



eclipse

3 Jahre „The Eclipse Way“ in einem Großprojekt

Ivo Eitner, ivo.eitner@arcor.de

Martin Lippert, martin.lippert@it-agile.de

Überblick

- Kurze Einführung in die Szenerie
- „The Eclipse Way“ im Überblick
- Die Schritte der Einführung
 - Von den einfachen zu den schwierigeren Techniken
- Erfahrungen und „Lessons Learned“
- Ausblick

Das Projekt-Umfeld

- Strategisches Projekt eines jungen Lebensversicherers
- System zur Unterstützung der Kernprozesse im Kundenservice
 - Prozessplattform als Integrationsschicht über heterogene Systemlandschaft
 - Abbildung der Prozesse Neugeschäft, Kündigung, Vertragsänderung
 - Zentrales System zur Sachbearbeitung für diese Prozesse
- Gleichzeitige Neuausrichtung...
 - der Software-Architektur (objekt- und service-orientiert, OSGi-basiert)
 - der fachlichen Prozesse (Scannen vor Sachbearbeitung, hoch-integrierte Bearbeitung, Automatisierung von Routinetätigkeiten)

Zahlen, Daten, Fakten...

- Projektstart 04.2004
- 15-20-köpfiges Team
 - nur im Teilprojekt der Software-Entwicklung
- Erstes Release nach exakt 8 Monaten

- Seitdem:
 - 8 Releases
 - 600 User
 - Umsetzung neuer Prozesse in komplett neuer Systemlandschaft im Zuge eines Mergers
 - weiterhin strategisches Projekt

Ausgangslage

- Das Projekt lief ca. 1 Jahr „ohne“ einen expliziten Entwicklungsprozess
- Das Resultat:
 - Unsicherheit bei der Planung
 - Unklare Prioritäten
 - Termine wurden nicht mehr gehalten
 - Verunsicherung der Fachabteilung (des „Geldgebers“)
- Deshalb:
 - Ein **expliziter** und **disziplinierter Entwicklungsprozess** war nötig
 - Techniken aus Scrum, XP und „**The Eclipse-Way**“

„The Eclipse Way“

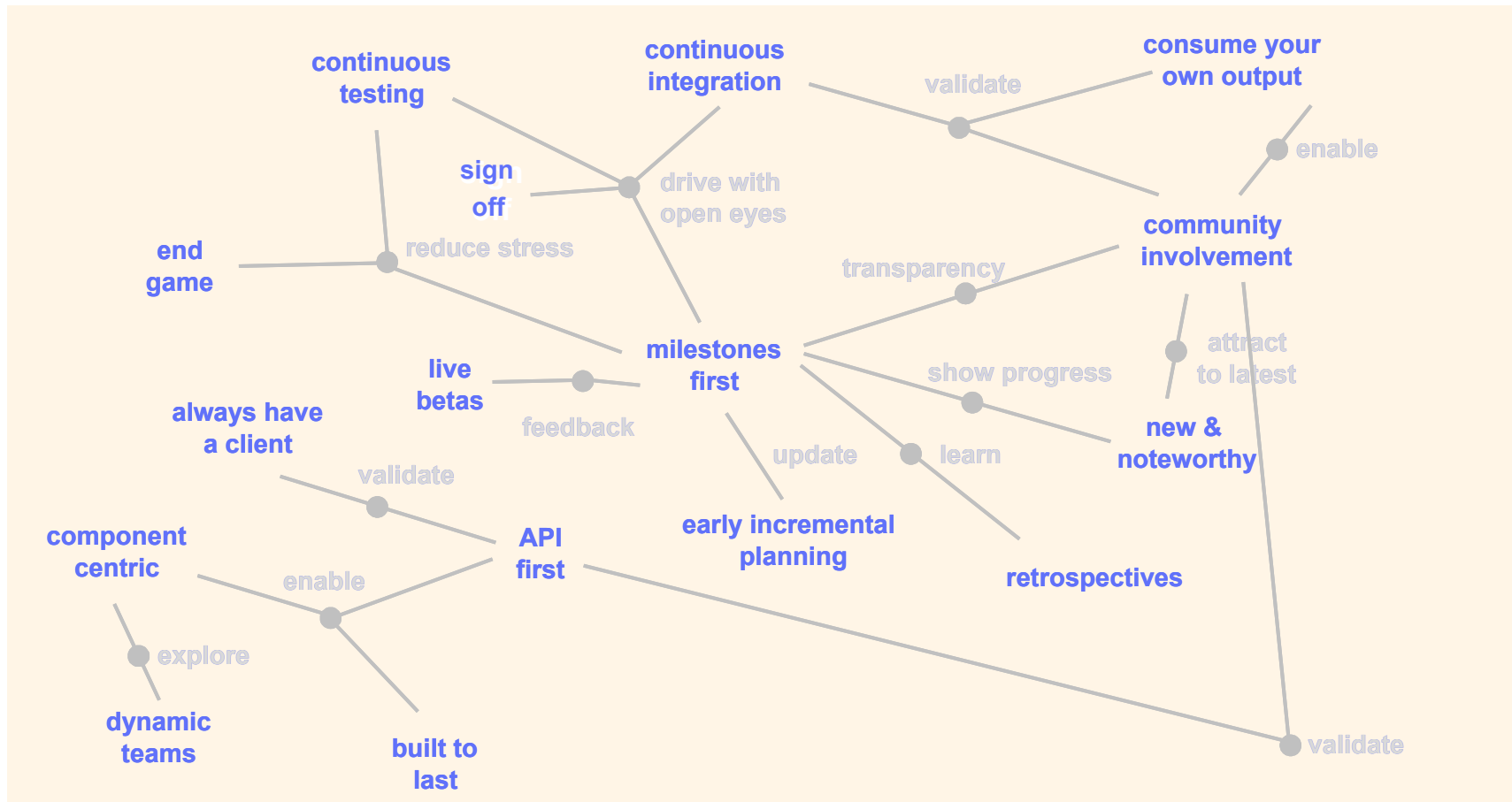
- Was ist „The Eclipse Way“?
 - Der Schatz an **agilen Entwicklungs-Praktiken** des Eclipse-SDK-Teams
 - Basiert auf grundsätzlichen **Werten**
 - Ist über die letzten sieben Jahre entwickelt worden
 - Im Zentrum steht: „**Shipping**“

- Beeindruckender Erfolg:
 - Über Jahre hinweg **jeden(!) Termin gehalten** (im 6-wöchigen Abstand)
 - **Qualitativ hochwertige Software** produziert
 - Wird mittlerweile von weiteren Projekten angenommen

„The Eclipse Way“: Grundsätzliches

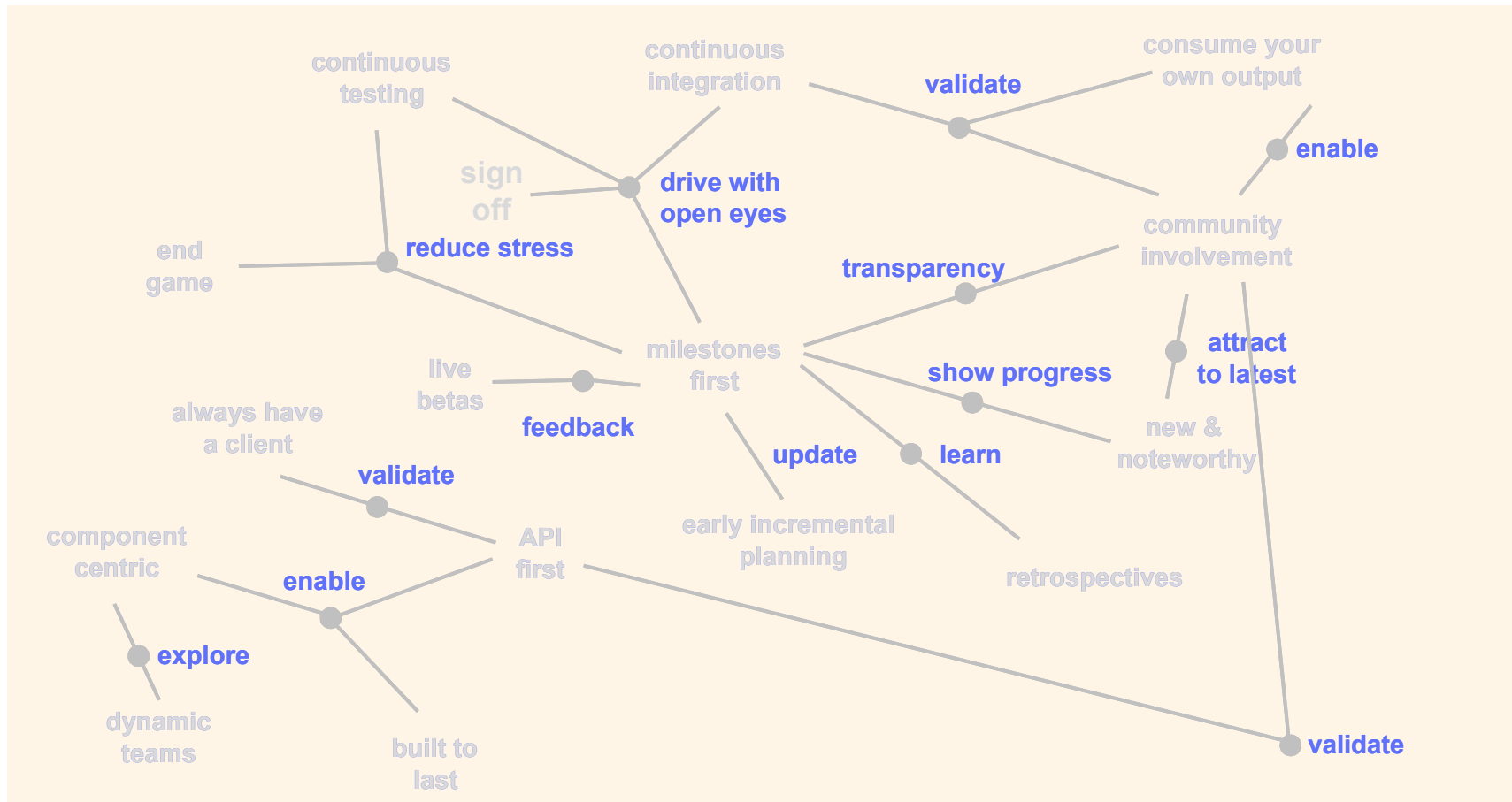
- **Quality:** “ship high-quality software”
- **Predictability:** “ship on time”
- **Transparency:** “no secrets about ship readiness”
- **Feedback:** “are we ready to ship?”

„The Eclipse Way“: Techniken



from: JAX 2005, The Eclipse Way – Part 1: The Eclipse Way explained, Tobias Widmer, Copyright by IBM

„The Eclipse Way“: Prinzipien

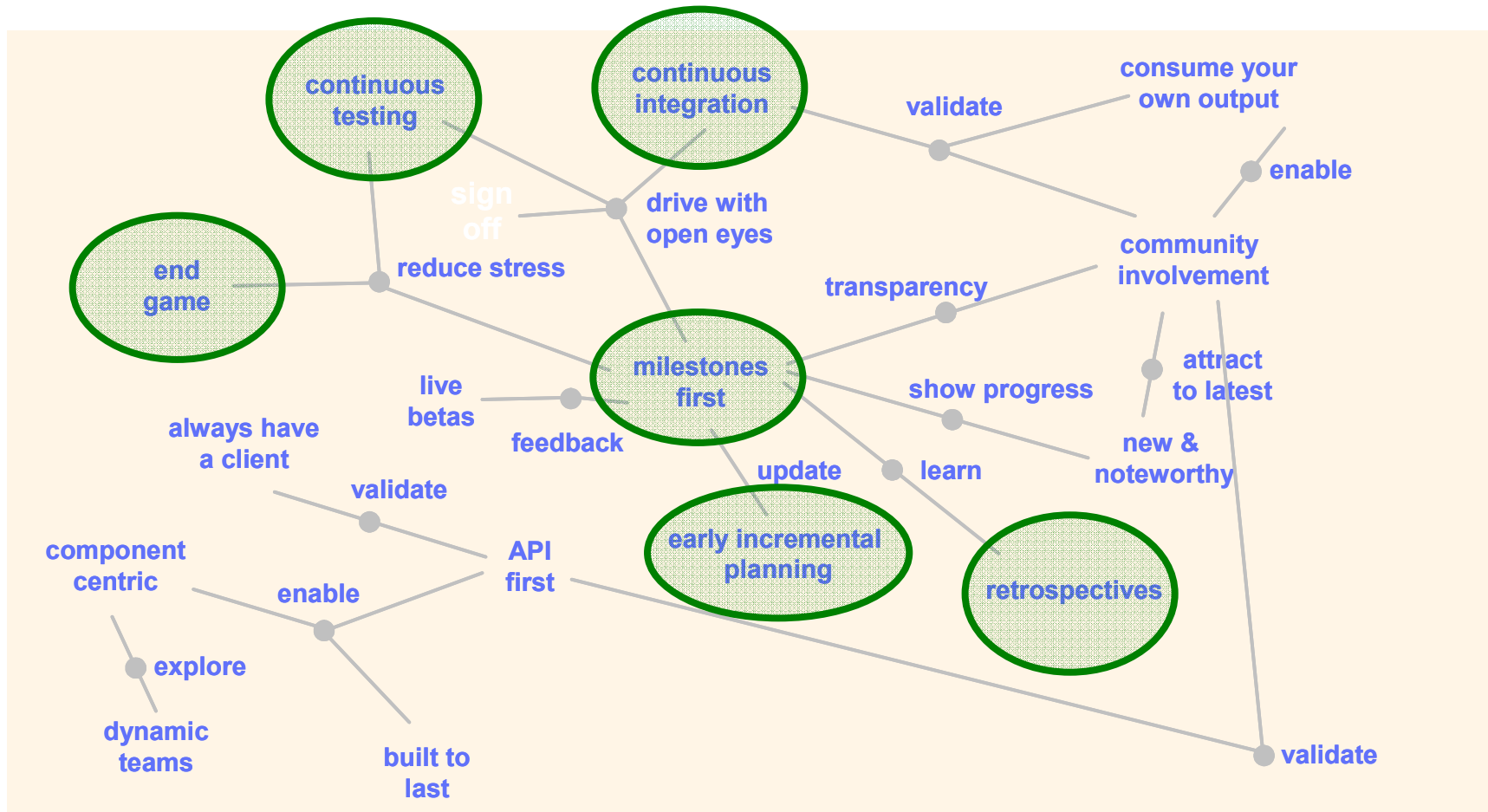


from: JAX 2005, *The Eclipse Way – Part 1: The Eclipse Way explained*, Tobias Widmer, Copyright by IBM

Warum „The Eclipse Way“?

- Harte, teilweise politische Deadlines
 - „ship on time“
- Kritische Unternehmens-Software
 - „ship high-quality software“
- Agile Entwicklung notwendig
 - Neuland, kein fertiges Konzept vorhanden
 - Enge Zeitpläne
- Gute Ausgangsbasis:
 - Team von hoch motivierten und sehr guten Entwicklern
 - Sehr kompetenter und entscheidungsfreudiger Fachbereich
 - Rückhalt aus dem Management

„Simple“ Things First...



Am Anfang stand die Retrospektive

- Wir sind mit einer Retrospektive gestartet
 - Den Prozess zum Thema machen
 - Aktiv über den Prozess nachdenken und offen zur Diskussion stellen
 - Offenheit und Bereitschaft zur Veränderung einfordern

- Ergebnis:
 - Feste Zyklen von 6 Wochen als Planungseinheit
 - Milestone-Builds zum Feedback durch den Fachbereich
 - Harte Steuerung durch die Fachseite
 - Tägliches Feedback über den Entwicklungsfortschritt

Nach den Milestones kommt das End-Game

- Ein Problem: Die Qualität des Systems
 - Hängt während des Entwicklungszyklus durch
 - Erzeugt Stress vor einem Release
 - Erzeugt Unzufriedenheit bei Testern und Anwendern
- Milestone Builds helfen
 - Verhindern das Durchhängen
 - Liefern früh Feedback
 - Fördern es, die Qualität durchgängig hoch zu halten
- Zusätzlich: **Das End-Game**

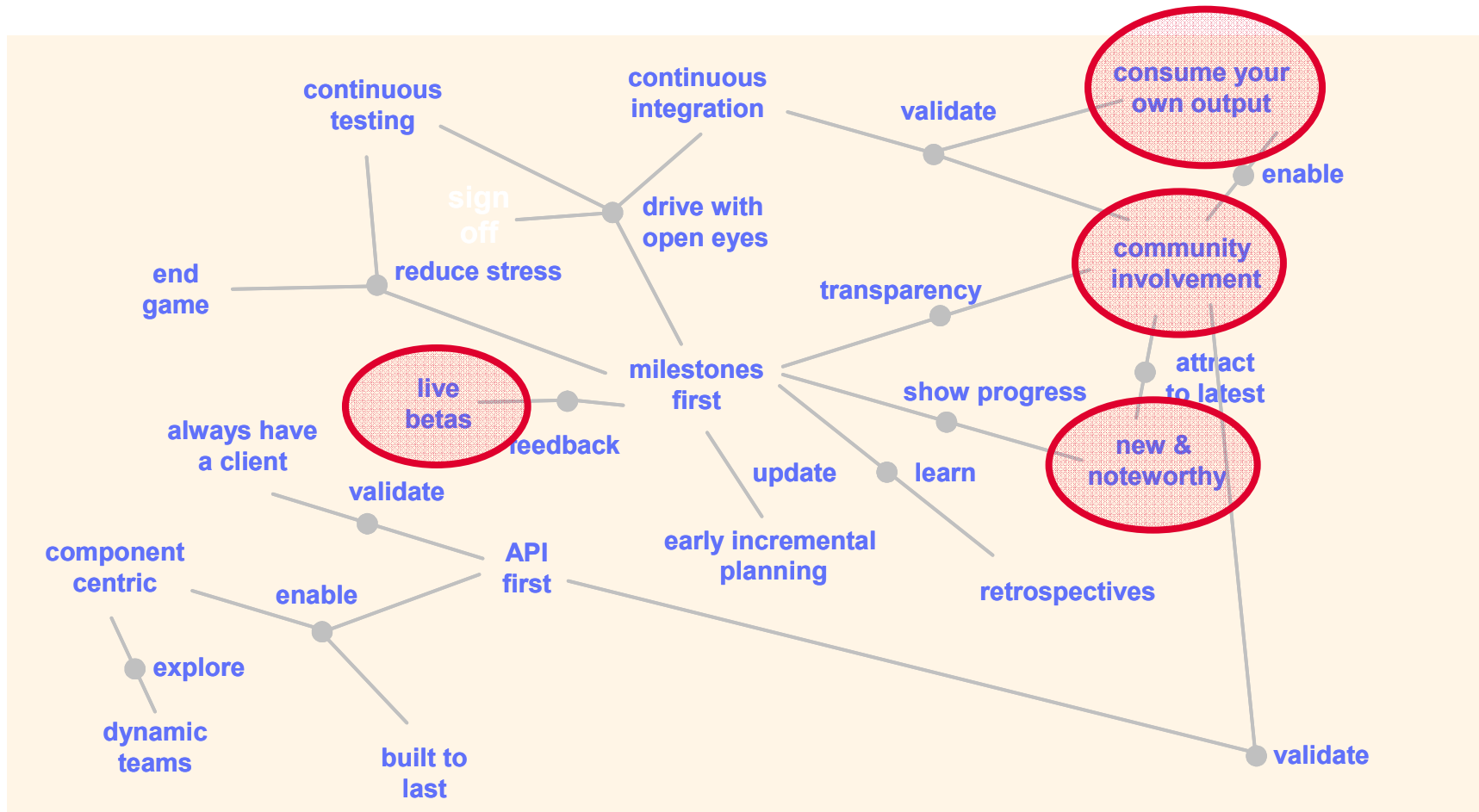
End-Game

- Das „End-Game“ bezeichnet die End-Phase vor einem Release
 - Alle arbeiten daran, die **Stabilität** und die **Qualität** des Systems zu erhöhen und ein sehr gutes Release auszuliefern
 - Es finden mehrere **Test- und Fix-Zyklen** statt (in hart definierten Zeiträumen)
 - „**Der Korridor wird enger**“ -> Stärkere Prozess-Regeln, je später ein Fix implementiert wird
 - Es werden **keine Features mehr** realisiert
- Sehr enge Zusammenarbeit mit dem Fachbereich
 - Täglich Prioritäten festlegen
 - Ständiges Feedback

Lessons Learned

- Retrospektiven sind unersetzlich, absolut elementar
- Milestone-Builds müssen ernst genommen werden
 - Möglichst produktionsnahes Testing durch den Anwender
 - Feedback einfordern und Ernst nehmen
- End-Game:
 - Erfordert viel Disziplin auf allen Seiten
 - Darf nicht zu einer heimlichen Feature-Iteration mutieren
 - Ist elementarer Bestandteil geworden
 - Sehr gute Erfahrungen (wenn diszipliniert durchgeführt)

Wer „A“ sagt...



Rund um die Community

- Das Eclipse-Team besitzt eine große Community
 - Aber das alleine hilft nicht automatisch
 - New & Noteworthy
 - Live Betas (produktiv einsetzbare Milestone Builds)
 - Feedback wird Ernst genommen

- Das Eclipse-Team ist „self-hosting“
 - Es verwendet seine eigene Software („Eat your own dog food“)
 - Wenn man Mist baut, tut es einem selbst sehr schnell weh

Consume Your Own Output

- Lässt sich so einfach nicht für Business-Anwendungen realisieren
 - Entwickler sind beispielsweise keine Sachbearbeiter

- Ansatz 1: Ausgewählte Anwender verwenden Milestone-Builds in Produktion
 - Guter Ansatz, wenn realisierbar
 - Oft schwierig, weil es schwer oder unmöglich ist, unterschiedliche Versionen in Produktion parallel zu betreiben

- Ansatz 2: Nähe herstellen
 - Echte Sachbearbeitung ins Team holen
 - Testing möglichst mit produktionsnahen Datenbeständen

Consume Your Own Output

- Ansatz 3: Produktionsbetreuung durch Entwickler-Team
 - Probleme in der Produktion schlagen schnell bei den Entwicklern auf
 - Sehr unmittelbares Feedback aus der Produktion
 - Viele Probleme führen zu Unterbrechungen in der täglichen Entwicklung und nerven die Entwickler

 - Das führt zu mehr Qualität und schnellerer Fehlerbehebung!!!

Lessons Learned

- **Letztendlich geht es um schnelles Feedback!!!**
- Mechanismen im Projekt etablieren, um schnell und ehrlich Feedback zu bekommen
 - Echte Anwender-Integration
 - Feedback ernst nehmen und Ergebnisse ausliefern
 - Ehrlich und frühzeitig den Stand der Dinge kommunizieren
 - Prioritäten der Anwender ernst nehmen

Grundsätzliche Voraussetzungen

- Unterstützung durch das Management
 - Freiräume
 - Ressourcen
- Unterstützung durch den Auftraggeber (Kunden)
 - Enge Einbindung der Kundenvertreter (Vollzeit-Teammitglieder)
 - Hohe Fachkompetenz
 - Schnelle Entscheidungen
- Bereitschaft aller Beteiligten, sich an die vereinbarten Regeln zu halten
 - Gemeinsame, transparente, verlässliche und verbindliche Planung
 - Hohe Disziplin

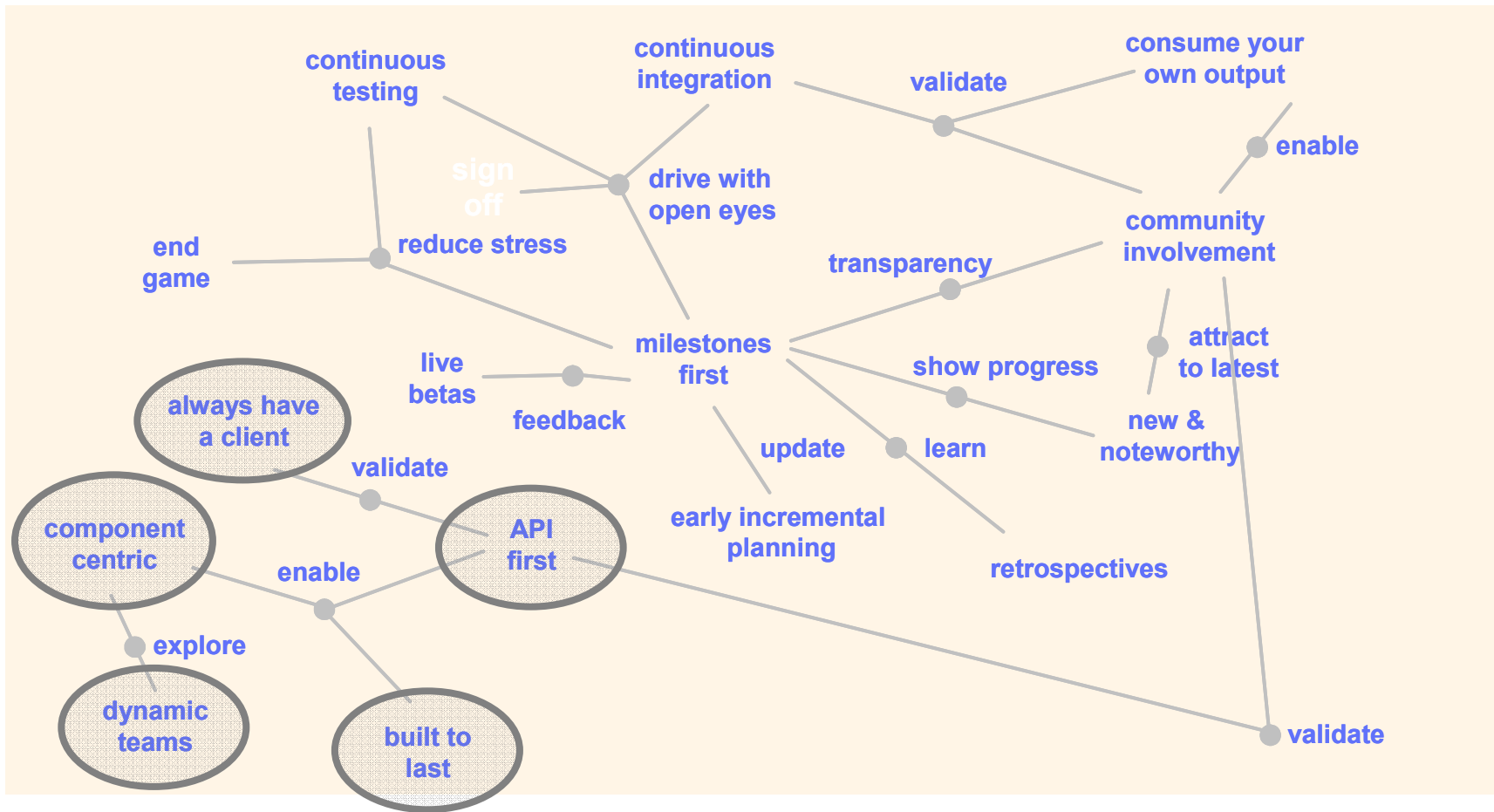
Gefahren

- Kommunikationsstörungen
 - Räumliche Entfernung
 - Erreichbarkeit
 - Persönliche Differenzen
 - Verborgene persönliche Ziele
- Unklare Rollen/Aufgaben
 - Zuständigkeiten
 - Verantwortung
- Unflexibles Umfeld
 - Weitgehend durch Management-Support auffangbar

Grenzen

- Unpassende Projekte
 - Zu klein
 - Zu einfach
 - Externe Entwicklung
- Unklare Ziele des Kunden
- Schleppende Entscheidungsfindung
- Unselbständige Entwickler

Last but not least...



Der Bau einer Plattform

- **Eine Plattform** für mehrere ähnliche und/oder hoch integrierte Anwendungen
 - Ähnlich wie Eclipse-RCP
 - Wesentlich spezifischer für Domäne
 - Enthält nicht nur UI-, sondern **auch Domain-Konzepte**

- **Erfahrungen:**
 - Realisiert für mittlerweile vier unterschiedliche Bereiche mit großem Erfolg
 - Anwendungen lassen sich schneller implementieren
 - Anwendungen werden homogener
 - Harmoniert gut mit einer SCA-Sichtweise für Unternehmen

Lessons Learned

- Die (Weiter-) Entwicklung einer Plattform ist nicht kostenlos
 - Höherer Entwicklungsaufwand
 - Mehr Gedanken über APIs unumgänglich
- Es können Seiteneffekte entstehen
 - Veränderungen an der Plattform wirken sich potentiell auf alle Anwendungen aus
- Eine klare Trennung schon bei der Entwicklung ist sinnvoll
 - Getrennte Workspaces
 - Fertig compilierte Plattform

Abschluss

- „The Eclipse Way“-Techniken eignen sich gut für In-House-Projekte
 - Viele Techniken auch bekannt von anderen agilen Methoden
 - Einige nützliche Ergänzungen (End-Game, Milestone-Rhythmus)
 - Zusätzliche Techniken gerade für größere und/oder verteilte Teams

- Der Prozess muss gelebt und gepflegt werden

- In Zukunft vielleicht mit mehr Tool-Support?
 - Heute Eclipse, Bugzilla, Wiki
 - Morgen vielleicht IBM Jazz ?

Vielen Dank für die Aufmerksamkeit

Fragen jederzeit herzlich willkommen.

Besuchen Sie uns auf dem it-agile-Stand!!!

ivo.eitner@arcor.de, martin.lippert@it-agile.de



Schulung verlängerte Werkbank
agile Softwareentwicklung
Festpreisprojekte Coaching
RCP **Systemintegration** Eclipse
h3270 Hostintegration
Scrum Refactoring testgetriebene Entwicklung
Hibernate SAP-Netweaver **OpenSource**
Ajax JBoss/JEMS Groovy
 **eXtreme Programming**