Maintaining Requirements for Long-Living Software Systems by Incorporating Security Knowledge

Stefan Gärtner and Kurt Schneider
Software Engineering Group, Leibniz Universität Hannover, Germany

Thomas Ruhroth, Jens Bürger, and Jan Jürjens
Chair of Software Engineering, TU Dortmund, Germany

International Requirements Engineering Conference (RE) 2014

Overview

- Motivation and Research Questions
- Our Approach and its Components
- iTrust Case Study
- Conclusion and Future Work

“Not bad kid, but you‘d vulnerable
to attacks here and here.“

Motivation

- Security is an important quality facet of software systems.
- Identifying vulnerabilities in requirements is important to elicit new security requirements as well as to make reasonable design decisions.
- Manual assessment approaches (e.g. reviews, inspections) are time-consuming and security expertise is required.
- Security assessments have to be repeated if environmental knowledge changes.

Motivation

Time

Assumptions about Environment and Knowledge of Attacker

"It is difficult to spy information from a secure chip."

Change in Knowledge

"Open APIs (display+keyboard) can be used to fake dialogs, phish info."

VULNERABLE!

Use of internal and secure chips **prevents the leakage of PINs**

No changes in System

Attack using additional dialogs, so that the **customer enters PIN in an insecure mode**

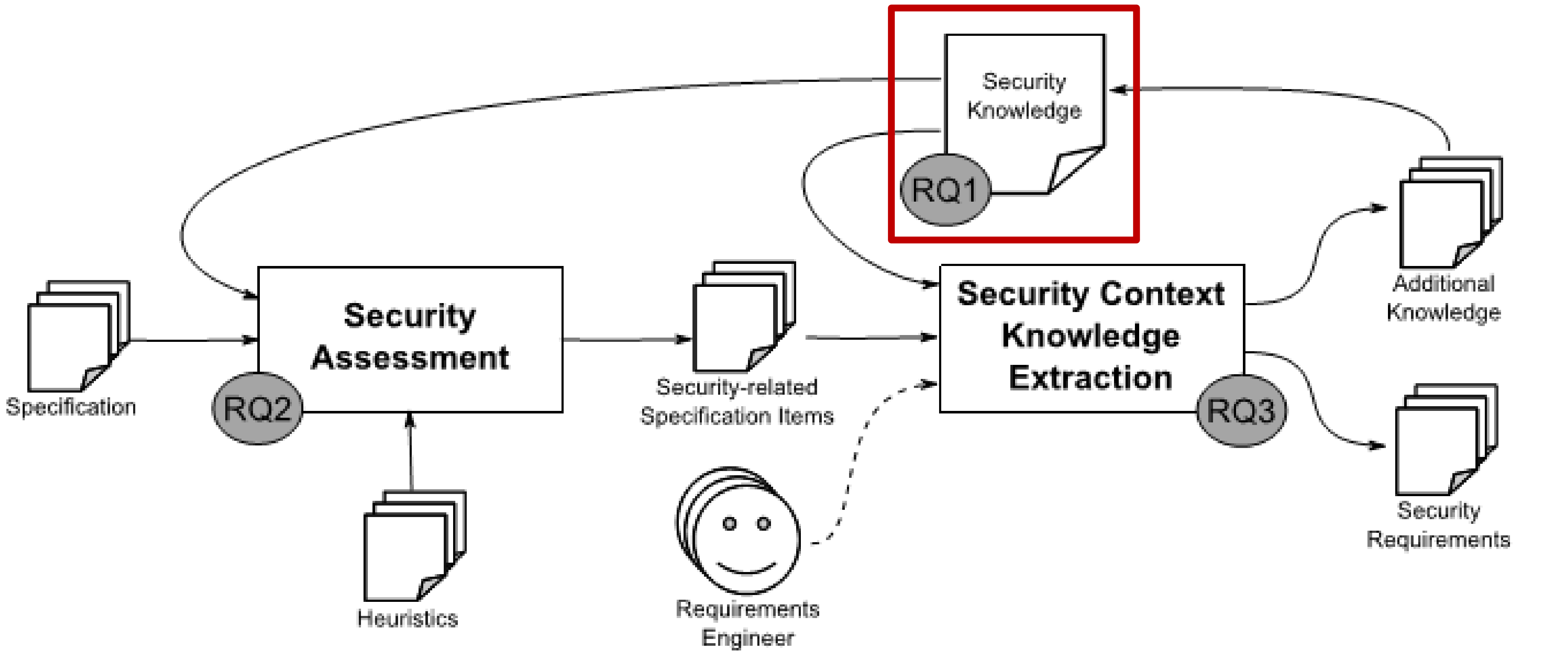
Research Questions

RQ1: How to organize security knowledge in a way that it can be used for assessing requirements of a long-living software system?

RQ2: How can requirements engineers identify security-critical issues in natural language requirements semiautomatically?

RQ3: How can requirements engineers be supported to extract proper security knowledge from identified security-critical issues in requirements?

Overview of our Approach



Security Knowledge

- Modeling security knowledge must be flexible enough to cope with *Unknown Unknows*

- Knowledge can rapidly change or become invalid

- Continously adapting knowledge is necessary

View	Structure Model	Content Model				Integrated Modelling Theory
		Generic Content Model		Domain-specific Content Model		
Exemplary Repre-sentation	Taxonomies	Narrative Description	Guidelines	Concept Models	Concept Models with Conformance Constraints	Mathematical Models
Characteristic	Flexibility					Calcuability

[Fernandez2010]

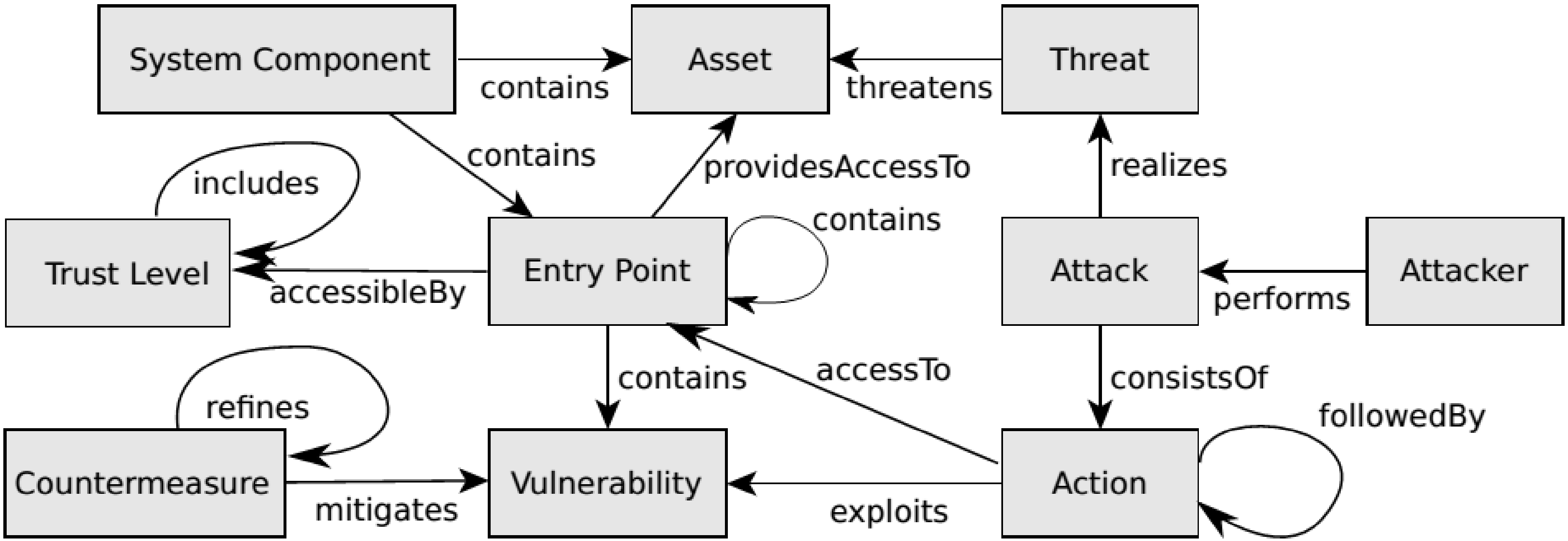
Security Concepts ans Relationships

- SLR to find a suitable security concepts and their relationships (attack-centric security knowledge)

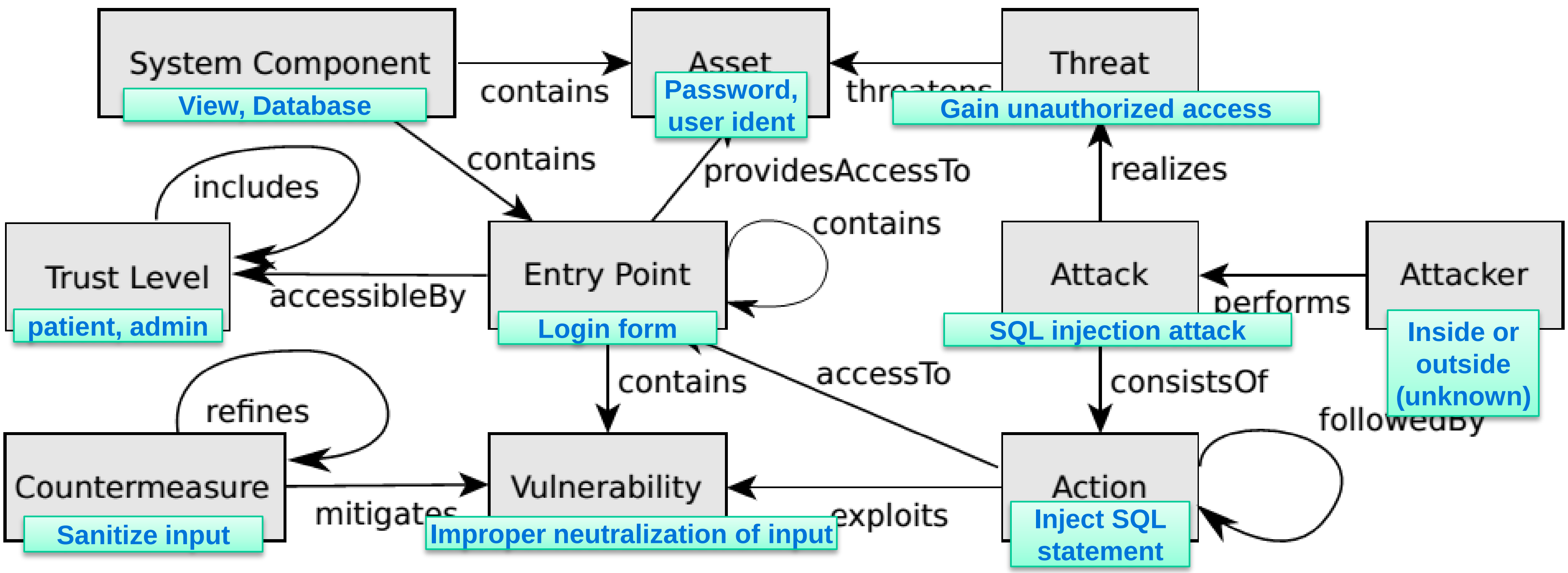
- Reviewed 16 publications from following areas:
  - Threat modeling
  - Risk analysis
  - Computer and network security
  - Software vulnerabilities
  - Information security management

- Focused on information systems, cyber-physical systems, distributed systems, and agent-based systems

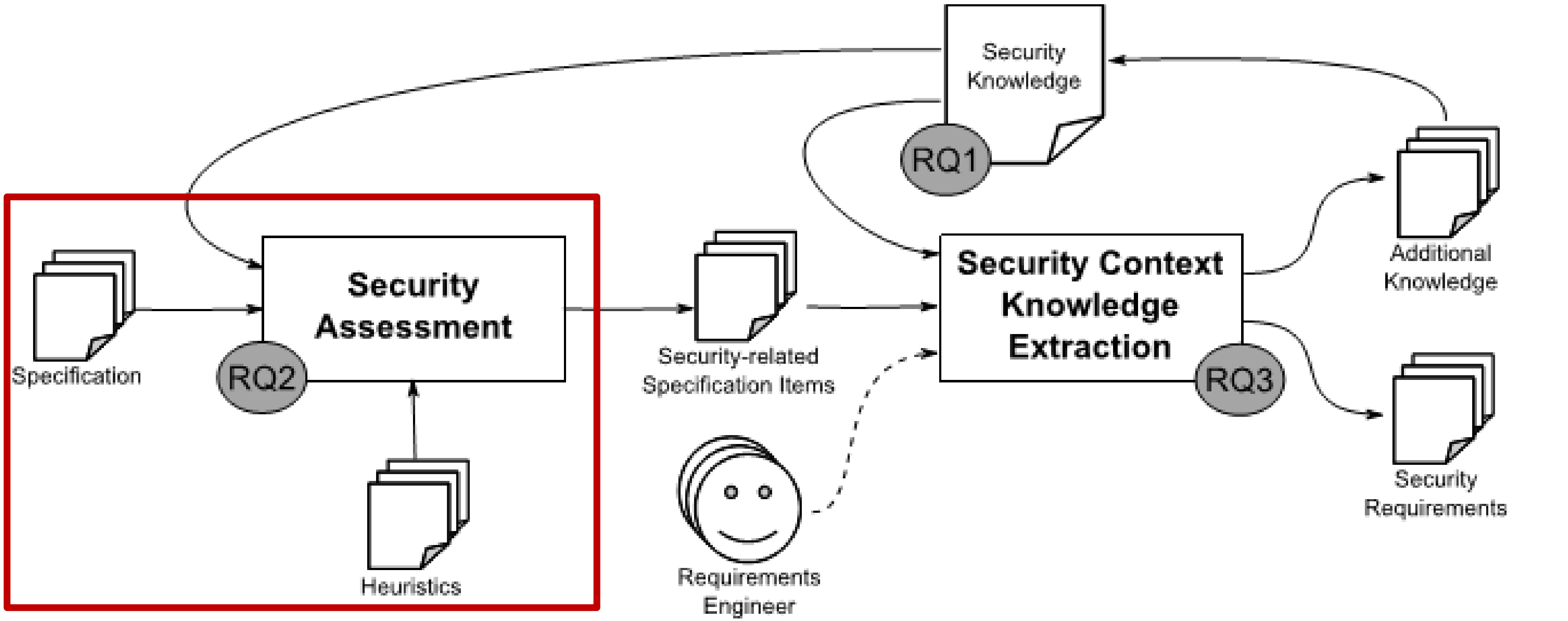
Security Concepts ans Relationships (cont.)



Security Concepts ans Relationships (example)



Overview of our Approach



Heuristics in Requirements Engineering

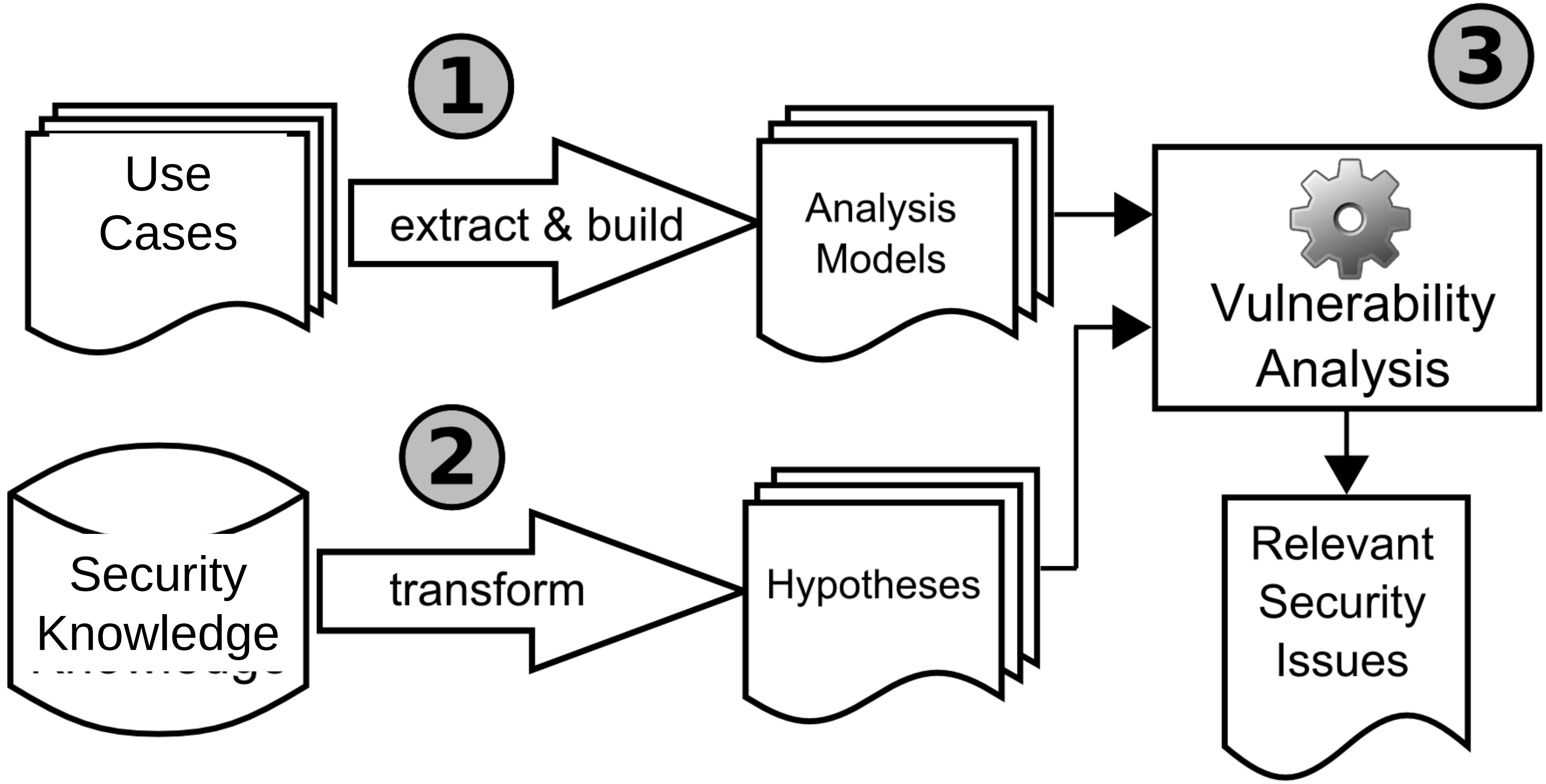
Definition: A heuristic is an analytical method based on hypotheses to assess requirements with respect to security.

Remarks:

- Heuristics are able to cope with incomplete and uncertain knowledge
- Heuristic findings are suboptimal (false positives)
- Hypotheses may evolve for long-living software systems

Security Assessment

- To decrease effort and support evolution of environmental knowledge, natural language requirements need to be assessed automatically



Step 1: Creating Analysis Model

1. The user enters an email address.
2. The user enters her PIN.
3. If successful the user is logged in. Otherwise, the system displays a message to inform the user whether the email address or the PIN are incorrect.

1. Extract relevant nouns

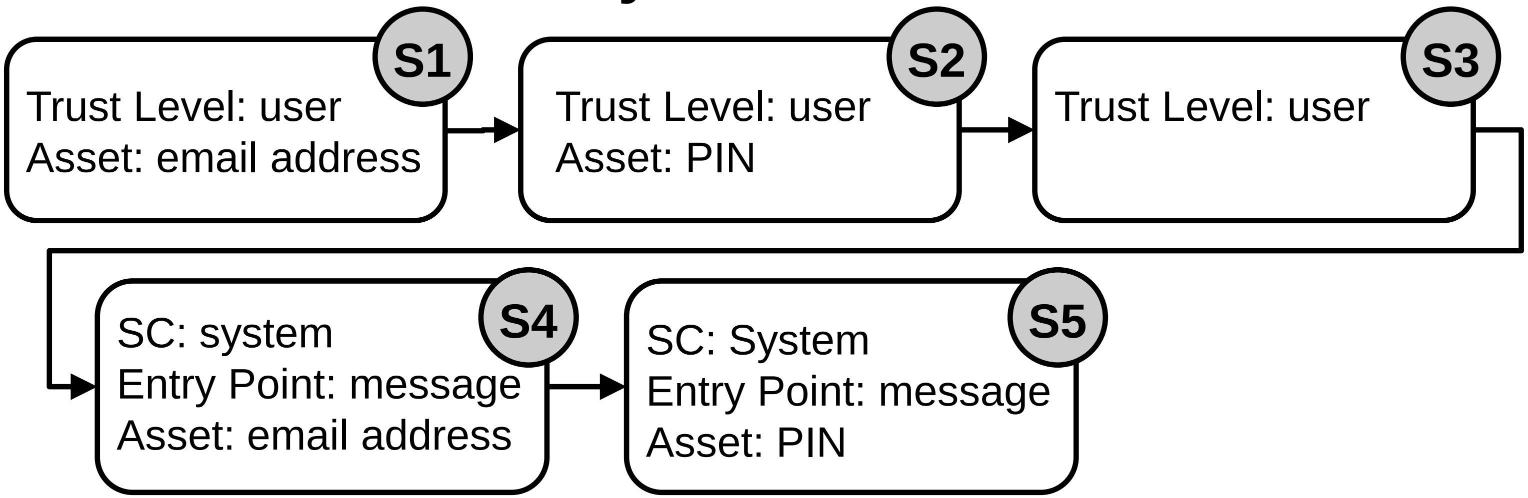
1. The **user** enters an **email address**.
2. The **user** enters her **PIN**.
3. If successful the **user** is logged in. Otherwise, the **system** displays a **message** to inform the **user** whether the **email address** or the **PIN** are incorrect.

Step 1: Creating Analysis Model (cont.)

2. Label nouns according to the security knowledge

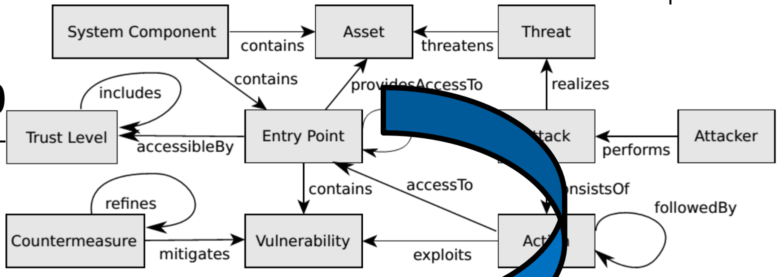
1. The user enters an email address.
2. The user enters her PIN.
3. If successful the user is logged in. Otherwise, the system displays a message to inform the user whether the email address or the PIN are incorrect.

3. Transform to analysis model



Step 2: Extract Hypotheses from Knowledge

1. The attacker selects an user identifier and attempts to login with a random password.
2. If the systems displays a message that the identifier is incorrect, the attacker knows that a corresponding account exists.
3. The attacker tries to guess the p



Transform to analysis model

A1
Trust Level: attacker
Asset: identifier, password

→

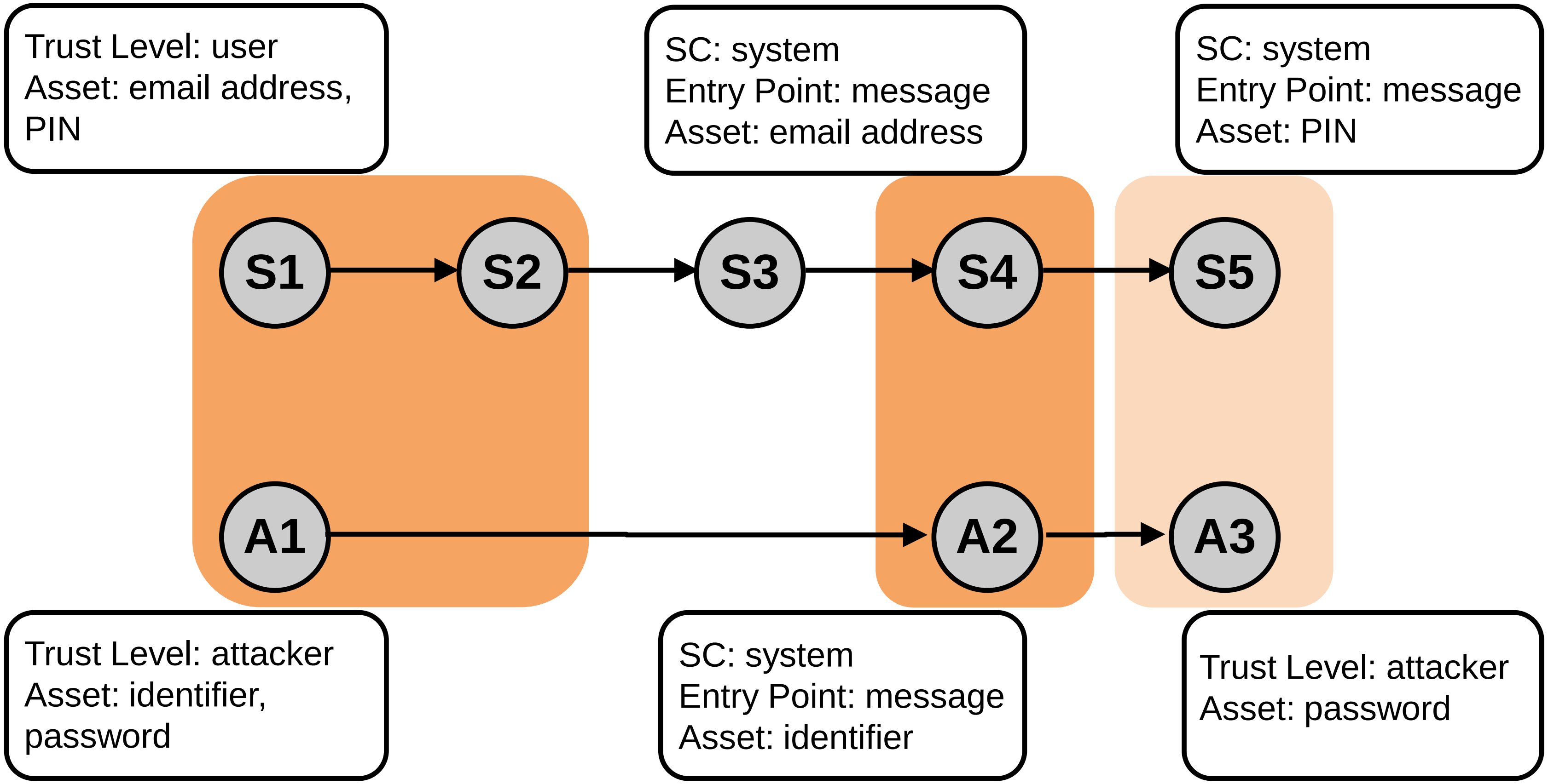
A2
SC: system
Entry Point: message
Asset: identifier

→

A3
Trust Level: attacker
Asset: password

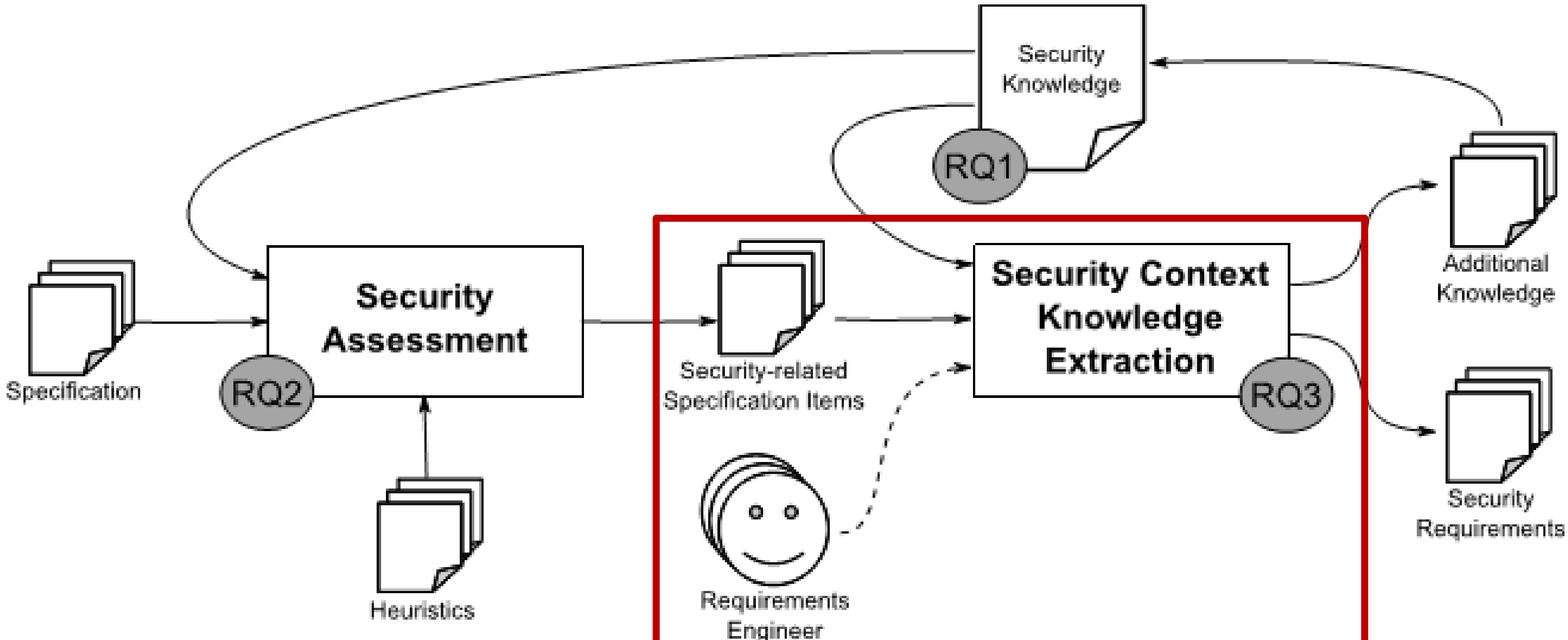
Step 3: Vulnerability Analysis

- Analysis models are semantically matched using WordNet (taxonomy-based semantic similarity)



→ Suspicious sequence has been detected (potential vulnerability)

Overview of our Approach



Security Context Knowledge Extraction

- To support manual knowledge extraction, the requirements engineering is guided by the heuristic findings

- Acquiring new knowledge by leveraging linguistic structure of sentences

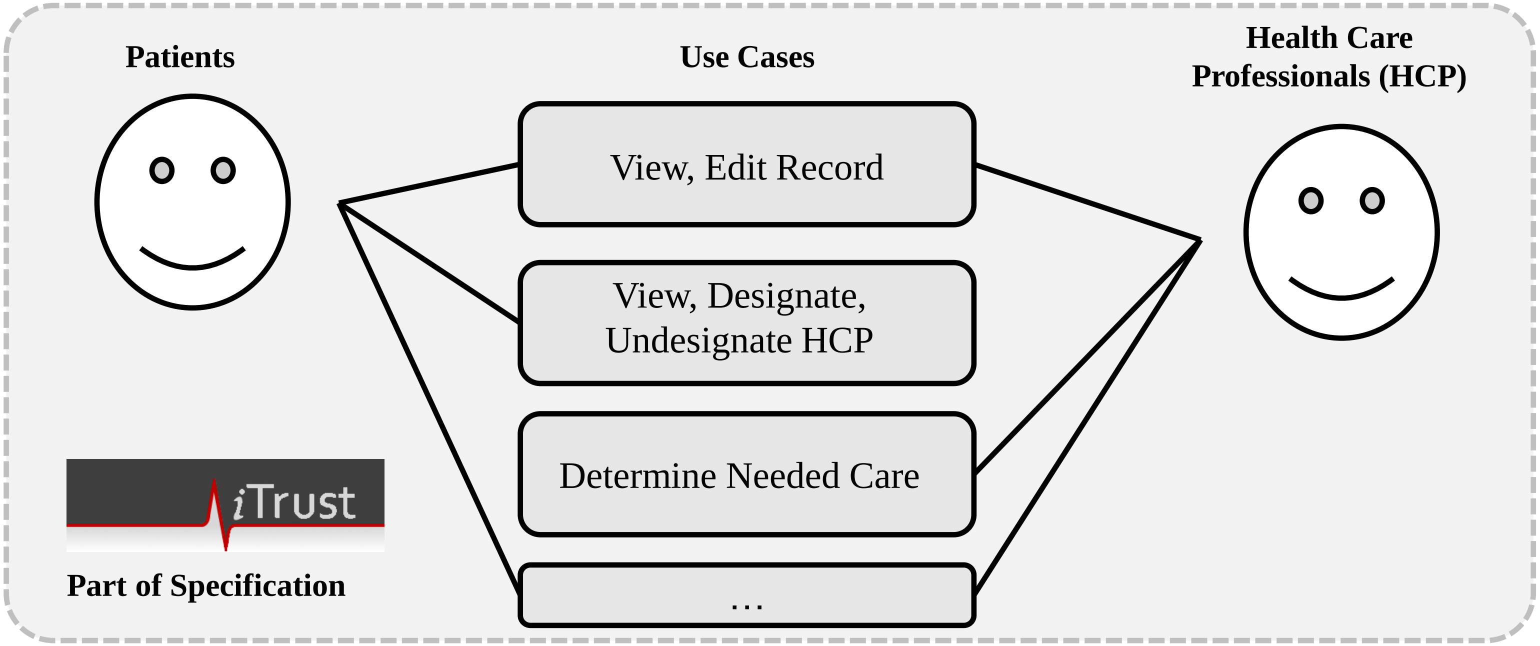
> The user is requested to enter her **<u>email address</u> {Asset}**, **<u>PIN</u> {Asset}**, and a secure transaction number {Asset?}.

- Modify, reinforce, and refine existing knowledge

> The **<u>IP address</u> {→ email address?}** of the user is logged after an error occurs.

iTrust Case Study

- Medical information system iTrust: Management of health records for patients and work schedule for staff
- Specified in 55 use cases written in natural language
- Implemented as web application by Realsearch Research Group (North Carolina State University)



iTrust Case Study - Design

- To setup security knowledge and misuse cases, 10 UCs have been selected randomly
- Misuse Cases (MUC) have been obtained manually
- All UCs were evaluated by a security expert according to the MUCs
- To simulate evolution, the case study is performed in 2 iterations (44/55 UCs)
- Our approach is compared to Naive Bayes (NB), Support Vector Machine (SVM), and k Nearest Neighbor (k-NN)

iTrust Case Study - Results

		ACC	FPR	FNR
1st Iteration (n=44)				
Our Approach	MUC 1	0.90	0.10	0.00
	MUC 2	0.64	0.55	0.15
Naive Bayes	MUC 1/2	0.61	0.00	0.89
SVM	MUC 1/2	0.57	0.00	1.00
k-NN	MUC 1/2	0.57	0.15	0.83
2nd Iteration (n=55)				
Our Approach	MUC 1	0.98	0.00	0.14
	MUC 2	0.85	0.14	0.17
Naive Bayes	MUC 1/2	0.71	0.11	0.67
k-NN	MUC 1/2	0.76	0.00	0.68

iTrust Case Study - Discussion

- Results indicate that the proposed concepts and their relationships are sufficient (**RQ1**)
- Vulnerable UCs could be identified automatically and results are better than NB, SVM, and k-NN (**RQ2**)
- After knowledge refinement (2nd iteration), false positive were reduced (**RQ3**)
- MUCs have been set up by the project team → more empirical studies are needed (e.g. industrial case study)

Conclusion and Future Work

- Heuristic security assessment and knowledge extraction approach to identify vulnerable requirements
- Our approach supports established assessment approaches
- Case study shows that the proposed approach basically works
- Leverage structural dependencies between UCs to consider attacks that affect more than one UC
- Further studies to evaluate the proposed approach