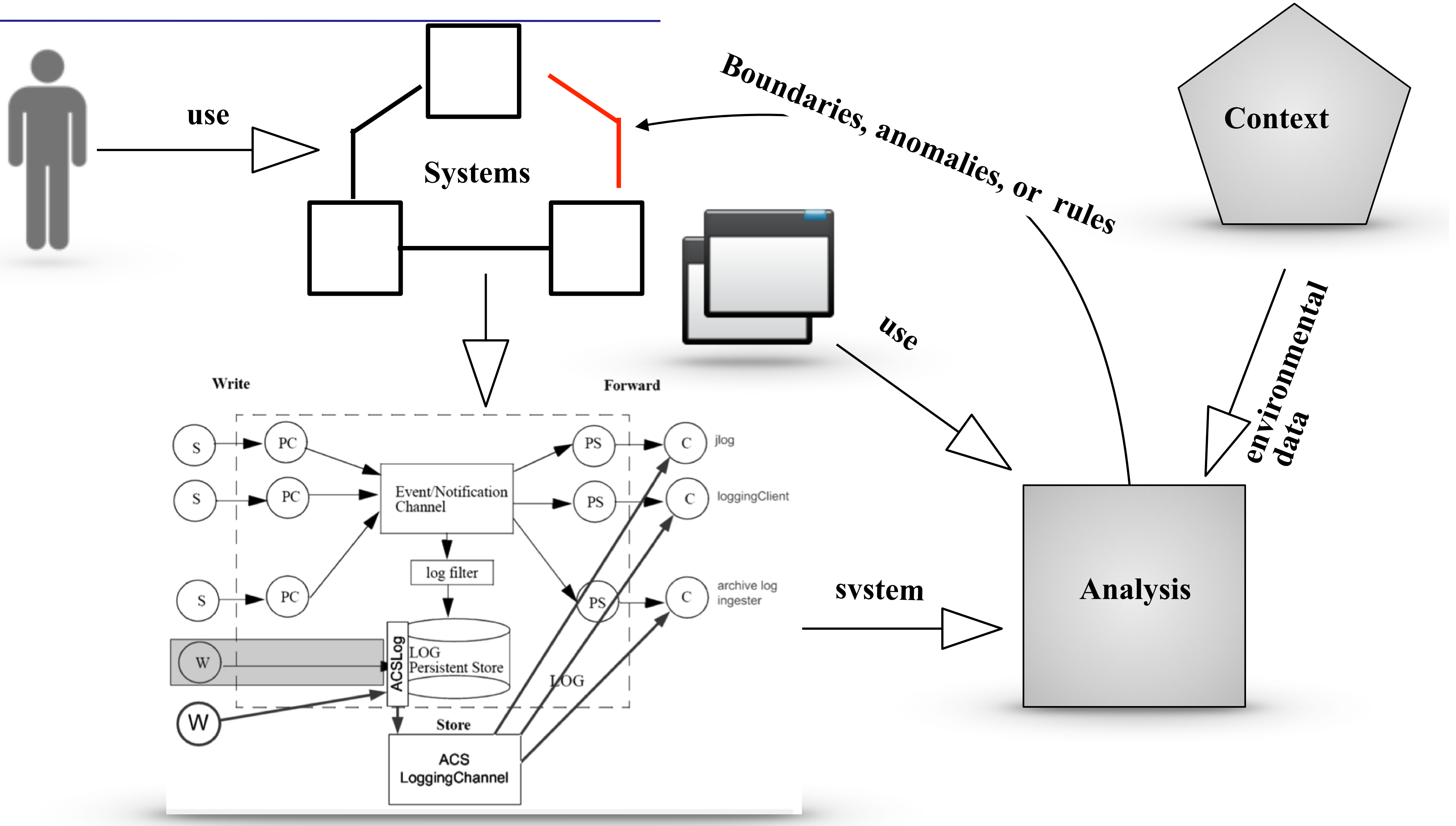
with log analysis

BEHAVIOURAL ANALYSIS

Two complementary perspectives

- System behaviour
  - Re-construct system's **tasks**
  - Characterise them by system's **operations** in response to the use (e.g., patterns)
- System use
  - Re-construct user's **goals**
  - Characterise them by user's **actions** on system (e.g., patterns)

Re-constructing system behaviour



Example of logs

System log

```
<Info TimeStamp=“2015-04-08T07:32:37.345”
 File=“XXX.Control.ObservingModes.ObservingModeBaseImpl" Line="231"
 Routine=“beginSubscan" Host="XXX01" Process=“XXX/javaContainer"
 SourceObject=“XXX/Array005" Thread="RequestProcessor-35023" LogId=“343355"
 Audience=“Operator">

<![CDATA[Text message … here]]>

</Info>
```

User log

```
CASE ID = DCBCB111-EFD7-483E-8DC3-AE4BE11F390B, Activity = Navigate to
ReservierungView, Resource =10.1.0.103., Start Timestamp = 31.03.2016 15:30,
Complete Timestamp = 31.03.2016 15:30, Variant=Variant 623, ApplicationVersion =
2016.330.1027.60, SequenceID= 3
```

Example: pre-processing two log data types

- Times stamps to identify common time window
- Code entities on which both log impact. E.g.,
  - Use log. Code classes that are affected by UI actions (e.g., through chain of responsibility of listeners)
  - System log. Code classes that are involved in the system operations

Available test beds

- Not encrypted data
  - Mobile ERP system (single system - user logs)
  - ACATAMA ALMA system (independent systems - system logs)
- Encrypted data
  - Paul Grünbacher University of Linz (system logs from simulator to define rules for embedded systems); goal: find more rules than expert’s experience to **increase safety**

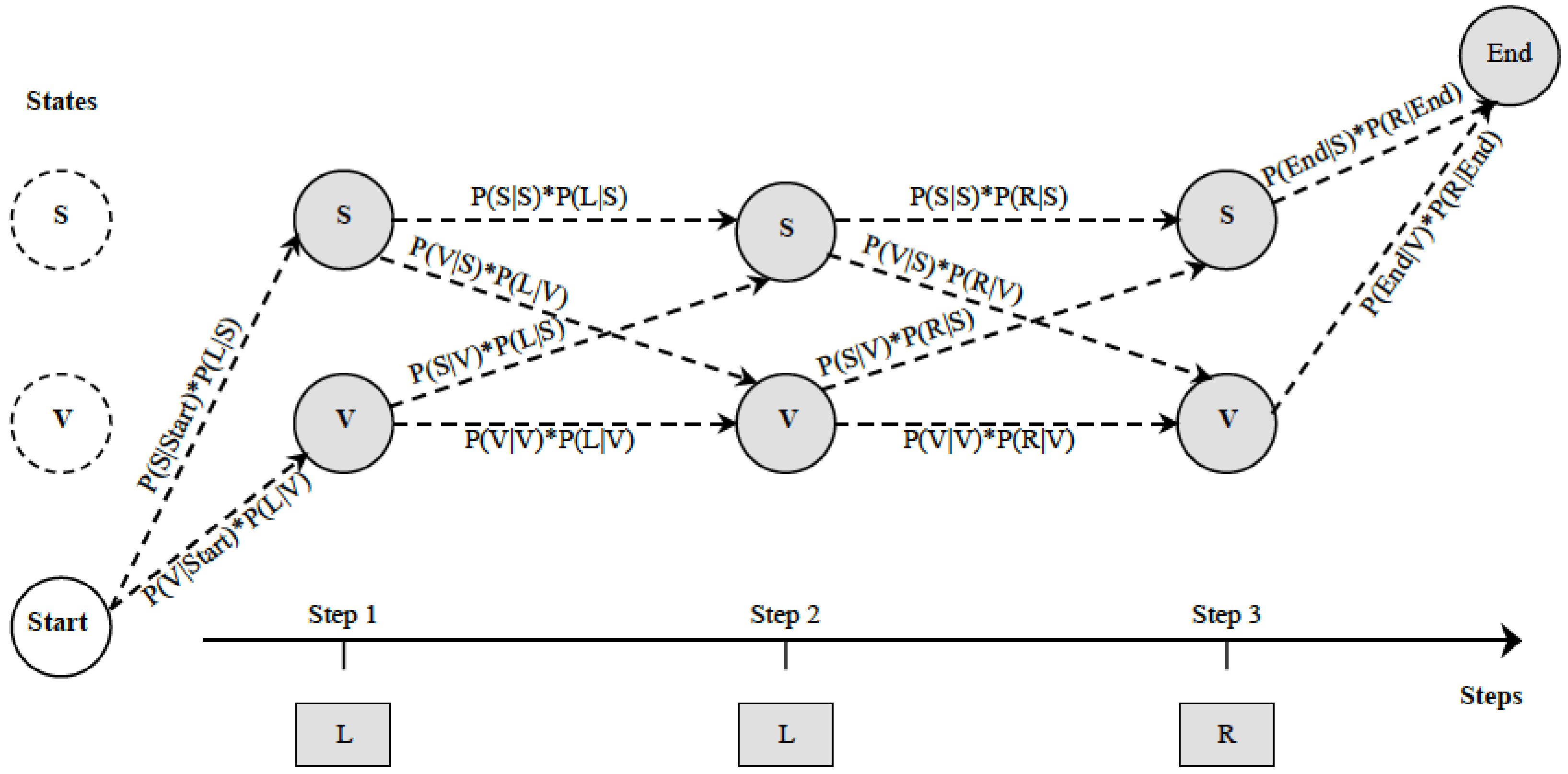
Techniques

- To label log events
  - Text processing for short messages
- To identify log sequences
  - clustering techniques to identify sequences
- To re-construct system's task or user's goals
  - Unsupervised learning (Iterative Hidden Markov Models)
- To detect pattern or characterise behaviour
  - Supervised learning (Classifiers) or genetic algorithms

Example

- Supervised learning, learn error predictors and **classify** log sequences for faultiness
  - User behaviour: sequences of user's actions that lead to deviation from **expected use output**
  - System behaviour: sequences of system's events that lead to deviation from **expected behaviour**

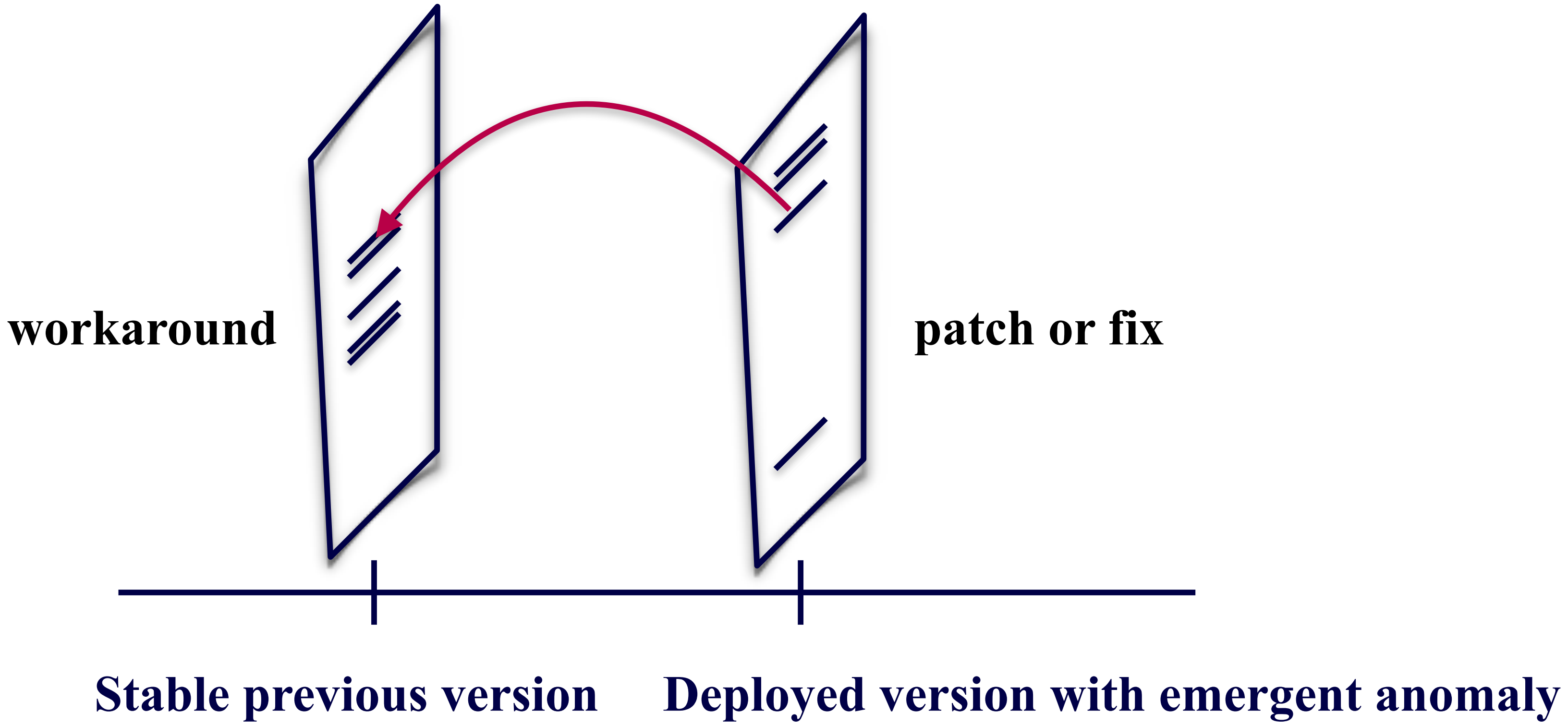
Example unsupervised learning - HMM



K. Damevski, H. Chen, D. Shepherd, L. Pollock. 2016. Interactive exploration of developer interaction traces using a hidden markov model. MSR '16, 126-136 to define rules of on-line repair

IDENTIFY CODE THAT SMELLS, BUT WORKED

Feature or patch back-porting



Y. Li, J. Rubin and M. Chechik, "Semantic Slicing of Software Version Histories (T)," *ASE2015*, Lincoln, NE 2015, pp. 686-696

Few recent references

- Barbara Russo, Giancarlo Succi, Witold Pedrycz (2015) Mining system logs to learn error predictors: a case study of a telemetry system, Empirical Software Engineering Journal
- Saulius Astromskis, Gabriele Bavota,; Andrea Janes, Barbara Russo, Massimiliano Di Penta, A 1,000-hour Study on Developers' Activities in an Industrial Context, under review
- Saulius Astromskis, Andrea Janes, Michael Mairegger, A process mining approach to measure how users interact with software: an industrial case study. ICSSP 2015: 137-141
- Gabriele Bavota, Barbara Russo: A large-scale empirical study on self-admitted technical debt. MSR 2016: 315-326