Semiautomatic Security Requirements Engineering and Evolution using Decision Documentation, Heuristics, and User Monitoring

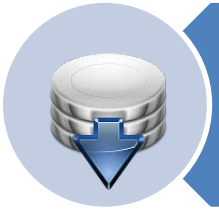
Tom-Michael Hesse, Barbara Paech
University of Heidelberg
Stefan Gärtner, Kurt Schneider
Leibniz Universität Hannover
Tobias Roehm, Bernd Bruegge
Technische Universität München

DFG
Design For
FUTURE

Overview

Motivation

Components for Our Approach



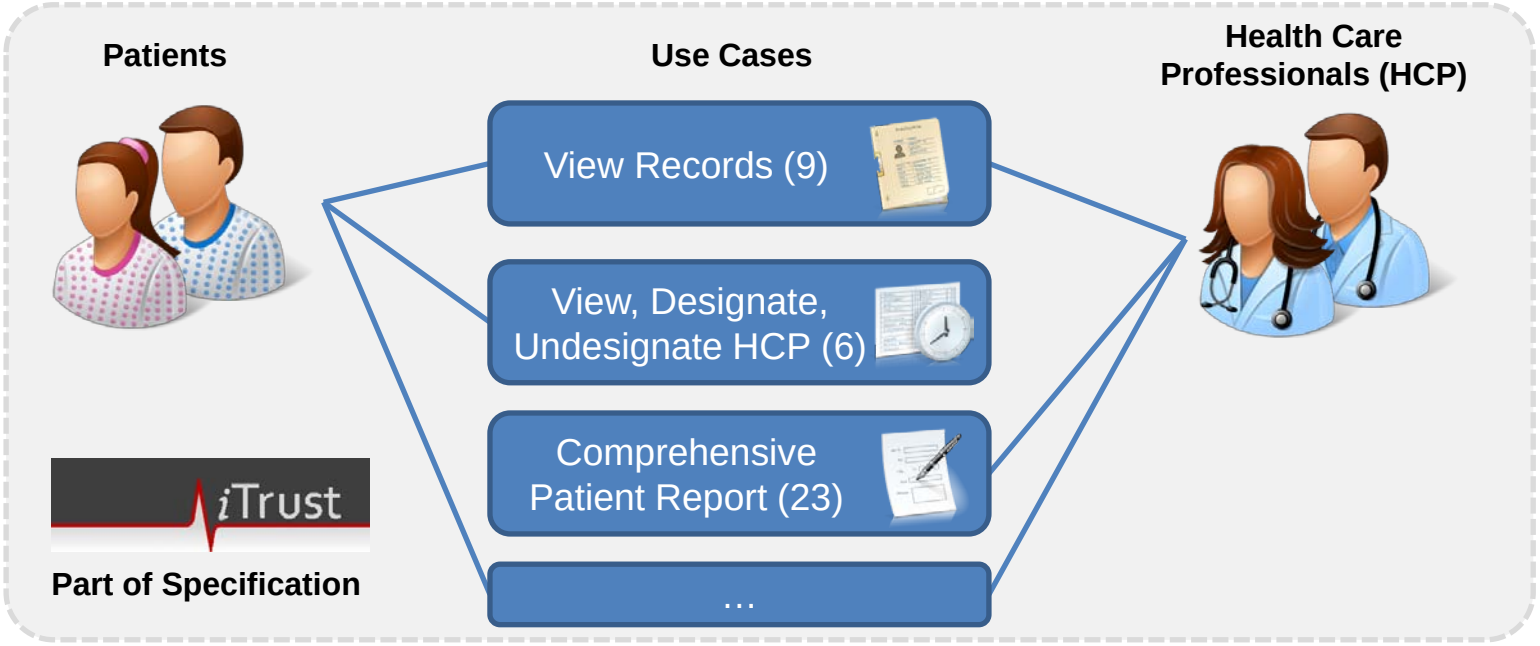
Incorporating Decision Knowledge from Different Sources

Benefits, Related Work, Conclusion

Running Example iTrust

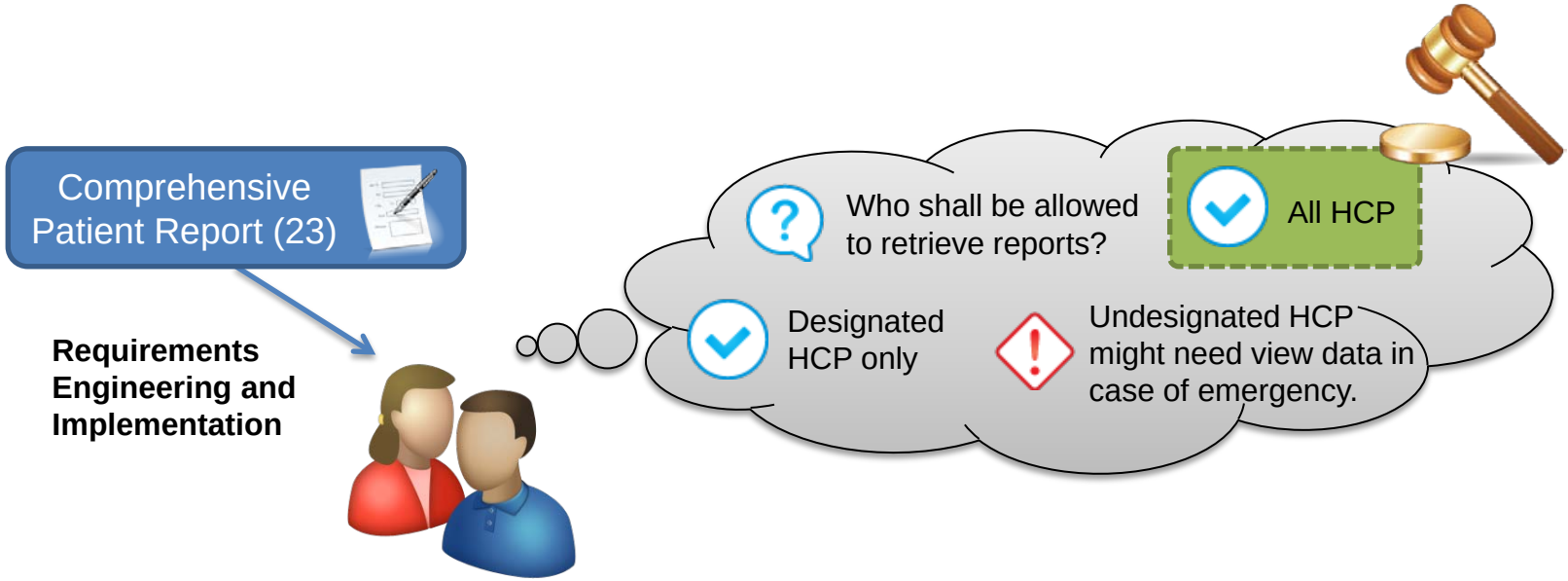
Running Example: Medical Information System iTrust

- Medical information system iTrust: Management of health records for patients and work schedule for staff
  - Specified in 55 use cases and implemented as web application by Realsearch Research Group (North Carolina State University)
  - Active evolution (23 versions of requirements, 17 versions of code)



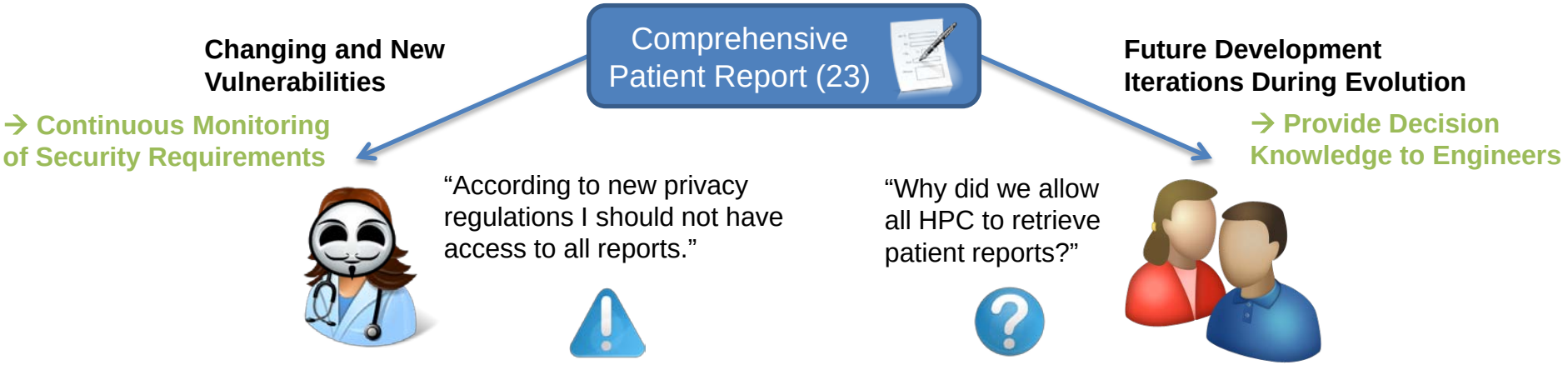
Motivation: Decisions on Security Are Important

- Great importance of security requirements for long-living information systems
- Many decisions on how to implement and prioritize security requirements



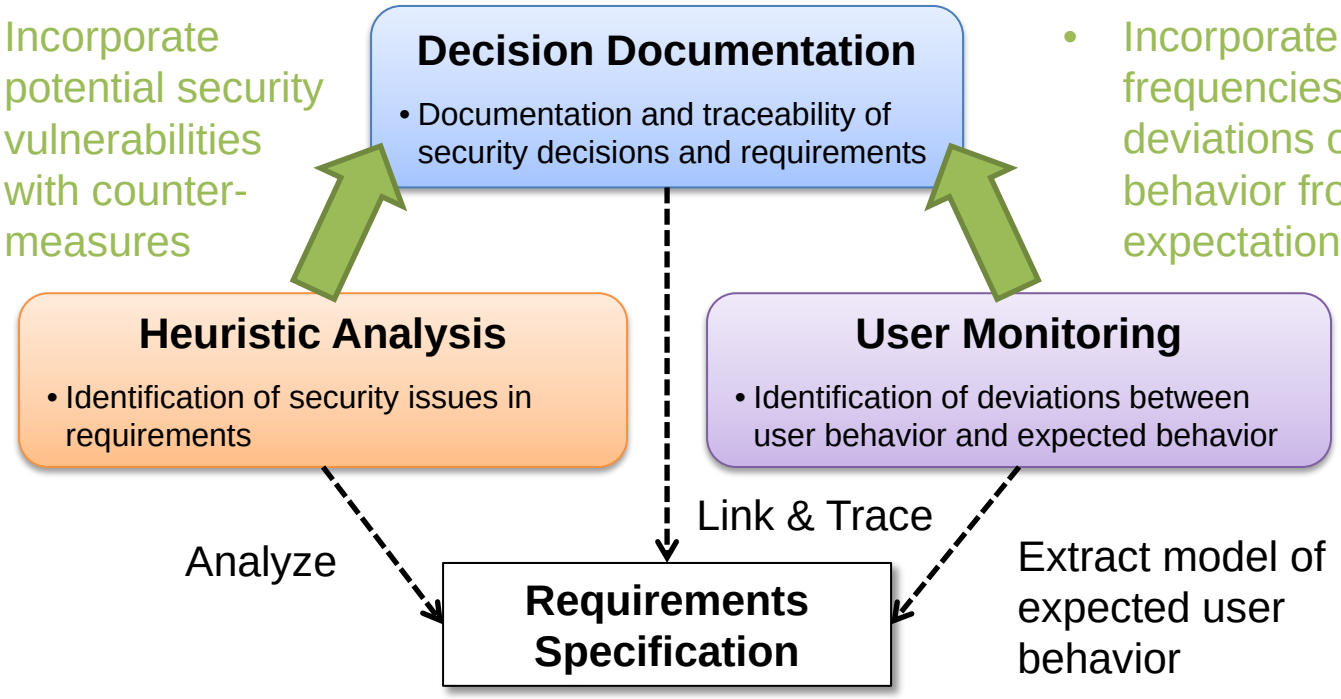
Motivation: Decision Knowledge Erodes Quickly

- But: Security requirements engineering is often performed manually → Erosion of knowledge about decisions on security requirements
  - Incomplete documentation
  - Quickly outdated decisions and documentation



Our Approach: Semiautomatic Decision Documentation

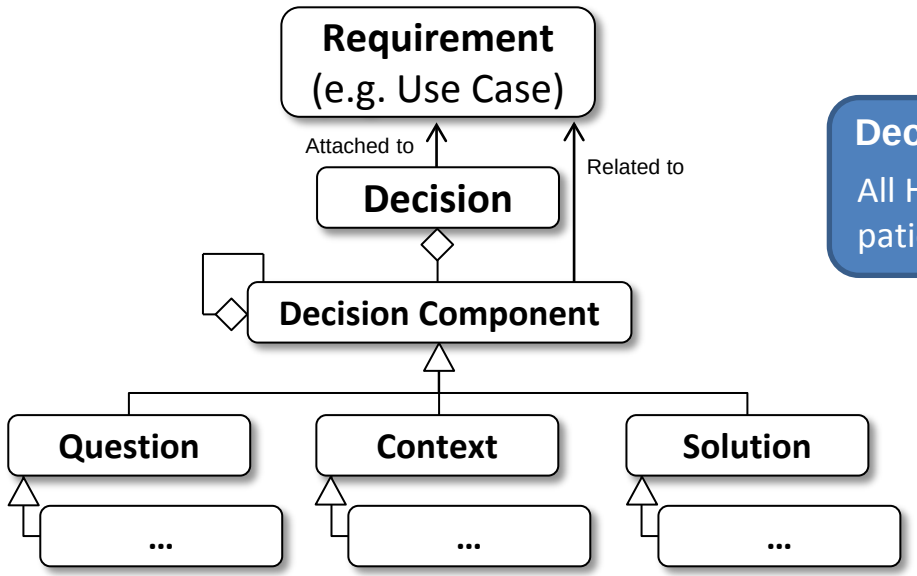
- Incorporate potential security vulnerabilities with counter-measures



- Incorporate use case frequencies and deviations of user behavior from expectations

Component: Decision Documentation

- Decision knowledge is knowledge about all decision-related questions, solutions, context information/rationales
- Benefit of documentation: Support software engineers in finding and exploiting previous security-related decisions



Decision
All HCPs can retrieve patient reports.

Question: Issue
Who shall be allowed to retrieve patient reports?

Solution: Alternative
Designated HCPs should be able to retrieve reports.

Context: Constraint
Undesignated HCPs might need to view data in case of emergency.

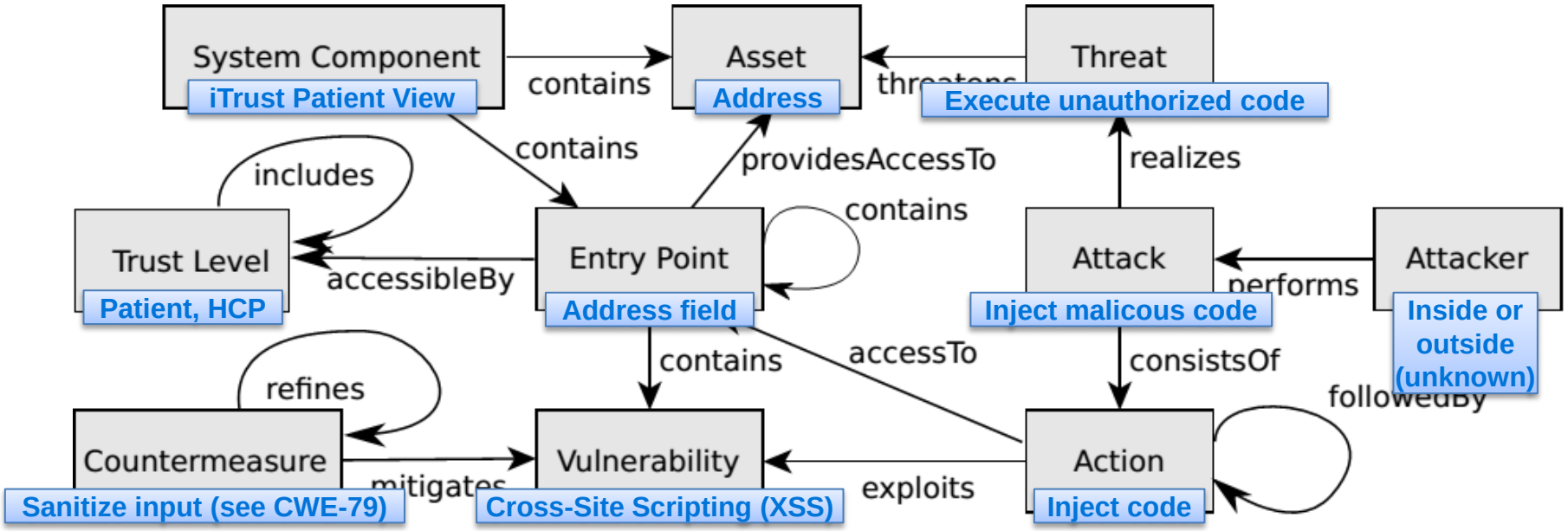
Component: Security Heuristics

- To decrease effort and support evolution, natural language requirements need to be assessed automatically
- Attack-centric approach:
  - Transform requirements in analysis model using NLP
  - Search for suspicious interactions by incorporating general attack patterns and reported security incidents
  - Retrieve potential vulnerabilities and countermeasures from the security knowledge base

- Example for use case 6:
  - The patient types a last name or partial last name, and/or providing the specialty or address.

Component: Security Heuristics

- Modeling security knowledge (e.g., incidents, security guidelines and policies, …)
  - Example for cross-site scripting in iTrust patient view



Component: User Monitoring

End users

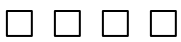
interact

Software

1. Monitor user interactions with software sensors

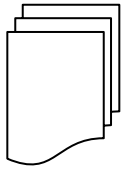
Result:
Use case frequency
→ Prioritization

2. Detect current use case



Log of User Interactions

3. Compare log and specification

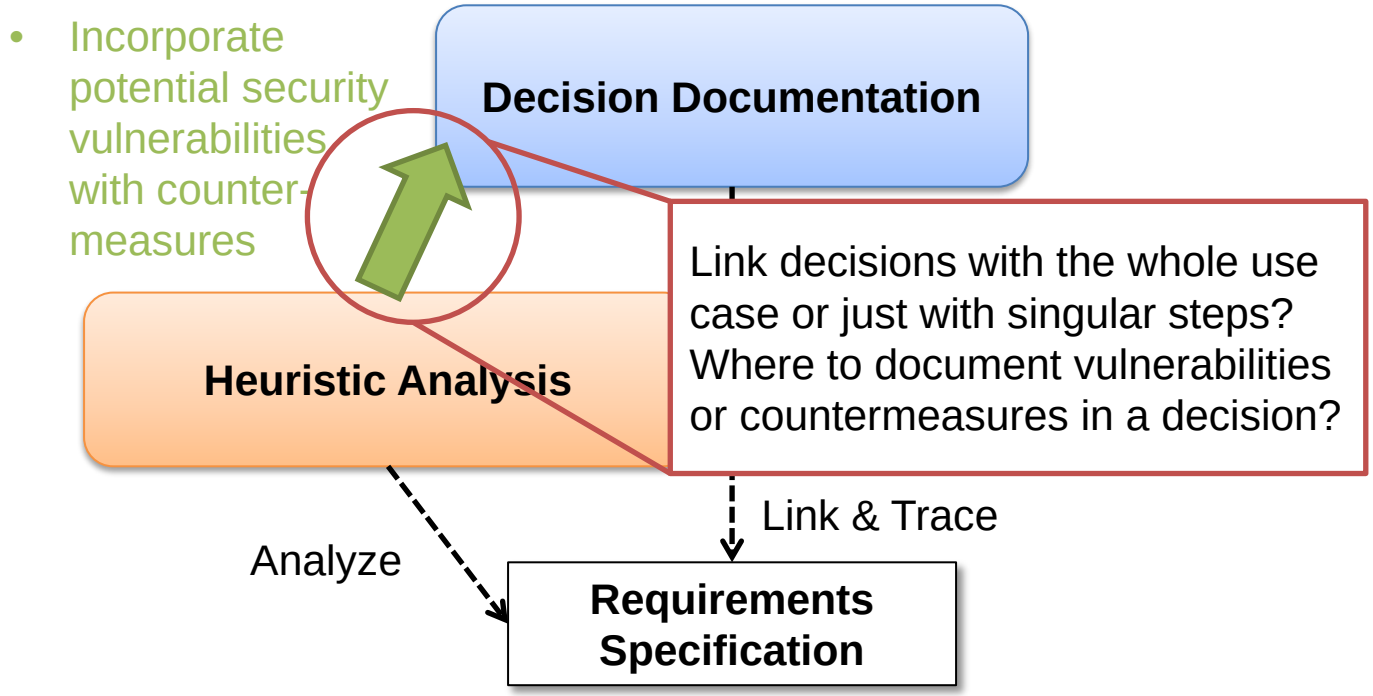


Use Case Specification

Result:
Difference between log and specification (e.g. missing logout)
→ Indicator for software improvement?

Incorporating Security Knowledge

- Now: How to incorporate security knowledge into decision documentation?



Incorporating Security Knowledge
Step 1: Group Potential Decisions

Group potential decisions → Map findings to decision knowledge → Elicit missing decision knowledge

- Step 1: One decision is created automatically for each issue with links to all steps of the usage scenario it was found in
  - Potential XSS vulnerability for step 3 of use case 6

Use Case 6: View, Designate, Undesignate HCP
1. Dislapy HCP‘s name, specialty, address.
2. Toggle to designate/undesignate an HCP.
3. Search an HCP by name or address.

- In some cases: Adjust decision grouping or discard them

Incorporating Security Knowledge
Step 2 and 3: Generate Decision Knowledge

- Step 2: Default rules are provided for mapping heuristic knowledge with decision knowledge and generate texts

Heuristic Knowledge	Decision Knowledge	Generated Content for XSS Decision on Use Case 6, Step 3
Vulnerability, System Component	Decision	Prevent *Cross-Site Scripting* in *iTrust Patient View*
Counter-measure	Alternative	*Sanitize input (link to mitigation strategies of CWE-79)*
…		

- Step 3: Add or adapt generated contents, if necessary

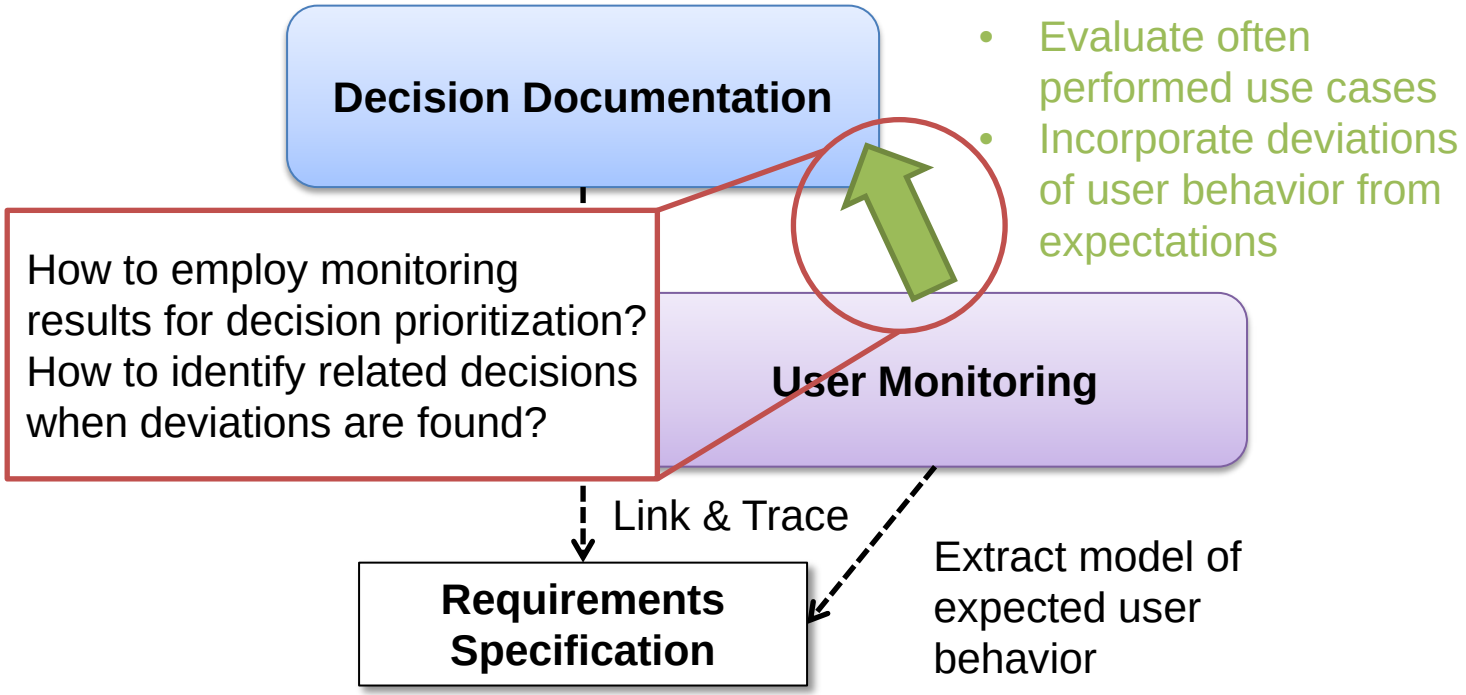
Benefits

- Comprehensive and fast access to knowledge on security-related decisions for software engineers and project managers
  - Decision knowledge is incorporated from heuristic findings by a semiautomatic transformation process

- Awareness for security issues in decision-making without requiring software engineers or project managers to be security experts

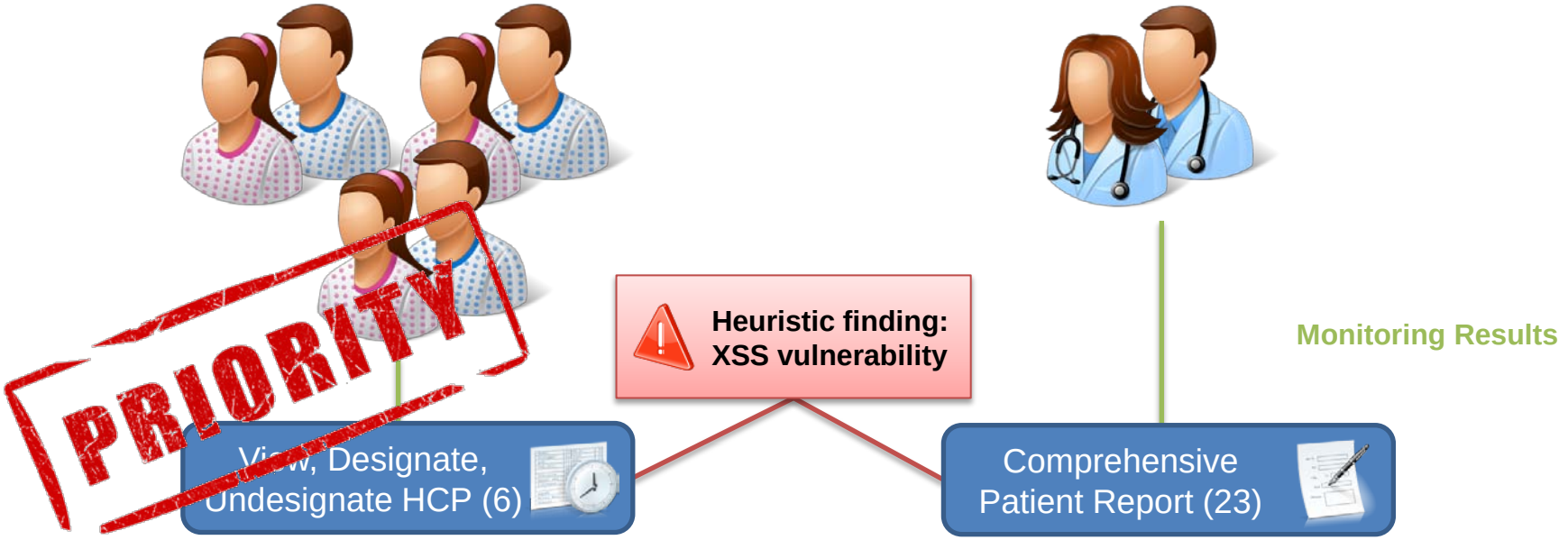
Incorporating User Monitoring Data

- Now: How to incorporate results from user monitoring?



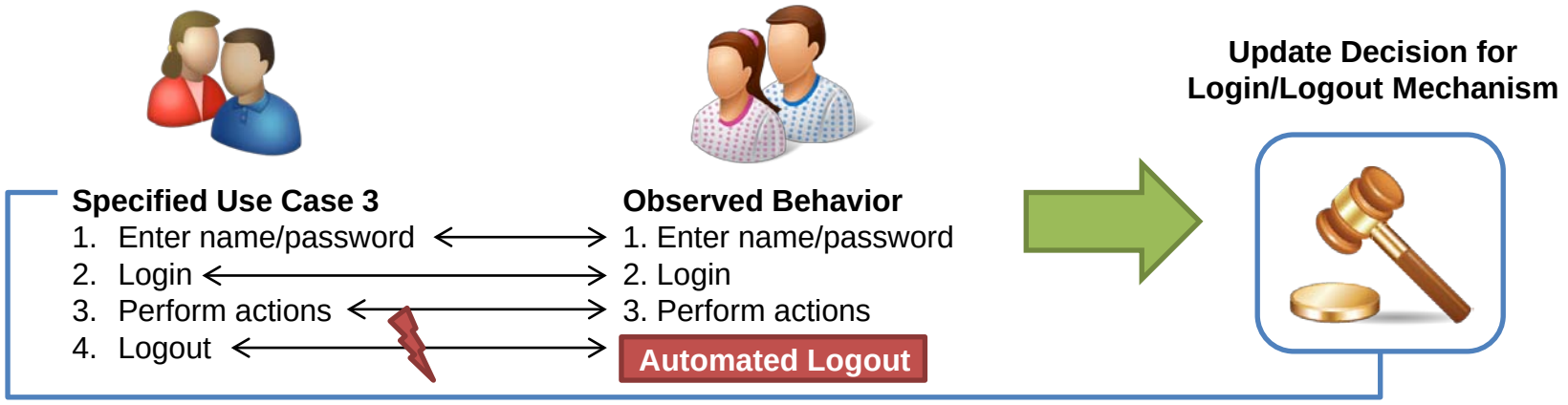
Incorporate User Monitoring Data Prioritize Decisions

- Case 1: Prioritize heuristic findings
  - Decisions on security requirements, which concern frequently performed usage scenarios, should be handled with priority
  - For instance, if patients perform use case 6 more often than use case 24, an XSS vulnerability should be handled first for use case 6



Incorporate User Monitoring Data Uncover Required Updates

- Case 2: Uncover required knowledge updates
  - Deviations are uncovered between specified (use cases) and monitored user behavior
  - Documentation component retrieves all decisions related to the use case for which the deviation was found
  - Software engineers can then decide how to update the given documentation



Benefits

- Prioritization of decisions helps software engineers to manage large amounts of potential decisions uncovered by heuristic analysis

- Updates on security-related decisions and their documentation can be located easily by software engineers
  - Deviations between specified and observed end user behavior indicate needed updates

Conclusion and Future Work

- We propose to document decisions on security requirements by incorporating different knowledge sources
- Our approach improves the documentation quality and completeness for decisions on security requirements
- We are currently developing a prototype to demonstrate our approach within the knowledge management tool UNICASE
- Directions for future work:
  - Evaluation of our approach in a controlled experiment
  - Integration of further knowledge sources for documentation

The End

Thank you for your attention!

Do you have any questions?

Tom-Michael Hesse
Institute of Computer Science
Chair of Software Engineering
Im Neuenheimer Feld 326
69120 Heidelberg, Germany
http://se.ifi.uni-heidelberg.de
hesse@informatik.uni-heidelberg.de

References

- **[Aurum2006]** A. Aurum, C. Wohlin, and A. Porter. Aligning Software Project Decisions: A case study. *International Journal of Software Engineering and Knowledge Engineering*, 16(06):795–818, Dec. 2006.
- **[Braz2008]** F. a. Braz, E. B. Fernandez, and M. VanHilst. Eliciting Security Requirements through Misuse Activities. In *Proc. of the 19th Int. Conference on Database and Expert Systems Applications*, pp. 328–333. IEEE, 2008.
- **[Haley2008]** C. B. Haley, R. C. Laney, J. D. Moffett, and B. Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Trans. Software Eng.*, 34(1):133–153, 2008.
- **[Nuseibeh2009]** B. Nuseibeh, C. B. Haley, and C. Foster. Securing the Skies: In Requirements We Trust. *IEEE Computer*, 42(9):64–72, 2009.
- **[Sindre2005]** G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.