Introduction to
Security Patterns

Prof. Peter Sommerlad
HSR - Hochschule für Technik Rapperswil
Institute for Software
Oberseestrasse 10, CH-8640 Rapperswil

download from: http://wiki.hsr.ch/PeterSommerlad/

Contact

- **Peter Sommerlad**
  - peter.sommerlad@hsr.ch
  - I am a software engineer by heart and soul. As co-author of "Pattern-oriented Software Architecture - A System of Patterns" I won the Software Productivity Award 1996. In addition to developing software I write patterns and shepherd other pattern authors, thus helping them improving their publications. I am co-authoring a book on Security Patterns due 2005.
  - since 09/2004 professor for informatics at HSR

Agenda

- General introduction to patterns
- **Introduction to Security Patterns**
- **Enterprise Security Patterns (one example)**
- **Operational Security Patterns (brief)**
- **Security Patterns describing Architectures**
- **Patterns' relationships**
- **Wrap up**

What are Patterns?

- **Descriptions of successful engineering stories**
- **Address recurring problems**
- **Describe generic solutions that worked**
  - Nothing new
- **Tell about the forces of the problem**
  - What makes the problem addressed hard?
- **Tell about the engineering trade-offs to take**
  - Benefits and Liabilities
- **Give us names to talk with and about**
- **A thing (solution) and how to get it (process for implementation)**

Example: Buy the first round

Kevlin Henney, *kevlin@curbralan.com*,
April 2001, revised July 2001

Context:

Meeting colleagues for a few drinks in a bar or pub that expects payment for drinks.

Problem:

How do you maximise your drinking, whilst both minimising expenditure and maintaining good will with your drinking colleagues?

Buy the First Round (2)

Forces:

- You have limited money, or at least limited desire to spend it.
- The drinks are not free.
- You want to drink (possibly lots).
- Your colleagues will not all turn up on time, but there will be quite a few of them eventually.
- You do not want your colleagues to think (or perhaps notice) that you are being tight git.

Solution:

Buy the first round of drinks before all your colleagues have arrived.

Buy the First Round (3)

Consequences:

- You need to get to the bar early, preferably first and with a couple of colleagues in tow.
- \+ Volunteering to buy the first drink shows good nature and enthusiasm, and spreads good will.
- \+ You get free drinks for the rest of the evening, assuming a reasonable increase in the number of colleagues and a steady rotation of round buying.
- \+ If there is enough drinking, others will not notice your use of this pattern or recall any other known uses. This supports repeatability in future as only your generous nature will be remembered.

What are no patterns?

- **General Principles, e.g.,**
  - KIS – Keep it simple
  - DRY – Don't repeat yourself
    - (aka OAOO: once and only once)
  - Need to Know (confidentiality)
- **Things that worked only once or twice or never:**
  - "I did it this way"
  - "I invented this pattern"
- **Problems without solutions**
- **Description of Solutions without problems**
- **Algorithms, Processes, cooking recipes**

What patterns are not?

- A panacea
- A silver bullet
- A novices' tool
- A ready made component
- A means to turn off your brain

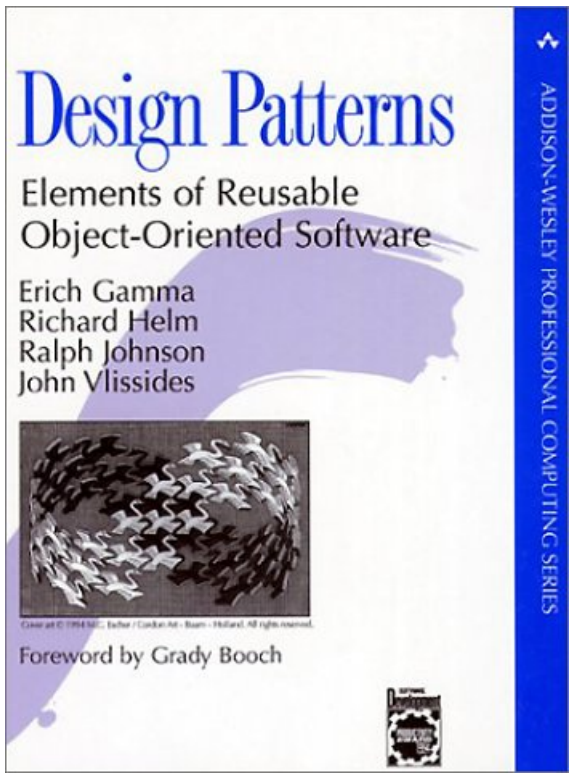
But

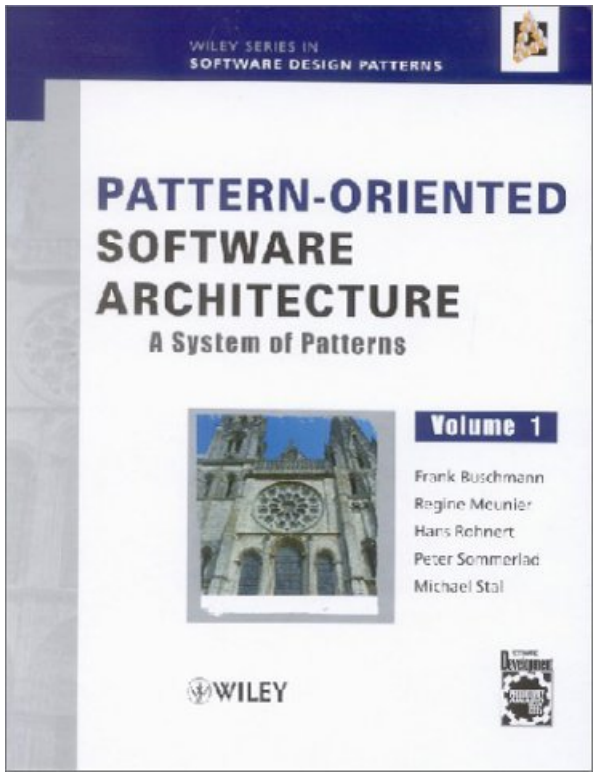
- Patterns proved to be useful engineering tools, especially for discussing and considering alternatives and different trade-offs.

What kinds of Patterns exist?

- **Architecture Patterns (Christopher Alexander)**
- **Software Patterns:**
  - Design Patterns (Gamma, Johnson, Helm, Vlissides)
  - Pattern-oriented Software Architecture (Buschmann,Meunier,Rohnert,Sommerlad,Stal)
  - Many more ...
- **Organizational Patterns**
- **Learning and Teaching Patterns**
- **Documentation Patterns**
- **Process Patterns**

Software Patterns

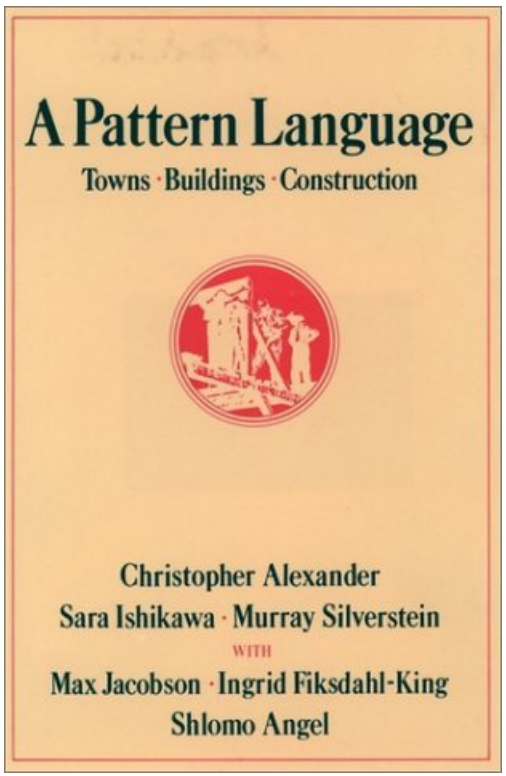


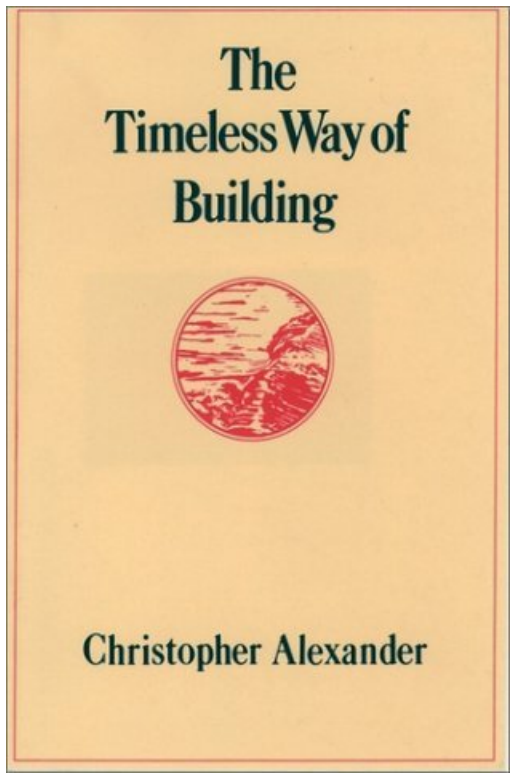


- **And many many more**

Example from Architecture

- **Christopher Alexander (and many colleagues)**





150 A Place to Wait
(Alexander et al: A Pattern Language)

The process of waiting has inherent conflicts in it.

On the one hand, whatever people are waiting for - the doctor, an airplane, a business appointment has built-in uncertainties, which make it inevitable that they must spend a long time hanging around, waiting, doing nothing.

On the other hand, they cannot usually afford to enjoy this time. Because it is unpredictable, they must hang at the very door. Since they never know exactly when their turn will come, they cannot even take a stroll or sit outside...

150 A Place to Wait (2)

(Alexander et al: A Pattern Language)

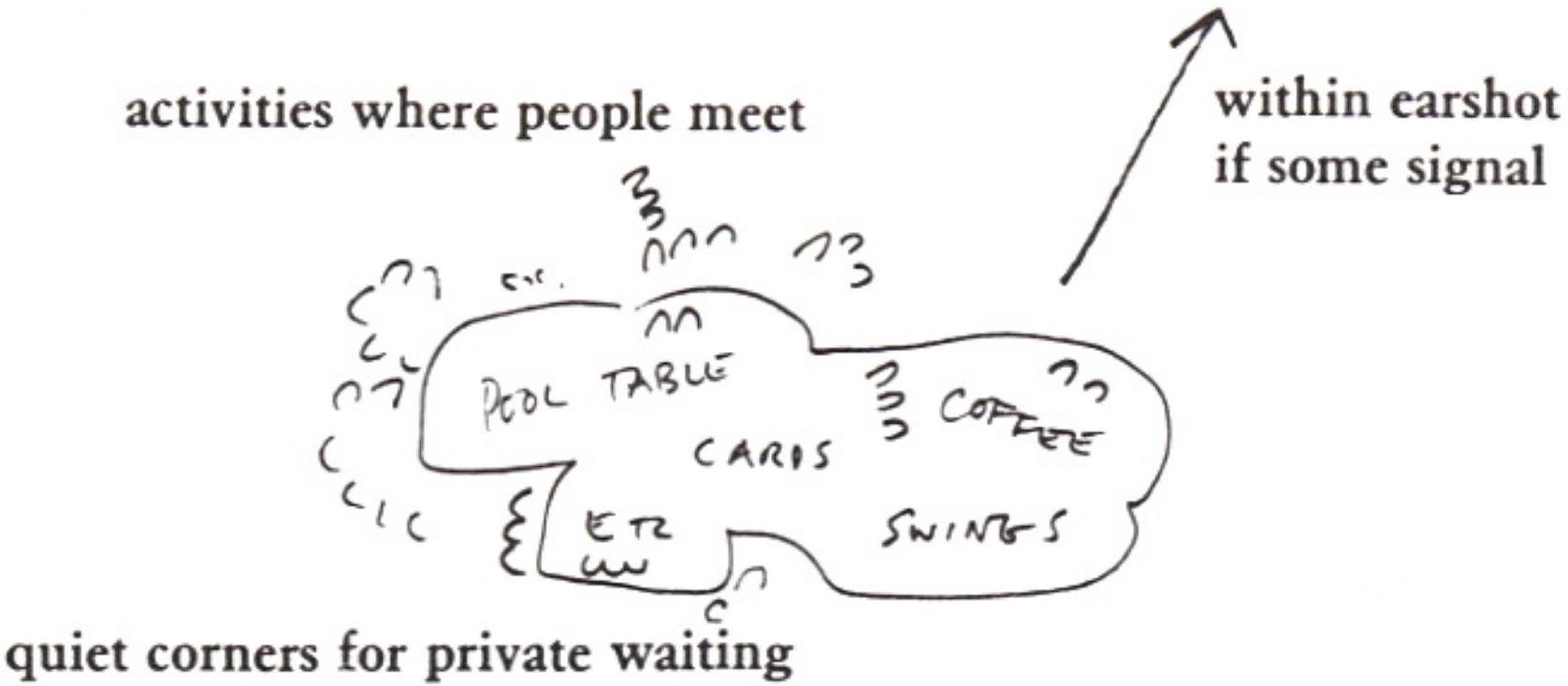
Therefore:

In places where people end up waiting, create a situation which makes the waiting positive. Fuse the waiting with some other activity - newspaper, coffee, pool tables, horseshoes; something which draws people in who are not simply waiting.

And also the opposite:
make a place which can draw a person waiting into a reverie; quiet; a positive silence.

150 A Place to Wait (3)

(Alexander et al: A Pattern Language)



150 A Place to Wait (4)

(Alexander et al: A Pattern Language)

The active part might be a window on the street – STREET WINDOWS (164), a café – STREET CAFE (88), games, positive engagements with the people passing by – OPENING TO THE STREET (165). The quiet part might have quiet garden seat – GARDEN SEAT (176), a place for people to doze – SLEEPING IN PUBLIC (94), perhaps a pond with fish in it – STILL WATER (71). To the extent that this waiting space is a room, a group of rooms, it gets its detailed shape from LIGHT ON TWO SIDES OF EVERY ROOM (159) and THE SHAPE OF INDOOR SPACE (191)…. **(Alexander et al., 1977: pp. 707 711)**

What have we seen in the pattern?

- **A Problem**
  - Waiting
- **Tension: Forces that make a problem hard**
  - Different attitudes to a waiting situation
- **Resolution of forces: Solution**
  - with alternatives

- **Relationship to other Patterns**

Agenda

- **General introduction to patterns**
- Introduction to Security Patterns
- **Enterprise Security Patterns (one example)**
- **Operational Security Patterns (brief)**
- **Security Patterns describing Architectures**
- **Patterns' relationships**
- **The pattern community**
- **Wrap up**

What is in a Security Pattern?

Security Pattern Template:

- **Name**
- **Intent - Summary**
- **Context**
- **Problem with Forces**
- **Solution (with Structure and Dynamics)**
- **Implementation**
- **Known Uses, Variants and See Also**
- **Benefits**
- **Liabilities**

Example: Single Access Point (1)

- First a story:

Single Access Point – Story ctd. (2)

- **Prosperity leads to security problems and solutions:**

Single Access Point (3)

Context:

- **You need to provide access to a system for external clients. The system should not be misused or damaged.**

Problem:

- Checking external interaction is required, but can be cumbersome. If too many interaction points exist, that are to be checked, validation of security is hard. Checking too often can be annoying for valid users.

Single Access Point – Forces (4)

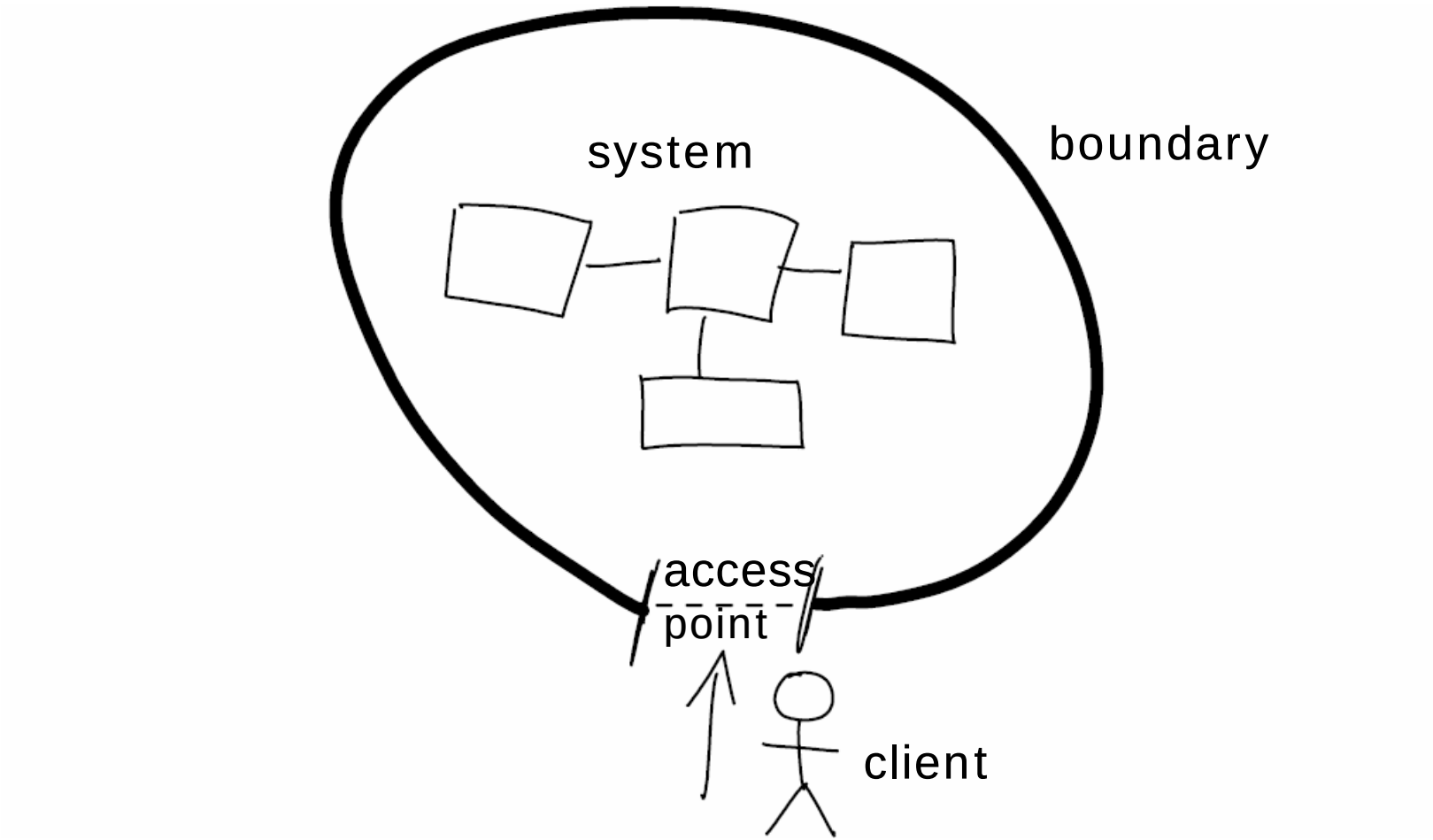
Forces

- You must provide access to the system
- No system is an island, it is interconnected
- Most systems have an inner structure of subsystems also needing protection
- Having many entry points reduces security
- Multiple entry points can result in duplicated code and checks
- Checking clients for allowance over and over is annoying and slows the system
- Uniform access can lower its usability, but is easier to control

Single Access Point – Solution (5)

- Provide a prominent Single Access Point to your system for external clients (users). At this access point check the legitimacy of the users according to your defined policy (see Checkpoint pattern)
- Associate a “Session” with a legitimate user to allow further access without additional annoyance.
- Protect the rest of the systems boundary, so that the single access point cannot be circumvented.

Single Access Point



Single Access Point (6)

Examples on different levels of abstraction:

- **Operating system login window**
- **Medieval city with guarded gate and a wall**
- **Function entry point with precondition check**
- **FrontDoor reverse proxy**
- **Security guard in an office building**

Single Access Point Consequences (7)

Benefits:

- **Clearly defined entrance for users**
- **One place to check for vulnerabilities**
- **Inner structure of system can be simpler**
- **No redundant checks, trust to single access point**

Liabilities:

- **Just one access point can be too cumbersome**
- **Trust in “gatekeeper” and “city wall” is required**
- **Checks might be too rigorous for a given situation**
- **Complex systems need more access points**

What have we learned?

- **Security Patterns can be very abstract**
  - Nevertheless relate to many concrete situations
- **A slide presentation is too limited to give all the subtleties of a pattern**
  - We need text book style for everything
- **Patterns can apply or have analogies beyond the technical domain**
  - This carries the danger of not seeing an analogy as one
- **Even “obvious” patterns can have non-obvious consequences**

Sources for Security Patterns

- **Well-known Practice**
- **Security Experts (and their publications)**
- **Standards**
  - ISO 17799/BS 7799
  - Common Criteria
- **BSI Grundschutzhandbuch (IT Baseline Protection Manual)**
- **Security Patterns book(s) (forthcoming)**
  - Markus Schumacher (SAP), Ed Fernandez, Aaldert Hofman (CGEY), Duane Hybertson (MITRE), Andy Longshaw, Joe Yoder, Peter Sommerlad and other contributors

Problem Categories in the SPB

- **Enterprise Security Patterns**
  - Process/Requirements Patterns for strategic concerns
  - Know what you need and want from security
  - Concious treatment of security
    - neither ignorance nor fear
- **Architectural Security Patterns**
  - Technology usage
  - Structure of networks, software and hardware
- **Operational Security Patterns**
  - Process Patterns and concepts for day to day issues
  - Human issues

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Agenda

- **General introduction to patterns**
- **Introduction to Security Patterns**
- Enterprise Security Patterns (one example)
- **Operational Security Patterns (brief)**
- **Security Patterns describing Architectures**
- **Patterns' relationships**
- **The pattern community**
- **Wrap up**

Identify Security Needs for Enterprise Assets (1)

Base pattern for all enterprise security concerns. It helps to resolve the issue wether security is really needed and, if it is, what properties of security should be applied for a particular enterprise. Security properties considered include confidentiality, integrity, availability, and accountability.

Context:

- An enterprise considers security as a significant requirement. Key business drivers and assets of the enterprise are understood.

Identify Security Needs for Enterprise Assets (2)

Problem:

- How can enterprise security needs be identified?

Forces: The enterprise needs to

- comply with laws and regulations (e.g., privacy)
- handle sensitive information (confidentiality)
- comply with its own policies
- provide sufficient protection coverage for mission critical business assets
- ensure security has minimum negative impact on business efficiency
- know, when undesired events occur
- minimize overall costs

Identify Security Needs for Enterprise Assets (3)

Solution

- Identify the relevant business assets
  - Information, data assets, physical assets
- Identify the relevant business drivers that influence security protection needs of assets
  - laws and regulations, partner relationships, goals and objectives, business processes, sensitives business events
- Determine relationship between assets and business drivers
- Identify what types of security is needed per asset
  - **Confidentiality, Integrity, Availability, Accountability**

Identify Security Needs for Enterprise Assets (4)

Benefits

- Facilitates balanced and informed decisions about security needs by making forces and business drivers explicit.
- Traceability of asset protection back to its originating business drivers. Provides rationale for evolution of security needs over time.

Liabilities

- **It is not free, requires honest work and commitment. Danger of wrong people doing too much work.**
- **Danger of mis-using the results**
- **Cost might exceed benefits**

What have we learned?

- **Patterns are not only about structure**
- **Patterns can describe (business) processes**
  - How you do things
- **good Patterns are honest**
  - Not: “do it this way”
  - But: “people have been successful this way, but first consider the consequences” (cost too high)
- **Patterns leave a lot of things to the “user”**
  - You are not allowed to turn off your brain and blame the pattern afterwards :-)

Enterprise Security & Risk Management

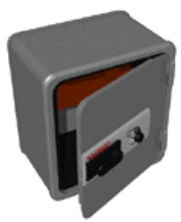
- Identify Basic Security Needs
- Identify Security Needs for Enterprise Assets

- Assessing Risks
- Asset Valuation
- Threat Assessment
- Vulnerabilty Assessment
- Risk Determination
- Moving toward Mitigation and Safeguarding
- Select Enterprise Security Approaches
- Select Enterprise Security Services

Enterprise Security Approaches

This pattern guides an enterprise in selecting security approaches, i.e., **prevention**, **detection**, and **response**. Security approaches are driven by required security properties of its assets, such as confidentiality, integrity, and availability, and by assessed security risks. Security approaches also provide a basis for deciding what security services should be established by the enterprise.

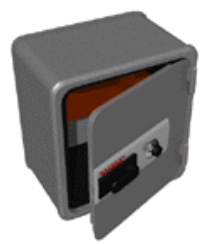
Prevention Detection Response



Enterprise Security Services

This pattern guides an enterprise in selecting security services for protecting its assets, after required security approaches (prevention, detection, response) have been identified. It helps to establish a level of strength or confidence each security service should offer based on priorities. Primary examples of such services are **identification and authentication**, **accounting / auditing**, **access control / authorization**, and **security management**.

Access Control



Auditing

Agenda

- **General introduction to patterns**
- **Introduction to Security Patterns**
- **Enterprise Security Patterns (one example)**
- Operational Security Patterns (brief)
- **Security Patterns describing Architectures**
- **Patterns' relationships**
- **The pattern community**
- **Wrap up**

Identification & Authentication (I&A)

- **Identification and Authentication Requirements**
- **Automated I&A Alternatives**
- **Password Design and Use**
- **Biometric Design Alternatives**

forthcoming in Volume 2:

- Hardware Token Design Alternatives
- Unregistered Users I&A Requirements

Security Models and Rights Management

- Example: Role-Rights Definition Pattern
- Authorization
- Role-based Access Control (RBAC)
- Multilevel Security
- Reference Monitor
- Role Rights Definition
- Task-based Rights Manager

Example: Role-Rights Definition Pattern

- **How do we assign appropriate rights to the roles when we want to implement a least privilege policy?**
  - Roles correspond to functional tasks, we need to assign enough rights to perform the work.
  - Rights should be assigned according to need-to-know, where each role gets the minimal set of rights to perform her duties.
  - New roles appear and some roles cease to be useful, thus changes to roles and rights should be easy.
  - Assignment of rights should be independent of implementation details.
- Therefore**, guide the assignment of rights based on the Use Cases for the system under consideration. Roles correspond to specific Actors in the Use Cases. Assign the access rights to roles based on the system functions associated with the corresponding use cases.**

Agenda

- **General introduction to patterns**
- **Introduction to Security Patterns**
- **Enterprise Security Patterns (one example)**
- **Operational Security Patterns (brief)**
- Security Patterns describing Architectures
- **Patterns' relationships**
- **Wrap up**

Architectural Security Patterns

- Access Control Architecture (Ch 8)
- Operating System Access Control (Ch 9)
- Audit and Accounting (Ch 10)
- Firewalls (Ch 11)
- Internet Application Security (Ch 12)

Access Control Architecture (Ch 8)

- Single Access Point
- Checkpoint
- Security Session
- Full Access with Errors
- Limited Access

Example: Checkpoint

Context

- You want to keep unwanted users from gaining access.
- Single Access Point helps to focus checks required for assuring validity of users (e.g. id and password).
- Dealing with mistakes of valid users and distinguishing these mistakes from attacks of unwanted users is hard and the corresponding policies to follow might need to change with time and place of deployment.

Problem

- How can you provide an architecture that allows to effectively protect system access while beeing able to tune I&A to evolving needs without impact to the system you protect?

Checkpoint Forces(2)

- You must authenticate and athorize clients
- Checks are a hindrance even for valid users.
- Human clients should not be punished for making mistakes, nor should making mistakes imply high costs (helpdesk!).
- “Mistakes” by “hackers” should imply some action. Different actions are required depending on the kind and severity of the failed attempt.
- Keeping the knowledge on dealing with valid vs. invalid credentials and authorization in one place makes them easier to adapt or change.
- You learn after deployment about vulnerabilities and need to adapt your system.
- Lots of error checking code throughout an application can make it difficult to debug and maintain.

Checkpoint Solution(3)

- Encaspulate the code for all checking of user credentials and acting on valid or invalid credentials in an object. Application code will access this code only by terms of the object's interface, thus making the Checkpoint object easy to change (apply the Strategy design pattern).
- Place the Checkpoint object as the guard in the Single Access Point pattern.
- Checkpoint can authenticate users and also authorize users by associating the user's session with a role.
- Linux-PAM (pluggable authentication modules) is an example of the application of Checkpoint.

Checkpoint Consequences (4)

- **Benefits**
  - Checkpoint's localizes security model can be readily analyzed and certified.
  - Checkpoint isolates the security algorithm from applications making both easier to change.
  - Checkpoint can use different implementations for different deployments (development, testing, production)
- **Liabilities**
  - Checkpoint is a critical security location where security must be enforced.
  - Checkpoint can become complex to implement and over-engineered, especially when authorization also comes into play

What can we learn from Checkpoint?

- Patterns can refine other patterns.
- Patterns often show how to provide flexibility we haven't dreamed of.
- Too much of a pattern or too many patterns tend to become over-engineering !
- Hard decisions often remain hard, but might be easier to change afterwards, such as the selection of the policy how to deal with wrong user credentials.

Session (patlet)

- How do you avoid to check a user's credentials for every operation after he has passed the Checkpoint at the Single Access Point structure?
- Associate a Session object with the user that is automatically attached with all operation requests from the user. Checkpoint ensures the correct initialization of a user's Session object. The Session object contains all security attributes with respect to the user for quick reference throughout the remaining application without the need to query the user over and over for his credentials.

What can we learn?

- Some patterns can be expressed very briefly.
- For well-known concepts this is easier.
- Some subtleties remain untold.
- Still some interesting story relating to other patterns.
- Allows to align our understanding of terminology and concepts.
- Allows to see discrepancies in concepts with a common name.

same Context and Problem – 2 patterns

- **How do you present your GUI when some users should not be allowed to perform illegal operations?**
- **Forces**
  - Users can be confused when options are not present or disabled.
  - If options pop in and out depending on a user's role users may not learn all relevant features or get confused.
  - Users should not be able to see operations not allowed.
  - Users should not see data without having permission.
  - Users do not like being told what they cannot do.

Full Access with Errors (solution brief)

- Design the application so users see and issue every possible operation. Notify a user with an error message if he tries to perform one that isn't allowed.
- Most useful if almost all operations are allowed .
- Easy to implement with simple error messages.
- Checks are done for every operation, easy to change privileges of a user.
- One view of the whole system provides consistency for documentation, training material, users.

Limited View (solution brief)

- Design your application so that users see only what they have access to. Only give them selections and menus to operations that their privileges permit.
- Best implemented using different user roles, where a role dynamically controls the GUI content.
- Can be used to provide second order incompetence: “you don't know that you don't know” required by some domains (banking).
- No application hiccups for users (WYSIWYG), no error-pop-up frustration.
- No verification on operation level needed

Patterns show Alternatives

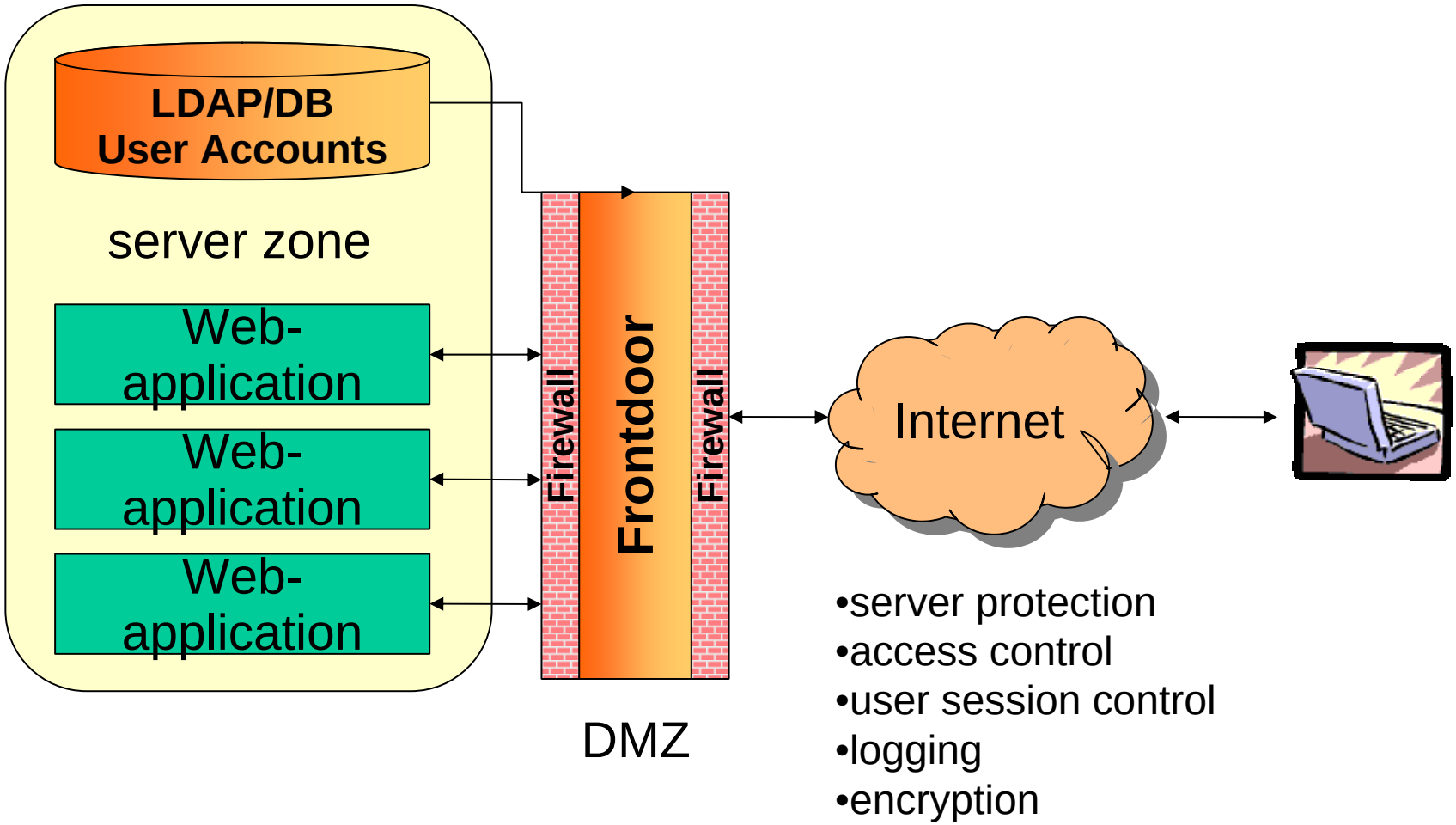
- **Full Access with Errors vs. Limited View**
- **Trade-offs can be conciously selected**
  - and are no longer arbitrary
- **Range of Design Options**
- **Change and Variation can be discussed and valuated by engineers**
- **Even user selectable variation possible.**

Internet Application Security Patterns (Ch 12)

- Information Obscurity
- Secure Channels
- Known Partners
- DMZ
- Firewall
- Protection Reverse Proxy
- Integration Reverse Proxy
- Front Door

Protection Reverse Proxy

Frontdoor


LDAP/DB
User Accounts
server zone
Web-
application
Web-
application
Web-
application
Firewall
Frontdoor
Firewall
DMZ
Internet
•server protection
•access control
•user session control
•logging
•encryption

What can we learn?

- **Some patterns can be represented by nice diagrams and a story.**
- **This can be insufficient but helps to get the essence.**
- **“Security Improvements” often increase complexity thus reducing security when you are not careful.**
- **Patterns relate to each other**
  - No pattern is an island.

Agenda

- **General introduction to patterns**
- **Introduction to Security Patterns**
- **Enterprise Security Patterns (one example)**
- **Operational Security Patterns (brief)**
- **Security Patterns describing Architectures**
- Patterns' relationships
- **Wrap up**

No Pattern is an Island

- Refinement:
  Patterns relate on others in their implementation
- Setting Context:
  Patterns are pre-requisites for others
- Variation:
  Patterns show alternative or variant solutions
- Specialization:
  Patterns are special variants of more general patterns.

Examples for Relationships

- Checkpoint and Session help to implement Single Access Point.
- Frontdoor is a more special form of Protection Reverse Proxy
- Frontdoor is also a more special form of Single Access Point
- Select an I&A Strategy should be a pre-requisite for implementing Frontdoor or Singe Access Point

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Agenda

- **General introduction to patterns**
- **Introduction to Security Patterns**
- **Enterprise Security Patterns (one example)**
- **Operational Security Patterns (brief)**
- **Security Patterns describing Architectures**
- **Patterns' relationships**
- Wrap up

What about the future?

- **Security patterns are helpful tools to be used conciously**
  - Analysing security solutions
  - Describing security solutions
  - Designing security solutions
- **Security Patterns might help overcoming the security experts shortage**
  - See OO developer mainstream today (1995 only relatively few)
  - Long term (5 + years)
- **Your collaboration is needed to gather more patterns and improve existing ones**
  - Volume 2 is waiting, see yahoo group "securitypatterns"

Your Security Patterns to be written

-
-
-
-
-
-
-
-
- **You can submit your ideas to** peter.sommerlad@hsr.ch

The Pattern Community

- **Yearly conferences on patterns**
  - EuroPLoP (Germany)
  - PLoP (USA)
  - VikingPLoP (Scandinavia)
  - KoalaPLoP, SugarLoafPLoP, ...
- **Improve skills in pattern writing**
- **Produce pattern papers and books**
- **Learning culture that will help you to improve your patterns and your attempts to apply patterns**

Wrap up

- Patterns tell stories of repeatedly successful engineering
- Honest pattern descriptions tell you the drawbacks
- Applying patterns is never mechanical
- Patterns allow more concious engineering

- **Security Patterns may become popular and produce more security awareness**
- **A lot of security experience out there waits to be written down as patterns and shared**

Have fun!

- Do not hesitate to contact me for further information and feedback:
  - **peter.sommerlad@hsr.ch**

Online-Resources:

- www.securitypatterns.org
- www.hillside.net
- http://groups.yahoo.com/group/securitypatterns/