Semantic Technology and ISO 15926

An Introduction

December 5, 2008

Overview

These are the lecture notes to an introductory course to the use of semantic technology in general, and ISO 15926 in particular, as a tool for providing data exchange standardization for industries, businesses, government agencies, and other organizations. The lecture notes should be read together with the slides for the course, which can be found at (TDB). The course is divided into two modules. The first, corresponding to Chapter 1 of these notes, is a first introduction to semantic technology and ISO 15926 for a general audience. The second module, corresponding Chapter 2 and Chapter 3, is a more technical introduction to the core parts of ISO 15926 for people who plan to use ISO 15926 for the modeling of a particular domain.

Contents

Chapter 1

General Introduction to Semantic Technology and ISO 15926

Ontologies in general, and ISO 15926 in particular, provide a partial solution to a data system integration problem which is ubiquitous both in industry and public administration. This introduction will describe challenges in data integration and how semantic technologies and ontologies, in particular, can help to support data interoperability. We also briefly sketch what the standard ISO 15926 is, and how it fits into this picture.

Slide 2

The current chapter will focus on challenges to data integration and how semantic technology and ISO 15926 can be used to address them. Section 1.1 presents what the so-called *datasheet problem*, and why it is an important and costly problem. A concrete example will be used for illustration. Section 1.2 presents in what way semantic technology and ontologies can be used as a solution. Finally, section 1.3 presents a brief overview of the semantic technology based industry standard ISO 15926, and outlines how it is useful for addressing the data sheet problem in the oil and gas industry and areas with similar needs. An important caveat for Section 1.3 is that ISO 15926 is a work in progress. This text does not indicate which parts of ISO 15926 are completed and which are planned or under development. The lecturer will provide an up to date indication of the status of the different parts of ISO 15926 as he goes through them. The lecturer should also point out which parts are formally ISO standards, and which parts are not (or not yet) ISO standards but rather built in accordance with such standards. This text will use "ISO 15926", or just "15926" as a general name for a cluster of ISO 15925-based semantic technologies. Chapters 2 and 3 will give a more detailed and technical introduction to the core structure of ISO 15926.

1.1 Background Problem and Business Case

3

1.1.1 The Datasheet Problem

The need for semantic technology and ontologies arises from a lack of standards for representing information. For instance, the same information, regarding some piece of machinery, say, may be found represented in significantly different ways on different datasheets[1]. This can present companies, government agencies, and other *agents* with serious difficulties, as documentation concerning e.g. products and services may be received in very unfamiliar formats[2]. In order to extract the relevant information from such documentation and translate it into a format with which the agent (company, government agency, or whatnot) is familiar, the agent may have to use specialists in the relevant subject matter, such as engineers or lawyers, which have the background knowledge required to understand what the received documentation is actually (or probably) saying. Naturally, this is both time consuming and costly. What seems, at the outset, to be an almost trivial, if irritating, problem can, and often do, therefore end up causing agents serious time delays and significant expenses. Slide 4

Although this is a ubiquitous and far ranging problem for ICT-based information representation and communication in general, we focus on the special case of translating between datasheets—the *datasheet problem*—as this provides a clear and perspicuous example of the general issue. In what follows, we shall briefly consider different datasheet representations of information in a concrete example in order to illustrate the kind of challenges that translation between datasheets involves. We shall also have a look at the problems that different representations of information may cause in somewhat more depth, before discussing solutions and the role of semantic technology.

1.1.2 An Example—the Emerson Pressure Transmitter

Consider the Emerson electric pressure transmitter on slide 5. The following slides contain two datasheets. One from the supplier SHARECAT specifying data about the transmitter, and one which you—as a NORSOK employee—are to fill out. We will point out a few things that will make this difficult for you (unless, perhaps, you are an engineer), as examples of the kind of issues that the datasheet problem consists in: Slide 5

- On slide 6 we have marked in yellow the entries where there is a straightforward transfer of the data, i.e. where you can just fill out the same text in the NORSOK sheet as you have in the SHARECAT sheet. The bad news is that there are only three of them, or two if you dare not trust that "Indicator" means the same as "Display type". Note also that the headings ("Transmitter", "Function") under which these entries are found are different. Slide 6

[1] A datasheet is a document summarizing the performance and other characteristics of a component (e.g. an electronic component) a sub-system (e.g. a power supply) or software in sufficient detail to be used by a design engineer to design the component into a system (Wikipedia). Datasheets are therefore convenient examples of how to represent information in various forms.

[2] In a wide sense of "format": in addition to the way the information is structured, there may be words with unfamiliar or ambiguous meaning, and a great deal of information may have been left implicit.

Slide 7

- On slide 7, we have boxed the entry "3.05 Process conn size/type", under "ELEMENT/SENSOR", and the data under the heading "Process Connection" which in fact contains the relevant data. Slide 8 shows you how to correctly fill out entry 3.05. As you can see, it is the same data, but in a different format.

Slide 8

Slide 9

- Finally, consider the data in the box on slide 9, under the heading "Area" (it is still data about the product). Here you are given a series of codes that tell you about e.g. the product's "Temperature class". The problem for translation is that such codes are of course highly context dependant. "T5", for instance, could be a code for all sorts of things (the first couple of results of a simple Google search are given on slide 10). You need to know what it is a code for, and knowing e.g. that it is a code for a temperature class may or may not be sufficient.

Slide 10

Typical problems for translation between datasheets are, then:

- Simple text searches, as may be done by a non-expert human or a machine, may not get very far. Data on one sheet may need both interpretation and rewriting in order to be correctly entered onto another, as we saw on slides 6–7.

- The context dependence of codes underlines the need for interpretation and background knowledge for datasheet translation. More generally, it is an example of the kind of *implicit information* that datasheets often contain and rely on, and which one simply may have to know if a translation into another datasheet format is to be possible. The precise meaning of a code or expression given by one sheet is crucial for an accurate translation, but may not be possible to divine from the datasheet by itself. Slide 11 presents an example of four different possible meanings of the term, "mean time between failure", the confusion of which may have serious practical consequences.

Slide 11

Slide 12

1.1.3 Significance of the Datasheet Problem

The datasheet problem, thus, arises because different agents represent various kinds of information in different ways, and therefore fail to easily or properly understand each other's documents. Consequently,

Slide 13

1. there is greater risk and tendency for mistranslation. The quality of data deteriorates as it is passed around between agents that have different formats and standards; and

2. the communication of data/information becomes costly and time-consuming because of the need for translation between formats, a translation which often requires specialists (engineers, lawyers, etc).

Notice that the problem is not limited to cases where several agents have different standards, but can also occur when the same agent has different standards over time, or different standards attached to, say, different phases of a project (such as development vs. fabrication, or installation vs. operation vs. modification).

Slide 14

The extent to which the datasheet problem is a problem depends on the amount of data that is communicated, and the need for maintaining a high quality of that data. As modern business and administration relies to an ever increasing extent on the communication of high quality data, the datasheet problem is set to become more and more important. Slide 16 is a (simplified) graphical representation of a typical situation, where a project contractor needs to communicate data with several suppliers as well as with operators. For a couple of more concrete examples: Slide 15

- (TBD) Slide 16
-
-

Questions, Comments, Discussion

Coffee break.

1.2 Ontologies

Slide 18

1.2.1 Solutions to the Datasheet Problem

Last section presented the datasheet problem: essentially (variations of) the problem that since different companies use different systems and formats to represent information, your company may be sent documentation that you are not sure exactly how to read. Now, since this is a problem which arises from a lack of standards, an obvious solution is to have companies and other agents agree on, and introduce, common standards. If two agents communicate using a common standard, then the datasheet problem does not arise. Slide 19

However, the idea of agreeing on common standards has its limitations, as there are, of course, good reasons for using different formats in different contexts rather than forcing the use of one, single standard format. Simply put, in many or most situations, you want to represent the information you want in a format which is tailored to your needs, rather than in a format which is developed to fit all. Using a common standard might mean using a format which is not the most efficient for you, because it also needs to incorporate somebody else's needs. So different agents may want to be able to use and develop different formats, rather than adopt the straightjacket of a common one. Slide 20

Another solution would be to let agents get on with using and developing whatever idiosyncratic standards they prefer, but adopt a common standard for independent representation of information. As illustrated on slides 21 and 22, this can greatly reduce the number of translations between different formats, since data would only have to be translated ('uploaded') into the common format, from which it could then be translated again ('downloaded') into the formats of the receiving agents. More importantly, this means that the data has never been translated more than twice, significantly reducing the risk for data degradation as a consequence of multiple translations. Especially if the data is (made sure to be) of high quality in the standard format. Slide 21 Slide 22

A driving force behind the use of ontologies for data integration is the idea of having such standard formats. In order to introduce ontologies it is therefore Slide 23

useful to consider what such a standard should do, and what requirements it should fulfill:

Slide 24

1. It should be capable of representing *different kinds of information*. Imagine, for instance, the various kinds of data that is involved in the operation of offshore production platforms. There are product design and life cycle data for equipment and machine parts; construction data; logistical data; production data; sales data; geological data; etc. An oil company which wants a common standard for communication of such information internally and externally thus needs a standard which is *flexible* enough to incorporate all these different kinds of data. Of course, one could have different standards, but then one might have to translate between them, and the datasheet problem resurfaces. Ideally, therefore, the standard should be as *inclusive* as possible.

2. Different sets of data are required for different purposes and activities, but a particular piece of information may span many such purposes. It would therefore be beneficial if information could be represented in a *use-neutral* way.

3. The data in the common standard should be of *high quality*. Hence it is vital that information should be represented as accurately as possible. Moreover, implicit information that might be taken for granted within a certain subject area, but that one cannot expect the general user of the standard to know, should be made *explicit*.

4. Finally, the standard should be capable of handling massive amounts of information. This is already clear from what was said in points 1 and 3 above: a lot of different kinds of information need to be represented, in a fair amount of detail. A common standard for use in oil production will have to deal with thousands of concepts and terms (note that e.g. the Schlumberger Oilfield Glossary alone contains 4600 entries). Essentially, this means that it cannot be handled by humans alone: the standard should be constructed so as to enable a large degree of *IT support* for maintaining, updating, retrieving, and otherwise handling the information represented in it.

Points 1–3 mean that we need a system which is very *expressive*, in the sense that it must be a flexible system for representing varied and detailed information. Point 4 points to the fact that this system must have an underlying IT structure, so we can have *computer support* in handling vast amounts of information. In general, however, these ideals of *expressive power* and *computability* are very hard to combine. For instance, databases are a good way to represent large amounts of information in a way that can be efficiently handled by computers, but they are not very flexible. On the other hand, it is not all that hard to conceive of a standard that would be very flexible and expressive if you don't have to worry about machines being able to handle it. Our usual natural language could be one, if we fixed the meanings of relevant terms so they became clear and unambiguous. That is, one could write a standardized *dictionary* of the relevant terms, and then use ordinary language to write down information in those terms. This would allow us to represent information in an unambiguous way, but it would of course be completely unmanageable if you were trying to

Slide 25

represent large amounts of information (you would, essentially, have a giant collection of text documents).

However, the idea of having a standard dictionary would look more promising if one could specify a way to write down information in the standard terms in a way that machines could process. This is, indeed, one way to think of ontologies: they are a form of machine-processable or 'smart' dictionaries. To spell this out, we can describe an ontology (for a domain of interest, such as oil production) to roughly consist of the following: Slide 26

The Domain A specification of the domain, that is, of the collection of objects that the ontology is about. Slide 27

Classes First, a dictionary, giving a collection of classifying concepts such as "`red object`", "`car`", "`vehicle`", "`Emerson pressure transmitter`", etc., with definitions that specify what these mean. We'll use typewriter font to indicate that a word is in the dictionary. These concepts allow you to classify or describe objects in your domain. For instance, you can say that some thing is a `red object`, is a `car`, or is a `Emerson pressure transmitter`, and you have a unambiguous definition of what exactly Emerson pressure transmitters, say, are. The concepts in the dictionary can be thought of as one-place predicates, i.e. words that stand for properties of things, such as being red or being a vehicle . Or they can be thought of classes, or collections, of things. E.g. the collection of all the red things in the domain, or all the cars. It does not matter (at this stage) which way you think of it, the important point is that the concepts classify objects in the domain (in an explicit and unambiguous way). We refer to these as *classifying concepts*, or just *classes*. Slide 28

Subsumption hierarchy Next, one organizes the classifying concepts in a hierarchy according to how inclusive they are. For instance, suppose that everything that is a `car` is a `vehicle` (a natural supposition). We'll add this information explicitly to the ontology by adding an inclusion relation, denoted by $\subseteq$, say, which indicates that a concept is included in, or subsumed by, another in the sense that everything that is classified by the first concept is also classified by the second (if something is a `car`, then it is also a `vehicle`). Slide 29

$$\texttt{car} \subseteq \texttt{vehicle}$$

This organizes the concepts of the dictionary in an inclusion, or subsumption, hierarchy, which is easy to handle by computers. Enter something as a `car`, and you have automatically also entered it as a `vehicle`. Search for `vehicle`s, and the computer will give you all the `car`s, `bus`es, etc. that have been entered. We also refer to a system of classifying concepts ordered in a subsumption hierarchy as a *taxonomy*.

Relations The next level of structure to add to this hierarchy of concepts consists of relations between objects, such as `part of`, `weighs more than`, or `is owned by`. Using these, you can say that an object, which has been entered as a `wheel`, say, is a `part of` another object, which has perhaps been entered as a `car`. Or that an object which is a `truck` `is owned by` something which has been entered as a `company`. These expressions should also be given clear, unambiguous meanings, so they should be entered in Slide 30

the dictionary. The dictionary, then, contains two kinds of terms at this step: classes (i.e. classifying concepts), and relations.

Slide 31

Constraints and 'concrete' domains On top of the basic structure of relations and a hierarchy of classes, there are a number of other constructs that can be added while still keeping things manageable for computers. We'll only indicate a few here. One thing that is clearly useful is to be able to put constraints on what kind of objects can be related by certain relations. Trucks do not *own* wheels, for instance, so if you enter an object as a `wheel` and then try to enter that it `is owned by` something that is a `truck`, you would like the computer to return an error message (perhaps reminding you that only agents like people and businesses can be owners of objects). Another useful thing is to have abstract entities like numbers, and also weights, lengths, volumes, etc. These can be dealt with as additional, so-called 'concrete' domains[3], so that you have one domain of all the physical objects you are interested in (pumps, valves, trucks, and whatnot), one domain of weights, another of volumes, etc. You can then have relations that hold between (certain classes of) physical objects and these more abstract objects, such as a "`has weight`" relation that holds, for instance, between a `car` and the weight 1200kg.

Slide 32

Language Having classifying concepts, relations, and abstract domains makes it possible to say quite a bit about objects in your domain of interest. You can enter information about an object specifying what it is (if the relevant concept is in the dictionary), perhaps specifying its weight and volume, as well as certain relations (as long as they're in the dictionary) that it has to other objects, such as being a part of another object, or having another object as part. However, given a dictionary of concepts and relations, it will really boost the number of things that you can say about an object if you can form new concepts and relations from old ones. An ontology will therefore seek to provide a number of ways that you can combine concepts and relations in order to form new ones. Typically, this will at least include concept operators like "AND", "OR", and "NOT". Given, say, the concepts "`man`", "`woman`", and "`child`", you can then define the concept "person" as "`man` OR `woman`" and "adult" as "(`man` OR `woman`)

Slide 33

AND (NOT`child`)". One can also add operators that construct new relations out of old ones, or new concepts out of relations and constructors that correspond to the words "some" and "all", so that you can say things like "this object is a `wheel` AND it is `part of` SOME object which is a `car`", i.e. that it is the wheel of a car (which is not the same as being a car wheel). Without going into detail, an important point about such operators is that they behave according to explicit rules: if something is classified by "`child` AND `man`" then it is classified by both "`child`" and by "`man`"; if something is classified by "`man`", then it is classified by "`man` OR `woman`"; and if something is classified by "`child`" and by "adult", as defined above, then an error has been made. Such rules make it possible for computers to handle the complex concepts, and to e.g. inform users when they have entered information which is inconsistent with pre-

[3] "Concrete domain" is a somewhat confusing name for these, as they are really more like abstract domains.

vious data in the ontology. Thus the classifying concepts, relations, and concept/relation operators form a machine-processable language, in which information can be encoded. However, the extent to which it is machine-processable will vary with what kind of operators you add. Adding more will increase what you can express, but may seriously hamper what you can compute; specifically, it can hamper or destroy the ability to check that the ontology is consistent, or that some information can consistently be added to it. For this reason, the W3C-endorsed *Web Ontology Language* (OWL) comes in three versions, OWL-Lite, OWL 2 (OWL-DL++), and OWL-Full, where the first two are decidable[4] and the last is not.

To summarize, then, an ontology can be seen as a machine-processable language built from a set of concepts and relations which have been given explicit and unambiguous definitions[5]. The fact that it is, to some extent, a language gives it great flexibility in representing information. The fact that the language it kept to a level which computers can handle allows for computer support in checking for errors, for maintenance, and for answering queries and retrieving information. (But keep in mind that there is always a balance to be struck between flexibility and expressivity, on the one hand, and on computability, on the other.) More conceptually, an ontology can be seen as a hierarchy of classes of objects and relations between them, which makes it possible to encode information about objects by indicating which classes they belong to and what relations hold between them. The classification of objects, and the recording of what relations they enter into, can be seen as a rigorous (if simplistic) conceptualization of a domain, which mimics the very advanced conceptualization of that domain which we have in our own heads. The point is that objects are classified and related according to the properties and relations that they have in the world, or that we would normally regard them as having, independent of the need for particular information for different purposes or projects. And this serves to maximize the use-neutrality of the information that ontologies represent. Thus, to conclude with a comparison with the list of requirements for a standard for information or data exchange above:

Slide 34

Slide 35

1. As a form of formal languages, ontologies provide a *flexible and expressive* way to represent information.

2. To the extent that ontologies manage to describe things as they are, so to speak, they are *use-neutral*.

3. To the extent that they provide good definitions and a rich structure of classes and relations, they are suitable for recording detailed and *explicit* information and data of *high quality*.

4. Because they are a form of machine processable formal languages, they enable *computer support*, in particular for answering queries and for checking consistency.

[4]Roughly, and with some abuse of terminology, a formal language is *decidable* if you can compute whether a given statement is true or not according to previously recorded assumptions (such as inclusions between classes).

[5]Ideally, that is. The next module will contain exercises that will illustrate how frustratingly difficult, or even impossible, it is to give completely unambiguous definitions.

Questions, Comments, Coffee Break

1.3 The Industry Standard ISO 15926

Slide 37

1.3.1 What It Is

ISO 15925 is rather more, and at the same time a bit less, than an ontology as described in Section 1.2. The reason for this is that ISO 15926 has been designed to function as an all-encompassing data exchange and integration standard for businesses involved in oil and gas production and related, or similar, areas. As we touched upon in some of the examples in Section 1.2.1, this means that ISO 15926 must be able to cope with multiple kinds of information: there are inventories, construction plans, production data, and sales data; there is information that something is a certain kind of pump, that it is currently a part of some kind of machinery, perhaps that it was previously a part of something else; there is data concerning when and who produced the pump, when it was bought, when it was installed and who installed it, who certified it as a pump, and when this was done; and there is *meta-data* e.g. concerning who entered the said information about the pump and when, and perhaps who certified the

Slide 38

entering of the information. Now, even though ontologies are flexible, this is a tall order. The basic ideas of ontologies, as we saw, is to encode information about objects in a domain by classifying them according to a set of concepts or classes and by specifying relations that they enter into. This means that certain kinds of information are more suitable for encoding in an ontology-style framework than others. Meta-data, in particular, significantly complicates matters, as this concerns information concerning the ontology itself (e.g. not that something is a pump, but that the information that something is a pump was entered into the ontology at a particular time).

For this reason, ISO 15926 does not seek to provide an all-encompassing simple ontology of the kind that we outlined in Section 1.2, i.e. which consists of a single domain of objects equipped with a class hierarchy. Rather, ISO

Slide 39

15926 provides, at its core, a complex system of classes and relations, which one can think of as an interconnected system of several domains with several class hierarchies (some of these are so-called 'concrete' domains, i.e. domains of numbers, volumes, etc.). This complex system hampers computability, but provides ISO 15926 with unparalleled flexibility and expressivity. IT support is, instead, introduced by constructing simpler ontologies on the basis of (parts of) the core system, and by constructing software which is designed to help in the entering and retrieving of information encoded in terms of the complex class structure. As a consequence, ISO 15926 consists of several parts (these parts have numbers, but as ISO 15926 is being developed these numbers, and the parts themselves, are subject to change), the most important of which are the

Slide 40

following:

- A core part (often referred to as *Part 2*), consists of the so-called *Data Model*, which specifies the structure of the class and relationship system of ISO 15926.
- Built on the Data Model, there is a part (known as *Part 4*) which contains so-called *reference data*. Reference data are specific classes and relations (such as `Pump` and `Well head`) which are entered in accordance with the

class and relationship structure specified in the Data Model and equipped with explicit definitions. (One can think of the resulting *Reference Data Library* (RDL) as the dictionary part of ISO 15926.)

- A third part contains OWL-based ontologies encoding geometrical and topological data (embodying the concepts defined by ISO 10303-42 and ISO 10303-104, including concepts in Earth models and the GIS standards ISO 19107 and ISO 1911).

- Finally, the so-called *Part 7* contains implementation methods for the integration of distributed systems. This includes so-called *templates*, which allow end users, which are unfamiliar with the complex structure of ISO 15926, to enter and retrieve information according to the standard. Templates provide users with simple schemas or tables to fill in, and automatically translate and record that information in terms of the class and relationship structure of ISO 15926. Templates also work the other way, that is, by retrieving information recorded in terms of the class structure and presenting it in easily readable tables. Slide 41

Compared with the description of an ontology in Section 1.2, then, ISO 15926 has a large standard dictionary, organized in a very complex class and relationship structure, including the 'constraints' and 'concrete domains' of Section 1.2. It does not have class constructors (although they can, in part, be simulated using the complex class structure) or a machine processable language. One could, of course, add a language, but due to the complexity of the class structure, it would not be machine processable to a satisfactory degree anyway. Instead, additional parts are being developed which contain IT support for the use of its class and relationship structure. This will enable ISO 15926 to be a, potentially, very powerful and flexible, yet usable and practical, standard for data integration. It purpose-built for representing a variety of different kinds of information, Slide 42
including meta-data, as well as life-cycle data; that is, information concerning the same object over the span of its life.

1.3.2 How To Use It

Sections 1.1–1.2 outlined the datasheet problem, and how ontologies and semantic technology can be used as a common standard to address that problem introduced ISO 15926, a semantic technology based standard and tool kit for data integration in the oil and gas industry well as in industries with similar needs. We conclude this presentation by summarizing the fundamental structure of ISO 15926 and outlining how it can be used to help businesses with their daily data integration and communication needs: Slide 43

Nomenclature Standardized nomenclatures are very useful and exist in abundance in the industry. A large number of these are recorded in ISO 15926. Of course, a nomenclature is only useful insofar as you know what the terms in it actually mean, which brings us to the next point.

Dictionary ISO 15925's Reference Data Library (RDL) is an accessible database with a large and growing number (40 000+) of concepts and concept definitions, including definitions explaining the meanings of terms in the nomenclatures encoded in ISO 15926. Independent from the rest of ISO 15926,

the RDL can be used by businesses as a joint standard in order to minimize confusion and uncertainty regarding the meaning of nomenclatures and central concepts in general.

Taxonomy The classes in the RDL are ordered in a subsumption hierarchy, which in itself encodes additional information in a manner which is accessible and useful for the IT support which ISO 15926 builds around the RDL.

Taxonomy with Relations The relations contained in the RDL makes it possible to represent project data information concerning e.g. ownership, constituent parts, etc.

Complex Class and Relationship Structure The rich class and relationship structure of ISO 15926, which goes well beyond just a taxonomy with relations, makes it possible to represent life cycle data, complex information such as operating ambient temperature specifications, and also makes it possible to record meta-data. This, then, is what elevates ISO 15926 from a standardized dictionary or a simple taxonomy to a standard for data exchange which can meet the needs of the process industry.

Templates and Project Data As a standard for data exchange, agents must be capable of recording their project data in terms of the complex class and relation structure of ISO 15926, and also be able to retrieve data encoded in terms of this structure. ISO 15926 contains machinery (the so-called templates) which allow users to to do this 'uploading' and 'downloading' of data without having to learn or master the class and relationship structure.

This concludes our presentation of semantic technology and ISO 15926. We end
Slide 44 with a new look at slide 15 from Section 1.1, which has been modified to show the place of ISO 15926 in the data exchange of the agents involved.

Chapter 2

Introduction to Modeling in ISO 15926: the Data Model

Slides 1–2 Course title and contents.

Slide 3
- Module 1 introduced you to the idea that an ontology is built on a system of classes and relations. It was also said that this structure is divided into two parts in ISO 15926 (we'll drop the "ISO" in the future and just say "15926"): the data model or Part 2 (15926-2); and the Reference Data Library (RDL) or Part 4 (15926-4).
- This module introduces the data model, which we will refer to as Part 2.

Slide 4
- Part 2 can be bought from ISO as a pdf file[1].
- An easily navigable overview of the Part 2 classes can be found at `http://ht.vestforsk.no/demo/iso15926/`.
- The EXPRESS code and the EXPRESS diagrams of Part 2 can be found at `http://www.tc184-sc4.org/wg3ndocs/wg3n1328`
- We recommend that you do obtain the pdf file if you plan to work with 15926. This course can not go through all the details of the data model, but will introduce its general principles and most important concepts, and prepare you for reading the remaining details by yourself.

2.1 Fundamental Concepts

Slide 5 Section overview.

Slides 6–8
- The two most fundamental notions for understanding 15926-2/4 are those of class and relationship.
- A class is a collection of objects. The objects in the class are called its members, and we say that an object is (or is not) a member of a class.

[1] `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29557`

A class is an abstract collection of (abstract or concrete) objects; it does not matter where the different objects are, for instance. In that respect a class of pebbles is different from a pile of pebbles. The pile can be destroyed by throwing the pebbles around, but the class of the same pebbles will remain the same.

- Classes are individuated by their members: if class **A** has exactly the same members as class **B**, then **A** and **B** are the same class, and **A**=**B**. Sometimes, this fact or convention is expressed by saying that classes are *extensional*.
- Classes are themselves objects, and so they can be members of other classes. This is important in 15926.
- By a classifying concept, we mean a concept which holds, or is true of, single objects. I.e. "is a car". Some things are cars, others are not. The collection of objects that are cars forms the class of cars. Classifying concepts differ from classes in not being extensional but intensional, in the sense that different concepts can be true of exactly the same things. E.g. "is a pig and can fly" is true of exactly the same objects as "is a man and can walk on water", namely none. By the extensionality principle, there is only one class that has no members (it is called the empty class).
- Does 15926 use classes or classifying concepts? This question is best left open. The Part 2 pdf explicitly states that classes are extensional, but tends to treat them as if they are intensional (e.g. by distinguishing between classes that, it seems, have the same members). For our purposes, nothing depends on whether they are extensional or not. So we call them classes, but bear in mind that they are perhaps better thought of as classifying concepts. Also, we say that an object is a member in the class **Pump** and that it is a **Pump** interchangeably (bold face indicates a 15926 class/classifying concept).

Slides 9–10 Two important notions when it comes to classes are membership, which we have already mentioned, and inclusion. We say that a class **A** in included in a class **B** if all members of **A** are also members of **B**, i.e. if everything that is an **A** is also a **B**.

Slide 11–12

- A relationship is something which holds of, or between, two objects. Examples include is father of, is part of, is resident of, and is taller than.
- In 15926, "relationship" means a particular relationship between two particular objects, e.g. the relationship that holds between me and the state of Norway by virtue of me being a resident of Norway. But "is resident of" is also an expression or concept which holds of many pairs of objects, such as me and Norway, George Bush and the USA, and you and some country. The latter kind of concept is called a "class of relationship" in 15926, and is thought of as the class (or classifying concept) of particular relationships.
- Accordingly, particular relationships (just "relationships" from now on) are thought of as objects (that can be members in classes). A relationship object is given by giving the two objects that are related

and so-called roles which, together with its membership in a certain class of relationship, serve to distinguish the relationship from other relationships that my hold between the same two objects. We'll get back to the details of this. For now, the important point is to distinguish between a (specific) relationship and a class of relationship.

Slide 13 In 15926, very general classes (such as **Physical object**) are given in Part 2, while more specific classes (such as **Pump**) are found in the Reference Data Library (Part 4) . The latter are referred to as "reference data". Even more specific, and outside of 15926, is the so-called "project data", e.g. that a certain individual is a **Pump**.

Slide 14

- The classes that are explicitly given in Part 2 are often referred to as "entity types". This has to do with their status in the Express language that is used for 15926. In the next section we explain the most important features of this language, and of Express diagrams.
- The distinction between the Part 2 classes and Part 4 classes is, however, an important one when you are working with 15926. Although, conceptually, both should be thought of as *classes*, in practice you also need to think of the Part 2 (Data Model) classes as entity *types*, and the Part 4 (Reference Data) classes as *entities*. What this means is that Reference Data are always considered to be, and are entered as, *objects*, whether they are classes or not. They then have to be classified according to the Part 2 classes, i.e. they have to be given a type. If your reference datum is a class (e.g. **Pump**) then it should have the entity type **Class** or one of its subtypes. In this context, you think of Part 4 classes (and other things, such as relations) as objects, and Part 2 classes/entity types as classes. Then, as you go on to record subsumption or membership relations involving your reference data classes, you think of them as classes again.

2.2 EXPRESS

Slide 15 Tutorial overview.

Slide 16

- EXPRESS is a data modeling language designed for specifying data structures in a machine readable format. It is It was originally written for the STEP standard (ISO 10303), the predesessor of ISO 15926.
- Because EXPRESS is a data modelling language, and not an ontology language, it does not recognize concepts such as Class and Relationship. Its basic component, the entity type, is not the kind of type you find in logic, but the kind you find in programming languages. Very superficially, you could say that EXPRESS is to 15926 as XML is to OWL.
- Only the parts of EXPRESS used in the 15926-2 specification will be explained. If you want to learn more about this language, more

information can be found at Because only a small part of EXPRESS is used in 15926-2, we don't need to explain the whole language[2].

Slide 17 An EXPRESS model, is composed of a set of entity types. An entity type is a specification of a type of data, and much like the tables in a database. For instance: If you want to store information about persons, you may want to store the ages and names of these persons. The age could be stored as an integer and the name as a string. In EXPRESS notation, this is written:

```
ENTITY      person
            name :  STRING;
            age :   INTEGER;
END_ENTITY;
```

Here, we see that a person has a name and an age. The name and age are attributes of the entity type.

Slide 18–19

- The "name" and "age" entries are called "attributes". Each attribute has a type, here they are STRING and INTEGER.
- Built-in types: These are fundamental data-types: string, boolean, integer, double, number, logical.
- Aggregate types: EXPRESS allows you to have lists of types as a type in itself. 15926 use only the LIST construct. Every element in a list must be of the same type.
- Entity types. By far the most important data type is the entity type. Each entity type in the data model may be the type of an attribute. This is used extensively in 15926, and is one of two ways of relating entities to each other.
- Ex: A person, with an optional mother

```
ENTITY Person
                age :     INTEGER;
                name :    STRING;
                mother :  OPTIONAL Woman;
END_ENTITY;
```

Slide 20 Entity types in a data model are usually organized in a type hierarchy. (Pic) Person → Man, Woman A subtype inherits all the attributes of its supertype. Informally we can say that a Man is a Person, i.e. that if an application requires data in the form of the data type Man, a Person will be applicable. (Due to the transitivity of the subtype hierarchy, it is possible for an entity to be a subtype of several entities. More uncommon, however, is that EXPRESS allows entities to be *direct* subtypes of many different entities.)

Slide 21–23 It is possible to restrict the hierarchy:

- Entities declared with the ABSTRACT keyword, may only be instantiated if a non-abstract sub-entity of this entity type is also instantiated. (Pic)

[2]More information about EXPRESS can be found at
`http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047`

- If two or more entity types are given within a ONE OF construct, these entities are disjoint. (Pic)
- If two or more entity types are separated by an ANDOR keyword, then an entity instance may instantiate all these entity types. (Pic)

2.3 The Basic Classes of 15926

2.3.1 Thing

Slide 24 Tutorial overview.

Slide 25–26
- The top class in 15926 (part 2) is **Thing**. All 15926 classes and all their members are **Things**. All 15926 classes are also subclasses of **Thing** (since all their members are **Things**.
- The class **Thing** is split into two subclasses: **Possible individual** and **Abstract object**. These two classes are disjoint, and together they include all **Things**.
- The class **Possible individual** has as members all physical things, and more generally, all things that could exist in space and time. There are two tings to note here:
- Possible and not just actual objects are included. This is to allow for the representation of e.g. planned objects in 15926. However, this part of the data model should be considered work in progress. Good practices for representing possible as opposed to actual objects have not yet been established, and we are mostly concerned with actual objects.
- **Possible individuals** are considered as four dimensional space time regions. This is convenient for many purposes, but it is also somewhat un-intuitive.

2.3.2 Possible Individual

Slide 27 Normal physical objects in 15926 are considered as essentially extended in time. That is, a pressure transmitter is considered as one continuous object through time from its construction to its destruction. It has temporal parts, just as it has spatial parts, one of which is the temporal part of it currently existing. Both the whole pressure transmitter and all of its temporal parts are **Possible individuals** (as are, of course, all its three dimensional parts, such as its casing). More generally, any space-time region is a **Possible individual**. Seeing objects as space-time regions is meant to be a powerful tool for the representation of life-cycle data.

Slide 28 A conceptual illustration of the space-time extension of a particular object, called #1234.

Slide 29 A pressure transmitter. It is considered as one continuous 4-dimensional object throughout its existence. It is a member of the class **Gauge pressure transmitter**.

Slide 30 (Self-explanatory) Note that just as the distinction between possible and actual objects, the 4-dimensionalism of 15926 is something that can be complicated to use in practice, and that can often be disregarded for the creation of simple reference data. But it is important to be aware of.

2.3.3 Abstract object: Class

Slide 31 **Abstract objects** are things such as numbers and classes that do not have space-time extensions.

Slide 32 The subtypes of **Abstract object** are **Class**, **Relationship**, and **Multidimensional object**. We'll focus on the first two. **Multidimensional object** consists of ordered n-tuples (lists) of objects .

Slide 33 **Class** is the entity type of Part 4 classes.

Slide 34 Notice the last example: **SAE 20-40W Motor engine oil**. What is the instances/elements of this class? The four-dimensionalist view is convenient for treating liquids and natural kinds. Any region which consists of SAE 20-40W Motor engine oil is a **Possible individual** which is an instance of this class.

Slide 35 Illustration.

Slides 36 Self-explanatory

Slide 37 Diagram notation. Notice that the membership relation is given a designated diagram symbol.

Slide 38 Example diagram. **Class** and **Thing** are Part 2 (Data Model) classes (and entity types); **Pump** is a Part 4 (Reference data) class, and #1234 is a particular pump, i.e. project data.

Exercise 1

Slide 40 Self-explanatory.

Slide 41–42 The four-dimensionalist perspective allows for classes with fixed membership. Otherwise, a red car would change class from **Red** to **Blue** if it were to be painted blue. Instead, one temporal part of the car is classified as **Red**, and another as **Blue**.

2.3.4 Abstract Object: Relationship

Slide 43–44 A **Relationship** is an abstract object. It could be thought of as an ordered pair of objects, were it not for the fact that the same two objects can enter into multiple relationships. Relationships are given names (as are all **Things**) to distinguish them.

Slide 45

- A **Relationship** relates two objects, which occur as attributes of the **Relationship**. That is to say, the subtypes of **Relationship** have appropriately named (Express) attributes, which are the objects that the **Relationship** relates.

- **Relationship** has several subtypes. We'll focus on **Classification** and **Specialization**, which have a special status at least in terms of diagrammatic representation, and on **Composition**, which will serve as an example of **Relationships** more generally.

Slide 46–47
- The **Classification** and **Specialization** relationships are really our familiar relationships of membership and inclusion, so why are they appearing with new names? The point here is the distinction between the entity types of part 2 and the reference data of part 4. Part 2 specifies a number of classes as entity types, which sets them apart from the classes entered in Part 4 as reference data.
- Anything entered as either reference or project data must be entered as a member of at least one of the entity type classes of Part 2, and that is done by entering it as having some particular entity type (as we shall see when we get to the RDL Editor later).
- A reference data class, e.g. **Pump** or **Mechanical device**, is a piece of reference data, which you should think of as an object (or **Thing**) with a certain entity type (**Class of arranged individual**, perhaps). But it is also a class, and so may have members or enter into inclusion relationships. So how do you indicate that two such reference data objects are related by inclusion, or that one is a member of the other? The answer is that you record the required relationship as a new piece of reference data, i.e. as a **Relationship**. The entity type **Classification** contains the membership **Relationships**, and the entity type **Specialization** contains the inclusion **Relationships**.

Slide 48–49 A **Classification** is a **Relationship** the first attribute of which, called the "classified", is a **Thing**, and the second attribute of which, called the "classifier" is a **Class**. A **Classification** between a **Thing** and a **Class** says that the classified **Thing** is a member of the classifying **Class**.

Slide 50 Illustration.

Slide 51 Example diagram. Note that"an object" can be either reference or project data. Notice, also, that arrows are used to indicate membership in these diagrams, both as regards membership in entity type classes and in reference data classes.

Slide 52–53
- **Specialization** is a relationship which holds between two classes (the attributes), the first of which is called the "subclass" and the other of which is called the "superclass". **Specialization** is thus the entity type of inclusion relationships.
- Illustration: a superclass including a subclass, and a member of them both.

Slide 54 An example. Notice how inclusion relationships, both in the form of the EXPRESS subtype relationship and the Specialization Relationship, are represented.

Exercise 2

Slide 56–57
- Transitivity is a property of some class of relationships. For instance, "is ancestor of" is transitive: if Adam is an ancestor of Abraham, and Abraham is an ancestor of Isak, then Adam is an ancestor of Isak. On the other hand, "is father of" is not transitive: Haakon VII is the father of Olav V, and Olav V is the father of Harald V, but Haakon VII is not the father of Harald V.
- **Specialization** is transitive (as a class of relationship), but **Classification** is not.

Slide 58
- **Composition** is a part-whole relationship, i.e. a **Relationship** which says that something is a part of something else.
- As such, there is a "part" attribute and a "whole" attribute. Both attributes should be **Possible individuals**.

Slide 59 Recall our four-dimensionalist representation of **Possible individuals**. A thing, a, can be recorded as a part of another thing, b, if the space-time extension of a is included in the space-time extension of b.

Slide 60 This diagram shows how to diagrammatically represent relationships in general, and the **Composition** relationships in particular: the **Relationship** itself is drawn as a diamond with two edges, representing the so-called roles, drawn between the relationship and its attributes. For a **Composition** relationship, the roles are called "part" and "whole". For a **Classification** relationship, they are called "classified" and "classifier". (Although for a Classification relationship, we more often used the shorthand notation of the arrow in diagrams. Still, when entering a Classification relationship in the RDL using the RDL Editor, you have to fill out these role names.)

Slide 61 Diagrammatic notation.

Exercise 3

2.4 Types of Reference Data Classes

Slide Tutorial overview.

Slide 64 The **Class** entity type class, and its subtype subclasses.

Slide 65–67
- **Class of class** is a subtype of **Class**. A class of class is a class all the members of which are also classes (remember that classes are objects, and so they can be members of other classes).
- One use that 15926 makes of classes of classes is exemplified by the class **Color**. First, a particular color like red is represented by a class **Red**, which is a **Class of individual** consisting of all possible individuals that are red. Then all those classes, **Red**, **Blue**, **Teal**, etc are collected into a class called **Color**.
- Another useful way to use classes of classes are for bookkeeping: suppose you are recording a number of different classes corresponding, say, to a set of concepts used for environmental reporting. To keep

track of these classes you can construct a new class, say **Environmental report class** which contains all those new classes. Anyone wondering "what are the classes used for environmental reporting" can then just look through the list of members of the class of class **Environmental report class** to find the answer.

Slide 68–69

- Classes of relationships are a particularly important class of classes. Examples of classes of relationships are the class **Ancestor of** and **Father of**, to use earlier examples.
- We'll use the subtype **Class of composition of individual** as an example to show how classes of relations are represented. This type has a "class of part" and a "class of whole" attribute.
- Recall that any member of **Composition of individual** comes with a "part" attribute and a "whole" attribute. Accordingly, if we have a **Class of composition of individual**, that is, a class X consisting of **Composition of individual** relationships, then we can form two classes: the class of all the parts occurring in X; and the class of all the wholes occurring in X. Those two classes are the attributes of X, as shown on this slide.

Slide 70 Example. The class **Impellers of model 106 pumps** contains the part-whole relationships between model 106 pumps and their impellers. Since all model 106 model pumps have an impeller as part, **Pump model 106** is the class of whole attribute of **Impellers of model 106 pumps**. If only some pumps had impellers, then the class of whole attribute would be the subclass consisting of those pumps that have impellers.

Exercise 5

Slide 72 The subtypes of Class of individual. We do not go through all of these in detail, see the 15926-2 pdf or the star tree from Vestlandsforskning[3] for the definitions.

Slide 73–75 Finding the appropriate Entity type for your reference data can be a problem, which we have to leave for a more detailed introduction to reference data construction. These slides provide some illustration, displaying the many subtypes of **Class of arranged individual** and the principle of increasing levels of aggregation by which they are organized.

Slide 76–78

- The entity type **Property** - a subtype of **Class of individual** - is used to represent such things as specific volumes and temperatures. Consider a volume, such as 1 litre. Several different terms refer to this volume, for instance "1 litre", "10 decilitres", and "0.001 cubic meters". The volume itself - the volume that these expressions refer to - is represented as an object (a **Thing**) with entity type **Property**. Specifically, it is represented as the class of all **Possible individuals** which has volume 1 liter. But the important thing is to notice that the volume itself, independent of which unit of measure is used to express it, is an object.

[3] `http://ht.vestforsk.no/demo/iso15926/`

- Because of the representation of such things as volumes and temperatures as the classes of all **Possible individuals** that have them, it is easy enough to say that a certain object e.g. has the temperature 21C: one simply records it as a member of the class which is the temperature 21C.
- The classes **Temperature** and **Volume** (etc) consist of all temperatures and volumes, respectively, and have entity type **Property space**.

Slide 79–81 Using **Property quantification**, a subtype of **Relationship**, you can record that a volume X, say, is 1 with respect to the unit of measure liter. I.e. that X is 1 liter. Enter a relationship between X and 1, and classify it as a member of the appropriate **Class of property quantification**—called a **scale**— that says that these relations connect volumes to their quantity with respect to the unit of measure *liter*.

Slide 82–84 Finally, we mention the entity type **Indirect property**. Suppose you want to express in 15926 that, according to an estimate of some kind, the load of grain # 1234 has mass 5kg. You cannot classify # 1234 as a member of the mass 5kg (let's just call that class **5kg**), for that would express that the mass of # 1234 is 5kg, and that is not what you want to say. In this, and similar, situations, you use the entity type Indirect property. An **Indirect property** is a relation between a **Possible individual** and a **Property**, in this case between # 1234 and **5kg**. By creating an appropriate **Class of indirect property**, for instance one called **Estimated according to method X** with a definition the method X used for arriving at the estimate, and then classifying the **Indirect property** between # 1234 and 5kg as a member of **Estimated according to method X**, you have then expressed that according to an estimate using method X, the load of grain # 1234 has mass 5kg.

2.5 Templates

Slide 85 Tutorial overview

Slide 86–87 Entering project data in 15926 is a complicated matter. Consider the statement that a certain pressure transmitter # 1234 has an ambient operating temperature range between -40C and 80C. It is possible to record an object # 1234, classify it as the appropriate type of pressure transmitter, and enter relationships that record this fact, but it is not simple. This slide shows you the relevant relationship structure.

Slide 88
- In order to make it end user friendly to enter information as reference or project data in 15926, a number of predefined 'statement types' are being developed. These are called *templates*. A template can be thought of as a statement with place holders, such as "Thing X has an ambient operating temperature between Y and Z degrees Celsius". The end user can then enter a name for X and numbers for Y and Z, and the resulting statement will be entered automatically into 15926 format. Information can also be automatically retrieved from 15926 and presented in template form.

- For the end user, interacting with 15926 will then be a question of filling in and reading from a predefined set of tables (the templates). The main challenge for end users are, therefore, to be able to understand and translate the information presented on their own documents and datasheets in order to correctly fill the template out. The template may ask for explicit information that is only implicitly present in the documentation of the end user.

Chapter 3

Introduction to Modeling in ISO 15926: Reference Data