drmCrop2 Guideline

The domain reference model for crop production, version 2

Document type:	Draft
Date:	August 26, 2015.
Background:	This document is an explication of the domain reference model drmCrop2.
Editor:	Dr. Ir. D. Goense, daan@pragmaas.com
Sponsors:	Het Programma Precisielandbouw (**PPL**)
	PPS AgroConnect.
	EU research project **FIspace**.
	Farm Digital

Table of Contents

1 Introduction.

1.1 Version 2

Version 1 was developed from the perspective of arable farming with emphasis on precision agriculture. Version 2 is developed in cooperation with stakeholders from the horticulture and flower production chains where focus is on tracking and tracing up to the retail market. The result is that a number of classes are added to cover the additional requirements and that some classes have name changes to become acceptable in a wider context. Also some definitions are sharpened or changed.

1.2 Developments in reference models for crop production.

During the years several agricultural reference models are developed as basis for standardised information exchange. For crop production these include:

- Informatie Model Open Teelten", IMOT
- EDI-Teelt 1-3
- Computer Integrated farming, CIA
- ISO11783-part10
- AgroXML
- EDAPLOS
- UN-CEFACT
- EDI-Teelt plus
- EDI-Teelt++

An important bases for many of the listed models is the "**Informatie Model Open Teelten", IMOT**, (1987) which is the first reference model developed for arable farming in the Netherlands.

EDI-Teelt version 1 to 3 is based on the IMOT model, though the entities and its mutual relations of IMOT were neglected and only a flat model with attributes from IMOT was left. In the version 3, some of the entities with their hierarchy were reintroduced.

Computer Integrated Farming, CIA. This model had IMOT as a basis, but the classical relational entity model was converted to an object oriented model, with extensions for precision Agriculture. It also took animal husbandry into account, resulting in some abstract classes covering both branches of agriculture.

ISO11783 Part 10. Part 10 of ISO11783 started as a subset of the CIA model, but developed itself under WG1 of ISO/TC23/SC19.

AgroXML is a German reference model which clearly shows characteristics from IMOT and ISO11783.

EDAPLOS is a French reference model, which has clear signs of inheritance of IMOT, but also some very pragmatic solutions based on messages between parties in France that were used at the moment of development.

UN-CEFACT. EDAPLOS was the start of the arable farming part for core components in UN-CEFACT.

EDI-Teelt plus was an initiative in the Netherlands under the umbrella of the GEO-Boer project. It is characterised by very abstract classes. It also introduced the geometry classes required for precision agriculture. This model was the basis for data exchange on agricultural fields between farm management systems and the Dutch government.

The reference model **EDI-Teelt++** was an initiative from the Program Precision Agriculture (PPL) of the Dutch Ministry of Economic Affairs, Agriculture and Innovation (EL&I, now EZ). PPL promoted the development and introduction of Precision Agriculture in the Netherlands. EDI_Teelt++ is intended as a reference model for all data that is handled in Precision Agricultural applications. Data is defined and the structure of the data is made visible in UML class diagrams. EDI-Teelt++ was clearly focussed on the use of XML messages between farm management systems and services.

1.3 Why drmCrop

When looking at the mentioned models, there is a large communality in the classes and attributes of the different classes. This is not surprising as all applications in crop production that are computerised require sooner or later the same information to perform properly. The main differences in the above mentioned models is in the way the information is structured in classes and the applied abstraction level. The different reference models also reflect the requirements of the time they were developed. During development of IMOT it was expected that everything would soon be computerised, so the model took all available knowledge of agricultural research into account. Much attention was paid to strategic, tactical and operational planning, based on models developed in academia. This proved to be too optimistic, and later models restricted themselves in the number of applications that were covered. At this moment (2013) we see, apart from the introduction of geographical aspects, applications in development which require again more types of data.

Another characteristic is that the reference models reflect their time of origin and the main technology used at that time. IMOT used entity relationship diagrams and was focussed on relational data bases as means of data storage. CIA introduced object orientation while ISO11783-part10 had an ERD approach. Mentioned models used initially a proprietary exchange format (ADIS), which was soon shifted to the use of XML. AgroXML had, as the name indicates, from the beginning onward XML in mind. The technology which is used in the different models is reflected in some attributes and the way in which relations between classes are modelled.

Now newer technologies for data communication like JSON and RDF come along, it is realised that a reference model should restrict itself to a domain model which is implementation and platform independent. That is the reason why drmCrop was started as a further development of EDI_Teelt++. The latter had clear characteristics of XML. It used data types which were based on XML; ID and sometimes IDREFS were modelled. It held also some characteristics of relational databases by the use of Id's and modelling junction tables for many to many relations. It is the intention to remove or change all those implementation specific characteristics from the domain reference model.

drmCrop is the basis for standardized messages that will be used in data exchange between applications and between parties in the Netherlands. Messages will now, 2013, be based on XML, but drmCrop is also intended as the reference for other methods of data exchange like JSON and RDF. On the longer term it should also be possible to generate messages for band width efficient protocols like that used on the CAN bus by ISO11783, or on message hopping wireless sensor networks from the reference model. The data elements that are relevant for specific messages are made visible in message specific diagram(s).

1.4 Objective

The objective is to provide a reference model for crop production which can be used as a basis for different means of standardised data exchange in agriculture.

The reference model reflects the classes of objects which are relevant in crop production, gives clear definitions of those classes, specifies the relevant attributes and shows the mutual relations between classes.

Different methods of data exchange will be specified by a well-defined transformation from the reference model to the implementation/platform specific model. This will take care that data types and attributes that are specific for the means of communication will be inserted by the transformation. An example is an attribute defined as Real in the reference model will be converted to gml:decimal in an XSD model. For different purposes, also different subsets of the reference model can be used. Though, the name and definition of transposed classes and attributes will not be changed, and for their meaning the domain reference model is the only source.

It is the intention to use as much as possible already existing standards. For geometries the classes will be used as defined by the OGC consortium (ISO19107) and the transformation of those classes to XML is

based on GML[1]. Data types are in the reference model limited to the basic, implementation independent data types like Boolean, Integer, Real and String. Some data types for attributes might require a further specification by their nature. Examples are URI's or designators with restrictions in allowed characters. In that case as much as possible XSD types are used. (Examples are anyURI or token).

2 Materials and Methods.

UML is used as the modelling language, Enterprise Architect (EA) is used as the tool to document the domain reference model. EA is also used to transform the domain model in platform specific models.

3 Structure of the rmCrop Model.

The Enterprise Architect model rmCrop.eap is divided in a number of packages and sub packages.

- BPMN 2.0 Business Process View (e)
- Use case model (e)
- Domain model
  - drmCrop
  - drmCropNL
  - FIspaceShortTermSolutions (e)
- Dynamic view (e)
- External models (e)
  - DataDictionaryPlantProtectionProducts
  - Fertilizers
  - ISO 00639 Language Codes
  - ISO 19107 Spatial schema
  - ISO 19111 ReferencingByCoordinates
  - ISO 19123 Coverage Geometry and functions.
  - ISO 19130 Sensor Data
  - AgroXML
- External XSDs (e)
  - GML
- Java model Crop, (e)
- Mapping drmCrop (e)
- MySQL model Crop (e)
- WSDL Phytophthora Control (e)
- XSD model crop. (e)
- Deployment Model (e)

(e) only available in the extended version of the model. This is available on request (e-mail: daan.goense@wur.nl)

3.1 General.

rmCrop is a model in development. It does not claim to be complete and not all classes, attributes and relations are checked yet in respect of modelling guidelines defined hereafter.

Classes which are relevant for sending a complete cropping scheme from a farm to other parties are, as far as foreseen yet, complete. This is also the case for classes which are used in a fertilizer advise coming form an advisory service to a farmer. These classes can also be used for a crop protection advise.

All other classes originate from historical models and mapping against other models or reflect requirements from projects in which drmCrop is used as a reference. A number of choices which are made must be seen as proposal, open for discussion.

A number of items still to be completed are:

[1] This proved to be difficult, as GML is not a one-to-one representation of ISO19107. See for example a SurfaceBoundary as element of a Polygon in ISO19107, which is not used in gml:Polygon.

- Definitions of attributes are not complete.
- Minimum and maximum values for those attributes which represent a value are not specified yet.
- A check on national classes or attributes that should belong in a national sub model like drmCrop_NL is not complete, so there might still be attributes which are typical for the Netherlands.
- Mapping on ISO11783 is not complete
- Mapping on AgroXML is not complete
- For classes with geometry aspects the most suited OGC standards must be selected. This is done for Point, Linestring, Polygon, MultiPoint and MultiSurface, but Grid has to worked out in more detail.
- The patterns for units and measures (Rate, Value, etc.) can be improved by (partly) adopting the QUDT ontology of NASA.
- The templates to generated XML, Java and DDL models need changes and the generated models have to be actualized. At this moment still manual changes in the transformed models must be made.
- Etc.

3.2 BPMN 2.0 Business Process View.

This package holds the description of some business processes which are realised in the FIspace trial "Crop Protection Information Sharing". This trial implements a system for Phytophthora control in potatoes.

There is some redundancy in the components.

A check has to be done on the correct message / data flows.

The "Total View Crop Protection" diagram is rather complete.

3.3 Use case model

The use case model shows some use cases as defined in the ISO/TC23/SC19/WG5 working group for wireless communication in Agriculture. It describes some aspects of fleet management, task update and job control.

3.4 Domain model

3.4.1 drmCrop

This is the core of the model. drmCrop describes classes of objects and their attributes which are relevant for crop production.

All identified classes for crop production are listed in the drmCrop package. Also a number of diagrams are specified. Each diagram has a name which indicates a particular aspect of crop production and shows the classes which are relevant for that aspect. Several aspects have a further explanation in the Appendices with examples.

The drmCrop domain model has four sub packages.

DataTypes, defines data types which are specific for drmCrop. (See "DataTypes" under "Some Modeling and naming conventions used")

Enumerations specifies enumeration tables that are specific for drmCrop. (See "Enumerations versus Code Listst" under "Some Modeling and naming conventions used")

Geometries. Specified geometries are based on ISO19107 and GML. As the ISO 19107 specification includes methods as part of its class specifications, this model should be seen as a platform specific implementation model, which in analogy to the approach in rmCrop should result from a transformation from a platform independent domain model. GML is the XML representation of ISO19107 and as such also a platform specific model. This package Geometries is a reverse engineering from both ISO19107

and GML and should be able to act as platform independent model. There is no transformation template specified for the Geometry package, as the results are already well published by ISO19107 and as GML schema's. The difficulty is that GML is not a one to one transformation from ISO19107. (see footnote 1 on one of the previous pages.)

XSD_Types. Here some data types from the XML data types as published by W3C are listed, which are used as data type in drmCrop.

3.4.2 DrmCropNL

There are some attributes of classes which are specific for the Netherlands. Most have to do with national legislation. It is assumed that they are not used in other countries. Those specific national attributes, or eventually classes are specified as subclass of classes in the drmCrop package.

The same procedure can be used for other countries. A clear example is the "Ackerzahl" for Fields in Germany (and some other countries in Europe?), which should be specified in a drmCrop_DE package.

3.4.3 FIspaceShortTermSolutions.

A short term solution is chosen in the FIspace project to represent some essential growth stages of a CropField. Emergence date and the date of start of tuber formation in potatoes are made attributes. A correct solution is a reference to a growth stage coding table, but this complicates the data structure and processing considerable.

3.5 Dynamic view

A start is made to model the dynamics of some classes which can change their state based on events.

In this package there is also **The Phytophthora Control Sequence Diagram**, which shows the messages exchanged so far in the FIspace trial for Crop Protection Information Sharing.

3.6 External models

External models are specifications defined as class models. Some specifications are specified as XSD schema's. See 3.7.

The whole model contains two types of external models:

1. External reference models which are only used as a reference, something to look at or to provide ideas. These can be models describing the same domain (crop production) or related domains. There is an external model for Fertilizers and Crop protection material. (See also external XSD schema's)
2. Models which are used in the standards based on drmCrop. It is not intended to (re)model entities already well defined by other organisations. A clear example is ISO19107 which describes geographic objects.

3.6.1 Models as reference

3.6.1.1 Data dictionary plant protection products.

This is a UML model representation of a data dictionary for plant protection products. This is formulated by the European Commission, DG SANCO.

3.6.1.2 Fertilizers.

UML model for fertilizers as formulated by AgroConnect, with consultation of the fertilizer industry.

3.6.1.3 ISO 1930 Sensor Data

The OGC model for sensor data. This is only used as a reference yet. It can be considered to use classes from this standard.

3.6.1.4 Left overs of EDI-Teelt plus.

Classes which were in the predecessor model EDI_Teelt++ and are not used in drmCrop. (Mostly junction tables in many to many relations) This package can be deleted when a complete check is made.

3.6.2 Models that are or can be used.

3.6.2.1 ISO 00639 Language codes.

This is a code list which can be relevant for data exchange in crop production.

3.6.2.2 ISO 19107 Spatial schema.

This is the basic class model on which GML is based. This model also specifies methods and can be seen as a basis for implementing geometries.

3.6.2.3 ISO 19111 Referencing by coordinates

This model specifies classes around coordinate reference systems, datums and conversions between systems.

3.6.2.4 ISO 19123 Coverage's

This model specifies Grid.

3.7 External XSD schemas

Some specifications are given in the form of xsd schema's They can in analogy to external models be separated in

1. External reference schema's which are only used as a reference, something to look at, to provide ideas. These can be models describing the same domain (crop production) or related domains. Important external schema's for reference are AgroXML and ISO11783. Both cover the same domain. The first one is used as reference. The same is true for ISO11783, but this schema is also essential in crop production as it is the standard to exchange data with task controllers on farm machinery. The possibility to map drmCrop on ISO11783 as specified in 3.9 is therefore a requirement.
2. Schema's which are used in the standards based on drmCrop. It is not intended to define entities already well defined by other organisations. The most basic schema is the basic schema of XML. Another important schema is GML, the XML specification of ISO19107.

3.7.1 Schema's for reference.

3.7.2 AgroXML

German XML model drafted in several German precision agriculture projects (PreAgro). Maintained by KTBL.

3.7.3 ISO11783.

ISO11783 part 10 is specified as xsd, but as its entities are identified by three character names and its attributes are in the XML schema specified as A, B, C, etc., a model reverse engineered from the schema is useless. For that reason an XML model of ISO11783 part 10 is drafted on basis of the entity relationship diagram in the most recent version of the FDIS of part 10.

This is not complete yet.

3.7.4 ISO11783_TaskFile_20050104.

This is the reverse engineered XML model representation forom the xsd's as published in 2005.

3.7.5 EDI_Crop version 4.

In the Netherlands EDI_Crop version 4 is the present standard for data exchange in crop production. For different types of messages, different schema's are generated, which holds only the mandatory and optional classes and attributes for those messages.

EDI_Crop V4.0 is based on drmCrop, though in some cases UNCEFACT constructs are preferred, and sometimes short term solutions are chosen by using constructs of EDI_Teelt version 3.

At present the schema for the crop recording is specified.

3.7.6 Used schema's

The following schema's are used in drmCrop:

3.7.7 XSDDatatypes.

A schema defined by W3C. It is the basis xsd of xml, defining a number of basic data types like "float", "int" and "string", but also subset of "string" like "token and "NCName". Also identifiers to use in XML files like "ID" are defined in this Schema. The XSD data types are used when generating a XML model from drmCrop.

3.7.8 GML version 3.2.1.

Contains different xsd''s as defined by OGC (Open Gis Consortium). They define the geometry objects, which are used to describe geometries in the context of drmCrop.

THE SCHEMA'S OF gml WHICH ARE OF RELEVANCE FOR drmCrop ARE IMPORTED AS MODEL IN THIS PACKAGE, THE CLASSES USED BY drmCrop ARE SPECIFIED IN THE SUB-PACKAGE "Geometries" OF drmCrop. IN A LATER STAGE A CROP PRODUCTION SPECIFIC GML PROFILE WILL BE SPECIFIED.

3.8 Java model Crop.

In principle it is possible to transform the domain model in an implementation model in java (or C++ or any other computer language). This is not the goal of the reference model rmCrop, but done for a limited number of classes to check whether the domain model is specified in such a way that this transformation is possible.

3.9 Mapping drmCrop

The package "Mapping drmCrop" holds two types of mapping:

1. Mapping Geometries of drmCrop on ISO19107 and GML.

   ISO19107 and GML are already specified by OGC, so no XML and java models are generated from drmCrop's sub package Geometries. To show how the classes specified in Geometries are mapped on on ISO19107 and GML this overview is made.

2. Mapping as a reference.

As check whether classes and attributes that are specified in drmCrop cover the whole domain of crop production, a start is made to compare drmCrop with other models or schema's as specified in the packages "External Models" and "External Schema's". There are mappings made by visualizing the different classes near each other.

As it is not possible to draw lines in EA diagrams, these overviews are not always very clear. A better method to make the mapping visible must be looked after.

3.10 MySQL model Crop.

Generating of database tables from a DDL model which is generated from a domain model is possible, but not the goal of rmCrop. To check whether the domain model is specified in such a way that this is possible at all, this is done for a limited number of classes.

3.11 XSD model crop.

As XML is foreseen now (2013) as the main protocol to exchange information between parties, an xsd model is generated from the domain model. Typical XML elements are added to the classes of the domain model like an ID for root classes and an IDREF when there is a relation to one instance of another class. (*IDREF IS NOT REALISED YET*).

The generic data types are converted to xsd data types as follows:

- Boolean --> xsd:boolean
- Integer --> xsd:int (which limits it to 32 bits). *WHEN MAXIMUM AND MINIMUM VALUES ARE SPECIFIED FOR THE RELEVANT ATTRIBUTES, A MORE SPECIFIC DATATYPE CAN BE GENERATED*.)
- Real --> xsd:decimal
- String --> xsd:string

Separate packages DataTypes and Enumerations are generated for the data types and enumerations specified in drmCrop.

There is no XML model generated for Geometries, as this is already published as GML models and schema's.

There is also no XML schema generated for XSD_Types, as this is available from W3C.

3.12 Deployment Model.

This will represent the devices and environments used to implement Crop productrion, but are in fact not part of the Crop Production Domain.

3.12.1 Platforms

At this moment only used to identify different types of (computer) platforms, which are used in the context of crop production.

3.12.2 Other Sub Packages.

Some devices are already represented in the other sub packages, but this must still be completed.

4 Modelling and naming conventions.

1. Modelling and naming conventions for the domain model drmCrop.

4.1.1 General

The domain model is kept as much as possible independent from all types of implementation characteristics. This means that:

- No identifiers are modelled, with exception of a Global Unique IDentifier (GUID), which is seen as an attribute of the object. (not strict an attribute because it has a complex nature)
- The model is not based on any computer language, which means that only a limited set of data types is used. (for example "Real")
- There are no association classes modelled for many-to-many relations, except when they hold attributes.
- There are no foreign keys modelled as attributes.

4.1.2 Specification and typographical conventions

For each Class a Definition is given and in most cases a Remark is made which contains further explanation. In some cases examples are given. When there are discussion points, those are printed in CAPITALS. When in the definition or remark a reference is made to other classes in the domain model,

the class name is printed in **bold**. To stay strict to the class name, multiples are written with a 's, in spite of the fact that this in most cases is grammatically not correct. When the name of an attribute is used, that name is given in ***cursive bold.***

The classes with their attributes are shown in the diagrams and have the default color of EA (light crème) Enumerations have a dark yellow background colour and data types defined in EDI_Teelt++ a light yellow colour. Geometries as specified for drmCrop have a light rose colour. Geometries that are data types following the criteria mentioned in

4.1.3 Naming of classes and attributes

For both, Classes and Attributes, the principle of "Upper Camel Case" is used. (THSI IS THE CONVENTION ALSO USED IN ISO11783. ONE CAN QUESTION WHETHER THIS IS CORRECT FOR ATTRIBUTES)

When a class is defined as a data type, the expression Type is always used at the end of the class name. When the class is not a data type the expression Type at the end is not used. In most cases, where there is a natural tendency to use the expression type, but not a data type as defined under 4.4 is intended, the expression Category is used. Examples are ProductCategory, TaskCategory or CropClass instead of ProductType, TaskType and CropType. (CropClass is used instead of CropCategory to stay conform FAO terminology)

Enumerations have always the expression 'Enumeration" at the end of their name. The values of the enumerations are written in CAPITALS.

4.2 Name versus Designator

In analogy to ISO11783 part 10, the expression "Designator" is used for the identification which is used when people talk or write about a specific instance of an object. (Following a language purist in the early ISO WG1 meetings, the expression "name" should be reserved for natural persons. (and I (DG) guess that some animals might also have a Name).

As a Party has a Designator, and Person inherits from Party, a Person has, apart from a first name and a last name also a Designator by inheritance, which could for example be used for a less formal name used in daily communication.

4.3 Identifiers

Identifiers and references are not specified in the domain reference model drmCrop. The only Exception is XxxGUID, where Xxx stands for the name of the class. GUID is seen as an attribute of the type GlobalUniqueIdentifier and must guarantee that the object is worldwide unique identified. The XxxGUID is given by the Organisation which is the creator of the object.

ThirdPartyxxxGUID is an optional additional attribute which is also of the the GlobalUniqueIdentifierType that is assigned by another Organisation then the "creator" of the object.

The name of the class, Xxxx is not used in case the identifier(s) are defined in a superclass with subclasses that inherit the identifier.

4.4 DataTypes.

A data type is a special kind of classifier, similar to a class. It differs from a class in that instances of a data type are identified only by their value.

So, when a class can have an identifier for identification of an instance (whether that is a GUID or a key in a database table), it is a class. A clear example is Party. As super class of Organization, Person and Equipment it has an identifier for identification of instances. An Address is an example of a data type. The value of the different attributes of the Address identifies the address. A decimal is also a very clear example of a data type.

4.5 Geometry types and or classes.

It is not the intention to model geometries as classes or data types in drmCrop. Classes from OGC models or schema's should be used. The sub package Geometries is only used to specify the OGC classes which are relevant for drmCrop. There are a few classes which can be called geometry, like for example a FieldBorder. This is done when they have a particular meaning in crop production. In general they inherit the characteristics of OGC geometries, or have relations with OGC geometries.

4.6 Code.

In the DataTypes package of drmCrop the data type CodeType is defined. The CodeType has a Code as attribute which is from the data type xsd:token. A Code is part of a CodeList, which is specified as CodeListType, so in implementing CodeType in a data base, a foreign key to CodeList can be specified, while in an XML file an IDREF can be given as reference to the CodeList.

The use of two data types; CodeType and CodeListType, deviates from "code" as defined in UN CEFACT. In UN CEFACT the "code" is not de-normalized, as all attributes of CodeListType as defined here, are made attribute of "code" in UN CEFACT.

In the description of the attributes which are of the data type CodeType, is specified in which CodeList the codes can be found. The value of the attributes of CodeListType are given in the attribute description. The CodeList specified there is either mandatory or default. In case of a mandatory or default CodeList, it is only required to specify the Code attribute. The CodeList can also be as "free", which means that it is left open to the user, or parties of users, to choose a CodeList. When a non default or free CodeList is used, the attributes of CodeList must be specified.

THIS MIGHT BE TOO MUCH FOCUSSED ON THE DUTCH IMPLEMENTATION.

4.7 Enumeration or coding list.

An enumeration is a data type with a limited number of possible items. The name represents the meaning of that item. A coding List is a list of objects of which the unique code is one of the attributes.

The enumeration becomes part of the software for implementation. The consequence is that enumerations should be restricted for those items who are expected to be stable over time. Addition of an item of an enumeration list requires a change in the specification of the standard and also a change and recompilation of the implementation software.

Items that change regularly must be specified in coding lists. The class of objects, with its attributes, in the coding list, is specified in the standard. The objects themselves are items in a database or communication file.

4.8 A Type enumeration or a Subclass.

Often a choice must be made whether a specific type of class can be indicated by an enumeration indicating the type/castegory of that class, or whether it should be modelled as a subclass. An example of the difference in approach is LineString as used in ISO11783 with enumeration values of 1 through 9 for different classes of object formed by the LineString. In drmCrop the different classes are modelled as for example GuidanceReferenceLine and Track which are sub classes of LineString. The consequence of the use of a subclass is that the software to implement the applications must be recompiled as soon as a new subclass is required. A comparable situation is described by Johnsons and Woolfe, 1996. The conclusion is that subclasses should be used when:

- The subclass has attributes which are relevant/specific for the subclass and not for other subclasses
- The subclass has different behaviour which should be taken care of by subclass specific procedures.
- The number of subclasses is known and expected to be stable over time.

5 ToDo

5.1 Issues to be solved.

1) All the data types of type "string" are to be checked whether they can stay as "string" (which seldom will be the case) or should be replaced by : normalizedString, token, Name or NCName. The "UN CEFACT Text . Type" allows strings, normalized strings and tokens, so this will leave too much freedom in cases where the allowed characters should be restricted.
2) Designator is now typed as String. Proposal: make it a token, which means that all the occurrences of #x9 (tab), #xA (linefeed), or #xD (carriage return) are replaced by an occurrence of #x20 (space) and contiguous occurrences of spaces are replaced by a single space. Leading and trailing spaces are removed. The use of Name adds the restriction that the Designator should start with a letter, underscore or colon. The use of NCName adds the restriction that no colon should be used.
3) Pal and Pw are now attributes from a Field. Proposal is to leave it as it is but make it in future versions a normal PropertyValue of a PropertyZone. (This is specific for the situation in the Netherlands)
4) Phosphate status is left as a separate class, connected to Farm. A change should be made toward a class QuotaTransfer which can be used to specify transfer between years for all categories of inputs. The phosphate status of arable land and grassland can be calculated from the PropertyValue's that refer to P_2O_5 for the PropertyZone's distinguished in the Farm. In that case point 3 should be adopted accordingly. (This is specific for the situation in the Netherlands)
5) Should we remove all attributes which are a reference to a Designators (or Name) of a coded item? The Designator is already specified in the coding list. (See for example apart from VarietyCode also VarietyName as attribute in CropField.
6) RegulatorySoilType is now an attribute of CropField. The Dutch government assigns those SoilTypes to what they call an AAN perceel, which means that it becomes an attribute of a "Field". Leave it for now to CropField or replace it to "Field"? See also issue 8) in the next paragraph to be solved later. (This is specific for the situation in the Netherlands)
7) When should we use different sub-classes and when use only one (upper) class with a type attribute defined in a coding list, which differentiates the meaning and role of the upper class?
8) Naming of attributes. For both, Classes and Attributes, the principe of "Upper Camel Case" is used. (There are modeling guidelines (DatexII) where for attributes the principle of "Lower Camel Case" is used.)
9) The relations between the classes qualified as Associations, Compositions or Aggregations are not checked on their correctness and consistency and require a further check. The relations drawn are found valid, and the multiplicities are specified as they are intended. The exact form of these relations, Associations, Compositions or Aggregations, might be changed.
10) There are in respect of converting XML schemes to UML diagrams two possibilities: 1) A DataType is an attribute of the Class 2) A DataType is drawn as a Class which has an association (Composition) with the Class that "Owns" the dataType. We must decide whether we model DataType in XML models as attribute or as composition.
11) Definitions of attributes must be completed.
12) Minimum and maximum values for those attributes which represent a value or quantity must be specified.
13) A check on national classes or attributes that should belong in a national sub model like drmCrop_NL is not complete.
14) Mapping on ISO11783 is not complete
15) Mapping on AgroXML is not complete
16) For classes with geometry aspects the most suited OGC standards must be selected.
17) Mapping of geometries on GML respectively ISO19107 must be validated.
18) The patterns for units and measures (Rate, Value, etc.) can be improved by (partly) adopting the QUDT ontology of NASA.
19) A Customer can also be a Person. How to deal with multiple inheritance?

5.2 Work still to be done.

1) The generated XML, Java and DDL models are not in line with the domain model drmCrop version 2. The templates need updates and the generated models have to be actualized whith changes from version 2.

2) Add attributes to Order and Delivery.
3) Look for a construct to specify between which possibilities an PROPOSED_CHOICE must be made.

6 Abbreviations

EA	Enterprise Architect	UML case tool.
OGC	Open GIS Consortium	

7 Definitions

Leaf class. A leaf class is a class which is a subclass of another class. This is relevant, as it is not required to generate an ID when transforming the domain model to an XML model.

Root class. A root class is a class which has sub-classes.

8 Literature.

Biesalski, E. 1954. Landarbeit und Technik. Heft 32. Eine schriftreihe des Max-Planck-Institutes für Landarbeit und Landtechnik. Herausgegeben von Prof. Dr. G Preuschen, bad Kreuznach. Fünfte, erweiterte Auflage. Verlag Paul Parey, Hamburg und Berlin. 1954.

Goense, D. and J. W. Hofstee. 1995. Deliverable Report 1.2.c, Information Model, Annex 2.3.9. ESPRIT III project 7318. Computer Integrated Agriculture.

Johnson, R. and B. Woolf. 1996. The Type Object Pattern.
http://www.cs.ox.ac.uk/jeremy.gibbons/dpa/typeobject.pdf

9 Appendix I. Background on some design patterns.

9.1 Allocations

Allocation has two subclasses: ProductAllocation and ProduceAllocation which deal with product and produce respectively.

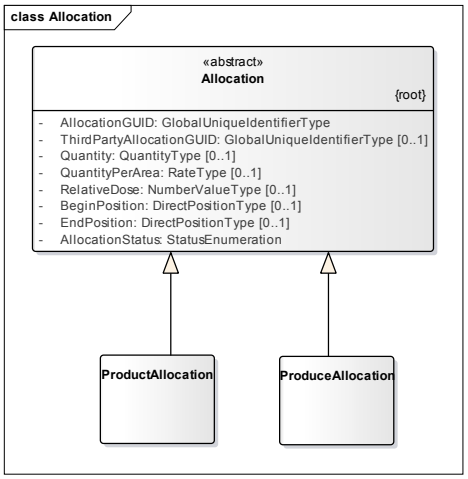


Figure 1. Allocation with two subclasses

Allocation is used to describe the application of products on field surfaces, application of (protecting) products on produce, sorting produce, mixing products or produce, etc. These activities in respect of allocations are specified hereafter.

9.1.1 Harvesting from plots.

Harvesting is an Operation which results in one or more ProduceAllocations. The farm business process determines on which level of detail the ProduceAllocations will be recorded.

In case of global registration only one ProduceAllocation and one HarvestingZone exists for the whole CropField. The BatchLot will be collected in one or more storages, but holds one BatchLot identifier pointing to that HarvestingZone.

In case of detailed registration, a harvesting machine will keep track of the surface which is harvested when a particular container is being filled. This container can be located on the harvesting machine like on grain combines, but can also be a tipping trailer driving near the harvester like most choppers. The surface is a **HarvestingZone** and will be the part of strips harvested for one particular filling of the container. After emptying the container a new BatchLot is created.

When implementing detailed registration, it will for example be the combine harvester which records the harvesting zones and the ProduceAllocation when unloading. The time and geographic position of this allocation can be recorded. When a trailer is equipped with loadcells for weighing, it could also record a loading action around the same time and very close to the position recorded by the combine. So there is a redundant recording of the allocation possible with slightly different quantities, time and location.

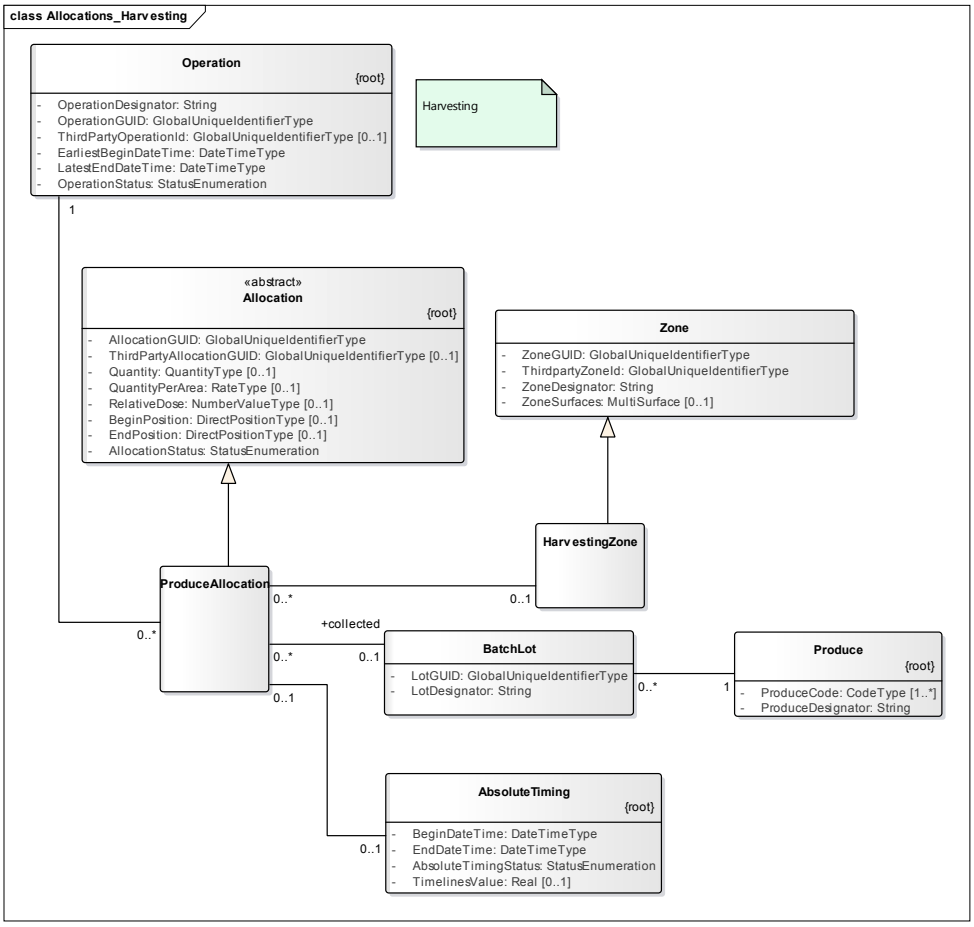


Figure 2. ProduceAllocation in the context of harvesting.

9.1.2 Applications on Crops.

An Operation which applies products (fertlizers, seed, plant protection products, etc.) is planned for a CropField.

An **Operation** of this type has one or more **ProductAllocation**s. In case of precision Agriculture there are different TreatmentZones for which a different ProductAllocation can be specified. (A CropField has always at least one TreatmentZone).

The ProductAllocation specifies the Quantity or QuantityPerArea and refers to either a Product or to a particular BatchLot of a Product. The latter will often be the case when applying manure, such that the mineral content of that BatchLot can be accounted for. (It is possible to specify which BatchLot of a Plant Protection product is used).

An Operation can also have more ProductAllocations to account for differences in rate depending on the time of application. An example is a fertilize advice, which might have different rates depending on the time of application. In the ProductAllocation, the status attributes with the value PROPOSED_CHOICE can be used for that.

TODO Look for a construct to specify between which possibilities an PROPOSED_CHOICE must be made.

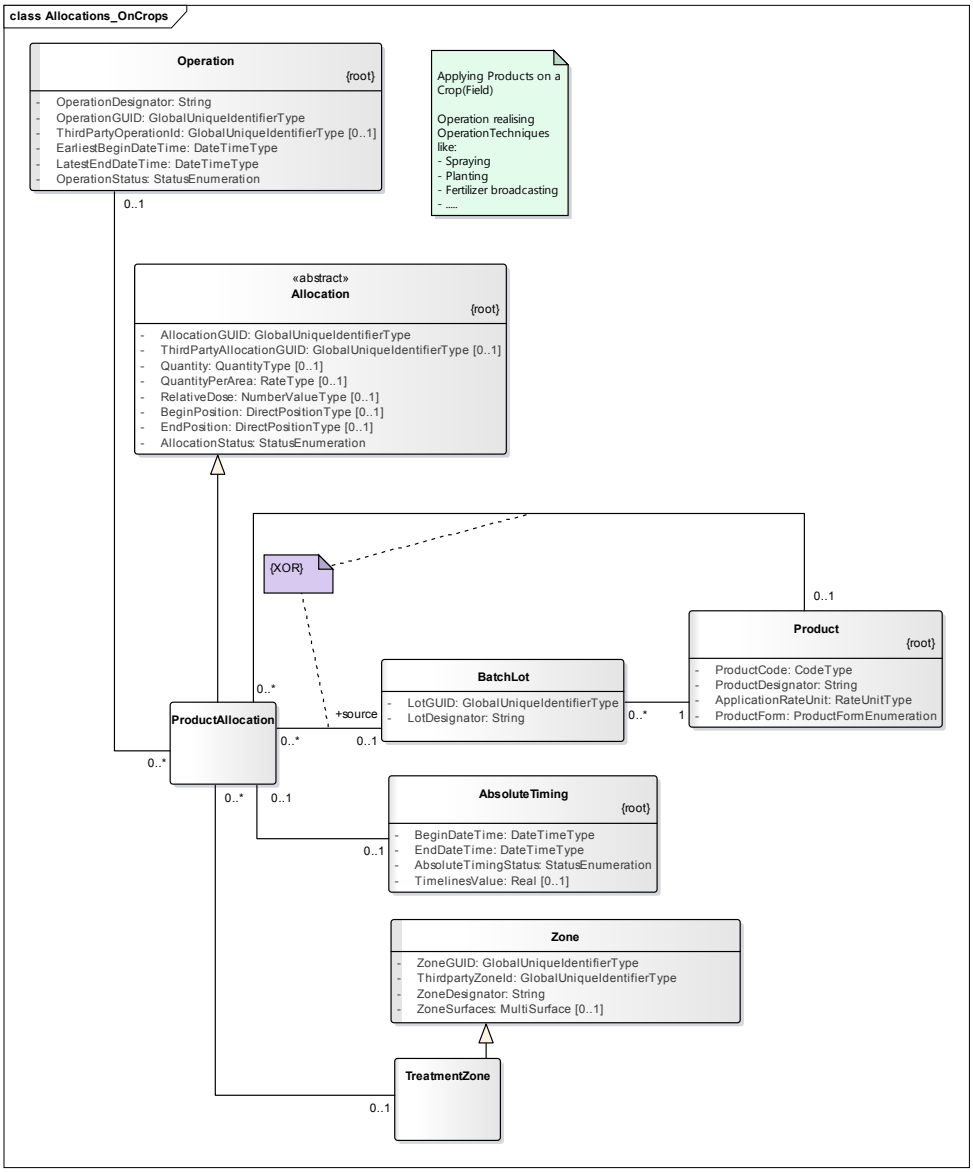


Figure 3. ProductAllocation in the context of application of products on a CropField.

9.1.3 Aggregating Lots

There are products for the retail market which can only be produced by assembling from different sources. An example are the packages with three colors paprika and mixed bouquets of flowers. Another clear example are farmers or cooperatives that blend fertilizers to a specific specification. Another example is that a trader can only deliver an order when he combines products from two different sources. In all these cases products or produce is used from BatchLots

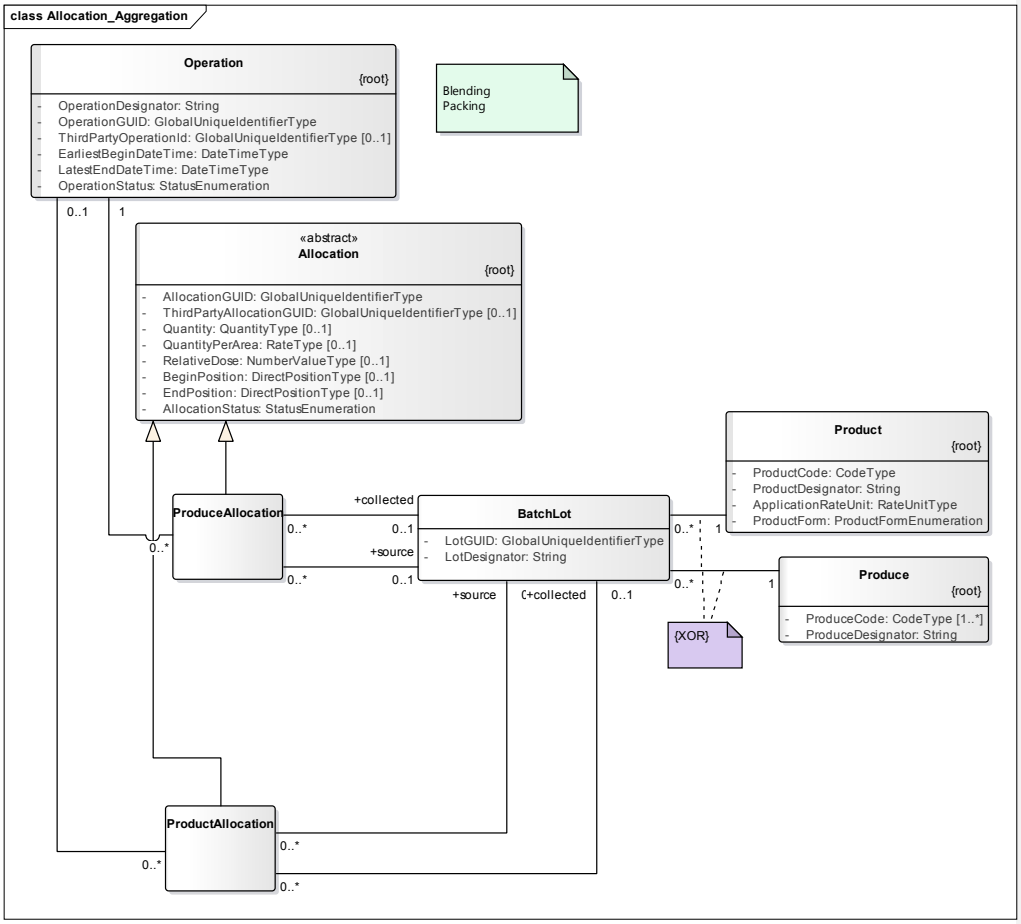
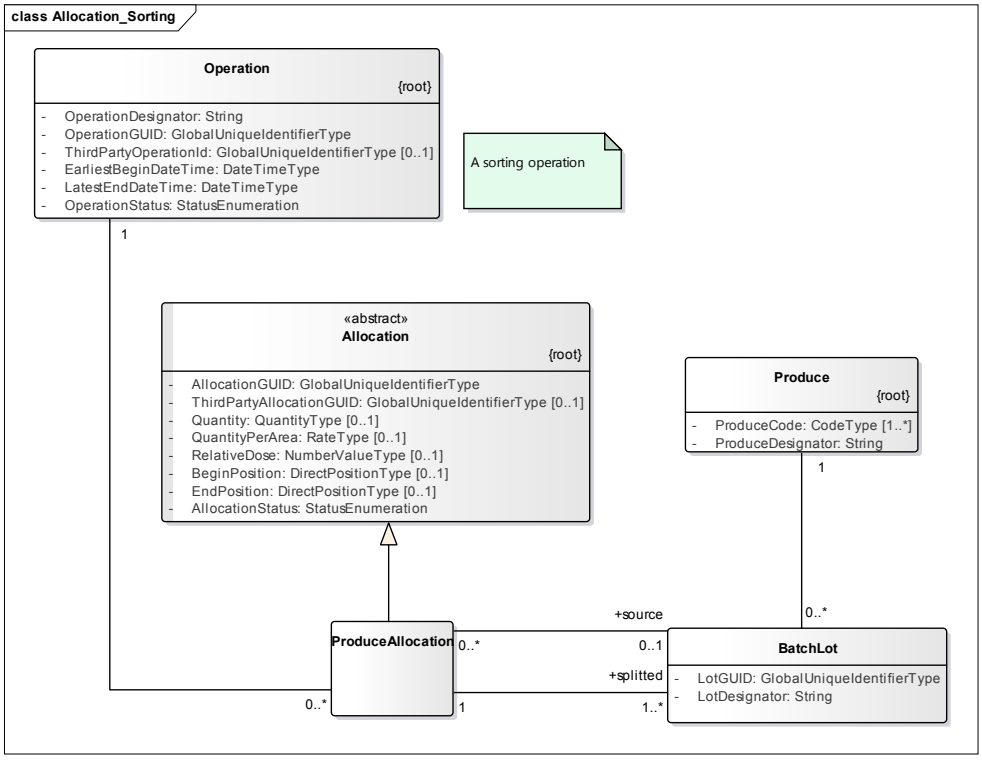


Figure 4. Poduce- and ProductAllocation when assembling new BatchLot's.

9.1.4 Sorting Lots

Lots of produce can be sorted in different ProduceQualityClass'es. In such a case each ProduceAllocation defines one source BatchLot and one splitted BatchLot. Sorting leads to at least two different BatchLots, and a sorting operation has therefore at least two ProduceAllocations.

It looks logical to have a multiplicity of 2..* on the splitted BatchLot side, but when we want to keep track of the quantities coming from a source and going to a particular splitted BatchLot, we need for each splitted BatchLot a ProduceAllocation, as this class specifies the quantity.



9.1.5 Application of Products on BatchLot's of Produce.

When products are stored, operations are carried out on the storage facilities (or containers) which hold the BatchLot's. These operations can control environmental variables like temperature and humidity, but there are also operations which add products to the Produce. An example is germination retarding products in potatoes, but also pesticides can be added to prevent pests. Another example is adding a conservation product to grass because it is too wet during harvest.

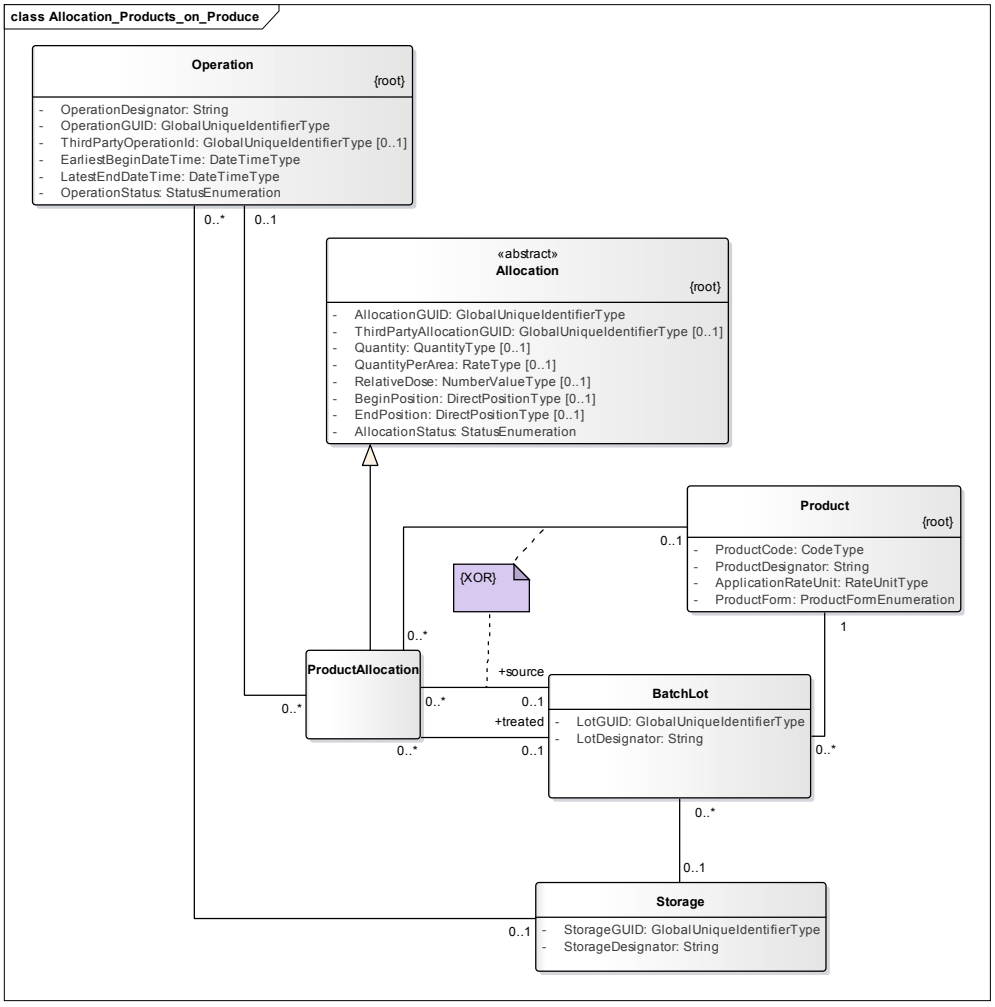


Figure 5. Application of Products on a BatchLot.

9.2 Analyses and sampling

9.2.1 Sampling

There are four types of samples recognized; soil samples, tissue samples, produce samples and product samples. Soil samples are collected from a particular vertical layer. One or more vertical layers form a profile. The profile has a position in space.

Tissue samples can be taken from growing crops (the possibility to take them from animals is not considered yet for this reference model). Also Produce samples can be taken from crops when it is intended to have a pre harvest analyses from those parts that will become Produce. Produce samples can also be taken from BatchLots of produce and ProductSamples only from BatchLots. An example of the latter is a sample from manure.

Samples on crops or on vertical layers as part of profiles can be taken either from one point, or from a collection of points or from a TreatmentZone. As an ActivityField and a CropField have always at least one TreatmentZone, this covers also sampling whole fields.

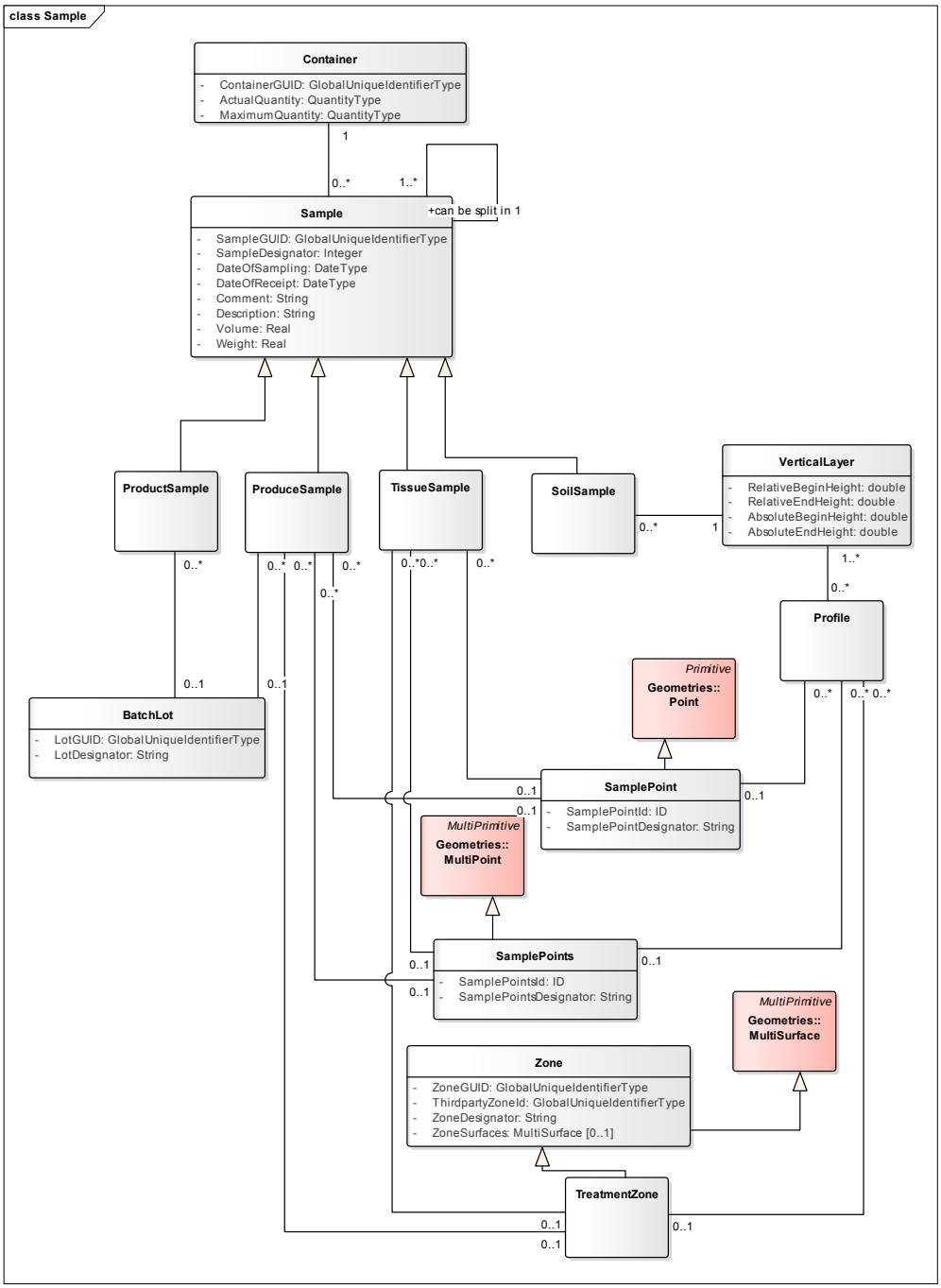


Figure 6. Different categories of samples and where they can be taken from.

Samples are collected in containers, which apart from conditioning of the sample and keeping it together plays an important role in identification of the sample. It is either the place to hold a sample id, or it has its own id which in the latter case must have a time indication to connect it to a particular sample.

Collection of samples is an Operation and has like other Operations also OperationTechniques. Manual sampling can for example be distinguished from sampling with a quad. By assigning the sampling operation to a Task it is possible to specify the Worker and participating equipment.

Samples can also be split. It is not uncommon to separate a part of a sample for biological soil analyses (nematodes) and send it to another laboratory for chemical analyses.

9.2.2 Analyses of Samples.

Samples are analysed and in this reference model it is assumed that this is done by a laboratory which has an accreditation (certificate).

The result of an analyses is a propertyvalue of a particular PropertyVariable. A number of specific PropertyVariables are shown as subclasses from PropertyVariable in Figure 7. In tracking and tracing the substance which is a generic expression used for active ingredients of PlasntProtectionProducts, plays an important role as Produce is restricted to certain levels of residue.

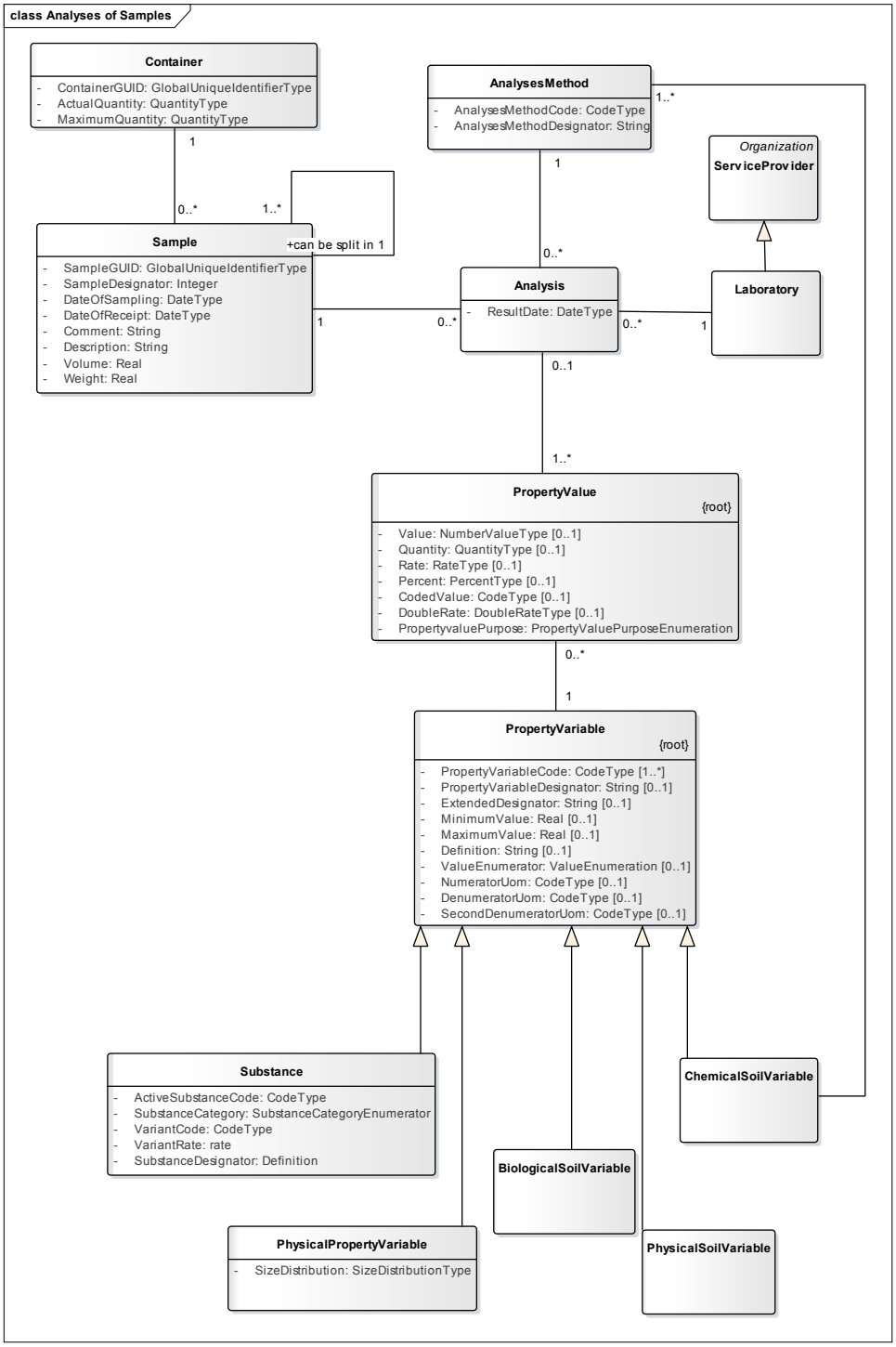


Figure 7. Analyses of samples.

9.3 BatchLot and Delivery

The class BatchLot is essential in tracking and tracing. When products are delivered to a retailers it is a requirement that the Lot is identified on a specific category of packages.

Opmerking [DG1]: Refer to GS1 for more details

The difference between Batch and Lot in the agricultural production chain is not clear defined and as long as that is not the case there is no difference made between Batch and Lot. The class **BatchLot** is used. It is defined as a specific identified number or quantity of a Product or Produce.

When products are delivered to a customer which can be an intermediary (trader, processor, and packer) or a retailer, one or more BatchLots are delivered conform the order of the customer.

The pattern is shown in Figure 8.

TODO Add attributes to Order and Delivery.

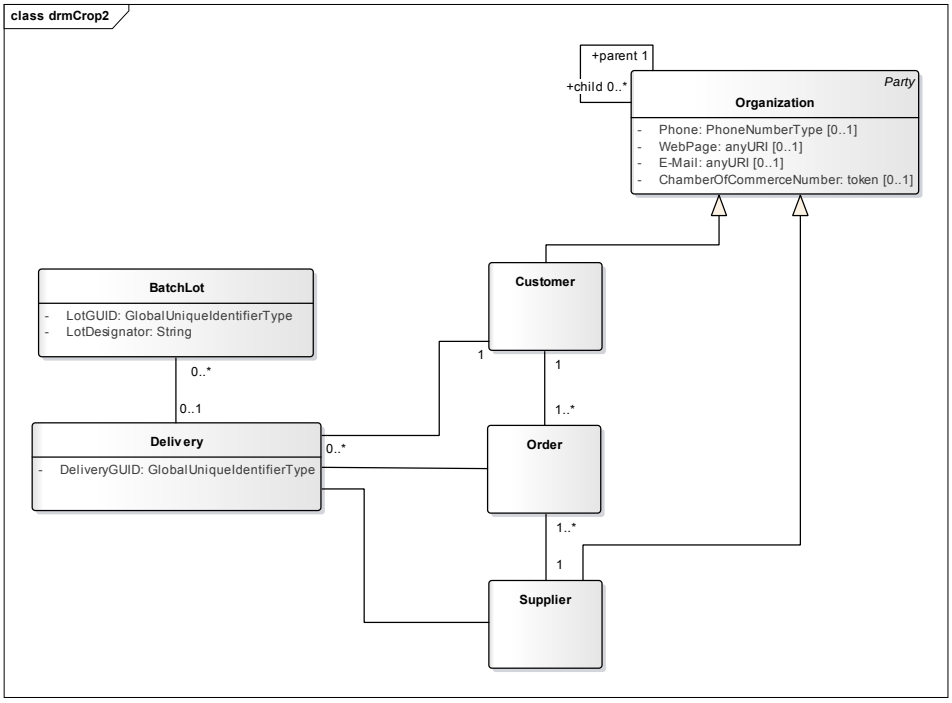


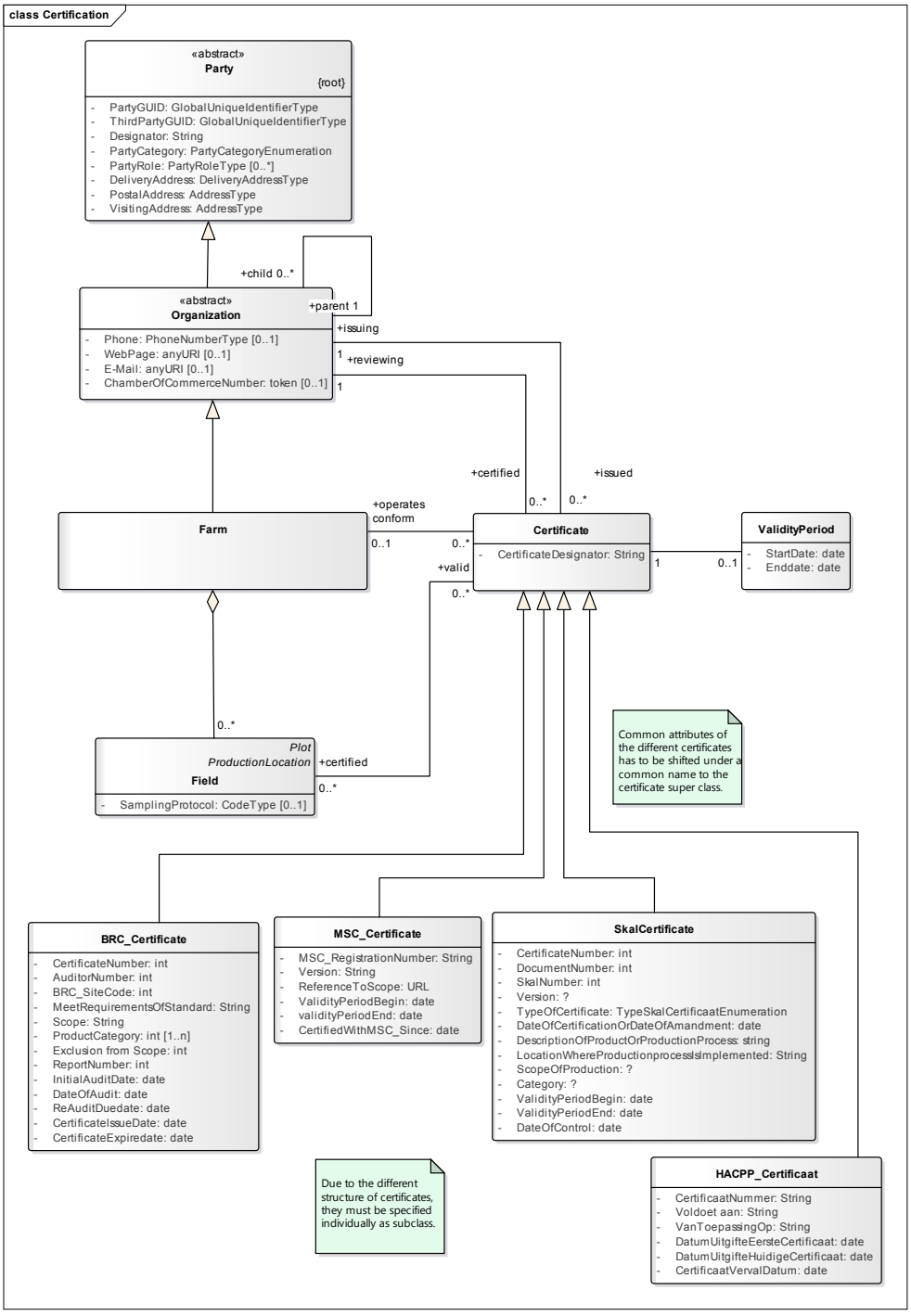
Figure 8. Pattern for Delivery and BatchLot.

Tracing back to the source of a Product is possible by defining allocations. See 9.1 for further details.

9.4 Certificates

An analyses is made of a number of certificates that are used in Agriculture. Up to now those certificates are explicitly modelled in the reference model as they have a quit different structure. The only attribute in common is the Designator, and likely the number can also be brought to a common attribute.

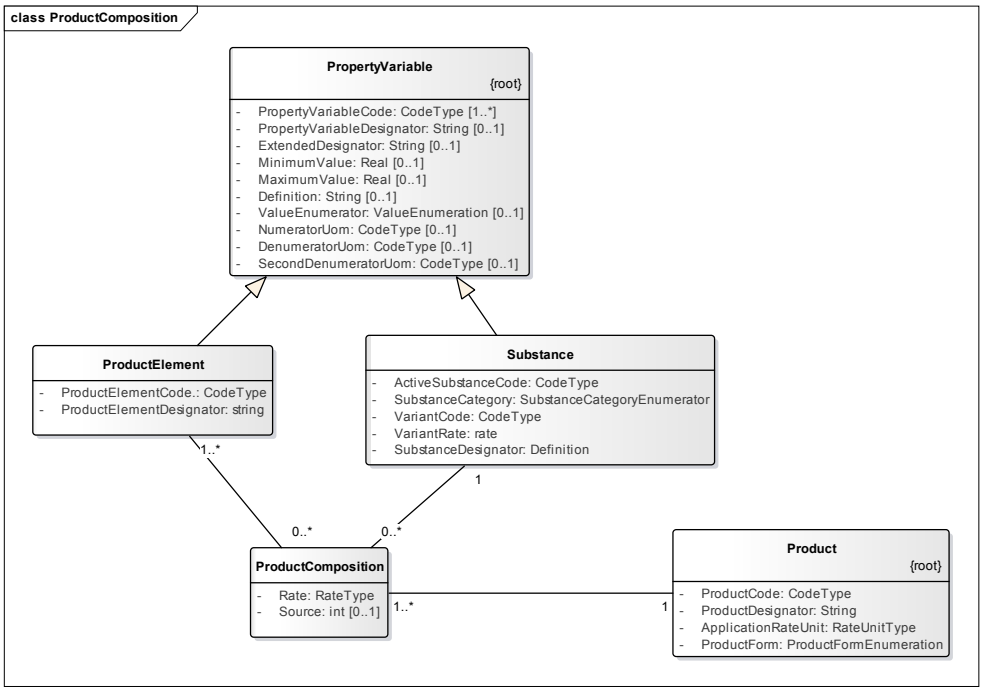
TODO: Issuing and reviewing of certificates must be worked out.


class Certification
«abstract»
Party
{root}
- PartyGUID: GlobalUniqueIdentifierType
- ThirdPartyGUID: GlobalUniqueIdentifierType
- Designator: String
- PartyCategory: PartyCategoryEnumeration
- PartyRole: PartyRoleType [0..*]
- DeliveryAddress: DeliveryAddressType
- PostalAddress: AddressType
- VisitingAddress: AddressType
+child 0..*
«abstract»
Organization
+parent 1
- Phone: PhoneNumberType [0..1]
- WebPage: anyURI [0..1]
- E-Mail: anyURI [0..1]
- ChamberOfCommerceNumber: token [0..1]
+issuing
1
+reviewing
1
+certified
+issued
0..*
0..*
Farm
+operates
conform
0..1
0..*
Certificate
- CertificateDesignator: String
1
0..1
ValidityPeriod
- StartDate: date
- Enddate: date
+valid
0..*
0..*
Plot
ProductionLocation
Field
+certified
0..*
- SamplingProtocol: CodeType [0..1]
Common attributes of the different certificates has to be shifted under a common name to the certificate super class.
BRC_Certificate
- CertificateNumber: int
- AuditorNumber: int
- BRC_SiteCode: int
- MeetRequirementsOfStandard: String
- Scope: String
- ProductCategory: int [1..n]
- Exclusion from Scope: int
- ReportNumber: int
- InitialAuditDate: date
- DateOfAudit: date
- ReAuditDuedate: date
- CertificateIssueDate: date
- CertificateExpiredate: date
MSC_Certificate
- MSC_RegistrationNumber: String
- Version: String
- ReferenceToScope: URL
- ValidityPeriodBegin: date
- validityPeriodEnd: date
- CertifiedWithMSC_Since: date
SkalCertificate
- CertificateNumber: int
- DocumentNumber: int
- SkalNumber: int
- Version: ?
- TypeOfCertificate: TypeSkalCertificaatEnumeration
- DateOfCertificationOrDateOfAmandment: date
- DescriptionOfProductOrProductionProcess: string
- LocationWhereProductionprocessIsImplemented: String
- ScopeOfProduction: ?
- Category: ?
- ValidityPeriodBegin: date
- ValidityPeriodEnd: date
- DateOfControl: date
Due to the different structure of certificates, they must be specified individually as subclass.
HACPP_Certificaat
- CertificaatNummer: String
- Voldoet aan: String
- VanToepassingOp: String
- DatumUitgifteEersteCertificaat: date
- DatumUitgifteHuidigeCertificaat: date
- CertificaatVervalDatum: date

9.5 Composition and Substance

For a number of products or Produce it is required that their composition is specified. At this moment two Patterns are developed. ProductElement is developed in drmCrop, while Substance comes from the Plat Protection Product datadixtionary.

TODO: fins a common approach for different categories of products.



9.6 CropProductionUnit

Crop Production unit is still in the reference model as a heritance of “Teelt” from IMOT. There are two arguuments to maintain it in the reference model.

1. There are farmers who want to aggregate different fields so they can use the same prescriptions for the different fields as they have properties in common. The definition of CropField, in line with rules from the EU does not allow to treat surfaces which are not continuous as one CropField. CropProductionUnit facilitates to group more CropFields together.
2. Activities on a CropProductionUnit can extend the time that it is grown on a field. Sometimes work on a BatchLot of planting material is for a particular CropProductionunit, and after harvesting also storage, drying, sorting on BatchLot’s of Produce can be assigned to a particular CropProductionUnit.

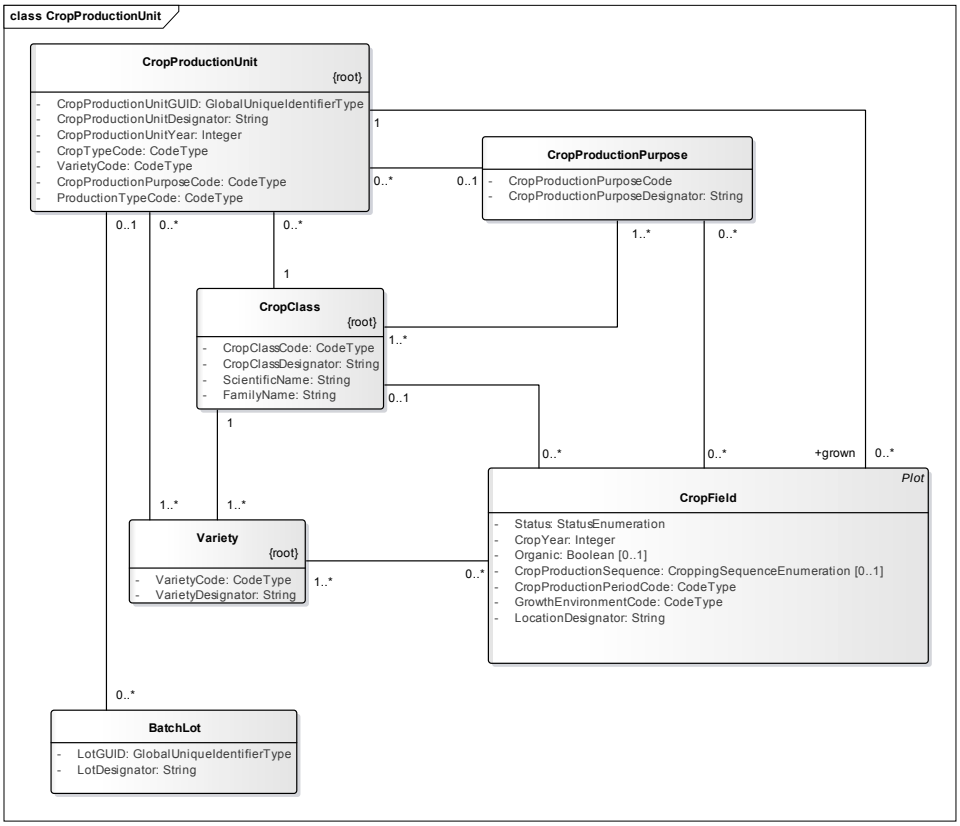


Figure 9. CropProductionUnit

9.7 Job Task Operation

9.7.1 Operation

From an agronomical point of view, certain activities are to be performed on crops growing on a field, or sometimes also on batches or Lots of product (for example seed potatoes) or stored produce (grain in a silo). Also repair and maintenance activities on equipement and buildings are Operation's.

Such an activity with an agronomical purpose to be performed on a certain object (Field, Batch,) in a certain time period is called an **Operation**.

Definition.
An **Operation** performs a particular **OperationTechnique** on a particular **CropField, ActivityField, CropProductionUnit or BatchLot** to realize a certain **CulturalPractise**.

REMARK. Repair and maintenance activities of machinery and buildings and activities around animal husbandry can also be called operations. In that sense the definition will be extended to more objects and more purposes apart from CulturalPractise.

9.7.2 CulturalPractice

The agronomical purpose (objective) of the **Operation** is called a **CulturalPractise**.

It is not sure whether **CulturalPractise's** can be standardized internationally. From one side they are quite common for arable farming and forage production all over the world, on the other side it is difficult to foresee all different regional situations and especially the level of required detail might differ. AgroConnect CodeLists CL291 and CL293 are an effort to standardise CulturalPractises on two levels of detail.

The need for a relatively short list of standardized CulturalPractice's might come from processors that need information on how the crop is grown for evaluation or for a certification authority. A solution could be that a farm management system leaves it open for the farm manager to define the CulturalPractice's. The farmer specifies the code and the coding list of the CulturalPractice's which are required by his processors or certification authorities. In that way it can be adapted on the local situation. (A problem might arise when different party's require a different structuring/categorization of CulturalPractices.) Maybe it is possible to come to a national standard list for CulturalPractices.

Operations are even much more region-, farm-, crop-, season- and time- specific. The CulturalPractice "Stubble tillage" can for example be realized by two operations "Chisel ploughing" (in Dutch "vaste tand cultivateren") followed by "stubble ploughing". (In Dutch "stoppel ploegen"). Operations that realize the Cultural Practice "Fertilizer application" can be: "phosphate fertilization", "first nitrogen application", "second nitrogen application"," third nitrogen application", "potassium application", etc. The Cultural practice "disease control" in potatoes will have such a large number of operations called "phytophthora spraying" that a farmer might even not take the effort to give them a distinguished Designator, while they are all different Operations, with a different time period (and have therefore a different Id).

It is impossible to have a standardized coding list for Operations. They are instances of an activity following an OperationTechnique with a certain purpose (CulturalPractise) on a certain object in a certain time period. The consequence is that a farm management system should give the farm manager (or advisor) the possibility to define Operations, which are linked to an eventually standardized "**CulturalPractice**" and performed following a standardized **OperationTechnique**. (See further on.) A TaskController on a tractor can show the designator for the Operation, which has a meaning for the driver, but has no meaning for the TaskController itself. (For the TaskController it gets a meaning by the link to the standardized **OperationTechnique**)

9.7.3 OperationTechnique

CulturalPractices, and as a consequence **Operation**s, can be performed by different **OperationTechnique**s. As example: fertilizers can be broadcasted or side banded. Planting of seeds can be done as broadcasting, drilling or precession planting. The **OperationTechnique**'s must be internationally coded and well defined because it must be known which implements are able to perform these **OperationTechnique**'s, and TaskControllers commanding those implements must know which implement are able to perform that technique. A proposed standard list of **OperationTechnique**s is CL292 from AgroConnect.

The farm manager is responsible that he has each of his CulturalPractice's connected to one or more OperationTechnique's. This is the list OperationTechniqueCulturalPractice, which is part of his Farm management System. (That can be his selection of the junction table of Appendix I) When specifying an Operation he will first choose the CulturalPractice to fulfill and then he can choose the OperationTechnique to specify how this Operation should be carried out. The part of the decision support system that is responsible for resource scheduling has the information on which implements are able to perform the Operation following the specified OperationTechnique, as Implements must have a list of one or more supported OperationTechniques. This requires a junction table Implement (Device in ISO11783) - OperationTechnique. An example of a list of Implements is provided in Appendix III and the junction table of the implements with the OperationTechniques in Appendix IV. Some Decision Support Systems that advise on the amount of inputs to use might also want information on the OperationTechnique, as the efficiency of an input can differ due to the chosen technique. Side banding of phosphate for corn is for example more efficient then broadcasting phosphate and an advice might differ in the amount

depending of the applied OperationTechnique. A possible list of OperationTechnique's which can realize certain CulturalPractices is given by AgroConnect.

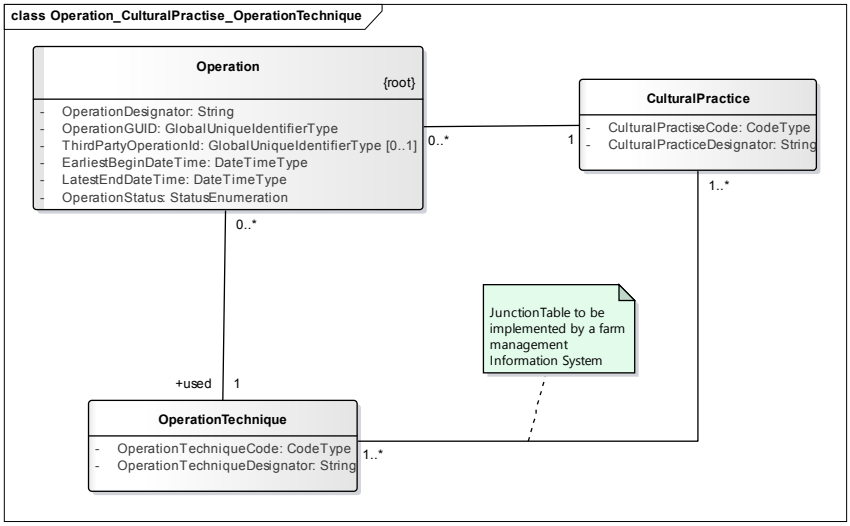


Figure 10.

9.7.4 Task.

Operations are seen from an agronomical point of view. Those Operations are to be performed by man and/or machinery. The execution of an Operation by man and/or machinery is called a **Task**.

Definition.
A **Task** is the execution of one or more **Operation**'s.

Remark.
An **Operation** can be executed by more **Task**s.(See example below). Therefore, there is a many to many relation between Task and Operation.

Example.
There is one **Task** "Harrowing and Drilling", which performs two **Operation**'s; one is "Harrowing" and the other is "Drilling".
On the other hand, there can be two **Task**'s that realize one **Operation**. For example two combines harvesting on one **ActivityField**.

It is possible that one Operation is realized by two different combinations of man and machinery. An example is combine harvesting, where one Operation called "Combine harvesting of wheat behind the barn" must be realized within a certain time period by two combines, each with a driver.

Another situation is where two different Operation's, which realize different CulturalPractise's are realized at the same time with one combination of men and machinery. An example is the Operation "seedbedpreparation on Over de Weg" which is combined with the Operation "wheat drilling on Over de Weg", by a combination of a driver, a tractor, a rotary harrow and a drill mounted on the rotary harrow.

The consequence is that an Operation can be performed by two Task's (the combine example) and that a Task can perform two Operation's (the harrowing drilling example). The drmCrop reference model specifies this as a many to many relation, which will require that the farm management System has a junction table.

> Problems that arise by creating yield maps when two combines worked on the same field can be caused by not realizing that there is one Operation, harvesting, performed by two Tasks. When there is defined that the Operation is performed by two Task's, in the junction table TaskOperation, there is a control that yield maps logged under those two tasks must be combined to realize the complete map for the Operation.

A combination of man and machine executing a Task is called a ManMachineSystem. A ManMachineSystem can be defined as man and machinery which are "physically connected". An example is a tractor with an implement and a driver. The practical reason to distinguish a ManMachineSystem is that it, normally speaking, will be controlled by one TaskController.

9.7.5 TaskCategory

As a Task is (part of) one or more Operations performed on a particular ActivityField in a particular time period, there is a need for a TaskCategory class to define the kind/type of Tasks which can be performed by the available equipment and that are relevant for the farm. As there is a large number of combination of implements possible and farmers will be very inventive in new efficient ways of working, it makes little sense to compile a standardized list of coded TaskCategories. There should be a possibility for a farmer to define his TaskCategories. An example of a list of Taskcategories is given below.

TODO Check example with latest version of CL292.

TaskCategory	**OperationTechnique(s)**
Stubble ploughing	Stubble moldboard plowing
Ploughing	Moldboard ploughing
Seedbed preparation	Rotary harrowing
Grain Drilling	Drilling
Drill combination	Rotary harrowing + Drilling
Granulate application	Drilling
Planter combination	Rotary harrowing + Potato planting + Full field granulate application + Band granulate application[2]
Spraying	full field spraying
Spraying and transport	Full field spraying + water transport by sprayer
Potato ridging	ridging
Fertilizer application	Broadcasting
Manure application	Injection
Baling	Square baling
Combining	Combining
Forage chopping	Forage chopping
Sugar beet harvesting	sugar beet lifting + Sugar beet topping + Sugar beet loading
Potato harvesting	Potato leaf chopping + Potato lifting + Potato loading
Water transport	water transport by tankwagon.
Grain Transport	Transporting grain
Sugar beet Transport	Transporting rootcrops
Potato Transport	Transporting rootcrops

[2] This is a combination with a rotary harrow in the front, a full field granulate applicator which disposes in front of the harrow, a potato planter in the rear hitch with a granulate applicator mounted in the planter for application in the plant row (banding). This is a challenging combination; in one ActivityField two different operation techniques (full field and banding) are applied for one CulturalPractise (nematode control). So apart from an application rate, also an application method must be specified in treatment zones. This situation needs to be worked out. (a solution could be to define two different Operations, one for banding and one for full field application).

Bale loading	Loading bales
Bale transport	Transporting bales

9.7.6 Job

A job is the work executed by one ManMachineSystem, or a number of ManMachineSystem's which work organisationally together. An example is one ManMachineSystem consisting of one combine harvester and a driver, working together with a number of other transport ManMachineSystem's who each exist of a driver, a tractor and a trailer (grain cart).

In resource planning it is relevant to formulate Job's, as it must be guaranteed that the resources (men, tractors and implements) required to perform the Job are available in the same time period and are not assigned to other Job's in that period. Formulation of JobCategories, which specify the TaskCategories that can be combined in a Job and the resources which can form a Job is required for efficient Job formulation. Estimation of the required time to execute a Job is determined by the resources which form the ManMachineSystem's that realize the Job and their way of operating (as example the size of grain carts and unloading during combining or unloading at the headland). Some examples of JobCategories based on the available resources are given below.

During execution of a Job it might be relevant to have a form of Job control. In a case of silage maize harvesting it is relevant to adjust the number of transport units, based on idle time of either the chopper or of the transport units.

The relation between Job and Task is quit straightforward; A Job exists of one or more Task's.

Example of Job Categories. TODO Formulate task categories in this example.

JobCategory	**Resource.**
Grain drilling	Worker 1
	Green tractor
	Grain drill
	Wagon 6 m old
Potato harvesting with two transport units	Worker 1
	Worker 2
	Worker3
	Worker4
	Big Green tractor
	Blue tractor
	Green tractor
	Potato harvester
	Tipping trailer 7 ton
	Tipping trailer 9 ton
	Potato Hopper
	Conveyer
Potato harvesting with one transport units	Worker 1
	Worker 2
	Worker3
	Big Green tractor
	Blue tractor
	Potato harvester
	Tipping trailer 9 ton
	Potato Hopper
	Conveyer

9.8 Party and sub classes

Party is the abstract root class of all entities that have a legal connotation in Business activities. drmCrop distinguishes two abstract subclasses; Organisation and Person.

In analogy of Inspire, there is an AgriculturalProductionHolding (Inspire's Holding), which can realise its activities on one or more sites. The sites are represented by Farm or GreenhouseComplex in drmCrop. (More categories of sites may follow when the scope will be broaddifferent classes of service providers etcened) Apart from Farm and GreenhouseComplex some sentities are specified like Contractor and different classes of service providers. Apart from that also Supplier and Customer are specified. In some processes they represent a specific role of an organisation, but they can be identical to one of the non-abstract organisations like contractor and farm.

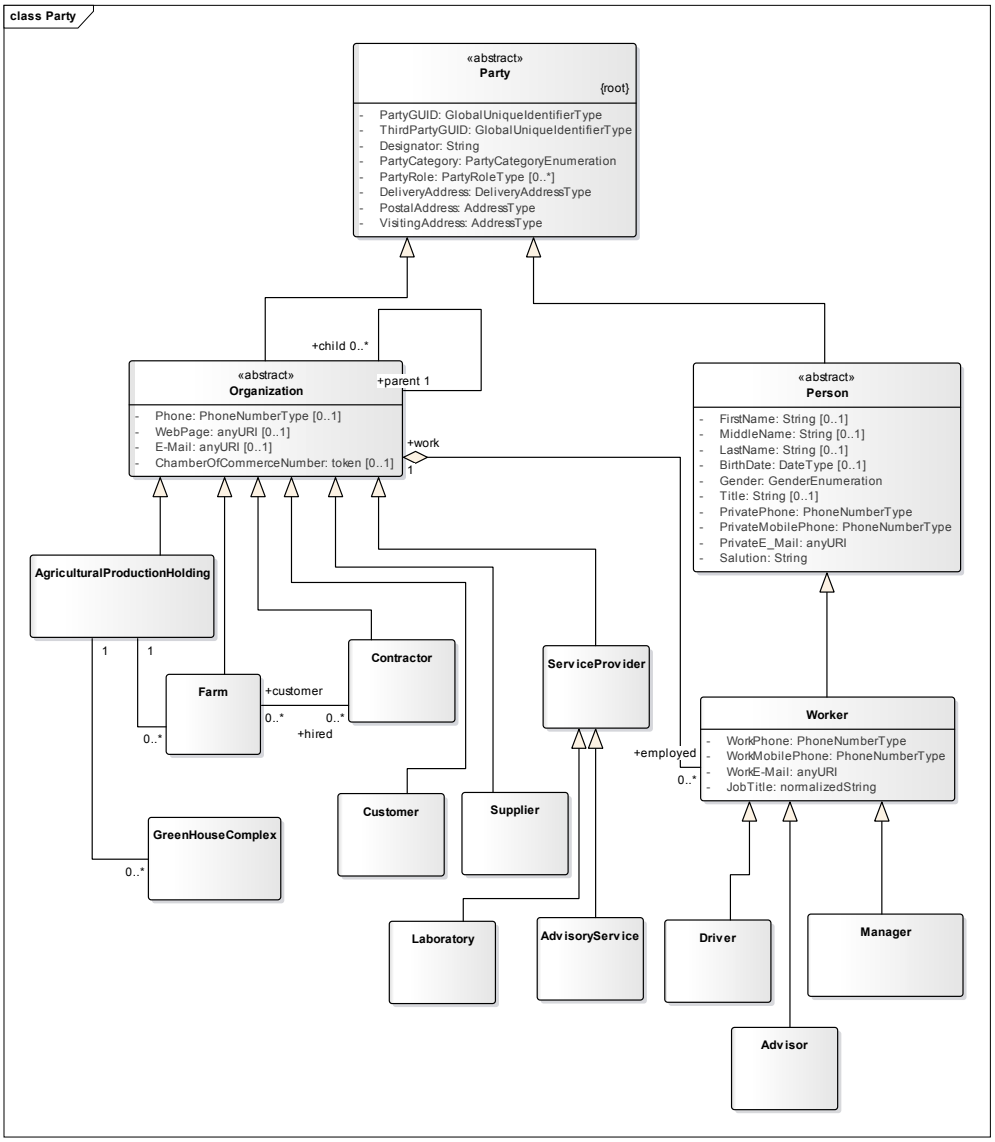


Figure 11. Party and its subclasses.

9.9 Plots and Fields.

Inspire refers to Plots which can be situated in other Plots. This allows for a lot of freedom to define different areas for CropProduction. In drmCrop is chosen to distinguish plots that have specific characteristics or have different purposes. All variables that the different categories of Plot have in common are specified in an abstract Object “Plot”, but this has to be implemented by a specific subclass. Like in Inspire, there is a reciprocating relation, but this will be eliminated later on, as the relations will be made specific for the different categories of plots.

All categories of plots in drmCrop are continuous surfaces.

Fields are the more or less permanent plots which stay over more years, up to a moment that structural changes occur. From season to season one or more CropFields can be situated in a Field. A CropField is grown with one CropClass and has a begin and an end date.

An activityField is the continuous surface at which an Operation will be executed. In most cases that will be the exact surface of a corresponding CropField, but it can be that two adjacent CropFields, which are identified separately (for example because a difference in the QualityClass of seed potatoes) will in executing Operations be operated as one unit. It is also possible that an order is given to a contractor to operate only a part of a CropField. (For example sugarbeet harvesting in two separate periods of time)

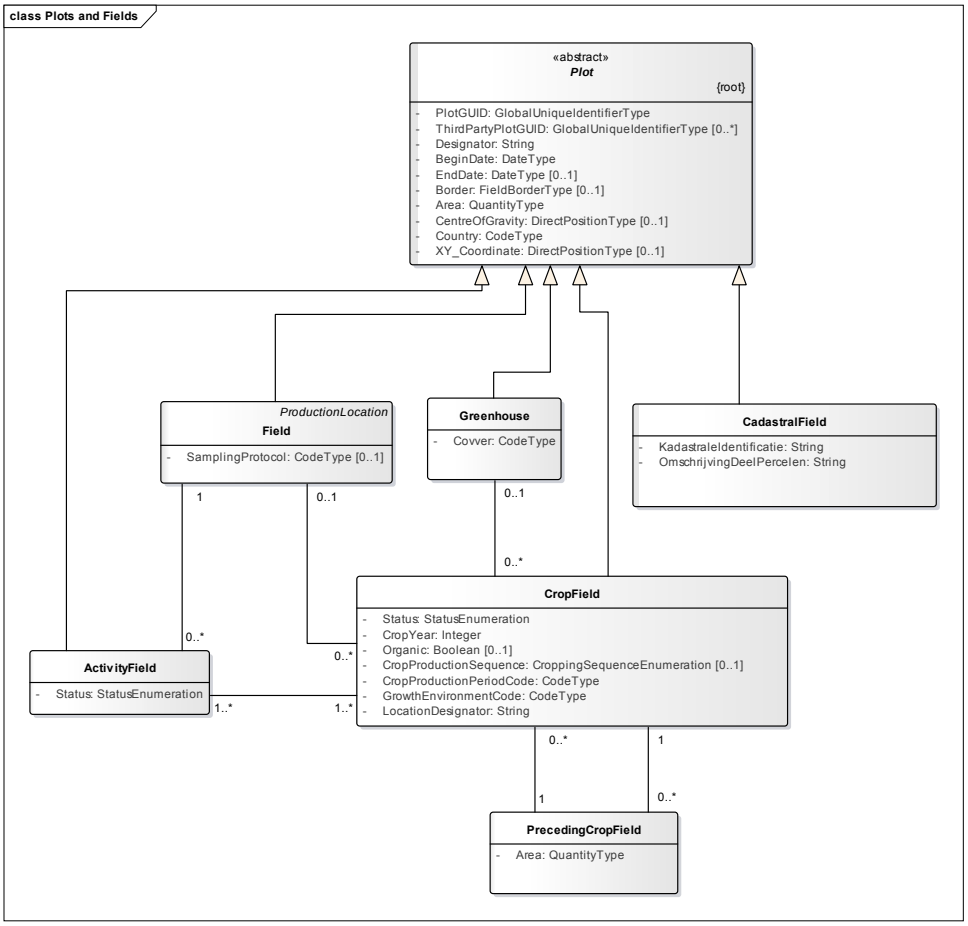


Figure 12. The different categories of Plot

A special category of plot is a Greenhouse. It is equivalent to a Field in that sense that its surface will stay constant for a longer period of time. Like a Field, it is possible to locate one or more CropFields during the growing season in the Greenhouse.

In the situation of CropFields in a Greenhouse, the restriction on a CropField of a continuous surface should be handled not too restrictive, as paths might separate surfaces, but it is left to the grower to neglect this. The requirement of one CropClass must be maintained, so surfaces separated by a path must be of the same CropClass when they are assigned to one CropField.

CadastralField is corresponding to the National registration systems and might have country specific subclasses.

9.10 Products and Produce

Products are the inputs for agricultural Production and Produce is the result.

Fertilizers, seed, planting material, irrigation water are among the products. Crops can produce one or more types of produce. A maize (corn) crop for example can produce whole crop silage, corn cob mix, kernels and fresh cobs. Tomatoes can be delivered in different grades and on a bunch or loose, wheat produces grain and straw.

Products can be worked out in more detail by subclasses, as can be seen by the example of Phytophthora fungicides.

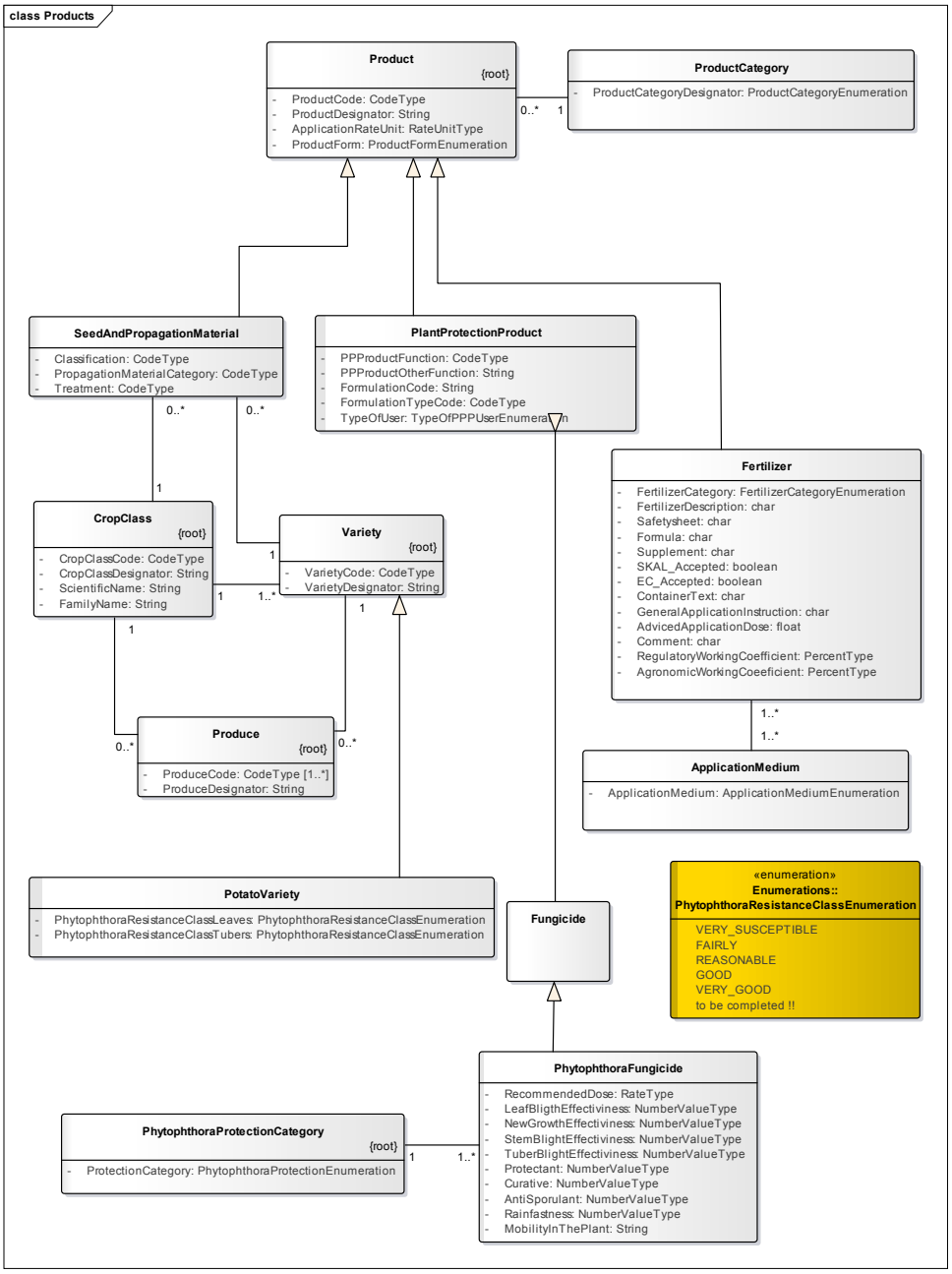


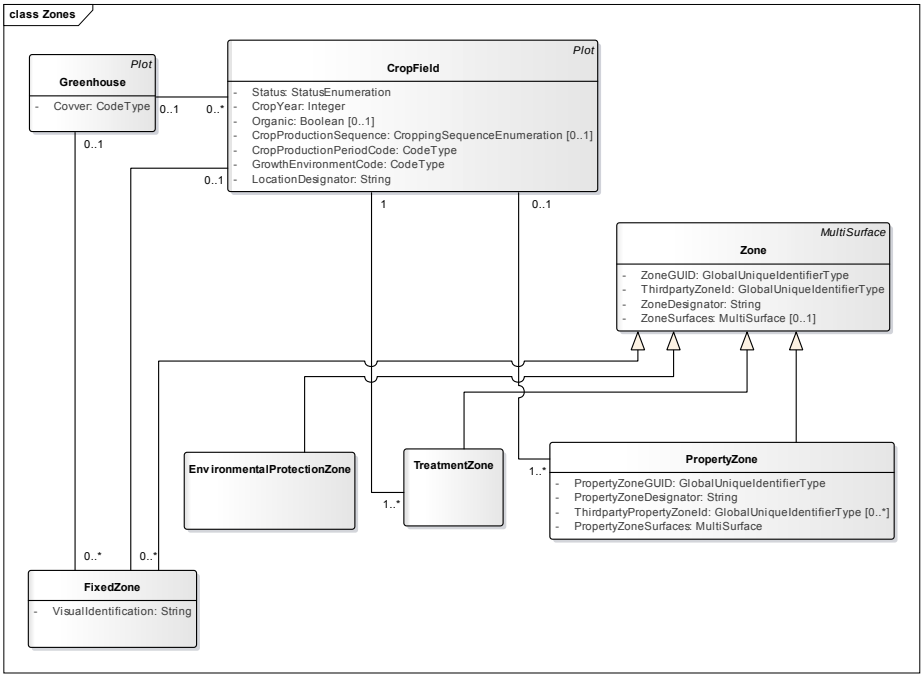
Figure 13. Products and Produce.

9.11

9.12 Values and Units

Will be worked out later

9.13 Zones



In a Greenhous it is possible to use a number of fixed zones to specify the surfaces of a CropField (or of a TreatmentZone. Ps the association is to be added)

10 Appendix II. Some use cases.

10.1 Potato chain.

10.1.1 From harvest onwards.

This part of the use case description starts at the moment of harvesting.