


Overall Progress: 

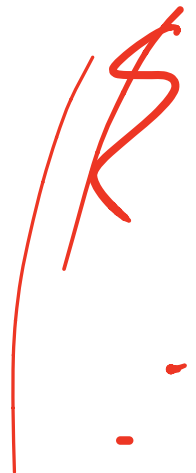
Bachelor's Thesis

Student Consulting Organizations

A Domain Ontology

Felix Förtsch
3708438

01.04.2020



Leipzig University



Abstract
















This work develops a domain ontology for Student Consulting Organizations (SCOs). The model declares the domain knowledge and defines its vocabulary. It contains the information necessary to establish or run such an organization in a university context. Additionally it allows for optimization in existing organizations and contributes to cooperation between SCOs by organizing the existing knowledge. It maximizes the use of vocabularies, relations, and classes from established ontologies like Friend of a Friend (FOAF), Financial Industry Business Ontology (FIBO), General Formal Ontology (GFO), and GIST (GIST) to link the domain knowledge into a bigger context. The main resource of the developed ontology are SCOs from Germany, but the concepts can be transferred and made applicable in a wider area.




Formatting

- Hyperlinks are embedded and clickable in the PDF. They are marked with an arrow and a light blue border: \hookrightarrow Hyperlink
- Everything related to the ontology implementation, such as references to classes or relations, is written as **typewriter text**.
- Relations are written in camelCase: **subclassOf**
- Classes are bold, capitalized, and use Snake_Case: **Awesome_Class**
- Name spaces may be added to a class for clarification; they are separated by a colon: **namespace:Class**

Leseanleitung

Contents

1. Introduction		1
1.1. Motivation		1
1.2. Goal and Scope of the Work		2
1.3. Deliverables		2
1.4. Out of Scope		2
1.5. Use Cases and Outlook		2
2. Preliminaries		4
2.1. Methodology for the Development of the Ontology		4
2.1.1. Research Phase		5
2.1.2. Analysis and Synthesis Phase		6
2.2. Definitions		7
2.2.1. Vocabulary		7
2.2.2. Domain Ontology		7
2.3. Related Work and Classification		10
2.3.1. Top-Level Ontologies		10
2.3.2. Upper-Domain Ontologies		10
2.4. Principles and Assumptions		11
2.4.1. Keeping Things Simple		11
2.4.2. The Unique Name Assumption		11
2.4.3. Open World Assumption		11
2.4.4. Content Completeness Problem		12
2.4.5. Time		12
3. Ontological Aspects in the SCO Domain		13
3.1. Context and Context Switches		13
3.1.1. Domain Context and Sub-Contexts		13
3.1.2. Contextual Ambiguity		14
3.1.3. Conclusion: Context in this Work		15
3.2. Social Constructs		15
3.2.1. General Implementation		15
3.2.2. Roles in the Organizational Context		18
3.2.3. Roles in the Project Context		20
3.2.4. Conclusion: Human Beings in this Work		21
3.3. Processes		21
3.3.1. Implementation in Related Ontologies		22
3.3.2. Structure of the Class Hierarchy		23
3.3.3. Ordering of Processes		23
3.3.4. Discreet Events and Liquid Processes		24
3.3.5. Legal Requirements in the Processes		24

3.3.6. Conclusion: Processes in this Work	24
4. Student Consulting Organizations: The Ontology 	26
4.1. Scope of the Domain 	26
4.1.1. What is the domain that the ontology will cover?	26
4.1.2. For what we are going to use the ontology?	26
4.1.3. For what types of questions the information in the ontology should provide answers?	26
4.1.4. Who will use and maintain the ontology?	27
4.2. Classes	27
4.2.1. Agent	27
4.2.2. Document	27
4.2.3. Processes	27
4.2.4. Projects	27
4.3. Relations	27
5. Conclusion	29
6. Further Research	30
A. Appendix	31
A.1. Term Enumeration	31
A.2. Diagrams	32
A.2.1. General Diagrams	32
A.2.2. Process Diagrams	34
A.3. Glossary	36
A.4. Bibliography	37
A.5. Dictionary Definitions	40
A.6. Ontology Import Links	42
A.7. Acknowledgments 	42

1. Introduction

1.1. Motivation

SCO¹ are student-run consulting businesses, that focus on teaching their members essential business and life skills exceeding the theoretical knowledge from university. They are very similar to small to medium consulting businesses, but are run and organized—most of the time exclusively—by students. And even though the concept is not universally known, these kind of organizations exist worldwide and have a history dating back to at least 1967². Germany has two different umbrella organizations for SCOs with more than 60 member organizations.³

known

But as far as we know, there hasn't yet been any effort to collect and compose the existing domain knowledge of German SCOs in a publicly available and usable form. We consider this an important task, since it is a contribution to prevent knowledge loss that is inherent in the dynamics of these organizations: the majority of the staff are students and thus their consulting career is inherently linked to their university career:

1. The career is time-bound to the duration of the education. A bachelor's degree in Germany averages 7,5-7,6 semesters and a master's degree 4,2-4,5 semesters, which adds up to a total of 11,7-12,1 semesters or ca. six years. [3] This frames the available time for the transfer of the domain knowledge.⁴
2. The career is ~~in~~ parallel to the curriculum. From our experience, freshmen that decide to join student organizations typically do so at the beginning of their second or third semester, after they got acclimated with the workload of their university classes. Since students usually participate in parallel to their education—and the focus is typically on the education—, they have to manage their time accordingly, which reduces time spent with the SCO. Furthermore, students may have other (e.g. personal) interests that compete with the same time budget.

The reasons above reduce the available time for knowledge transfer and persistence and make these problems harder. Many SCOs have worked on and developed solutions to help with this problem. Some of them are informal, some formal in nature. For example: One particular organization we worked with, ↔Hanseatic Consulting (HC), used process methodology to document a lot of their knowledge. Another ~~we~~ we worked with, ↔Campus Inform (CI), was focusing on lectures and training sessions.

one

However, the majority of available domain documents are highly individualized and miss the necessary level of abstraction to make them directly applicable to other SCOs. But even though

¹Also known as ↔Junior Enterprises in some parts of the world.

²The founding of ↔Junior ESSEC in France.

³↔Bundesverband Deutscher Studentischer Unternehmensberatungen (BDSU), ↔Junior Consultant Network (JCNetwork)

⁴There sometimes are also PhD students, but they can be considered outliers and are atypical.

every SCO is organized slightly differently than the next, uses different vocabulary and each has their individual culture, they all share the idea of teaching consulting and project work to their members. Since they aim for the same goal, they are very similar at their core.

Therefore we try to contribute a more general model in the form of an ontology that tries to combine the domain knowledge, vocabulary, and common concepts.

1.2. Goal and Scope of the Work

The goal of this work is the description of one abstract SCO. It extracts the available implicit expert knowledge, links it with related work, and transforms it into explicit knowledge by using an ontology as its vehicle. It defines common classes and relations required to describe such an organization using domain vocabulary. Additionally it provides terminology explanations, background knowledge, and links into other ontologies where it is sensible.

1.3. Deliverables

The output of this work are two documents:

1. This thesis as a documentation and explanation of the ontology development process including but not limited to: methodology, background information, decisions in regards to the ontology, etc.
2. The ontology document as a representation of the domain knowledge.

The ontology is publicly hosted and is freely available to any interested parties, such as the umbrella organizations, SCOs, or students.

1.4. Out of Scope

1. This work is not a thesaurus and not a documentation about a specific Student Consulting Organization.
2. No diagram for individual orgs
3. The ontology will not include the individual project process, since projects differ vastly between each other and more general ontologies and frameworks for projects already exists.
4. No model for projects

1.5. Use Cases and Outlook

The main use case of this work is supporting SCO idea by the documenting the domain knowledge and helping others to better understand it. We are trying to design the ontology in a way to enable its use for everyone: From the layperson to the expert. It should be possible to support the founding process of a new SCO, as well as optimize an existing SCO by comparing it to the model.

Furthermore, we hope to aid the advance of the SCO idea, by creating a computer-readable ontology that can enable the creation of software projects.

2. Preliminaries

Ontologies have many applications in various fields. They are used in artificial intelligence research, database design and integration, semantic web, and many more. [6, p.1] One of these applications is knowledge representation.

“**Knowledge Representation** is the field of Artificial Intelligence that focuses on the design of formalisms that are both epistemologically and computationally adequate for expressing knowledge about a particular domain.” [2, p. XV, Preface]

This work is concerned with the knowledge representation of one particular domain: SCOs. The following section describes how we approach the development of this particular ontology, define the necessary vocabulary and relations between the terms, and structurally explore the domain.

2.1. Methodology for the Development of the Ontology

The primary goal of this work (see section 1.2) is the creation of a particular domain ontology. Our language of choice for this task is Web Ontology Language (OWL). It is an ontology language developed by the consensus-driven and well respected World Wide Web Consortium (W3C). [2, p. 206] It is widely used and offers very good tooling in the ontology editor \hookrightarrow *Protégé*. It is built and maintained by ontology researchers of *Stanford University*. [20] Furthermore the research group provides a ontology development methodology [21], that we can build upon. It involves the following steps:

- (1) Determine the domain and scope of the ontology,
- (2) consider reusing existing ontologies,
- (3) enumerate important terms in the ontology,
- (4) define the classes and the class hierarchy,
- (5) define the properties of classes-slots,
- (6) define the facets of the slots, and
- (7) create instances.

It is important to note, that even though these steps look like they should be performed sequentially, this is not the case. Instead, the ontology starts out as a draft and is refined during development [21, Section 3, Introduction], following the iterative approach, that is common for ontology development. [28, p.158, section 1.5.1] This quickly becomes apparent during the process of answering the suggested *Competency Questions* to (1) determine the domain and scope of the ontology [21, Section 3, Step 1] and taking into account (2) existing ontologies. And this also is true for steps (3) to (6). Therefore the steps are grouped together to make the overall structure of this work easier to follow.

The phases of the methodology are discussed in more detail in the following two sections and group the proposed steps as follows:

1. Steps 1 and 2 are performed during the *Research Phase*.
2. Steps 3 to 6 during the *Analysis and Synthesis Phase*.

The last step, (7) the creation of instances, is omitted in this work. It is only really relevant if the ontology is used to describe one specific SCO. [CN] However, this ontology is operating on a higher level of abstraction, trying to describe a more general case.

2.1.1. Research Phase

To our understanding, the main goal of the first part of the methodology is the creation of a foundation for the ontology: It needs to have a clearly defined scope. Additionally the recommended reuse of other ontologies helps creating a web of linked knowledge and reduces the amount of duplicate work.

To find a starting point for data collection and identify existing ontologies, we take an intuitive first look at SCOs and their driving factor:

The Idea of Student Consulting Organizations

Selecting a career is a very difficult and important choice in a young persons life. University education is closely linked to this choice and entering a specific field often requires a specific degree (e.g. to become a lawyer, a student has to pass the bar exam).

Most universities know this and have set up dedicated offices to offer career advice to their students. They not only help picking a fitting course of studies at the beginning of a university career, but also help the students to aim for a fitting job.

Doing an internship with a company working in the field the student is interested in, is a widespread recommendation. It allows for a glimpse into the profession as well as gathering work experience.

SCOs offer an option to investigate a career in business consulting, as well as learning the associated skills and getting paid in the process. They offer the students a way to learn about concept like project based work—the modus operandi of consulting companies—, e.g. project planning and management, as well as structuring and presentation of information.

Consulting is a very diverse field of work. Since consulting can be applied to any field of business, it can be used as a stepping stone into a career.

Observing this intuitive perspective, we can see, that SCOs are connected to other knowledge domains in various ways: They are a type of social organization and thus are driven by people and processes. Organizations and in extension their processes have actors with responsibilities. This is a hint that the concept of roles might ~~to~~ be a part of the ontology. SCOs can be generally considered a form of business and therefore business aspects have to be taken into account. The fact that they do consulting work, creates a connection into the domain of (business) consulting and the domain of projects, since consulting work is project based.

This intuitive approach generates a the starting point for the research:

- Previously developed ontologies in related domains, e.g. consulting, project management, educational organizations.
- Available domain knowledge, e.g. process documentation of HC and CI.

- Personal expert domain knowledge and peer-review by other SCO members.

Furthermore it implies some more general research topics:

- Implications of other general, upper-level, and top-level ontologies, e. g. GFO, Basic Formal Ontology (BFO), GIST.
- Theory of description logic and ontologies, e. g. modeling of roles and processes.

Defining the scope of the ontology is the formal step that concludes the *Research Phase*. This work accomplishes this by answering the Competency Questions. Since the questions can be considered a part of the ontology, they and their corresponding answers can be found as part of the ontology in section 4.1.

2.1.2. Analysis and Synthesis Phase

The majority of this work happens during the Analysis and Synthesis Phase. Its goal is the review, interpretation, and structuring of the collected data; ultimately generating an ontology in the target format OWL.

Based on the Protégé-methodology, the first two steps of this phase are: (3) the creation of an enumeration of terms that are relevant for the domain. And (4) the translation of the terms into the backbone of every ontology: the class hierarchy. Both are rooted in the results of the previous phase and further supplemented by expert knowledge.

At the core of this thought process is the conversion of available implicit knowledge into explicit knowledge. This is a difficult task, since typically the implicit knowledge is not easily available; it's considered an aspect of knowledge engineering. [17, p.30–31] To help with it, we introduce a creative step: We start with a brainstorming to create a domain vocabulary collection in the form of a *word cloud*. This simple first step involves writing down all the terms that might have something to do with the ontology. We then can transition this word cloud to a *word graph*, by using the terms as vertices and implement associations between terms (e. g. connected ideas or concepts) using the edges. We try to keep the word graph as simple as possible by focusing on the important connections and use existing vocabulary to prepare the links into other domains that will be done in the later stages of development.

To progress from the word graph to the more rigorous class hierarchy, we transcribe the vertices into a first-draft/skeleton class hierarchy—using the Protégé editor—, starting with the most influential concepts. These can be identified by the amount of edges connecting them to other concepts; more connections indicate higher influence. Furthermore we can identify and assign trivial sub-classes during that process, by evaluating the quality of the edge connections. Fewer connections might indicate a more direct relationship between two terms. After these steps, the draft hierarchy contains mostly high-level classes and trivial sub-classes (e. g. high-level class **Process** and all the identified processes as trivial sub-classes). It can then be further modified, refined, and polished by adding clarifications, delimitations, definitions, and descriptions to all terms as well as relations within the class hierarchy.

During this development process we can identify the aspects that play the most important role for our domain. We document and discuss the challenges and our approach to solving them in detail in section 3.

The output of the final steps is the first major version of the ontology in the form of an OWL file, which completes the *Analysis and Synthesis Phase* and also the second deliverable of this work.

2.2. Definitions

Depending on the ambiguity and complexity of concepts, natural language can quickly become a limiting factor for precision. Definitions are a tool to avoid confusion that arises from unclear semantics. To ensure a common understanding, this section discusses some key terms and their usage throughout this work.

2.2.1. Vocabulary

Commonly, the term *vocabulary* is used to describe the collection of all words one specific person knows. But this already implies that one person's vocabulary can be different to someone else's. Furthermore, the word vocabulary itself is also overloaded, as the dictionary definition (see appendix A.5) shows. Depending on context the described collection changes. [16]

In the context of ontologies it is used interchangeably with *alphabet* and *signature*. On an abstract level it simply describes a set of symbols. [15, p. 46] Transferred to this work, it is the collection term for all class (see section 2.2.2.1) and object property names (see section 2.2.2.2.1), which are mainly words from the English language in conjunction with the underscore (“_”) as delimitation symbol. The word choices draw inspiration or are translated from the HC process documentation and our experience with different SCOs, as well as the business world. Some of these words are not special to the domain: **Organization**, for example, is not only used in many different related ontologies, but also very common in natural language.

To ensure adequate precision, each entry in the vocabulary has a few guaranteed properties (see section 2.2.2.2.2).

2.2.2. Domain Ontology

The term *domain* is the easy part of this definition, since it intuitively and simply refers to the knowledge area it describes. [15, p. 7] This is close to the dictionary definition (see section A.5), where every listed sense has a delimiting meaning to it. [16]

However, at the time of writing, an exact definition of the term *ontology* is hard to come by. There is no unified understanding across the sciences. [10] Numerous authors tried defining the term as shown by Loebe [15, p. 4-6] and Gomez et al. [6, p. 6-9], but to our knowledge there has not yet been a conclusive definition. Since it's not our objective to end this discourse, we will not engage in a detailed analysis. Instead, we construct a *working definition* by illuminating different aspects that are important for this work's goal of representing a knowledge domain.

The influential paper [25, p. 9] [15, p. 4] “*A Translation Approach to Portable Ontology Specifications*” by Thomas R. Gruber—a paper that has been cited more than 19.000 times according to Google Scholar—contains the following definition:

“An **ontology** is an explicit specification of a conceptualization.” [7, p. 1]


Since it is on the more abstract sides for definitions, it is often used as a first step of narrowing down the intended meaning. Baader builds on it and proposes this slightly refined definition:

“In computer science, an **ontology** is a conceptual model specified using some ontology language.” [2, p. 205]

This gives us two anchor points to attach it to our work: 1. Our ontology is a conceptual model of a SCO. We try to adhere as best as possible to the suggestions from the Good Ontology Guidelines: being formal, using explicit specifications, and being adequate for the domain it represents. [25, p. 10] 2. Our ontology language is OWL.

To works further towards our secondary goal of providing an ontology that is usable for software projects, we extend our requirements to the definition for domain ontologies from Jean:

“A **domain ontology** is a formal and consensual dictionary of categories and properties of entities of a domain and the relationships that hold among them.” [13, p. 240]

We try to achieve formality by adhering to OWL standard and using the *HermiT* reasoner during development. To aim for consensus, we prepare the ontology in way so that it can be peer reviewed in the future. 

Similar to the general discussion about ontologies, there is an ongoing debate on which term best represents the elements of an ontology. We are following the Protégé naming convention and are calling the collection of elements *entities*. Furthermore we use the term *Classes* to represent things, *Object Properties* to represent relationships between things, and *Annotation Properties* to describe both more closely with additional comments.

2.2.2.1. Classes and Class Hierarchy

Classes are the most basic building blocks of an ontology. Each class tries to capture and describe an entity of the knowledge domain. The name of the class creates a connection between the entity within the ontology and the thing it tries to describe. It is supplemented by additional information via its annotation properties (see section 2.2.2.2.2). The classes are aggregated and structured in a meaningful way by the *class hierarchy*.

The class hierarchy is a tree where the children of a node have the built in relation (see section 2.2.2.2.1): `subclassOf`. There typically exists a root element called **Thing**, which all classes are `subclassOf`. As pointed out before, this work follows the OWL standard and hence we use `owl:Thing` as the root element; it has no properties. The built in relation gives additional meaning to classes in the hierarchy. `subclassOf` implies that a class isA type of the class it sub-classes. It is therefore key, to only introduce a sub-class relationship, if it is correct for the representation of the domain.

This has implications for our ontology. For example, it is the reason for the different structuring of **Agent** and **Process**: The former sub-domain can make use of the sub-class relation, whereas the latter cannot (see section 3.3.2).

2.2.2.2. Properties

The classes and the class hierarchy give an ontology its basic structure; they define what things exist in the world the ontology describes. The *properties* bring this world to life. A property is a binary relation: It can be used to assert facts about a class, describe it in more detail, and to establish relationships between classes. [27] We primarily use two types of properties in our work: *Object Properties* and *Annotation Properties*. To make them more distinguishable we also introduce and use their colloquial references *Relationships/Relations* and *Annotations*.

2.2.2.2.1. Object Properties a.k.a. Relationships a.k.a. Relations To express relationships within the knowledge area, we use object properties. The meaning of a thing is not only defined by its name and description, but also by the connections it has to other things. For example: A **Car** by itself conjures a certain image in the readers mind. Adding the relation **hasBrandName Mercedes-Benz** changes the definition of the car by stipulating an additional restriction.

There are different ways of structuring object properties. The one end of the spectrum holds generalized relations, like **hasA** or **isPartOf**. This type of relationship can be applied to many different classes, because the semantics lies in the connection between the classes, instead of the relation itself. This type of generalized relation can be applied in different contexts. The **isPartOf** relation, for example, can be used in different ways: A **Wheel isPartOf a Car**; but also: A **House isPartOf a Neighbourhood**.

On the other end of the spectrum are highly specific relations that carry a lot of meaning by themselves and thus can not be applied to other contexts easily. For example: A **Person isMemberOfTheMajorityCouncil Member_Council**.

As with many aspects of ontology development, it is important to strike the right balance and achieve the correct level of abstraction. In this work, we use a bottom-up approach. We start with specific relations and generalize them, if that is possible.

As already mentioned before, there is one special object property that is built into the class hierarchy: **subclassOf**. It would be possible to link a child class to its parent and expresses that the child is of the same type as the parent. For example: In our ontology, the class **Agent** is sub-classed by **Person** and **Organization**. This implies that both **Person** and **Organization** are Agents. Everything a Agent can do, a **Person** or an **Organization** can also do.

2.2.2.2.2. Annotation Properties a.k.a. Annotations We use annotation properties to add additional information to entities in our domain. They can be thought of simple key-value pairs that are attached to a specific entity. The key might be defined within the scope of the ontology or may be sourced from knowledge organization conventions like Resource Description Framework Schema (RDFS) or Simple Knowledge Organization System (SKOS). If a key belongs to such a schema, it uses the corresponding prefix, separated by a colon “:” (e.g. **rdfs:label**). The value holds the additional information about a class.

Each class of our domain has two guaranteed annotation properties: 1. A **rdfs:label** to define its name in both English and German and 2. a **rdfs:comment** in English, to describe the class more closely. Furthermore we provide one or more **skos:example** where examples help describing the class more clearly.

2.3. Related Work and Classification

As already discussed previously, the SCO domain touches on various other knowledge areas. Related source material could be found in other ontologies, Project Management (PM) models and concepts, organizational theory, process documentation and models, established vocabulary, and so forth. The challenge lies in identifying the most useful information and merge the input from many different sources to construct the ontology.

This is made harder by the vastness of some related fields. PM, for example, is a central topic for our domain and it is easy to find many different books, theories, and models about it on the market. It is also an complex, overarching, and very general subject matter. Part of its domain are concepts of complex nature like time, problem analysis, and project structuring. If we were to focus too much on this aspect within our ontology, it would easily dominate it and the ultimate goal of our ontology would be missed. It is therefore imperative to identify the correct level of abstraction and connect the related work in an adequate manner.

This section discusses the related work and shows how we are applying it to the SCO ontology.

dctterms¹

To reflect on this fact in the ontology, we try to merge the

2.3.1. Top-Level Ontologies

1. What is a TLO (Definition)?
2. What can we use them for in this work? -> Processes, Roles, Time?
3. What are popular TLOs and what are their strengths and weaknesses?
 - a) BFO [26]
 - b) DOLCE
 - c) GIST
 - d) BPMN [22]
 - e) GFO [9]

2.3.2. Upper-Domain Ontologies

1. What is a UDO (Definition)?
2. What can we use them for in this work?
3. What are popular UDOs and what are their strengths and weaknesses?
 - a) OWL-S
 - b) SUMO
 - c) FIBO
 - d) Description of a Project (DOAP) <https://github.com/ewilderj/doap>
 - e) Schema.org Ontology (Schema.org): not really ideal, but useful for general concepts like Person or Organization
 - f) FOAF is close to schema, link to dublin core: "dct:Agent"

¹dctterms is used in the **FOAF** rdf file, dct is used in the **FOAF** documentation.

g) Enterprise Ontology <http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html>

Dublin Core's notion of Agent is much like FOAF's; Dublin Core says "A resource that acts or has the power to act.", we say "things that do stuff". As nobody has provided a counter-example of something fitting one definition but not the other, we say here that foaf:Agent stands in an 'equivalent class' relationship to dct:Agent (and vice-versa)." [4, External Vocabulary References]

2.4. Principles and Assumptions

2.4.1. Keeping Things Simple

Polysemy Paper [1] Keep It Stupid Simple (KISS)

keep it as simple as possible (e. g. *contract* and *contract document* can be considered two distinct things, but this distinction is not important for the domain knowledge – maybe add a relation "has document"?)

Example: A contract is a document that captures a business agreement. The word "contract" can refer to the immaterial agreement between the parties, but it can also refer to the document itself. Depending on the use case of the ontology it might be useful to separate these two things.

However, in this ontology the goal is to keep it a simple as possible, since the potential users of this ontology are not necessarily experts.

2.4.2. The Unique Name Assumption

[23, p. 299]

2.4.3. Open World Assumption

Closed world: if a value is not stated, it's false [23, p. 299] [19, p. 388]

Open world: if a value is not stated, it's unknown [23, p. 417]

OWL is based on the OWA [19, p. 389]

2.4.4. Content Completeness Problem

The *Content Completeness Problem* states, that an ontology is incapable of containing all the concepts relevant to its users.

Developing an ontology involves thinking about the correct level of abstraction and making choices on what to focus on.

https://en.wikipedia.org/wiki/Content_completeness_problem As is true for any domain ontology [CN], the content completeness problem exists for this ontology as well.

active/passive content completeness

bewusst weggelassen:

2.4.4.1. Consulting Topics

The main goal of consulting companies is in their name: consulting. They are a source of expertise and knowledge and can be employed as an option to solve a difficult problem at hand. The problem space of consulting companies is vast; examples are: Digitization, Human Resources, Knowledge Management, Market Research, Marketing, Corporate Strategy, etc. These topics are obviously part of the consulting domain. However, they are deliberately omitted, since their exploration would exceed the scope of the work.

2.4.4.2. IT and Communication Systems

IT systems are an essential part of modern business and there are companies where these systems are integral to everything (e. g. AI companies). However, in the context of a consulting company they are mainly used to support, supplement, and optimize the already existing processes. Hence, a model of an IT system would not contribute in a meaningful way to the ontology.

2.4.5. Time

Implement time abstract -> only needed for processes before/after no absolute time

BFO GFO have complex implementations of time that are not needed in this ontology.

3. Ontological Aspects in the SCO Domain



At the center of our domain lies a social structure. It is driven by processes and interactions that involve people. Such social constructs are inherently complex. For example: The same individual can often act in multiple capacities, actors can be part of different groups and these groups can have different degrees of formalization, the contexts of interactions are dynamic and fluid, etc. The challenge lies in modeling the domain in a way that it is adequately represented, but not too complex to use.

In the following section, we discuss three distinct and noteworthy aspects that we encountered during the modeling of our domain: The impact of *context* and the relevant contexts in this work, *social constructs* and how we work with them, and our model for *processes*.

3.1. Context and Context Switches



Context plays an important role in knowledge engineering. Things can take on a different meaning, depending on which context they are presented. It poses a challenge for any model, because of its inherent complexity and domain specificity [18, p. 1].

To better understand how context influence our work, we start with its definition:

“**Context** is any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity, and state of people, groups, and computational and physical objects.” [5, p. 106]

This means that every entity of an ontology is affected by context. In extension, the same is true for the ontology itself and the user who uses the ontology. Solving the contextual problem is out of scope of this thesis; however, we try to alleviate the challenges that context brings into play.

We focus on two aspects: 1. The description of the domain context and sub-contexts within it. 2. Contextual ambiguity and how we are trying to mitigate its impact.

3.1.1. Domain Context and Sub-Contexts

To define the primary context of our domain, we first have to take a look at *what* exactly we are modeling. As stated in the introduction (see section 1.2), the goal of our work is the creation of an abstract SCO. We consider it a form of *idealized blueprint*, that can later be used to create concrete instances of this type of organization. It is clear that this can happen more than once and

that different SCOs can co-exist and interact with each other, influencing each others contexts. However, we are deliberately **not** modeling all SCOs in one ontology.

Hence, one SCO is the primary and default context of the ontology and that all the classes have to be interpreted and understood in this particular way. If one were to extend our model to incorporate multiple SCOs, it would be necessary to develop dedicated links between instances. This primary context is active for all the classes within the scope.

For SCO the primary goal is the education of its members in the consulting profession. As already pointed out, the central idea is *learning/teaching by doing* by creating project opportunities with clients and enabling the members to work on these projects together. The organization built around this idea is a tool that helps to achieve this goal; the same is true for the projects.

We can identify two different sub-contexts whose interplay describe the inner workings of an SCO: The project and the organization. Everything that happens within an SCO can be attributed ot at least one of these contexts.

Projects are a very goal orientated flexible work structures that can be applied in a wide array of situations. There exist a multitude of concepts, models, and books on the topic and there even exist a dedicated profession that specialists in this kind of work: *Project Management*. The project domain itself is vast and therefore it is out of scope of this work to model the project domain completely. However, since projects are undeniably a major part of the SCO domain, we have to identify the aspects that have an impact on our ontology.

A project is an organizational construct that exists to reach a common goal, as pointed out by the International Organization for Standardization (ISO) definition:

“A **project** is a unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources” [11]

As already pointed out, projects are the tool of choice for educating the SCO members. The organization can be thought of as the central hub for multiple projects: It is the entity that frames each projects life cycle. It ensures sufficient starting conditions, provides external support during a projects duration, and accompanies its finalization.

Hence, it is sufficient for our work to look at the project itself as a black box similar to a function. Our model has to consider the project inputs, e.g. personnel, and its outputs, e.g. documents, as well as a model for interactions between a running project and the organization.

The limitations include:

1. The typical SCO requires project members to be formal members of the organization itself.
2. A member can be part of a project and part of the organizational structure at the same time.

3.1.2. Contextual Ambiguity

Ontology engineering relies on natural language to describe concepts. This work uses primarily English words (see section 2.2.1) for this task. That naturally introduces a form of contextual ambiguity, because people often use words in a flexible manner. [29, p. 7, 1.5.2] It's our goal to

keep the ontology as simple as possible. It is therefore important to find a way to avoid ambiguity as much as possible.

Let's imagine a domain **Dreamworld** that is home to a **Person**. At first glance the concept seems to be intuitive. However, when looking more closely and realizing that in the **Dreamworld** a **Dreamworld:Person hasPart Dreamworld:Wheel**, this concept of a **Person** would suddenly appear odd. It would stray away from the common understanding.

This small example shows: Contexts are part of the concept definition—and natural language comes *pre-loaded* with context. In the case of **Person**, the injected context is the common understanding of what a person is, implied by the English language. We call a context that is automatically assumed in the absence of a context an *Implied Context*.

It is also possible to specify the context of a class in such a way, that ambiguity is impossible. We call such a context a *Defined Context*. One way to impose a defined context, is to avoid common words all together and use cryptic randomly generated word-letter combinations—e. g. TJKY0123—as class or property names. When the name has no meaning, all its meaning comes from its definition and relationships.

However, this approach discards all previous knowledge about a concept. Furthermore, an ontology built this way would be hard to work with for another human being.

Another way is to start with the implied context and add explanations, comments, clarifications, and restrictions to it, to make the meaning as clear as possible; building the required context in the process. This can be furthered by using already existing definitions from pre-defined name spaces and established ontologies (see section 2.3).

We are using the latter approach and try to avoid additionally ambiguity by discussing the relevant contexts explicitly as part of this work, as well as add explanation texts to the ontology.

3.1.3. Conclusion: Context in this Work

3.2. Social Constructs

This section deals with the model for human beings. e. g. as a SCO member or -non-member, as part of a project, as a customer, etc. Aggregation of these actors can occur in different degrees of formalization, e. g. informal meeting of SCO members as friends, a project team meeting, an official meeting of the member council, etc.

Since this is not a domain specific phenomenon, it is sensible to consider how existing and related ontologies (see section 2.3) represent these cases.

3.2.1. General Implementation

Starting with the general model of human beings, FOAF is a very common choice when thinking about representing social structures. It is a well established ontology and referenced multiple times as backbone for social concepts. Its implementation and description are relatively basic: The anchor is the top-level class **foaf:Agent**¹, which is referred to as the class of “things that

¹foaf:Agent rdfs:comment: "An agent (eg. person, group, software or physical artifact)."

do stuff". It is connected to the name space of the Dublin Core Metadata Terms (DCMT) via `equivalentTo dcterms:Agent`. It is sub-classed by `foaf:Group`², `foaf:Organization`³, `foaf:Person`⁴, `Person`⁵, and `schema:Person`⁶. `Person` and `schema:Person` are `equivalentTo foaf:Person`.⁷ `foaf:Person` and `foaf:Organization` are disjoint. `foaf:Group` aggregates any type of `foaf:Agent`. DOAP reuses exactly the same classes as FOAF. It also has the same links to Schema.org and DCMT.

Schema.org implements `schema:Person`⁸ and `schema:Organization`⁹. `schema:Person` is considered `equivalentTo foaf:Person`. This establishes a two-way link between FOAF and Schema.org. `schema:Organization` is sub-classed to accommodate for specialized forms of organizations that are relevant for the use cases schema was developed for, e.g. `schema:Airline`, `schema:NGO`. A collection class like `foaf:Group` does not exist explicitly, but a `schema:Person` as well as a `schema:Organization` can be a `memberOf` an `Organization`.

FIBO uses very similarly named classes with a more complex description. The root class is called `fibonacci:AutonomousAgent`¹⁰. It is sub-classed by `fibonacci:Person`¹¹, representing individual humans. Like in FOAF, this class is disjoint with `fibonacci:Organization`¹². `fibonacci:Group`¹³ exists as a `subclassOf fibonacci:Collection`¹⁴ and is described as collection of `fibonacci:AutonomousAgent`, which in turn is `subclassOf fibonacci:Arrangement`¹⁵. GIST offers the three classes `gist:Person`¹⁶, `gist:Group`¹⁷, and `gist:Organization`¹⁸ as its implementation of the social structure. However, the classes are organized very differently in the hierarchy and use the `subclassOf` relation more extensively compared to e.g. FOAF: To fully extract all information about the class `gist:Person`, its whole class path has to be taken into account. A `gist:Person` is `subclassOf`

²`foaf:Group` `rdfs:comment`: "A class of agents."

³`foaf:Organization` `rdfs:comment`: "An organization."

⁴`foaf:Person` `rdfs:comment`: "A person."

⁵Note: The ontology doesn't offer any description.

⁶See footnote 5.

⁷The link to Schema.org was added in the last update in 2014.

⁸`schema:Person` `rdfs:comment` "A person (alive, dead, undead, or fictional)."

⁹`schema:Organization` `rdfs:comment`: "An organization such as a school, NGO, corporation, club, etc."

¹⁰`fibonacci-fnd-aap-agt:AutonomousAgent` `skos:definition`: "An agent is an autonomous individual that can adapt to and interact with its environment."

¹¹`fibonacci-fnd-aap-ppl:Person` `skos:definition`: "a person; any member of the species homo sapiens"

¹²`fibonacci-fnd-org-org:Organization` `skos:definition`: "a unique framework of authority within which a person or persons act, or are designated to act, towards some purpose, such as to meet a need or pursue collective goals on a continuing basis"

¹³`fibonacci-fnd-org-fm:Group` `skos:definition`: "a collection of autonomous entities"

¹⁴`fibonacci-fnd-arr-arr:Collection` `skos:definition`: "a grouping of some variable number of things (may be zero) that have some shared significance"

¹⁵`fibonacci-fnd-arr-arr:Arrangement` `skos:definition`: "an organizing structure for something"

¹⁶`gist:Person` `rdfs:comment`: "NEGATIVE EXAMPLE: fictional characters."

¹⁷`gist:Group` `rdfs:comment`: "A collection of People. The group may or may not be an Organization. Many organizations consist of groups of people, but that is not a defining characteristic."

¹⁸`gist:Organization` `rdfs:comment`: 1. "A generic organization that can be formal or informal, legal or non-legal. It can have members, or not.", 2. "EXAMPLES: Legal entities like companies; non-legal entities like clubs, committees, or departments.", 3. "NOTE: There are a plethora of different kinds of organizations that differ along many facets, including members, structure, purpose, legal vs. non-legal, etc."

`gist:LivingThing`¹⁹, which in turn is `subclassOf gist:PhysicalIdentifiableItem`²⁰; and both parent classes are carrying additional properties. Similarly `gist:Group` is `subclassOf gist:Collection`²¹ with the limitation of every `gist:Group` `hasMember` some `gist:Person`.

BFO and GFO don't offer any directly usable implementation for this specific problem, since they operate on a different level of abstraction.

When looking at the related work, we make the following observations:

1. The modeling of human beings is concrete and intuitive: It operates on a low level of abstraction. Concepts that are in use by the layperson, e.g. **Person** and **Organization**, are commonly used in reviewed ontologies; except for the top-level ontologies that operate on a much higher level of abstraction and are therefore not concerned with the concreteness of modeling human beings.
2. The class **Agent** represents actors of an action. Sub-classing it gives the model freedom to express what exactly acts: It can be an **Agent**, a **Person**, a **Group**, or an **Organization**. This flexibility makes the models powerful. They can describe general (e.g. somebody) or other agents (e.g. robots) directly via **Agent**, but can also be used more specifically via a sub-class (e.g. a **Person**). Additionally, a group is a collection class that is also `subclassOf Agent` and hence can become an **Agent**—an actor—itself.

As shown above, the classes **Agent**, **Person**, **Organization**, and **Group** are common in the class hierarchies of the related ontologies. Therefore this ontology will use these classes. However, the different ontologies also use different ways of defining classes. Ranging from the very direct and simple way of FOAF, to the very intricate way of GIST. Since this ontology is trying to be as intuitive to use as possible, the more simple approach from FOAF is adapted.²²

After deciding on these basic building blocks, they can be extended according to the additional domain specifications. For example, a **Person** might need further differentiation based on SCO membership status, on business rank for internal organization and career progress, or on organizational roles. But there are other concepts that can involve a **Person**—e.g. “*being a customer of the SCO*”—that could just as easily involve an **Organization**. This observation points to the requirement of a more general approach for modeling these cases: Roles.

¹⁹`gist:LivingThing` `rdfs:comment`: 1. "EXAMPLES: A cat, a mushroom, a tree.", 2. "NEGATIVE EXAMPLES: fictional life forms such as Unicorns or Mickey Mouse.", 3. "NOTE: In the open world, you must assume that it might have since died.", 4. "Something that is now, or at some point in time was, alive and growing."

²⁰`gist:PhysicalIdentifiableItem` `rdfs:comment`: 1. "EXAMPLES: a computer, a book.", 2. "NEGATIVE EXAMPLE: A discontinuous thing like a manufacturing line cannot reasonably have an RFID attached to it, even though its parts are not the same kind of thing as the whole.", 3. "NOTE: You could, at least in principle, put an RFID tag on members of this class. Physical things are made of something. E.g., statues are made of bronze.", 4. "NOTE: In practice, this always means that the parts are not the same kind of thing as the whole."

²¹`gist:Group` `rdfs:comment`: 1. "Any identifiable grouping of instances. For instance, a jury is a collection of people.", 2. "EXAMPLES: A jury is a group of people, a financial ledger is a collection of transaction entries; a route is an (ordered) collection of segments."

²²This decision is a direct application of the KISS principle (see section 2.4.1). Having more information in an ontology can obviously be useful for a very detailed model of a domain. However, its size can be kept smaller and the complexity lower by omitting information (e.g. certain relations or attributes) that can be inferred from linked ontologies when necessary. For example: FIBO and GIST offer attributes for a **Person**; e.g. in FIBO a **Person** `hasDateOfBirth` exactly 1 **Date**, in GIST a **Person** is `offspringOf` another **Person** and need to have a name `xsd:string`. These attributes can be extracted on demand, by following the `equivalentTo` relation.

As already shown by *Loebe*, the concepts and ideas about roles have been heavily discussed in the ontology community and literature. [14, p.130 1.2] The role concept is not trivial, very fundamental, and using it as part of an ontology allows for a flexible and powerful model. Since there is no clear agreement on a particular role concept, we are adopting the approach from *Loebe* (2007) and its basic role model. It is very general and can be applied in various ways: It can be used in the intuitive way regarding social roles, e.g. thinking about a human being playing the role of a patient and another the role of a doctor. But it is also possible to think about numbers and relationship between them in the form of abstract roles. [14, p. 131–133]

Since SCOs are a social construct and are defined by the people of the organization, the modeled roles are primarily from the type *social role*. Furthermore, the contexts described in section 3.1.1 also have implications for the roles that exists in the domain.

3.2.2. Roles in the Organizational Context

3.2.2.1. Membership

Within the Organizational Context (OC) (see section 3.1.1), the most basic property is membership: Either being part of the organization and thus being a **Member** or not participating and a **Non-Member**. **Members** of the SCO can play different roles in the OC. **Non-Members** don't play roles within the organization, but can play external roles. For example: The role of a **Customer**. The distinction is important for this ontology, since the status is typically used in the internal organizational procedures. For example: Someone might be required to be a proper member to be allowed to vote in the Member Assembly, to be part of a project, or to become part of the Executive Board.

It is important to note, that within SCOs the **Member** role can only be played by human beings. For this model this means a restriction to **Person**. The members are the defining group that fills all the organizational functions, works on the projects, and participates in the majority of processes. Even though membership does not have to be limited like this in general—there are many examples where organizations are members of other organizations—it is limited in this domain. Since **Member** are necessarily **always** human beings, we introduce the role as `subclassOf Person`.

The **Non-Member** role, however, is not limited to only **Person**. In fact, everything and everyone that is not a **Member** is by definition a **Non-Member**. Therefore it is sufficient to model the class **Member** and omit **Non-Member**.

3.2.2.2. Business Ranks

The second property a **Person** can have in the OC is the business rank. Examples from the business world are: Associate, Senior Associate, Consultant, Partner, etc. Similar to regular businesses, SCOs also organize these ranks around their career process: A person receives the lowest available rank at the begin of their career. During the time with the organization a person is awarded higher ranks based on some organizational system (e.g. a merit-based system), until the highest rank is reached or the person leaves the organization.

The exact terms for the ranks, their meaning, gradation, and organizational implications are highly specific to the SCO instance. We therefore introduce a rough and extensible skeleton representation:

1. **Trainee**: The entry level rank, without a formal membership.
2. **Junior Consultant**: The first rank after someone acquires an official, formal membership.
3. **Consultant**: The rank that implies someone has reached the required amount of knowledge and experience to fulfill all organizational functions.
4. **Senior Consultant**: The ultimate rank of the organization that can be reached after gathering a substantial amount of knowledge, experience, and organizational social status.

Ranks can only be attained by **Members** and therefore both are directly related. Since a **Member** can only hold exactly one **Rank** and the **Rank** further specifies the **Member** in the OC, we introduce the ranks as `subclassOf Member`. This is similar to the Schema.org approach that sub-classes **Organization** to be more specific about the type of organization: We are sub-classing **Member** to be more specific about it.

3.2.2.3. Corporate Officers

The complete set of tasks and responsibilities for the organizations day-to-day leadership is associated with the group of people referred to as Corporate Officers (COs). This set is typically divided into sub-sets and each sub-set is associated with a different role, to organize the work loads; and each role is played by a different person. For example: An organization may have the roles Chief Executive Officer (CEO), Chief Operating Officer (COO), and Chief Financial Officer (CFO) and each is played by a different individual.

The exact set of task differs from SCO to SCO and is also dependend on the organizational form, e.g. a SCO using the form of a registered association has different (legal) obligations than a SCO organized as university group. Completely specifying it is impossible on the level of abstraction this ontology operates on. In the same vein, it is impossible to define which tasks are attributed to which role, since every SCO organizes differently. Therefore we fall back on an extensible model and introduce the general class **Coporoate_Officer** as `subclassOf Organizational_Role`. To play a leadership role, it is required to be a formal **Member** of the SCO. Hence: **Coporoate_Officer** `isPlayedBy` only (**Junior_Consultant** or **Consultant** or **Senior_Consultant**).

There exist some common roles, that arise from the necessities of an SCO: There is typically a person that is formally responsible for the organization, a person that takes care of the finances, and a person that takes care of legal aspects of the project work. We differentiate between these three branches explicitly and introduce dedicated roles for each: **Chief_Executive_Officer**, **Chief_Financial_Officer**, and **Chief_Legal_Officer**.

It is important to note that these roles can all be played by the *same* **Person**. And: That we do not introduce any concrete tasks these roles have to fulfill.

3.2.2.4. Alumni, Advisor, and Patron

In the duality of membership—**Member** vs. **Non-Member**—exists some roles, where an assignment to either group is not clear cut when projecting on the real world: Alumni, advisors and patrons.

All of these roles can be played by **Members**, **Non-Members** or a **Group** of both. The role attribution depends on the internal organization of the particular SCO. Furthermore each of the roles have their own restrictions.

Alumni are a group of people that have been affiliated with an organization in the past, but are not members of that organization anymore. A good example are university alumni: The group of people that graduated from a specific university. Becoming an alumni typically is an informal and passive process that only requires previous SCO membership. However, it is also possible to interpret alumna as a more formal role and title, requiring being a **Member**. The common denominator in our model is the *previous* membership. Since this is a requirement and **Member** is restricted to be played by only **Person**, the same holds true for alumna. Furthermore, the previous membership also implies, that an alumna does not hold an internal rank anymore. We introduce **Alumna** as `subclassOf` **Organizational_Role**, restrict the player to **Person**, and specify a `disjointWith` (**Trainee** or **Junior_Consultant** or **Consultant** or **Senior_Consultant**).

Advisors are selected (e.g. appointed, chosen, elected) to assist the SCO leadership with a neutral perspective in their decisions. Becoming an advisor is an active, conscious process. Both parties, advisees and advisors, are necessarily restricted to **Person**: The leadership of the organization is recruited from the pool of members; and the advisory concept models a direct and personal exchange of assistance. We introduce **Advisor** as `subclassOf` **Organizational_Role** that can only be played by **Person**.

Patrons are financial and/or ideological supporters of the SCO: A financial patron directly contributes to the monetary funds of the organization; an ideological patron primarily supports the idea of SCO and contributes through non-financial means. Both roles can be played by one player simultaneously. Often times ideological patronage also involves a form of financial support and vice versa. For example: The associated university of the SCO may provide patronage (e.g. allowing promotion on the university website and campus) and infrastructure (e.g. offices, meeting rooms, etc.). We introduce **Patron** as `subclassOf` **Organizational_Role** and further specify **Financial_Patron** and **Ideological_Patron** as `subclassOf` **Patron**. Since patronage, especially financial support, require contracts, the role can only be played by a formal entity: It is restricted to **Person** and **Organization**.

Note: The model says nothing about social status and political power that typically come with ranks and roles, such as being a CO or advisor, within an organization (e.g. a person that holds a rank or role for a long time may still have organizational power after stepping down: \hookrightarrow Éminence grise).

3.2.3. Roles in the Project Context

3.2.3.1. Project Team: Member, Leader, Controller

A project team in its most basic form consists of team members that are lead by a team leader. Additionally a project controller can be employed to observe and measure the progress of the project, giving feedback on the project work, and helping the team in various capacity where necessary. The controller role is usually played by a person that has gathered experience with projects and sharing them further supports the idea of SCO by furthering the learning process of the project team members.

We model the **Project_Team** as subclassOf **Group**, since it is a collection of people that play

We model **Project_Team_Member**

We model **Project_Team_Leader**

We model **Project_Controller**

3.2.3.2. Customer

The second party of a project is the customer.

The **Customer** role can be played by an **Agent**

3.2.4. Conclusion: Human Beings in this Work

The strength of the role concept is its flexibility. A player can play multiple roles at the same time and each role can be associated with a different context. An example for this is the CEO role. It has defined responsibilities and playing the role means a requirement to fulfill certain tasks. With SCOs typically any formal **Member**—in our ontology this means any **Member** with rank **Junior Consultant** or above—can become CEO by being elected. When elected, the **Member** plays two roles in the OC. Additionally, the same **Member** could work on a project as **Project Leader**, a role from the Project Context (PC).

3.3. Processes

Processes are a helpful concept when describing organizations: They are created to achieve a goal and its processes are the steps needed to reach that goal. [30, p. 5, Definition 1.1] In theory, every organization can be decomposed to a sequence of single activities, which, when executed correctly and in the correct order, terminate in reaching the goal of the organization.

Since processes are a commonly used concept in the business world, it is not surprising, that many different methods and frameworks for modeling them have been developed. Their output often are visual representations of all workflows that make up an organization. Combining process models with goals and measurements makes them a powerful tool for optimization and quality control. For example, ISO 9001 is an industry standard that uses a process approach as the foundation of measuring quality. [12] Because process documentation contains a lot of data about organizations, it is a valuable source for ontology development.

Widely known representations and methods include: Flowcharts, Business Process Modeling and Notation (BPMN), Event-Driven Process Chain (EPC), Unified Modeling Language (UML) Activity Diagrams, and Object Process Methodology (OPM)²³. There are also contributions rooted in ontology research, such as the BPMN ontology (an OWL ontology for the BPMN notation) [22], the Process Specification Language (PSL)²⁴, and processes concepts as part of GFO or BFO.

²³Standardized as ISO 19450.

²⁴Developed by the National Institute of Standards and Technology (NIST) and standardized as ISO 18629.

3.3.1. Implementation in Related Ontologies

When compared to the rather practical and direct implementation of social structures discussed in section 3.2.1, processes are a more abstract concept. The impact of abstraction levels clearly shows when analyzing related ontologies. For example: While FOAF is a good source when discussing its niche—the modeling of connection between human beings—it does not require an implementation of a process concept. The closest possible link between these two knowledge domains is the class **foaf:Project**²⁵, which can be viewed as a procedural concept. However, it doesn't offer any additional reusable detail.

A similar observation can be made for Schema.org. Its primary purpose is adding semantic meaning to the internet: "Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond." [24] Hence, it is not surprising, that it doesn't implement a detailed process representation.

On the other hand, the two related top-level ontologies, BFO and GFO, deal with time (see section 2.4.5) on a very high level and also implement process concepts: BFO uses the class **bfo:Occurent**²⁶ as entry point for its process concepts. It is sub-classed by **bfo:Process**²⁷, **bfo:Process_Boundary**²⁸, **bfo:Spatiotemporal_Region**²⁹, and **bfo:Temporal_Region**³⁰. Looking closer at the given examples³¹ for **bfo:Process** emphasizes the classes high-level nature.

GFO uses the class **gfo:Processual_Structure**³² as its entry point. It is sub-classed by **gfo:Occurent**³³ and **gfo:Processes**³⁴.

DOAP

FIBO

²⁵foaf:Project rdfs:comment: "A project (a collective endeavour of some kind)."

²⁶bfo:Occurent elucidation: "An occurrent is an entity that unfolds itself in time or it is the instantaneous boundary of such an entity (for example a beginning or an ending) or it is a temporal or spatiotemporal region which such an entity occupies_temporal_region or occupies_spatiotemporal_region. (axiom label in BFO2 Reference: [077-002])"

²⁷bfo:Process definition: "p is a process = Def. p is an occurrent that has temporal proper parts and for some time t, p s-depends_on some material entity at t. (axiom label in BFO2 Reference: [083-003])"

²⁸bfo:Process_Boundary definition: "p is a process boundary =Def. p is a temporal part of a process and p has no proper temporal parts. (axiom label in BFO2 Reference: [084-001])"

²⁹bfo:Spatiotemporal_Region elucidation: "A spatiotemporal region is an occurrent entity that is part of spacetime. (axiom label in BFO2 Reference: [095-001])"

³⁰bfo:Temporal_Region elucidation: "A temporal region is an occurrent entity that is part of time as defined relative to some reference frame. (axiom label in BFO2 Reference: [100-001])"

³¹bfo:Process example of usage: 1. a process of cell-division, 2. a beating of the heart, 3. a process of meiosis, 4. a process of sleeping, 5. the course of a disease, 6. the flight of a bird, 7. the life of an organism, 8. your process of aging.

³²gfo:Processual_Structure dc:description: "The category of processual structures centers around the more intuitive notion of processes. It captures processes themselves and occurrents, i.e., primarily structures of several other kinds that can be derived from processes."

³³gfo:Occurent dc:description: "The category of occurrents comprises several categories that can be derived from processes."

³⁴gfo:Processes dc:description: "Processes are directly in time, they develop over and unfold in time. Processes have characteristics which cannot be captured by a collection of time boundaries. In particular, processes exhibit internal coherence."

GIST uses the root class **Event**³⁵ to deal with time-related constructs such as processes.

bfp gfo contain valuable information, that can be modified and adapted for the process implementation of the SCO domain.

3.3.2. Structure of the Class Hierarchy

Processes need special attention when implementing them in a domain ontology, since their nature is quite different from other classes that represent physical, e. g. **Document**, or intuitive concepts, e. g. **Person**. As mentioned in section 2.2.2.1, the built in `subclassOf` relation of the class hierarchy already carries semantic meaning, that is generally not applicable to processes. For example: a **Delivery_Process** may involve a **Food_Preperation_Process** as a procedural step. However, it is easy to see and understand that a **Food_Preperation_Process** is *not* `subclassOf` a **Delivery_Process** and therefore should not inherit its properties.

To model processes correctly, one could consider introducing a class like ***_Process_Part** (in the given example: **Delivery_Process_Part**) and use it to collect and connect sub-processes to their parent process. However, this results in many additional *helper* classes in the class hierarchy, since every level of sub-processes requires another ***_Process_Part** class. This makes the class hierarchy harder to read and understand, since the process structure is encoded in these helper classes.

Another solution is the use of a root **Process** class to collect all processes and the relation `isProcessPartOf`³⁶ to connect a sub-process to its parent process. This results in a completely flat structure of the class hierarchy: every process is directly `subclassOf` **Process**, independent from the level of abstraction.

3.3.3. Ordering of Processes

Another aspect that has to be discussed is the ordering of processes.

Processes are a concept that heavily relies on abstraction. The right level of abstraction depends on the use case.

1. It is hard to create a complete process diagram/describe a complete process
2. A successful process model relies on the correct
3. Besides from the problem of completeness, if you go on the lowest level of abstraction ordering the steps become easier (true?!)
4. Trying to create a generally applicable domain ontology brings up an interesting question in regards to process concepts: what is the correct level of abstraction and is it possible to bring certain processes in the correct order
5. It is therefore necessary to look at every process and its parts – and discuss if 1) correct level of abstraction 2) can it be ordered
6. A parent process aggregates child process
7. When discussing the lowest level of a process Depending on the level of abstraction,

³⁵`gist:Event rdfs:comment: "Something happening over some period of time, often characterized as some kind of activity being carried out by some person, organization, or software application."`

³⁶`Inverse: hasProcessPart.`

8. Independent from the class hierarchy problem, another topic: The procedural information is encoded in the relation.
9. Processes are a sequence of action -> but not always
10. there should be a `next_process` relation, but it's not straight forward, since there is not necessarily a strict ordering.
11. introduce a non strict ordering?

3.3.4. Discreet Events and Liquid Processes

1. In addition to that a distinction has to be made between *discreet* events and *liquid* processes. [23, p. 447]
2. Processes are an immaterial concept that is strongly connected to relations of relative time, such as *before* and *after*.

3.3.5. Legal Requirements in the Processes

SCOs are organizations in the social context; German law applies to them, like it applies to every other German organization. It influences their structure and processes. However, discussing the impact of the law onto internal workings of an organizations would go far out of scope of this work. Therefore the ontology omits a detailed description of the legal obligations, but references them abstractly where it is necessary.

For example: German law requires every company to pay taxes on their earnings. Depending on the SCO and the way projects are handled, this influences the process that is concerned with taxation. To develop a perfectly correct model, a very detailed discussion of specific processes would be required; this is out of scope. However, interacting with the tax authorities and learning about and filing the correct paper work is an important part of the learning experience for student consultants. It is therefore important for the ontology. To address this fact, but keep the ontology focused, it is condensed into the class *Project Taxation Process* as part of the *Project Process*.

3.3.6. Conclusion: Processes in this Work

They intuitively have a start, duration, and end.

Since the focus of the ontology is on simplicity, we decide to use a single class **Process** as root for all processes in conjunction with the `isProcessPartOf` relation. This method utilizes the core concepts of ontologies, classes and relations, and avoids encoding extra information in unconventional ways. To compensate for the resulting un-intuitive flat class hierarchy, we add diagrams to describe the processes and their relationships graphically (see appendix A.2.2).

The primary goal of an SCO is teaching students project work. They reach this goal by training their members and offering them opportunities to work on real-world projects. Looking at this from a high-level process perspective, this can be boiled down to distinct steps that have to be performed by the organization:

- Members have to be recruited.
- Members have to be taught the necessary skills, to be able to work on projects.
- Projects have to be acquired.

- Projects have to be worked on by members.

Additionally these steps have various amount of support processes that help facilitate them, e. g. technical and legal support.

Conflating this intuitive view results in two complex processes that are commonly known in the business world [CN] and are also present in the available process documentation [8]:

1. A Human Resource Process (HRP), that focuses on the recruitment, training, and generally enabling of human beings (or in the case of SCOs: Members).
2. A Project Process (PP), that documents the way an organization handles projects from start to finish.

Again, both processes are influenced by the context switches discussed in section 3.1.1. On the one hand, they can be viewed as an individual instance for one of the main protagonists of the process. On the other ^{hand} they can ^{be} viewed as the process of the organization. Example:

- The main protagonist of the HRP is one individual student. This individual student is following one instance of the HRP; this instance does not have to be identical with the instance of a second individual student, nor with the planned process of the organization. Both individuals might do different educational courses, hold different business ranks within the organization, or might be at different points in time of their career.
- The SCO itself has a HRP. It structures important aspects of the organization such as the career path. It describes the complete path from recruitment of a new member to offboarding at the end of the membership. Most importantly this process describes the plan on an abstract level and knowingly omits parts of the real world process that are not important to the organization.

4. Student Consulting Organizations: The Ontology

4.1. Scope of the Domain

4.1.1. What is the domain that the ontology will cover?

SCOs are a form of consulting firms. They can be compared to small consulting businesses, but are staffed – most of the time exclusively – by students. In other countries, e.g. in France and Brazil, they are also referred to as *Junior Enterprises* (JE). In Germany they are usually a registered association (German: *Verein*) and/or a group associated with a specific university (German: *Hochschulgruppe*). They aim to teach students about consulting as a profession by providing a platform that educates and trains students in the craft and provides them with the organizational means to work on consulting projects.

The domain is a specialization of the a classical consulting firm. It differs especially in terms of professionalization, since companies are focused on profit using education as the means, whereas SCOs focus on the educational aspect and on providing experience, while having profit as secondary goal.

4.1.2. For what we are going to use the ontology?

This ontology is a contribution to the knowledge management of SCOs. It can be used to learn or teach about the domain. It can also be used as a starting point for projects that require a model of the domain.

4.1.3. For what types of questions the information in the ontology should provide answers?

The ontology serves as an abstract description of the SCO domain. It defines all classes and relations that are typically present in this type of organization. Therefore it can answer questions like:

- What processes exist and are required in an SCO?
- What roles exist and have to be filled in an SCO?

4.1.4. Who will use and maintain the ontology?

The users of this ontology are the leadership of SCOs in Germany as well as the leadership of the SCO umbrella organizations. The release version coincides with the finalization and grading of this work. If the ontology sees use by the target group, it will be maintained by the author. Access will be publicly provided on a GitHub repository. It is considered a living document, hence not necessarily complete until otherwise stated. Contributions and forks will be possible via the GitHub interface.

4.2. Classes

4.2.1. Agent

4.2.1.1. Group

4.2.1.2. Organization

4.2.1.3. Person

4.2.1.3.1. Trainee

4.2.1.3.2. Junior Consultant

4.2.1.3.3. Consultant

4.2.1.3.4. Senior Consultant

4.2.2. Document

4.2.3. Processes

4.2.3.1. Human Resource Process

4.2.3.2. Project Process

4.2.3.3. Support Processes

4.2.4. Projects

4.3. Relations

Syntactic decision: is/has relations

Agent

- All members except trainees and almunus can be corporate officers ◇
- Non-members can't become corporate officers ◇
- Members can play project team roles ◇
- Every agent can play the customer role in a project ◇
- Every member goes through his individual HRP
- An organizational rank has tbd requirements
- Customer, Team, Contract, etc are **part of** a project
- Organizations can only **play** the customer **role** in a project ◇
- Organization can only play external roles ◇

Processes

- All processes have a **next_process** ◇
- **previous_process** should be inferred?!
-

before/after:

- FIBO: relates to -> precedes/succeeds?
- plays role

5. Conclusion

6. Further Research

A. Appendix

A.1. Term Enumeration

A.2. Diagrams

A.2.1. General Diagrams

Figure A.1.: Ranks

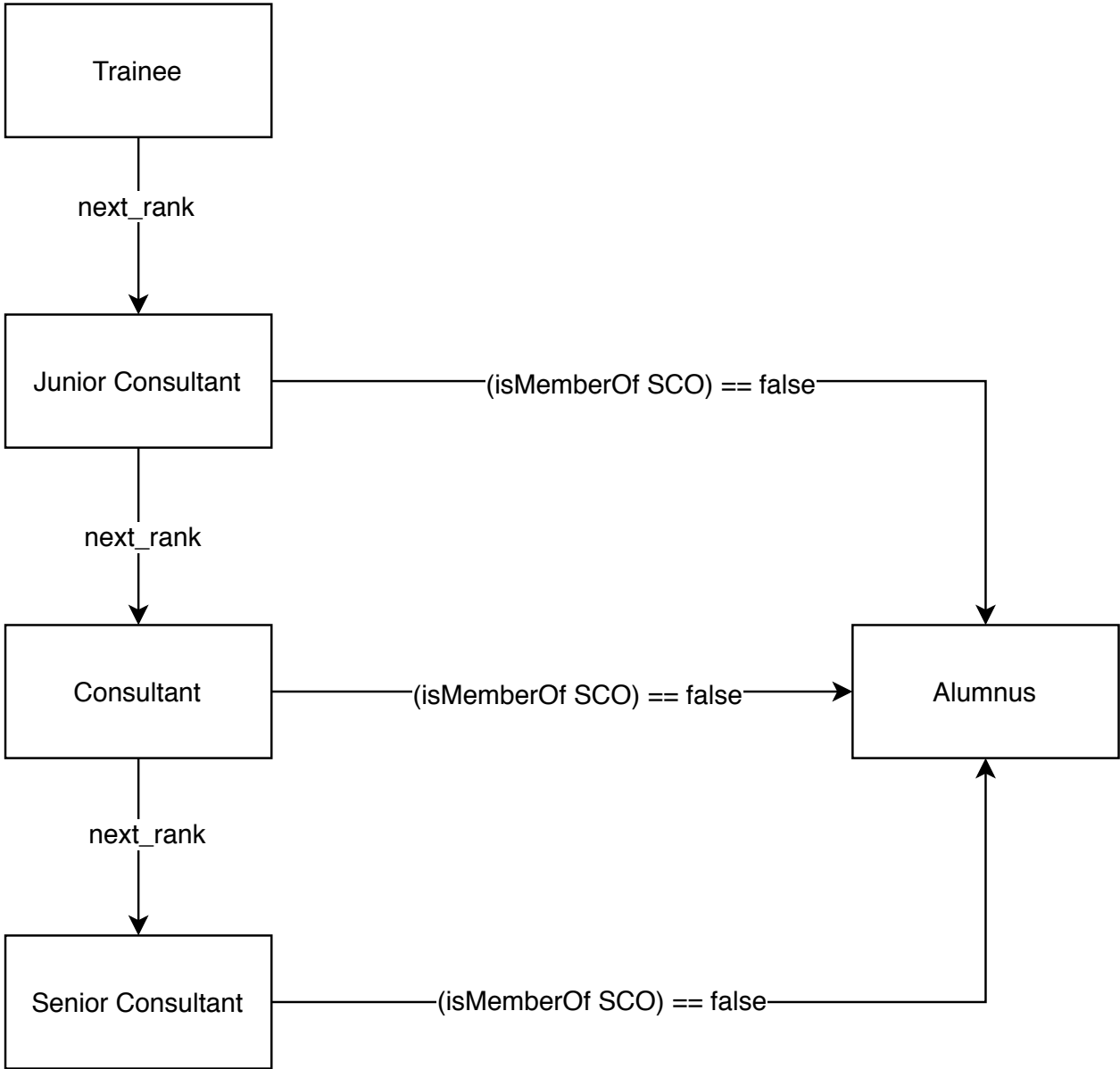
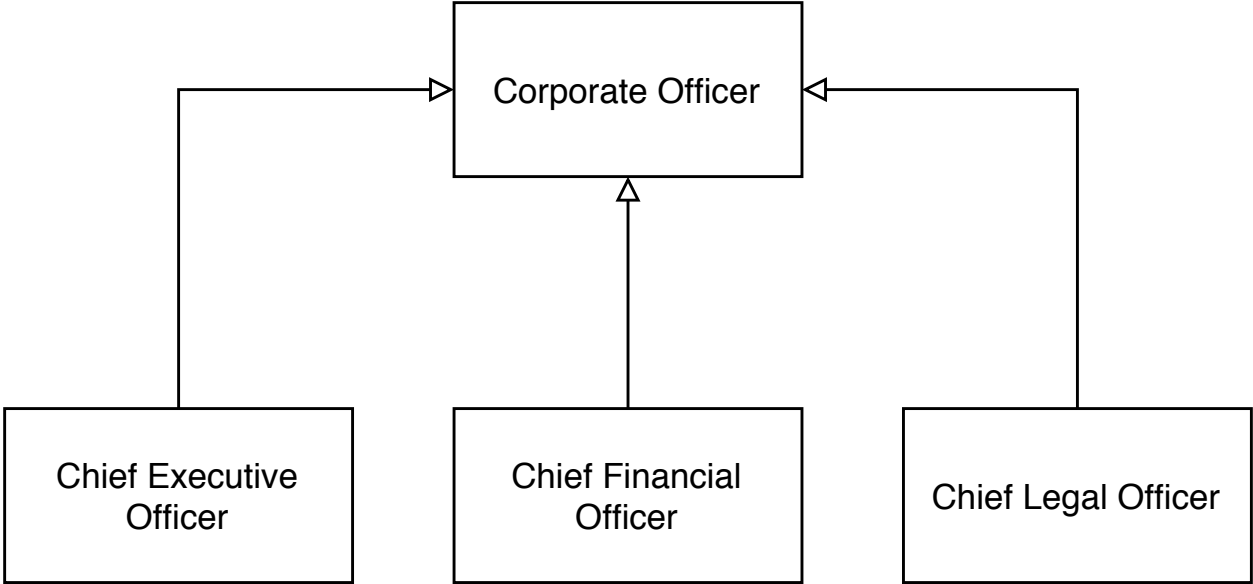


Figure A.2.: Corporate Officers



A.2.2. Process Diagrams

Figure A.3.: Project Process

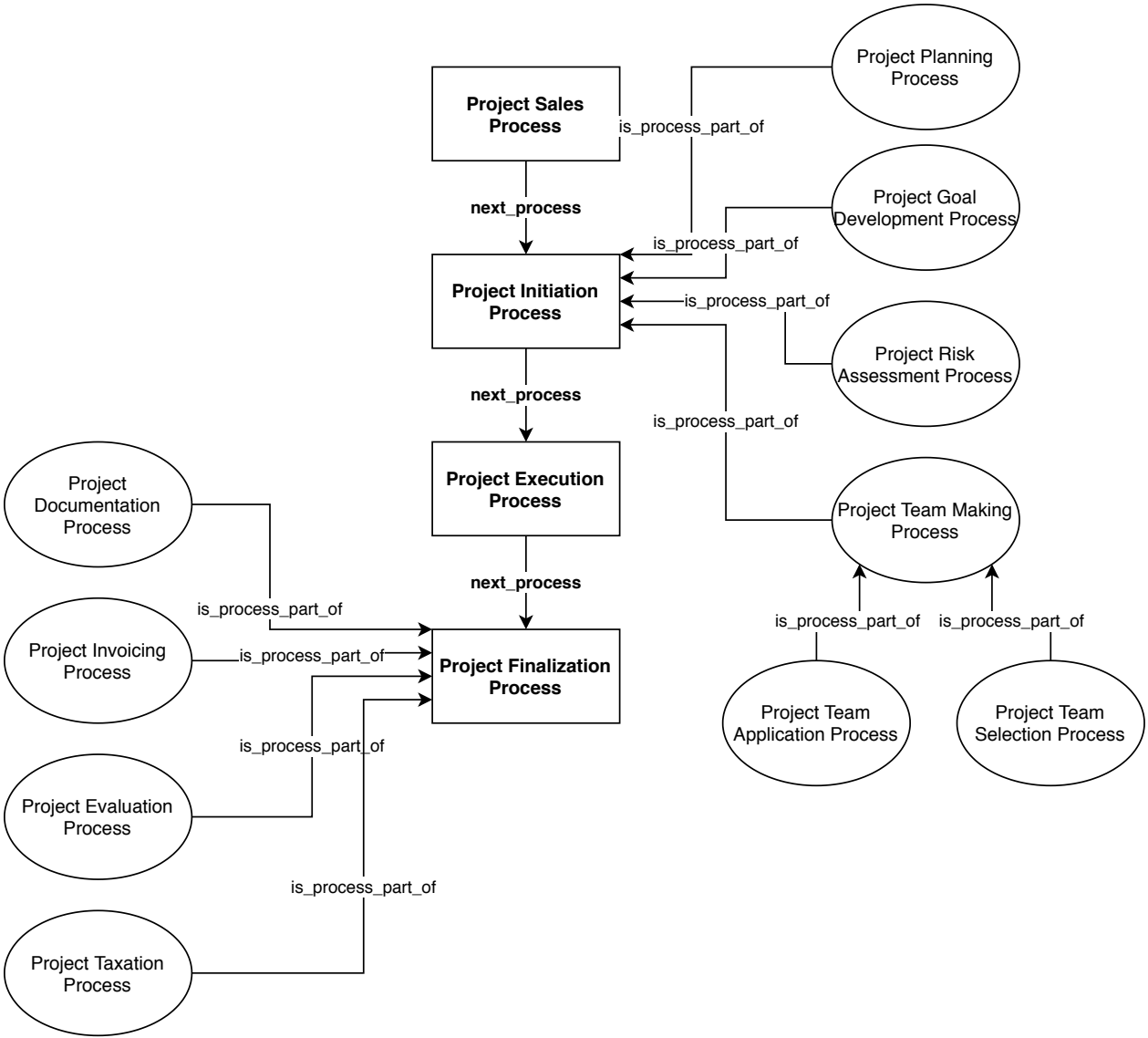
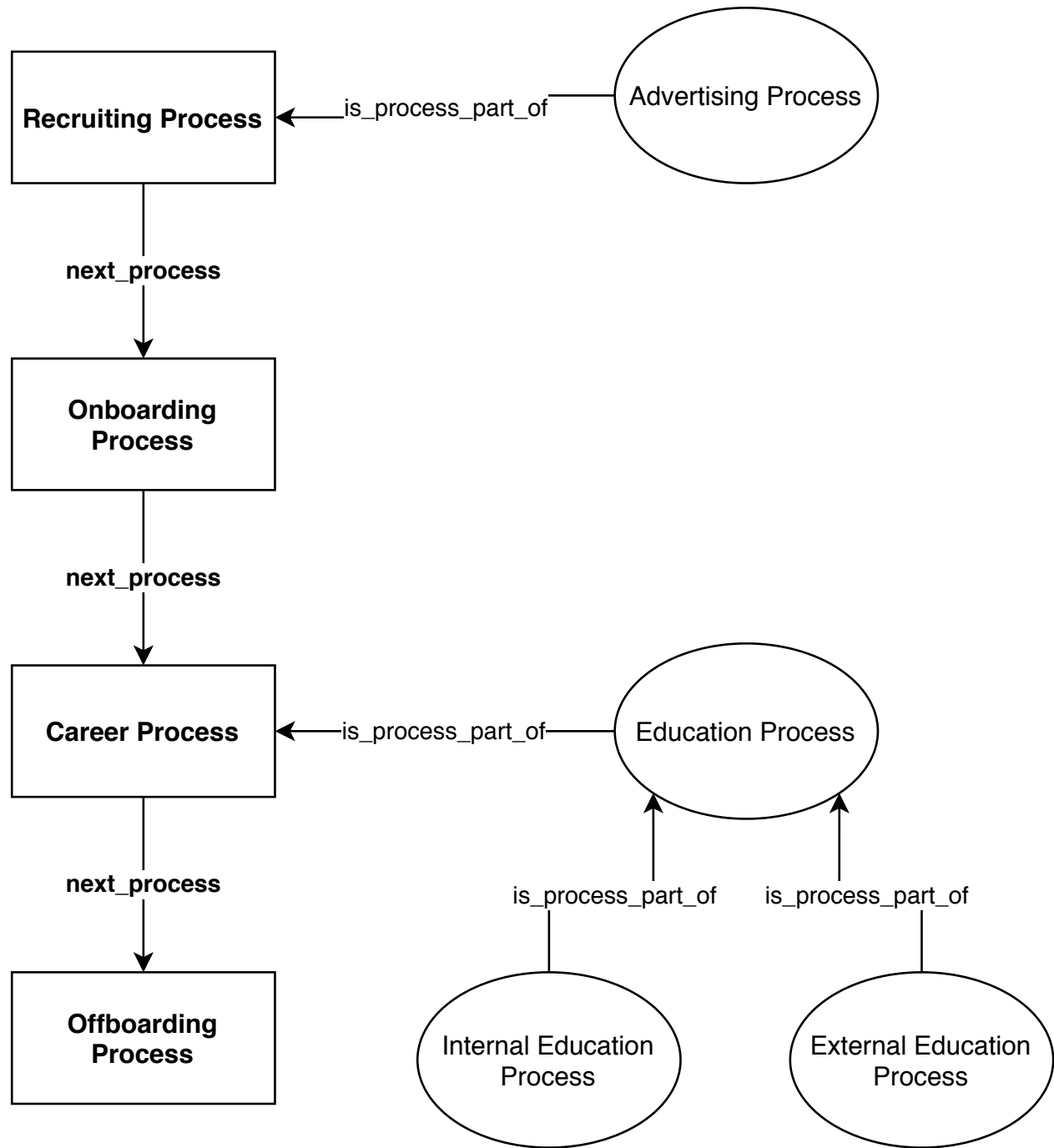


Figure A.4.: Human Resources Process



A.3. Glossary

BDSU \leftrightarrow Bundesverband Deutscher Studentischer Unternehmensberatungen. 1

BFO Basic Formal Ontology. 6, 12, 17, 21, 22, 42

BPMN Business Process Modeling and Notation. 21, 42

CEO Chief Executive Officer. 19

CFO Chief Financial Officer. 19

CI \leftrightarrow Campus Inform. 1, 5

CO Coporate Officer. 19, 20

COO Chief Operating Officer. 19

DCMT Dublin Core Metadata Terms. 16

DOAP Description of a Project. 10, 16, 22, 42

EPC Event-Driven Process Chain. 21

FIBO Financial Industry Business Ontology. i, 10, 16, 17, 22, 42

FOAF Friend of a Friend. i, 10, 15, 16, 17, 22, 42

GFO General Formal Ontology. i, 6, 12, 17, 21, 22, 42

GIST GIST. i, 6, 16, 17, 23, 42

HC \leftrightarrow Hanseatic Consulting. 1, 5, 7

HRP Human Resource Process. 25

ISO International Organization for Standardization. 14

JCNetwork \leftrightarrow Junior Consultant Network. 1

KISS Keep It Stupid Simple. 11, 17

NIST National Institute of Standards and Technology. 21

OC Organizational Context. 18, 19, 21

OPM Object Process Methodology. 21

OWL Web Ontology Language. 4, 6, 7, 8, 21

PC Project Context. 21

PM Project Management. 10

PP Project Process. 25

PSL Process Specification Language. 21

RDFS Resource Description Framework Schema. 9

Schema.org Schema.org Ontology. 10, 16, 19, 22, 42

SCO Student Consulting Organization. i, 1, 2, 3, 4, 5, 6, 7, 8, 10, 13, 14, 15, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27

SKOS Simple Knowledge Organization System. 9

UML Unified Modeling Language. 21

W3C World Wide Web Consortium. 4

A.4. Bibliography

- [1] Alexandra Arapinis and Laure Vieu. “A plea for complex categories in ontologies”. In: *Applied Ontology* 10.3-4 (2015), pp. 285–296.
- [2] Franz Baader et al. *Introduction to Description Logic*. Cambridge University Press, 2017. ISBN: 978-0-521-87625-4.
- [3] Statistisches Bundesamt. *Zusammengefasste Abschlussprüfungen mit erstem und weiteren Abschluss sowie Gesamtstudienzeit (2016-2018)*. Oct. 2019. URL: <https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Bildung-Forschung-Kultur/Hochschulen/Tabellen/bestandeneproofungen-studiendauer.html> (visited on 2020-03-01).
- [4] Libby Miller Dan Brickley. *FOAF Vocabulary Specification 0.99*. English. Jan. 2014. URL: <http://xmlns.com/foaf/spec/> (visited on 2020-02-27).
- [5] Anind K Dey, Gregory D Abowd, and Daniel Salber. “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications”. In: *Human-Computer Interaction* 16.2-4 (2001), pp. 97–166.
- [6] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering - with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004. ISBN: 1-85233-551-3.
- [7] Thomas R Gruber et al. “A translation approach to portable ontology specifications”. In: *Knowledge acquisition* 5.2 (1993), pp. 199–221.
- [8] Hanseatic Consulting Hamburg. *Prozesshandbuch*. Internes Dokument. May 2014.
- [9] Heinrich Herre. “General Formal Ontology (GFO): A foundational ontology for conceptual modelling”. In: *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 297–345.

- [10] Wolfgang Hesse and Hermann Engesser. “Ontologie”. In: *Informatik-Spektrum* 37.4 (May 2014), pp. 281–282. ISSN: 1432-122X. DOI: 10.1007/s00287-014-0808-2. URL: <http://dx.doi.org/10.1007/s00287-014-0808-2>.
- [11] ISO. *Quality management systems - fundamentals and vocabulary (ISO 9000:2015)*. 4th edition. [Geneva]: ISO Copyright office, 2015.
- [12] ISO. *The Process Approach in ISO 9001:2015 (ISO/TC 176/SC 2/N1289)*. online. 2015. (Visited on 2020-03-10).
- [13] Stéphane Jean, Guy Pierra, and Yamine Ait-Ameur. “Domain Ontologies: A Database-Oriented Analysis”. In: *Web Information Systems and Technologies* (2007), pp. 238–254. ISSN: 1865-1348. DOI: 10.1007/978-3-540-74063-6_19. URL: http://dx.doi.org/10.1007/978-3-540-74063-6_19.
- [14] Frank Loebe. “Abstract vs. social roles—Towards a general theoretical account of roles”. In: *Applied Ontology* 2.2 (2007), pp. 127–158.
- [15] Frank Loebe. “Ontological Semantics: An Attempt at Foundations of Ontology Representation”. PhD thesis. Universität Leipzig, Mar. 2015.
- [16] Merriam-Webster.com. *Dictionary*. URL: <https://www.merriam-webster.com/dictionary/>.
- [17] Philip Moore, Cain Evans, and Hai V Pham. “Towards integrating emotion into intelligent context”. In: *Web Information Systems Engineering—WISE 2011 and 2012 Workshops*. Springer. 2011, pp. 27–40.
- [18] Philip Moore and Hai V Pham. “Intelligent context with decision support under uncertainty”. In: *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*. IEEE. 2012, pp. 977–982.
- [19] Philip Moore and Hai Van Pham. “On context and the open world assumption”. In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*. IEEE. 2015, pp. 387–392.
- [20] Mark A Musen. “The protégé project: a look back and a look forward”. In: *AI matters* 1.4 (2015), pp. 4–12.
- [21] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. URL: https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html (visited on 2020-02-20).
- [22] Marco Rospocher, Chiara Ghidini, and Luciano Serafini. “An ontology for the Business Process Modelling Notation”. In: *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS2014, September, 22-25, 2014, Rio de Janeiro, Brazil*. Ed. by Pawel Garbacz and Oliver Kutz. Vol. 267. IOS Press, 2014, pp. 133–146. DOI: 10.3233/978-1-61499-438-1-133. URL: <http://dx.doi.org/10.3233/978-1-61499-438-1-133>.
- [23] Stuart Russell. *Artificial intelligence : a modern approach*. 3th ed. New Jersey: Pearson, 2010. ISBN: 978-0-136-04259-4.
- [24] Schema.org. *About Schema.org*. URL: <https://schema.org/docs/about.html> (visited on 2020-03-16).
- [25] S Schulz et al. *Guideline on developing good ontologies in the biomedical domain with description logics*. Tech. rep. Technical Report December, Universität Rostock. 2012., 2012.
- [26] Barry Smith et al. “Basic formal ontology 2.0”. In: *Specification and User’s Guide* (2015).

- [27] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. *OWL Web Ontology Language Guide*. Feb. 2004. URL: <https://www.w3.org/TR/owl-guide/>.
- [28] Heiner Stuckenschmidt. *Ontologien: Konzepte, Technologien und Anwendungen*. Springer, 2010. ISBN: 978-3-642-05403-7.
- [29] Mike Uschold et al. “The enterprise ontology”. In: *The knowledge engineering review* 13.1 (1998), pp. 31–89.
- [30] Mathias Weske. *Business Process Management : Concepts, Languages, Architectures*. Springer, 2019. ISBN: 978-3-662-59432-2.

A.5. Dictionary Definitions

context NOUN	MERRIAM-WEBSTER
con · text	
1. the parts of a discourse that surround a word or passage and can throw light on its meaning	
2. the interrelated conditions in which something exists or occurs : ENVIRONMENT, SETTING	
<i>“the historical context of the war”</i>	

domain NOUN	MERRIAM-WEBSTER
do · main	
1. law	
a) complete and absolute (see absolute sense 3) ownership of land	
<i>“our highways and roads have been in the domain of state and local governments— T. H. White b. 1915”</i>	
— compare EMINENT DOMAIN	
b) land so owned	
2. a territory over which dominion (see DOMINION sense 2) is exercised	
<i>“The forest is part of the king’s domain.”</i>	
3. a region distinctively marked by some physical feature	
<i>“a domain of rushing streams, tall trees, and lakes”</i>	
4. a sphere (see SPHERE sense 4b) of knowledge, influence, or activity	
<i>“the domain of biblical scholarship”, “outside the domain of city police”</i>	
5. mathematics : the set of elements (see ELEMENT sense 2b(3)) to which a mathematical or logical variable is limited	
specifically : the set on which a function (see FUNCTION entry 1 sense 5a) is defined	
6. physics : any of the small randomly oriented regions of uniform magnetization in a ferromagnetic substance	
7. mathematics : INTEGRAL DOMAIN	
8. biology : the highest taxonomic category in biological classification ranking above the kingdom (see KINGDOM sense 4b)	
9. biochemistry : any of the three-dimensional subunits of a protein that are formed by the folding of its linear peptide chain and that together make up its tertiary (see TERTIARY entry 1 sense 3c) structure	
10. computers : a subdivision of the Internet consisting of computers or sites usually with a common purpose (such as providing commercial information) and denoted in Internet addresses by a unique abbreviation (such as com for commercial sites or gov for government sites)	
<i>“The domain ca is used for sites located in Canada.”</i>	
also : DOMAIN NAME	
<i>“Our domain is Merriam-Webster.com.”</i>	

vocabulary NOUN	MERRIAM-WEBSTER
vo · cab · u · lary <i>plural</i> vocabularies	
1. a list or collection of words or of words and phrases usually alphabetically arranged and explained or defined : LEXICON	
<i>“The vocabulary for the week is posted online every Monday.”</i>	
2. a) a sum or stock of words employed by a language, group, individual, or work or in a field of knowledge	

“a child with a large vocabulary”, “the vocabulary of physicians”, “a writer known for employing a rich vocabulary”

b) a list or collection of terms or codes available for use (as in an indexing system)

“... the oldest Sumerian cuneiform writing could not render normal prose but was a mere telegraphic shorthand, whose vocabulary was restricted to names, numerals, units of measure, words for objects counted, and a few adjectives.” — JARED DIAMON

3. a supply of expressive techniques or devices (as of an art form)

“an impressive musical vocabulary”

A.6. Ontology Import Links

This work lists different ontologies in the related work section. To import them into the Protégé editor, the following links can be used:

BFO: <http://purl.obolibrary.org/obo/bfo/2.0/bfo.owl>

BPMN: https://dkm-static.fbk.eu/resources/ontologies/BPMN/BPMN_2.0_ontology.owl

DOAP: <http://usefulinc.com/ns/doap>

FIBO: <https://spec.edmcouncil.org/fibo/ontology/master/2019Q4.1/LoadFIBOProd.rdf>

FOAF: <http://xmlns.com/foaf/spec/index.rdf>

GFO: <http://www.onto-med.de/ontologies/gfo-basic.owl>

GIST: <https://ontologies.semanticarts.com/o/gistCore9.0.0.owl>

Schema.org: <http://schema.org/version/latest/schema.rdf>

A.7. Acknowledgments

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.