





Project number:	693729
Project acronym:	smarticipate
Project title:	smart open data services and impact assessment for open governance
Instrument:	Horizon2020
Call identifier:	H2020-INSO-2015-CNECT
Activity code:	INSO-1-2015

Start date of Project:	2016-02-01
Duration:	36 month

Deliverable reference number and title (as in Annex 1):	D5.2 Interface and Toolkit Redbook
Due date of deliverable (as in Annex 1):	Month 28
Actual submission date:	<i>see "History" Table below</i>
Revision:	1

Organisation name of lead contractor for this deliverable:
Fraunhofer IGD

Project funded by the European Commission, Horizon2020, topic INSO-1-2015			  European Commission Information Society and Media
Dissemination Level			
PU	Public	x	
PP	Restricted to other programme participants (including the Commission Services)		
RE	Restricted to a group specified by the consortium		
CO	Confidential, only for members of the consortium (including the Commission Services)		

Title:
Interface and toolkit redbook
Author(s)/Organisation(s):
Jens Dambruch / IGD
Working Group:
WP5
References:
D3.2, D3.3, D3.4

Short Description:
<p>This document is the description of the smarticipate toolkit from a technical user`s perspective. The application programming interfaces (API) for smarticipate linked-data backend services, service interfaces to extract data, data storage, interactive, automated feedback and how they are implemented in the smarticipate project are topics for this document. However, at present only documentation on the feedback service is included in this version. Also in scope are possible recommendations for refining OGC standards based on our experiences and especially after evaluation results are available.</p>
Keywords:
Automated feedback, linked-data, inference, geo-data storage, Application programming interface description

History:				
Version	Author(s)	Status	Comment	Date
001	Jens Dambruch	rfc	Initial draft of 1 st intermediate version for month 12	22.01.2016
002	Jens Dambruch	Final	Rework after review	06.02.2017

Review:			
Version	Reviewer	Comment	Date
001	Zaheer Khan	See supplied review forms in intranet	03.02.2017



About this Document	4
1 Introduction	4
2 Formal Language Design	4
3 Feedback Service Tree Planting	6
3.1 Analysis	6
3.2 Design	7
3.3 Formal Language Definition	9
4 Feedback Service Recovery Planning	9
5 Outlook	9
6 References	10

About this Document

This document is the description of the smarticipate toolkit from a technical user's perspective. The application programming interfaces (API) for smarticipate services defined in D3.3 chapter 2.1.1 such as linked-data backend services, service interfaces to extract data, data storage, interactive, automated feedback and how they are implemented in the **smarticipate** project are topics for this document. However, at present only documentation on the feedback service is included in this version. Also in scope are possible recommendations for refining OGC standards based on our experiences and especially after evaluation results are available. A glossary of terms is available in D3.3. The adopted methodology of developing software is detailed in D3.4.

Please note that this version of the document is the preliminary revision 1 and the current state of progress is given as of month 12 in the project, where formal design and development has just started. A final version will be available in month 28.

1 Introduction

A general introduction to the project is available in D2.1 and D3.2. In the following chapters the results of the methodology introduced in D3.2 chapter 2 are documented regarding automated feedback services.

These are structured as follows:

1. Domain Analysis results with Terminology and Domain model (steps 3 and 4 as of D3.2 chapter 2)
2. Design of the language implementation backend if needed (step 5 as of D3.2 chapter 2)
3. A formal grammar description with examples (step 7 as of D3.2 chapter 2)

With regard to the techniques outlined in chapter 3 of D3.2, we analysed the input from city stakeholders to identify concepts, actions and attributes in order to derive a suitable domain model. This is the basis for the definition of a Domain-Specific Language on a user's level. The first step in analysis is to identify the terminology and the concepts of the domain. This is documented in the chapters dealing with use cases identified in D3.2. The focus in later version will be the Domain-Specific Language for those use cases and special attention is given for the language design.

2 Formal Language Design

There are several options for a concrete language syntax, ranging from a more mathematical notation to pseudo natural language constructs on the other side. Though there are many programming language paradigms available and also mixes, there are 4 prominent basic paradigms, as explained in an overview from Nørmark (2014) and in comprehensive detail from Van Roy and Haridi (2004):

- *Imperative* – a sequence of commands leading to state changes, the classic approach to programming

- *Functional* – based on the mathematical concept of functions
- *Logic* – based on mathematical logic with axioms and rules
- *Object-Oriented* – Very “natural” modelling of concepts by grouping into classes of objects and encapsulating them via message passing

At present most geographic information systems allow for imperative programming, for example ESRI ArcGIS and QGIS use Python and embed it in their application. According to van Roy (2009) a practical programming language needs to involve various concepts to be tailored well for the intended purpose. The concepts relevant have to be in principle identified and implemented for each use case. We think that commonalities between languages can be identified, based on concrete use cases and therefore it may be possible to develop families of languages suited for a specific topic. This means that languages may be similar in structure and semantics and only differ in concrete tokens or words for the same operation or concept. As a consequence, we assume that the reusability of basic language elements will be considerably high.

Generally, we think that a declarative approach is very useful as it defines what should be the outcome and not how to compute it step by step. In the next version of this document an analysis of Languages, paradigms and concepts based on the taxonomy from van Roy (2009) will highlight the relevant ones for smarticipate use cases.

For now, as an example the rule HH-12 from D3.2 in Table 2:

“Distance to Street Lights: Trees grow and possibly will mask street lights nearby. A minimum distance should be kept from such positions. Positions of street lights need to be given.”

can be given as the following rule (or formula) inspired by predicate logic:

$$Tree(x) \wedge \neg \exists Lamp(y) \wedge distance(x, y) \leq 8$$

This means for a given tree there is no lamp nearer than 8 meters. If this evaluates to true, then a tree can be placed at x' position. It can be rephrased to be readable for non-mathematicians and some adjustments by exchanging symbols to:

$$X \text{ of Tree and not exists } Y \text{ of Lamp and distance}(X, Y) \text{ less than or equals } 8m$$

This term will be evaluated by substituting the variables (e.g. X and Y) and finally will be true or false. True means there isn't any lamp nearer than 8 meters to tree X. This assumes that distance is a function of two object positions returning a metric distance value. Keywords found: *exists* and *distance* as domain elements, *function* for distance on a computational level.

What should happen if the evaluation leads to false? In principle it would be sufficient to return the result along with a description. Another option would be to trigger another piece of code carrying out an action of choice like this:

When X of Tree and not exists Y of Lamp and distance(X, Y) greater than or equals 8m then return failure description

In general, this can be modelled as simple rule like:

When <Term> then <Action>

The actions in this development phase may be simplistic as returning direct results as numbers and text; 2nd cycle will extend this to returning graphical results. Further aspects for the next version includes the following aspects:

- In any case the results computed should be traceable, e.g. which data was used and which rules applied?
- Results are mandated by clients and should be provided as they need it
- Results are functions of inputs and state
- Outline of steps necessary to compute the example
- Spatial query for not exits
- Additional language elements needed, but not explicitly mentioned in use cases (Table 1), e.g. numbers or spatial analysis functions

Operators / Datatype	Description
Distance	Euclidian distance between points
Spatial relation	Intersection, inclusion of geometries...
Bounding Box	Simple hull of an object
General relation	Is, less than, greater than, not operators
Real number	Usual operators on real numbers
String	Usual operators on character strings

Table 1: Additional elements for technical reasons defined in D3.2, Table 6

3 Feedback Service Tree Planting

For definitions of requirements please see D3.2 chapter 3.2.

3.1 Analysis

The analysis is performed using Noun-Verb analysis technique. A thorough analysis of the use case and related documents provided a list of nouns, verbs and properties which are used to define basic concepts and possible actions. This starting list is presented in Table 1, Table 2 and Table 3.

Table 2 shows the nouns, representing concepts in the domain.

Proposal	Cost	Goal	Power Lines
Infrastructure	Water	Gas	Communication
Private Land	Land use	Planned Actions	Tree
Species	Neighbourhood	Building	Street Lighting
Traffic Signs	Flooding area	Condition of soil	Shadow and Light
Sidewalk	Street Access	Bus Lane	

Table 2: Nouns representing concepts

These are the basic things to be considered for giving feedback. The next step is to identify possible actions, which are typically represented by verbs or verb-derived nouns. This is shown in Table 3.

Agree/disagree	Calculate costs	Link	Exclude / include
Measure distance	Determine species	Define species	Growth simulation
Flooding simulation	Intersection		

Table 3: Words representing actions

And finally some attributes or properties related to use case and requirements are shown in Table 4:

Estimate	Suitability	Specific	Under ground
Owned land	Usage	Species definition	Age
Health State			

Table 4: Attributes

In the above description and analysis, no actors or persons in specific roles are mentioned, and more importantly no indicators for actions or a process which changes data. This supports our assumption that feedback service does not change data at all and is merely used for analytic purposes.

Also, very few attributes/properties were given, which indicates that for the time being, almost no available real data is considered by users. It is obvious that all concepts identified here need to be backed with available data and models to implement feedback feature and give reasonably useful feedback.

3.2 Design

On the basis of D3.2 we designed a domain model for the use case shown in Figure 1.

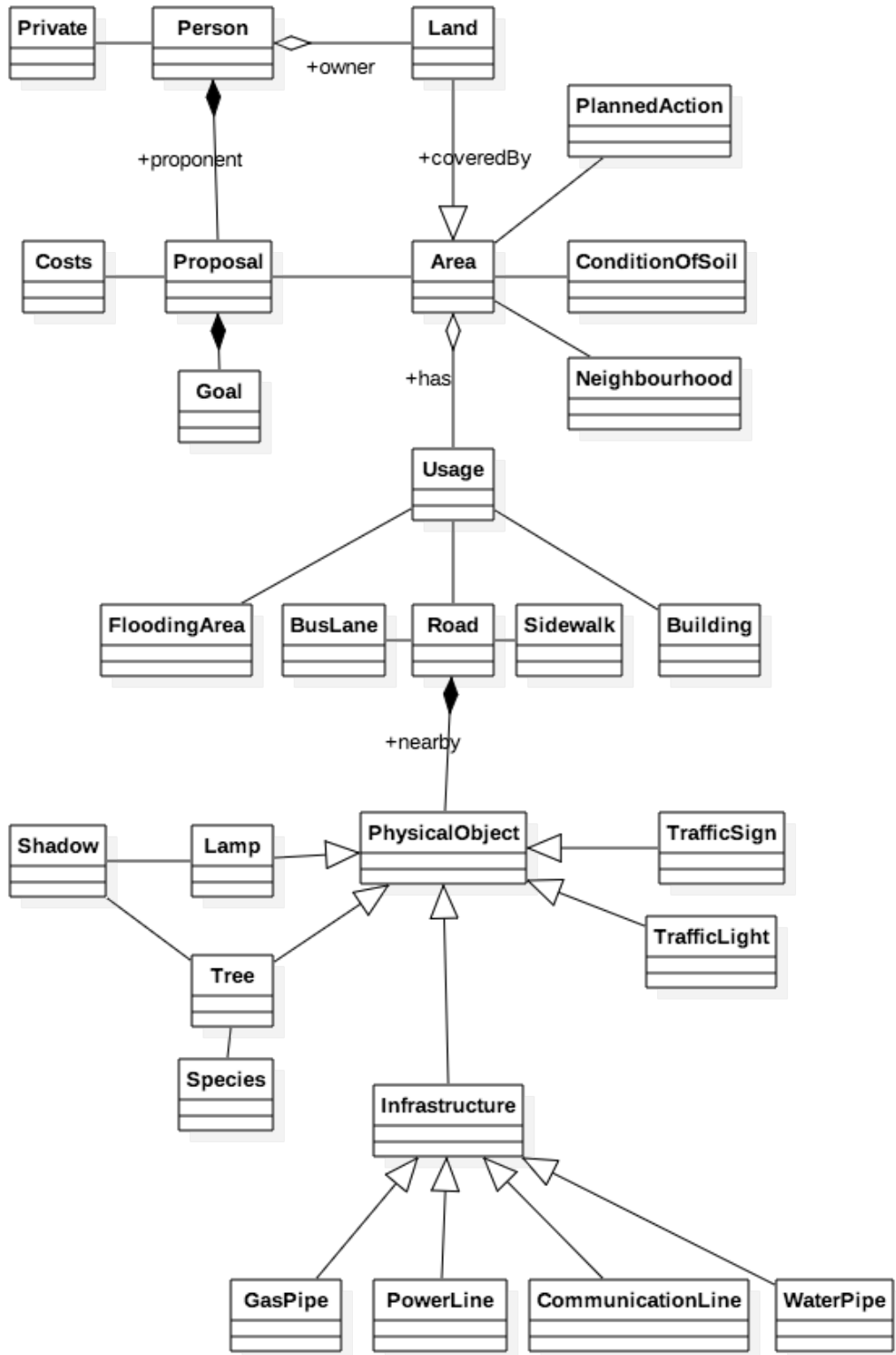


Figure 1: Domain model for use case tree planting

The purpose of the domain model is to show which concepts (and their relationships) are included and can be handled by the feedback feature service. This means that the feedback service will rely on the richness of this domain model i.e. more concepts (and their relationships) will allow to handle more objects/concepts as part of feedback analytics. In the model several nodes have been introduced due to generalization which have not been derived from the terminology. *Physical Object* allows to group things collocated to a certain object such as roads. It will be investigated in next months that whether or not such a semantic link is really needed as such relations can also be derived by the topology given by spatial references. Nevertheless, such relationships might be useful in specific use cases, which has to be investigated further. For example, buildings in a street have a distance between them, are located in a specific direction (north, east, west, south of...) and may be at different ground levels. This alone has very little impact, but combined with other data such as range of shadows cast by buildings or street accessibility this can be valuable information.

Wherever enumerations are used also a generalization relation was used as for *Infrastructure*. Regarding such intangible relationships, it needs to be considered if such relationships should be introduced in the language itself, if there is no clearly visible need for it. A generic “linking” mechanism might be more adequate and also better to maintain from a language engineering point of view.

3.3 Formal Language Definition

This topic is still in work.

4 Feedback Service Recovery Planning

This topic regarding Rome use cases is still in work. Aspects to be covered include Analysis, Design and Formal Language Definition.

5 Outlook

In January 2017 a workshop commenced in Rome, analysing potential domains for automated feedback computation. The same methodology as used in Hamburg will be applied to define a DSL - if possible - and implemented afterwards. In summary the following can be expected from the next version:

- Selection of concepts and paradigms suitable for use cases and how to match them
- Language examples and execution plans for them
- Definition of a formal grammar for languages for use cases mentioned above
- Definition of execution semantics on the implemented execution environment
- Draft of visualization language to display results

6 References

Van Roy, Peter, Haridi, Seif (2004): Concepts, Techniques, and Models of Computer Programming. Massachusetts Institute of Technology

Van Roy, Peter (2009): Programming Paradigms for Dummies: What every Programmer Should Know. <http://www.info.ucl.ac.be/~pvr/VanRoyChapter.pdf>

Nørmark, Kurt (2014): Functional Programming in Scheme, Aalborg University, Denmark. http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html