Neno Loje
teamsystempro.de

31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Kenntnisstand:

Was ein Professional (Scrum) Developer alles können muss

Beginn: 15:40

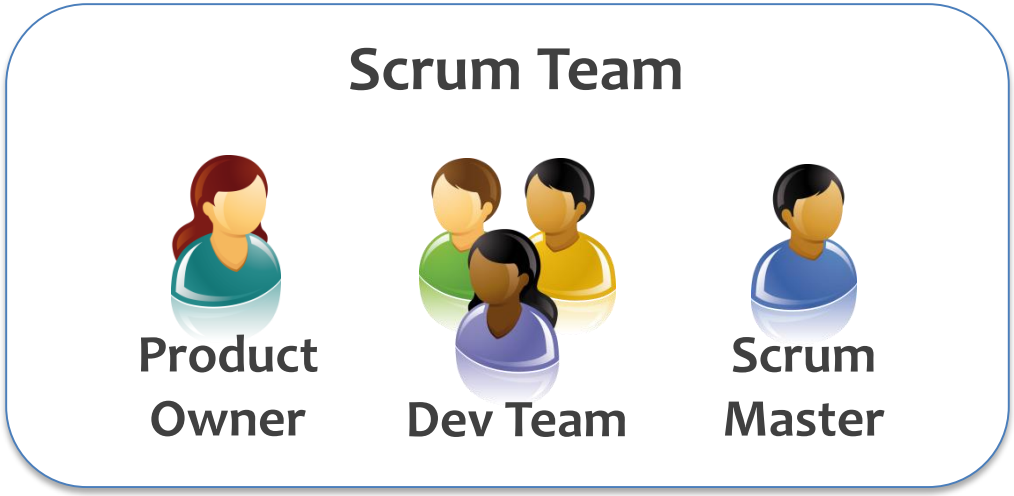
Neno Loje

Team Foundation Server (TFS) | Application Lifecycle Management (ALM)

Moderne Softwareentwicklungsmethoden (Scrum)

Scrum Team

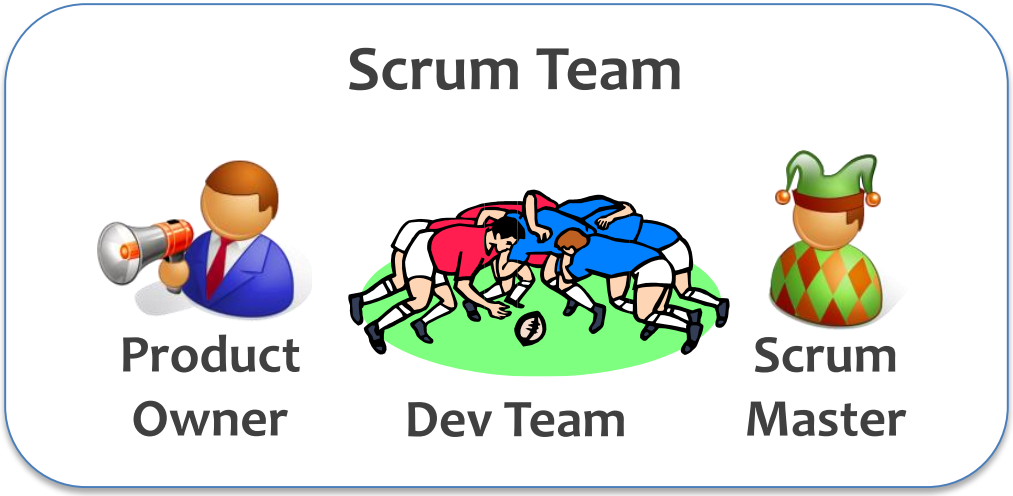
- The Scrum Team consists of
  - a Product Owner,
  - the Development Team,
  - and a Scrum Master



Quelle: http://scrumguides.org/scrum-guide.html#team

Scrum Team

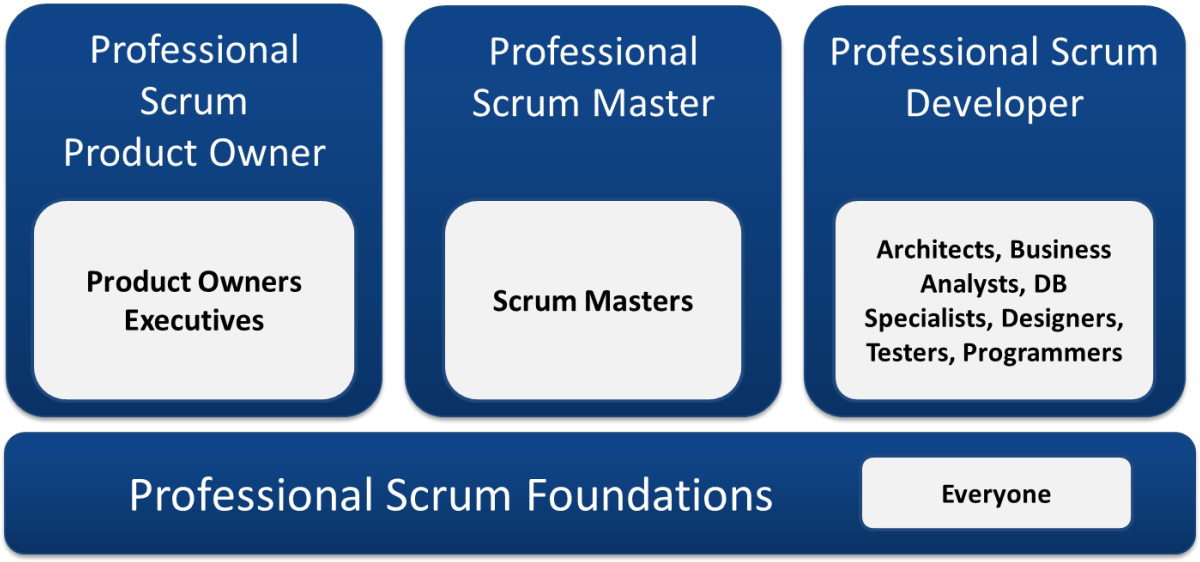
- The Scrum Team consists of
  - a Product Owner,
  - the Development Team,
  - and a Scrum Master



Quelle: http://scrumguides.org/scrum-guide.html#team

Professional Scrum Developer (PSD) Program

Gegründet 2009 von Microsoft und scrum.org



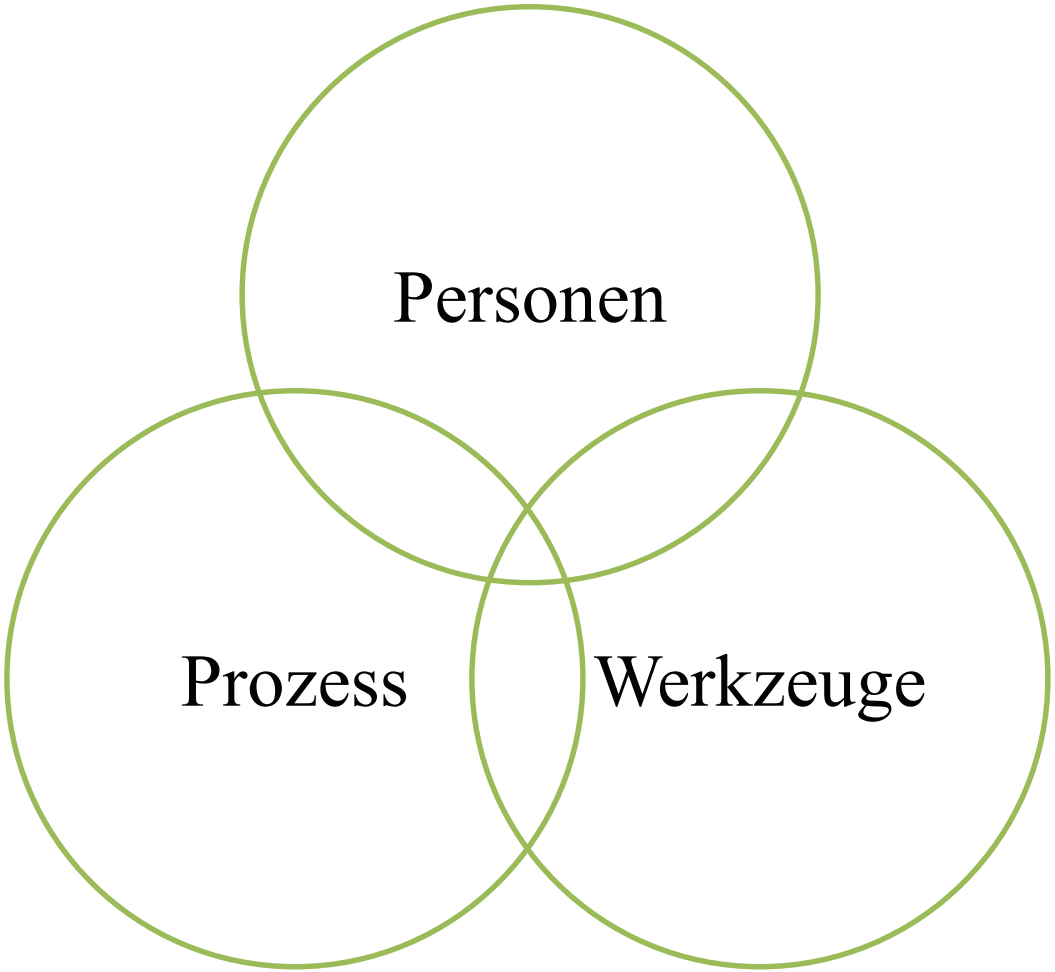
Mehr über das PSD Program: http://msdn.microsoft.com/en-us/vstudio/ff433643.aspx

Anna-Karenina-Prinzip

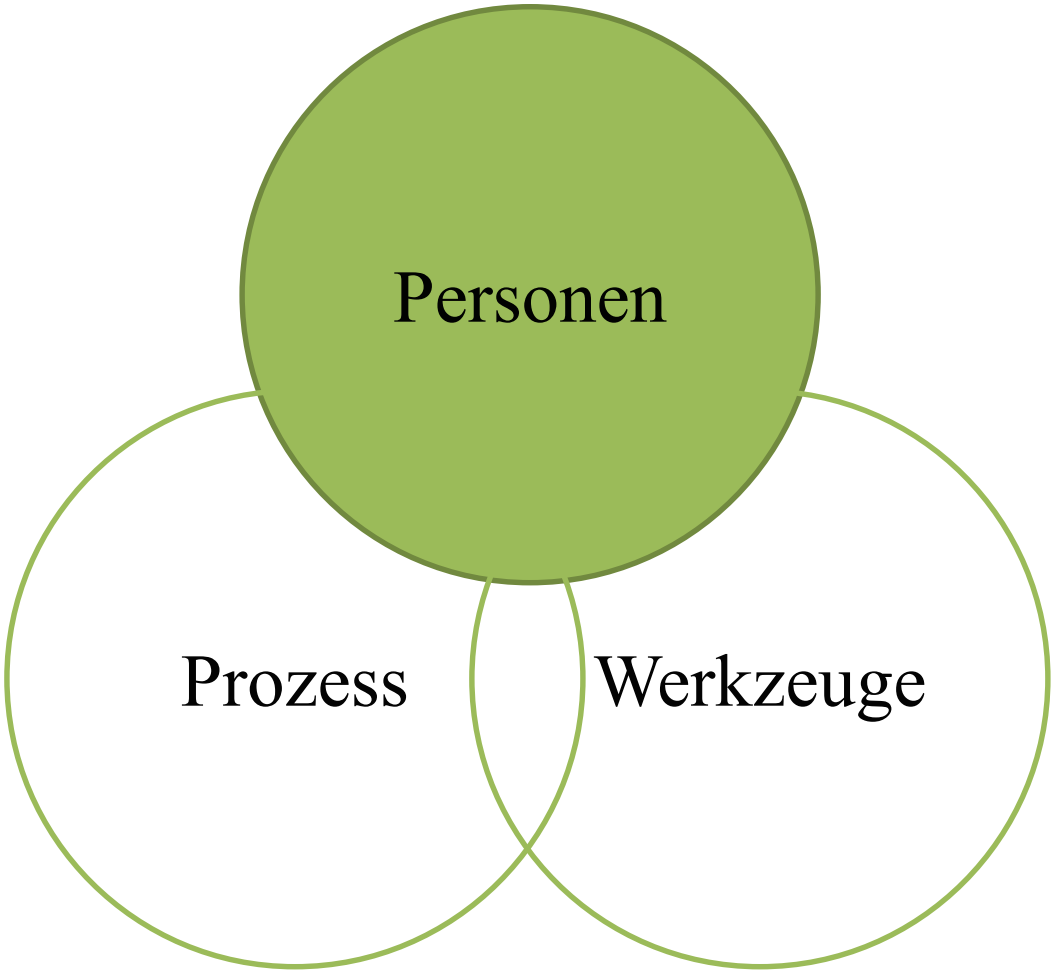
„Alle glücklichen Familien gleichen einander, jede unglückliche Familie ist auf ihre eigene Weise unglücklich.“

-- Leo Tolstoi , 1877/78

Erfolgsfaktoren



Beginnen wir mit dem schwierigsten…



Welche Rolle spielt das Development Team?

"The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint."

--Scrum Guide, July 2013

- They are **self-organizing**
- Development Teams are **cross-functional**
- **No titles** for Development Team members
- **No sub-teams** in the Development Team
- **Accountability** belongs to the Development Team

Quelle: http://scrumguides.org/scrum-guide.html#team-dev

Teams bestehen aus Menschen

- Unterschiedliche Kulturen
- Unterschiedliche Persönlichkeiten
- Unterschiedliche Arbeitsgeschwindigkeit
- Menschen sind keine Maschinen
- Menschen haben Gefühle
- Menschen haben schlechte Tage

Dysfunctions of a Software Development Team

- No collaboration
- Don’t care / rude
- Alphas / cowboys
- Won’t listen
- Won’t learn
- Won’t teach
- Specialists
- Slackers
- Too many meetings

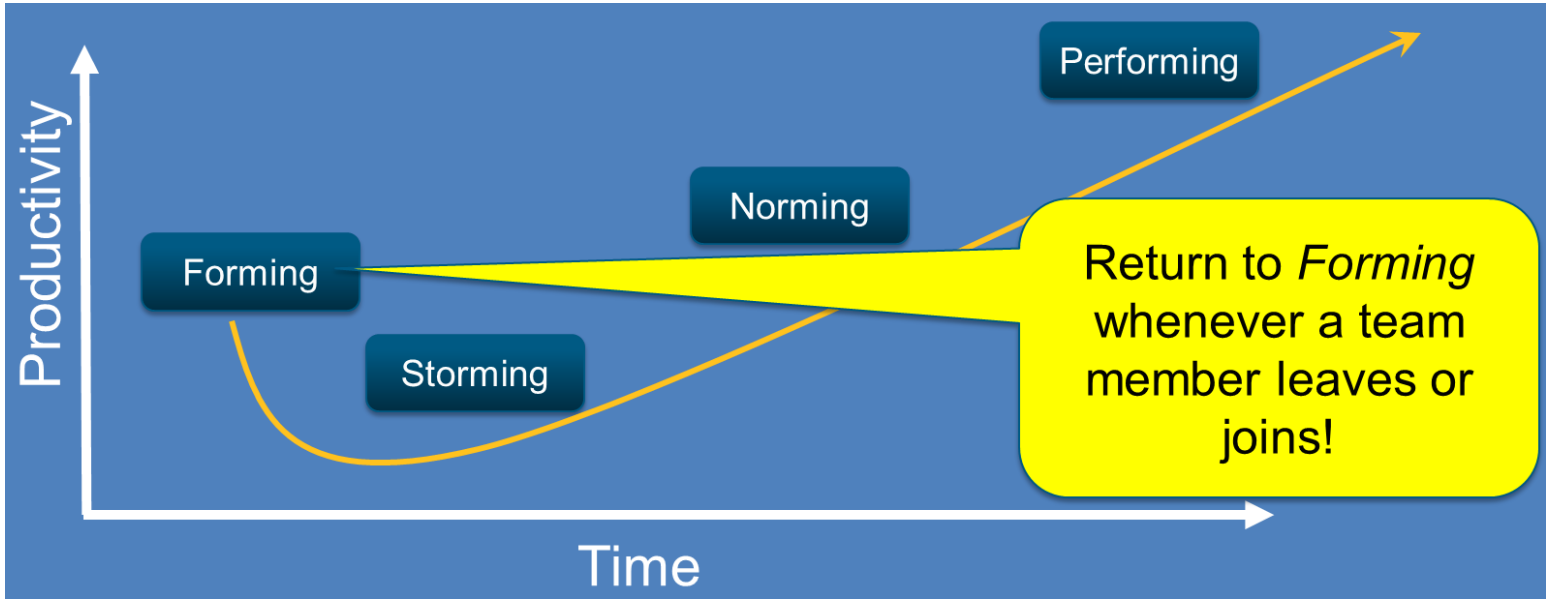
- “That’s not my job”
- “I’m not a tester”
- “I’m 80% done”
- “It compiled on my machine”
- “I did my part”
- “Nothing is blocking me”
- “It was your code that failed”
- “I was on time, you were early”
- “You just can’t implement my design”

Quiz: What about Dieter?

Dieter is your organization's only SQL Server administrator. He holds all the passwords and all database changes must go through him. He's happy to help your team with its database tasks, but only for a few days. He's very busy, and important. He makes it clear that he doesn't work for you or your boss and his help is just a courtesy.

(Bruce) Tuckman Model

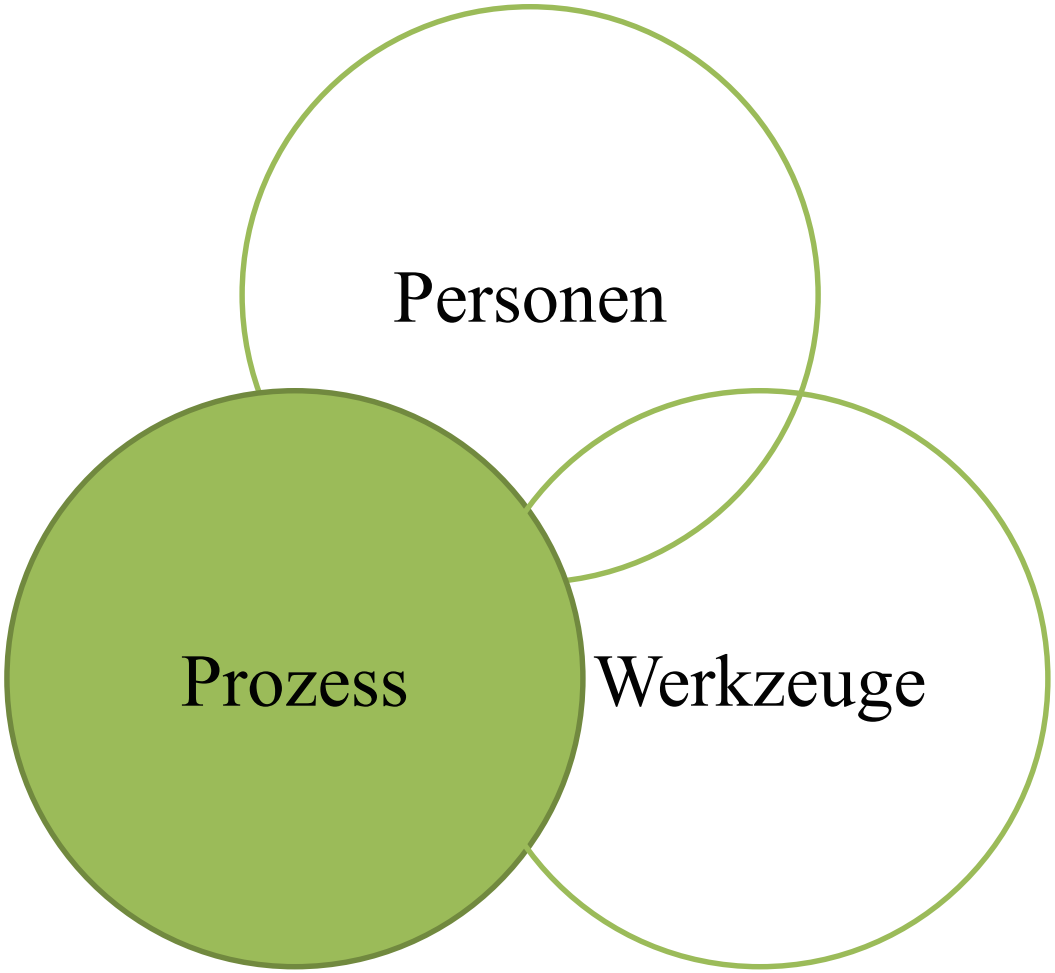
- In 1965, Bruce Tuckman identified four necessary phases of group (team) development



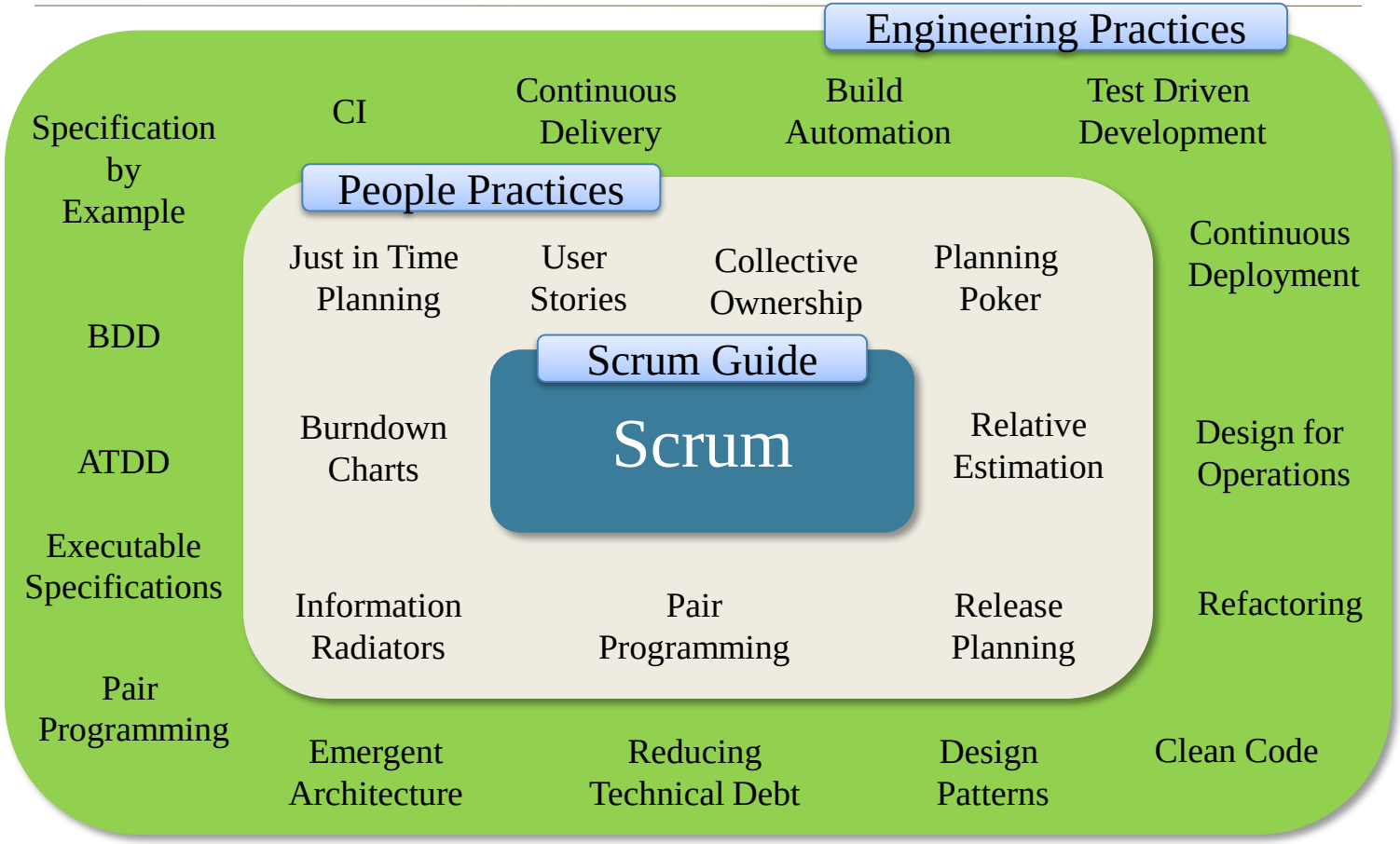
Some interesting Tactics

- Use the Pomodoro Technique
- Minimize eMail Distractions
- Implement the *Forgiveness* Pattern
- …

Erfolgsfaktoren



Complementary Practices



Professional Scrum Developer-Knowhow

Development Practices

- Product Backlog refinement
- Relative (team) estimation
- Burndowns and Cumulative Flow Charts
- Emergent architecture
- Continuous Integration (CI)
- Test-Driven Development (TDD)
- Test Smells
- Acceptance testing
- Clean coding
- Pair Programming
- Code Review
- Overcoming dysfunction

Scrum

- Plan Releases and Sprints
- Create Sprint Backlog
- Definition of Done
- Conduct Sprint Reviews
- Reflection in Retrospectives

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

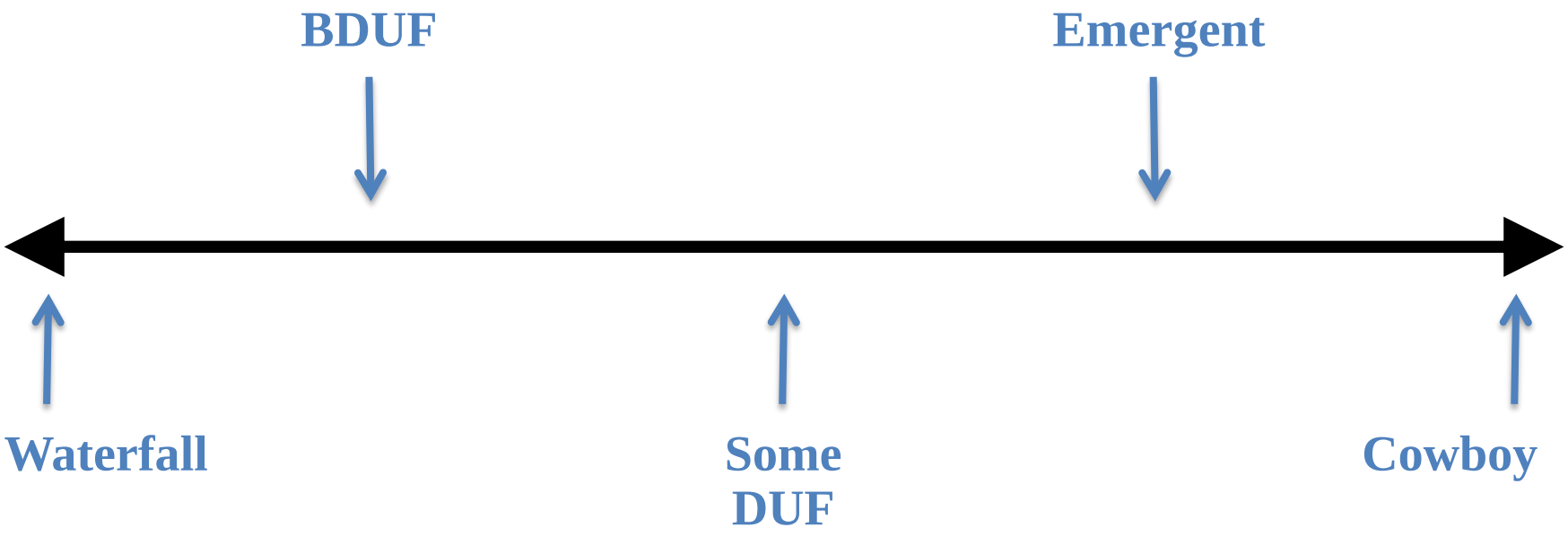
That is, while there is value in the items on
the right, we value the items on the left more.

Big Design Up Front (BDUP)

- Why Big Design Up Front *Seems* Good
  - Requirements are better understood
  - Thinking things out ahead of time saves headaches
  - Changes to specs are cheaper than changes to code
  - Bugs are cheaper to fix early
- Why Big Design Up Front *Isn't* Good
  - Requirements aren't well known in the first place
  - Requirements change
  - Developers are not able to foresee all problems
  - Customers are not able to tell you what they want

There is no business value in architecture or design!

Where are you today?

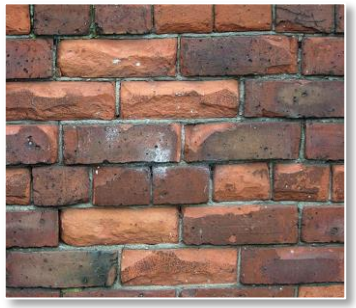
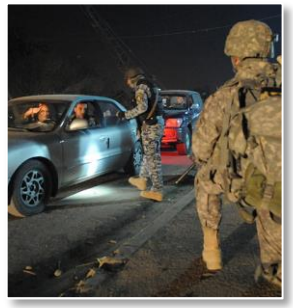


What is *Emergent Architecture?*

- The opposite of Big Design Up Front
- It allows architecture to emerge as you develop
- Some architectural decisions will need to be made before you write your first line of code
  - Technologies, frameworks, languages, patterns, practices
- Most of these decisions will be hard to change later

Gebrauchseignung (Fitness for Purpose)

- Eine Lösung sollte für den geplanten Zweck geeignet sein.
- Beispiel für eine User Story:
  - Titel: Alle Autos müssen in der Bahnstr. 87 halten
  - Beschreibung: Als Sicherheitsbeauftragter möchte ich, dass alle Autos vor dem Gebäude halten, sodass keine Teilnehmer verletzt werden.
  - Abnahmekriterien:
    - Die Lösung sollte universell verständlich sein,
    - 24/7 Verfügbar sein
    - niedrige Betriebskosten haben

It's all About Business Value

- Every sprint should generates an increment of potentially-shippable business value
- Therefore:
  - It is not ok to have an "architecture sprint"
  - It is not ok to have an "infrastructure sprint"
  - There is no such thing as sprint 0
  - And while we're on the topic: there's no such thing as "done done"

Don’t Over-Architect the Product

- Build at least one increment of business functionality every sprint
- Build enough of the architecture and design to support that business functionality
- Build adequate architecture and design to meet any non-functional requirements

- **Solve today’s problem today, tomorrow’s problem tomorrow**

Quizfrage

- In sprint 3 you start implementing the customer management component of the web site. You've committed to delivering basic add, edit, and view functionality.
- But, you know that in sprint 4 the product owner is going to want the ability to delete a customer.
- How should you consider this while delivering the sprint 3 increment?

Was bedeutet für uns Qualität?

- Erfüllt es die Anforderungen?
- Erfüllt es die Erwartungen?
- Hat es einen Nutzen?
- Ist die Lösung für den Zweck geeignet?
- Sind erforderliche Qualitätsattribute eingehalten?

Development / Definition of Done

Wann sind wir fertig?

- Explizite "Definition of Done" (DoD)
- Konstante (Mindest-)Qualität
- Basis für Schätzungen
- Beeinflusst von unternehmensweiten Richtlinien
- Messbar
- Automatisiert

Example: A Simple Definition of Done

1. Coded
2. Acceptance criteria met
3. Tests pass
4. Checked-in

Be Transparent

- Everyone should be able to inspect …
  - The team's Definition of Done (DoD)
  - The product backlog (user stories)
  - The sprint backlog (tasks)
  - The impediments
  - Deployments
  - Test results
  - Bugs
- Everyone should adapt to these inspections

What is Done?

Fundamental:

- Code Analysis
- Coded
- Refactored
- Code Reviewed
- Unit Tested
- Documented
- Continuous Build
- User Acceptance

Mature:

- Design Review
- Design Refactored
- Integration Tested
- Regression Tested
- Security Tested
- Load/Soak Tested
- Scalability Tested
- Performance Tested

Unit Tests – warum?

- Sicherheitsnetz
- Weniger Bugs
- Schöneres Design
- Ermöglichen Refactoring
- Fokus bleibt auf "Done"
- Hilft eigentliche Absicht zu dokumentieren
- Forciert Entwickler über das Ziel nachzudenken
- Umgehende Belohnung
- *Prüft auf die Existenz eines Bugs*

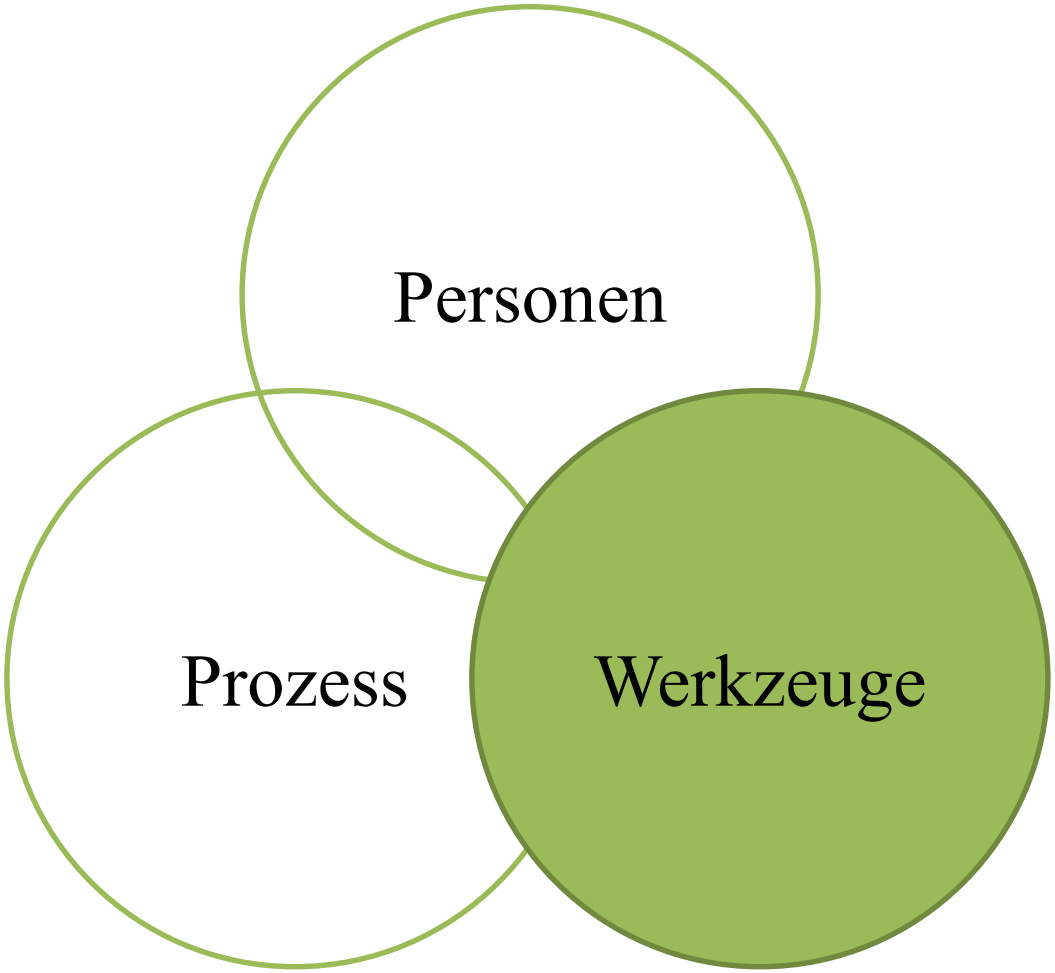
Nur Sauberer Code ist qualitativer Code

- Sauberer Quellcode erfordert ständiges, schonungsloses säubern (Refactoring)
- Niemand sonst wird Deinen Code aufräumen
  - Kein Zimmer-Service….
- Broken-Windows-Theorie

Refactoring

- The process of improving your code after it has been written by changing the internal structure of the code without changing its external behavior
  - In other words, improving your code without breaking your unit tests
- Refactoring should only be done to add business value or to meet the team's definition of Done

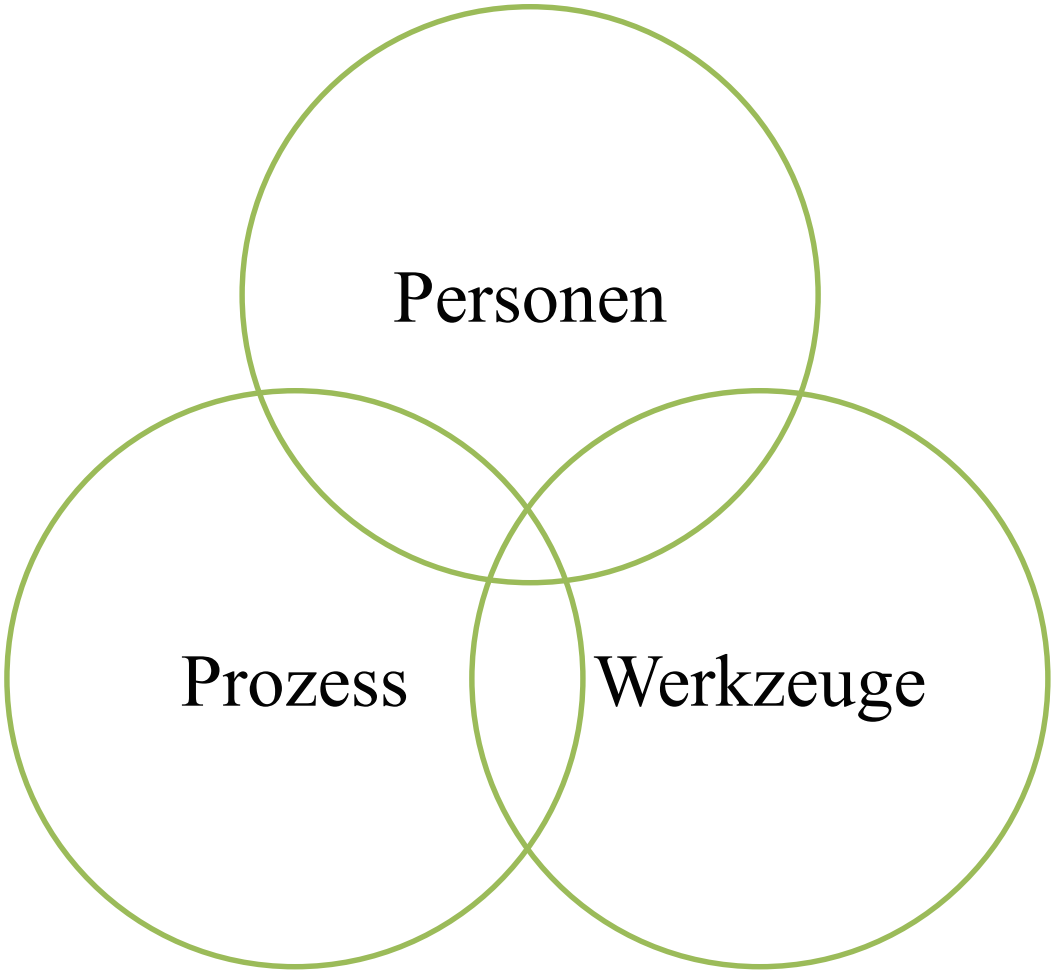
Erfolgsfaktoren



Tool-Kenntnisse

- Application Lifecycle Management (ALM) Tools
- Development Environment
- Configuration Management
- Branching & Merging
- Continuous Integration (CI) Tools
- Unit Testing
- Measuring Code Quality
- Refactoring Tools

Zusammenfassung



Professional Scrum Developer Glossary

- ALM (Application Lifecycle Management)
- ATDD (Acceptance Test-Driven Development)
- BDD (Behavior-Driven Development)
- Branching
- Clean Code
- Code Coverage
- Collective Code Ownership
- Continuous Delivery
- Continuous Deployment
- Continuous Integration (CI)
- Cyclomatic Complexity
- Cross-functional
- Definition of Done
- Developer
- DevOps
- Development Team
- DRY (don’t repeat yourself)
- Engineering Standards
- Extreme Programming (XP)
- Feature Toggle
- Increment
- Pair Programming
- Refactoring
- Scout Rule
- Scrum Board
- Scrum Guide™
- Scrum Team
- Self-organization
- Specification by Example
- TDD (Test-Driven Development)
- Technical Debt
- User Story
- Unit Test
- Velocity

Quelle: http://www.scrum.org/Resources/PSD-Glossary

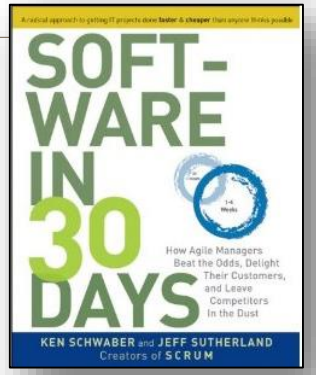
Weitere Informationen:

- Scrum Guide - das offizielle Regelwerk zu Scrum
- Scrum Open Assessment - Kostenfreie Online-Prüfung (30 Fragen) zu Scrum-Themen
- Developer Open Assessment - Kostenfreie Online-Prüfung (30 Fragen) zu Scrum Developer-Themen
- PSD Course Topics - Themenliste aus dem PSD-Kurs
- PSD Subject Areas - Themen, die jeder PSD kennen sollte
- PSD Glossary - Glossar zu den wichtigsten Begriffen
- Professional Scrum Developer auf scrum.org - "The Home of Scrum"

Buchempfehlungen zu Scrum

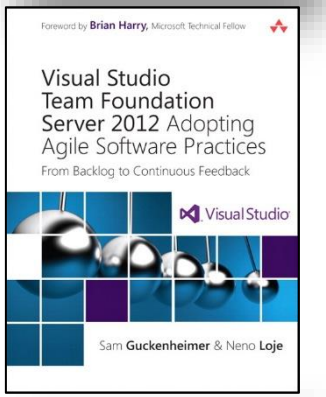
- Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust

  Von den Erfindern von Scrum. Zielgruppe: Entscheider



- Visual Studio Team Foundation Server 2012: Adopting Agile Software Practices, From Backlog to Continuous Feedback

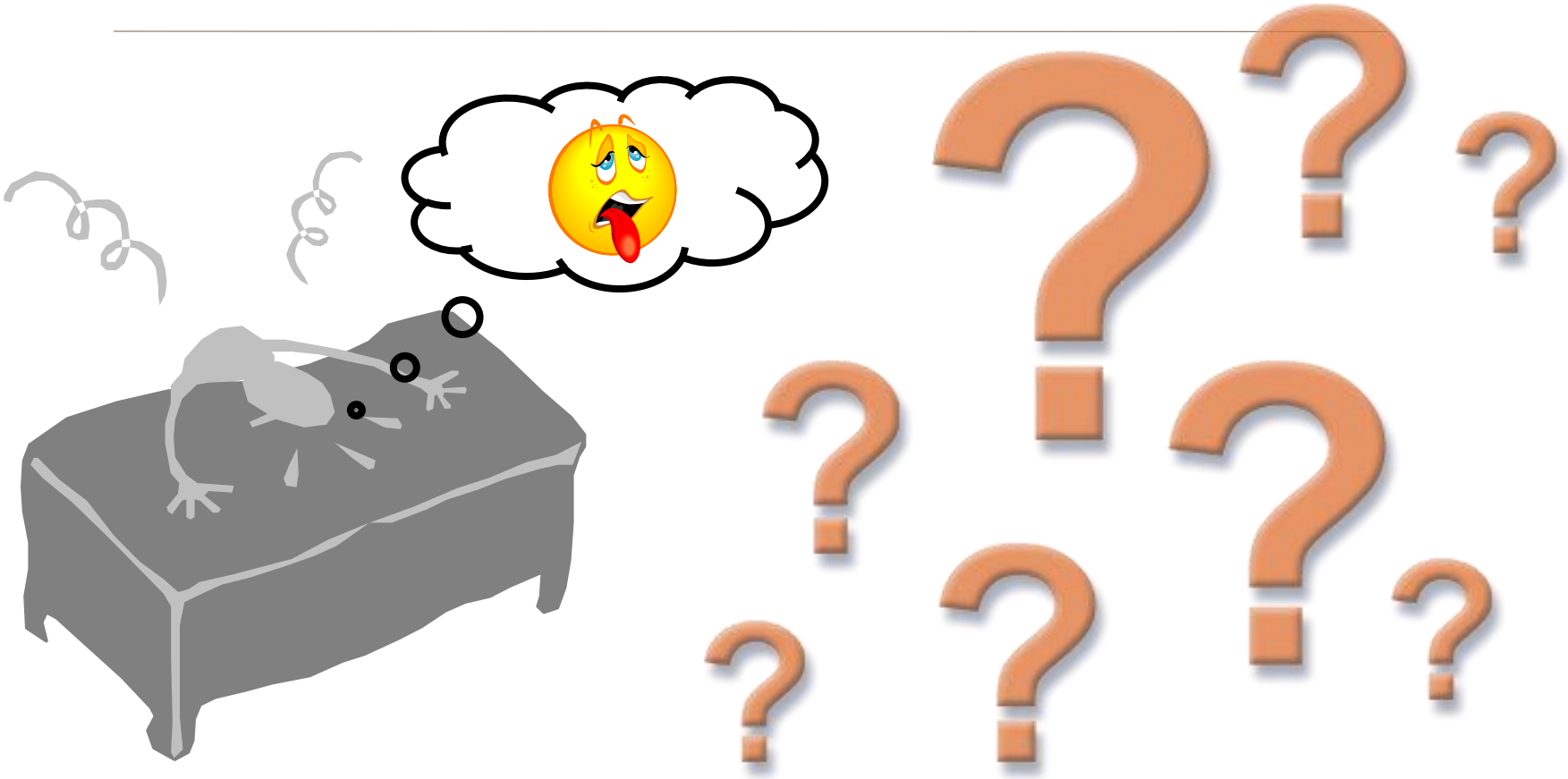
  Vom Product Owner von Visual Studio. Zielgruppe: jeder



- Professional Scrum Development with Microsoft Visual Studio 2012

  Empfohlen für alle Scrum-Teammitglieder

Danke für Ihre Aufmerksamkeit!



Im Anschluss oder per E-Mail an:
neno.loje@teamsystempro.de

Über Neno Loje

- Neno unterstützt Unternehmen und Teams **moderner** und **agiler** zu werden.
- Seine Schwerpunkte sind **Scrum**, **TFS** und **DevOps**.
- Auch die Einführung des **Team Foundation Servers** – von der Entscheidung über die Migration bis zur individuellen Anpassung.
- Besonders stolz ist er auf den **Professional Scrum Developer**-Kurs, dem Ausbildungsprogramm für Scrum-Teammitglieder.
- Neno ist seit 2010 Referent beim **Herbstcampus**.