Abstract

This document contains the user requirements for the *DiTriS* project. These requirements were established according to requests formulated by *Venturespring*. The document complies with the User Requirements Document (URD) from the Software Engineering Standard, as set by the European Space Agency [12].

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to specify the requirements of *DiTriS* in a clear and consistent manner. The user requirements document contains the requirements for *DiTriS*. These requirements are a negotiated agreement between *Venturespring* and software engineering project (SEP) group *Somebody Else's Project* (Spring 2012). Listed requirements, and only these, will be implemented in *DiTriS*, according to their priorities. Any changes to these requirements, require the full consent of both parties. This document is intended for *Venturespring*, *DiTriS*, and the senior management of SEP, to get a clear understanding of what product is to be made.

1.2 Scope

The software system *DiTriS* implements a web service that provides functionalities to retrieve, add, and delete triples distributed over many locations. Trusted parties may use *DiTriS* to retrieve information from these distributed sets of triples. Each *DiTriS* triple store offers multiple APIs to make use of the provided functionalities and is able to communicate with other triple stores.

DiTriS is intended to improve on IGLO, a triple store designed by K. van Overveld [14]. This is accomplished by making the system distributed, and enabling triples to store any kind of information (that is either URI addressable or text).

DiTriS builds further on the RDF data model [4] by extending RDF triples with information regarding origin - by means of a named graph label. By origin, we mean for example the identity of a user inserting the triple via a web application, or the identity of a user that makes a copy of a legally obtained triple, or the identity of a web application creating a triple based on other legally obtained triples. Because it is the web application that is responsible for adding this information and enforcing its implications to its users, *DiTriS* will only occupy itself with the verification of trusted parties. That is, all other authorization issues are not part of the *DiTriS* scope.

1.3 List of definitions

In this section definitions and abbreviations are listed and explained.

1.3.1 Abbreviations

RFID	Radio Frequency Identification
W3C	World Wide Web Consortium
SPARQL	SPARQL Protocol And RDF Query Language
Admin	Administrator
IGLO	Interface to Generic Local Ontology
URI	Uniform Resource Identifier [2]
IRI	Internationalized Resource Identifier [2]
URL	Uniform Resource Locator [2]
XML	eXtensible Markup Language
URD	User Requirements Document
RDF	Resource Description Framework
API	Application Programming Interface
DTS	*DiTriS* Triple Store

1.3.2 Definitions

RDF triple	A triple as defined by the RDF standard [5].
DiTriS triple	An RDF triple, extended with a named graph label.
Triple store	An application storing RDF triples and providing a SPARQL interface.
DiTriS triple store (DTS)	An application storing *DiTriS* triples locally, and optionally containing references to any collection of RDF triples from any DTS.
DiTriS system	A collection of connected *DiTriS* triple stores.
DiTriS API	An interface providing limited access to a DTS.
Resource_1	A resource as defined by the W3C standard.
Resource_2	An object, on which an action could be performed.
Action	An activity on a resource_2 in the *DiTriS* system to accomplish an objective.
Ownership	The exclusive rights of a person or application to copy, and remove his triples. The owner decides what should happen to his triple collection, and who has access to them.
Permission	Allowance for an actor to perform an action on a resource_2.
(Access) Right	A certain permission.
Account	An entity for which a user can authenticate itself. Every account has a specific role.
System Event	A software event accessible by the system administrator.
Anonymous web application	A web application that is not trusted by the administrator of the corresponding DTS.
Trusted web application	A web application trusted by the corresponding DTS administrator.
Trusted party	A trusted web application or a trusted DTS acting on behalf of a trusted web application.
DTS_X administrator	An administrator of DTS_X, where X represents an identifier. This notation is introduced to distinguish between multiple DTSs, for example DTS_1 or DTS_{42}.
DTS administrator	User of a DTS admin web application.
Admin web application	A web application trusted by a DTS to use the admin API of the corresponding DTS.
Admin API	A programming interface providing access to the actions of an admin role of a DTS.
Named graph	A collection of RDF triples identified by a URI as defined by the W3C standard. Optionally associated with authorization and ownership data.
DiTriS API	A programming interface allowing for delegation of (parts of) SPARQL/IGLO API requests.
SPARQL/IGLO API	A programming interface as defined by the W3C SPARQL standard and IGLO documentation, respectively.
Certificate Authority	An external entity validating certificates, and providing a list of validated certificates.
Node	An object or subject in a RDF graph as defined by the RDF standard [5].

To get a good understanding of the introduced definitions we now present a short description of the relations between the different definitions.

Given a collection of triple stores (both *DiTriS* triple stores and other implementations), *DiTriS* enables these triple stores to communicate with each other. *DiTriS* also provides a web interface per *DiTriS* triple store. This interface enables a DTS administrator to add, delete, and retrieve triples. In addition it enables the administrator to see the structure of the triple store. Furthermore, *DiTriS* provides a APIs that enable trusted parties to interact with the *DiTriS* system.

A user issues a command to a web application, which generates queries. The application, in turn, can forward those queries to the *DiTriS* system. The *DiTriS* system returns the results to the web application, after which the application will visualize the triples to the user.

1.4 References

[1] Cross-Site Scripting. `http://en.wikipedia.org/wiki/Cross-site_scripting`.

[2] Naming and Addressing: URIs, URLs, ... `http://www.w3.org/Addressing/`.

[3] Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. `http://wwwconference.org/www2008/papers/pdf/p585-schenkA4.pdf`.

[4] RDF Primer. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[5] Resource Description Framework (RDF). `http://www.w3.org/RDF/`.

[6] W3C Standard: Hypertext Transfer Protocol. `http://www.w3.org/Protocols/`.

[7] W3C Standard: RDF SPA RQL XML conversion. `http://www.w3.org/TR/rdf-sparql-XMLres/`.

[8] W3C Standard: RDF SPARQL 1.1 Query. `http://www.w3.org/TR/sparql11-query/`.

[9] W3C Standard: RDF SPARQL Protocol. `http://www.w3.org/TR/rdf-sparql-protocol/`.

[10] W3C Standard: SPARQL 1.1 Update. `http://www.w3.org/TR/sparql11-update/`.

[11] Dai Clegg and Richard Barker. *Case Method Fast-Track: A Rad Approach.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[12] ESA. *ESA Software Engineering Standards.* Mar 1995.

[13] R.T. Fielding. *Architectural styles and the design of network-based software architectures.* Phd thesis, University of California, Irvine, 2000.

[14] K. van Overveld. Iglo system. `http://keesvanoverveld.com/index.php?page=iglo`.

1.5 Overview

This document is structured according to the standard described in [12].

Chapter 2 provides a general description of the *DiTriS* system. It gives an overview and describes the capabilities of *DiTriS*. Furthermore, the characteristics of the users are also considered.

Chapter 3 describes a set of user stories. Chapter 4 describes specific requirements for *DiTriS*.

Chapter 2

General Description

2.1 Product perspective

DiTriS is a software package enabling software applications to store and query triple collections that are distributed over the Internet. The goal of *DiTriS* is to provide applications with a general, but intuitive interface to share, and combine, knowledge spread throughout the world. *DiTriS* differs mainly from existing systems (e.g. *IGLO*) because it is a distributed system. Moreover, data ownership is based on "Own your own data".

2.2 General capabilities

Nowadays, there are many ways in which you want to offer your data for re-use by other parties. *DiTriS* offers functionalities to support distributed triple storage. The system consists of three connected parts: the DTSs, the APIs, and the Admin web application.

First, the *DiTriS* system provides functionality to web applications to store their data in a DTS. The data stored in a DTS can be shared between and accessed by any trusted party. This sharing of triples is taken care of by the API described below.

Second, the DTS has an Admin web application that allows a system administrator to add or delete triples and perform other administrative tasks. The DTS enforces a set of rules that specifies which parties are trusted to request, add, and delete certain triples.

Third, *DiTriS* provides multiple APIs. The Admin API links *DiTriS* to the Admin web application as discussed above. The SPARQL/IGLO API provides a direct interface to insert or output SPARQL/IGLO queries.

Finally, the *DiTriS* API handles the internal communication. Here queries are translated to internal requests to collect the data from different DTSs.

An overview of *DiTriS* is shown in Figure 2.1.

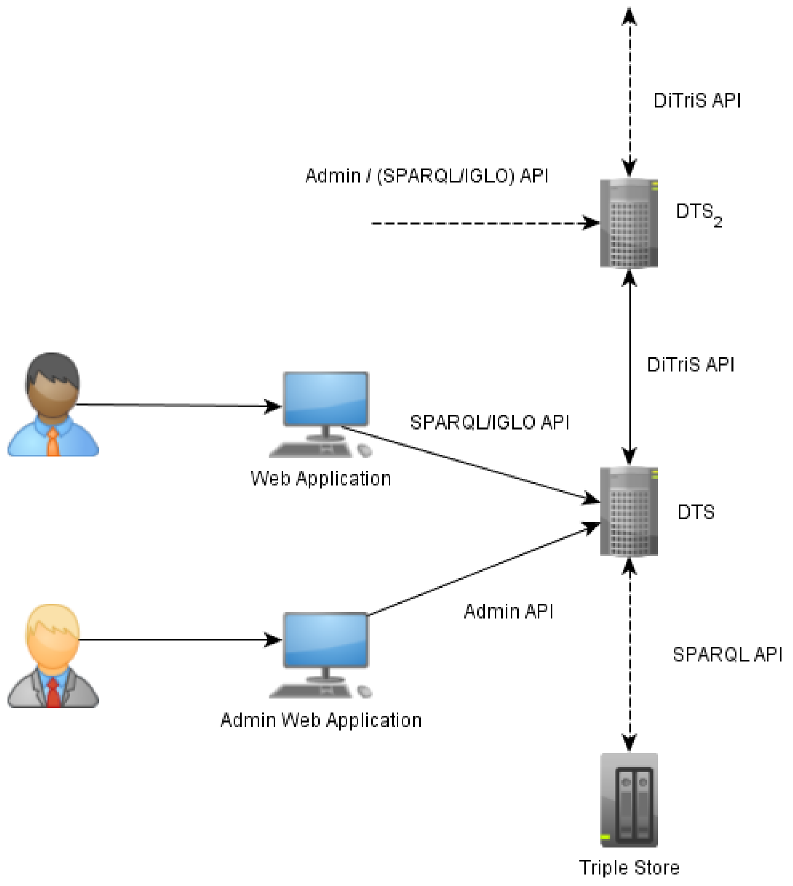

Figure 2.1: Environment Diagram.

2.3 General constraints

The system is required to adhere to several W3C standards [8, 9, 7]: data is modeled into RDF graphs which are queried using the SPARQL query language.

2.4 User characteristics

In this section we describe the characteristics of different user roles. The different actors using the *DiTriS* system are given below.

- User: a person using a web application which uses the *DiTriS* system.
- Web application: a program providing access to the *DiTriS* system by interfacing with triple stores, directly or indirectly.
- DTS administrator: a person or program with administrator rights to a DTS.

Relations between the different users and other components of *DiTriS* are visualized in Figure 2.2.

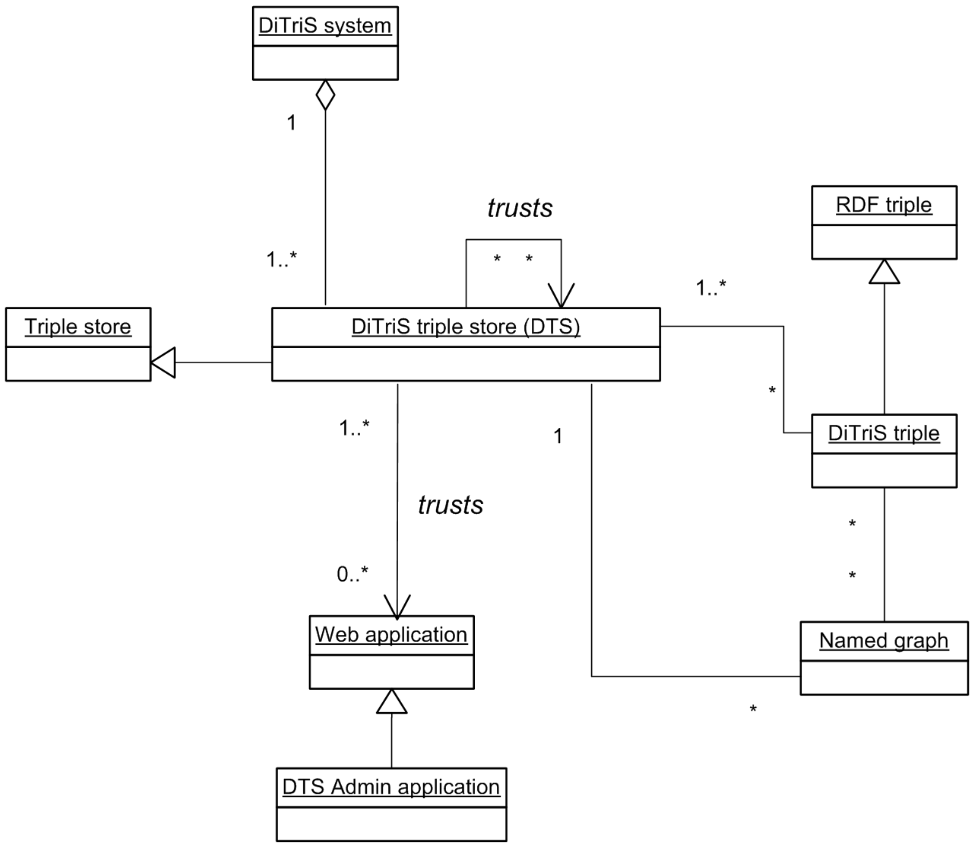

Figure 2.2: UML domain model for the *DiTriS* system.

2.5 Environment description

In this section, we will look at the environment in which *DiTriS* is expected to operate. Initially, its main area of employment will be within the cultural domain. This is a very large and varied domain. Possible applications include RFID tags that register a person's actions and extract data from a set of triple stores such that personal recommendations can be given, and an application that makes it possible to explore a network of associations between images. In this sense, it is not the end user, but rather the web application that uses *DiTriS*.

DiTriS will be distributed over many systems and is expected to be built on top of a database, using a standardized scripting language to implement features such as the web interface.

Due to the generic character of *DiTriS*, it is very hard to predict on which systems it will be employed. For this reason, *DiTriS* possessive implementation should only depend on common tools and frameworks.

2.6 Applications

The software can, for instance, be used for data acquisition during cultural events. Participants can query data that is somehow related to these events, thereby gaining more insight/knowledge about anything that is related through the triple stores.

The distribution of these triple stores enables these cultural events to gather and share triples with other organizations. Since we introduce a distributed system, the amount of redundant data can be decreased as data stored on the triple stores can be shared between multiple organizations. Hence, through specialization, certain companies can deliver a very specific subset of data. The union of all these sets is the data available in the system. Especially in large scale events such as "STRP", participants can obtain information about anything that is present at such an event.

For commercial purposes this is also interesting, since users can obtain recommendations for several activities inside an event. This way, the application user can be introduced to new activities that he might be interested in. Note that for the ownership data of each triple, the web application using the services of the *DiTriS* system is fully responsible.

2.7 Assumptions and dependencies

AD_001	The *DiTriS* system in general, and a DTS in particular, is not intended to store privacy sensitive information.
AD_002	A DTS does not provide any guarantees that *all* triples satisfying the request will be returned.
AD_003	(External) triple stores can accept and execute SPARQL requests and return XML responses in standard format [7].
AD_004	Trusted web applications are responsible for providing and processing ownership information of *DiTriS* triples using named graphs.
AD_005	It is up to the administrator to determine which parties are trusted.
AD_006	A DTS will accept and process any request from any trusted party.
AD_007	An administrator is a user.
AD_008	A query received by a trusted party is assumed to be safe from malicious contents, such as code injections.

Chapter 4

User Requirements

In this chapter, we describe the user requirements as established by a number of interviews between *Kees van Overveld and Eelco Dijkstra* and *Somebody Else's Project*. These requirements are subdivided into four different priority levels, according to the MoSCoW [11] principle: Must (top priority), Should (highly desirable), Could (if time allows), and Won't (not today).

4.1 Capability Requirements

4.1.1 General Requirements

UR_001	A DTS conforms to the W3C RDF standard in storing RDF triples. See also section 4.3. Priority: Must

Trusted parties

UR_010	Messages between web applications and a DTS use the Hypertext Transfer Protocol [6]. Priority: Must
UR_011	Only a trusted party is authorized to execute queries on a DTS. Priority: Must
UR_012	A request from a trusted party can imply requests of triples to external triple stores. Priority: Could
UR_013	A request from a trusted party can imply requests of triples to other DTSs. Priority: Must

DiTriS **Triple Store**

UR_022	A DTS is able to query another DTS on behalf of a requesting trusted party. Priority: Must
UR_023	A DTS is able to query an external triple store on behalf of a requesting trusted party. Priority: Could
UR_024	A DTS makes a backup of deleted triples. Priority: Could
UR_028	A DTS can execute IGLO queries. Priority: Should

SPARQL

UR_035	A DTS can execute at least a subset of SPARQL queries. See 4.3 Priority: Must

Authorization

UR_040	Queries originating from a trusted party are always authorized to be executed Priority: Must

Administrator web application

UR_050	The administrator web application can support multiple administrator accounts. Priority: Could
UR_051	DTS administrators can be managed in the administrator web application. Priority: Could
UR_057	A DTS administrator is able to manage the trusted parties of that DTS. Priority: Could
UR_060	A DTS administrator is able to delete a triple. Priority: Could
UR_061	A DTS administrator is able to add a triple. Priority: Could
UR_063	A DTS administrator logs in with his password and username. Priority: Could
UR_065	A DTS administrator is able to change his or her password. Priority: Could
UR_021	A DTS administrator can view a log of system events of the respective DTS. Priority: Could

4.2 Constraint Requirements

4.2.1 General

UR_100	All triples within a named graph are stored in a single DTS. See also 4.3. Priority: Must
UR_004	After a triple is deleted from a DTS it cannot be retrieved by means of a query. Priority: Should
UR_160	Any data processed by a *DiTriS* system should preserve the data's integrity. Priority: Must

4.2.2 Interfaces

UR_110	The admin web application is menu driven. It will provide dialog boxes, radio buttons, help screens, and forms for user inputs. Priority: Could

4.2.3 Portability

UR_120	A DTS runs on PHP and MySQL. Priority: Must

4.2.4 Adaptability

UR_131	Documentation and APIs are in English. Priority: Must
UR_132	Documentation and APIs are in Dutch. Priority: Could

4.2.5 Security / Safety

UR_141	A DTS provides security mechanisms against cross site scripting attacks [1]. Priority: Could

4.2.6 Standards

UR_151	*DiTriS* systems are compliant with the W3C SPARQL Protocol for RDF [9]. Priority: Must
UR_152	*DiTriS* systems are compliant with the W3C SPARQL Query Results XML Format [7]. Priority: Must
UR_153	*DiTriS* adheres to the REST architectural style [13]. Priority: Should

4.2.7 Performance

UR_161	Performance of SPARQL queries on *DiTriS* conforms to performance of standard algorithms. Priority: Should
UR_162	The performance of SPARQL queries on DTSs is improved compared to standard algorithms. Priority: Could