Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

TECHNISCHE UNIVERSITÄT DARMSTADT

Software-Engineering in the industrial practice

Performance, Operations and Continuous Integration

TU Darmstadt WS 2015/2016
Oliver Hecker
Darmstadt, 05.02.2016

People matter, results count.

Oliver Hecker

Technical Architect at Capgemini

- Studies of Control Engineering at TU Darmstadt (formerly THD) – end of the 80s
- Started his professional career writing software for electronic brake systems (ABS, ...)
- “Switched” to IT in 1999
  - Started working in a COBOL / C based system for the user administration and billing of Germany’s largest mobile network
- Since 2009 at Capgemini
  - Started working with CMS at that time, but mainly in backend integration
  - Slowly expanded his knowledge to the CMS as a whole
  - Working in the Delivery Unit “Service Industries” (Telecom, Media, Public) in Offenbach
  - Several years Technical Chief Architect for the projects at our customer ZDF (www.zdf.de / www.heute.de / mediathek.zdf.de)
  - Current Role: Architect for a project at DHL

Agenda

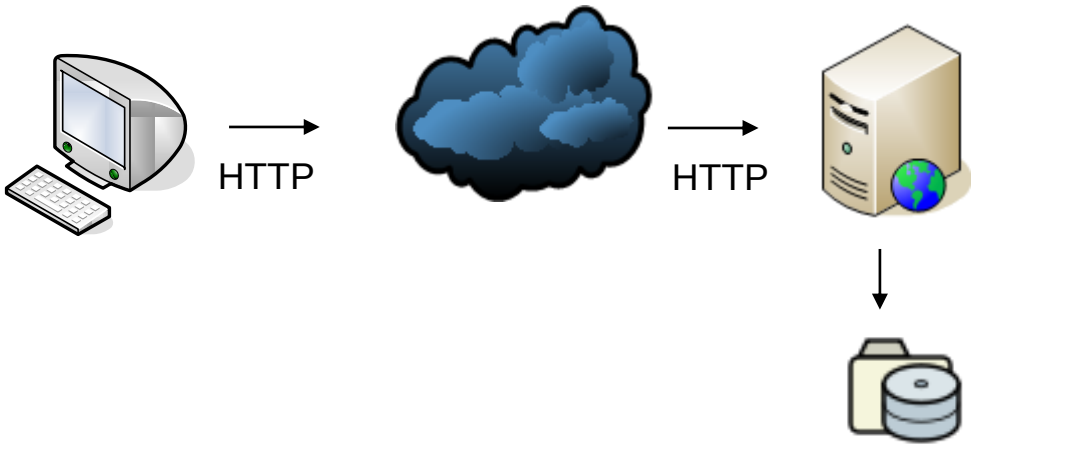
- Performance and Operations of Web Applications
- Load-Balancing and high availability
- Caching on different layers
- Software Configuration Management
- Continuous Integration

Agenda

- **Performance and Operations of Web Applications**
- Load-Balancing and high availability
- Caching on different layers
- Software Configuration Management
- Continuous Integration

The minimal web application contains browser, HTTP server with CGI scripts and file system.

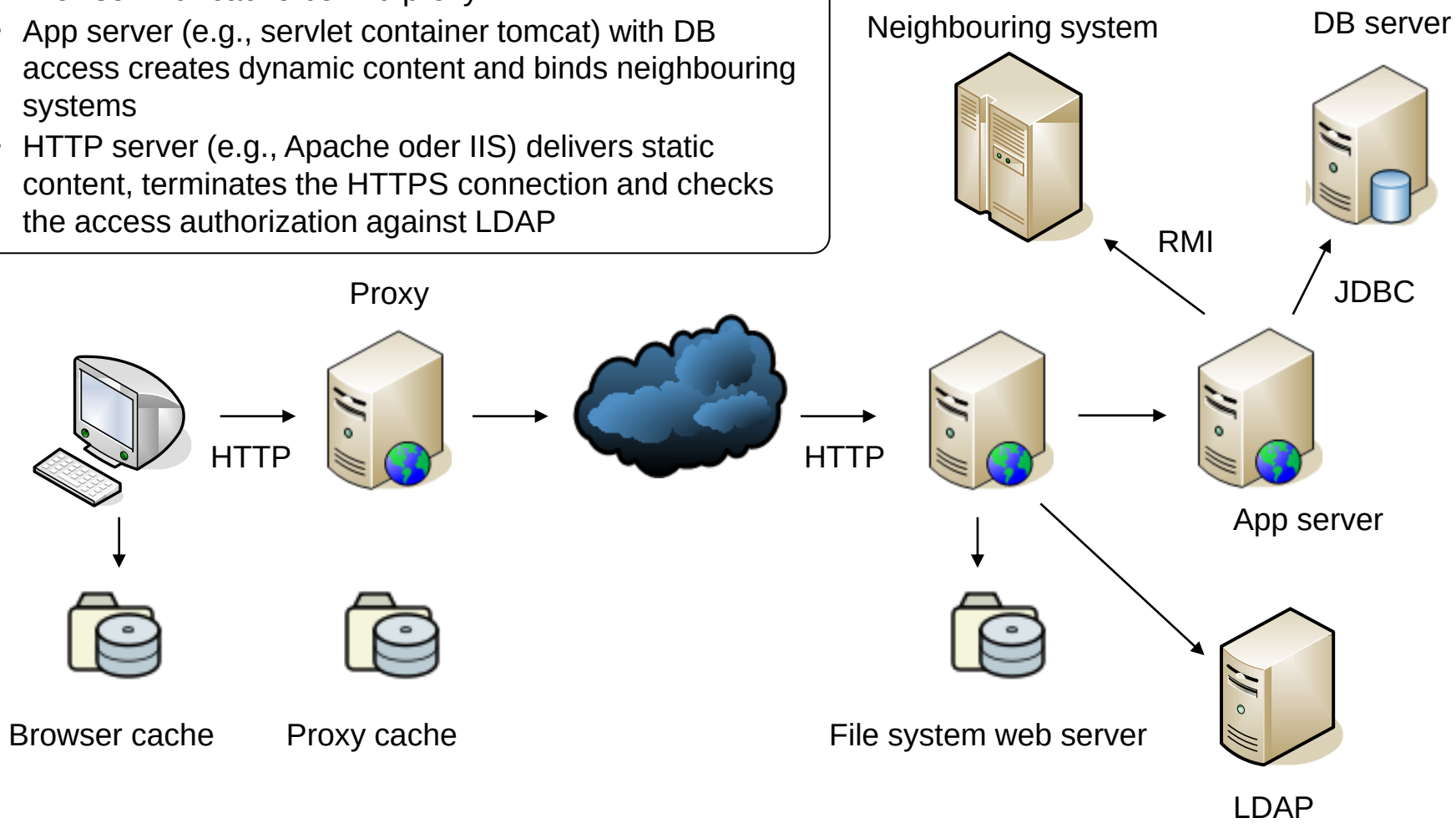
- Browser sends requests to HTTP server
- HTTP server (e.g. Apache or IIS) provides static content from the file system (GET) or calls scripts (POST, PUT, DELETE)



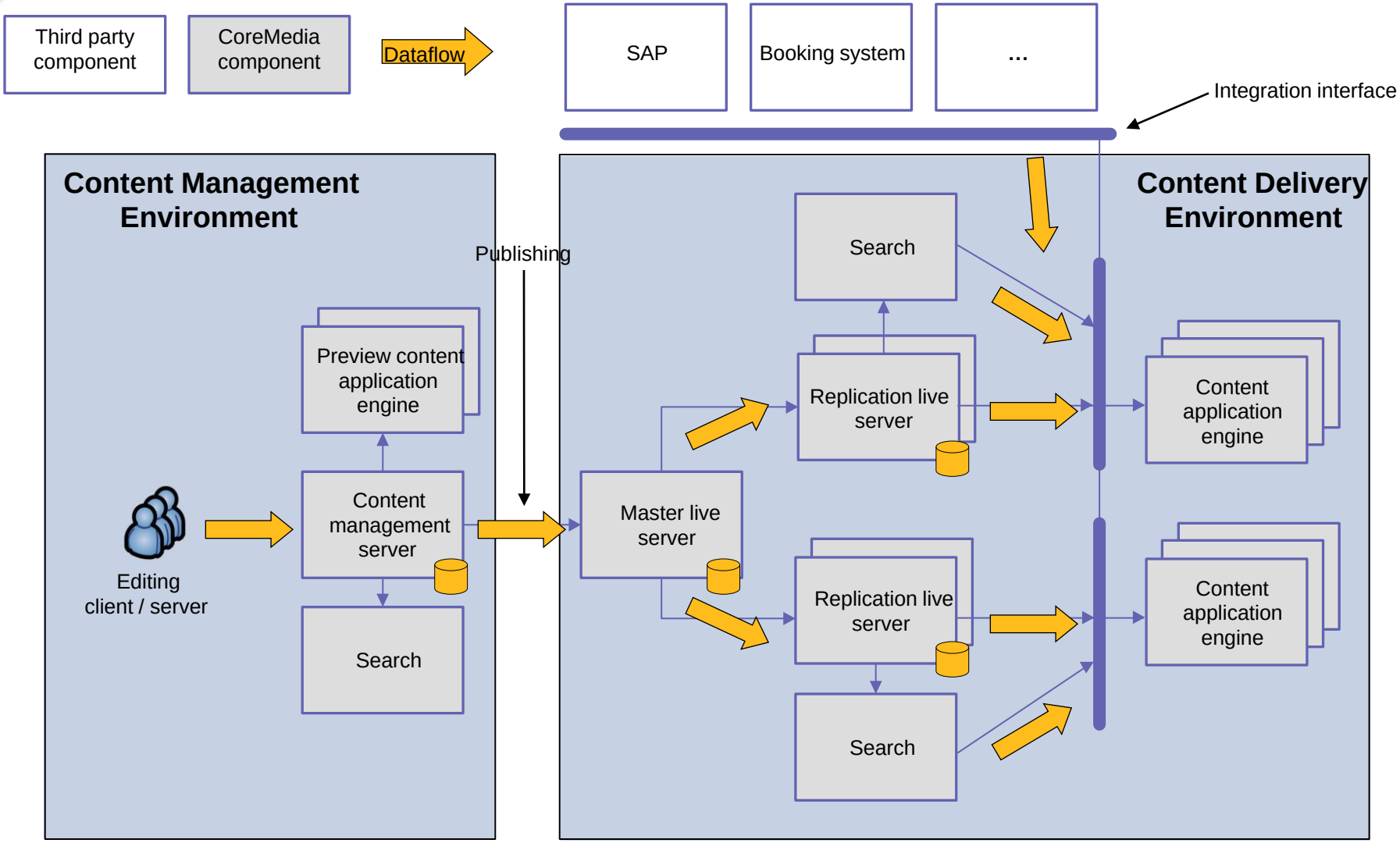
File system web server

Realistic web applications are more complex.

- Browser with cache behind proxy
- App server (e.g., servlet container tomcat) with DB access creates dynamic content and binds neighbouring systems
- HTTP server (e.g., Apache oder IIS) delivers static content, terminates the HTTPS connection and checks the access authorization against LDAP



Example: a typical CoreMedia system



(Non functional) performance of a web application is not characterized by response time only

Definition of load, response time and throughput of web applications

Criteria	Importance within the context of web applications
Load	• Microscopic perspective: number of arriving requests per time • Macroscopic perspective: number of requested pages per time
Response time	• System perspective: time for processing of a request • User perspective: time for composing a page in the web browser (or reaction to interaction) • Important: consistency, not only the average value should be considered • Considerable influence: duration of a transaction
Throughput	• Microscopic perspective: number of (successfully) processed requests per unit of time • Macroscopic perspective: number of (successfully) delivered pages per unit of time

Characteristic for web applications: many requests with short process time (several 10 to several 100 ms)

If the throughput does not correspond to the load, the performance will collapse in short time up to denial of service

Scalability of web applications is the ability to increase the maximal throughput by enhanced resource input

Theoretical approaches: vertical and horizontal scalability

Vertical scalability

- Increasing of peak throughput by increasing the system components' performance
  - Faster server
  - Faster disks / IO systems
  - Faster network connections
- Typically performance improvement, if throughput < peak throughput
- Only possible to a limited extent
- Typically disproportional and so very expensive
- Conceptually simple

Horizontal scalability

- Increasing of peak throughput by distribution of tasks around more system components
  - More server
  - More disks / IO systems
  - More network connections
- If throughput < peak throughput, typically no performance improvement
- Theoretically unlimited
- The costs are in best case proportional to increasing the peak throughput
  - Linear increase of peak throughput by used resources cannot be achieved in most cases
- Must be supported by system architecture, difficult

As the growth of load, the growth of managed data can also require scalability

Vertical scalability is disproportionately expensive

Example: larger hardware vs. more hardware

Sunfire E20k

36x Dualcore 1.8GHz processor
SPECint_rate2000 = 853

$450,000 - $2,500,000

Dell PowerEdge SC1435

Dualcore 1.8 GHz processor
SPECint_rate2000 = 44.8

Approximately $1,500, $54.000 for 36

The line between vertical and horizontal scalability is quite blurred

Vertical scalability	Horizontal scalability
• More processors, more memory, more disks • One tower with 32 CPUs and 512 GB RAM • One tower with 32 hard disks • One IP address, one power source etc. • Manufacturers integrate and test the system • High availability due to internal mechanisms and redundant components	• More processors, more memory, more disks • 32 devices each with 1 CPU, 16 GB RAM and one hard disk • 18 IP addresses, 18 power connectors • User integrates and tests the system – The test of load behaviour and failover is not easy! • High availability due to external fail-over mechanisms

High availability is the ability of web application to remain usable if some components fail

High availability requires redundancy of all components

There should be no single point of failure

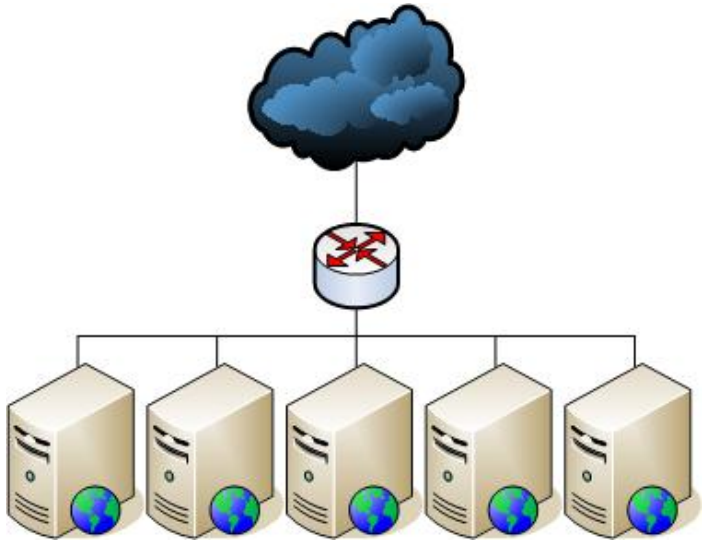
Redundancy degree

- Claim: if M out of N+M similar components fail, the system remains usable
- Consequence: system must be able to bear the load with N components
  - Overcapacity of M/N*100% required
- Typically the aim is M = 1
  - Prob(2 simultaneous failures) is approximately Prob(1 failure)^2 and thereby already very small
- For N=1 is the required overcapacity 100%, expensive
- High availability is cheaper if the load is distributed between more systems (larger N)
  - Only if the component price decreases proportionally to its size and the failure probability doesn't increase significantly at the same time

Agenda

- Performance and Operations of Web Applications
- **Load-Balancing and high availability**
- Caching on different layers
- Software Configuration Management
- Continuous Integration

Load distribution can be achieved by distribution of tasks or distribution according to responsibilities



Distribution according to responsibilities

- By user
- By IP address
- By requested data

Distribution of tasks

- Round-Robin
  - Simple but unreliable method: per DNS
- By fixed factors (two on the left, three on the right)
- By current load
- By open connections

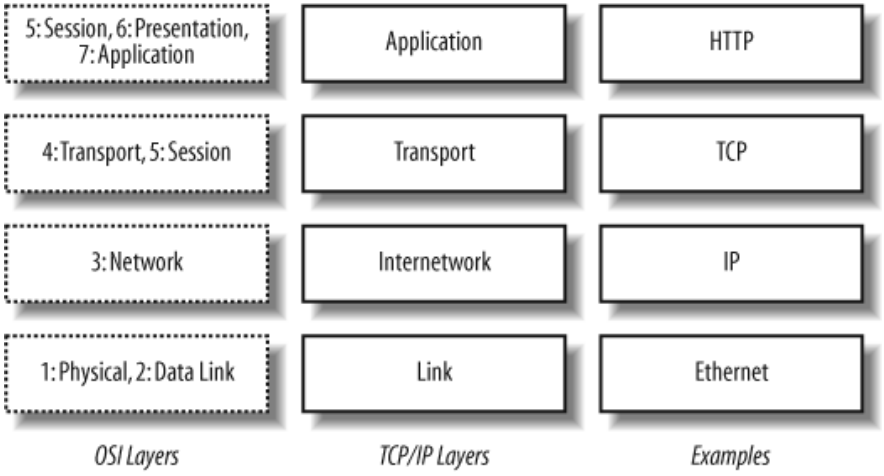
Load balancer can operate at various protocol levels

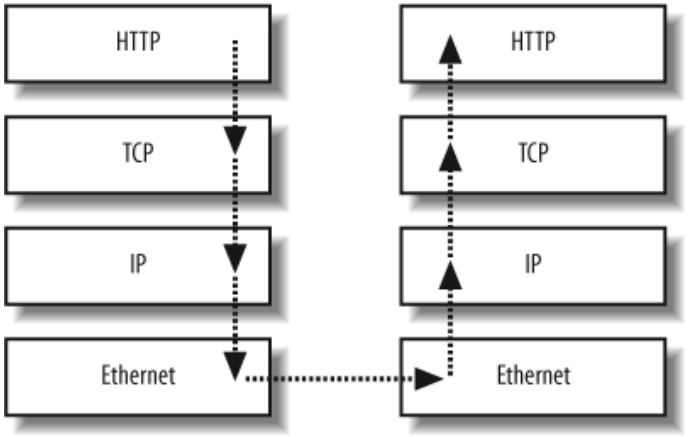
Level 4 load balancer

- Distributes TCP connections
- Not limited to HTTP
- No information about user or request
- Pooled connections may lead to stickiness
- Relatively simply to implement in hardware

Level 7 load balancer

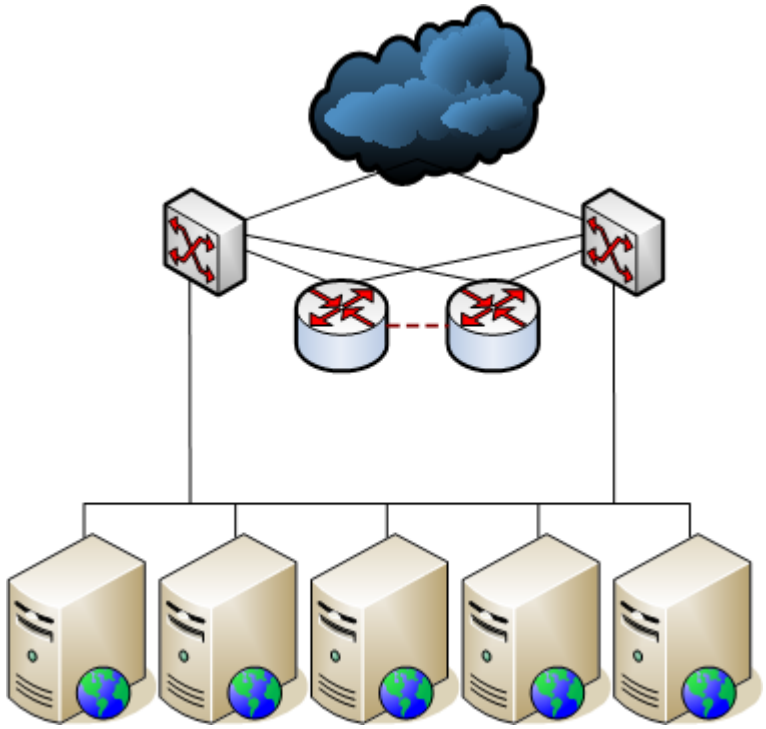
- Distributes HTTP requests
- Knows HTTP, possibly also other protocols
- Can access the requested URL, cookies, user names etc.
- Too expensive as hardware load balancer
- Throughput of hardware solutions is much better as of software load balancers





There is no high availability without additional actions, since the load balancer is a single point of failure itself

Typical configuration: a pair of hardware load balancers

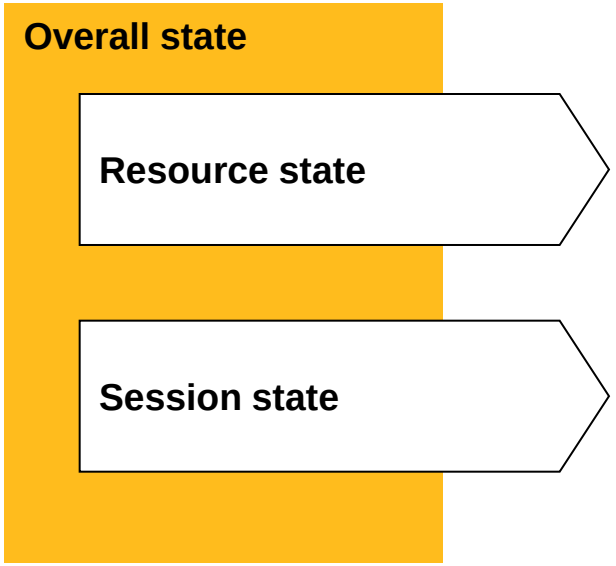


Fail-Over

- Load balancers send heart beat messages and observe the heart beat messages of the partner
- One balancer in standby modus
- If the partner fail, the standby balancer undertakes the MAC and the IP addresses of the failed balancer
- VRRP: virtual router redundancy protocol
- HSRP: hot standby router protocol

Web applications can have two kinds of state: resource state and many session states

Resource and session states



Properties

- State of the resources that are managed by the application
- Once per application
- Change slowly, often not by “normal user”
- Meaning and purpose of the application

- State of the user session(s)
- Once per each concurrent session
- Change rapidly, with each mouse click
- A means to an end

Session – is not a clear concept for web applications

What defines limits of a session?

- **Closing the browser, the window, the tab, …**
- **Not visiting the certain URL group for some time**
- **Log in / log out, …**
- **The web application defines the session**

The session state handling is a central and far-reaching architecture decision

Implementation alternatives

Former client state

- Extreme case: no session state except of visible GUI
- Invisible state by
  - hidden fields
  - JavaScript variables
- Browsers have in fact more functionality
  - authentication header
  - cookies
- Cookies and authentication header will be typically saved in browser and can last the session

Server state

- Client keeps only a session reference (session key)
  - Classic way: in a cookie
  - Alternative way: URL rewriting – session specific resources
- Often (primarily) comfortable for the application developer
  - Implications on the scalability

The server side handling must be considered in load balancing, otherwise the user loses his session in case of server switch

Alternatives for server side sessions' handling

Session stickiness

- All session requests will be directed to the same server
- Identification of session requests
  - by IP address
    - Dangerous because of NAT routers
  - by session ID from cookie or URL
    - requires layer-7-balancer
- Conceptually simple solution
  - if you have the matching LB
- Session termination in case of server loss
- Can interact with load balancing

Central sessions

- Central component that saves the session state
  - At request start – read-only access rights
  - In case of session state changes – write access rights
- Without additional actions – new SPOF

Mobile sessions

- Local sessions, session memory location is contained in a cookie
- First of all get the session data in case of server switch
- Session termination in case of server loss

Agenda

- Performance and Operations of Web Applications
- Load-Balancing and high availability
- **Caching on different layers**
- Software Configuration Management
- Continuous Integration

The goal of caching is to provide the expensive information cheaper at the next access

In a web application there is caching on different levels

Browser cache saves network transfer by last mile

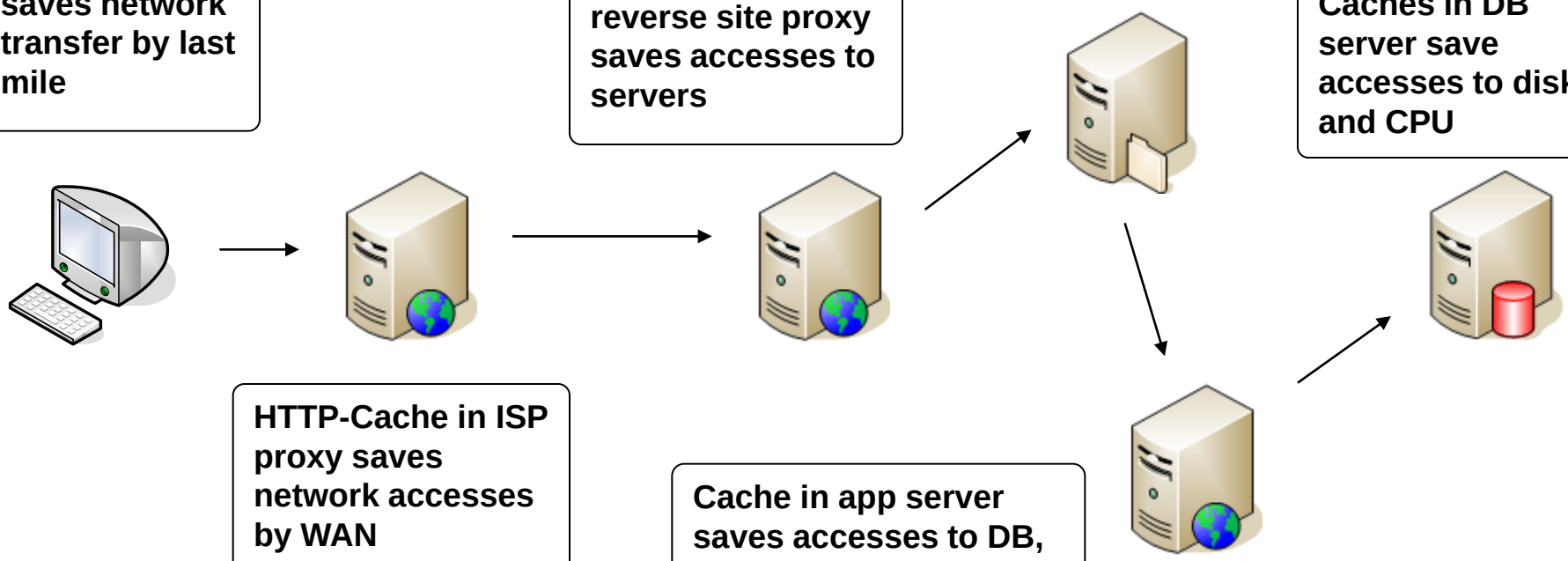
HTTP-Cache in ISP proxy saves network accesses by WAN

HTTP cache in reverse site proxy saves accesses to servers

Cache in HTTP server saves accesses to disks for static content

Cache in app server saves accesses to DB, own process logic and neighbouring systems

Caches in DB server save accesses to disks and CPU



The challenge of caching: limited and various validity of cached information

Requirements of the caching strategy in web applications

Provide no outdated information

- Time based invalidation
  - Always a compromise between cache efficiency and information up-to-dateness (time to web)
- Explicit invalidation
  - Sending of invalidation events to the cache
    - information source must recognize the cache
  - Invalidation events polling using cache
    - time-controlled or with every access (HEAD request, conditional GET)
  - URL rotation: outdated content will not be linked any more (today img_v1.jpg, tomorrow img_v2.jpg)

Handling of pages that contain components of different life span

- Cache the fragments
- Assemble the fragments
  - in server (JSP, Server Side Includes, …)
  - in network (edge side includes, see also content delivery frameworks)
  - in client (pictures, scripts, stylesheets, iFrames, AJAX)

Cache types

Cache type	Characteristics
HTTP caches	• Caching the total HTTP responses • Key: URL of the GET request • Control via request/response header • Can be simply interconnected
Fragment and object caches	• Caching of already generated page fragments … • … or objects, from which the HTML can be generated with minimal effort • Typically integrated in application server • Sophisticated invalidation mechanisms are possible • Access to fragments – Within the application server - e.g. from JSP, intern assembling – Or from the outside via HTTP - e.g. extern assembling via SSI from Apache
Not web specific caches in application server	• Like in any other application with complex process logic and/or DB accesses – General purpose caches like OSCache or EhCache – Caches in DB access layers, …

HTTP 1.1 provides detailed options to control the HTTP caches behaviour – but no option for invalidation

	Meaning
Cache control response directives	
public/private	• Response may be cached in any/private caches
no-cache	• Response must not be cached
no-store	• Response may be cached, but may not be persisted
max-age	• Life span in cache (in HTTP 1.0: header field expires), the cached response is then “stale”
must-revalidate	• Cache must validate response after its life time expires and before it will be passed on
Cache control request directives	
max-age, min-fresh, max-stale	• Requirements for “freshness” of the response
no-cache	• Response to this request must not be cached
Other header fields	
response: last-modified	• Last modification date of the resource
request: if-modified-since	• Send only in case of modifications (cache validation)

Near browser caches have only limited effect in case of high resource usage, as well as source caches

Cache types	Properties
Browser cache	• Per user • No synergies between users • Works only by repeated user access to the same resource
Proxy caches of ISPs	• For a small/big group of users • Capacity and availability are unknown
Caches of the page	• Far away from browser • Accelerate the provider's infrastructure, not the network • Reduce unnecessary bandwidth

Agenda

- Performance and Operations of Web Applications
- Load-Balancing and high availability
- Caching on different layers
- **Software Configuration Management**
- Continuous Integration

Definition of Software Configuration Management

- Software configuration management is the discipline of managing the evolution of complex software systems [IEEE STD 1987].
- It is also defined as ‘the process of controlling and monitoring change to work products’ [Herbert 1999].
- A set of management disciplines within a software engineering process to develop a baseline.
- Software Configuration Management encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after a software project
- Standards (approved by ANSI)
  - IEEE 828: Software Configuration Management Plans
  - IEEE 1042: Guide to Software Configuration Management.

Why Software Configuration Management?

- The problem:
  - Multiple people have to work on software that is changing – how to notify everyone who needs to know about a change
  - Software must run on different machines and operating systems
  - Simultaneous updates – how to prevent one person from undoing the changes of another
  - More than one version of the software has to be supported:
    - Released systems
    - Custom configured systems (different functionality)
    - System(s) under development

- Need for coordination:
  - Software Configuration Management
    - Manages evolving software system
    - Tracks the costs involved in making changes to a system

Terminology: Configuration Item

"An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process."

- Software configuration items are not only program code segments but all type of documents according to development, e.g.:
  - all type of code files
  - drivers for tests
  - analysis or design documents
  - user or developer manuals
  - system configurations (e.g. version of compiler used)
- In some systems, not only software but also hardware configuration items (CPUs, bus speed frequencies) exist!

SCM Processes*

- Build Management
  - Managing the build process and its tools
- Process Management
  - Ensuring adherence to the development process
- Change Management
  - Managing the Changes to all the CIs and its tools
- Release Management
  - process of managing software releases from development stage to software release
- Environment Management
  - Managing the components that host our system.

* Part of Project Management

NOTE

- SCM is used for both
  - Source Code Management
  - Software Configuration Management
- but they are not the same! Source Code Management is only a part of Software Configuration Management

SCM Tools

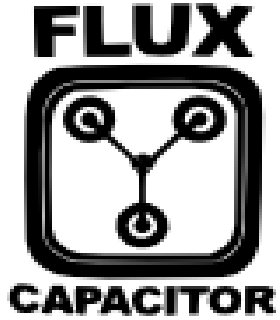
- RCS: The first on the block [Tichy 1975]
- CVS (Concurrent Version Control)
  - Based on RCS, allows concurrency without locking
  - http://www.cvshome.org/
- Subversion
  - Based on CVS
  - Open Source Project (http://subversion.tigris.org/)
- Perforce
  - Repository server, keeps track of developer's activities
  - http://www.perforce.com
- ClearCase
  - Multiple servers, process modeling, policy check mechanisms
  - http://www.rational.com/products/clearcase/
- Github
  - http://github.com/
- TFS (Team Foundation Server)
  - For .Net

Source Code Management

Software Configuration Management

Benefits of Source Code Management

Travel through time



Get yourself out of trouble

Track Changes

Source Code Management Activities

- Configuration item identification
  - Modeling of the system as a set of evolving components
- Version Control Management
  - Tracking the history of the configuration items
- Promotion Management
  - Creation of versions for developers
- Release Management
  - Creation of versions for clients and users
- Branch Management
  - Management of concurrent development
- Variant Management
  - Management of versions intended to coexist

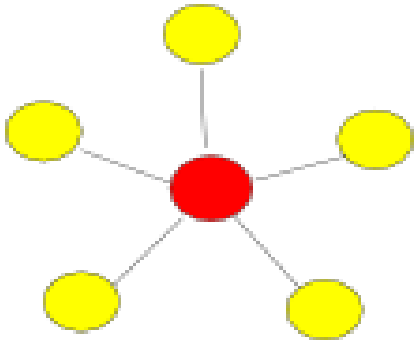
Locking

- Lock
- Modify
- Unlock

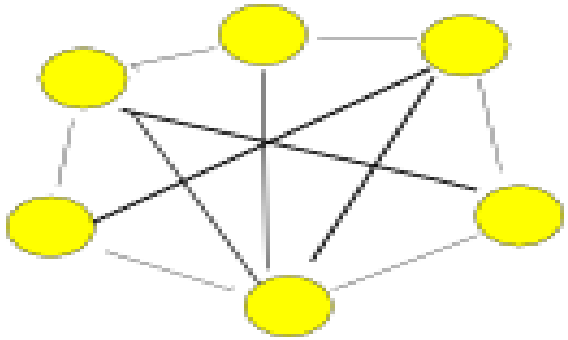
- Copy
- Modify
- Merge

Source Code Management Models

Models of Version Control

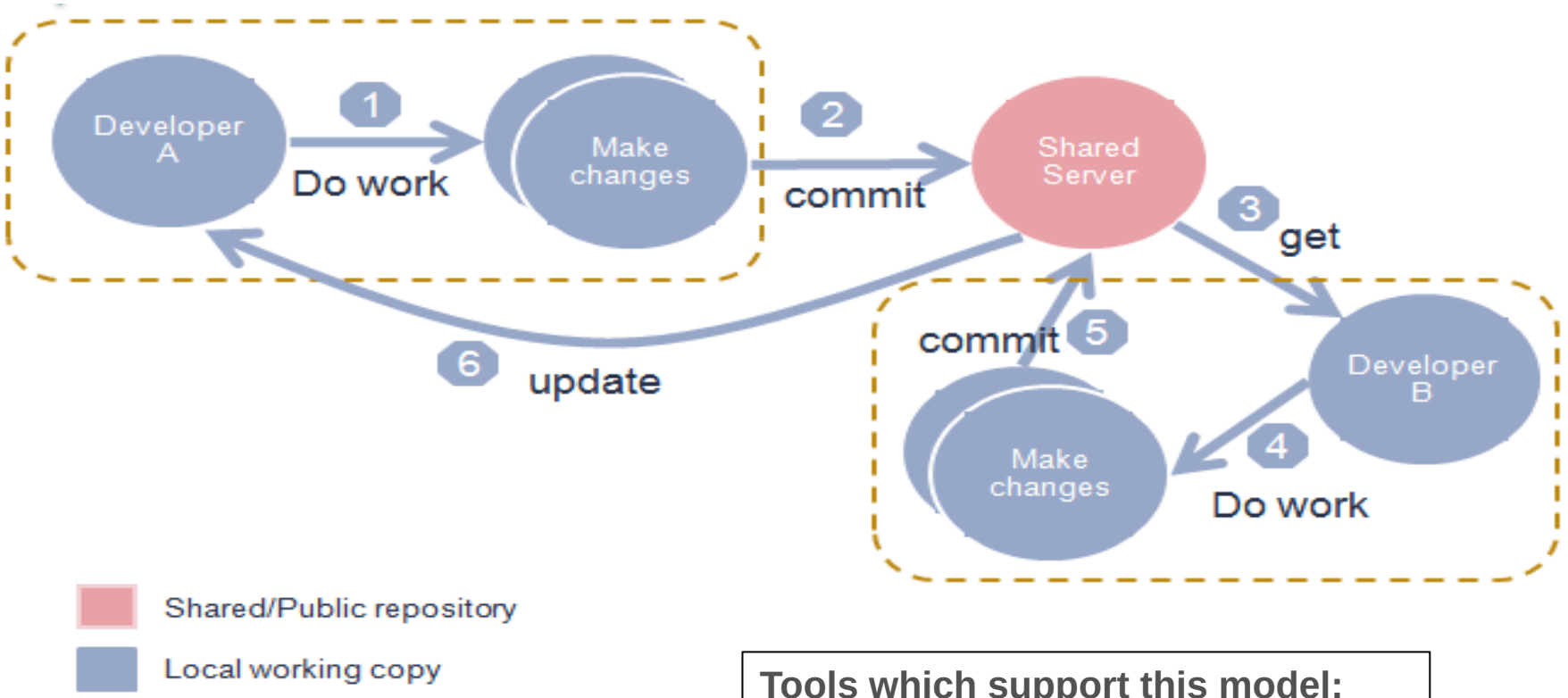


Central



Distributed

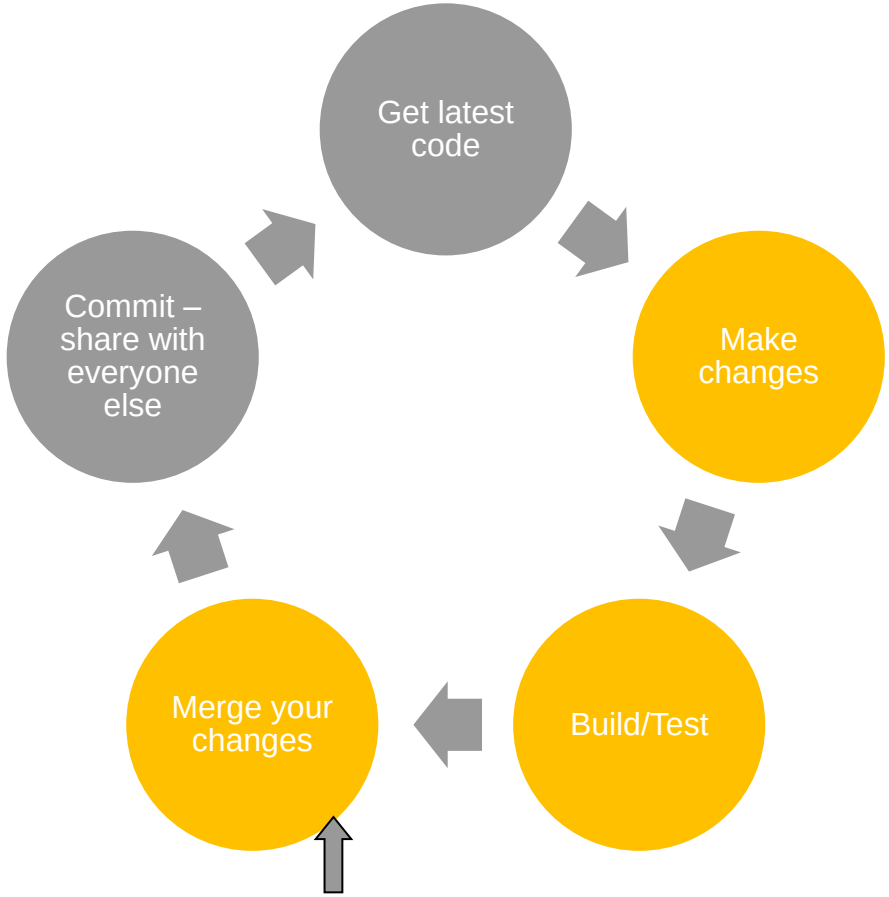
Standard – Centralised



Tools which support this model:

- SVN
- CVS
- Perforce
- Team Foundation Server

Standard VCS Workflow



Shared

Yours

Note: When merging changes you may need to go back round the build and test steps

Agenda

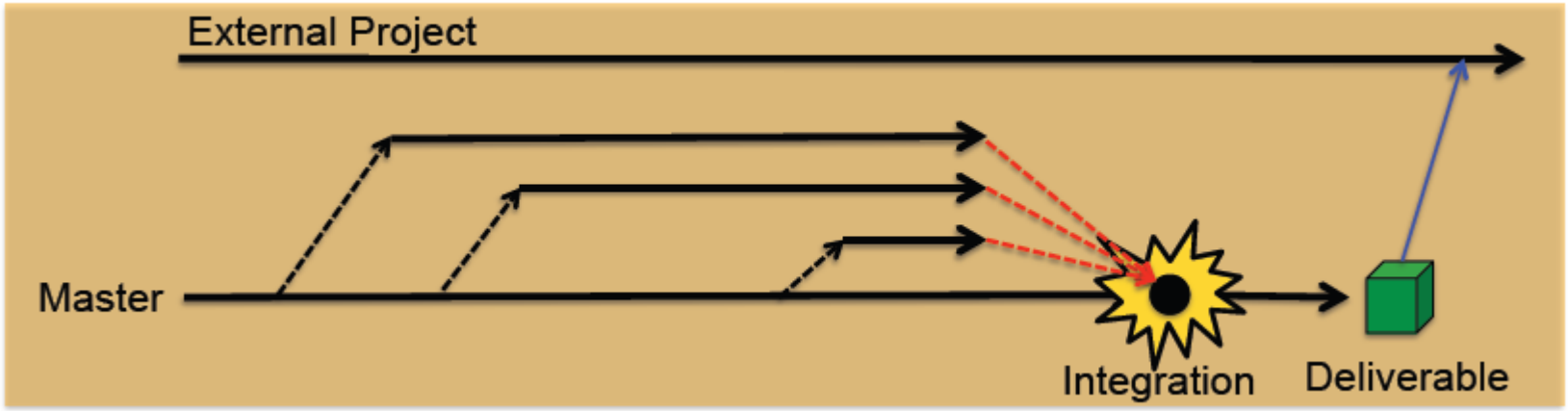
- Performance and Operations of Web Applications
- Load-Balancing and high availability
- Caching on different layers
- Software Configuration Management
- **Continuous Integration**

Why Integration?

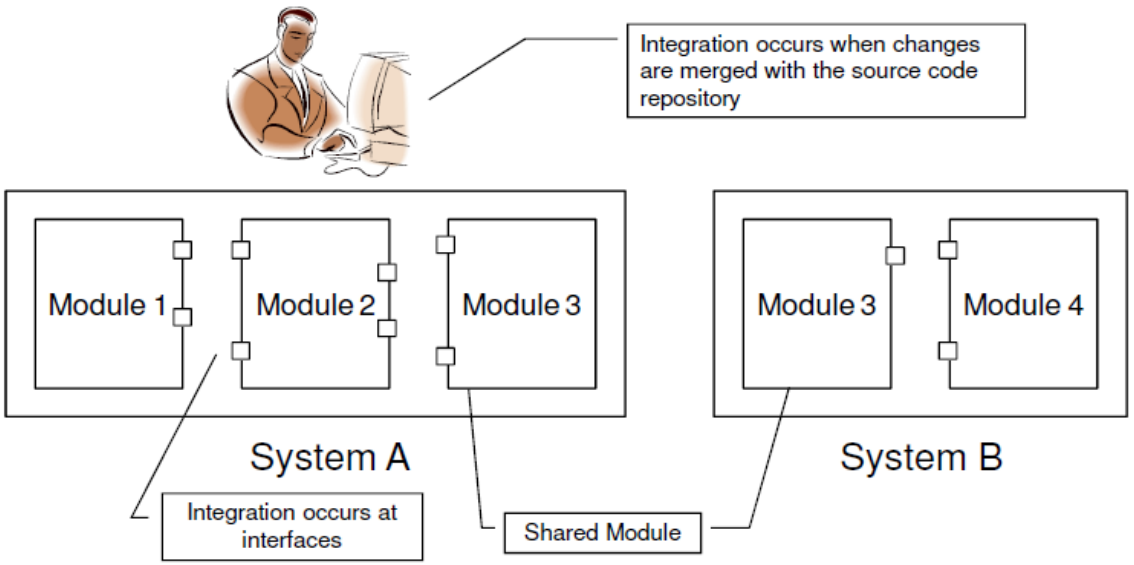
- *Integration is making different modules work together.*
- Modularization
  - Enables team development
  - Makes complex systems manageable
- Modules have to work together
  - i.e. They must be integrated
- Integrated Modules need to successfully…
  - ...Compile, Run, Pass Test & Deploy
- Quality insurance
  - Automatic feedback
  - Gain of time for the QA

Why Integration?

- Late Integration is… not good
  Can Lead to “Big Bang Integration”



Integration Challenges



- Integration becomes expensive
  - If made manual (build, test, deployment…)
  - With too less checkin's (hours or days)
  - If integration problems and bugs are detected too late
  - Long time between integration increases risk of merge

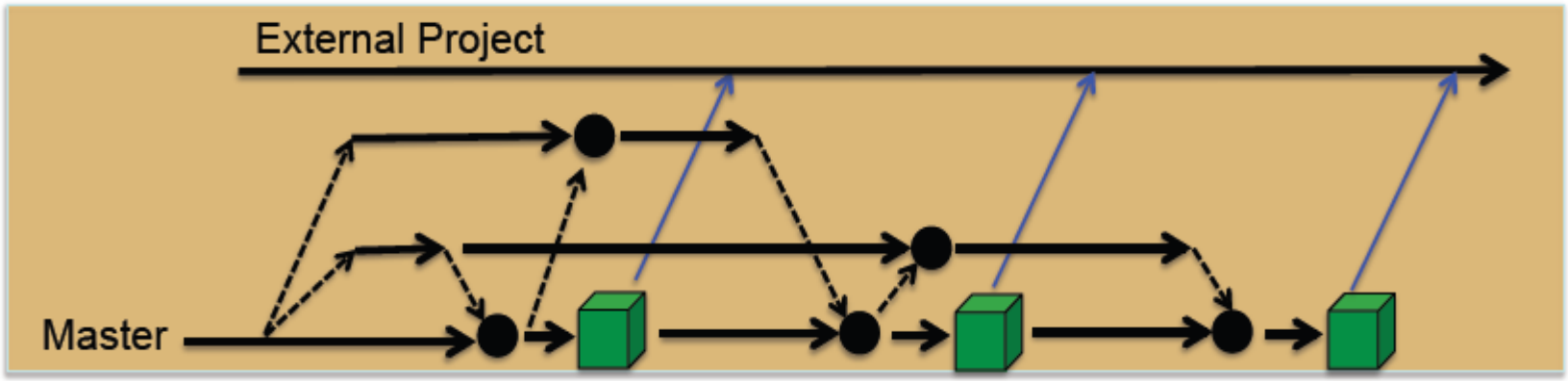
What is Continuous Integration

- *"Continuous Integration is a software development practice where members of a team integrate their work frequently, each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible"*

- CI is a part of XP practice

- Benefits of CI:
  - Reduces Development Risks
  - Gives a Better Understanding of Where You Are, What Works, & How Many Bugs are Left
  - Find Developer Conflicts Sooner
  - Increased Productivity
  - Release Software that Contains Fewer Bugs and is More Stable

Continuous Integration



Deliverable

Integration

What Should the Build Do?

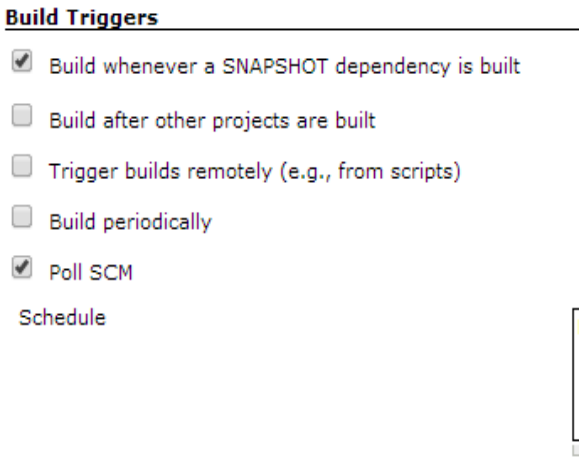
- A build can:
  - Setup local configuration
  - Create local database(s)
  - Run scripts to migrate the db to the latest schema
  - Generate code
  - Assign version # the project
  - Compile/deploy solution
  - Run unit tests
  - Run integration tests
  - Run acceptance tests
  - Generate documentation
  - Run static analysis (code coverage / naming standards)

Jenkins for CI

- Jenkins is a central build server
  - Written in Java
  - Runs as Servlet in Tomcat or standalone (e.g. Windows Service)
- Perform “jobs”
  - Configurable
  - Does all the tedious work (checkout, build, test, package, deploy)
- Adapts to your workflow
  - More than 350+ plugins available
- Jenkins forked from Hudson (Oracle) in January 2011
  - Most Hudson developers switched to Jenkins
  - Oracle donated Hudson to the Eclipse Foundation in May 2011

Jenkins

- Jenkins supports many common SCM systems like SVN, CVS
  - Automatic checkout/update on build
  - Build on updates in the SCM
- Jenkins relies on external build tools
  - Many common tools supported -like Ant, Maven, Ivy (plugins)
  - Allows to reuse the build scripts used locally
- Jenkins builds the project:
  - after another project has been build
  - according to a given schedule
  - after code changes in the SCM
  - many other (=> Plugins)



Build Tool: Maven

- High level open source build scripting framework extensively used in JAVA world which helps in standardizing the overall build process

- Maven helps in:
  - Standards
  - Conventions
  - Lower Maintenance Costs
  - Knowledge Sharing
  - Dependency Management
  - Promoting good architecture

Maven

- Standards and Conventions
  - A standard directory structure
  - A standard, but extensible build lifecycle
- Technical Documentation
  - Generate technical project documentation
  - Easy to integrate code quality metrics
- Project Architecture
  - Encourages developers to use modular design
  - More flexible architecture
  - Reduced complexity and maintenance costs
- Dependency Management
  - Understand precisely what libraries your application needs
  - Safer and more reproducible builds
  - A standard way to share internal libraries
- Release Management
  - A standard way to track and release versions
  - Official versions stored on a central server
  - Can be used to automate the deployment process

Code Quality

- Better Code quality
- Enforce corporate coding standards
- Detect potential bugs
- Code is easier to maintain
- Train new staff
- Keep technical debt down
  - Technical Debt: The cost of poor quality code:
  - Harder to make changes
  - Too much time spent on fixing bugs
  - Takes too long to add competitive new features

- How do we pay off technical debt?
  - Enforce coding standards
  - Teach developers good coding practices
  - Spend time keeping the code clean (refactoring)
  - Review code as a group
  - Long and slow if done manually
  - Automating code review using tools

PMD

- PMD scans Java source code and looks for potential problems like :
  - Possible bugs - empty try/catch/finally/switch statements
  - Dead code - unused local variables, parameters and private methods
  - Suboptimal code - wasteful String/StringBuffer usage
  - Overcomplicated expressions - unnecessary if statements, for loops that could be while loops
  - Duplicate code - copied/pasted code means copied/pasted bugs

- PMD Basic Rules
  - EmptyCatchBlock: Empty Catch Block finds instances where an exception is caught, but nothing is done. In most circumstances, this swallows an exception which should either be acted on or reported.
  - EmptyIfStmt: Empty If Statement finds instances where a condition is checked but nothing is done about it.
  - EmptyWhileStmt: Empty While Statement finds all instances where a while statement does nothing. If it is a timing loop, then you should use Thread.sleep() for it; if it's a while loop that does a lot in the exit expression, rewrite it to make it clearer.

PMD Example Report

PMD Results

The following document contains the results of PMD 4.2.2.

Files

com/ /OrgTierBean.java

Violation	Line
Avoid empty catch blocks	251 - 253

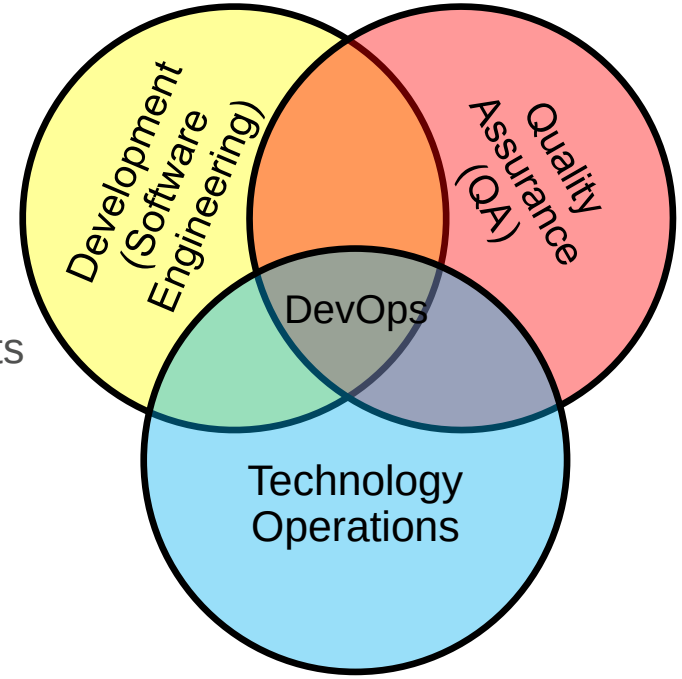
com/ /PrntSummBean.java

Violation	Line
Avoid unused private fields such as 'orgCol'.	30

Outlook: DevOps

- Continuous Integration focuses on Development Process
- Short Time-to-Market requires additional actions
  - Speed up Quality Assurance
  - Shorten deployment cycles to production
- Development / QA / Operations need to work together
  - These are often different teams / companies in classical projects
  - Continuous Integration is technology/tools
  - DevOps is a cultural change enabled by technology/tools

- “DevOps ([…]) is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.” (Wikipedia)



Original by By Rajiv.Pant [CC BY 3.0],

For further information:

Cal Henderson
Building Scalable Web Sites
O'Reilly Media, Inc.Pub 2006
ISBN-10: 0-596-10235-6

Theo Schlossnagle
Scalable Internet Architectures
Sams 2005
ISBN-10: 0-672-32699-X

Steve Souders
High Performance Web Sites
O'Reilly Media, Inc. 2007
ISBN-13: 978-0-596-52930-7

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

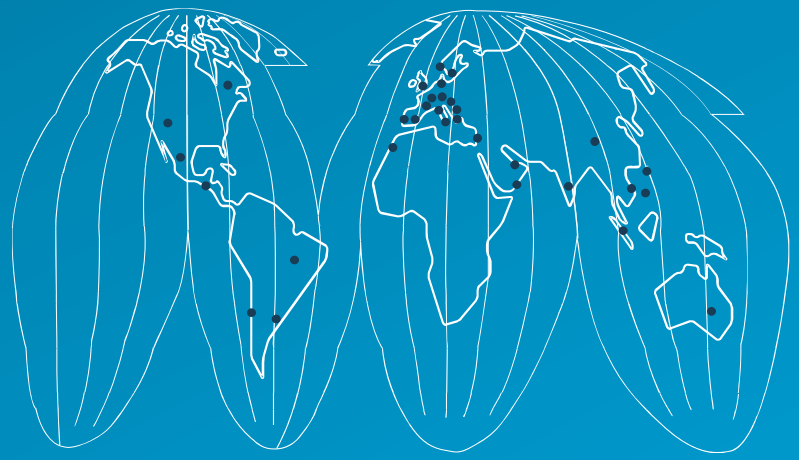
People matter, results count.

About Capgemini

With more than 120,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2011 global revenues of EUR 9.7 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com

Research and science live on the exchange of ideas, the clear arrangements are thereby useful

Disclaimer:
Although this presentation and the related results were created carefully and to the best of author's knowledge, neither Capgemini nor the author will accept any liability for it's usage.

If you have any questions, please contact:
Capgemini | Offenbach
Dr. Martin Girschick
Berliner Straße 76, 63065 Offenbach, Germany

Telephone +49 69 9515-2376
Email: martin.girschick@capgemini.com