Die Glaskugel hat ausgedient, wir machen Software Analytics

Stephan Pirnbaum @spirnbaum

Dirk Mahler @dirkmahler

buschmais

Stephan Pirnbaum
stephan.pirnbaum@buschmais.com
Twitter: @spirnbaum

Dirk Mahler
dirk.mahler@buschmais.com
Twitter: @dirkmahler

buschmais

Inhaber: Torsten Busch, Frank Schwarz, Dirk Mahler, Tobias Israel

buschmais GbR
Leipziger Str. 93
01127 Dresden

info@buschmais.com
http://www.buschmais.com
Twitter: @buschmais

Legacy-Systeme

buschmais

- Eine Medaille - Zwei Seiten
  - Hohe Komplexität, aufwändig zu pflegen
  - Bewährtes Rückgrat betrieblicher Abläufe

Das einzig Stete…

buschmais

- Geschäftliche Rahmenbedingungen ändern sich
  - Mitbewerber, Geschäftsmodelle, ...
- Kann das System folgen?

Schlüsselbegriff Architektur

buschmais

- Grundlegene qualitative Eigenschaften
  - Skalierbarkeit, Performanz, Sicherheit, Wartbarkeit, ...
- Manifestierung in Code und Prozessen
  - Technologien, Strukturen, Muster, Test-Strategie, ...
- Änderungen? Aufwändig und riskant!

Das Architektur-Dilemma

buschmais

- Notwendigkeit der (kontinuierlichen) Anpassung
- Begrenztes und Unsicheres Wissen
  - Dokumentiert
  - Gefühlt
  - Real

Entscheidung verzweifelt gesucht

buschmais

- Neu-Implementierung!
- Zerlegung in Microservices!
- Doch nur Restrukturierung?

Ich habe (k)einen Plan!

buschmais

- Refactoring im Blindflug
  - Risiko?
- Vorgelagerte Analyse
  - Aufwand?

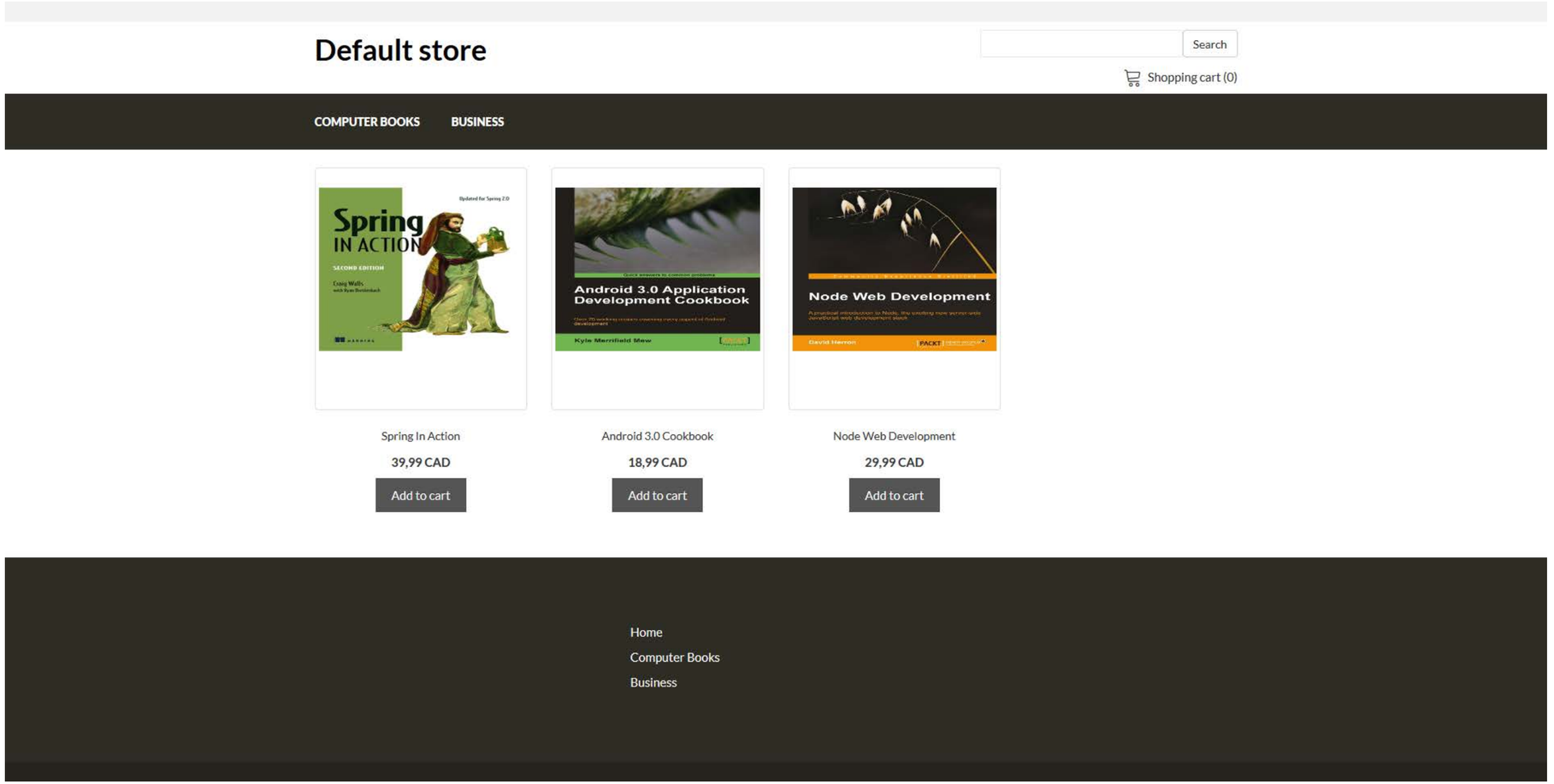
Ein Beispiel

buschmais

- Open-Source E-Commerce System "shopizer"
  - Warenkorb
  - Katalog
  - Suche
  - Bestellung
  - Shop-Administration
- Fork: https://github.com/buschmais/shopizer

Ein Beispiel

buschmais



Ein Beispiel

buschmais

- Szenario
  - Katalog sorgt für hohe Systemlast zu Spitzenzeiten
    - Gesamtes System muss skaliert werden
      - Allokation theoretisch unnötiger Ressourcen
    - Wettkampf um verfügbare Ressourcen
      - Behinderung anderer Funktionalitäten
    - Lastbedingte Ausfälle betreffen gesamtes System

Ein Beispiel

buschmais

- Lösung
  - Herauslösen des Katalogs
- Problem
  - Wieviel (und welcher) Code ist betroffen?
  - Welche Schnittstellen gibt es?
  - Welche Daten müssen repliziert werden?
  - Welches Vorgehen ist das richtige?
  - ... ?

Software Analytics

Software Analytics

buschmais

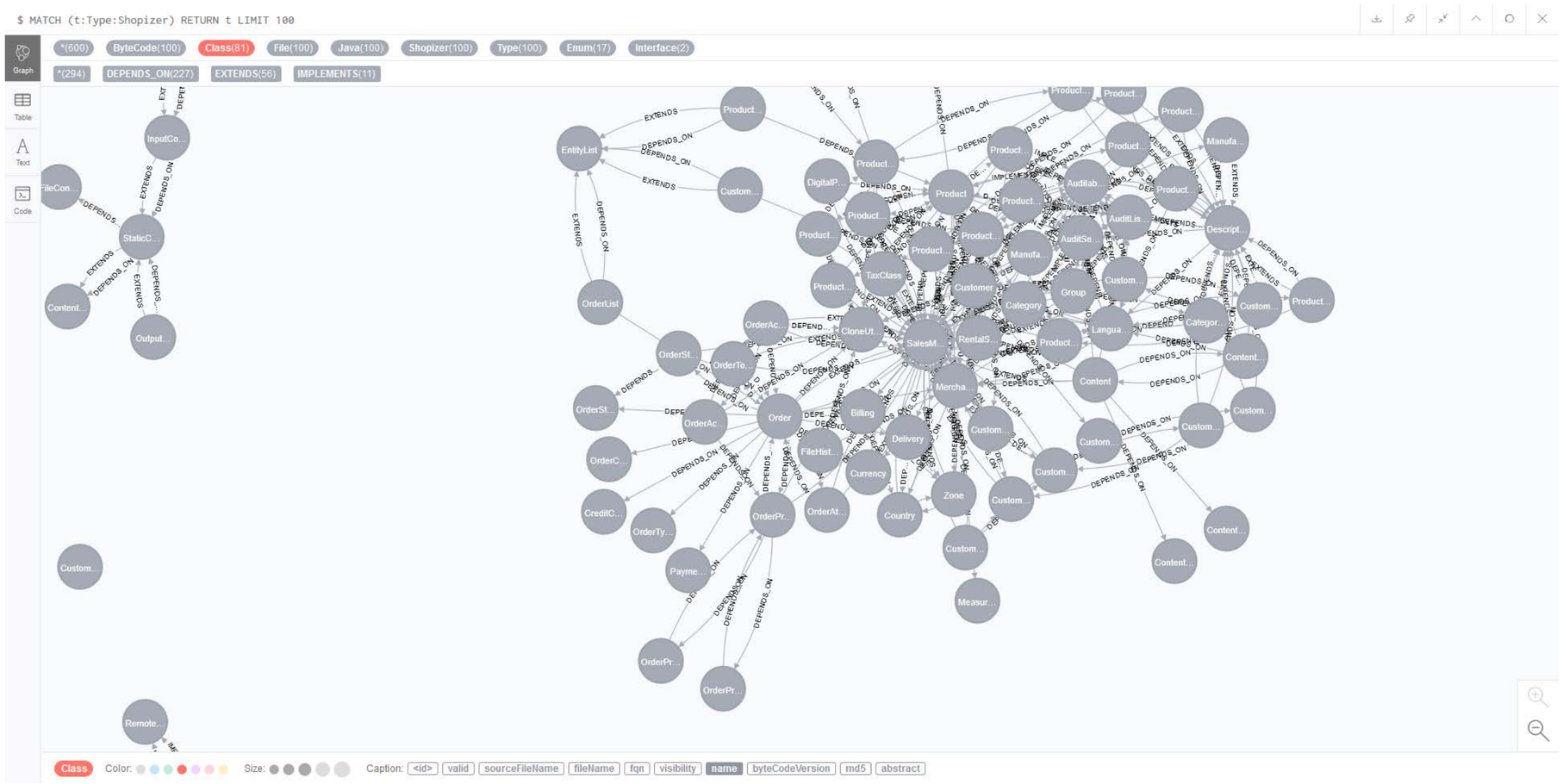
- Data Analytics für Software Systeme
  - Extraktion und Zusammenführung von Informationen aus
    - Source Code
    - statischen und dynamischen Eigenschaften
    - Entwicklungshistorien
  - Schlussfolgern von neuen Information zum Aufbau von Wissen
- Dokumentation und Kommunikation von Erkenntnissen

Software Analytics

buschmais

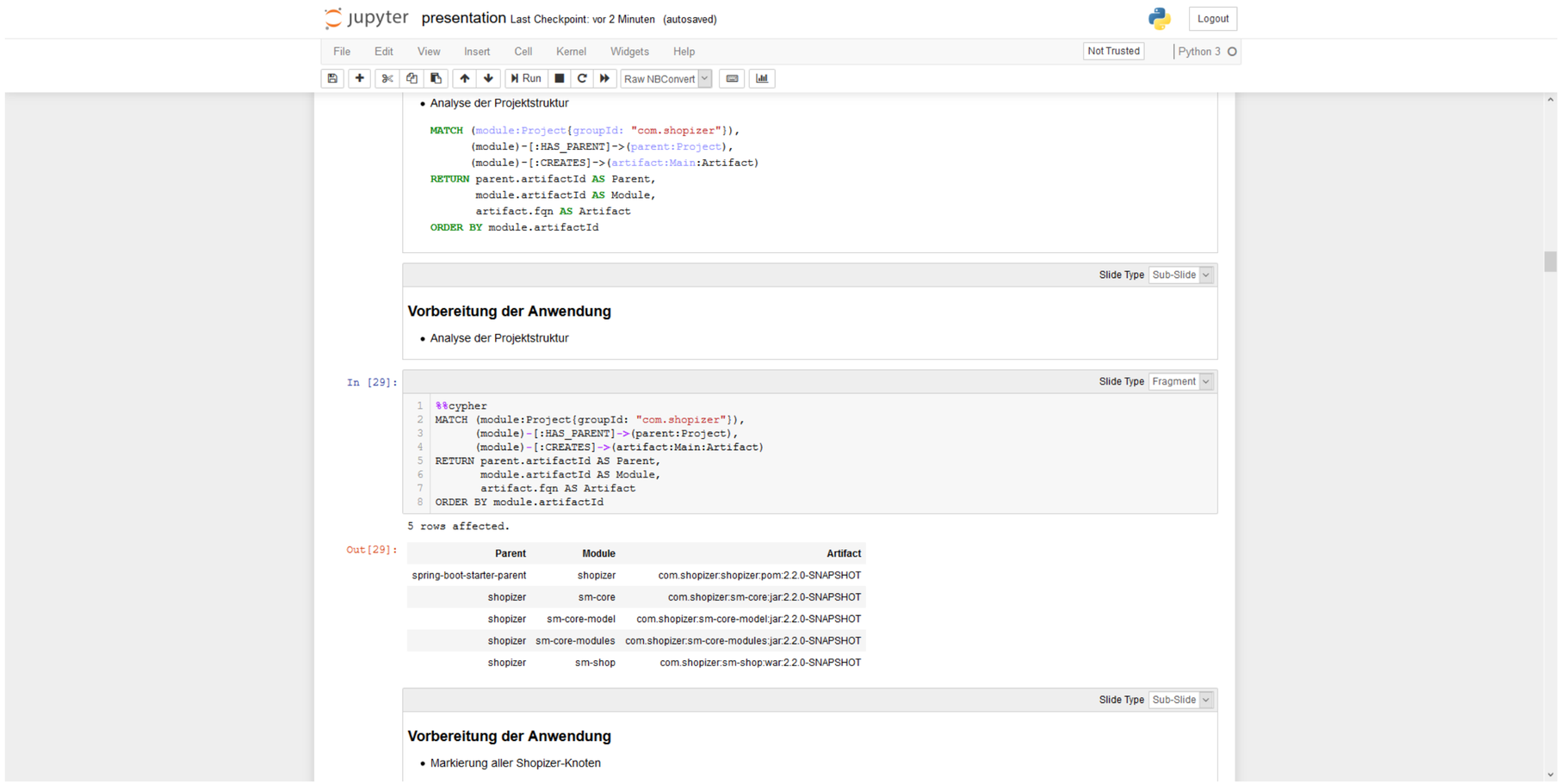
- Tooling
  - jQAssistant
    - Scan der Applikation und Abbildung in Graphdatenbank
  - Neo4j
    - Persistierung und Abfrage von Informationen
  - Jupyter Notebook
    - Dokumentation der Analyseschritte und Ergebnisse

Software Analytics mit jQA & Neo4j buschmais



Software Analytics mit Jupyter

buschmais



Software Analytics

buschmais

- Fragestellungen
  - Wie ist die Anwendung technisch strukturiert?
  - Wie ist die Anwendung fachlich strukturiert?
  - Wie hängen die einzelnen Fachlichkeiten zusammen?
  - Welcher Code wird am häufigsten geändert?

Software Analytics

Vorbereitung der Anwendung

Vorbereitung der Anwendung

buschmais

```
<plugin>
   <groupId>com.buschmais.jqassistant</groupId>
   <artifactId>jqassistant-maven-plugin</artifactId>
   <version>1.6.0</version>
   <executions>
      <execution>
         <goals>
            <goal>scan</goal>
            <goal>analyze</goal>
         </goals>
      </execution>
   </executions>
   <.../>
</plugin>
```

Vorbereitung der Anwendung

buschmais

```
<plugin>
   <.../>
   <configuration>
      <scanIncludes>
         <scanInclude>
            <path>${project.basedir}/.git</path>
         </scanInclude>
      </scanIncludes>
   </configuration>
   <.../>
</plugin>
```

Vorbereitung der Anwendung

buschmais

```
<plugin>
   <.../>
   <dependencies>
      <dependency>
         <groupId>de.kontext-e.jqassistant.plugin</groupId>
         <artifactId>jqassistant.plugin.git</artifactId>
         <version>1.5.0</version>
      </dependency>
   </dependencies>
</plugin>
```

Vorbereitung der Anwendung

buschmais

- Analyse der Projektstruktur

```
MATCH (module:Project{groupId: "com.shopizer"}),
      (module)-[:HAS_PARENT]->(parent:Project),
      (module)-[:CREATES]->(artifact:Main:Artifact)
RETURN parent.artifactId AS Parent,
       module.artifactId AS Module,
       artifact.fqn AS Artifact
ORDER BY module.artifactId
```

Vorbereitung der Anwendung

buschmais

- Analyse der Projektstruktur

Parent	Module	Artifact
spring-boot-starter-parent	shopizer	com.shopizer:shopizer:pom:2.2.0-SNAPSHOT
shopizer	sm-core	com.shopizer:sm-core:jar:2.2.0-SNAPSHOT
shopizer	sm-core-model	com.shopizer:sm-core-model:jar:2.2.0-SNAPSHOT
shopizer	sm-core-modules	com.shopizer:sm-core-modules:jar:2.2.0-SNAPSHOT
shopizer	sm-shop	com.shopizer:sm-shop:war:2.2.0-SNAPSHOT

Vorbereitung der Anwendung

buschmais

- Markierung aller Shopizer-Knoten

```
MATCH (artifact:Main:Artifact{group: "com.shopizer"})
SET artifact:Shopizer
WITH artifact
MATCH (artifact)-[:CONTAINS*]->(c)
SET c:Shopizer
RETURN artifact.name AS Artifact,
       count(DISTINCT c) AS Content_Cnt
ORDER BY artifact.name
```

Vorbereitung der Anwendung

buschmais

- Markierung aller Shopizer-Knoten

Artifact	Content_Cnt
sm-core	412
sm-core-model	220
sm-core-modules	27
sm-shop	585

Software Analytics

Wie ist die Anwendung technisch strukturiert?

Wie ist die Anwendung technisch strukturiert?

buschmais

- Analyse der existierenden Artefakte und deren Abhängigkeiten

```
MATCH (a1:Artifact:Shopizer)-[:CONTAINS*]->(t1:Type:Shopizer),
      (a2:Artifact:Shopizer)-[:CONTAINS*]->(t2:Type:Shopizer),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE a1 <> a2
RETURN a1.name AS Source,
       a2.name AS Target,
       COUNT(DISTINCT dep) AS Dependencies
ORDER BY a1.name
```

Wie ist die Anwendung technisch strukturiert?

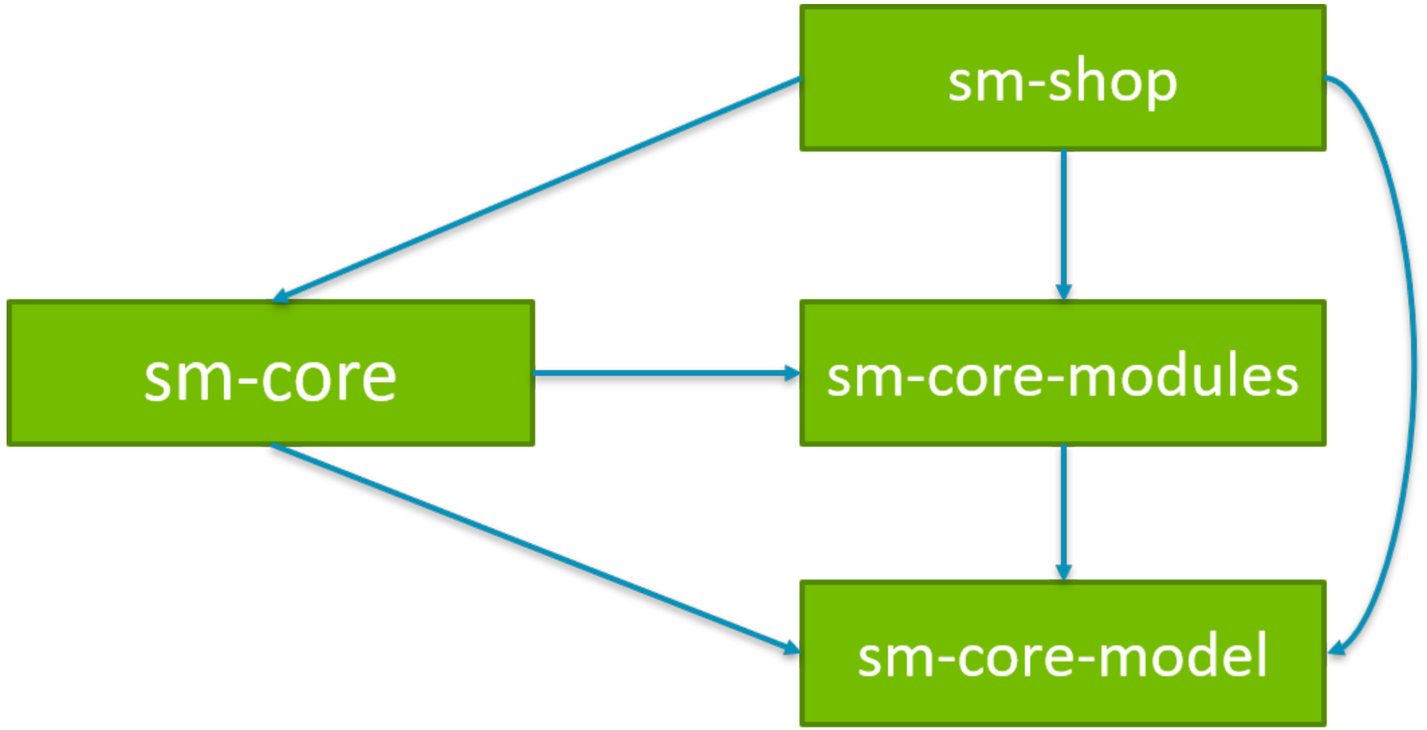
buschmais

- Analyse der existierenden Artefakte und deren Abhängigkeiten

Source	Target	Dependencies
sm-core	sm-core-modules	45
sm-core	sm-core-model	999
sm-core-modules	sm-core-model	40
sm-shop	sm-core-modules	8
sm-shop	sm-core-model	1180
sm-shop	sm-core	641

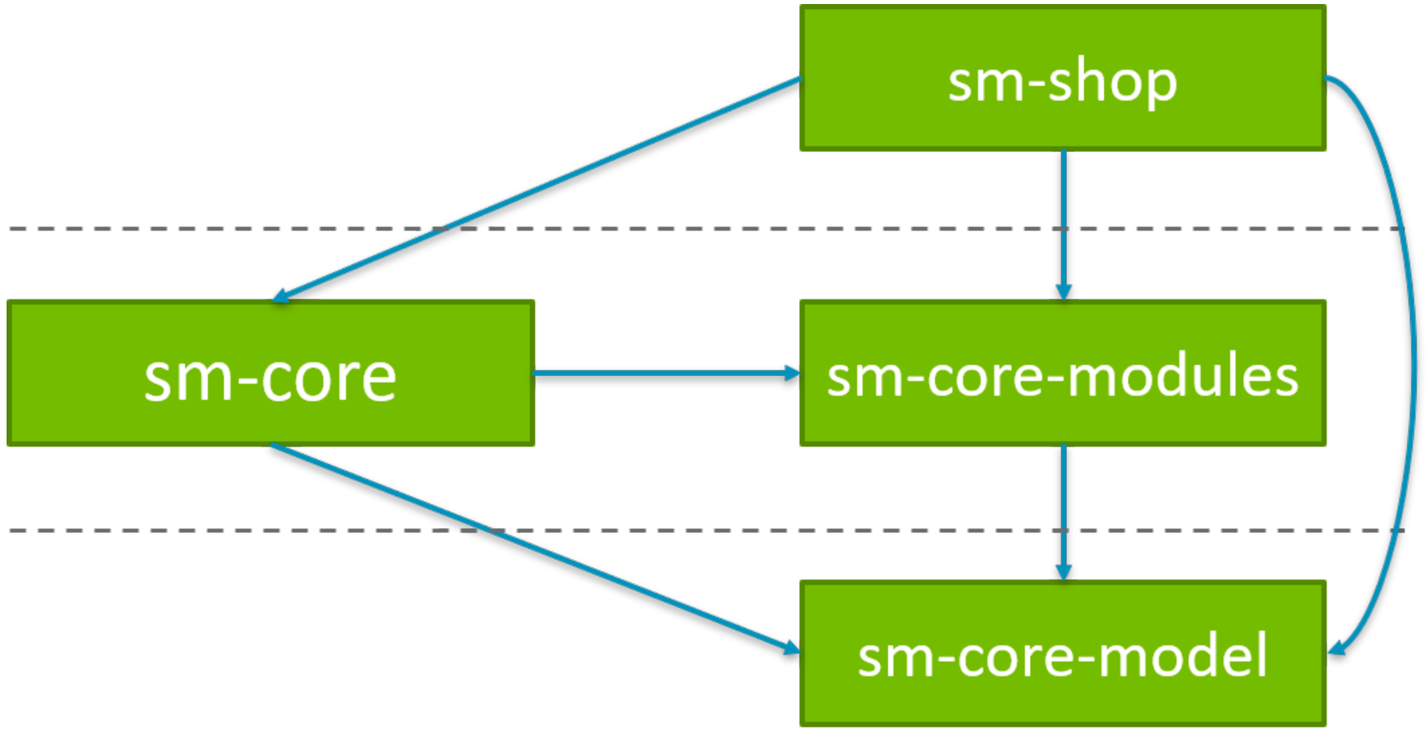
Wie ist die Anwendung technisch strukturiert?

buschmais



Wie ist die Anwendung technisch strukturiert?

buschmais



Wie ist die Anwendung technisch strukturiert?

buschmais

- Anlegen der einzelnen Knoten für die technischen Schichten

```
MERGE (dataLayer:Layer {name:"DataLayer"})
MERGE (domainLayer:Layer {name:"DomainLayer"})
MERGE (presentationLayer:Layer {name:"PresentationLayer"})
```

Wie ist die Anwendung technisch strukturiert?

buschmais

- Zuordnung der Artefakte zu den Layern

```
MATCH (a:Artifact:Shopizer)-[:CONTAINS*]->(t)
WITH DISTINCT t,
    CASE a.name
        WHEN "sm-shop" THEN "PresentationLayer"
        WHEN "sm-core" THEN "DomainLayer"
        WHEN "sm-core-modules" THEN "DomainLayer"
        WHEN "sm-core-model" THEN "DataLayer"
    END AS layer
MATCH (l:Layer {name:layer})
MERGE (l)-[:CONTAINS]->(t)
RETURN l.name AS Layer,
       count(DISTINCT t) AS Size
ORDER BY Size DESC
```

Wie ist die Anwendung technisch strukturiert?

buschmais

- Zuordnung der Artefakte zu den Layern

Layer	Size
PresentationLayer	585
DomainLayer	439
DataLayer	220

Wie ist die Anwendung technisch strukturiert?

buschmais

- Learning Outcomes
  - die Anwendung folgt einer klassichen 3-Schichten-Architektur
  - die Richtungen der Abhängigkeiten wurden korrekt implementiert
  - es konnten 100% des Codes zu technischen Schichten zugeordnet werden

Software Analytics

Wie ist die Anwendung fachlich strukturiert?

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Wo ist eine gute fachliche Strukturierung am ehesten zu erwarten?
  - suche nach Spring-Services
  - Paketnamen (service, konkrete Fachlichkeiten wenn bekannt)
  - Explorieren der Anwendung in IDE

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Identifikation der Packages unterhalb von "com.salesmanager.core.business.services"

```
MATCH (p:Package:Shopizer)-[:CONTAINS]->(subDomain:Package:Shopizer)
WHERE p.fqn = "com.salesmanager.core.business.services"
RETURN subDomain.name AS SubDomain
ORDER BY subDomain.name
```

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Identifikation der Packages unterhalb von "com.salesmanager.core.business.services"

SubDomains
['system', 'merchant', 'common', 'content', 'shipping', 'user', 'order', 'shoppingcart', 'payments', 'customer', 'catalog', 'reference', 'tax', 'search']

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Anlegen eines Knoten je Fachlichkeit

```
MATCH (p:Package:Shopizer)-[:CONTAINS]->(subDomain:Package:Shopizer)
WHERE p.fqn = "com.salesmanager.core.business.services"
WITH collect(DISTINCT subDomain.name) AS subDomains
UNWIND subDomains AS subDomain
MERGE (sD:SubDomain {name: subDomain})
```

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Zuordnung aller Klassen zu den Knoten anhand ihrer Paketnamen

```
MATCH (sD:SubDomain),
      (p:Package:Shopizer)-[:CONTAINS*]->(t:Type:Shopizer)
WHERE p.name = sD.name
MERGE (sD)-[:CONTAINS]->(t)
RETURN sD.name AS  SubDomain,
       count(DISTINCT t) AS Classes
```


shipping
reference
commor
tax
shoppingca
mercha
order
catalog
content
user
customer
ayments
system
search

buschmais

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Prozentualer Anteil der zugeordneten Klassen

```
MATCH (t:Type:Shopizer)
WITH count(t) AS Total
MATCH (:SubDomain)-[:CONTAINS]->(t:Type:Shopizer)
RETURN 100 * count(t) / Total AS Coverage
```

Coverage
69

Wie ist die Anwendung fachlich strukturiert?

buschmais

- Learning Outcomes
  - Die Geschäftslogik teilt sich in 13 fachlich motivierte Subdomänen.
  - Es konnten 69% aller Klassen zu Fachlichkeiten zugeordnet werden.
  - Der Katalog ist mit 140 Klassen die größte Subdomäne.

Software Analytics

Wie hängen die einzelnen Fachlichkeiten zusammen?

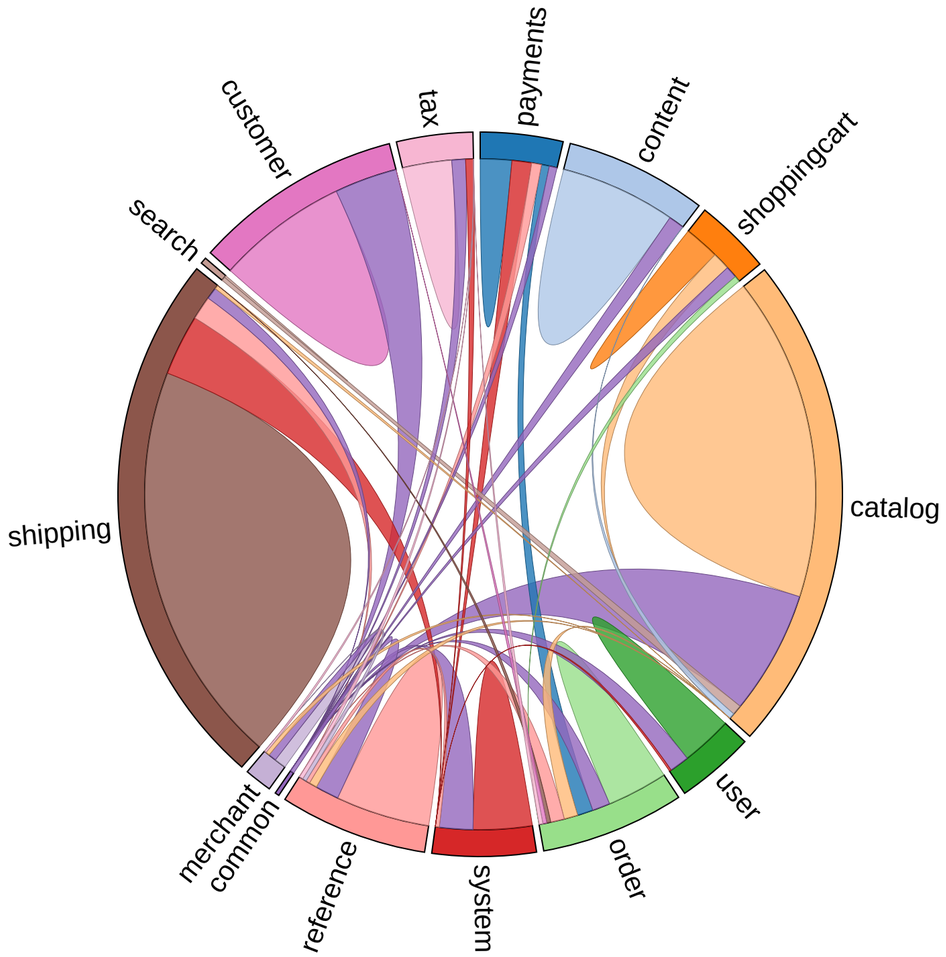
Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Visualisierung der Fachlichkeiten im Domain Layer

```
MATCH (sD1:SubDomain)-[:CONTAINS]->(t1:Type:Shopizer),
      (dL:Layer{name:"DomainLayer"})-[:CONTAINS]->(t1),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (dL)-[:CONTAINS]->(t2),
      (t1)-[dep:DEPENDS_ON]->(t2)
RETURN sD1.name AS Source,
       sD2.name AS Target,
       sum(dep.weight) AS X_Count
```

buschmais


payments
content
shoppingcart
catalog
user
order
system
reference
common
merchant
shipping
search
customer
tax

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten des Katalogs

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE sD2.name <> "catalog"
RETURN sD1.name AS Source,
       sD2.name AS Target,
       count(DISTINCT dep) AS Dependencies
ORDER BY Dependencies DESC
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten des Katalogs

Source	Target	Dependencies
catalog	common	52
catalog	reference	42
catalog	merchant	41
catalog	customer	16
catalog	content	11
catalog	tax	6
catalog	search	1

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der ausgehenden Abhängigkeiten

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (targetLayer:Layer)-[:CONTAINS]->(t2),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE sD2.name <> "catalog" AND
      targetLayer.name <> "DataLayer"
WITH count(dep) AS Total
...
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der ausgehenden Abhängigkeiten

```
...
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Interface:Shopizer),
      (targetLayer:Layer)-[:CONTAINS]->(t2),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE sD2.name <> "catalog" AND
      targetLayer.name <> "DataLayer"
WITH Total,
     count(dep) AS Interfaces
RETURN Total,
       Interfaces, 100 * Interfaces / Total AS Percentage
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der ausgehenden Abhängigkeiten

Total	Interfaces	Percentage
41	22	53

- Zum Vergleich: Alle Layer

Total	Interfaces	Percentage
169	27	15

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten, Anzahl der Codestellen

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE sD2.name <> "catalog"
RETURN sD1.name AS Source,
       sD2.name AS Target,
       sum(dep.weight) AS Dependencies
ORDER BY Dependencies DESC
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten, Anzahl der Codestellen

Source	Target	Dependencies
catalog	merchant	306
catalog	reference	221
catalog	common	139
catalog	customer	74
catalog	content	42
catalog	tax	15
catalog	search	5

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten, Hot Spots

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t1)-[dep:DEPENDS_ON]->(t2)
WHERE sD2.name <> "catalog"
RETURN t2.fqn AS Dependency,
       sum(dep.weight) AS Dependencies
ORDER BY Dependencies DESC
LIMIT 5
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der ausgehenden Abhängigkeiten, Hot Spots

Dependency	Dependencies
com.salesmanager.core.model.merchant.MerchantStore	306
com.salesmanager.core.model.reference.language.Language	190
com.salesmanager.core.business.services.common.generic.SalesManagerEntityServiceImpl	50
com.salesmanager.core.model.customer.Customer	36
com.salesmanager.core.model.common.audit.AuditSection	32

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehende Abhängigkeiten des Katalogs

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t2)-[dep:DEPENDS_ON]->(t1)
WHERE sD2.name <> "catalog"
RETURN sD2.name AS Source,
       sD1.name AS Target,
       count(DISTINCT dep) AS Dependencies
ORDER BY Dependencies DESC
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehenden Abhängigkeiten des Katalogs

Source	Target	Dependencies
order	catalog	80
search	catalog	28
customer	catalog	19
shoppingcart	catalog	14
shipping	catalog	8
reference	catalog	5
tax	catalog	4
content	catalog	2
merchant	catalog	2

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der eingehenden Abhängigkeiten des Katalogs

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (targetLayer:Layer)-[:CONTAINS]->(t1),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t2)-[dep:DEPENDS_ON]->(t1)
WHERE sD2.name <> "catalog" AND
      targetLayer.name <> "DataLayer"
WITH count(dep) AS Total
...
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der eingehenden Abhängigkeiten des Katalogs

```
...
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (targetLayer:Layer)-[:CONTAINS]->(t1),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Interface:Shopizer),
      (t2)-[dep:DEPENDS_ON]->(t1)
WHERE sD2.name <> "catalog" AND
      targetLayer.name <> "DataLayer"
WITH  Total, count(dep) AS Interfaces
RETURN Total,
       Interfaces, 100 * Interfaces / Total AS Percentage
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der Art der eingehenden Abhängigkeiten

Total	Interfaces	Percentage
96	2	2

- Zum Vergleich: Alle Layer

Total	Interfaces	Percentage
162	5	3

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehenden Abhängigkeiten, Anzahl der Codestellen

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t2)-[dep:DEPENDS_ON]->(t1)
WHERE sD2.name <> "catalog"
RETURN sD2.name AS Source,
       sD1.name AS Target,
       sum(dep.weight) AS Dependencies
ORDER BY Dependencies DESC
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehenden Abhängigkeiten, Anzahl der Codestellen

Source	Target	Dependencies
order	catalog	288
search	catalog	113
customer	catalog	102
shipping	catalog	84
shoppingcart	catalog	76
reference	catalog	21
content	catalog	7
tax	catalog	6
merchant	catalog	2

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehenden Abhängigkeiten, Hot Spots

```
MATCH (sD1:SubDomain {name: "catalog"})-[:CONTAINS]->(t1:Type:Shopizer),
      (sD2:SubDomain)-[:CONTAINS]->(t2:Type:Shopizer),
      (t2)-[dep:DEPENDS_ON]->(t1)
WHERE sD2.name <> "catalog"
RETURN t1.fqn AS Dependency,
       sum(dep.weight) AS Dependencies
ORDER BY Dependencies DESC
LIMIT 5
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der eingehenden Abhängigkeiten, Hot Spots

Dependency	Dependencies
com.salesmanager.core.model.catalog.product.Product	152
com.salesmanager.core.business.services.catalog.product.PricingService	67
com.salesmanager.core.model.catalog.product.attribute.ProductAttribute	48
com.salesmanager.core.business.services.catalog.product.ProductService	48
com.salesmanager.core.model.catalog.product.price.FinalPrice	41

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der zeitlichen Kopplung zwischen Subdomänen - Vorbereitung Git-Analyse

```
MATCH (p:Package)-[:CONTAINS]->(t:Type)
WITH t, p.fileName + "/" + t.sourceFileName as sourceFileName
MATCH (f:Git:File)
WHERE f.relativePath ends with sourceFileName
MERGE (t)-[:HAS_SOURCE]->(f)
RETURN f.relativePath, collect(t.fqn)
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der zeitlichen Kopplung zwischen Subdomänen - Vorbereitung Git-Analyse

```
MATCH (c:Commit)-[:HAS_PARENT]->(p:Commit)
WITH c, count(p) as parents
WHERE parents > 1
SET c:Merge
RETURN count(c)
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der zeitlichen Kopplung zwischen Subdomänen

```
MATCH (:Branch{name:"heads/2.2.0-Analysis"})-[:HAS_HEAD]->(h:Commit),
      shortestPath((h)-[:HAS_PARENT*0..]->(c:Commit)),
      (c)-[:CONTAINS_CHANGE]->()-[:MODIFIES]->(f:File)
WHERE NOT c:Merge
WITH c, collect(f) as files
WHERE size(files) < 100
UNWIND files as f
MATCH (c)-[:CONTAINS_CHANGE]->()-[:MODIFIES]->(f2:File)
WHERE f <> f2
WITH f, f2, count(f2) as weight
MERGE (f)-[t:HAS_TEMPORAL_COUPLING]->(f2)
SET t.weight=weight
RETURN count(t)
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der zeitlichen Kopplung zwischen Subdomänen

```
MATCH (s1:SubDomain{name: "catalog")-[:CONTAINS]->(t1:Type)-[:HAS_SOURCE]-(f1),
      (s2:SubDomain)-[:CONTAINS]->(t2:Type)-[:HAS_SOURCE]-(f2),
      (f1)-[t:HAS_TEMPORAL_COUPLING]->(f2)
WHERE s1 <> s2
WITH s1, s2, sum(t.weight) as weight
MERGE (s1)-[t:HAS_TEMPORAL_COUPLING]->(s2)
SET t.weight = weight
RETURN s1.name, s2.name, weight
ORDER BY weight desc
LIMIT 5
```

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Analyse der zeitlichen Kopplung zwischen Subdomänen

s1.name	s2.name	weight
catalog	customer	228
catalog	order	63
catalog	search	14
catalog	content	10
catalog	shipping	5

Wie hängen die einzelnen Fachlichkeiten zusammen?

buschmais

- Learning Outcomes
  - Die Suche gehört exklusiv zum Katalog.
  - Die Kommunikation zwischen Katalog und Kern ist schwach durch Interfaces abstrahiert.
  - Der Katalog definiert 169 Abhängigkeiten zum Kern auf Klassenebene, 27 davon gegen Interfaces.
    - Hot Spots sind MerchantStore und Language.
  - Der Kern definiert 162 Abhängigkeiten zum Katalog auf Klassenebene, 5 davon gegen Interfaces.
    - Hot Spot ist Product.

Software Analytics

War's das?

War's das?

buschmais

- Nein, Software Analytics bietet nahezu unbegrenzte Möglichkeiten.
  - Analyse abhängig vom jeweiligen Softwaresystem und vorhandener Daten
- Weitere Analysen
  - Ownership von Subdomänen, Fix-Commits
  - Testcoverage, Teststruktur
  - Defektdichten
  - Toter Code
  - Analyse von Laufzeitdaten
  - ...

One more thing…

Wir sind dabei!

JavaLand - Schulungstag "Lasst uns einen Monolithen (z)erlegen!"

21. März 2019, 09:00 - 17:00 Uhr

Nur noch wenige Plätze frei.

Anmeldung und Infos unter: www.javaland.eu

Vielen Dank!


BUSCHMAIS WÜNSCHT
FROHE WEIHNACHTEN
UND
EINEN GUTEN RUTSCH
INS NEUE JAHR!