Ludwig-Maximilians-Universität München
CIS

OpenGalen Projekt

Hauptseminar
„Beschreibungslogik“
SoSe 2009
H.Leiss, M. Hofmann

verfasst von
Svitlana Bezrukova

Inhalt

1.OpenGalen: Definition und Begriffe

In der vorliegenden Ausarbeitung des Referats „OpenGalen“ werden die wichtigsten Aspekte des Galen-Projekts kurz zusammengefasst und ein allgemeiner Überblick über seine Zwecke und die Themenbereiche gegeben werden.

GALEN (**Generalised Architecture for Languages, Encyclopadias and Nomenclatures in Medicine)** ist ein Name für die Technologien, die sich seit 1991 mit der klinischen medizinischen Terminologie beschäftigen. Von Anfang an war es als ein Forschungs- und Entwicklungsprojekt (1991-1994) zur Verarbeitung von Konzepten für klinische Informationssysteme und elektronische Patienten-Aufzeichnungen in Europa gedacht.

Ursprünglich war die Zielsetzung von Galen, das gesamte medizinische Wissen und dessen Begriffe in einem formalen Modell darzustellen. Auch formale Regeln über medizinische Konzepte mussten damals zum ersten Mal erstellt werden. Für diese Zwecke war eine neue und besondere formale Sprache erforderlich, die medizinisches Wissen und seine Begriffe beschreiben konnte. Um diese Ziele zu erreichen, schufen die Spezialisten in Galen drei medizinische Informationsmodelle. Darunter sind das Modell der medizinischen Konzepte, das Modell der Informationssysteme, das die Konzepte aus dem ersten Modell verwendet, um die Daten von Patienten und medizinische Berichte darzustellen, und das dritte Modell von Patienten, geschrieben mit einem „Wissenssystem“, was auf den medizinischen Berichten basiert. Alle drei Modelle werden in einer von Galen-Experten entwickelten formalen Sprache Grail (GALEN Representation and Integration Language: GRAIL) realisiert und beschrieben.

Zwecke von OpenGalen

Heutzutage produziert Galen ein computerbasierendes multilinguales Kodierungssystem für für Medizin in Europa. Es versucht, medizinische Anwendungen zu erleichtern und Ärzte in ihrem beruflichen Leben zu unterstützen. Der Galen - Ansatz stellt einen Paradigmawechsel in der Herangehensweise an die medizinischen Terminologien dar. Diese Verschiebung lautet :

1) weg von umfangreichen Klassifizierungen der Begriffe hin zur Zusammensetzung der medizinischen Konzepte mit formalen Eigenschaften;

2) weg von einzelnen Konzept-, oder Begriffstrukturen hin zu einer konzeptunabhängigen Sprache, die durch separate Grammatiken und Lexika interpretiert wird.

Das Ziel des Galen - Projektes ist die Erfassung der wesentlichen Grundsätze und

Kenntnisse aus der medizinischen Terminologie in ein formales Modell aufgrund der erstellten Ontologie. So wie Menschen Sprache nach dem „LEGO-Prinzip“ verwenden, d.h durch Zerlegung in kleinste Bausteine (Vokabular) und regelkonformes Kombinieren zu sinnvollen Sätzen (Grammatik), so werden in kompositionellen Terminologien wie OpenGalen komplexe Begriffe durch Verwendung einer geeigneten Logik rechnergestützt verarbeitet. Solche Terminologien ermöglichen eine maschinelle Interpretation ausgetauschter Patientendaten, zum Beispiel für Prüfmodule zur Verbesserung der Arzneimitteltherapiesicherheit.

Die folgende Schema stellt allgemein die Struktur bzw. Themenbereiche von OpenGalen dar.

OpenGalen-Projekt

(A management architecture for clinical information that includes an ontology for human anatomy)

<table>
<tr>
<td><u>Galen Terminology Server</u></td>
<td>Galen Common Reference Model
besteht aus vier Ebenen :
<table>
<tr><td>1. Hihg-Level Ontology</td></tr>
<tr><td>2. The Common Reference Model selbst</td></tr>
<tr><td>3. Ausfürliche Erweiterungen für Subdomainsbildungen
und für spezielle Anwendungen</td></tr>
<tr><td>4. Modell der chirurgischen Verfahren/Prozeduren für Anatomie, Chirurgie, Krankheiten und ihre Modifikatoren in den chirurgischen Verfahren.</td></tr>
</table>
</td>
<td><u>Grail</u></td>
</tr>
</table>

Die Galen Concept Representation Language (GRAIL) kann als eine Menge von Bildungs- , Modell-, und Steuerungsregeln beschrieben werden.

GRAIL selbst wird vom **Galen Terminology Server** implementiert: dieser Server unterstützt eine Reihe von verschiedenen Servern in medizinisch orientierten Anwendungen. Mit Hilfe dieses Ansatzes werden umfangreiche medizinische traditionelle Schemata in ein formales Modell transformiert. Der Server wird in sieben Sprachen implementiert: Englisch, Französisch, Italienisch, Dänisch, Deutsch, Finnisch und Schwedisch. Die Erweiterung der Sprache wird nicht manuell gemacht, sondern es wird das Prinzip der Zusammensetzung der Galen Technologie benutzt : komplexe Konzepte werden erst dann erstellt und klassifiziert, wenn es nötig wird. Aber vorher müssen relativ kleine Gruppen von Basiskonzepten manuell übersetzt werden, um die komplexeren Strukturen weiter zusammensetzen zu können.

Dieses Modell heißt **The Galen Common Reference Model** und ist eine in GRAIL geschriebene medizinische Terminologie[1]. Sie besteht aus vier Ebenen. Es gibt keine strengen Unterschiede zwischen diesen vier Ebenen, sondern sie sind mehr methodisch als streng logisch zusammengebunden. Die deutlichen Unterschiede liegen in den Konzepten selbst, wie Phänomen oder Krankheit; Körperteil oder Distal-Phalanx des linken vierten Fingers.

Dieses Modell präsentiert alle vernünftigen medizinischen Konzepte. Im Modell gibt es die Informationen über medizinisches Wissen, worüber Ärzte sich einig sind. Im Fall einer Auseinandersetzung zwischen den Wissenschaftlern wird im Modell darüber verständlich berichtet.

Für die Entwicklung des Galen Common Reference Modells ist eine Zusammenarbeit von Experten erforderlich. Eine kleine Gruppe von Experten entwickelt eine abstrakte Ontologie und definiert die Strukturen des Modells. Sie arbeitet direkt in GRAIL.

Die andere Gruppe von Ärzten arbeitet selbständig mit Abstraktionen des höheren Niveaus, genannt „Intermediate Representation“. Diese Gruppe verwendet von den Experten bereits entwickelte Tools.

2. Teile von OpenGalen : Hierarchien,Themenbereiche

The GALEN High Level Ontology

Die Ontologie von OpenGalen wird aufgrund von zwei grundlegenden Prinzipien aufgebaut, nämlich Subsumption und Taxonomie. Die Subsumption-Relation ('kind-of') ist von den anderen transitiven Relationen wie partitive (‘part-of’) und causative (‘caused-by’) Relationen getrennt.

Die Taxonomie der elementaren Konzepte umfasst die pure Hierarchie, wo jedes elementare Konzept nur einen einzigen elementaren Vorgänger hat. Daraus folgt, dass alle anderen Klassifikationen automatisch auf der Basis von Beschreibungs- und Definitionskonzepten präsentiert werden.

Zum Beispiel basiert die Taxonomie der elementaren Konzepten für Chemikalien auf ihrer **Struktur.** Es gibt aber noch eine andere Hierarchie - die **Hierarchie der Funktionen, wo** Testosteron Steroid (in Struktur) ist und gleichzeitig die Funktion eines männlichen Sexualhormon (in Funktion) hat. **Diese zwei unterschiedlichen Hierarchien - chemische Struktur und chemische Funktion - bleiben unabhängig von einander**

1 In OpenGalen wird der Begriff Terminologie statt Ontologie benutzt.

und können für verschiedene Zwecke benutzt werden. Für diese Zwecke wird in Grail der Operator 'Rolle' verwendet. Er zeigt die Disjunktion zwischen den elementaren Konzepten in einer der Hierarchien:

<<<<(Steroid which playsRole HormoneRole)

(Steroid which playsRole DrugRole)

(Steroid which playsRole NeurotransmitterRole). >>>>[2]

Die Ontologie ist so aufgebaut, dass die Klassifikationsgründe immer im Modell ausführlich aufgelistet sind und für jede Analyse bzw. jedes Verfahren leicht angewendet werden können. Dieses Prinzip führt dazu, dass die Hierarchien bottom-up aufgebaut sind. Die Klassifikation von zerlegbaren Konzepten läuft automatisch ab und basiert auf formalen logischen Kriterien.

Allgemein unterscheidet man zwei Hierarchietypen:

1) Zusammensetzung von verschiedenen Konzepttypen und Kompositionalprinzipien in eine neue Hierarchie, d.h die Ursache und die entsprechende Behandlung werden in die gleiche Hierarchie eingetragen:

respiratory condition
..cough
....cough reported by patient
....cough observed by doctor
....productive cough

2) Zusammensetzung von verschiedenen Relationen (part-whole and generic) :

Bone
..Flat Bone
..Long Bone
....Femur
.....Periosteum
.....Subperiosteum
.....Shaft
.....Proximal third of shaft of bone
.....Marrow

Die Zusammensetzung von verschiedenen Kriterien hängt vom Anwendungsbereich dieser Informationen in der Praxis ab.

Taxonomy of high level categories

In Galen unterscheidet man zwischen den "Dingen" und den "Eigenschaften von Dingen". Unter „Domain Categorie“ werden die „Dinge“ beschrieben und in der Praxis werden diese "Dinge" unter drei Arten unterteilt (sehe Tabelle 2):

- GeneralisedStructures — abstrakte oder physikalische Dinge, unabhängige von dem

2 Imlementierungsbeispiel in Grail

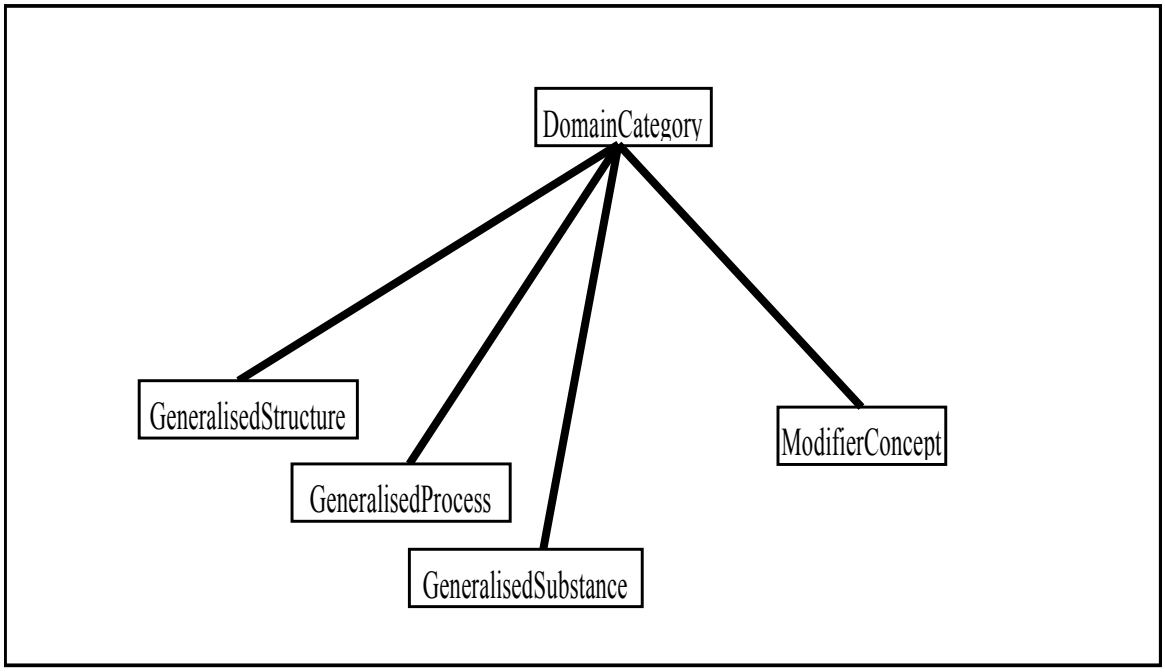
Zeitraum.

- GeneralisedSubstances — vorübergehende abstrakte oder physikalische Dinge, unabhängige von dem Zeitraum.
- GeneralisedProcesses — Veränderungen in einem beliebigem Zeitraum.

Entitäten aus diesen drei Kategorien beziehen sich generell auf eine andere Familie von semantischen Links, die als 'Constructive Attributes' bezeichnet werden. Sie drücken 'is part of', 'has function', 'is consequence of' - Merkmale aus. Im Unterschied zu 'Constructive Attributes' beschreibt

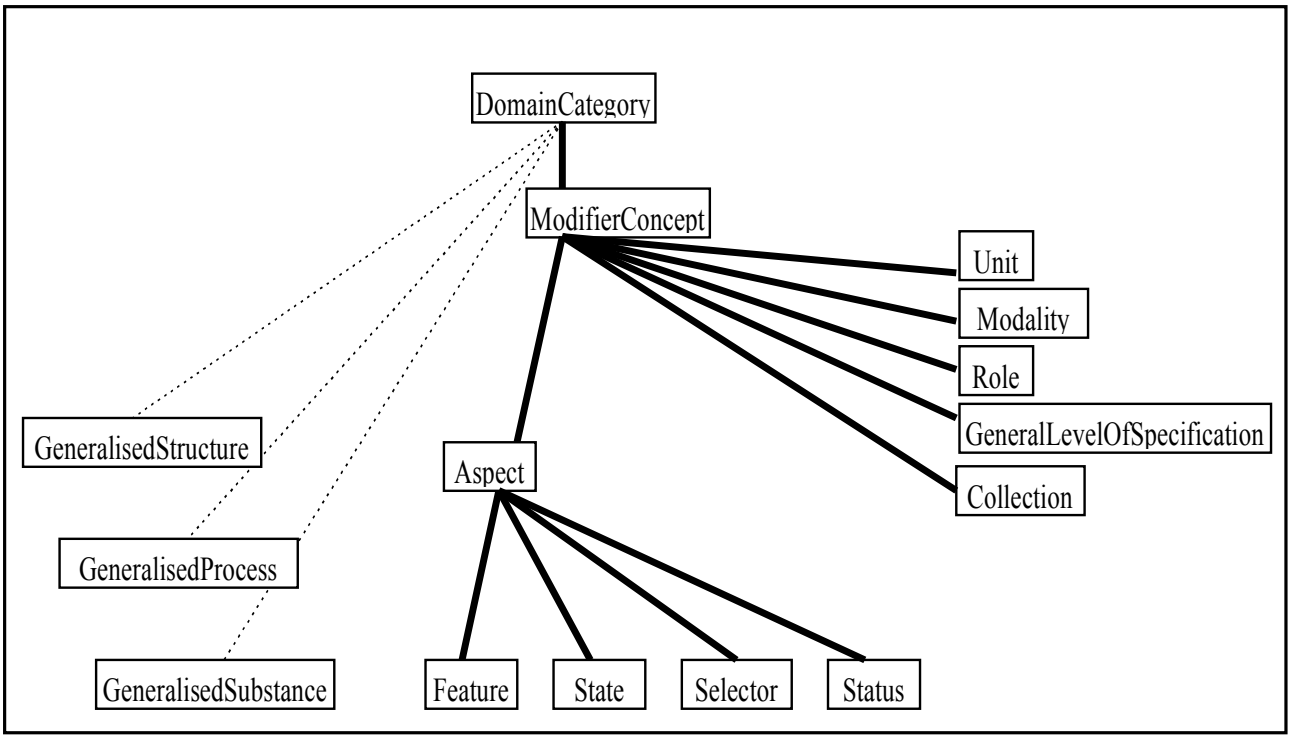
- ModifierConcept die Eigenschaften bzw. Charakteristiken der Strukturen, der Substanzen und der Prozessen. 'Modifier concepts' haben generell Links wie 'hasShape', 'hasFrequency', 'hasLeftRightSelector' und weitere.

Tabelle 2 : Overview of GALEN ontology



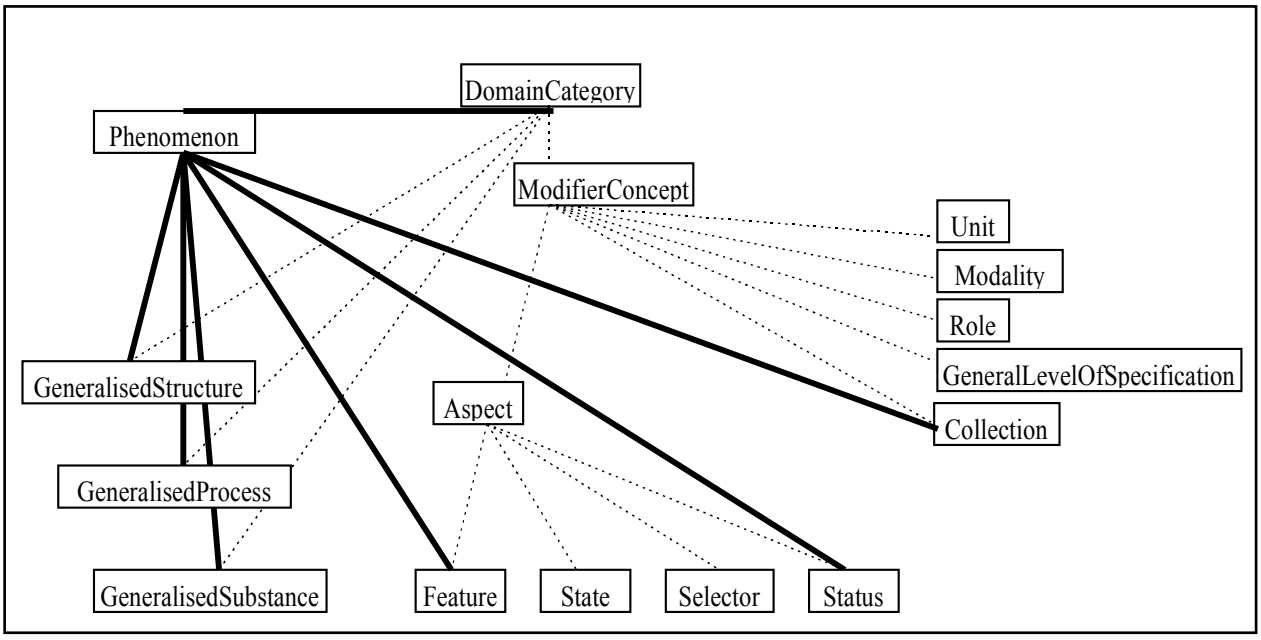
In der Tabelle 3 wird dieses Framework mit dem erweiterten Teil von ModifierConcept durch spezifische modifizierte Konzepte gezeigt. Diese Taxonomie präsentiert 'the primary, asserted GALEN high level ontology'.

Tabelle 3 : Primary High Level GALEN Ontology



'A secondary asserted taxonomy' ist der primary Taxonomie untergeordnet und damit wird die Basisordnung der OpenGalen Ontologie geschaffen.

Tabelle 4 : Secondary High Level Taxonomy or Secondary Structure



Bemerkenswert ist, dass die Kategorie Phenomenon die Disjunktion zwischen den beobachteten Kategorien präsentiert und gleichzeitig die Störungen und Krankheiten zusammenfasst. In OpenGalen wird der Begriff Krankheit wie

<<<<(Phenomenon which hasPathologicalStatus pathological)
name PathologicalPhenomenon. >>>> definiert.

Unter PathologicalPhenomenon werden die wichtigsten Arten der Pathologien aufgelistet:

- PathologicalPhenomenon
  - BodySystemPathology
    - ImmuneSystemPathology
    - ReproductiveDisorder
    - MentalIllness
    - MusculoSkeletalPathology
    - SkinPathology
    - OrodentalPathology
    - EndocrineDisorder
    - CardiovascularPathology
    - GenitoUrinaryPathology
    - DigestiveSystemPathology
    - RespiratorySystemPathology
    - CortisolOverload
    - ChemoreceptorSystemPathology
    - SensorySystemPathology
    - NervousSystemPathology
      LymphoreticularDisorder

Als medizinisch orientierte Ontologie unterteilt Galen ' structures, substances and processes ' weiter in organische und unorganische, biologische und nicht biologische. '

BodyStructure, BodySubstance and BodyProcess' werden verwendet, um die elementaren Kategorien der menschlichen Anatomie und Patho-Physiologie zu vereinigen. Die Taxonomie der elementaren Kategorien aus der Tabelle 4 wird noch tiefer weiter untergliedert, zum Beispiel :

GeneralisedProcess

* SpecificProcess
  * BiologicalProcess
    BodyProcess
    Behaviour
  * NonBiologicalProcess
    PhysicalProcess
    ChemicalProcess
  * GenericProcess

Die Ausarbeitung bzw. praktische Einsetzung der oben beschriebenen Ontologien erfolgt durch die OpenGalen-Tools , die wir als nächstes betrachten.

3.Grail und seine Ausdrucksmöglichkeiten

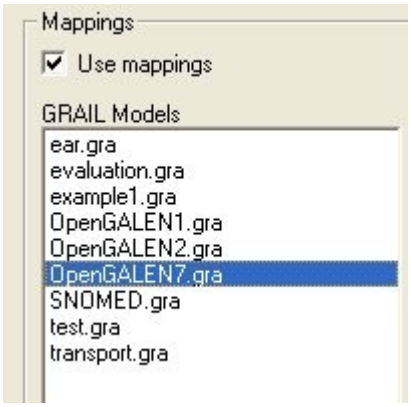
(die beschreibende Logik in Form einer formalen Sprache)

Für das Arbeiten in OpenGalen wurden zwei Arbeitstechniken entwickelt, die „ Intermediate Representation“ und die direkte Implementierung in Grail.

Unter „Intermediate Representation“ versteht man eine Arbeitsweise mit der High Level Language. Damit wird die medizinische Ontologie direkt im Interface von GliGlow (*GirC)* manuell erstellt. Das erinnert sich an die Arbeitsweiese in Protége.[3]

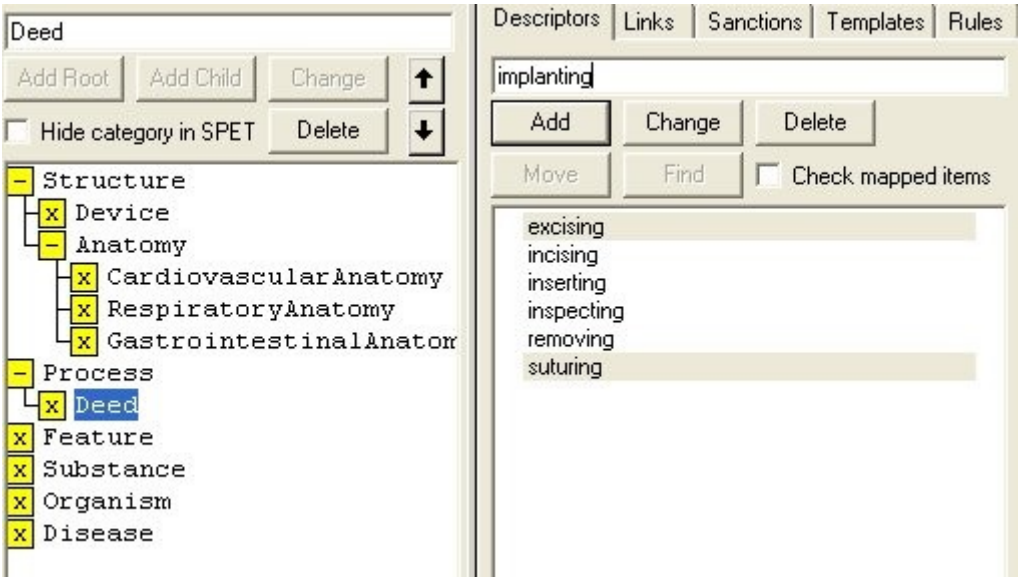
Als erstes wird ein neues Datafile (bekannt als Dictionary) geöffnet und ihm wird der Name GalenAuthoringTerminology zugeschrieben.

Der nächste Schritt ist sehr wichtig, wo ein richtiges Model ausgewählt wird :



3 http://protege.stanford.edu/

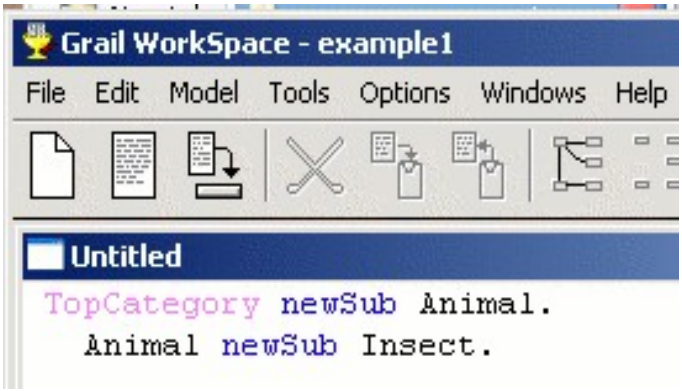
Alle weiteren Schritte bestehen aus dem Erzeugen von Kategorien (Deed) und ihren Eigenschaften (Descriptors). Normalerweise werden zehn bis fünfzehn TopCategory mit mehreren Levels gebaut. Im Prinzip gibt es keine feste Zahlbegrenzung.



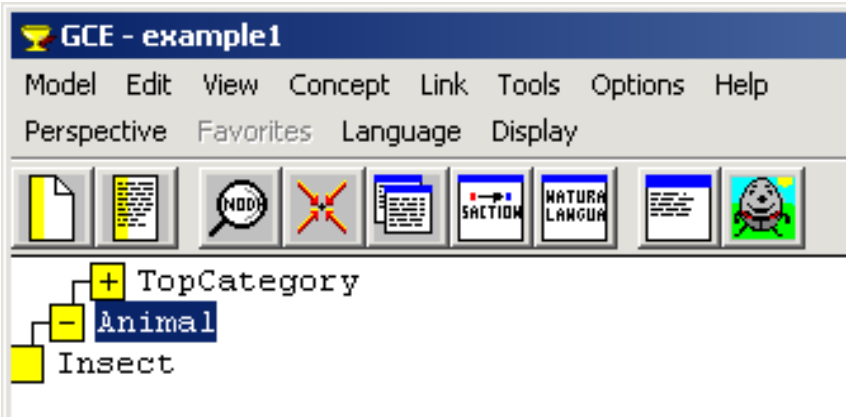
Mit Operatoren wie „Add, Change, Delete“ beschreibt man die Eigenschaften einer TopCategory . Allein stehendene Discriptors haben keinen Einfluss auf die Hierarchien, deswegen muß man entsprechende semantische (Main) Links definieren, um Ausdrücke für komplexere Konzepte zu schaffen.

Das weitere Verfahren richtet sich auf Sanctions, Mappings (Templates), Rules und andere Bestandteile, welche stufenweise mit ihren eigenen Charakteristiken vollständig erfasst und definiert werden.

Eine andere Möglichkeit, die medizinischen Begriffe zu definieren, bietet Grail. Um direkt in Grail implementieren zu können, muß man „The Classifcation Workbench “ (ClaW) herunterladen. Das Implementieren in Grail sieht folgendermaßen aus :



Um die Hierarchie dieser implementierten Ausdrücke zu sehen, müssen wir das GCE Interface mit dem entsprechenden Modellnamen öffnen : Hier sind zwei unseren bereits in Grail implementierten Statements zu sehen. Sie werden automatisch an das GCE Framework geliefert und damit wird die Ontologie von Animal erstellt.



Die grundlegenden Prinzipien und Implementierungsregel gibt es im Grail Tutorium auf der Internetseite von OpenGalen. Die detaillierte Beschreibung und ausführliche Erklärung der Grail-Techniken ist aus Platzgründen in dieser Ausarbeitung nicht möglich. Dazu wird aber versucht, einen kurzen Überblick in das Grailthema zu schaffen. Seit einigen Jahren wird systematisch OpenGalen in Form von OWL-Implementierungen ausgearbeitet und verbessert.[4]

4 Sehe Appendix 2 und www.w3.org/2004/OWL/#ontologies oder http://www.co-ode.org/ontologies/

Merkmale von Grail

Grail ist eine objektorientierte Programmiersprache. Abgesehen davon, dass sie allgemeine Eigenschaften von objektorientierten Programmiersprachen besitzt, hat Grail ihre Besonderheiten. Sie ist entwickelt worden, um wirksame klinische Anwenderschnittstellen sowie Modelle der medizinischen Terminologie in Galen Programmen zu unterstützen. Grail fing als rein experimentelle Sprache an. Heutzutage findet man sie auch außerhalb von GALEN, das heißt in anderen Projekten. Sie wird nur für das Galen Common Reference Modell verwendet.

Die Grail-Regeln selbst definieren die Klassifikationsregeln. Die Implementierungen der Grail-Regeln im Galen Terminologie Server erlauben die Aufbau des Galen Common Reference Modells in deklarativer Weise.

Grail-Modelle enthalten in sich die begrenzte Anzahl von elementaren Konzepten, eine Menge von „attributes“ (oder linked Relationen) und eine Reihe von Regeln, die bestimmen, wie individuelle Elemente zusammengesetzt werden können, um die komplexen Konzepte zu erstellen.

Grail ist eine methodologische Sprache und ist analog zur 'T-Box' in Beschreibungslogiken. Eine 'A- box' gibt es in Grail nicht.

Subsumption

Der Begriff Subsumption steht für (*A ist ein Typ von B*) Beziehungen. Man merkt sich, dass *B* in der Hierarchie dem A übergeordnet ist. In Grail wird es mit der Hilfe vom newSub-Operator realisiert und asserted – Subsumption - Hierarchie genannt. Diese Hierarchie kennt man unter dem Name Asserted Taxonomy. Alle Relationen zwischen Konzepten werden ohne vorher definierte Gründe eingegeben.

Es gibt zwei Ausdrucksmöglichkeiten von Subsumption:

- 'assereted subsumption hierarchy' oder 'asserted taxonomy' ;
- formal subsumptions (die Regel lautet - eine beliebige Kategorie *X* subsumiert eine Kategorie der Form ' *X which hasSomeAttribute Y '). Dazu gehören '*Statements, Definitions, und Composite Categories'.

Elementare Kategorien und Attribute sind in die Subsumption -Taxonomie organisiert und als azyklischer Graph dargestellt. Subsumption ist in der Form *E1 ≥ E2 geschrieben und bedeutet „E1 subsumes or is equal to E2”.*

Transitivität

Grail unterscheidet sich von den anderen Beschreibungslogiksystemen durch seine Fähigkeiten in der Beseitigung von Transitivitätsproblemen. Das erlaubt Grail ein sicheres Umgehen mit part-whole - und causal - Relationen. Transitivität wird durch das folgende Basismuster realisiert :

“The part of the part is a kind of part of the whole”.

Anderes gesagt : Ist A ein Teil von B und B ein Teil von C, dann ist A ein Teil von C.

In der Medizin bedeutet es zum Beispiel, dass eine Verletzung eines Körperteils eine Verletzung des Körpers ist.

(The pattern is that *Damage to a part is a kind of damage to the whole*.)

In Grail gibt es gar keine Negationsmöglichkeiten, kein Gleichungssystem, keine Disjunktionsmöglichkeiten.

Grail : Vokabular und seine Ausdrucksmöglichkeiten

Im Laufe der Projektentwicklung wurde von Spezialisten ein Grail-Vokabular erstellt. Mit Hilfe dieses Vokabulars vermeidet man die Missverständnisse in sowohl Definitionsaufklärungen als auch bei Implementierungen. Die wichtigsten davon werden hier aufgelistet.[5]

Categories : werden immer großgeschrieben; z.B. Heart, Bone, Fracture. Sie sind in Klassen zugeordnet. In der Beschreibungslogik werden sie "concept names" genannt.

Attributes : sie verbinden Categories miteinander und entsprechen den Relationen oder Rollen in Beschreibungslogik. Attribute haben die Form **hasX** und die zu ihnen gehörenden "inversen attributes" die Form **isXOf :**

<<<< Tumour hasLocation Arm.
Arm isLocationOf Tumour. >>>>

<<<< DomainAttribute newAttribute hasOwner isOwnerOf allAll manyOne.
Virus which hasOwner Person. >>>>

Statement : sind zwei Categories, verbunden durch Attributes. Die Beschreibungslogik verwendet die gleiche Definition. Es gibt zwei Arten davon, nämlich Necessary und Canonical Forms.

TopCategory : die höhste eingebaute Kategorie; ist ein Startpunkt für alle neuen Kategorien.

DomainCategory : kommt immer nach der TopCategory vor :

5 Appendix 1

```
<<< < TopCategory newSub DomainCategory.
        DomainCategory newSub Anatomy. >>>>
```

NewSub ist ein Grail Operator; er erzeugt elementare Konzepte und gleichzeitig weist er auf die Kategoriezuordnung hin. NewSub drückt die Subsumption - Relationen aus.

Operator *newAttribute* hat die gleiche Funktion wie newSub, wird aber für Attributen verwendet.

Für die Erzeugung von Composite Categories braucht mann den Operator *"which":*

```
<<<< Person which hasSex male. >>>>
```

Ähnlich wie die anderen formalen Sprachen hat Grail noch eine Ausdrucksmöglichkeit wie *"ValueTypes "*. Es sind Konzepte mit verschiedenen Subsumptionsregeln.

Mit der Hilfe des ValueTypes werden Basiskonzepte und ihre Kombinationen beschrieben.

Der Operator < *SymbolicValueType* > ist bereits im System eingebaut.

```
<<<< SymbolicValueType newSub SexValueType.
                SexValueType newSub male.
                SexValueType newSub female.
        SymbolicValueType newSub AgeValueType.
            AgeValueType newSub young.
            AgeValueType newSub old.
        DomainAttribute newAttribute hasAge  isAgeofall.  >>>>
```

Es ist auch möglich diesen Ausdruck in einer anderen Form zu formulieren :

```
<<<< Person which  <hasSex male hasAge young.>>>>
```

Criteria : Begriff, der eine Kombination von Attributen und von Kategorien beschreibt.

Naming : ist ein Komplex aus mehreren Kategorien :

```
<<<<< Virus which hasOwner
      (( Person which hasAge old)
       which hasSex male) . >>>>
```

Damit ist gemeint, dass es ein Virus gibt, das ein alter Mann in sich trägt bzw. Ein alter Mann ist ein Virusträger. Die Interpretation dieses Konzeptes hängt von unseren linguistischen Kenntnissen ab.

Ausgehend von Naming-Operator erlaubt Grail sogenannte *Nick-Namen* von Kategorien:

```
<<<< (( Person which hasSex male) which
        hasAge young ) name Boy. >>>>

<<<< ((Person which hasSex female) which hasAge old)
        name Woman. >>>>
```

Nach Naming können wir die Definitionen Boy und Woman für die neue Präsentation von komplexeren Kategorien verwenden.

Cardinality : Eigenschaften, die beschreiben, wie viele Kategorien man durch Attribute miteinander verbindet: oneOne; manyMany; allAll; manyOne.

Da die Konzepte in Kategorien und Individuen unterteilt werden, gibt es unterschiedliche beschreibende Statements für beide Varianten:

- deskriptive statements über Kategorien werden üblicherweise als *"necessary statements"* bezeichnet;
- deskriptive statements über Individuen werden als *"facts"* bezeichnet.

Necessary statements tragen das Schlüsselwort necessarily, facts - das Schlüsselwort really :

<<<< Femur necessarily isComponentOf Thigh. >>>>
<<<< HollowObject necessarily defines Cavity. >>>>
<<<< John really isObservedToHave Diabetes. >>>>

Beschränkungen (Begriff der Sanktionierung)

Sie sind in Grailsyntax eingebaut, d.h. kein einziges Konzept kann ohne Beschränkungen erzeugt werden. Mit Hilfe dieser Sanktionierungen wird entschieden, was ausdrücklich vernünftig und sinnvoll ist; zum Beispiel die Ausdrücke wie „abscess of blood“ (Abszess des Bluts) oder „fractured eyebrow“ (gebrochene Augenbraue) sind nicht erlaubt !!!

Um solche Ausdrücke innerhalb des medizinischen Modells zu vermeiden, werden zwei Beschränkungsarten eingefügt. Sie werden *"grammatical" und "sensible"* genannt. Die "grammatical" Einschränkungen regeln den Satzaufbau und ermöglichen damit eine Abfrage innerhalb des Systems. In diesem Sinne wird die formale Beschreibung aller vorhandenden Konzepte oder Prozeduren realisiert. Im Gegensatz zu den grammatischen Einschränkungen kontrollieren die sensiblen Einschränkungen den Sinn oder besser gesagt die Logik des Ausdrucks.

<<<< Person grammaticaly hasSex.
Person sensibly hasSex.>>>>

Zusammenfassung

Im breiten Sinne ist GRAIL eine Beschreibungslogik, d.h sie gehört zu den Formalismen, die eine präzise Notation für die Informationsdarstellung erstellen.

Im Vergleich zu den anderen Beschreibungslogiken hat Grail die Eigenschaften der part-whole Relationen und der einfachen Regeln. Aber es fehlt ihm die Eigenschaft, mit Zahlen umzugehen, was eigentlich für medizinische Terminologie keine große Role spielt.

The Common Reference Model ist schon längst komplett fertig, aber man hat vor, es weiter mit den anderen medizinischen Bereichen auszufüllen.

Die Entwicklung von OpenGalen war erforderlich, da es damals keine formale Sprache bzw. System oder Techniken gab, die mit den medizinischen Kategorien umgehen konnten.

Ein weiterer Vorteil des OpenGalen - Projekts ist die Web - Semantikentwicklung und vorgeschrittenes Implementieren, welches die anderen Programmiersprachen und Ontologien teilweise ausgeprägt hat .[6]

6 http://www.openclinical.org/descriptionlogics.html

Appendix 1: Die wichtigsten GRAIL Operatoren und ihr Syntax

Category newSub string *e.g.* *BodyPart newSub Stomach*	Creates a new elementary category with the name *string* subsumed by *Category*
Category1 addSuper Category2	Causes *Category1* to subsume *Category2*. Both *Category1* and *Category2* must be elementary.
Attribute newAttribute String1 String2 cardinality	Creates a new attribute with name *string1* and inverse named *string2* with cardinality *cardinality. cardinality* is one *oneOne, oneMany, manyOne* or *manyMany.*
Category necessarily Attribute Value *e.g.* *Hand necessarily isPartOf UpperExtremity*	Enters a bidirectional necessary statement thereby adding the criterion *Attribute-Value* to the essential criteria set of *Category* and *inv Attr-Category* to the essential criteria set of *Value*
Category topicNecessarily Attribute Value *e.g.* *HollowObject topicNecessarily defines Cavity*	Enters a unidirectional necessary statement thereby adding the criterion *Attribute-Value* to the essential criteria set of *Category*.

Appendix 2 : Translation Guide

Die folgenden Beispiele sind die formale Übersetzung von der einigen wichtigsten Grailimplementierungen in OWL-Form .

```
<!--TopCategory newSub [ClassA ClassB ClassC]. -->
<owl:Class rdf:about="#ClassA">
      <rdfs:subClassOf rdf:resource="#TopCategory"/>
</owl:Class>

<owl:Class rdf:about="#ClassB">
      <rdfs:subClassOf rdf:resource="#TopCategory"/>
</owl:Class>

<owl:Class rdf:about="#ClassC">
      <rdfs:subClassOf rdf:resource="#TopCategory"/>
</owl:Class>

<!-- Attribute newAttribute RoleA inverseRoleA allAll manyMany. -->
<owl:ObjectProperty rdf:about="#RoleA">
      <rdfs:label>RoleA</rdfs:label>
      <owl:inverseOf rdf:resource="#inverseRoleA"/>
      <rdfs:subPropertyOf rdf:resource="#Attribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#inverseRoleA">
      <rdfs:label>inverseRoleA</rdfs:label>
      <rdfs:subPropertyOf rdf:resource="#InverseAttribute"/>
</owl:ObjectProperty>

<!-- ClassA grammatically RoleA ClassB. -->
```

```
<owl:AnnotationProperty rdf:about="#G-SANCTION-RoleAAP"/>
<owl:Class rdf:about="#ClassA">
        <G-SANCTION-RoleAAP>ClassB</G-SANCTION-RoleAAP>
</owl:Class>

<!-- ClassA sensibly RoleA ClassB. -->
<owl:AnnotationProperty rdf:about="#S-SANCTION-RoleAAP"/>
<owl:Class rdf:about="#ClassA">
        <S-SANCTION-RoleAAP>ClassB</S-SANCTION-RoleAAP>
</owl:Class>


<!-- ClassA necessarily RoleA ClassB. -->
<owl:Class rdf:about="#ClassA">
      <rdfs:subClassOf>
<owl:Restriction>
        <owl:onProperty rdf:resource="#RoleA"/>
              <owl:someValuesFrom rdf:resource="#ClassB"/>
                    </owl:Restriction>
                          </rdfs:subClassOf>
                                </owl:Class>

                                   <owl:Class rdf:about="#ClassB">
                                  <rdfs:subClassOf>
                                 <owl:Restriction>
                               <owl:onProperty rdf:resource="#inverseRoleA"/>
                          <owl:someValuesFrom rdf:resource="#ClassA"/>
                  </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>

<!-- ClassA extrinsically RoleA ClassB. -->
<owl:AnnotationProperty rdf:about="#RoleAAP"/>
 <owl:Class rdf:about="#ClassA">
              <RoleAAP>ClassB</RoleAAP>
</owl:Class>

<!-- ClassA extrinsically RoleA 'a piece of text'. -->
<owl:AnnotationProperty rdf:about="#RoleAAP"/>
<owl:Class rdf:about="#ClassA">
             <RoleAAP rdf:datatype="&xsd;string">a piece of text</RoleAAP>
</owl:Class>

<!-- ClassA extrinsically RoleA 2009. -->
<owl:AnnotationProperty rdf:about="#RoleAAP"/>
<owl:Class rdf:about="#ClassA">
           <RoleAAP rdf:datatype="&xsd;int">2009</RoleAAP>
</owl:Class>

<!-- ClassB grammaticallyAndSensibly RoleA ClassC. -->
<owl:AnnotationProperty rdf:about="#G-SANCTION-RoleAAP"/>
<owl:Class rdf:about="#ClassB">
            <G-SANCTION-RoleAAP>ClassC</G-SANCTION-RoleAAP>
</owl:Class>

<owl:AnnotationProperty rdf:about="#S-SANCTION-RoleAAP"/>
<owl:Class rdf:about="#ClassB">
```

```
            <S-SANCTION-RoleAAP>ClassC</S-SANCTION-RoleAAP>
</owl:Class>

<!-- (ClassA which RoleA (ClassB whichG RoleA ClassC) )name ClassD. -->
<owl:Class rdf:about="#ClassBwhichRoleAClassC">
  <owl:equivalentClass>
     <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                 <rdf:Description rdf:about="#ClassB"/>
                      <owl:Restriction>
                     <owl:onProperty rdf:resource="#RoleA"/>
                              <owl:someValuesFrom rdf:resource="#ClassC"/>
                                   </owl:Restriction>
                                           </owl:intersectionOf>
                                               </owl:Class>
                                                       </owl:equivalentClass>
                                                            </owl:Class>
                                                <owl:AnnotationProperty rdf:about="#SANCTION-LEVELAP"/><owl:Class
                                               rdf:about="#ClassBwhichRoleAClassC">
                          <SANCTION-LEVELAP rdf:datatype="&xsd;string">GRAMMATICAL</SANCTION-LEVELAP>
                    </owl:Class>
                   <owl:Class rdf:about="#ClassD">
            <owl:equivalentClass>
          <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
   <rdf:Description rdf:about="#ClassA"/>
<owl:Restriction>


<owl:onProperty rdf:resource="#RoleA"/>

       <owl:someValuesFrom>
              <owl:Class rdf:about="#ClassBwhichRoleAClassC"/>
                    </owl:someValuesFrom>
                         </owl:Restriction>
                                 </owl:intersectionOf>
                              </owl:Class>
                        </owl:equivalentClass>
                     </owl:Class>
```