DEAD END

Anti-Pattern

Sackgasse

von Sebastian Leopold, Jens Kleehammer

28.10.2014 - TINF12B4 - Angewandte Informatik - Software Engineering 2

DEAD END

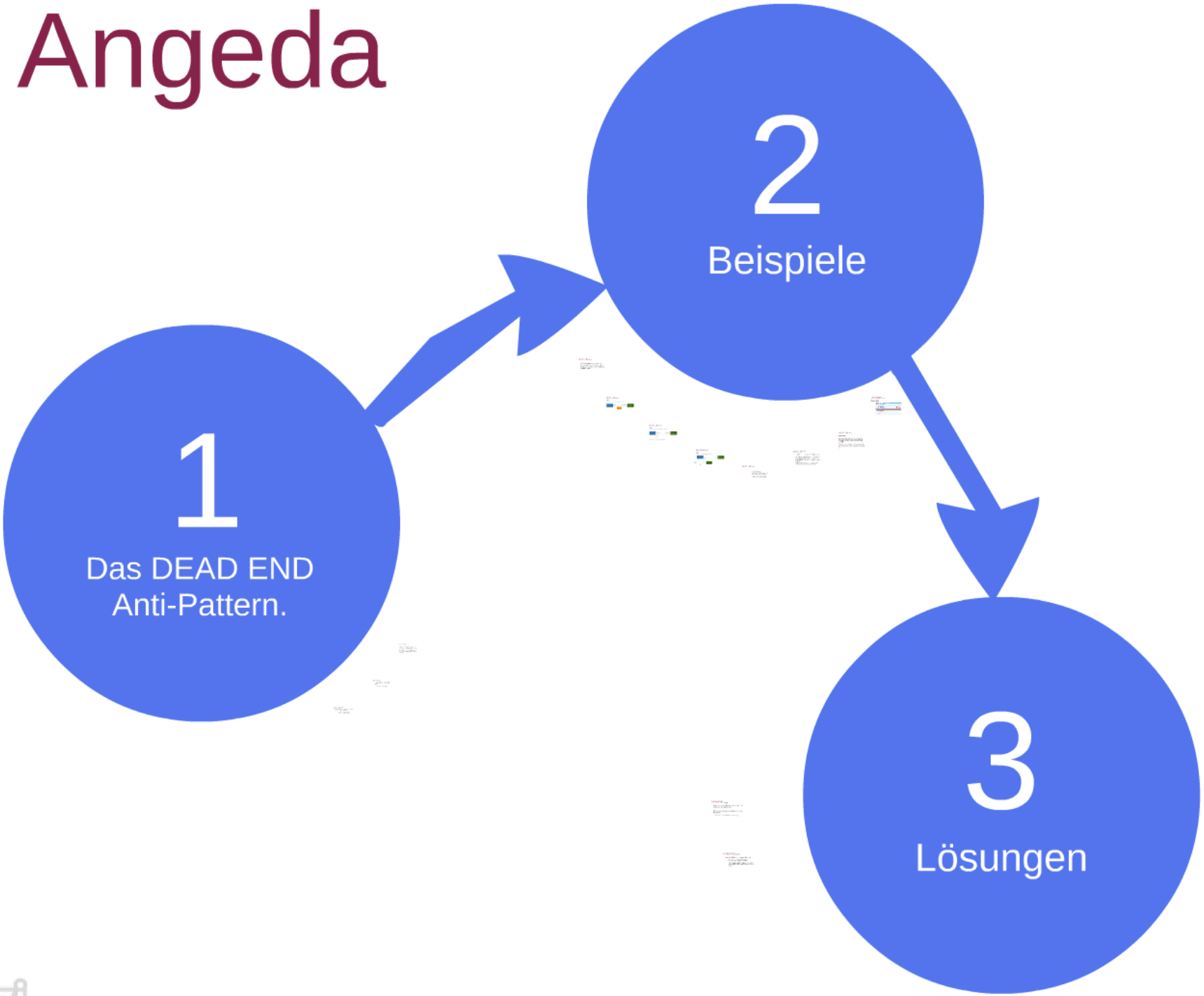
Anti-Pattern

Sackgasse

von Sebastian Leopold, Jens Kleehammer

28.10.2014 - TINF12B4 - Angewandte Informatik - Software Engineering 2

Angeda



1

Das DEAD END Anti-Pattern.

DEAD END ANTI-PATTERN

Die Modifikation einer Komponente die von einem anderen Hersteller erstellt wurde.

→ Massiver Wartungsaufwand

DEAD END ANTI-PATTERN

Wird oft als Workaround für fehlende Features oder bestehende Fehler in der Komponente eingesetzt.

Hilft nur Kurzzeitig!

DEAD END ANTI-PATTERN

Funktioniert nur zufällig wenn die Ziele des Herstellers und die eigenen die selben sind.

Nur dann werden die Modifikationen vom Hersteller der Komponente in sein Produkt integriert.

2

Beispiele

DEAD END

BEISPIELE

Bei einem Update der Komponente müssen eigene Änderungen angepasst und in die neue Version der Komponente integriert werden.

DEAD END **BEISPIELE**

Ablauf:

1. Update der Komponente (ohne Schnittstellenänderung der Komponente)

Update der Komponente
(Behebt in der Regel Bugs oder implementiert neue Features)

wird integriert

Eigene Code-Basis:

Eine Anpassung der eigenen Software ist in der Regel nicht notwendig, da sich keine Schnittstelle geändert hat.

wird veröffentlicht

Neue Softwareversion mit aktualisierter Komponete
(Alle Fehlerkorrekturen der Komponente sowie neue Features sind auch in der eigenen Software enthalten.)

DEAD END BEISPIELE

Ablauf:

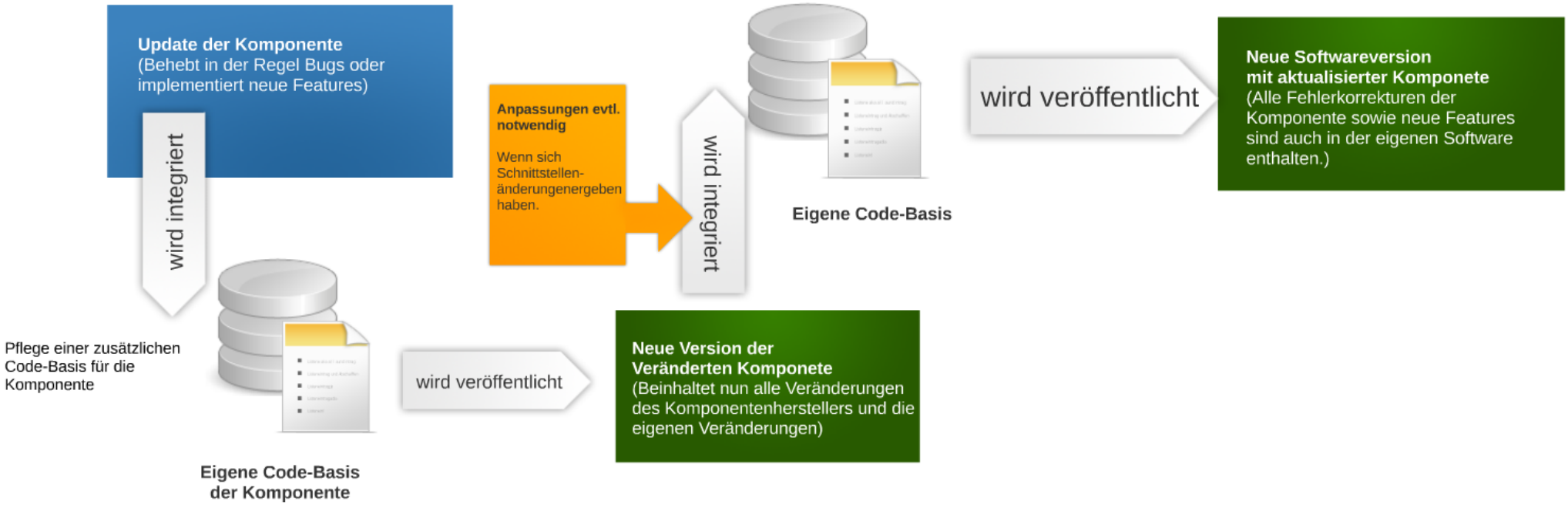
2. Update der Komponente (mit Schnittstellenänderung der Komponente)

3. Update der Komponente (eigene Veränderungen vorhanden)

DEAD END BEISPIELE

Ablauf:

3. Update der Komponente (eigene Veränderungen vorhanden)



DEAD END

BEISPIELE

Größtes Problem:
Wenn eigne Veränderungen nicht mehr Integriert werden können:

-> DEAD END / SACKGASSE

DEAD END BEISPIELE

2. Einstellung der Entwicklung einer Komponente.

- Der Wechsel zu einer anderen Komponente ist schwierig da Anpassungen am Code vorgenommen wurden um die eigenen Ziele zu erreichen.
- Alle Fehlerkorrekturen an der Komponenten müssen nun selbst Durchgeführt werden.

DEAD END BEISPIELE

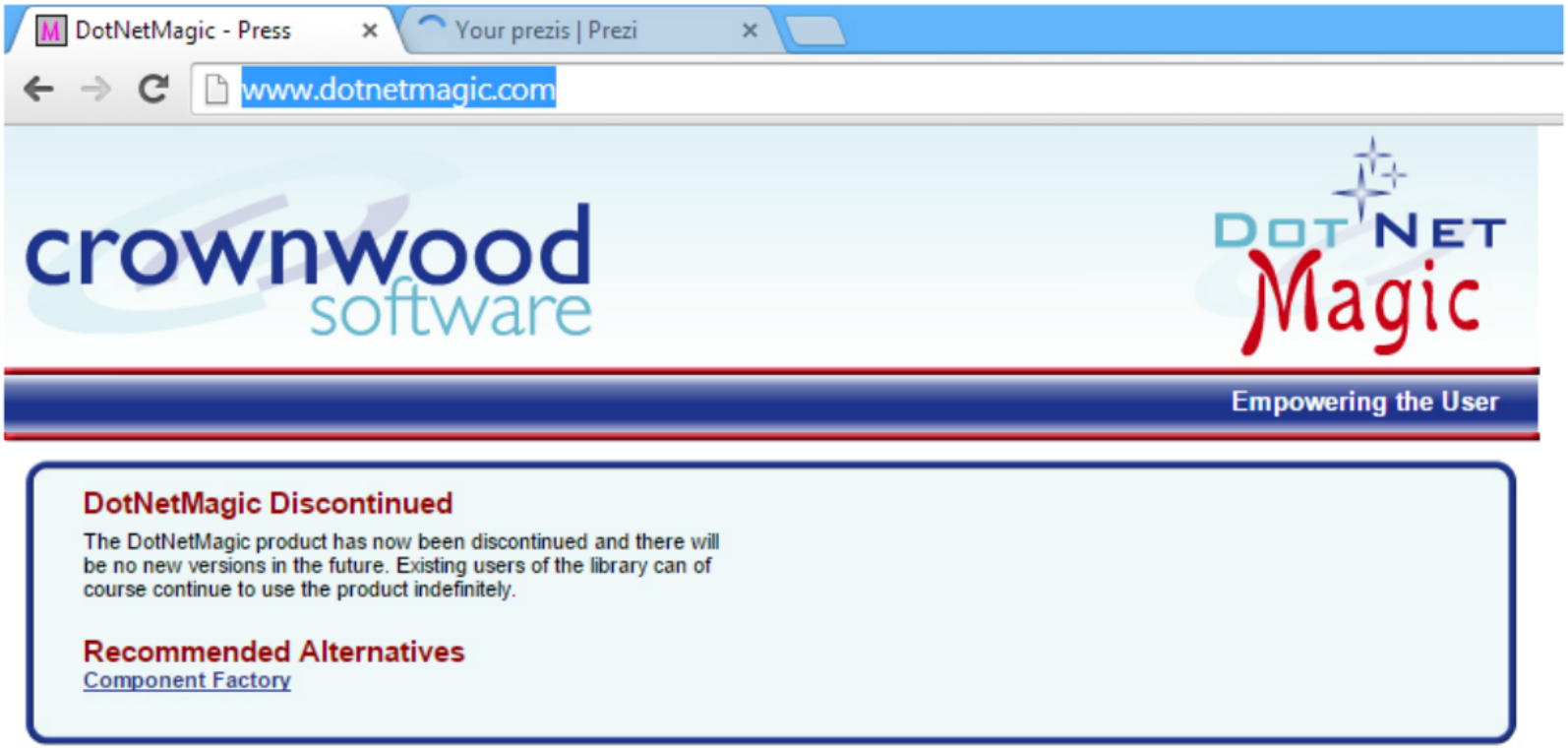
Realer Fall:

Die Entwicklung und Wartung der GUI-Komponente DotNetMagic für Windows Forms wurde aufgrund der schlechten Code-Basis vom Entwickler (Phil Wright) eingestellt.

Er gründete anschließend eine neue Firma, welche die gleichen Komponenten anbietet (auf einer besseren Code-Basis), die jedoch nicht zur alten Komponente kompatibel ist.

DEAD END BEISPIELE

Realer Fall:



3

Lösungen

DEAD END LÖSUNGEN

Vermeiden von Modifizierungen an zugekaufter Software oder Komponenten.

Einsetzen von Mainstream Komponenten und Plattformen.

Beispiel: .NET Framework, Java-Laufzeitumgebung

DEAD END LÖSUNGEN

Wenn Modifikationen unvermeidbar sind:

- Einsatz eines ISOLATION Layers.

  Trennung von eigenen Anpassungen und der Komponente. Die Komponente austauschbar halten.