HSLU Hochschule Luzern

Verteilte Systeme und Komponenten

Einführung

Martin Bättig

Letzte Aktualisierung: 21. September 2022

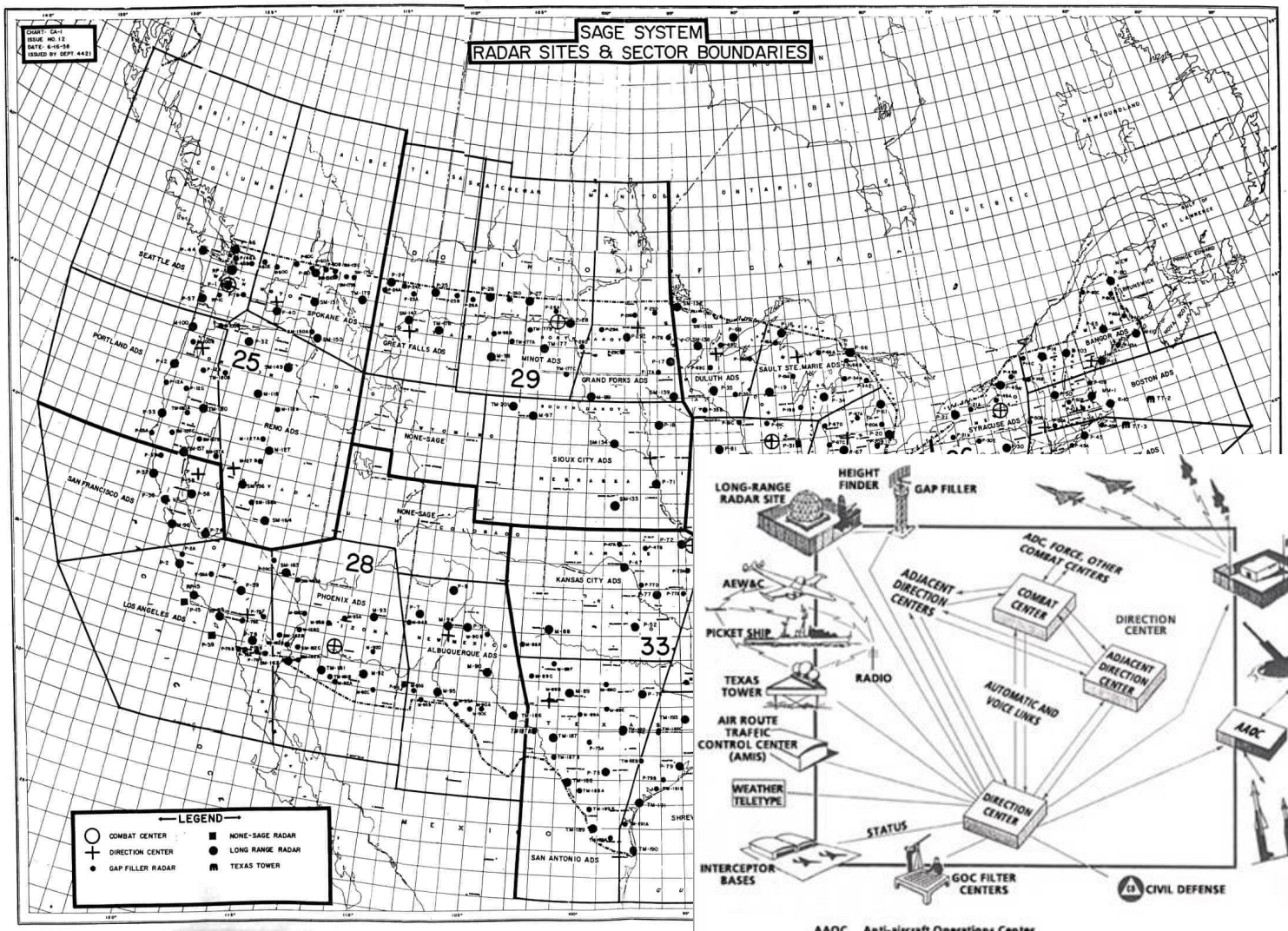
FH Zentralschweiz

Inhalt

- Einführung in das Modul
- Projektmanagement und -durchführung
- Einführung in verteilte Systeme
- Netzwerk-Einmaleins

Einführung in das Modul

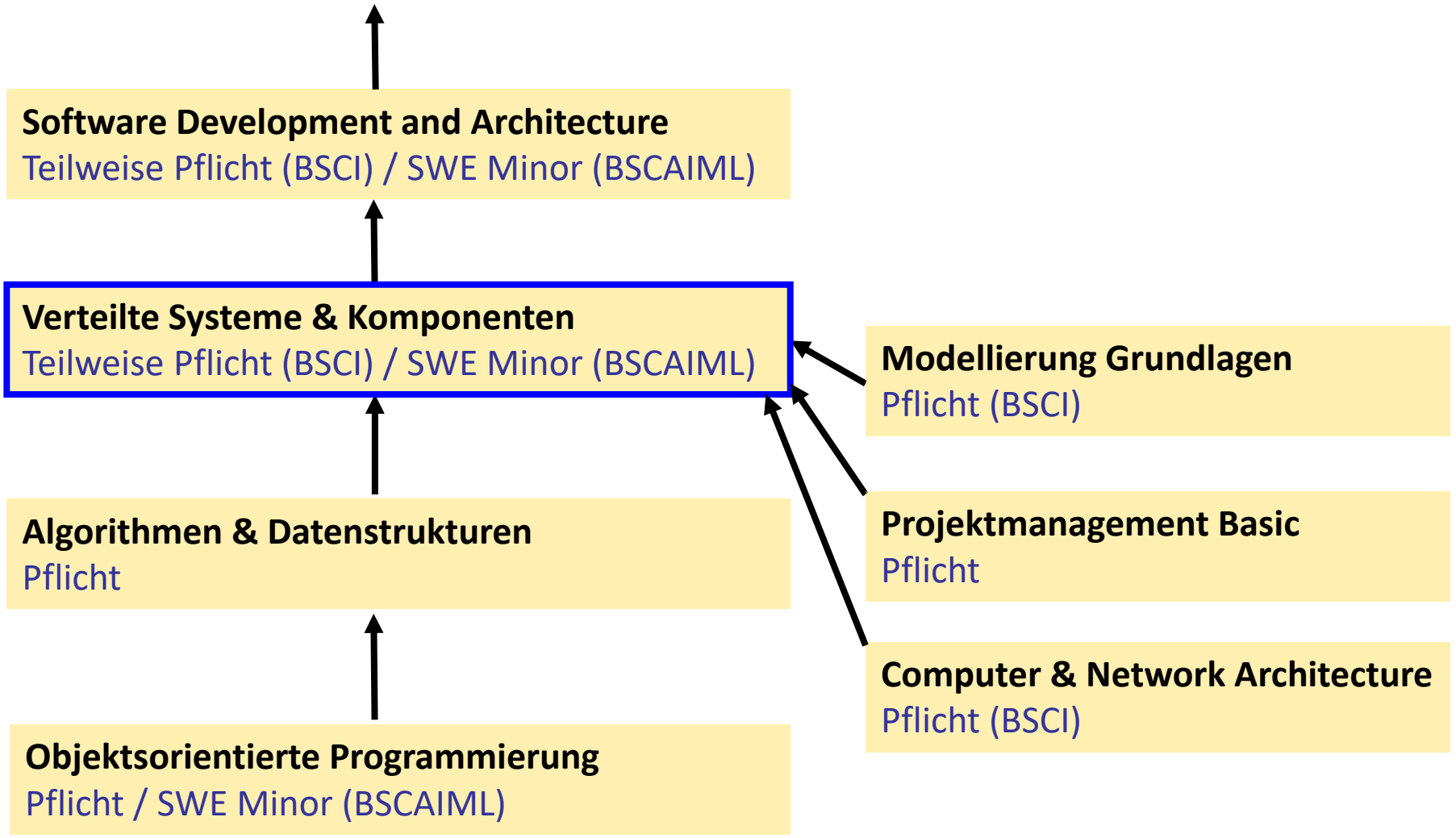
Einleitung



Organisatorisches - Plattformen

- **ILIAS:** Inputs & Projektmaterial
  https://elearning.hslu.ch/ilias/goto.php?target=crs_5501092
- **Forum:** Für Fragen und Fragen und Diskussionen
  https://elearning.hslu.ch/ilias/ilias.php?ref_id=5569017&cmd=showThreads&cmdClass=ilrepositorygui&cmdNode=10h&baseClass=ilrepositorygui
- **Zoom:** Streaming von Inputs für Ausnahmefälle (Krankheit, Termine, usw.)
  https://hslu.zoom.us/j/94103562967?pwd=VU9jRU5XSXZ2bnNFNXNkeFc3VEhuQT09
  (Meeting-ID: 941 0356 2967 / Kenncode: 146601)
  Hinweis: *Das Modul ist auf «vor Ort»-Unterricht ausgelegt.*

Einbettung im Curriculum



Agenda

	Block 3 - Donnerstag: 15:30 - 17:50		Block 2 - Freitag: 12:50 - 15:10		
	Thema	**Raum**	**Thema**	**Raum**	**Projekt**
SW01 *KW38*	PR: Einführung (Modul, Projekt, Gitlab, Netzwerke) [Bam] Raum: S12_013	*S1A_220*	VS: Serverprozesse mittels Sockets und RPC [Bam]	*S1A_220*	**Einführung**
SW02 *KW39*	EP: SCM und Buildautomatisation (CI) [Gro]	*S1A_220*	KO: Komponenten- & Schnittstellendesign [Bam]	*S1A_220*	**Sprint 1** 29. Sep. - 20. Okt.
SW03 *KW40*	VS: Message-orientierte Kommunikation [Bam]	*S1A_220*	EP: Dep.-Management / Buildserver (CI) [Gro]	*S1A_220*	
SW04 *KW41*	KO: Schnittstellenabstimmung & -Freigabe / Architekturbeschreibung [Bam]	*S1A_220*	KO: Modularisierung und Schichtenarchitektur [Bam]	*S1A_220*	
SW05 *KW42*	PR: Sprintreview und Sprintplaning	*S1A_220*	KO: Containervirtualisierung [Gro]	*S1A_220*	**Sprint 2** 20. Okt. - 10. Nov.
SW06 *KW43*	KO: Integrations- und Systemtest [Bam]	*S1A_201*	EP: Testing [Gro]	*S1A_201*	
SW07 *KW44*	EP: Continuous Integration [Gro]	*S1A_220*	VS: Fehlertoleranz und Resilienz [Bam]	*S1A_220*	
SW08 *KW45*	PR: Sprintreview und Sprintplaning	*S1A_330*	EP: Entwurfsmuster [Gro]	*S1A_220*	**Sprint 3** 10. Nov. - 1. Dez.
SW09 *KW46*	VS: Konsistenz und Replikation [Bam]	*S1A_220*	EP: Deployment [Gro]	*S1A_201*	
SW10 *KW47*	VS: Sicherheit in verteilten System [Bam]	*S1A_220*	VS: Skalierung und Verteilung [Bam]	*S1A_201*	
SW11 *KW48*	PR: Sprintreview und Sprintplaning	*S1A_220*	EP: Code-Qualität [Gro]	*S1A_220*	**Sprint 4** 1. Dez. - 22. Dez.
SW12 *KW49*	Feiertag: Maria Empfängnis [Kein Unterricht]		Freie Projektarbeit [Kein Unterricht]		
SW13 *KW50*	VS: Koordination verteilter Systeme [Bam]	S1A_220	KO: Komponentenmodelle [Bam]	*S1A_220*	
SW14 *KW51*	PR: Sprintreview und Bereinigung	*S1A_220*	PR: Abschluss (Prüfungsinfo, Wettbewerb, Retrospektive)	*S1A_220*	**Projektende**

Hinweis: *Raumangaben sind ohne Gewähr. Bitte beachten Sie MyCampus oder die Anzeigetafeln für allfällige kurzfristige Anpassungen.*

Studienelemente:

- VS: Verteilte Systeme
- KO: Komponenten
- EP: Entwicklungsprozess
- PR: Projekt

Dozententeam:

- Martin Bättig [Bam] (Modulverantwortung)
- Roland Gisler [Gro]

Konzept

- Inputs mit drei komplementären Tracks:
  - Entwicklungsprozess (EP)
  - Verteilte Systeme (VS)
  - Komponenten (KO)
- Übungen in **3er** oder **4er** Gruppen (Selbststudium)
  - Projekt "verteiltes Logger-System«.
  - fünf auf einander aufbauende Arbeitsaufträge.
  - Bearbeitung ist Testatpflichtig.
- Leistungsnachweis:
  - Mündliche Prüfung basierend auf Ergebnissen des Projekts

Testatbedingung

- Bearbeitung der fünf Arbeitsaufträge.
  - Schnittstellendefinition und Sprints 1-4.
  - Abgabe der geforderten Zwischenresultate pro Sprint.
  - Erwartet wird ein ernstgemeinter Lösungsversuch.
- Aktive Projektmitarbeit in der Gruppe.
  - Ausgewiesene Teilaktivitäten pro Gruppenmitglied.
- Obligatorische Teilnahme **aller Teammitglieder** an folgenden Terminen:
  - Schnittstellenabstimmung.
  - Sprint-Reviews 1-4 mit den Coaches.

GitLab – auf EnterpriseLab (ELAB)

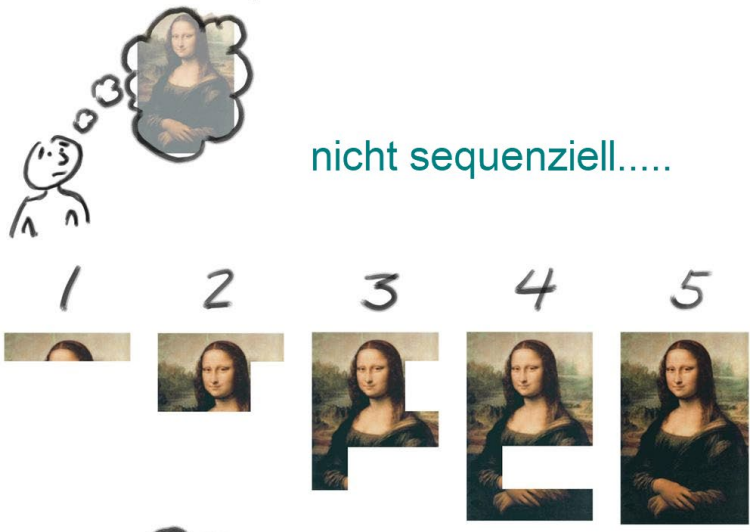
- Wir verwenden die Versionsverwaltung der GitLab der HSLU:
  https://gitlab.enterpriselab.ch
- Bitte ELAB-Konto erstellen, falls noch nicht vorhanden:
  https://eportal.enterpriselab.ch/ (Registration -> Registration SwitchAAI)
- Anschliessend bitte **einmal** beim enterprise.gitlab.com einloggen
  - Benutzer ist dann in GitLab erfasst und wir können die Rechte vergeben.
- EnterpriseLab FAQ: https://el-core.pages.enterpriselab.ch/faq/
- Andere Probleme: https://ticket.enterpriselab.ch/otrs/customer.pl

Prüfung

- Format: Mündlich.
- Dauer: 25 Minuten.
- Basiert auf der Projektarbeit (Architekturbeschreibung und Programmcode).
  - «Closed Book Exam»: Material von uns bereit gestellt
- Prüfer: Martin Bättig und Roland Gisler.

Projektmanagement und -durchführung

Sequential vs. Iterativ



aus Jeff Patton (2014), User Story Mapping

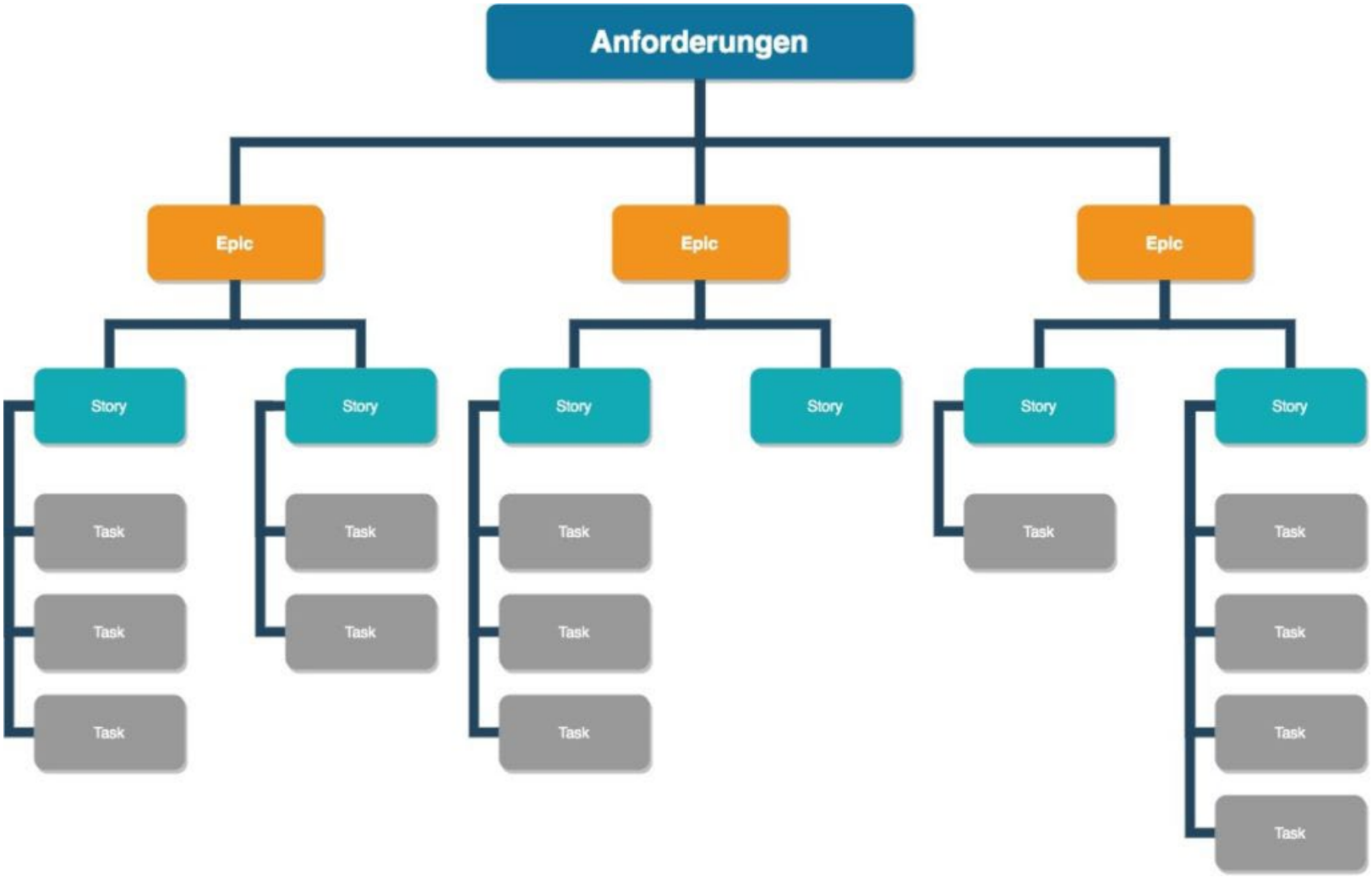
Scrum: Agiles Produktentwicklungsframework

Agiles Vorgehen ermöglicht:

- Start der Konzeption und Realisation mit unvollständigen Anforderungen.
- Änderungen der Anforderungen während der Konzeption und Realisation.
- Priorisierung der Anforderungen (u.U. werden nicht alle erfüllt).

In einer geordneten Art und Weise.

Gefässe in Scrum-Vorgehensmodell



Sprintplanung

- Sprintziel festlegen: Vom PO zu erarbeitendes und mit Team abzustimmendes Ziel, das mit dem Sprint erreicht werden soll.
  => beantwortet die Frage nach dem Sinn des Produkt-Inkrements, das im Rahmen dieses Sprints erstellt wird.
- Schätzbare Items aus dem Produkt-Backlog werden in den Sprint aufgenommen: Einträge mit hoher Priorität zuerst.
- **Eine Aufnahme in den Sprint verpflichtet zur Abarbeitung.**
- Ob schätzbar oder nicht entscheidet Scrum-Team.
  - Konfliktpotenzial zwischen Product-Owner und Scrum-Team.
- Schätzung typischerweise durch alle Teammitglieder.
- Einheit nach Scrum: Storypoints.
  - Storypoints nur bei eingespielten Teams sinnvoll.

Sprintplanung im GitLab – Begriffe

- GitLab kann zur Verwaltung der Stories im Backlog und zur Planung und Controlling der Sprints eingesetzt werden.
- Abweichende Terminologie in GitLab bzgl. Standard Scrum-Terminologie:

Scrum-Artifakt	GitLab-Feature
User story	Issue
Task	Task lists
Epic	Epics
Points and estimation	Time tracking & Weights
Produkt-Backlog	Issue-Liste mit Prioritäts-Labels
Sprint / Iteration	Milestone
Burndown-Chart	Burndown-Chart

Sprintplanung im GitLab – Vorbereitung

Konzept: jedes Team ist eine Gruppe in GitLab.

- Epics, Boards, Labels, Milestones sind auf Ebene Gruppe
- Issues sind auf Ebene Projekt: Produkt-Backlog
- Priorisierung von Issues im Produkt-Backlog:
  - Zusätzliche Labels wie z.B. Prio_A, Prio_B, Prio_C einführen
  - Labels entsprechend den Issues zu ordnen.
  - Issue-Board kann danach gefiltert werden.

Sprintplanung im GitLab – Neuen Sprint planen

Auf Gruppenebene:

- Einen entsprechenden Milestone mit sprechendem Namen (Sprint 01, Sprint 02..) und Daten wählen.
- Board öffnen, ohne Milestone Filter -> gesamtes Backlog sichtbar.
- Stories aus Backlog auswählen und Labels (ToDo etc.) zuweisen.
- Alle zugewiesenen Issues auf den aktuellen Milestone setzen (Sprint-Backlog):
- Issue auf Board anwählen -> rechts erscheint Sidebar-Menü -> dort Milestone wählen
- Abschluss, nicht erledigter Stories: Milestones weg und remove from Board (d.h. Label weg).

Projektcontrolling

Zentrale Frage: Erreichen wir das Projektziel? (Zeit, Kosten):

- Wieviel ist noch zu tun und viel Zeit und Ressourcen haben wir noch?
- Wie gut wird geschätzt? Wie kann man es bessern?
- Controlling im VSK wird mittels Gitlab gemacht.

Sprintplanung im GitLab – Controlling

- Um den geschätzten Aufwand für eine User-Story festzuhalten im Kommentarfeld des betreffenden Issue /estimate eingeben und die geschätzte Zeit in Wochen, Tagen, Stunden und Minuten angeben.
  Beispiele: 1w 3d, 5h 40m oder 4h 30m
- Um die Zeit einzutragen, wie lange man bis jetzt an der User-Story gearbeitet hat, /spend plus die Zeit im Kommentarfeld des Issue eingeben.
- Im Issue rechts unter time tracking sieht man die geschätzte Zeit und die bereits aufgewendete Zeit für die User-Story.
- Anleitung um ein Issue Template im GitLab zu erstellen: https://gitlab.com/help/user/project/description_templates.md

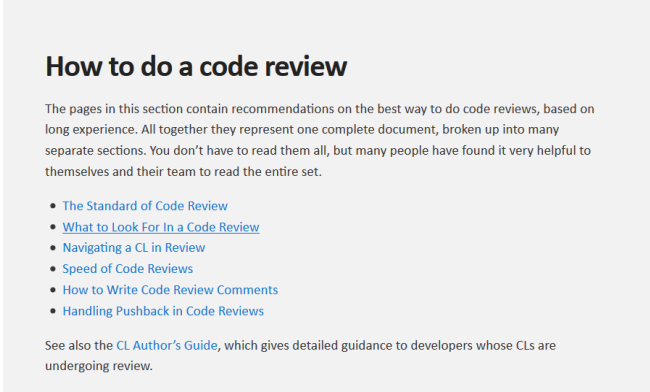
Sprintplanung im GitLab – Bemerkungen

- Auswertungen erfolgen auf Ebene "Milestone" -> diese verwenden (pro Sprint!).
- Boards sind sowohl auf Gruppen als auch Projektebene die gleichen.
- Milestones sind entweder auf Projekt oder Gruppenebene vorhanden / sichtbar. Können zu Gruppen-Milestones promoted werden.

Code-Reviews nach Google

Googles Code-Review-Guidelines
sind öffentlich:

https://google.github.io/eng-practices/review/reviewer/



How to do a code review

The pages in this section contain recommendations on the best way to do code reviews, based on long experience. All together they represent one complete document, broken up into many separate sections. You don't have to read them all, but many people have found it very helpful to themselves and their team to read the entire set.

- The Standard of Code Review
- What to Look For In a Code Review
- Navigating a CL in Review
- Speed of Code Reviews
- How to Write Code Review Comments
- Handling Pushback in Code Reviews

See also the CL Author's Guide, which gives detailed guidance to developers whose CLs are undergoing review.

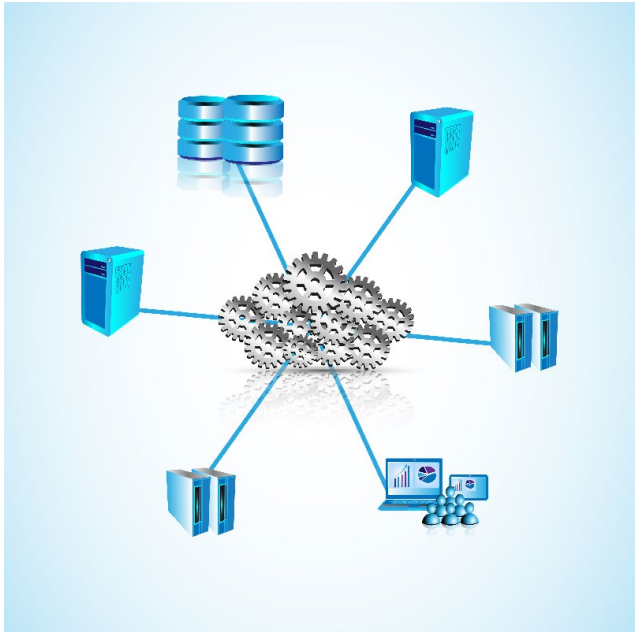
Einige Auszüge (übersetzt, vereinfacht):

- Eine Änderung muss nicht perfekt sein, muss das System in seiner Gesamtheit jedoch verbessern.
- Der Review sollte sobald als möglich nach Anfrage gemacht werden (z.B. nach einen Kontext-Wechsel). Maximal ein Arbeitstag.
- Falls eine Änderung zu gross ist: Eine Aufteilung der Änderung verlangen. Erkennungsmerkmal: Man hat keine Zeit für den Review. Bei der Linux-Kernel-Entwicklung gibt es eine ähnliche Praxis.

Einführung in verteilte Systeme

Übung: Nutzen und Fallstricke verteilter Systeme

- **Gruppe A:** Überlegen Sie sich, welche Fallstricke in einem verteilten System auftreten können.
- **Gruppe B:** Überlegen Sie sich, welchen Nutzen man aus einem verteilten System ziehen kann.

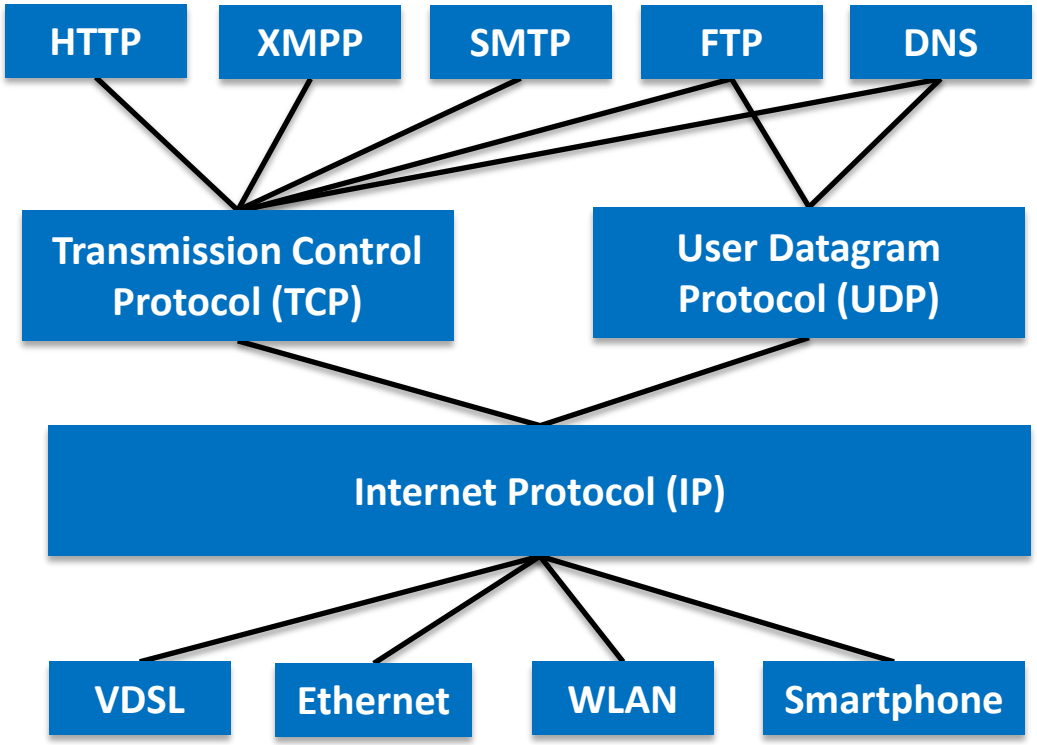


Netzwerk-Einmaleins

* Im Modul "Computer & Network Architecture"
werden die Netzwerk Konzepte ausführlich behandelt.

Internet-Schichtenmodel

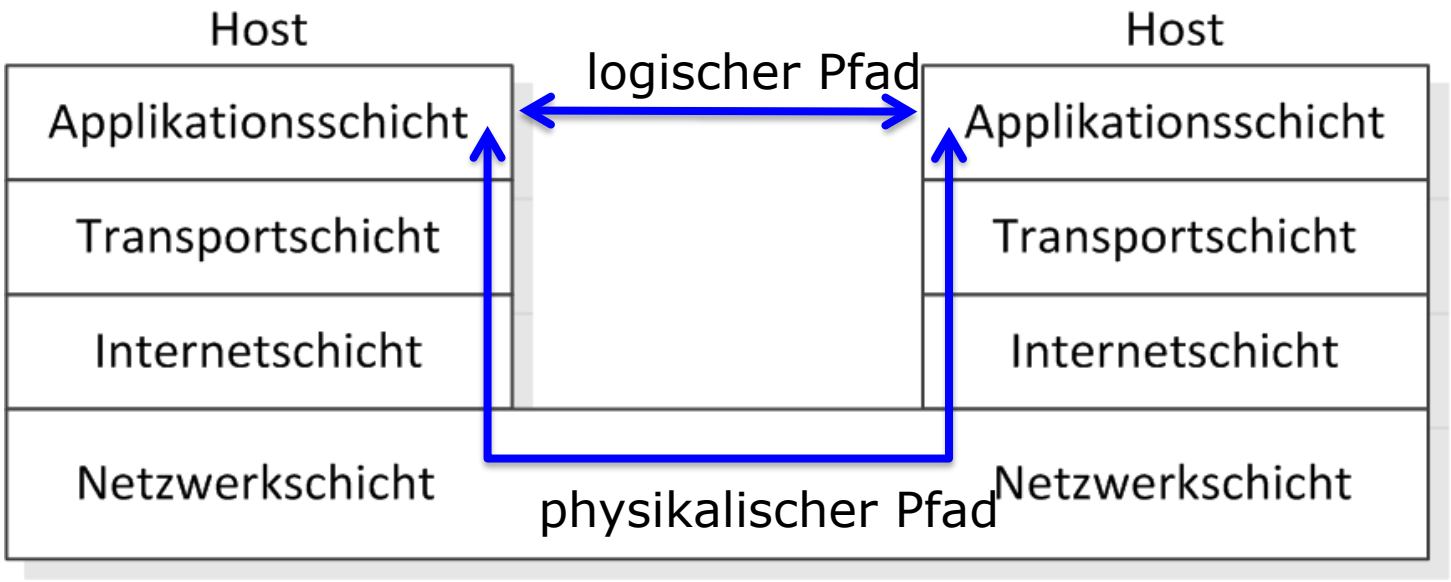
- **Applikationsschicht** (application layer)
- **Transportschicht** (transport layer)
- **Internetschicht** (internet layer)
- **Netzwerkschicht** (network layer)



Beispiele von Protokollen in den verschiedenen Schichten

Kommunikation

- Physikalisch gehen die Daten der Host-zu-Host Kommunikation durch alle Schichten.
- Logisch gehen die Daten von Applikation zu Applikation.
- Die Kommunikationsdetails sind für die Applikation **transparent**:



Begriffsdefinitionen

- **Host:** Ein am Netzwerk angeschlossener Rechner**.**
- **IP-Adresse:** Jeder Host bekommt eine im Netzwerk eindeutige IP-Adresse.
- Versionen von IP-Adressen sind:
  - **IPv4** eine 32-Bit-Zahl
  - **IPv6** eine 128-Bit-Zahl
- **Hostnamen:** Statt IP-Nummern verwendet.
- **Domain Name Service (DNS):** Dienst zur Zuordnung zwischen Name und IP-Adresse.

https://de.wikipedia.org/wiki/IPv4
https://de.wikipedia.org/wiki/IPv6
https://de.wikipedia.org/wiki/Domain_Name_System

Teilnehmer in einem Netzwerk

- **Server (dt. Bediener):** Dienstleister, der in einem Computersystem Daten oder Ressourcen zur Verfügung stellt.
  - System kann aus einem Computer oder einem Netzwerk mehrerer Computer bestehen.
  - Mehrdeutiger Begriff: Serverprogramm (Dienstanbieter) vs. Servercomputer (auf welchem Serverprogramme laufen).

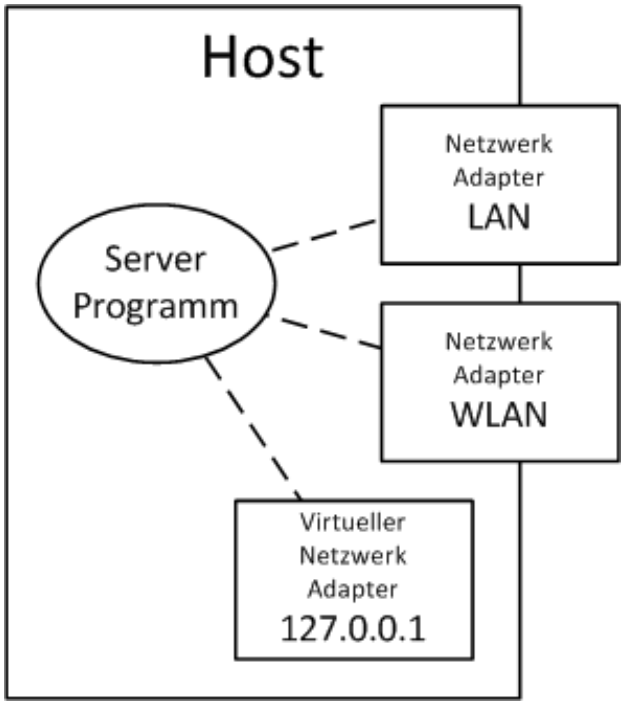
- **Client (dt. Kunde):** Dienstnehmer, der in einem Computersystem Dienste von Servern benutzt.

Network Interface Controller (NIC)

- Verbindung zwischen Computer und Netzwerk (privaten oder öffentlich).
- Physisch (On-board, PCI, USB, …) oder virtuell (Loopback, Docker, etc.)
- Loopback-Netzwerk: Virtuelles Netzwerk, welches nur eigenen Host umfasst:
  - Kein Zugriff von aussen
  - Hostname: localhost (default)
  - IPv4: 127.0.0.1
  - IPv6: :::1

NICs im Geräten

- Oft mehr als ein NIC pro Gerät, z.B. für:
  - LAN (Local Area Network)
  - WLAN (Wireless LAN)
  - Virtuelles Netzwerk
- Serverprogramm muss wissen, auf welchen Netzwerkadaptern es auf Verbindungen warten soll.



IP-basierte Netzwerke

- **IP-Adresse** identifiziert Teilnehmer in IP-basierten Netzwerken:
  - Stufe Internetschicht.
  - Kein Host im Internet hat die gleiche IP-Adresse (* Spezialfall: NAT).
- **Portnummer** identifiziert Serverprogramm auf einem Teilnehmer:
  - Stufe Transportschicht.
  - Leitet empfangene Daten an Serverprogramm weiter.
- **Socket:** Kommunikationsendpunkt
  - Spezifiziert mittels Tupel (IP-Adresse, Port-Nummer).

Host- und IP-Adressen

- Datenaustausch im Internet geschieht durch IP-Pakete.
- Empfänger der Pakete wird durch eine Kennung, die numerische IP-Adresse, identifiziert.
- Diese Zahl ist schwer zu behalten und z.T. volatil, weshalb oft der Hostname Verwendung findet.
- Die Konvertierung von Hostnamen in IP-Adressen übernimmt ein Domain Name Server (DNS).

Wichtigste Protokolle der Transportschicht

Transmission Control Protocol (TCP):

- Zuverlässiges, verbindungsorientiertes, Bytestrom-Protokoll.
- Hauptaufgabe: Bereitstellung eines sicheren Transports von Daten durch das Netzwerk.

User Datagram Protocol (UDP):

- Unzuverlässiges, verbindungsloses Protokoll.
- Unzuverlässig: Daten kommen möglicherweise nicht bei Ziel-Host an. Daten, welche ankommen, sind jedoch korrekt.

Übung: Mini-Logger mittels *ix-Tools (1)

Übung mittels Online-Programmierumgebung:

https://replit.com/@mbaettig/Logger-using-SOCAT

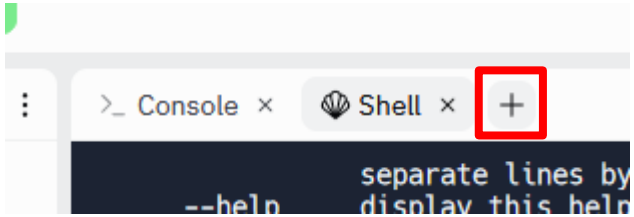
Kein HSLU-Dienst: Neues Konto anlegen, wenn Sie die Übung machen möchten.

- Logger-Server starten: `./logger_server.sh <tcp|udp> <port>`
- Logger-Client starten: `./logger_client.sh <tcp|udp> <host> <port> <msg>`

Aufgabe:

- Experimentieren Sie mit dem Mini-Logger. Ideen auf nächstem Slide.
- Was funktioniert? Was nicht?
- **Notieren Sie sich Ihre Beobachtungen. Wir diskutieren im Plenum.**

Übung: Mini-Logger mittels *ix-Tools (2)



Machen Sie folgendes:

- Starten Sie einen Logger-Server.
- Senden Sie einige Nachrichten an diesen Server. Machen Sie dafür eine neue Shell auf (siehe Bild oben).
- Starten Sie einen zweiten (oder noch mehr) Server.
- Senden Sie jeweils eine Log-Message, an die unterschiedlichen Server.
- Senden Sie auch mal eine Message an den Host: grey.baettig.ws / Port: 12345.
- Kommen Ihre Nachrichten an?
- Variieren Sie Protokoll und Port.

- **Hinweis:** Bei replit.com kann der Server nur ans Loopback-Netzwerk binden.

Fragen?

Literatur und Quellen

- Der offizielle ScrumGuide von Ken Schwaber and Jeff Sutherland, 2017: https://www.scrumguides.org/scrum-guide.html
- How to use GitLab for Agile software development von Victor Wu, 2018: https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/