Document Version: v16
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 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.

PRODUKTWERBUNG

Formatting

- Hyperlinks are embedded and clickable in the PDF. They are marked with an arrow and a light blue border: ↪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`**
- The bibliography is sorted by order of appearance.

Leseanleitung

Contents

1. Introduction

1.1. Motivation

SCO[1] are student-run consulting businesses that focus on teaching their members essentials 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, this kind of organization exists worldwide and has a history dating back to at least 1967[2]. Germany has two different umbrella organizations for SCOs with more than 60 member organizations.[3]

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. [1] This frames the available time for the transfer of the domain knowledge.[4]
2. The career is 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 one we worked with, ↪Campus Inform (CI), was focusing on lectures and training sessions.

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

[1] Also known as ↪Junior Enterprises in some parts of the world.
[2] The founding of ↪Junior ESSEC in France.
[3] ↪Bundesverband Deutscher Studentischer Unternehmensberatungen (BDSU), ↪Junior Consultant Network (JCNetwork)
[4] There sometimes are also PhD students, but they can be considered outliers and are atypical.

every SCO is organized slightly differently from 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

Goal 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.

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.

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

1.3. 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. [2, 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." [3, 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 the Web Ontology Language (OWL). It is an ontology language developed by the consensus-driven and well respected World Wide Web Consortium (W3C). [3, p. 206] It is widely used and offers very good tooling in the ontology editor ↪***Protégé***. It is built and maintained by ontology researchers of ***Stanford University***. [4] Furthermore the research group provides an ontology development methodology [5], 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[1],
(6) define the facets of the slots[2], 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 [5, Section 3, Introduction], following the iterative approach, that is common for ontology development. [6, 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 [5, Section 3, Step 1] and taking into account (2) existing ontologies. And this also

[1]The guide is written for an earlier version of Protégé and the terms have since been updated. Slots are now referred to as annotations and facets are the data types of annotations.
[2]See above.

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. 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 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 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 such as the implications of other general, upper-domain, and top-level ontologies as well as theory of description logic and theory of 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. [7, 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 concepts with the highest 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 chapter 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. [8]

In the context of ontologies it is used interchangeably with *alphabet* and *signature*. On an abstract level it simply describes a set of symbols. [9, p. 46] Transferred to this work, it is the collection term for all classes (see 2.2.3) and object property names (see 2.2.4), 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 2.2.4).

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. [9, p. 7] This is close to the dictionary definition (see appendix A.5), where every listed sense has a delimiting meaning to it. [8]

However, at the time of writing, an exact definition of the term *ontology* is difficult to find. There is no unified understanding across the sciences. [10] Numerous authors tried defining the term as shown by Loebe [9, p. 4-6] and Gomez et al. [2, 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.

?

aim?

target?

goal?

The influential paper [11, p. 9] [9, 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." [12, 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." [3, 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. [11, p. 10] 2. Our ontology language is OWL.

To work 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 [14] and using the *HermiT* reasoner during development. To aim for consensus, we prepare the ontology in a 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.

2.2.3. 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 2.2.4). 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 2.2.4): `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. 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. Since *everything* in our ontology inherits from `owl:Thing`, it would pass on its properties as well. Since there are no properties that apply to all classes of our ontology, our **owl:Thing** does not have properties.

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 subsection 3.3.2).

2.2.4. 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. [14] 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*.

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** `isMemberOfTheMajorityCounil` **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 links 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 **Agent** can do, a **Person** or an **Organization** can also do.

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 or metadata schemas (see section 2.3). 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

To our knowledge, there has not yet been any effort to model the SCO domain explicitly as an ontology. However, it intersects with various other knowledge areas and related source material can be found in other ontologies, Project Management (PM) models and concepts, established ontology schemas and vocabulary, and so forth. The challenge lies in identifying the most useful information and merge the input from many different sources to construct our ontology.

Ontologies We already touched the complicated nature of ontologies in section 2.2. There are so many different ideas, concepts, and proposals available in this space, that it is impossible to completely work through it in the limited scope of this work. We briefly discuss three categories of ontologies and how we are using them in this work. We group them by their level of abstraction, going from the highest to the lowest: Top-Level Ontologies (TLOs, also sometimes called *Upper Ontologies*), Upper-Domain Ontologies (UDOs), and Domain Ontologies (DOs).

Note: These categories are a tool to organize the related ontologies. They are not absolute, but instead can be thought of as labels for areas on a linear scale that is based on the level of abstraction. The borders of the areas are not fixed and other researchers might place them differently.

TLOs deal with the highest level concepts that relate other ontologies. [15, p. 3] These may include: Entities, Relators, Time, Space, etc. Examples are: The General Formal Ontology (GFO) [16], Basic Formal Ontology (BFO) [17], and GIST (GIST) [18]. More examples are available on ↪Wikipedia: Upper Ontology. Since TLOs operate on such a high level, we do not expect to be able to apply them directly. We will, however, try to find links to this class of ontology for the domain topics, that are more abstract in nature (e. g. processes). For this
? process, we focus on the three aforementioned TLOs examples.

UDOs occupy the space between the high-level TLOs and the lower-level DOs. They describe a wide and general area, but start going into specific descriptions. Examples are: The Financial Industry Business Ontology (FIBO)[3] [19], Schema.org Ontology (Schema.org)[4] [20], Friend of a Friend (FOAF)[5] [21], Business Process Modeling and Notation Ontology (BPMNO)[6] [22], Description of a Project (DOAP)[7] [23], and Enterprise Ontology (EO) [24][8].We consider UDOs as

[3]Describes the world of financial businesses.
[4]Develops common vocabulary for internet semantics.
[5]Models relationships between people.
[6]Models the notational system of Business Process Modeling and Notation (BPMN) as an ontology.
[7]Defines the entities of a software project.
[8]Defines terms and definitions relevant to business enterprises.

a very promising source of information, since their domains are relatively wide, but still concrete. They explore them thoroughly and we expect them to model classes that are also required in our domain (e. g. people, organizations). These classes can then be analyzed and serve as inspiration for our work.

DOs are specialists. They focus on a narrow domain and its vocabulary, classes, and relationships. [15, p. 3] Examples are: This work, the Wine Ontology from the W3C OWL guide [14], and many more that can be found in the ↪Protege Ontology Library.

Metadata Schemas Metadata Schemas provide the scaffolding for an ontology. They implement a vocabulary that helps organizing the knowledge and add structure by standardization. Examples are: Resource Description Framework Schema (RDFS) [25], Simple Knowledge Organization System (SKOS) [26], and Dublin Core Metadata Terms (DCMT) [27]. As with the grouping of ontologies, categorizing something as a *metadata schema* is not necessarily clear-cut. The schemas themselves can be considered ontologies and some of them are very extensive. The DCMT, for example, is not limited to metadata and properties. It also contains classes such as `Agent`, `Location`, or `Policy` that would indicate a classification as UDO. However, since they typically occupy their own name space (e. g. when used in annotation properties, see section 2.2.4), it is easily possible to *mix and match* and individually select the parts required. This work primarily uses the annotation properties (e. g. `rdfs:label`) from these schemas.

Project Management Concepts Projects management is a big part of the consulting domain. As such, it also plays a very important role for SCOs. It is an overarching and very general subject matter and contains many concepts of complex nature: Time, problem analysis, project structuring, etc. It is easy to find many different books, theories, or models about these topics. Each brings their own approach, flavor, scope, and goal for project management. Examples are: The Project Management Body of Knowledge [28] for a general and universal project management concept, SCRUM [29] for a specific type of software development. We use these concepts as guidelines for our domain classes and descriptions.

2.4. Principles and Assumptions

Ontologies are a powerful tool, rooted in description logic. They build on this ancestry as well as its rigorous- and correctness to try to describe the world with a structured approach. But even though an ontology *can* be correct, its development process is not a hard science. It is hard
?
to describe the world and there exist ~~underlying~~ assumptions, principles, and various trade-offs
subtle
that impact an ontology's design. [30, p. 383–394]

This poses the challenge to make good design decisions and select adequate abstractions of the world. But it also gives the ontology developer some room and flexibility to impose a design idea on the work.

Our primary principle is simplicity. This manifests in various ways during the design process. It is our goal to make this ontology as accessible and as easy to understand as possible, because its potential users are not necessarily experts. Since it is more a *design philosophy*, we will not document every single occurrence. We will, however, describe our reasoning in a few important instances:

First: Avoid complexity by allowing inference of details High information-density can be useful, if the goal is a highly detailed model. However, a smaller and more concise ontology is easier to grasp and process. Since we strive for simplicity, we reduce the size and complexity of our ontology by omitting information (e. g. certain relations or attributes) that can be inferred from linked ontologies when required.

For example: Both 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`. Instead of copying these attributes to our ontology, we use this the relation `rdfs:seeAlso` [25, https://www.w3.org/TR/rdf-schema/#ch_seealso] as a soft reference for attributes that can be extracted on demand.

Second: Avoid content overload by utilizing the open world assumption OWL, our ontology language of choice, is built on the Open World Assumption[9]. [31, p. 388–389] [32, p. 299, 417] [33, p. 372] to *hide* unhelpful information.

The first example is the description of the SCO's problem space. It is as varied as their clients, since it is easily conceivable to offer consulting in any area imaginable: Digitization, Human Resources, Knowledge Management, Market Research, Marketing, Corporate Strategy, etc. Depending on the individual SCO instance, each topic can be considered part of the consulting domain and should therefore be included in the model. However, this would lead to an infinite list of possible topics and would quickly extend the scope far beyond the true intention of the ontology.

Another example are IT systems. They are an essential and often integral part of modern business. The same is true for consulting companies. The systems are used to support, supplement, and optimize all processes. However, they are not the core value proposal of consulting companies. Hence, a detailed model of the IT systems would not contribute in a meaningful way to the ontology, even though we can assume that every SCO uses these kind of systems to facilitate their business.

Third: Allow individual complex classes Ontologies allow dissecting the world to a high degree of detail. Some real-world objects, however, posses inherent polysemy. Decomposing them can introduce unneeded or—in our case—unwanted complexity, without offering a significant value-gain.

The classic example for inherent polysemy is the book. It can be viewed from two different perspectives: The physical object and its contents[10]. Another example from our domain is the contract: It is a document that captures a business agreement. The term *contract* can refer to the immaterial agreement between the parties, but it can also refer to the physical document. ?

Even though it's *possible* to decompose these objects, it's not *necessary*. The usefulness of such a decomposition depends on the use case. In our case, with the focus on simplicity, it is enough to describe most classes on a high level, since the exact design is up to the user. *Arapinis* argues in [34] to allow the use of complex classes to avoid introducing incoherence and inconsistency.

[9]The Open World Assumption states that if a value is not stated, it's `unknown` opposed to the Closed World Assumption that would default to `false`.

[10]"physical object reading" and "abstract informational reading". [34, Section 2.1]

3. Ontological Aspects in the SCO Domain

In? 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*.

d?

Please note, that there are other classes and relations present in the ontology—and also describes in chapter 4—that are relevant to the domain, but *not* discussed in detail in this part of the work (e. g. `Document`).

3.1. Context

Context plays an important role in knowledge engineering. Things can take on a different meaning, depending on which context they are presented. Accounting for it poses a challenge for any ontology, because of its inherent complexity and domain specificity [35, p. 1]. It is important to note that context can, but doesn't have to be modeled explicitly. It indirectly influences all entities; however, the degree of influence can change from one to another.

To better understand how context is handled in 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." [36, 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. Primary 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 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" [37]

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. The content of each individual project does not impact the ontology. Hence, we can look at each project 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.

3.1.2. Contextual Ambiguity

Ontology engineering relies on natural language to describe concepts. This work uses primarily English words (see subsection 2.2.1) for this task. That naturally introduces a form of contextual ambiguity, because people often use words in a flexible manner. [38, 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-definied name spaces and established ontologies (see section 2.3).

We are using the latter approach and try to avoid additional ambiguity by discussing the relevant contexts explicitly as part of this work, as well as add explanation texts to the ontology. Furthermore we try to lean on common understand of terms as much as possible, by creating links to well-known ontologies like FOAF.

ing

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

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

that do stuff" [21, http://xmlns.com/foaf/spec/#term_Agent]. It is connected to the name space of the 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 **`fibo:AutonomousAgent`**[10]. It is sub-classed by **`fibo:Person`**[11], representing individual humans. Like in FOAF, this class is disjoint with **`fibo:Organization`**[12]. **`fibo:Group`**[13] exists as a `subclassOf` **`fibo:Collection`**[14] and is described as collection of **`fibo:AutonomousAgent`**, which in turn is `subclassOf` **`fibo: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]`foaof:Organization rdfs:comment: "An organization."`
[4]`foaof:Person rdfs:comment: "A person."`
[5]Note: The ontology doesn't offer any description.
[6]See footnote 5.
[7]The 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]`fibo-fnd-aap-agt:AutonomousAgent skos:definition: "An agent is an autonomous individual that can adapt to and interact with its environment."`
[11]`fibo-fnd-aap-ppl:Person skos:definition: "a person; any member of the species homo sapiens"`
[12]`fibo-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]`fibo-fnd-org-fm:Group skos:definition: "a collection of autonomous entities"`
[14]`fibo-fnd-arr-arr:Collection skos:definition: "a grouping of some variable number of things (may be zero) that have some shared significance"`
[15]`fibo-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 do not 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.

who?

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.

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.

As already shown by *Loebe*, the concepts and ideas about roles have been heavily discussed in the ontology community and literature. [39, 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

[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."`

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. [39, p. 131–133]

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 Organizational Context (OC). Additionally, the same **`Member`** could work on a project as **`Project Leader`**, a role from the Project Context (PC).

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 also have implications for the roles that exist~~s~~ in the domain.

3.2.2. Roles in the Organizational Context

Membership Within the OC (see section 3.1), the most basic property is membership: Either being part of the organization and thus being a **`Member`** or not participating and being a **`Non-Member`**. **`Members`** of the SCO can play different roles in the OC. **`Non-Members`** do not play roles within the organization, but can play external roles such as 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`**.

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.

Corporate Officers The complete set of tasks and responsibilities for the organization's day-to-day leadership is associated with the group of people referred to as Coporate 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; 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. d

The exact set of tasks differs from SCO to SCO and is also dependent on the organizational form, e. g. an 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 that this ontology operates on. In the same vein, it is impossible to define which tasks are attributed to which role, since every SCO organizes itself 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.

Alumni, Advisor, and Patron In the duality of membership—**`Member`** vs. **`Non-Member`**—exist 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. In many cases 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: ↪Éminence grise).

3.2.3. Roles in the Project Context

As discussed in section 3.1, we look at projects as black boxes. We focus on their inputs and outputs, as well as the support during its life cycle. This perspective influences the role model, since the primary inputs are people playing a certain role in the PC. The *roles* of a project are: the team—consisting of team leader and team members—, the controlling entity, and the customer. The roles in the PC operate on a higher, more general level of abstraction when compared to the classes in the OC, to account for the generality if projects.

Project Team A project team in its most basic form consists of team members that are lead by a team leader. The project team leader is typically a more experienced member of the organization, to ensure the knowledge transfer and successful operation of the project, whereas the team members are recruited from the general member pool.

We model `Project_Team_Member` and `Project_Team_Leader` as `subclassOf` `Project_Role`. To indicate that these roles are only played by members of the organization, both are `playedBy` `only` `Alumna` or `Consultant` or `Junior_Consultant` or `Senior_Consultant`. Additionally the `isPlayedBy` relation of the role `Project_Team_Member` is extended with `or` `Trainee`, to account for the fact that trainees in a SCO can not become the `Project_Team_Leader`. By using `Trainee` our model binds the cut-off point for the leadership role to the formal membership (see 3.2.2).

The `Project_Team` itself is not a role. However, it is a relevant entity of the ontology. It is a collection of certain `Members` of the SCO, that each either play the role of a team member or a leader. We model it as `subclassOf Group`. This also gives it the ability to act as an `Agent`: It `participatesIn` a `Project` and `signsContract` the `Project_Contract`.

Project Controlling In addition to the team, the SCO typically provides a controlling entity for the project. It observes and measures the progress of the project, gives feedback on the project work, and helps the team in various capacity where necessary. The role of the controlling entity is usually played by someone or some group—both are common in the domain—, that has gathered experience with projects in general and has specific knowledge that is useful in the project it supports. Additionally, using an experienced controlling entity helps the learning process of the team.

We model `Project_Controlling` as `subclassOf Project_Role`. We model the players as a group of experienced members; if an organization only uses a single member to play the controlling entity, our models consider~~s~~ it a group of one ($n = 1$). We encode the required experience into the ranks (see 3.2.2), cutting out `Trainee` and `Junior_Consultant`. This means the role `isPlayedBy only` (`Group` and `hasMember` some (`Alumna` or `Consultant` or `Senior_Consultant`)).

Customer The remaining party of a basic project is the `Customer`. It represents the actor that wants to reach a goal and sets up a project with the SCO to reach it. Similar to other aspects in the PC, we use a minimal approach to the role to accommodate for the required flexibility. We model it as `subclassOf Project_Role` and allow it to be played by any `Agent`. To work on a `Project`, the `Customer signsContract` the corresponding `Project_Contract`.

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. [40, 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. They often revolve around 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. [41] Because process documentation contains a lot of data about organizations, it is also a valuable source for ontology development.

Widely known representations and methods include: Flowcharts, BPMN, Event-Driven Process Chain (EPC), Unified Modeling Language (UML) Activity Diagrams, and Object Process Methodology (OPM)[22]. There are also contributions rooted in ontology research, such as the

[22]Standardized as ISO 19450.

?

BPMN ontology (an OWL ontology for the BPMN notation) [22], the Process Specification Language (PSL)[23], and processes concepts as part of GFO or BFO.

Our intent is the representation of the most important processes of an SCO in such a way, that it makes it clear to the ontology user which process have to be implemented, but leaves enough creative freedom in regards to how to implement them.

has

3.3.1. Implementation in Related Ontologies

When compared to the rather practical and direct implementation of social structures discussed in subsection 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 link from FOAF to a process representation is the class **`foaf:Project`**[24], 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." [42] Hence, it is not surprising, that it doesn't implement a detailed process representation. The other time-related class that could be connected to processes, **`schema:Event`**, is geared towards content description (e. g. **`schema:Business_Event`**, **`schema:Sports_Event`**) and website interaction (e. g. **`schema:User_Likes`** `subclassOf` **`schema:User_Interaction`**) instead.

On the other hand, the two top-level ontologies, BFO and GFO, implement process concepts on such a high level of abstraction, that they are not directly applicable for our domain process model: BFO uses the class **`bfo:Occurent`**[25] as entry point for its process concepts. It is sub-classed by **`bfo:Process`**[26], **`bfo:Process_Boundary`**[27], **`bfo:Spatiotemporal_Region`**[28], and **`bfo:Temporal_Region`**[29], to enable it to distinguish between the process itself and its temporal parts. [17, p. 66–68] Looking closer at the given examples[30] for **`bfo:Process`** further emphasizes the classes high-level nature. GFO uses the high-level term *Temporal Complexes* to denote the

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

[24]`foaf:Project rdfs:comment: "A project (a collective endeavour of some kind)."`

[25]`bfo: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])"`

[26]`bfo: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])"`

[27]`bfo: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])"`

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

[29]`bfo: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])"`

[30]`bfo: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.`

"most general kind of concrete individuals which have a temporal extension" [16, p. 26], which includes processes: "The set of processes is a proper subset of the set of connected temporal complexes." [16, p. 27]. In the ontology, the class **`gfo:Processual_Structure`**[31] represents its entry point. It is sub-classed by **`gfo:Occurent`**[32] and **`gfo:Processes`**[33].

GIST is also not directly applicable, since it does not account for processes. It also deals with time on a comparatively high level with the class **`gist:Event`**[34] that uses **`gist:Time_Instant`** to specify start and end of the event.

Note: Even though GIST is not useful for our *process* model, it offers an intuitive approach for the other time-related concepts that we can apply to our ontology, namely *tasks* and *projects*: A **`gist:Project`**[a] is considered a longer duration task that is made out of a number of other **`gist:Tasks`**[b]. **`gist:Project`** is `subclassOf` **`gist:Task`** is `subclassOf` **`gist:Event`**. We share this understanding of projects and tasks and can therefore link it in our ontology: **`Project`** `rdfs:seeAlso: "gist:Project"`.

[a] `gist:Project rdfs:comment "A project is a task (usually a longer duration task) made up of other tasks."`

[b] `gist:Task rdfs:comment: "A task which has been defined and either scheduled or accomplished, or both."`

FIBO also does not offer a process model. The time related concepts it offers have a small overlap with GIST in the class **`fibo:Time_Instant`**. Additionally it offers **`fibo:Time_Interval`** and **`fibo:Time_Direction`**. However, their sub-classes—for **`fibo:Time_Instant`**: **`fibo:Date`**, **`fibo:Date_Time`**, **`fibo:Date_Time_Stamp`**; for **`fibo:Time_Interval`**: **`fibo:Calendar_Period`**, **`fibo:Duration`**, **`fibo:Date_Time_Stamp`**, **`fibo:Recurrence_Interval`**—make it clear, that it focuses more on accurately modeling calendar dates and operations, instead of procedures.

DOAP does not reference processes at all.

3.3.2. Structure of the Class Hierarchy

Since none of the related works offer a model to make it applicable in our ontology, we have to develop the structure for processes according to the general ideas and principles we outlined before.

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 2.2.3, the built in `subclassOf` relation of the class hierarchy already carries semantic meaning, that is generally not applicable to processes. For

[31] `gfo: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."`

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

[33] `gfo: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."`

[34] `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."`

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 a relation to connect a sub-process to its parent process. This method utilizes the core concepts of ontologies, classes and relations, and avoids encoding extra information in unconventional ways. This, however, results in a loss of readability because of a completely flat structure of the class hierarchy: every process is directly `subclassOf` **`Process`**, independent from its level of abstraction.

We make use of the second approach. To compensate for the resulting reduced readability, we add diagrams to describe the processes and their relationships graphically (see appendix A.2.2).

3.3.3. Relations Between Processes

After establishing the structure of processes in the class hierarchy, we have to describe the relationships between them. Processes embody a kind *of order of operations* and as such, an order relation is the first thing that comes to mind to describe them. However, within our domain exists different types of processes. Some can be strictly or semi-strictly ordered and some that can not. Additionally processes can have sub-processes, since processes can be decomposed into individual actions and these sub-processes—or even actions—can overlap with parts from previous processes.

For example: In the beginning of a project, it is planned and its exact goal has to be defined. Both—the project plan and the project goal—are stated in the project contract that is signed by the project parties and hence share the moment they are *finished*. But since project planning and goal development are both very complex tasks and they often influence each other, they are often worked on at the same time. Their ordering needs to be: *At the same time*. However, all the procedural details can change from project to project. Sometimes one step indeed finishes before the other: *At the same time* would not be exact anymore. It is impossible to exactly define their order in a general manner without a complete implementation for a ontological process system, which is out of scope of this work.

But since processes play a very important role in the domain and are required, we introduce simple process relations, that sufficiently model the relationships between the processes for this particular domain:

1. If a process is guaranteed to start before another, it can use `nextProcess` to point to the follower. Its inverse is `previousProcess`. This relation allows overlap between the processes.

2. If a process is part of another process (see subsection 3.3.2), it can point to its parent using `isProcessPartOf`. Its inverse is `hasProcessPart`. This relation does not express anything about the ordering of the process parts. They may be ordered between each other via `nextProcess`, but do not necessarily have to be.

This model makes no statements about the actual implementation of the processes, since it can differ from one SCO to the other. This also means that implementation details such as process properties (e. g. process duration, process responsibility) are deliberately left out.

3.3.4. Processes Selection and Level of Abstraction

As stated in the introduction of this section (see 3.3), the goal is to describe the processes in
a such way that makes it clear which processes have to be implemented, not how. At the extreme, To
a very detailed process enforces everything, which goes contrary to our goal.

Therefore it is necessary to identify the relevant processes and select their correct level of abstraction individually. This also includes a cut-off point for the level of details. On the one hand, each process class should convey enough information to understand its purpose. On the other, it should not be overloaded. This approach helps to ensure that the ontology user is able to understand the process system of the domain as a whole.

For example: German law requires every company—and therefore also every SCO that does paid project work—to pay taxes on their earnings. Depending on the way projects are handled, this influences the process that is concerned with taxation. It is commonly known, that the German taxation system is daunting and hard to understand for a layperson and including the complete taxation process required by law is out of scope of the domain model. However, interacting with the tax authorities and filing the correct paper work is an important part of the learning experience for student consultants. It is therefore also important for the ontology. However, each SCO handles this differently, so it can not be modeled generally. In this case, the selected cut-off point is very high-level: The process is condensed into the class *Project Taxation Process* as part of the *Project Process*. This makes it clear that an SCO has to deal with the project taxation, but does not prescribe how to do it.

To select the relevant processes, we use a top-down approach based on expert knowledge and the HC process documentation [43]. The cut-off point is selected for every individual process in the ontology.

3.3.5. Core Processes

The primary goal of an SCO is teaching students project work. This involves training their members and offering them opportunities to work on real-world projects. The topmost level of the process perspectives has to reflect this goal, by formulating the core actions an SCO has to perform: 1. Members have to be recruited and have to be taught the necessary skills, to be able to work on projects. 2. Projects have to be acquired and worked on by members. All other processes (e. g. technical support, legal support, or marketing processes) only exist to support these actions.

The HC Process Documentation [43] calls[35] these two processes Human Resource Process (HRP) and Project Process (PP). The sub-processes of the HC HRP focus on recruitment, training, and generally enabling of the SCOs members. The HC PP establishes the way the organization handles projects from start to finish. Both of these labels are also very commonly found in the business world. We adapt the naming for our ontology.

Since all other processes are supplementary and the ontology is focused on simplicity, we exclusively model these two core processes and their sub-processes up to a certain depth. Furthermore it is important to note, that each core process can be viewed from two different perspectives:

be

On the one hand, it can viewed as the process of the organization. For example: The SCO itself has a HRP. It structures important aspects of the organization. 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 of the organization on an abstract level and knowingly omits parts of the real world process that are not important to the organization.

On the other hand, it can be viewed from an individuals perspective. Each member has her own instance of an HRP. For example: The protagonist is one individual student. She is following her own instance of the process from the individuals perspective; this instance does not have to be identical with an instance of a second individual. Both individuals might partake in the education process, but might do different courses. Both individual might hold the same rank in the organization, but ~~might~~ their membership duration might be different.

s

Both perspectives are relevant to the reality of an SCO. However, as discussed in section 2.1, the individual instance are not part of our model. This holds for the individual HRP instance as well. Furthermore, as we consider the individual project as a black box (see section 3.1), the PP in our ontology is viewed only from the perspective of the organization.

The detailed descriptions of the rest of the modeled processes can be found in the ontology (see chapter 4) and the OWL file.

[35]Translated from German.

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.

es

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

What are we going to.... for?

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 authors. 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

- Agent
  - Group
    - Executive_Board
    - Member_Assembly
    - Project_Team
  - Organization
    - Student_Consulting_Organization
    - Umbrella_Organization
    - University
  - Person
    - Member
      - Consultant
      - Junior_Consultant
      - Senior_Consultant
      - Trainee

Agent

- All members except trainees and almunus can be corporate officers ◇
- Non-members can not 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 and Further Research

A. Appendix

A.1. Term Enumeration

A.2. Diagrams

A.2.1. General Diagrams

Figure A.1.: Ranks

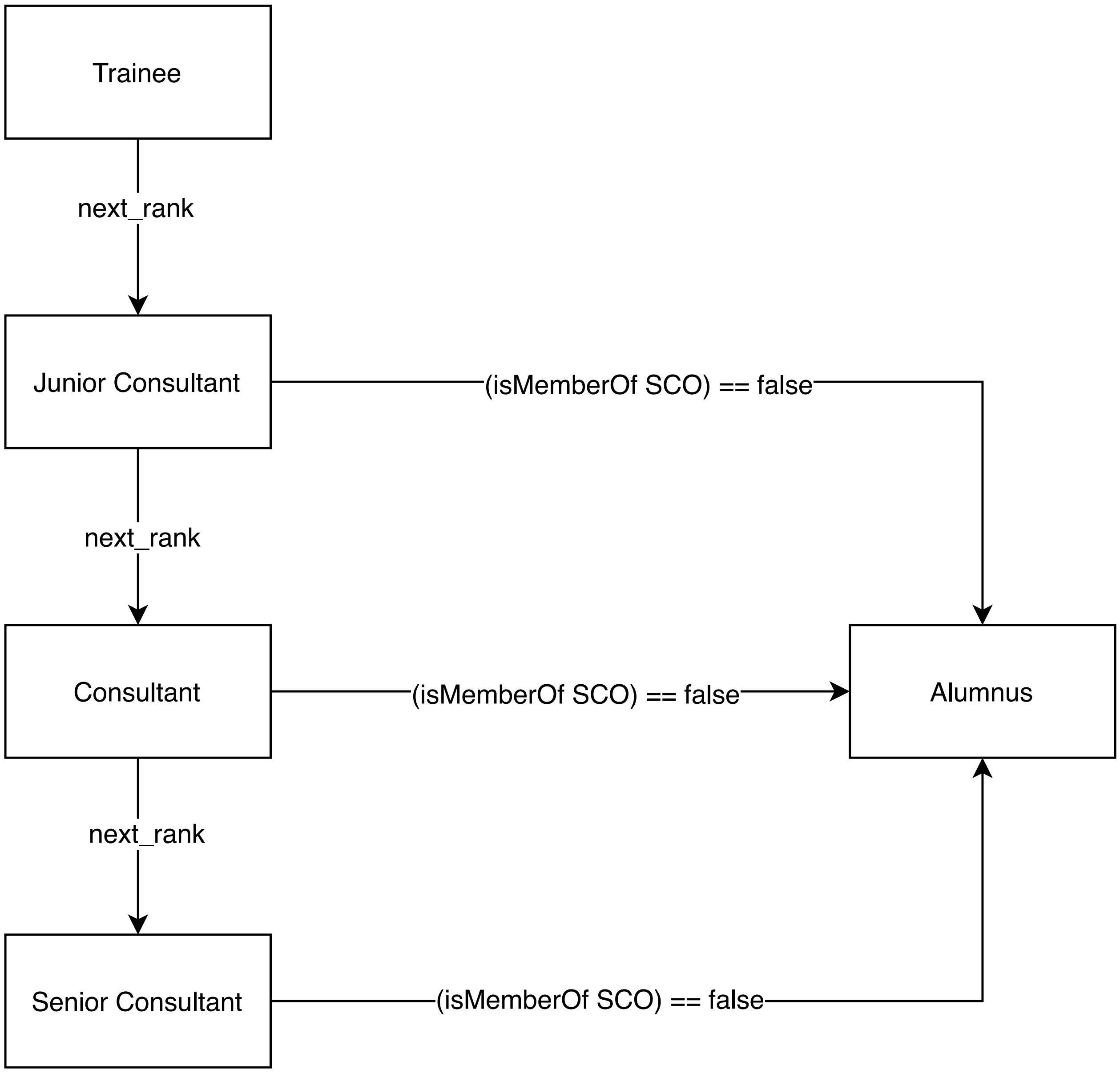
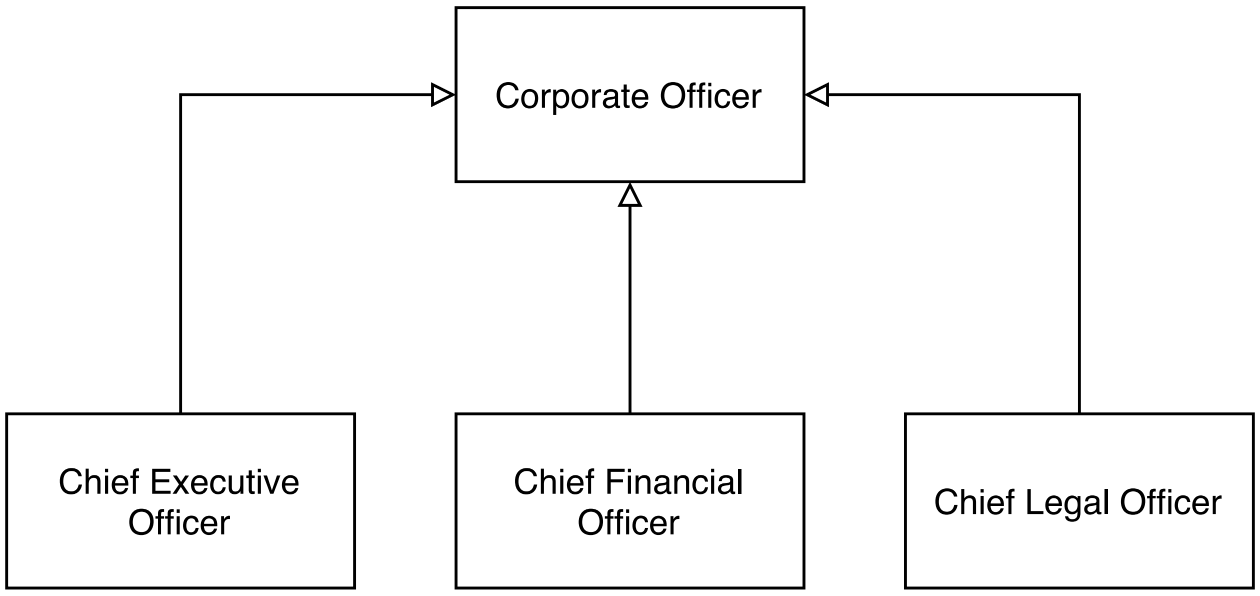


Figure A.2.: Corporate Officers



A.2.2. Process Diagrams

Figure A.3.: Project Process

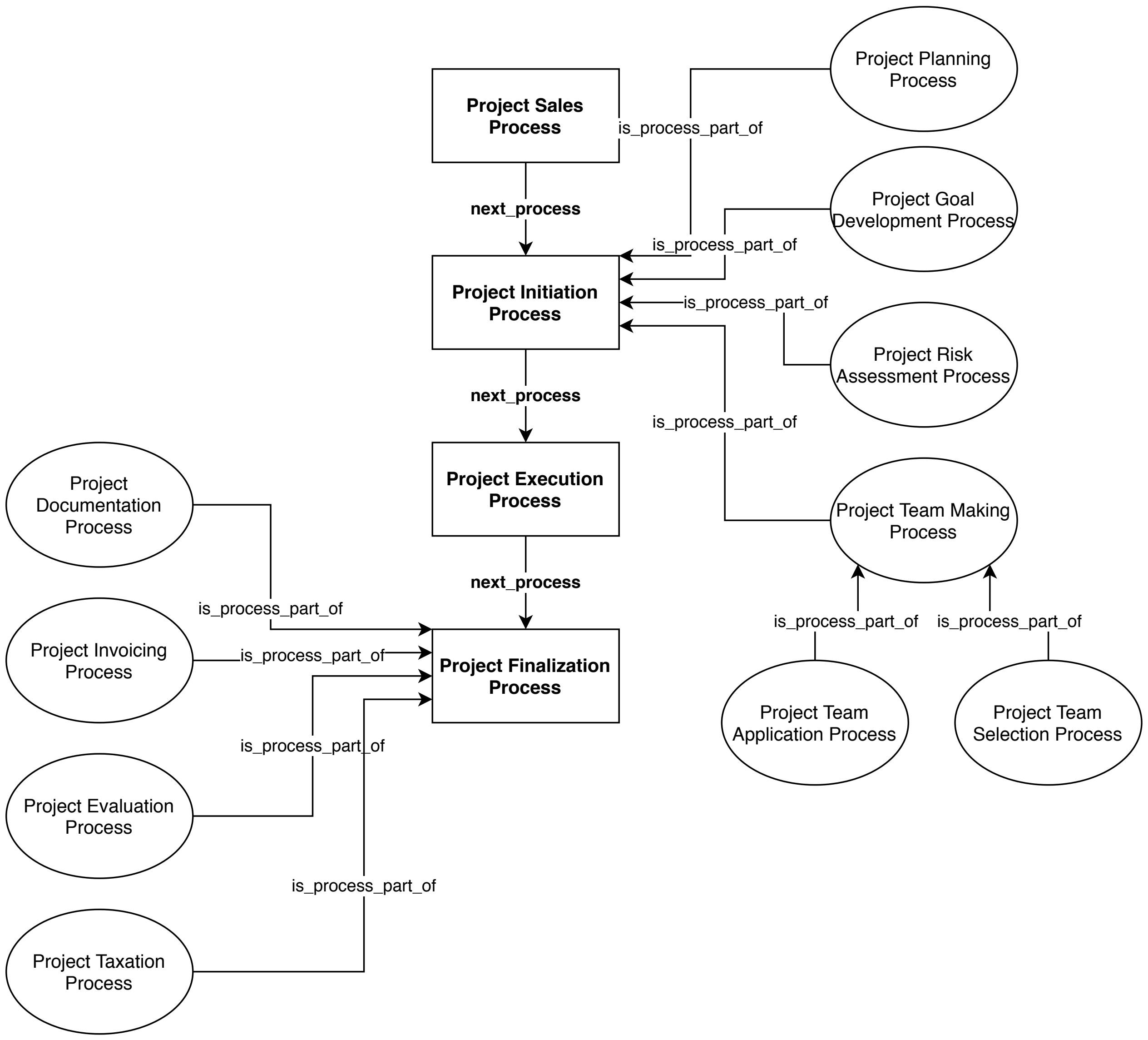
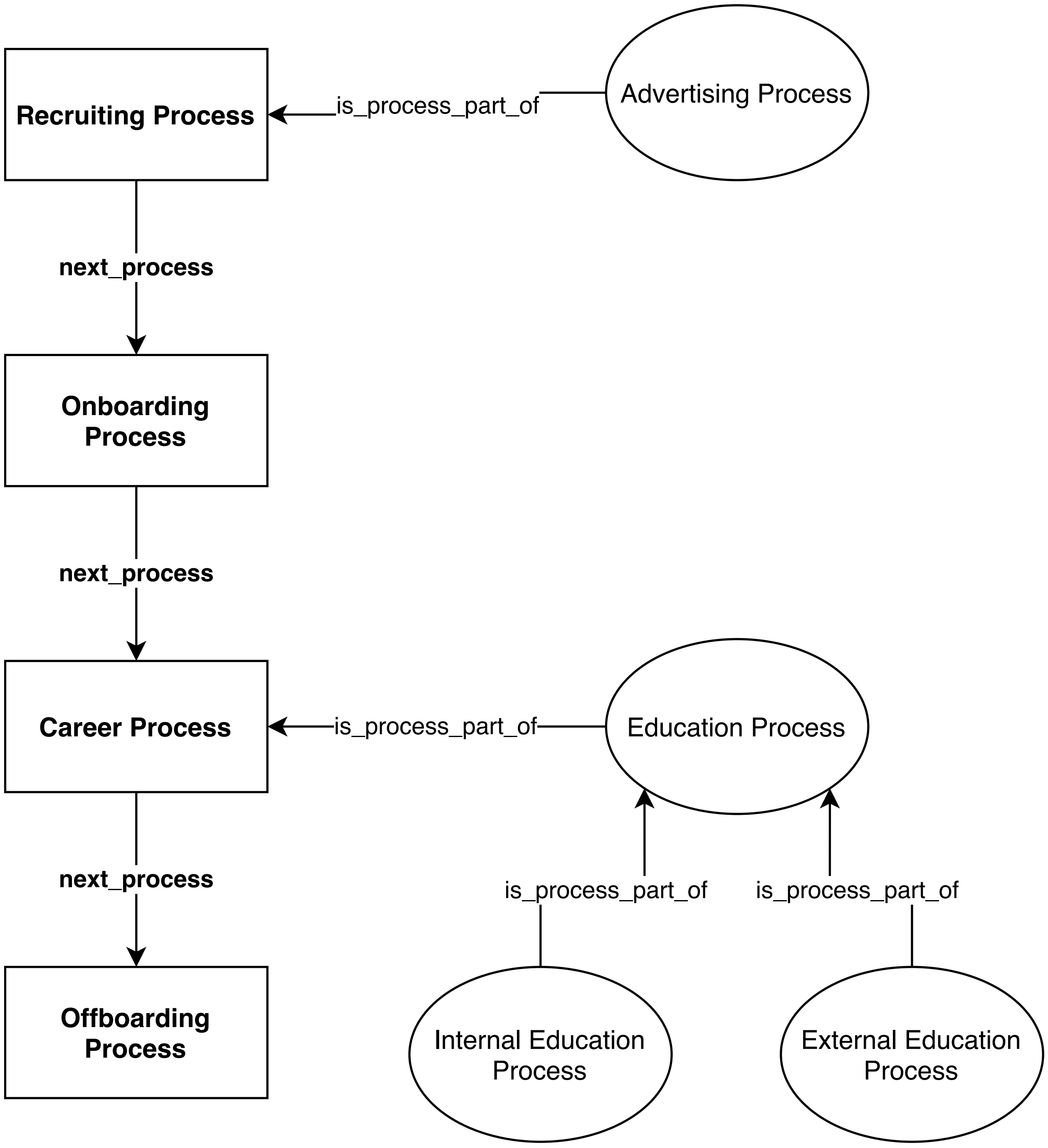

Project Sales Process
next_process
Project Initiation Process
next_process
Project Execution Process
next_process
Project Finalization Process
Project Planning Process
is_process_part_of
Project Goal Development Process
is_process_part_of
is_process_part_of
Project Risk Assessment Process
is_process_part_of
Project Team Making Process
is_process_part_of
is_process_part_of
Project Team Application Process
Project Team Selection Process
Project Documentation Process
is_process_part_of
Project Invoicing Process
is_process_part_of
is_process_part_of
Project Evaluation Process
is_process_part_of
Project Taxation Process

Figure A.4.: Human Resources Process



A.3. Glossary

BDSU ↪Bundesverband Deutscher Studentischer Unternehmensberatungen.

BFO Basic Formal Ontology.

BPMN Business Process Modeling and Notation.

BPMNO Business Process Modeling and Notation Ontology.

CEO Chief Executive Officer.

CFO Chief Financial Officer.

CI ↪Campus Inform.

CO Coporate Officer.

COO Chief Operating Officer.

DCMT Dublin Core Metadata Terms.

DO Domain Ontology.

DOAP Description of a Project.

EO Enterprise Ontology.

EPC Event-Driven Process Chain.

FIBO Financial Industry Business Ontology.

FOAF Friend of a Friend.

GFO General Formal Ontology.

GIST GIST.

HC ↪Hanseatic Consulting.

HRP Human Resource Process.

ISO International Organization for Standardization.

JCNetwork ↪Junior Consultant Network.

NIST National Institute of Standards and Technology.

OC Organizational Context.

OPM Object Process Methodology.

OWL Web Ontology Language.

PC	Project Context.
PM	Project Management.
PP	Project Process.
PSL	Process Specification Language.
RDFS	Resource Description Framework Schema.
Schema.org	Schema.org Ontology.
SCO	Student Consulting Organization.
SKOS	Simple Knowledge Organization System.
TLO	Top-Level Ontology.
UDO	Upper-Domain Ontology.
UML	Unified Modeling Language.
W3C	World Wide Web Consortium.

A.4. Bibliography

[1] 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/bestandenepruefungen-studiendauer.html` (visited on 2020-03-01).

[2] 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.

[3] Franz Baader et al. *Introduction to Description Logic*. Cambridge University Press, 2017. ISBN: 978-0-521-87625-4.

[4] Mark A Musen. “The protégé project: a look back and a look forward”. In: *AI matters* 1.4 (2015), pp. 4–12.

[5] 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).

[6] Heiner Stuckenschmidt. *Ontologien: Konzepte, Technologien und Anwendungen*. Springer, 2010. ISBN: 978-3-642-05403-7.

[7] 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.

[8] Merriam-Webster.com. *Dictionary*. online. URL: `https://www.merriam-webster.com/dictionary/`.

[9] Frank Loebe. “Ontological Semantics: An Attempt at Foundations of Ontology Representation”. PhD thesis. Universität Leipzig, Mar. 2015.

[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] 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.

[12] Thomas R Gruber et al. “A translation approach to portable ontology specifications”. In: *Knowledge acquisition* 5.2 (1993), pp. 199–221.

[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] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. *OWL Web Ontology Language Guide.* online. Feb. 2004. URL: `https://www.w3.org/TR/owl-guide/`.

[15] Asunción Gómez Pérez and V Richard Benjamins. “Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods”. In: *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving methods (KRR5), Stockholm, Sweden.* 1999, pp. 1–15.

[16] 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.

[17] Barry Smith et al. “Basic formal ontology 2.0”. In: *Specification and User’s Guide* (2015).

[18] Semantic Arts. *gist - minimal upper ontology.* online. Apr. 2019. URL: `https://www.semanticarts.com/gist/` (visited on 2020-06-12).

[19] EDM Council. *Financial Industry Business Ontology (FIBO).* online. Oct. 2019. URL: `https://spec.edmcouncil.org/fibo/` (visited on 2020-06-12).

[20] Schema.org. *Schema.org.* online. URL: `https://schema.org/docs/schemas.html` (visited on 2020-03-16).

[21] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.99.* English. Jan. 2014. URL: `http://xmlns.com/foaf/spec/` (visited on 2020-02-27).

[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] Edd Wilder-James Edd Dumbill. *Description of a Project Ontology (DOAP).* online. URL: `http://usefulinc.com/ns/doap` (visited on 2020-06-12).

[24] Enterprise Project by the Artificial Intelligence Applications Institute at the University of Edinburgh. *The Enterprise Ontology.* online. URL: `http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html` (visited on 2020-06-12).

[25] Dan Brickley, Ramanathan V Guha, and Brian McBride. “RDF Schema 1.1”. In: *W3C recommendation* 25 (2014), pp. 2004–2014.

[26] Alistair Miles and Sean Bechhofer. “SKOS simple knowledge organization system reference”. In: *W3C recommendation* 18 (2009), W3C.

[27] DCMI Usage Board. *DCMI Metadata Terms*. online. Jan. 2020. URL: `https://dublincore.org/specifications/dublin-core/dcmi-terms/` (visited on 2020-06-13).

[28] Project Management Institute Inc. *A guide to the project management body of knowledge (PMBOK guide)*. Sixth edition. Newtown Square Pennsylvania EE. UU., 2017. ISBN: 978-1-628-25184-5.

[29] Jeff Sutherland and Ken Schwaber. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*. online, pdf. Nov. 2017. URL: `https://www.scrum.org/resources/scrum-guide`.

[30] John F Sowa. *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co., 1999.

[31] 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.

[32] Stuart Russell. *Artificial intelligence : a modern approach*. 3th ed. New Jersey: Pearson, 2010. ISBN: 978-0-136-04259-4.

[33] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. CRC press, 2009.

[34] Alexandra Arapinis and Laure Vieu. “A plea for complex categories in ontologies”. In: *Applied Ontology* 10.3-4 (2015), pp. 285–296.

[35] 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.

[36] 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.

[37] International Organization for Standardization. *Quality management systems - fundamentals and vocabulary (ISO 9000:2015)*. 4th edition. [Geneva]: ISO Copyright office, 2015.

[38] Mike Uschold et al. “The enterprise ontology”. In: *The knowledge engineering review* 13.1 (1998), pp. 31–89.

[39] Frank Loebe. “Abstract vs. social roles–Towards a general theoretical account of roles”. In: *Applied Ontology* 2.2 (2007), pp. 127–158.

[40] Mathias Weske. *Business Process Management : Concepts, Languages, Architectures*. Springer, 2019. ISBN: 978-3-662-59432-2.

[41] International Organization for Standardization. *The Process Approach in ISO 9001:2015 (ISO/TC 176/SC 2/N1289)*. online. 2015. (Visited on 2020-03-10).

[42] Schema.org. *About Schema.org*. online. URL: `https://schema.org/docs/about.html` (visited on 2020-03-16).

[43] Hanseatic Consulting Hamburg. *Prozesshandbuch*. Internes Dokument. May 2014.

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
   "the historical context of the war"

domain NOUN — MERRIAM-WEBSTER

do · main

1. law
   a) complete and absolute (see absolute sense 3) ownership of land
      "our highways and roads have been in the domain of state and local governments— T. H. White b. 1915"
      — compare EMINENT DOMAIN
   b) land so owned
2. a territory over which dominion (see DOMINION sense 2) is exercised
   "The forest is part of the king's domain."
3. a region distinctively marked by some physical feature
   "a domain of rushing streams, tall trees, and lakes"
4. a sphere (see SPHERE sense 4b) of knowledge, influence, or activity
   "the domain of biblical scholarship", "outside the domain of city police"
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)
    "The domain ca is used for sites located in Canada."
    also : DOMAIN NAME
    "Our domain is Merriam-Webster.com."

vocabulary NOUN — MERRIAM-WEBSTER

vo · cab · u · lary *plural* vocabularies

1. a list or collection of words or of words and phrases usually alphabetically arranged and explained or defined : LEXICON
   "The vocabulary for the week is posted online every Monday."
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.