Fabian Schmengler,
Dr. Nikolai Krambrock

Herausforderungen bei Konzeption und Test von Magento-Modulen

Wechsel von C# zu Magento

- Ursprünglich Entwicklung von Thick-Client Applikationen mit C#
- Vollständiger Wechsel auf Magento innerhalb von zwei Jahren
- Neuer Herausforderungen im (testgetriebenen) Entwurf und im (automatisierten) Testen

Forschungsarbeit durchgeführt

- Typische Probleme bei der Entwicklung von Magento-Modulen analysiert (Interviews)
- Maßnahmen aus Praxis und Literatur sammeln, neu entwickelt und bewertet
- Dazu passende Werkzeuge festgehalten und Best Practices formuliert

Ergebnis unter code4business.de/carbon

Agenda

- Probleme bei Magento-Entwicklung
- Maßnahmen gegen die Probleme
- Maßnahmen vorgestellt
  - UML-Magento
  - Akzeptanztests abstrahieren
  - Smoke-Test-Suite
- Best Practices

Ergebnisse der Forschungsarbeit

PROBLEME DER MAGENTO-ENTWICKLUNG

Analsyierte Probleme bei Magento-Entwicklung

Analyse | **Entwurf** | Entwicklung | **Test** | Betrieb

- Entwurfsmethoden versagen (Entwurf)
- Wissensweitergabe erschwert (Entwurf)
- Automatisierte Tests sind weniger effektiv (Test)

1. Entwurfs-Methoden

- Entwurf mit UML: zu aufgeblasen, wichtige Elemente nicht darstellbar
- Test Driven Development: setzt Unit-testbaren Framework-Code voraus
- Programming by Contract: Notwendige Informationen fehlen

2. Weitergabe von Know-How

- Wissensvermittlung läuft aktuell am Besten über Schulungen und „Danebensitzen“
- Unvollständige Coding Standards
- Ressourcen im Web verstreut

3. Automatisierte Tests

- Funktionale Tests: Langsam, schlechte Wiederverwendbarkeit
- Unit-Tests: Umständliche Setups, Interaktion mit Framework schwer testbar

Übersicht Typische Probleme

Nummer	Problem	Tätigkeit
P1	Ungenaue oder widersprüchliche Anforderungen	Analyse
P2	Zusätzlicher Recherche-Schritt	Analyse
P3	Zeitliche Aufwandschätzung falsch	Entwurf
P4	Formaler Entwurf ist erschwert	Entwurf
P5	Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft	Entwurf
P6	Anlernen dauert lange	Codierung
P7	Debugger und/oder Versionskontrolle wird nicht verwendet	Codierung
P8	Geringe Code-Qualität	Codierung
P9	Unerwartete Seiteneffekte	Codierung, Test
P10	Viele Iterationen zwischen Codierung und Test	Codierung, Test
P11	Automatisierte funktionale Tests sind aufwändig	Test
P12	Unit Tests sind aufwändig und ineffektiv	Test
P13	Geringe Code-Qualität von Fremd-Extensions	Integration
P14	Mehrere Schleifen zwischen Layoutern, Templatern und Entwicklern	Integration
P15	Für Entwickler aufwändige Einarbeitung in Werkzeuge	Integration
P16	Fehleranfälliges Deployment	Integration
P17	Hinterhof-Image	Allgemein
P18	Viele Change Requests	Allgemein

Ergebnisse der Forschungsarbeit

MAßNAHMEN BEI DER MAGENTO-ENTWICKLUNG

Maßnahmen für die Magento-Entwicklung

- 50 Maßnahmen aus Interviews, Literatur und eigenen Tests identifiziert
- Probleme und Maßnahmen gegenübergestellt
  (Im Web verfügbar)
- Neu entwickelte Methoden evaluiert
- Best-Practice-Katalog entwickelt
  (Web-Version noch im Aufbau)

Übersicht Maßnahmen (1/3)

Maßnahme	Adressierte Probleme
M1 Schulung des Kunden / Product Owners	P1 Ungenaue oder widersprüchliche Anforderungen, P18 Viele Change Requests
M2 Einbeziehung des Kunden in den Entwicklungsprozess	P1 Ungenaue oder widersprüchliche Anforderungen, P18 Viele Change Requests
M3 Abstrakte Schätzung mit Story Points und Team Velocity	P3 Zeitliche Aufwandschätzung falsch
M4 Standards selber dokumentieren (basierend auf Reverse Engineering und Erfahrung)	P2 Zusätzlicher Recherche-Schritt, P3 Zeitliche Aufwandschätzung falsch, P5 Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft
M5 Schulungen	P6 Anlernen dauert lange
M6 Negativbeispiele	P6 Anlernen dauert lange
M7 Bei der Einstellung berücksichtigen: Erfahrung mit C#, Java u.ä. hilfreicher als mit PHP	P6 Anlernen dauert lange, P15 Für Entwickler aufwändige Einarbeitung in Werkzeuge
M8 Vorhandene Werkzeuge verwenden	P7 Debugger und/oder Versionskontrolle wird nicht verwendet, P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung, P9 Unerwartete Seiteneffekte, P16 Fehleranfälliges Deployment
M9 Disziplin und Entwickler-Anspruch	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung, P13 Geringe Code-Qualität von Fremd-Extensions
M10 Code Reviews	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M11 Automatisierte Tests	P9 Unerwartete Seiteneffekte
M12 Selenium-Tests von Programmierern erstellen lassen	P11 Automatisierte funktionale Tests sind aufwändig
M13 Review von Fremd-Extensions	P13 Geringe Code-Qualität von Fremd-Extensions
M14 Eigenentwicklungen nach Möglichkeit vorziehen	P13 Geringe Code-Qualität von Fremd-Extensions
M15 Abhängigkeiten zwischen Modulen und Templates vermeiden	P14 Mehrere Schleifen zwischen Layoutern, Templatern und Entwicklern

Übersicht Maßnahmen (2/3)

Maßnahme	Adressierte Probleme
M16 Alternative Entwürfe evaluieren	P3 Zeitliche Aufwandschätzung falsch, P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M17 Schätzungen hinterfragen	
M18 Pair Programming	P5 Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft, P6 Anlernen dauert lange, P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung, P10 Viele Iterationen zwischen Codierung und Test
M19 Mehr Zeitdruck	
M20 "aufpassen und nachschauen"	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M21 Abrechnung nach Aufwand	P18 Viele Change Requests
M22 Kein vorgefertigtes finales Anforderungsdokument, transparent kontinuierliche Änderung der Spezifikation	P11 Automatisierte funktionale Tests sind aufwändig, P18 Viele Change Requests
M23 Entwicklungs-, Test- und Live-Systeme synchronisieren	P11 Automatisierte funktionale Tests sind aufwändig
M24 Manuelle Unit Tests	P12 Unit Tests sind aufwändig und ineffektiv, P13 Geringe Code-Qualität von Fremd-Extensions, P16 Fehleranfälliges Deployment
M25 Konfigurations-Änderungen der Anwendung nur über Update-Scripts	P2 Zusätzlicher Recherche-Schritt, P9 Unerwartete Seiteneffekte
M26 Informiert bleiben über aktuelle Entwicklungen, auch in der Community	P16 Fehleranfälliges Deployment
M27 Ein Team aus Templatern (Frontend) und Entwicklern (Backend), an Blockern wird gemeinsam gearbeitet (Scrum-Sirene)	P14 Mehrere Schleifen zwischen Layoutern, Templatern und Entwicklern
M30 Framework-Dokumentation	P2 Zusätzlicher Recherche-Schritt, P4 Formaler Entwurf ist erschwert, P5 Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft

Übersicht Maßnahmen (3/3)

Maßnahme	Adressierte Probleme
M31 Reverse Engineering	P2 Zusätzlicher Recherche-Schritt, P4 Formaler Entwurf ist erschwert, P5 Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft
M32 Pattern Detection	
M33 Visuelle Modellierung mit UML	
M34 Visuelle Modellierung mit Magento-UML	P2 Zusätzlicher Recherche-Schritt, P4 Formaler Entwurf ist erschwert, P5 Know-How von erfahrenen Entwicklern wird nicht ausgeschöpft
M35 FDD-Prozess	
M36 Testgetriebene Entwicklung (TDD)	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung, P10 Viele Iterationen zwischen Codierung und Test
M37 Einfacher Entwurf	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M38 Programming by Contract	P4 Formaler Entwurf ist erschwert, P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M39 Coding Guidelines	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M40 SOLID Prinzipien	P8 Geringe Code-Qualität (insb. Wartbarkeit und Zuverlässigkeit) Codierung
M41 Testfall-Generatoren	P11 Automatisierte funktionale Tests sind aufwändig
M42 Built-in Tests	P12 Unit Tests sind aufwändig und ineffektiv
M44 Integrierte White-box Tests	
M45 Test-Automatisierung systematisch entscheiden	
M46 Smoke Test Suite	P11 Automatisierte funktionale Tests sind aufwändig
M47 Mutation Testing	P11 Automatisierte funktionale Tests sind aufwändig, P12 Unit Tests sind aufwändig und ineffektiv
M48 Modellbasiertes Testen	P11 Automatisierte funktionale Tests sind aufwändig
M49 Abstraktion von funktionalen Tests, automatisierte Akzeptanztests	P11 Automatisierte funktionale Tests sind aufwändig
M50 Kompatibilitätsanalyse mit FDMM	

Probleme vs. Maßnahmen

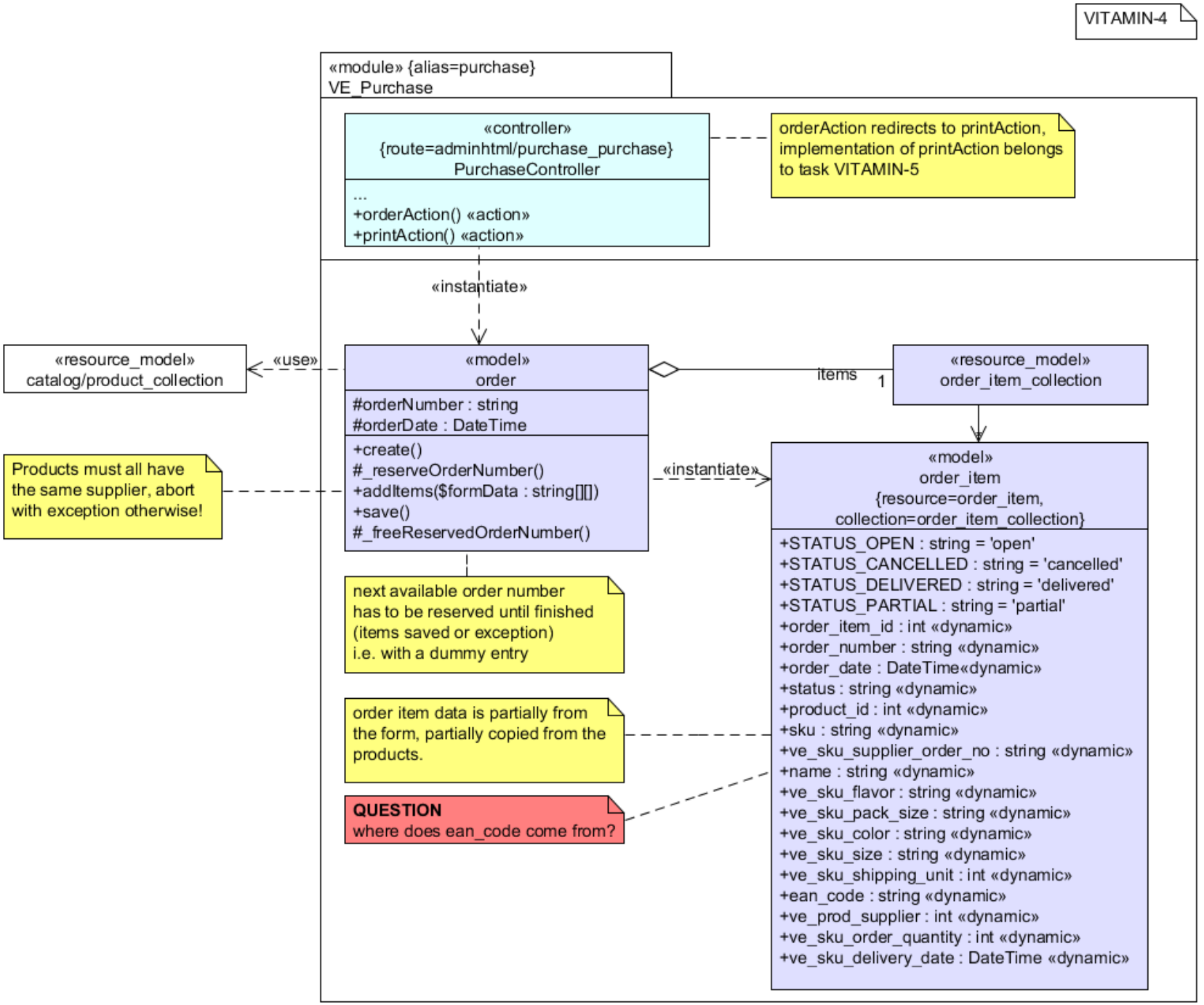
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18
M1 Schulung des Kunden / Product Owners	•																	•
M2 Einbeziehung des Kunden in den Entwicklungsprozess	•																	•
M3 Abstrakte Schätzung mit Story Points und Team Velocity			•															
M4 Standards selber dokumentieren		•			•													
M5 Schulungen						•												
M6 Negativbeispiele						•												
M7 Erfahrung mit C#, Java u.ä. hilfreicher als mit PHP						•									•			
M8 Vorhandene Werkzeuge verwenden							•	•	•				•			•		
M9 Disziplin und Entwickler-Anspruch								•										
M10 Code Reviews								•										
M11 Automatisierte Tests									•									
M12 Selenium-Tests von Programmierern erstellen lassen											•							
M13 Review von Fremd-Extensions													•					
M14 Eigenentwicklungen nach Möglichkeit vorziehen													•					
M15 Abhängigkeiten zwischen Modulen und Templates vermeiden														•				
M16 Alternative Entwürfe evaluieren			•					•										
M17 Schätzungen hinterfragen																		
M18 Pair Programming					•	•		•		•								
M19 Mehr Zeitdruck																		
M20 "aufpassen und nachschauen"								•										
M21 Abrechnung nach Aufwand																		•
M22 Kein vorgefertigtes finales Anforderungsdokument																		•
M23 Entwicklungs-, Test- und Live-Systeme synchronisieren											•					•		
M24 Manuelle Unit Tests												•						
M25 Konfigurations-Änderungen der Anwendung nur über Update-Scripts									•							•		
M26 Informiert bleiben über aktuelle Entwicklungen, auch in der Community		•																
M27 Ein Team aus Templatern (Frontend) und Entwicklern (Backend)														•				
M28 "Technical Debt"-Philosophie: Refactoring, Clean Code								•										
M29 Inkrementeller Entwurf				•														
M30 Framework-Dokumentation		•		•	•													
M31 Reverse Engineering		•		•	•													
M32 Pattern Detection																		
M33 Visuelle Modellierung mit UML																		
M34 Visuelle Modellierung mit Magento-UML		•		•	•													
M35 FDD-Prozess					•													
M36 Testgetriebene Entwicklung (TDD)								•		•								
M37 Einfacher Entwurf								•										
M38 Programming by Contract				•				•										
M39 Coding Guidelines								•										
M40 SOLID Prinzipien								•										
M41 Testfall-Generatoren											•							
M42 Built-in Tests												•						
M43 Assertion-basierte Tests																		
M44 Integrierte White-box Tests																		
M45 Test-Automatisierung systematisch entscheiden																		
M46 Smoke Test Suite											•							
M47 Mutation Testing											•	•						
M48 Modellbasiertes Testen											•							
M49 Abstraktion von funktionalen Tests, automatisierte Akzeptanztests											•							
M50 Kompatibilitätsanalyse mit FDMM																		

Beispielhafte Anwendung

METHODEN FÜR ENTWURF UND TEST

UML-Magento

- UML 2.0 Profil *UML-Magento*
  - Abstrahiert Implementierungs-Details des Frameworks
  - Fördert prägnante Diagramme
  - Erweitert um Namenskonventionen

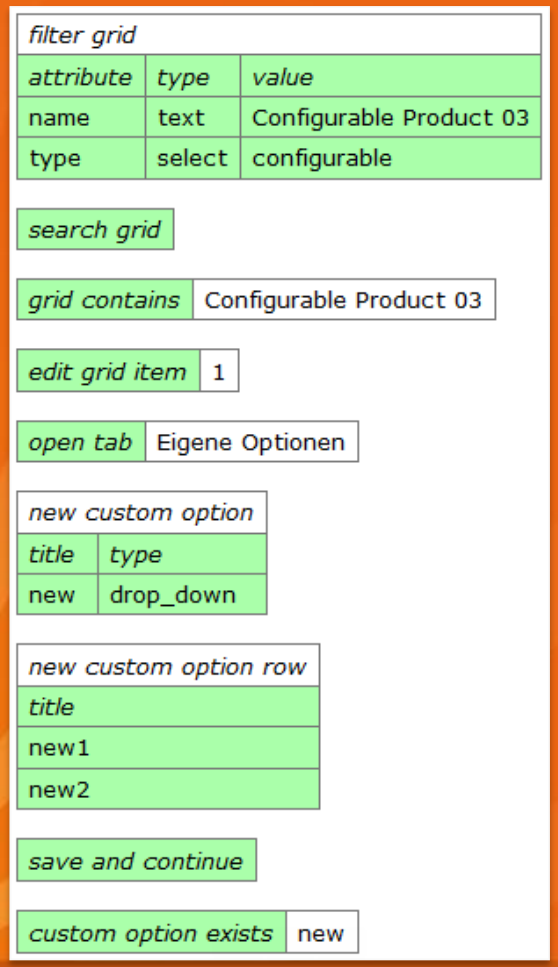

VITAMIN-4
«module» {alias=purchase}
VE_Purchase
«controller»
{route=adminhtml/purchase_purchase}
PurchaseController
...
+orderAction() «action»
+printAction() «action»
orderAction redirects to printAction, implementation of printAction belongs to task VITAMIN-5
«instantiate»
«resource_model»
catalog/product_collection
«use»
«model»
order
#orderNumber : string
#orderDate : DateTime
+create()
#_reserveOrderNumber()
+addItems($formData : string[][])
+save()
#_freeReservedOrderNumber()
items
1
«resource_model»
order_item_collection
«instantiate»
«model»
order_item
{resource=order_item,
collection=order_item_collection}
+STATUS_OPEN : string = 'open'
+STATUS_CANCELLED : string = 'cancelled'
+STATUS_DELIVERED : string = 'delivered'
+STATUS_PARTIAL : string = 'partial'
+order_item_id : int «dynamic»
+order_number : string «dynamic»
+order_date : DateTime«dynamic»
+status : string «dynamic»
+product_id : int «dynamic»
+sku : string «dynamic»
+ve_sku_supplier_order_no : string «dynamic»
+name : string «dynamic»
+ve_sku_flavor : string «dynamic»
+ve_sku_pack_size : string «dynamic»
+ve_sku_color : string «dynamic»
+ve_sku_size : string «dynamic»
+ve_sku_shipping_unit : int «dynamic»
+ean_code : string «dynamic»
+ve_prod_supplier : int «dynamic»
+ve_sku_order_quantity : int «dynamic»
+ve_sku_delivery_date : DateTime «dynamic»
Products must all have the same supplier, abort with exception otherwise!
next available order number has to be reserved until finished (items saved or exception) i.e. with a dummy entry
order item data is partially from the form, partially copied from the products.
QUESTION
where does ean_code come from?

UML-Magento: Profil

- Ausgelegt für Klassendiagramme
- Aktuell 20 neue Stereotypen, für
  - Observers
  - Models, Collections, EAV-Attribute
  - Controller Actions
  - Dynamische Attribute und Methoden
  - Konfigurations-Werte
  - Blocks, Templates

Methode: Akzeptanztests abstrahieren

- Test Cases werden auf höherer Abstraktionsebene geschrieben
- Verständlich für Nicht-Entwickler
- Unabhängig von UI-Details
- Test und Implementierung der Test-Aktionen getrennt



Defined Actions

login	user	*with password*	pass

get url	@{baseurl}/customer/account/login/		
with	//input[@id="email"]	*set text*	@{user}
with	//input[@id="pass"]	*set text*	@{pass}
submit	//form[@id="login-form"]		
page source	**contains**	Hallo	

custom option exists	title

checking timeout	1000	
element	//input[@value='@{title}']	exists
checking timeout	0	

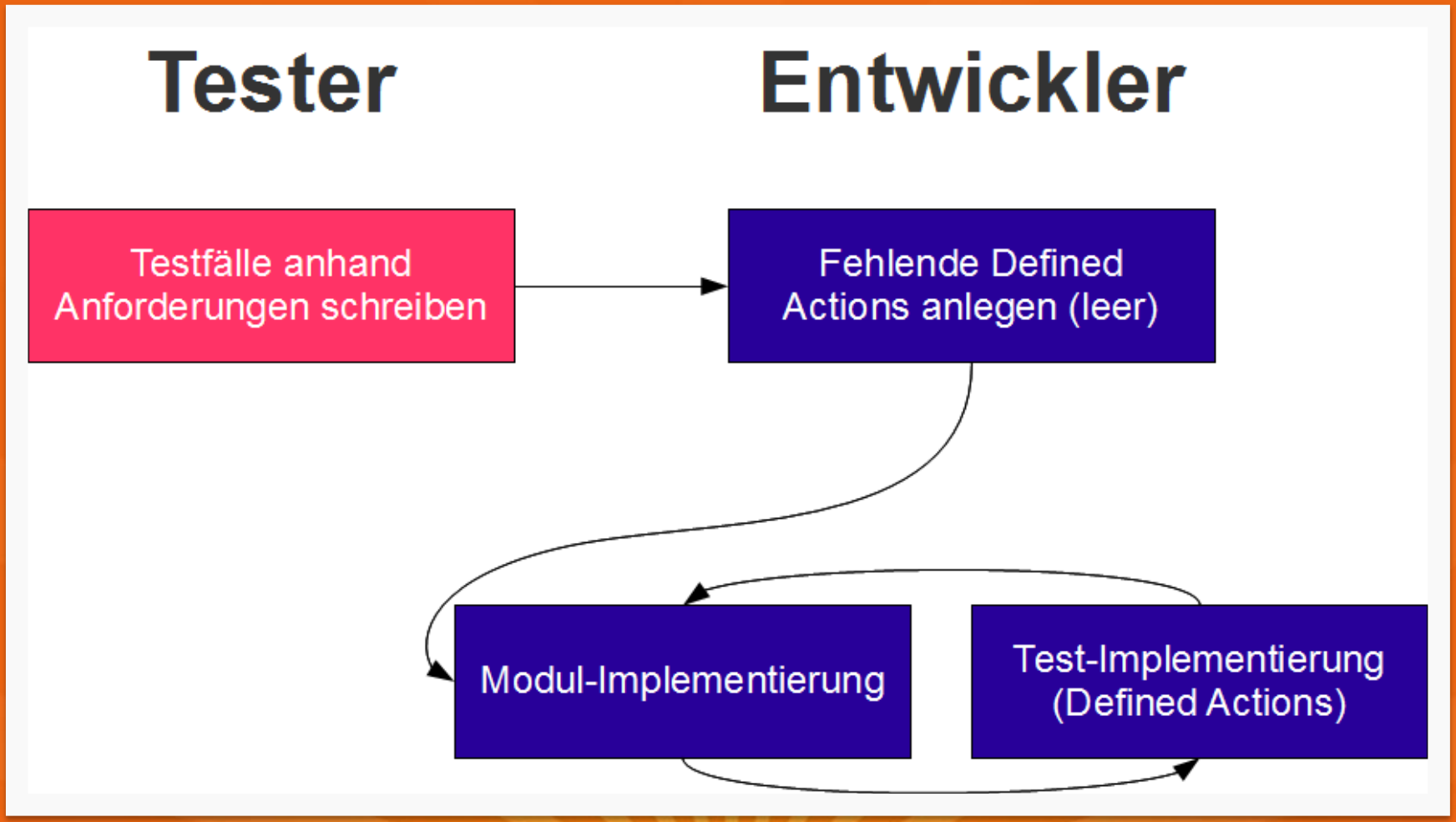
- Implementierung der Tests
- Auch als Tabellen im Wiki
- Organisierbar:

  Projektübergreifend, projektspezifisch, testspezifisch

Toolchain

- FIT – Framework for Integrated Test
  - Tabellenbasiertes Testwerkzeug
- FitLibrary
  - Schnittstelle zu Selenium WebDriver u.v.m.
- FitNesse
  - Wiki-Frontend für FIT
- Selenium WebDriver
  - Browser-Steuerung

Funktionale Tests: Prozess



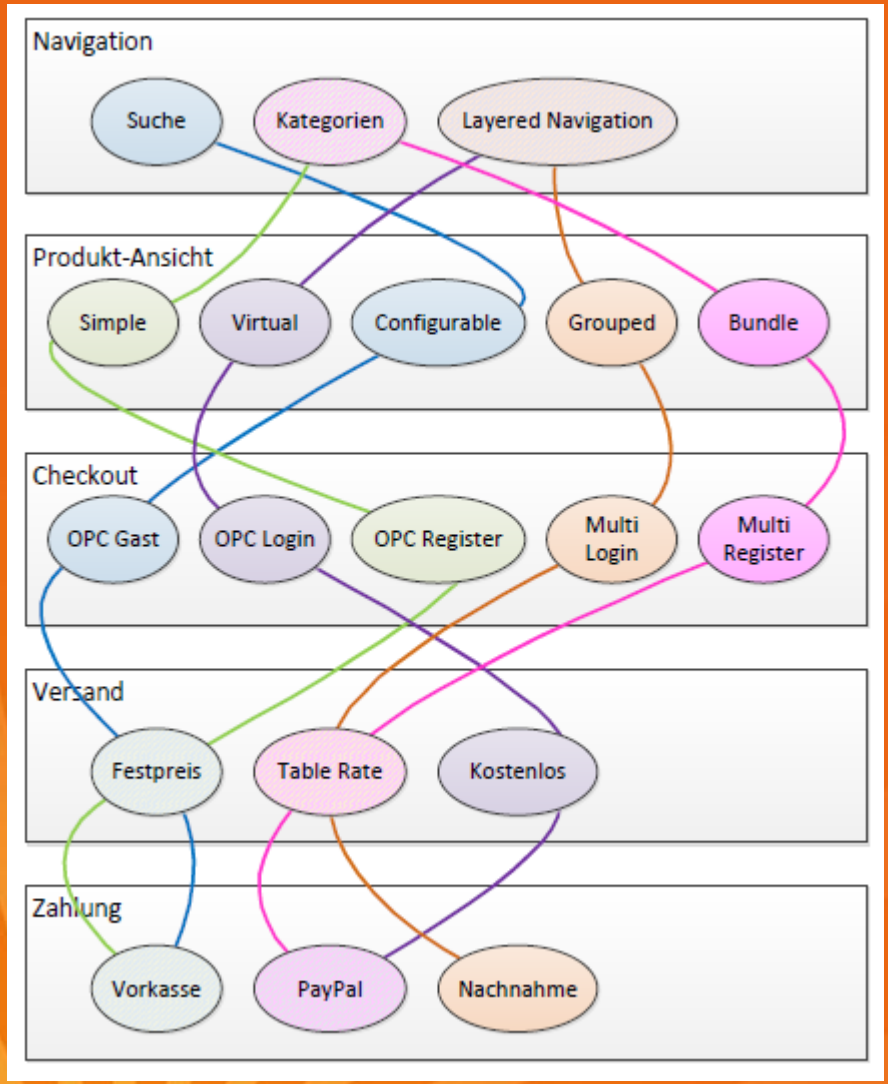
Smoke Test Suite

- Schnelles Auffinden von schwerwiegenden Fehlern
- Zweigabdeckung in Magento Code
- Regressionstest
- Monitoring
- Werkzeuge wie für Akzeptanztests

Foto: flickr, „tbone_sandwich“
CC BY-NC-SA 2.0

Smoke Test Beispiel

- Kernfunktionen des Shops
- Typische Klickpfade
- Reibungsloser Durchlauf



Beispielhafte Anwendung

BEST PRACTICES

Problem: Viele Change Requests

- Maßnahmen:
  - Schulung des Kunden
  - Einbeziehung des Kunden in den Entwicklungsprozess
  - Abrechnung nach Aufwand
  - Transparent kontinuierliche Änderung der Spezifikation
  - Regelmäßig ausliefern

Problem: Fehleranfälliges Deployment

- Maßnahmen:
  - Modulmanager *modman einsetzen*
  - Entwicklungs-, Test- und Live-Systeme synchronisieren
  - Konfigurations-Änderungen nur über Update-Scripts durchführen

Problem: Geringe Code-Qualität von Fremd-Extensions

- Maßnahmen
  - Tools zur Konflikterkennung einsetzen (z.B. Firegento Debug Extension)
  - Review von Fremd-Extensions
  - Eigenentwicklungen nach Möglichkeit vorziehen

Zusammenfassung und Referenzen

UNSER ERGEBNIS

Prozessmodell? Konstruktionsansatz!

- Kompatibel mit agilen Prozessmodellen
  Klassisches Wasserfall-Modell ist ungeeignet!
- Bausteine zur Integration in bestehenden Prozess
- Werkzeug-Empfehlungen
- Best Practices

Foto: imgur, „MikroMan“

Referenzen

- Probleme und Maßnahmen
- Links zu empfohlenen Werkzeugen
- Bald: Vollständige Ergebnisse der Forschungsarbeit

http://code4business.de/carbon

Nikolai Krambrock
krambrock@code4business.de

http://code4business.de/carbon

$installer->endSetup()

DANKE FÜR IHRE AUFMERKSAMKEIT!