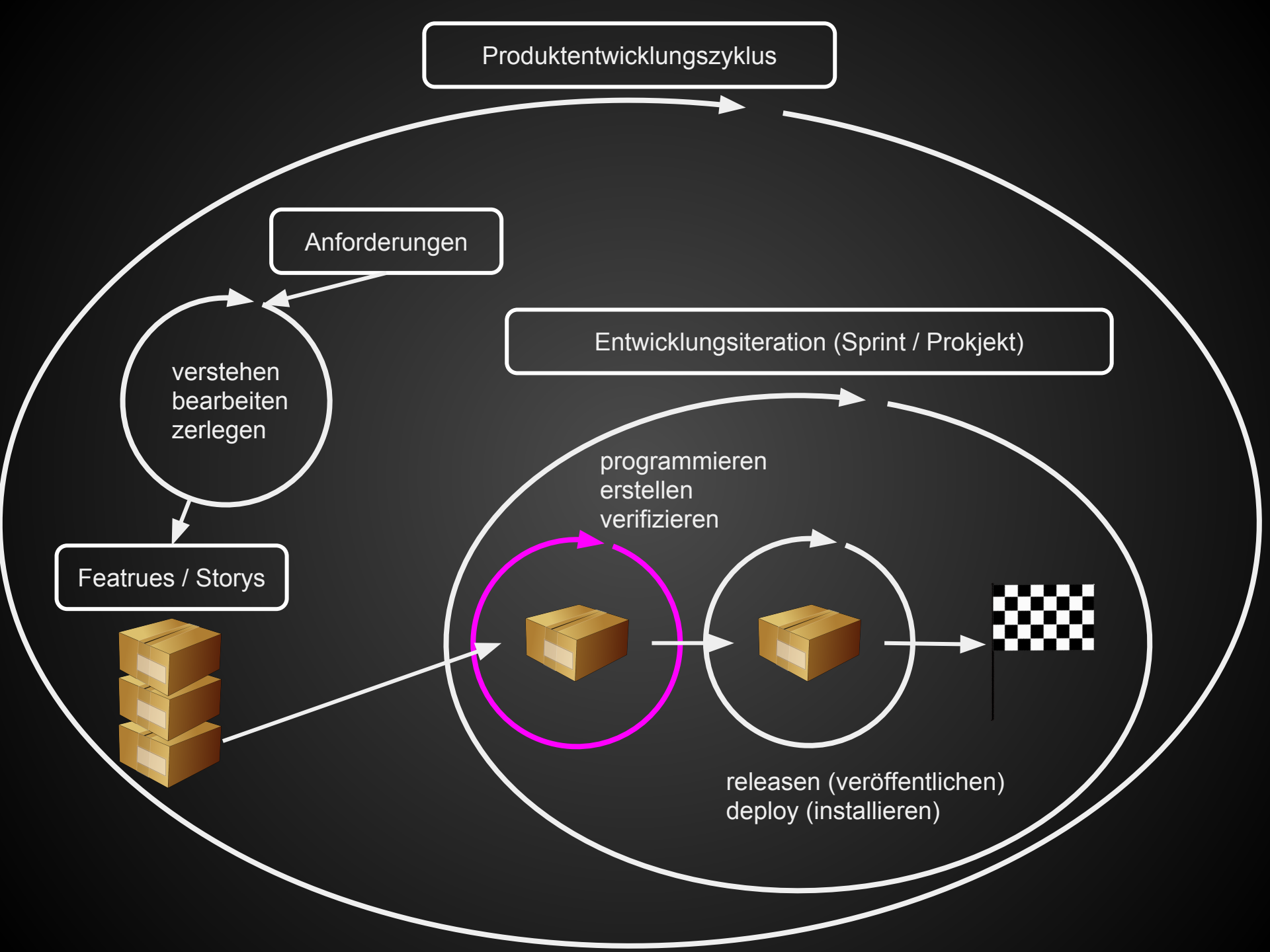
VCS - Versionskontrollsystem

Git / SVN


Produktentwicklungszyklus
Anforderungen
verstehen
bearbeiten
zerlegen
Featrues / Storys
Entwicklungsiteration (Sprint / Prokjekt)
programmieren
erstellen
verifizieren
releasen (veröffentlichen)
deploy (installieren)

Was ist VCS?

VCS = Version Control System

“... System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird...”

- Versionierung
- Beschreibung einer Änderung
- Unterstützung beim Zusammenfassen von Änderungen

Quelle: http://de.wikipedia.org/wiki/Versionsverwaltung

VCS / SCM

- SCM = Source Configuration Management
  - weiter gefasster Begriff inkludiert “build, package, deploy”

- SCM = Source Control Management
  - SCM == VCS

Arbeitsweisen

Lock Modify Unlock

- Sperren der Datei, bearbeiten der Datei, entsperren

Copy Modify Merge

- Checkout oder Update der Datei, bearbeiten Datei, gegebenenfalls Merge der Datei

Systeme

- Visual SourceSafe - http://msdn.microsoft.com/de-de/library/cc434922(v=vs.71).aspx
- CVS - http://cvs.nongnu.org
- Subversion - http://subversion.apache.org
- Git - http://git-scm.com
- Mercurial - http://mercurial.selenic.com

Alternative

- Code per E-Mail oder USB-Stift
- Unterschiedliche Stände, keine einheitliche neuste Version
- Sicherung der Software schwierig
- fehlerfreies Zusammenfügen von Änderungen fast unmöglich
- sinnvolle Automatisierung des Erstellungsprozesses unmöglich

Arten - Zentral vs. Dezentral

Zentral

- Daten auf zentralen Server
- lokale Arbeitskopien
- “History” auf Server
- Server hat immer den neusten Stand

Dezentral

- Daten liegen lokal
- “History” liegt lokal
- lokale Repositorys müsse synchronisiert werden

Zentral - VCS - Client-Server

Subversion - zentralisiertes VCS

checkout

- Arbeitskopie vom Server holen (auschecken)

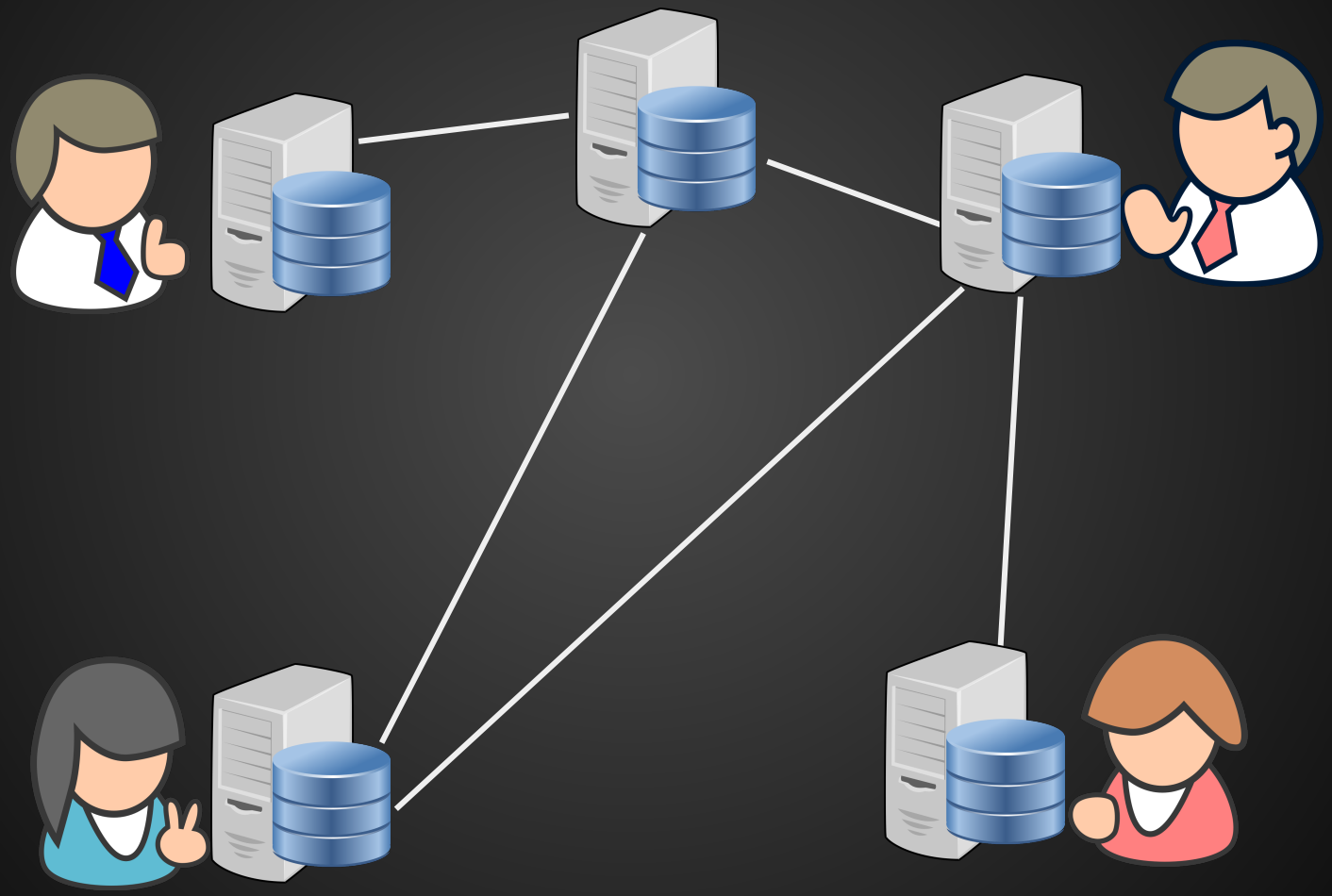
commit

- Änderungen zum Server schicken (einchecken)

update

- Änderungen vom Server holen (aktualiseren)

Dezentral / Verteilt - DVCS



Git - verteiltes VCS

checkout

- Reposistory von einem “Server” kopieren

commit

- Änderung lokal einchecken

push

- Änderungen zu einem Server übertragen

pull

- Änderungen von einem Server holen

GitHub - https://github.com

SaaS - “Software as a Service”-Anbieter für Git in der Wolke (“Cloud”)

Wenn kostenlos dann öffentlich oder privat
dann kostenpflichtig

Weitere Anbieter:

- CodePlane - https://codeplane.com
- BitBucket - https://bitbucket.org
- GoogleCode - https://code.google.com

Was gehört rein

- Quelltext
- Konfiguration
- DB-Setup-Skipte
- Build-Skripte
- weitere Skripte
- IDE-Konfiguration (Formatregeln (Spaces vs. Tabs), etc.)
- Dokumentation
- Ressourcen (Bilder / CSS / etc.)
- Maschinenkonfiguration (Puppet, etc.)

Was gehört nicht rein

- IDE-Projektdateien (z. B. .idea / .project / .classpath)
- Bibliotheken (z. B. JARs)
- Binär-Dateien (z. B. ISO-Images)

Hauptlinie - Trunk vs. Master

- enthält die aktuelle Entwicklungsversion
- wird je nach VCS als Trunk oder Master bezeichnet
- Entwickler synchronisieren sich über den Trunk/Master
- Branches werden von hier gezogen und auch wieder integriert (merge)
- sollte gesichert werden

Verzweigungen (Branches)

Ist eine Kopie des Hauptzweiges (Trunk/Master) der zu einem bestimmten Zeitpunkt gezogen wurde mit dem Ziel neue Funkionen (Feature) umzusetzen, Fehler (Bug) zu beheben und Versuche (Spikes / Prototypes) durchzuführen.

Verzweigungen II

- Branches sind sinnvoll aber nicht mehr als absolut nötig verwenden!
- Warum: Erhöhen die Gesamtkomplexität durch Erhöhung der Anzahl der Versionen der Software
- Sinnvoll für zeitlich begrenzte Experimente oder als Kopie der Produktionsversion des Codes für Bugfixes

Arten

Feature Branch

- dient Entwicklung neue Funktionalität

Release Branch

- wird genutzt um eine Bestimme Menge an Funktionalität in ein Release zusammen zufassen

Feature Branch

Vorteil:

- Funktion kann losgelöst entwickelt werden bis sie fertig ist
- ein Commit auf Mainline
- leichte Entfernung (Rollback des Commits)

Nachteil:

- Gefahr von “Integrationsschmerzen”
- Andere Entwickler sehen die Änderungen sehr spät

Release Branch

Vorteil:

- Branch beschreibt Release
- Gut wenn Produkt in mehreren Versionen live (z. B. Libre Office)
- Fehler können für bestimmt Releases behoben werden

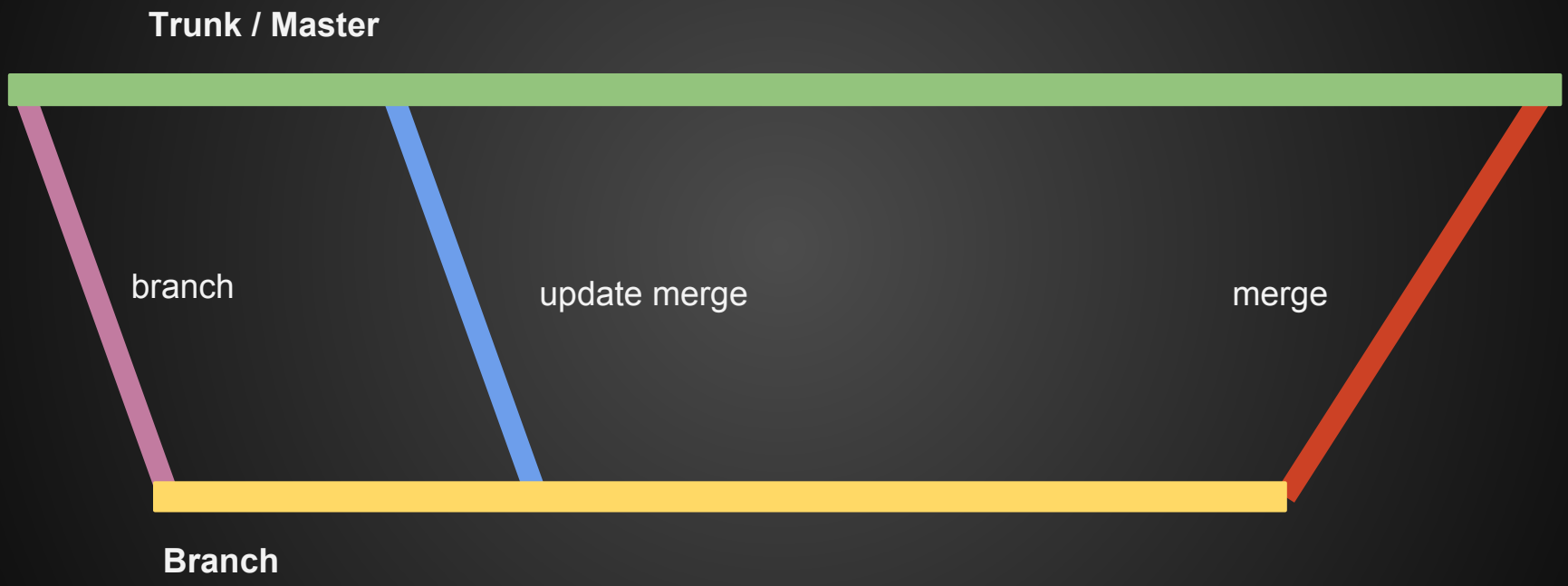
Nachteil:

- ungeeignet wenn nur eine Version zu einer Zeit live (Apps und WebApps) - Manageingaufwand

Tag

- Tag = read-only Branch
- Kann genutzt werden um Releases oder erfolgreiche Builds im VCS zu markieren (taggen)

Zusammenfügen (aka Merge)



Fragen?