University of Essex

UMM AL-QURA UNIVERSITY
Honored Place and Honored Education

Structural Analysis of the Check Point Pattern

Abdullah A. H. Alzahrani,
Amnon Eden,
Majd Zohri Yafi

Cyberpatterns 2014 (SOSE 2014), 7-8 April 2014, Oxford, The United Kingdom

Motivation

- Claim

JAAS implements the checkpoint pattern* (Alur et al. 2003)

- Question
  - Can we tell it's true?

- Problem:
  - Verify conformance of source code to a security pattern

* Oracle Java Technology (2011)

So, what we need to answer the question?

1. Check Point pattern description

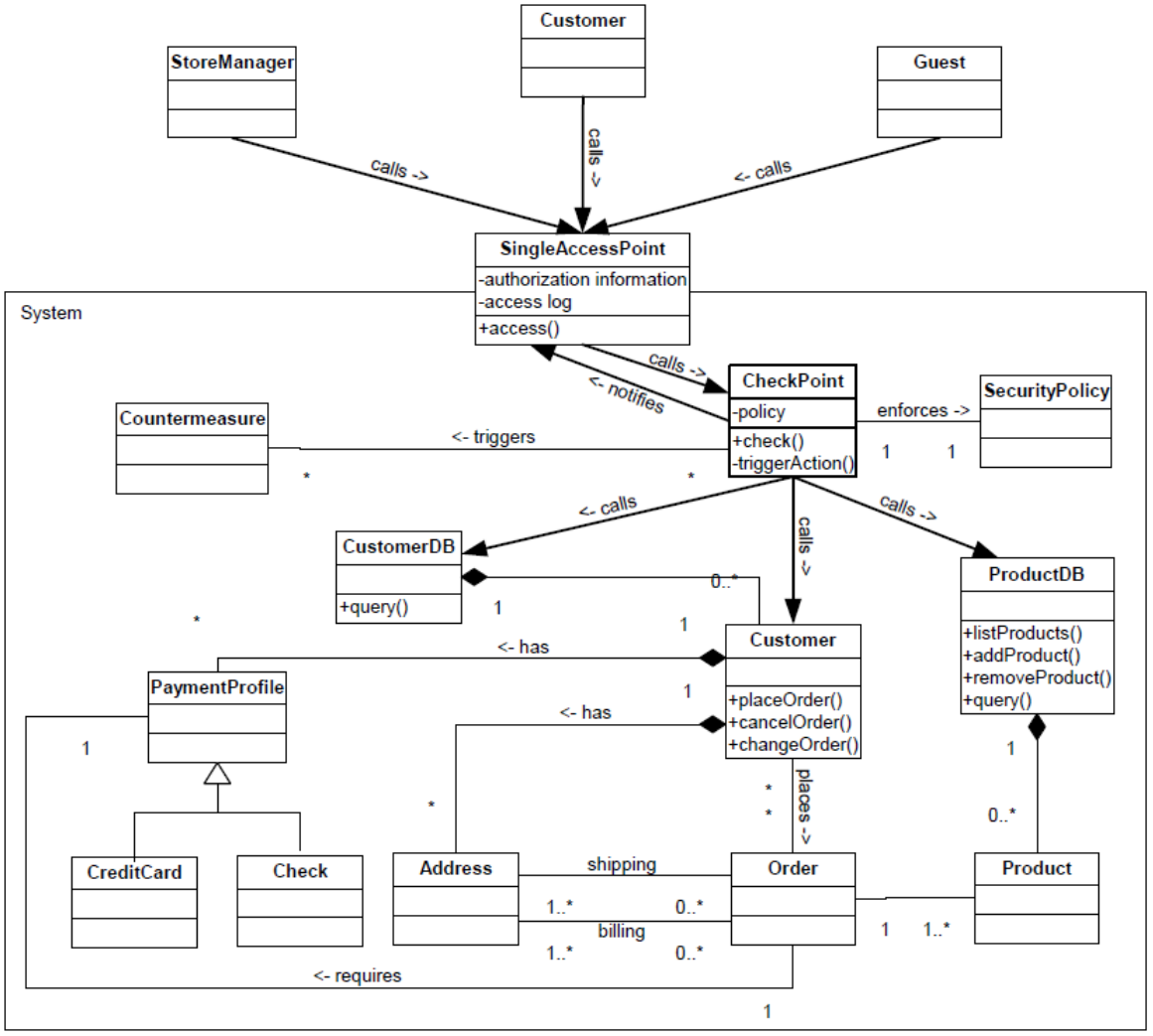
2. JAAS source code
   - Can be found in http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html.

Check Point Pattern

Intent: A component that intercepts and monitors all incoming requests. In case of violations the it is responsible for taking appropriate countermeasures.

Participants:

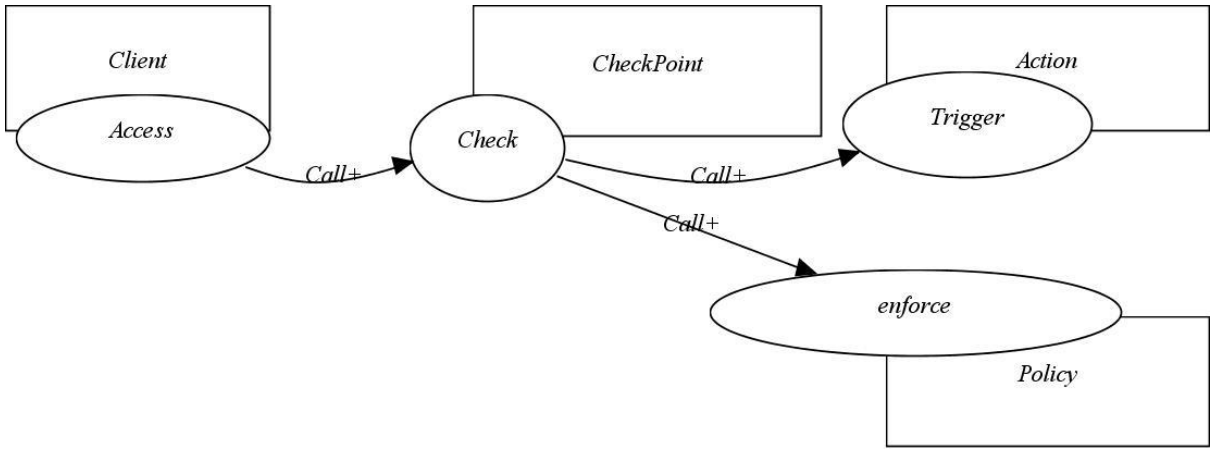
- Check Point
  - implements a method to check messages according to the current security policy.
  - if msg violates policy, triggers countermeasures
  - otherwise, allows msg to proceed to the intended recipient
- Countermeasure
  - provides actions that can be triggered in order to react to an access violation
- Security Policy
  - implements the rules that determine weather a request is granted



Wasserman & Cheng (2003)

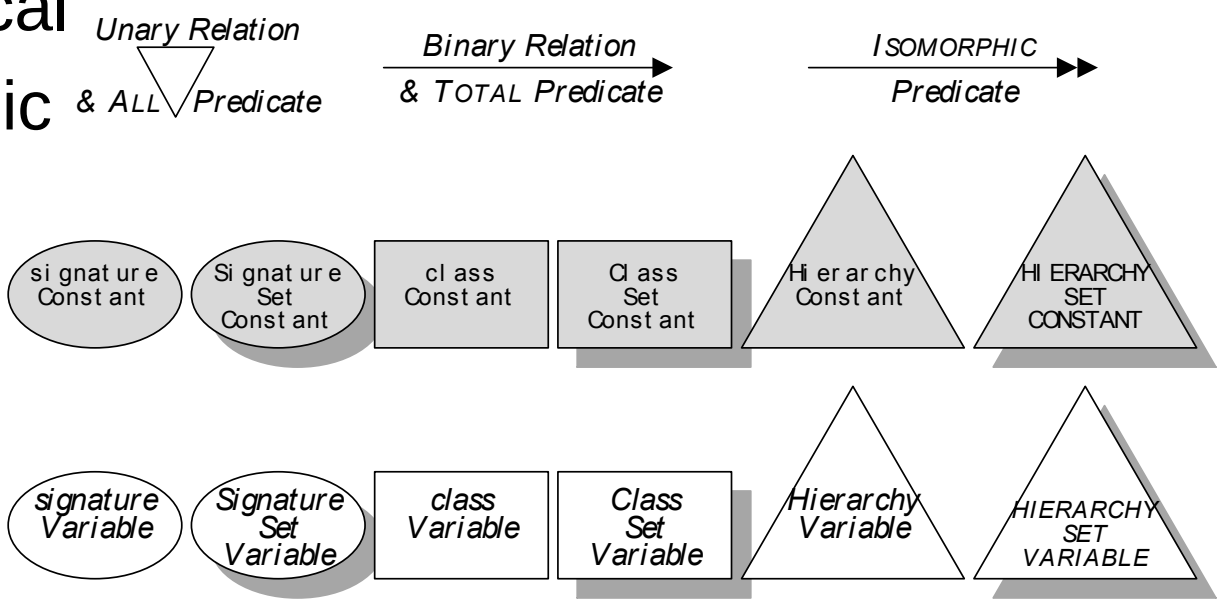
Solution (structure)

- Formalise structure using a Codechart
- Use the TTP Toolkit to reverse engineer source code and to verify conformance



Codecharts

- Automatically verifiable
- Modelling & visualization
- Parsimonious
- Formal & practical
- Visual & symbolic
- Object-oriented
- Scalable



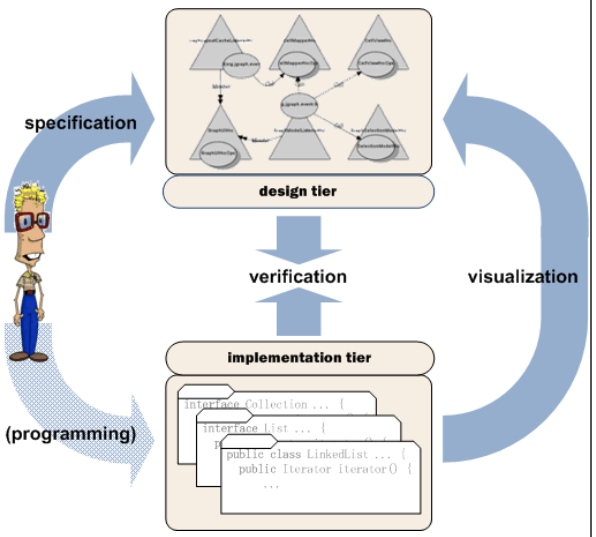
LePUS3 Vocabulary (Eden & Nicholson 2011)

Scope

- Static structure only
- Why ??
  - Behaviour are not decidable
- Examples:
    - *CheckPoint* checks if msg conforms to the policy.
      - If no, triggers a countermeasure
      - If yes, allows msg to proceed to the intended recipient
    - *Countermeasure* reacts to an access violation when triggered
    - *Client* receives granted/denied access message
    - …(Wasserman & Cheng 2003)

TTP Toolkit

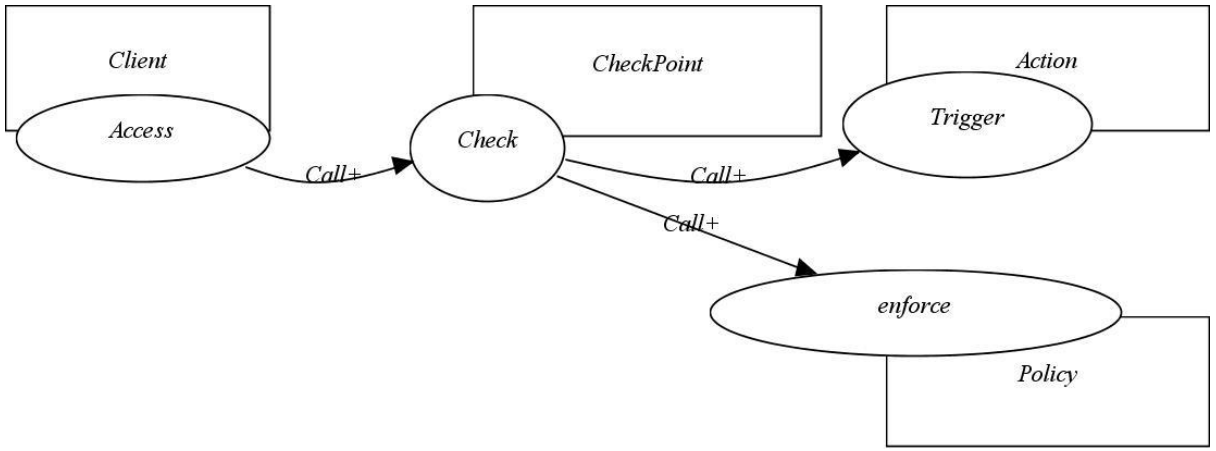
- Two-Tier Programming Toolkit.
- Prototype round-trip software engineering tools of object-oriented (Java, C++, C#, ...).
- Supports:
  - Software modelling and specifications
  - Design verification
    - Fully-automated conformance checking
    - Java 1.4 programs
  - Visualization and design recovery
    - Reverse-engineering charts from Java 1.4 program



Nicholson, Jon, and Eden (2009)

Toolkit: modelling patterns

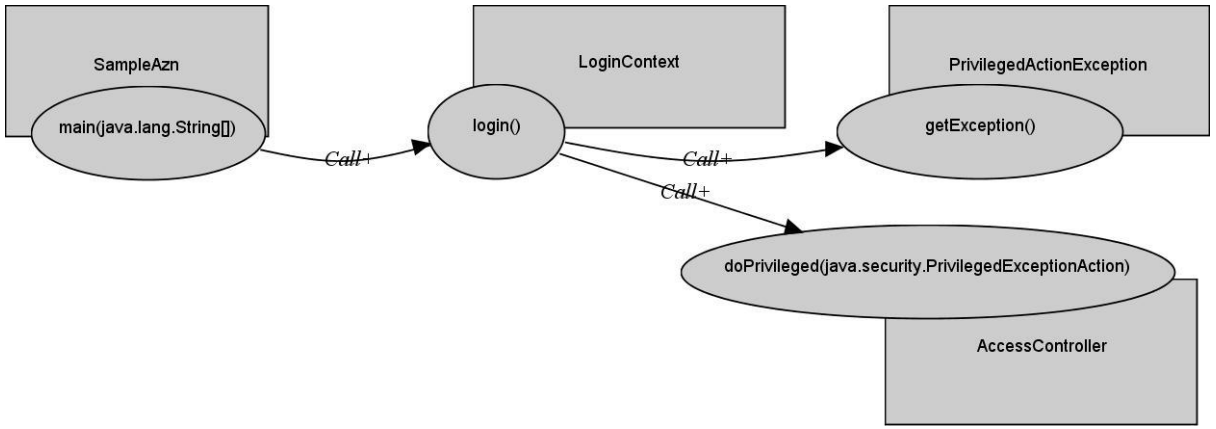
- Codechart formalising the structure of the Check Point pattern:



how modelling works.mp4

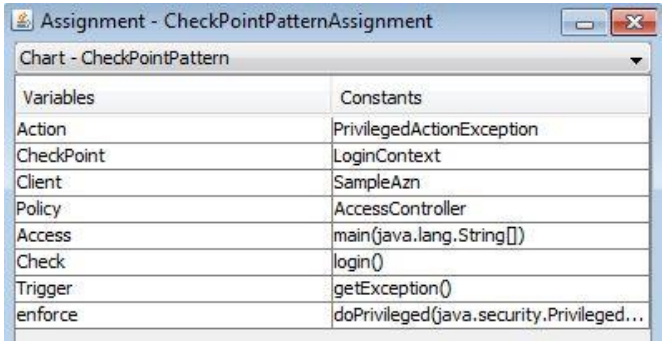
Toolkit: modelling programs

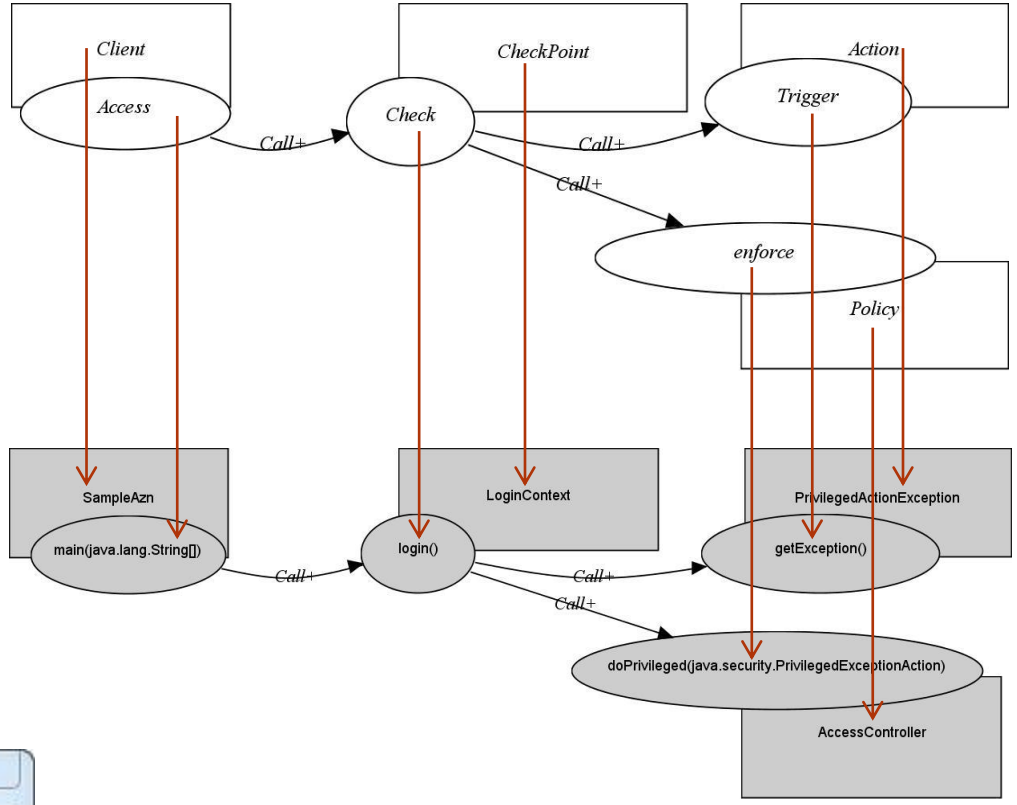
- Codechart visualizing elements of JAAS intended to implement the pattern:



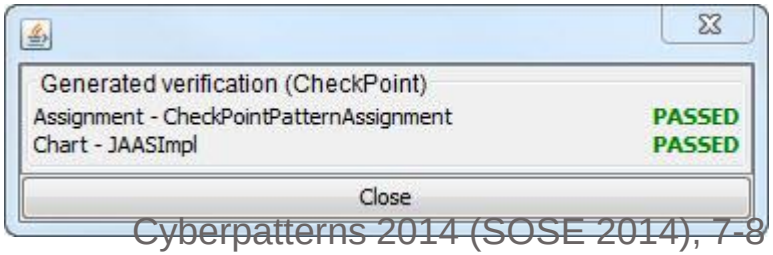
Toolkit: checking conformance

Assignment





Result



What is all this good for?

- Maintaining conformance in evolution
  - Implementations change, frequently
  - Conformance can easily be violated
  - The Toolkit can be there to remind us the motivation for certain design decisions
- What else?
  - Checking claims ("*JAAS implements the checkpoint pattern*)
  - Consistency
  - Understanding

Applications

Inconsistency between catalogues

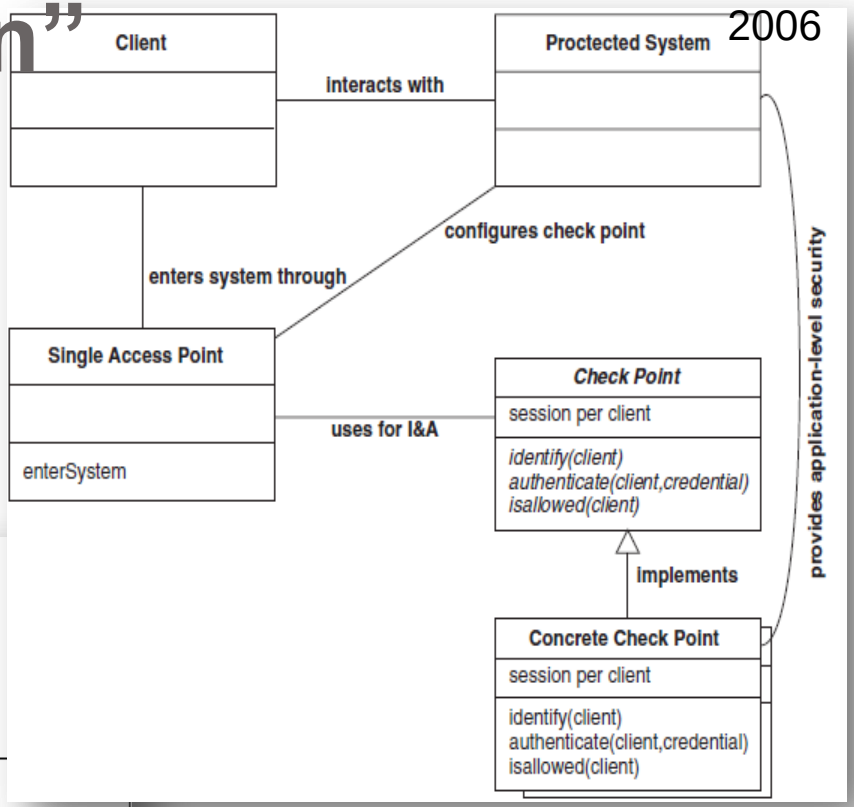
Cyberpatterns 2014 (SOSE 2014), 7-8 April 2014, Oxford, The United Kingdom

“Check Point pattern”

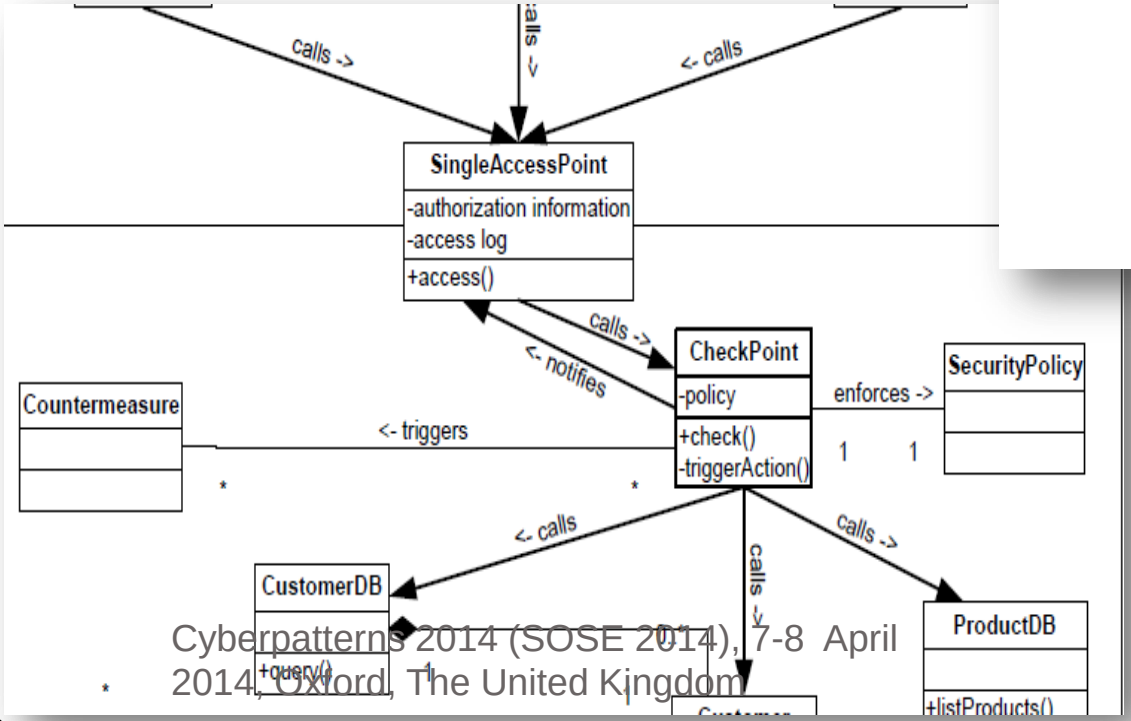
According to:
Schumacher et al. 2006

- What’s in a name?

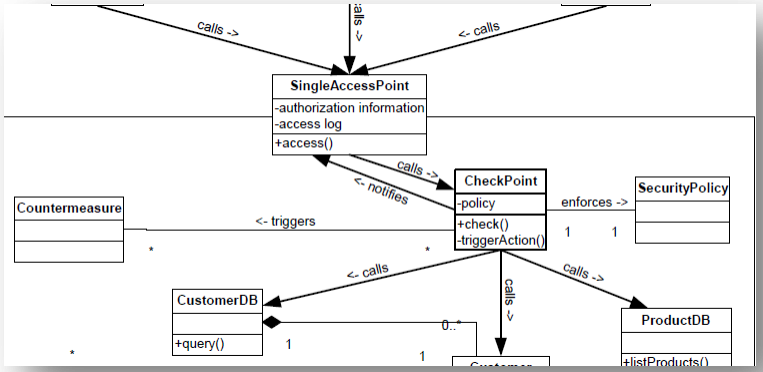
Same patterns? Not likely…



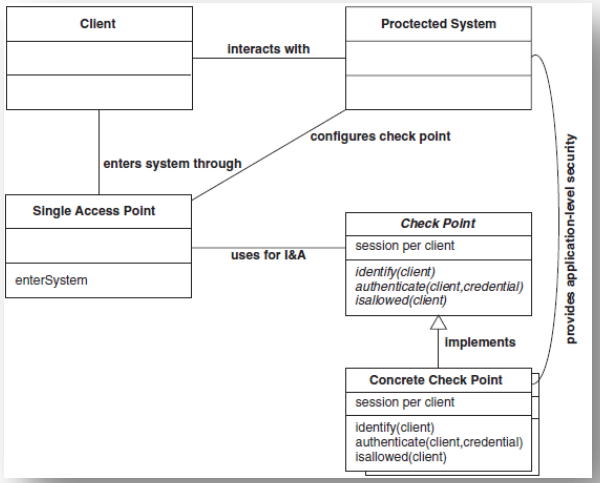
According to
Wasserman & Cheng 2003



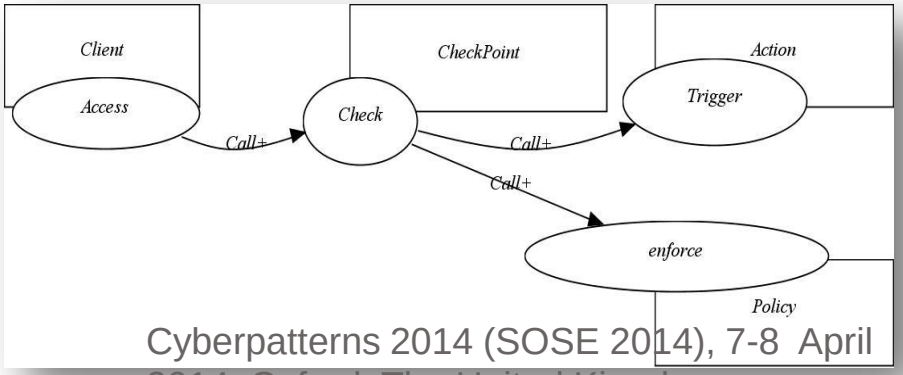
Let's check:

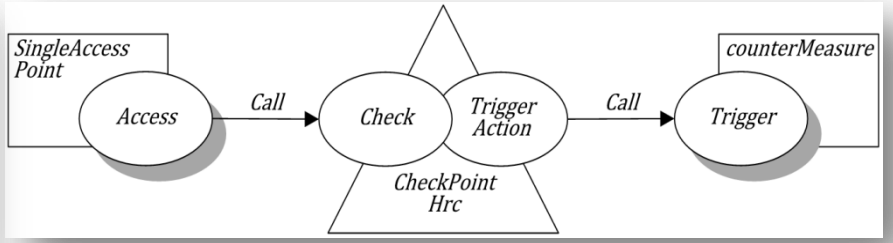


Wasserman & Cheng 2003



Schumacher et al. 2006



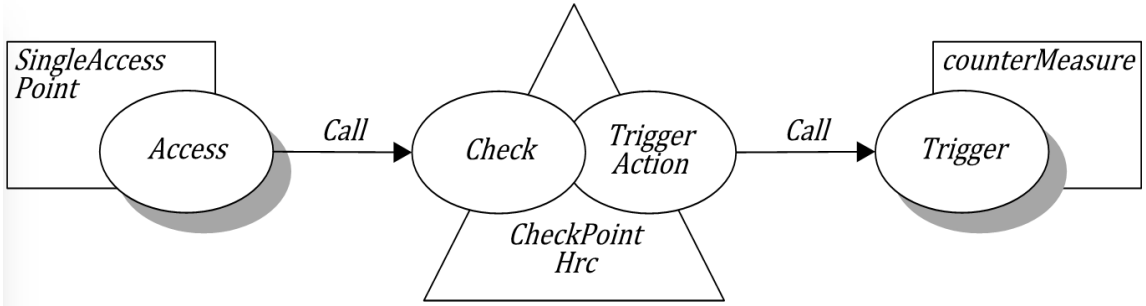


Applications

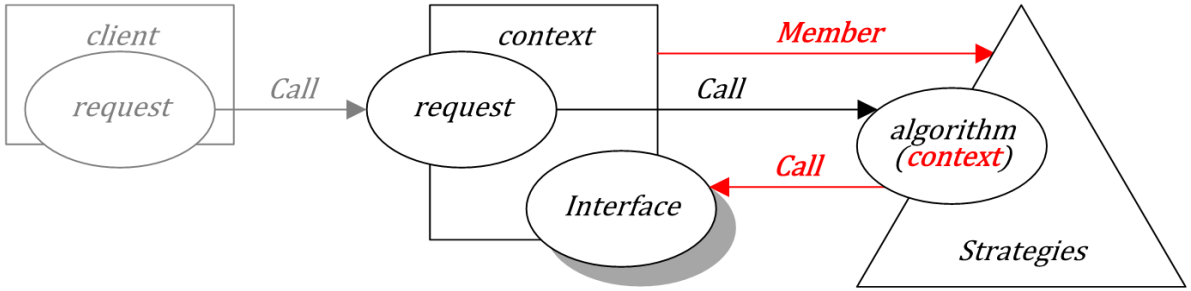
Relation between patterns

(Claims validation)

- Check Point and Strategy pattern
  - Schumacher et al. (2006)'s description of the Check Point starts with: "Apply the Strategy design pattern [GoF95] to vary the checking behavior".
  - The claim is repeated and explained e.g., stating that "Check Point interface corresponds to the abstract strategy in Strategy [GoF95]." [7, p. 296]



Codechart modelling the Check Point pattern according to Schumacher et al (2006)



Codechart modelling Strategy pattern (Eden & Nicholson 2011). In red: elements missing from the Check Point pattern

- Not the same
- Need a proof ?
  - Visually (easy way)
  - Formally (hard way)
    - No worries, TTP will automatically do it for you.

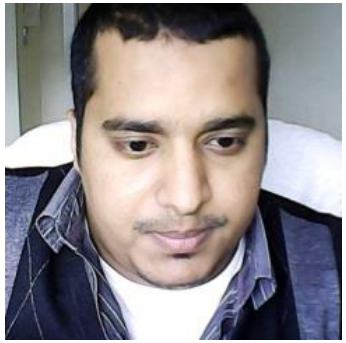
Conclusion

- Address pattern variants
- Verify design conformance
- Validate patterns relations
- Assure same understanding of pattern among (implementer-maintainer-software architect)

Future work

- Automatically detect the pattern instances
- Automatically generate assignments
- Structure a catalogue which contains a set of precise and practical security patterns

Authors



Abdullah A. H. Alzahrani
aahalz@essex.ac.uk
University of Essex

Amnon Eden
eden@essex.ac.uk
University of Essex

Majd Zohri Yafi
mzohri@essex.ac.uk
University of Essex

Cyberpatterns 2014 (SOSE 2014), 7-8 April 2014, Oxford, The United Kingdom

References

1. Eden, Amnon H., and J. Nicholson. 2011. *Codecharts: Roadmaps and Blueprints for Object-Oriented Programs*. John Wiley & Sons.
2. Wassermann, R., and B. H. C. Cheng. 2003. 'Security Patterns'. In *Michigan State University, PLoP Conf*.
3. Schumacher, M., E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. 2006. *Security Patterns. Integrating Security and Systems Engineering*.
4. Alur, Deepak, John Crupi, and Dan Malks. 2003. *Core J2EE Patterns: Best Practices and Design Strategies*. 2nd ed. Prentice Hall Professional.
5. Nicholson, Jon, and Amnon Eden. 2009. 'TTP Toolkit - Home - Object-Oriented Design, Visual Modelling, Formal Specification, Automated Verification, Reverse Engineering, Design Mining, Traceability, Scalability'. May. http://ttp.essex.ac.uk/.
6. Oracle Java Technology. 2011. 'JAAS Reference Guide'. http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html.


ANY
QUESTIONS
?

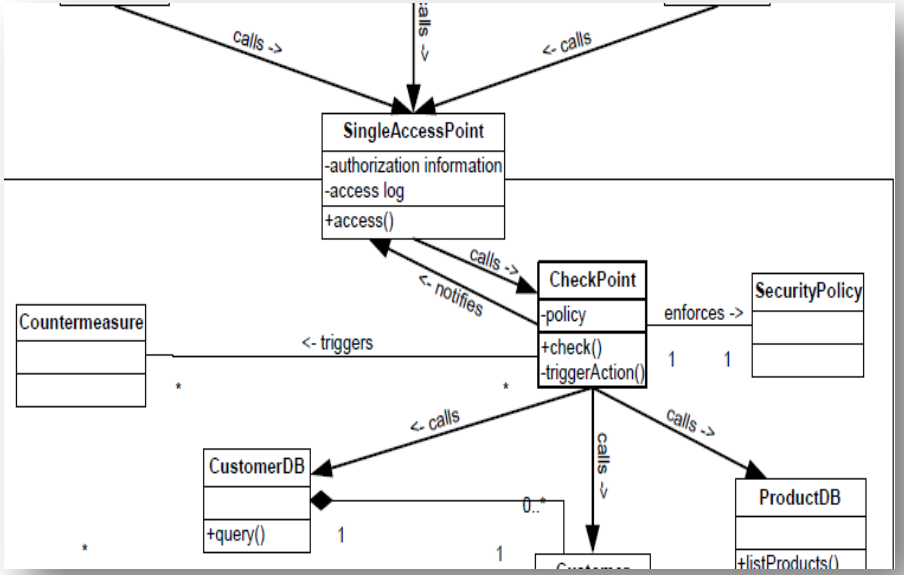
Z spec auto-generated by TTP toolkit

CheckPointPattern

$$Action\ ,\ CheckPoint\ ,\ Client\ ,\ Policy : \mathbb{CLASS}$$
$$Access\ ,\ Check\ ,\ Trigger\ ,\ enforce : \mathbb{SIGNATURE}$$

$$Call^{+}\ (Access \otimes Client\ ,\ Check \otimes CheckPoint)$$
$$Call^{+}\ (Check \otimes CheckPoint\ ,\ Trigger \otimes Action)$$
$$Call^{+}\ (Check \otimes CheckPoint\ ,\ enforce \otimes Policy)$$

Leading example: Check point pattern



Check Point (Wasserman & Cheng 2003)

JAAS

- *Java Authentication and Authorization Service*
- Used in Apache Web server
  - validate each HTTP request according to a configured activation sequence
- Java implementation of the Pluggable Authentication Module (PAM) information security framework
  - originally developed for Solaris operating system
  - Other implementations: PAMLinux
- **Implements the Check Point pattern**