Unified Service Description Language (USDL) Foundation

December 28, 2009

Edited by SAP Research

Abstract. This document describes what is called the Foundation of the third version of the Unified Service Description Language (USDL). USDL was developed as a holistic approach to describe entities provisioned into service networks, which considers and connects business, operational (functional) and technical aspects of service description. The Foundation covers common concepts that are reused throughout other parts of USDL.

Table of Contents

1 Introduction

As outlined in the central document of this series *"USDL Technical Overview Paper"*, services are becoming the backbone for electronic commerce. Especially the trend to provision IT-based services outside company "firewalls" with the help of intermediaries is on the increase, as it allows organizations to take new opportunities relatively quickly. In this context services are seen as tradable entities that constitute a well defined, encapsulated, reusable and business-aligned set of capabilities. The term business service is used for such services, in order to distinguish them from other types, e.g., those that are provided in a service-oriented IT infrastructure within an organization.

The Unified Service Description Language (USDL) defines a way to describe services from a business and operational point of view and align this with the technical perspective. While the latter is captured quite well by existing service description languages, USDL explicitly enables to express business characteristics set by an organization for the purpose of providing means for consumers to invoke and use business services and for intermediaries to (re)use and repurpose services. A detailed explanation of the scope and objectives of USDL is given in *"USDL Technical Overview Paper"*.

USDL on a whole is made up of a set of modules, each addressing different aspects of the overall service description. Modularization was introduced to improve readability of the model, which drastically grew in size compared to its predecessor. The modules have dependencies among each other (shown in Figure 1), as they may reuse concepts from other modules. Currently, there are 8 modules in the set that constitutes USDL version 3.0.

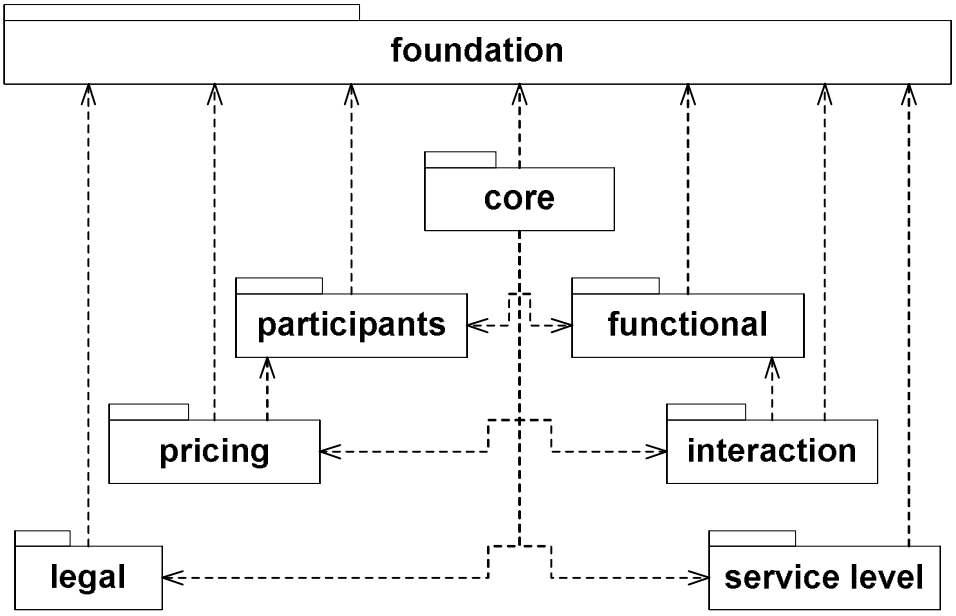

Figure 1 Packages comprising the USDL model and their dependencies (represented as arrows)

1.1 About this document

The USDL meta-model is formally defined in Ecore (the meta-modeling language of EMF), with each USDL module being captured in a separate package. This document is one in a series of USDL documents and covers the Core module defined in package “core”. The series also includes:

- *USDL Technical Overview Paper*
- Module-specific documentation of the modules *Core*, *Participants*, *Functional*, *Interaction*, *Pricing*, *Service Level* (includes geographical and temporal availability) and *Legal*

The document only provides insights into the concepts of the Foundation. For a complete overview of USDL it is necessary to go through all documents of the series.

1.2 Acknowledgements

Work on USDL has been mainly carried out in the context of the THESEUS TEXO– a project in the frame of the THESEUS Lighthouse research program initiated by the German Ministry of Information and Technology.

Several European, German and Australian research projects also contributed to the development of USDL. Naturally, there is an extensive number of people that have contributed to conceptualization and documentation of USDL either directly or through feedback. Rather than giving the full list of individuals, it shall suffice to name the institutions they work for. The full list is available in the *USDL Technical Overview Paper* and on http://www.internet-of-services.com.

SAP Research

European Research Center for Information Systems (ERCIS), University of Muenster

Queensland University of Technology, Brisbane, Australia

University of New South Wales, Sydney, Australia

2 USDL Foundation

The foundation contains generic concepts that are reused in different modules such as the artifact concept, which represents an external entity that can be linked to a USDL elements (concept) such as a WSDL document (Functional module), a file defining the terms of use of the service (Legal module), or a piece of documentation for the service (Core module). Resource is another generic concept that is defined in the foundation and represents real-world objects of various types that either perform the service or upon which the service acts, e.g. an application, a system, a tool, an employee, etc.

Another important set of elements provides means to identify, name and generically describe other USDL elements. Unifying the latter two, the element description concept is probably the most widely re-used element of USDL. Its objective is to provide an advanced and fairly complete solution to describe the entities of the USDL as precisely as possible.

In addition, the foundation defines some concepts to define locations as this is needed in various modules for example to define the delivery and availability location of a certain service, or to define a specific price for certain locations, or to specify certain attributes of the users and organizations that have to do with the service.

Furthermore, the foundation defines some concepts to express time as this is also needed in various modules to express for instance quality of service and service level aspects or the availability time of the service or to define the validity period of prices.

2.1 Module Info

Parameters of the package that captures the module

- Namespace: *http://internet-of-services.com/usdl/foundation*
- Name: *foundation*

The remainder of this section describes the classes and enumerations that are part of the package. Figure 2 depicts a class diagram of the package.

Note: Example fragments are provided for some of the classes. In order to improve readability they are presented in XML-based pseudo syntax. This is **NOT** the official USDL syntax, which is still under development. However, there currently exists a serialization format that is XMI-based and supported through a USDL editor developed by SAP Research.

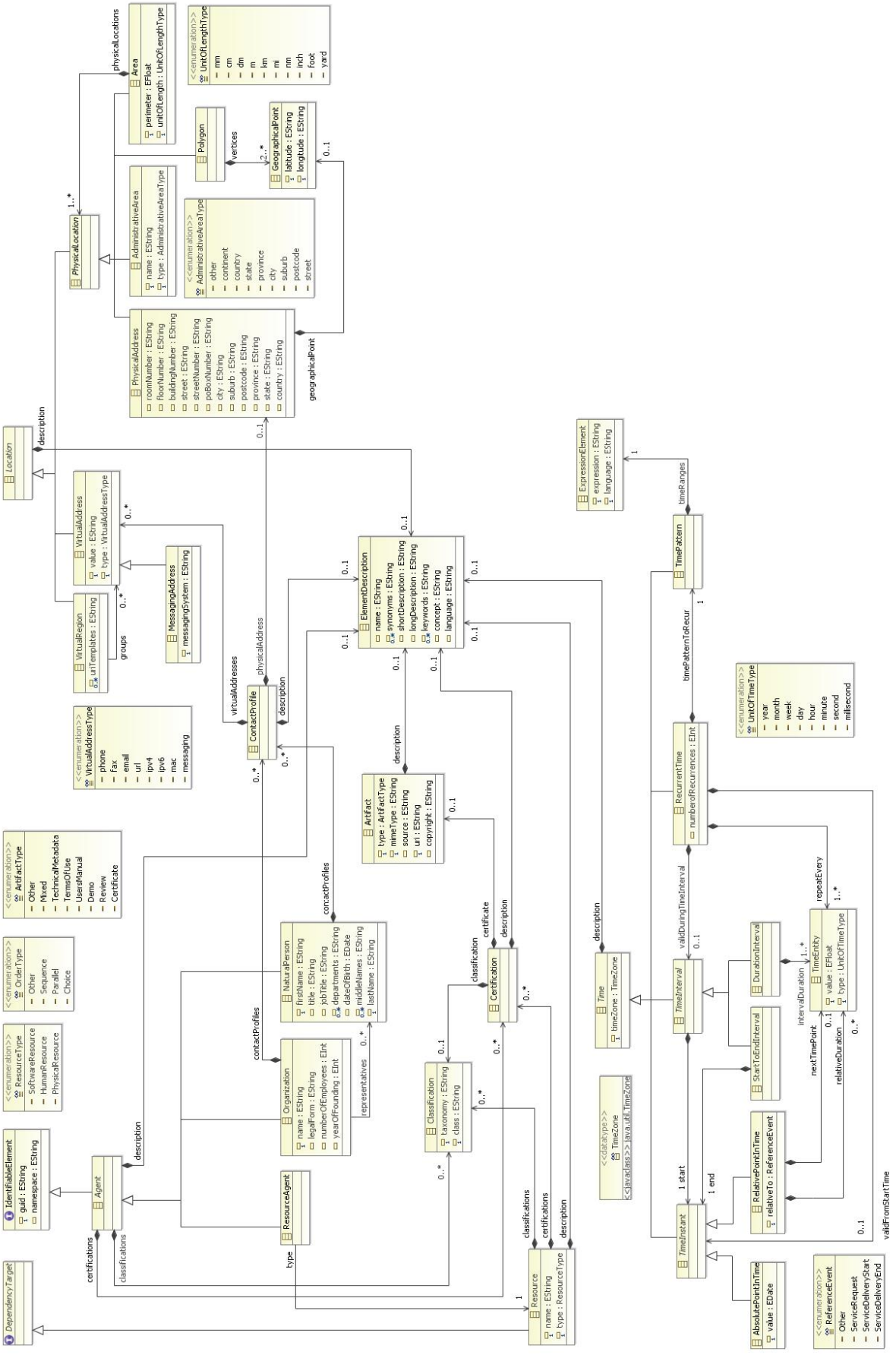

Figure 2 Class diagram of the package that captures the Foundation

3 General Foundation Elements

3.1 IdentifiableElement

IdentifiableElement serves as the super type of all elements/concepts of USDL that represent independent entities that can be uniquely identified, either globally or within a certain namespace.

- Ecore Type: Interface EClass
- Interfaces: N/A

IdentifiableElement			
Attributes			
Name	Type	Cardinality	Description
guid	EString	1	The identifier uniquely distinguishing a USDL element/concept, globally or in a certain namespace
namespace	EString	0..1	The identifier of a namespace, wherein *guid* is unique
Examples (in pseudo concrete syntax)			
<service> … <guid> P05529-RSM-DL006 </guid> <namespace> http://www.moonbank.com/BankingServicesNetwork/ <namespace> … </service>			

3.2 ElementDescription

ElementDescription is a generic class used to describe elements/concepts of USDL and offers a number of different information elements. Out of these we will only describe the keywords and concept elements since their purpose is not self-evident.

1. **Keywords**. Keywords provide a tagging mechanism for adding re-usable information to the information space. The use of keywords will allow applications to sort, cluster and query masses of services in a similar way that sites such as del.icio.us and flickr have been able to achieve.
2. **Concept**. Another approach to the use of keywords is to provide semantic web data with well-defined global ontologies. The description of services' properties can be achieved using well-defined identifiers. While this approach is more precise than using keywords, it is slower since users cannot use word off the top of their heads but must locate a specific ontology and select the correct concept. The attribute concept can be applied to ontologies but can also be used to refer to concepts presents in classification schema and taxonomies (such as industry standards).

- Ecore Type: EClass
- Interfaces: N/A

ElementDescription			
Attributes			
Name	Type	Cardinality	Description
name	EString	0..1	Adds a single name to a previously unnamed USDL element/concept

synonyms	EString	0..*	A list of alternative names of a USDL element/concept
shortDescription	EString	0..1	The short description of a USDL element/concept; ***max. 200 characters***
longDescription	EString	0..1	The extended description of a USDL element/concept
keywords	EString	0..*	A list of keywords associated with a USDL element/concept
concept	EString	0..1	Unique identifier that links a USDL element/concept to a concept in another language, in a taxonomy or in an ontology
language	EString	0..1	Language used for describing a USDL element/concept (ISO 639-1)

Examples (in pseudo concrete syntax)

```
<service>
…
 <description>
  <synonyms> RB-COMP-Term_Deposit </synonyms>
  <synonyms> P05529-RSM-DL006 </synonyms>
  <shortDescription>
   Term Deposit service provides single-point management of term deposit accounts in retail banking.
  </shortDescription>
  <keywords> Term Deposit </keywords>
  <keywords> Retail </keywords>
  <keywords> Banking </keywords>
  <keywords> Account Management </keywords>
  <language> ENG </language>
 </description>
…
</service>
```

3.3 Artifact

Artifact is a generic concept that allows including links to USDL-external service metadata, as well as arbitrary documents, files, web pages, etc.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

Artifact			
Attributes			
Name	Type	Cardinality	Description
type	ArtifactType	1	Type indicates the semantics of the content of the linked artifact
mimeType	EString	1	Mime type indicates the syntactic of the content of the linked artifact
source	EString	0..1	A description of the source of the artifact; ***examples:*** *service provider, reviewer Web site, blogs*
uri	EString	1	The universal resource identifier pointing to the artifact
copyright	EString	0..1	Information regarding copyrights of the artifact

Relations			
Name	Type	Cardinality	Description
description	Element Description	0..1	A description of the artifact

Examples (in pseudo concrete syntax)

```
<service>
…
 <additionalDocumentation>
  <type> TermsOfUse </type>
  <source> service provider </source>
  <mimeType> application/pdf </mimeType>
  <uri> http://www.moonbank.com/services/terms-of-use/general.pdf </uri>
  <description>
   <name> Moonbank Services General Terms of Use </name>
  </description>
 </additionalDocumentation>
…
</service>
```

3.4 ArtifactType

ArtifactType gives information about semantics of the content of an artifact.

- Ecore Type: EEnum

ArtifactType			
Items			
Name	Literal	Value	Description
Other	Other	0	The artifact contains data that cannot be associated with any semantic type
Mixed	Mixed	1	The artifact contains data that can be associated with a multiple semantic types
TechnicalMetadata	Technical Metadata	2	The artifact contains additional (technical) metadata extending the description of a USDL element/concept; *examples: service WSDL, service policy*
TermsOfUse	Terms Of Use	3	The artifact contains terms of use regulating the consumption of a concept (e.g. service, resource)
UsersManual	Users Manual	4	The artifact contains a user's manual explaining informally how a concept should be used or interacted with
Demo	Demo	5	The artifact contains a demo of a concept, how to use it, or how to interact with it
Review	Review	6	The artifact contains a review of a concept, e.g. a service, an agent involved in the delivery of the service, etc.
Certificate	Certificate	7	The artifact contains a certificate associated with a concept, e.g. a service, an agent involved in the delivery of the service, etc.

3.5 DependencyTarget

DependencyTarget serves as the super type of all elements/concepts of USDL that can be the target of a dependency (**Resource**, **NetworkProvision**, **AbstractService**).

- Ecore Type: Interface EClass
- Interfaces: N/A

3.6 Resource

Resource is a generic concept to represent real-world objects of various types, e.g. an application, a system, a tool used to perform a service, or an object a service is performed on.

- Ecore Type: EClass
- Interfaces: DependencyTarget
- Superclass: N/A

Resource			
Attributes			
Name	Type	Cardinality	Description
name	EString	1	The name of the resource
type	ResourceType	1	Type denotes the nature of a resource
Relations			
Name	Type	Cardinality	Description
classifications	Classification	0..*	The set of classifications into taxonomies associated with the resource
certifications	Certification	0..*	Certifications held by the resource; ***example:*** *an application is certified to run on SAP NetWeaver 7.20*
Examples (in pseudo concrete syntax)			

3.7 ResourceType

ResourceType gives information about the general nature of a resource.

- Ecore Type: EEnum

ResourceType			
Items			
Name	Literal	Value	Description
SoftwareResource	Software Resource	0	The resource is implemented in software; ***examples:*** *an operating system, a project management application*
HumanResource	Human Resource	1	The resource is a person or a group of people; ***examples:*** *customer service representative, call center agent*
PhysicalResource	Physical Resource	2	The resource is of general physical nature and does not fall in one of the other two categories; ***examples:*** *truck, shovel, driver's license*

3.8 Agent

Agent serves as the abstract super type of all entities that can participate in the delivery of a network provisioned entity (service or service bundle). An agent can be a natural person, an organization represented by a natural person, or a resource (e.g. a system or application).

- Ecore Type: Abstract EClass
- Interfaces: IdentifiableElement
- Superclass: N/A

Agent			
Relations			
Name	Type	Cardinality	Description
classification	Classification	0..*	A set of classifications that characterize the agent; ***example:*** *a hair dresser in the role of a service provider may indicate membership in a hair dresser's guild in order to increase its credibility as a provider of professional services related to hair dressing*
certifications	Certification	0..*	Certifications held by the agent; ***example:*** *an organization is ISO 9001 certified, a person is a graduated M.D.*
description	Element Description	0..1	A description of the agent

3.9 Organization

Organization represents unique organizational legal entities that exist in the real world.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Agent

Organization			
Attributes			
Name	Type	Cardinality	Description
name	EString	1	The name of the organization
legalForm	EString	0..1	Information about the type of legal entity the organization is incorporated as
numberOfEmployees	EInt	0..1	The (rough) number of people employed by the organization; **valid values:** non-negative integers
yearOfFounding	EInt	0..1	The year in which the organization was founded; **valid values:** positive, non-null integers
Relations			
Name	Type	Cardinality	Description
contactProfiles	ContactProfile	0..*	A set of contact profiles with addresses (physical and virtual) where the organization can be officially contacted
representatives	NaturalPerson	0..*	A set of persons that may officially represent the organization

Examples (in pseudo concrete syntax)
<organization xmlns:bsn="http://www.moonbank.com/BankingServicesNetwork/"> <guid> ORG-X37578-456 </guid> <namespace> http://www.moonbank.com/BankingServicesNetwork/ </namespace> <name> Moonbank Inc </name> <legalForm> Public Corporation <legalForm> <numberOfEmployees> 12000 </numberOfEmployees> <yearOfFounding> 1955 </yearOfFounding> <contactProfiles> … </contactProfiles> <representatives> bsn:USER-A325-342678 </representatives> </organization>

3.10 NaturalPerson

NaturalPerson represents unique human legal entities that exist in the real world.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Agent

NaturalPerson			
Attributes			
Name	Type	Cardinality	Description
firstName	EString	1	The person's first (given) name
middleNames	EString	0..*	The person's middle (given) names
lastName	EString	1	The person's last (family) name
title	EString	0..1	The person's official or academic title; *examples: Sir, Prof., etc.*
jobTitle	EString	0..1	The person's job title
departments	EString	0..*	Information about the departments within the organization the person belongs to
dateOfBirth	EDate	0..1	The person's date of birth
Relations			
Name	Type	Cardinality	Description
contactProfiles	ContactProfile	0..*	A set of contact profiles with addresses (physical and virtual) where the person can be contacted
Examples (in pseudo concrete syntax)			
<naturalPerson> <guid> USER-A325-342678 </guid> <namespace> http://www.moonbank.com/BankingServicesNetwork/ </namespace> <firstName> Chris </firstName> <lastName> Jasons </lastName> <jobTitle> Architect </jobTitle> <contactProfiles> … </contactProfiles> </naturalPerson>			

3.11 ResourceAgent

ResourceAgent represents concrete incarnations of resources. The concept of **Resource** describes resources abstractly, i.e. as a class of resource objects or agents. Given this definition, it is therefore possible to uniquely identify resource agents within a service network.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Agent

ResourceAgent			
Relations			
Name	Type	Cardinality	Description
type	Resource	1	Reference to the (abstract) resource that is incarnated by the agent
Examples (in pseudo concrete syntax)			

```
<resourceAgent>
  <giud> RES-XCORP-1234 </guid>
  <type>
    <name> SAP ERP 6 </name>
    <type> SoftwareResouce </type>
  </type>
  <description>
    <name> SAP ERP 6 running at XCorp </name>
    <shortDescription>
      The installation of SAP ERP 6 deployed at company XCorp and accessible at 175.26.11.219 on port 55000.
    </shortDescription>
  </description>
</resourceAgent>
```

3.12 ContactProfile

ContactProfile is used to collect physical and logical locations (virtual addresses) where persons or organizations can be contacted, e.g. postal address, telephone number, email address, etc. Several different communication methods can be included in a profile. The purpose is to group contact details with regard to a particular aspect of the associated person or organization. E.g., a person could have two profiles, one capturing private contact details and the other one capturing business details.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

ContactProfile			
Relations			
Name	Type	Cardinality	Description
physicalAddress	Physical Address	0..1	Reference to a physical address where the owner of the profile can be contacted via mail or in person
virtualAddresses	Virtual Address	0..*	References to virtual addresses (phone, mobile, email, etc.) where the owner can be contacted via electronic means
description	Element Description	0..1	A description of the contact profile

Examples (in pseudo concrete syntax)

```
<organization>
…
 <contactProfiles>

  <virtualAddresses xsi:type=”foundation:VirtualAddress”>
   <value> http://www.moonbank.com/ </value>
   <type> url </type>
   <description>
    <name> Official Web Site </name>
   </description>
  </virtualAddresses>

  <virtualAddresses xsi:type=”foundation:VirtualAddress”>
   <value> info@moonbank.com </value>
   <type> email </type>
   <description>
    <name> Enquiries via email </name>
   </description>
  </virtualAddresses>

  <virtualAddresses xsi:type=”foundation:VirtualAddress”>
   <value> +1 323 555 1001 </value>
   <type> phone </type>
   <description>
    <name> Enquiries via phone </name>
   </description>
  </virtualAddresses>

  <physicalAddress>
   <street> Wide Avenue </street>
   <streetNumber> 13 </streetNumber>
   <city> New York City </city>
   <suburb> Financial District </suburb>
   <postcode> 10005 </postcode>
   <state> New York </state>
   <country> USA </country>
   <geographicalPoint>
    <latitude> 40.707655 </latitude>
    <longitude> -74.008505 </longitude>
   </geographicalPoint>
   <description>
    <name> Moonbank Headquarters New York </name>
    <synonyms> Moonbank HQ </synonyms>
   </description>
  </physicalAddress>

 </contactProfiles>
…
</organization>
```

3.13 Classification

Classification is a generic concept that can be used to classify description elements into defined taxonomies.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

Classification			
Attributes			
Name	Type	Cardinality	Description
taxonomy	EString	1	An identifier of the taxonomy defining the different valid classes / categories; ***examples:*** *CMMI, UNSPSC*
class	EString	1	An identifier of the class / category within the taxonomy; ***examples:*** *CMMI Level 5, 43230000 (UNSPSC code for Software)*
Examples (in pseudo concrete syntax)			
<service> … <functionalClassifications> <taxonomy> http://www.moonbank.com/BankingServicesNetwork/FunctionArea </taxonomy> <class> AccountManagement </class> </functionalClassifications> … </service>			

3.14 Certification

Certification is a generic concept that can be used to associate description elements with information about certifications they hold. Certification goes beyond classification, as it also implies that a certification process has been conducted by an external party. For example, classification into CMMI is usually not done by an organization itself, but obtained from an external reviewer. A link to a certificate may prove the fact of certification.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

Certification			
Relations			
Name	Type	Cardinality	Description
classification	Classification	0..1	Reference to the class of the obtained certification in a certification taxonomy
certificate	Artifact	0..1	Link to the document holding the certificate
description	Element Description	0..1	A description of the certification

Examples (in pseudo concrete syntax)
<organization> … <certifications> <classification> <taxonomy> http://www.moonbank.com/BankingServicesNetwork/Providers/ </taxonomy> <class> PremiumProvider </class> </classification> <certificate> <type> Certificate </type> <mimeType> application/x509 </mimeType> <uri> http://www.moonbank.com/certificates/bsn/premium.cert </uri> <description> <name> Banking Services Network Premium Provider Certificate </name> <shortDescription> A certificate stating that Moonbank Inc. is a premium provider on the Banking Services Network, digitally signed by Moonbank Banking Services Network Certification Authority. </shortDescription> </description> </certificate> <description> <shortDescription> Certification of Premium Provider status on Moonbank Banking Services Network </shortDescription> </description> … </organization>

3.15 ExpressionElement

ExpressionElement represents a generic concept to capture formal expressions in arbitrary expression languages.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

ExpressionElement			
Attributes			
Name	Type	Cardinality	Description
expression	EString	1	The expression string
language	EString	1	A unique identifier of the language used to formulate the expression
Examples (in pseudo concrete syntax)			

3.16 OrderType

OrderType determines the type of ordered structures. Ordered structures are used in USDL to express that concepts follow a well-defined order, i.e. have workflow/process characteristics. Currently, this applies to composite services and complex service interaction protocols.

It was decided to only support simple block-structured order types, as the need for more complex constructs could not be substantiated yet (through concrete examples).

- Ecore Type: EEnum

OrderType			
Items			
Name	Literal	Value	Description
Sequence	Sequence	0	The parts in the ordered structure occur/execute sequentially
Parallel	Parallel	1	The parts in the ordered structure occur/execute in parallel
Choice	Choice	2	One or more parts are selected from the overall set of parts in the ordered structure; selection criteria are not specified

4 Location-related Foundation Elements

4.1 Location

Location serves as the super type of the different concrete location concepts.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: N/A

Location			
Relations			
Name	Type	Cardinality	Description
description	Element Description	0..1	A description of the location concept

4.2 PhysicalLocation

PhysicalLocation serves as the super type of concrete location concepts that exist in the physical world.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: Location

4.3 GeographicalPoint

GeographicalPoint represents a specific point location in the physical world, defined by geo-spatial coordinates.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

GeographicalPoint			
Attributes			
Name	Type	Cardinality	Description
latitude	EString	1	The latitude coordinate of the geo-spatial position
longitude	EString	1	The longitude coordinate of the geo-spatial position
Examples (in pseudo concrete syntax)			

```
...
<physicalAddress>
  <geographicalPoint>
    <latitude> 40.707655 </latitude>
    <longitude> -74.008505 </longitude>
  </geographicalPoint>
</physicalAddress>
...
```

4.4 PhysicalAddress

PhysicalAddress represents a specific point-location in the physical world, defined in the form of a traditional postal address. It can be supplemented with absolute geo-spatial coordinates.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: PhysicalLocation

PhysicalAddress			
Attributes			
Name	Type	Cardinality	Description
roomNumber	EString	0..1	The number of the room (on the floor) in the building located at the street address
floorNumber	EString	0..1	The number of the floor in the building located at the street address
buildingNumber	EString	0..1	The number of the building located at address (in case several buildings share the same street address)
street	EString	0..1	The name of the street in the street address
streetNumber	EString	0..1	The number of the street in the street address
poBoxNumber	EString	0..1	The number of a post office box
city	EString	0..1	The name of the city in the street address
suburb	EString	0..1	The name of the suburb in the street address
postcode	EString	0..1	The postcode in the street address
province	EString	0..1	The name of the province in the street address
state	EString	0..1	The name of the state in the street address
country	EString	0..1	The name of the country in the street address
Relations			
Name	Type	Cardinality	Description
geographicalPoint	Geographical Point	0..1	Reference to a geographical point matching the address, defined in geo-spatial coordinates

Examples (in pseudo concrete syntax)
<naturalPerson> … <contactProfiles> <physicalAddress> <floorNumber> 26 </floorNumber> <buildingNumber> A4 </buildingNumber> <street> W Common Blvd. </street> <streetNumber> 33 </streetNumber> <city> Philadelphia </city> <postcode> 19132 </postcode> <state> Pennsylvania </state> <country> USA </country> <description> <name> Office Address </name> </description> </physicalAddress> </contactProfiles> … </naturalPerson>

4.5 AdministrativeArea

AdministrativeArea represents a finite area in the real world with more-or-less specific boundaries. It groups multiple physical addresses into a logical aggregate.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: PhysicalLocation

AdministrativeArea			
Attributes			
Name	Type	Cardinality	Description
name	EString	1	The name of the administrative area
type	AdministrativeAreaType	1	This attributes denotes the type of administrative area
Examples (in pseudo concrete syntax)			
<service> … <deliveryLocation xsi:type=”foundation:AdminstrativeArea”> <name> United States of America </name> <type> country </type> <description> <shortDescription> The service can only be delivered to applicants with a valid address in the USA. </shortDescription> </description> </deliveryLocation> … </service>			

4.6 AdministrativeAreaType

AdministrativeAreaType is used to express types of aggregators of physical addresses, e.g. continent, country, city, postcode, etc.

- Ecore Type: EEnum

AdministrativeAreaType			
Items			
Name	Literal	Value	Description
other	other	0	an area type not represented by any of the other types
continent	continent	1	area type representing a whole continent
country	country	2	area type representing an area delineated by the official borders of a country
state	state	3	area type representing an area delineated by the official borders of a state within a country
province	province	4	area type representing an area delineated by the official borders of a province, either within a state or a country
city	city	5	area type representing an area delineated by the official borders of a city
suburb	suburb	6	area type representing an area delineated by the official borders of a suburb within a city
postcode	postcode	7	area type representing an area delineated by the official borders of a postcode area
street	street	8	area type representing an area delineated by the official borders of the lots located on the same street (identified by the official street address)

4.7 Polygon

Polygon represents a geometrical shape delimited by an array of two or more geographical points (geo-spatial coordinates). Interpreting the points as the corners (vertices) of a two-dimensional polygon, an area is spanned by computing the convex hull.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: PhysicalLocation

Polygon			
Relations			
Name	Type	Cardinality	Description
vertices	Geographical Point	2..*	References to 2 or more geographical points that mark the corners of the polygon and are used to initialize the computation of the convex hull
Examples (in pseudo concrete syntax)			

4.8 Area

Area represents areas relative to a physical location (point location or one of the administrative area types). The Area is defined by a perimeter around the given reference location(s).

Examples:

- *a 5.5 km circle around a given address (point location)*
- *15 km around the borders of a country (administrative area)*

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: PhysicalLocation

Area			
Attributes			
Name	Type	Cardinality	Description
perimeter	EFloat	1	The number of units marking the perimeter
unitOfLength	UnitOfLength Type	1	The unit of length used to compute the perimeter
Relations			
Name	Type	Cardinality	Description
physicalLocations	Physical Location	1..*	The reference locations around which perimeters are drawn
Examples (in pseudo concrete syntax)			

4.9 UnitOfLengthType

UnitOfLengthType is used to express different units of length/distance, such as meter, miles, etc.

- Ecore Type: EEnum

UnitOfLengthType			
Items			
Name	Literal	Value	Description
mm	mm	0	millimeter
cm	cm	1	centimeter
dm	dm	2	decimeter
m	m	3	meter
km	km	4	kilometer
mi	mi	5	statute mile (1609.344 meters)
nm	nm	6	nautical mile (1852 meters)
in	in	7	inch (0.0254 meters)
ft	ft	8	foot (0.3048 meters)
yd	yd	9	yard (0.9144 meters)

4.10 VirtualAddress

VirtualAddress represents a specific point-location in an electronic network, e.g. telephone number, email address, IP address, etc.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Location

VirtualAddress			
Attributes			
Name	Type	Cardinality	Description
value	EString	1	The address value, i.e. a unique identifier that can be used to resolve the address in the overall address space; ***examples:*** - *+4901234/56789 (phone number)* - *user@domain.com (email address)* - *robbi1982 (messenger ID)*
type	VirtualAddress Type	1	The type of virtual address indicating which address space to use
Examples (in pseudo concrete syntax)			
<naturalPerson> … <contactProfiles> <virtualAddresses xsi:type=”foundation:VirtualAddress”> <value> +1 366 555 1234 </value> <type> phone </type> <description> <name> Work Phone </name> </description> </virtualAddresses> </contactProfiles> … </naturalPerson>			

4.11 VirtualAddressType

VirtualAddressType represents types of virtual addresses and indicates which address space to use when resolving virtual addresses.

- Ecore Type: EEnum

VirtualAddressType			
Items			
Name	Literal	Value	Description
phone	phone	0	Phone number
fax	fax	1	Fax number
email	email	2	E-Mail address
url	url	3	URL
ipv4	ipv4	4	Internet Protocol v4 address

ipv6	ipv6	5	Internet Protocol v6 address
mac	mac	6	Media Access Control address
messaging	messaging	7	Messaging system identifier; should only be used with MessagingAddress, which specifies the actual system used (e.g. Skype, ICQ, etc.)

4.12 MessagingAddress

MessagingAddress is a special type of virtual address that is used for addresses in instant messaging systems. Because there are quite a number of such systems, it is necessary to identify a particular system in order to narrow down the address space and successfully resolve the address.

Note that instances of this class have to be of ***type*** *messaging*.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: VirtualAddress

MessagingAddress			
Attributes			
Name	Type	Cardinality	Description
messagingSystem	EString	1	The identifier of the messaging system in whose address space the address is located
Examples (in pseudo concrete syntax)			

```
<naturalPerson>
…
 <contactProfiles>

  <virtualAddresses xsi:type=”foundation:MessagingAddress”>
   <value> c.jasons@moonbank.com </value>
   <type> messaging </type>
   <messagingSystem> MSN@Moonbank </messagingSystem>
   <description>
    <name> Corporate Messenger </name>
   </description>
  </virtualAddresses>

 </contactProfiles>
…
</naturalPerson>
```

4.13 VirtualRegion

VirtualRegion represents a set of virtual addresses that belong to a section of an address space (network).

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Location

VirtualRegion			
Attributes			
Name	Type	Cardinality	Description
uriTemplates	EString	0..*	A URI template reflects a range of addresses; ***example:*** *an IP address range 192.168.0.11-192.168.0.35*
Relations			
Name	Type	Cardinality	Description
groups	VirtualAddress	0..*	References to individual virtual addresses that are grouped into a virtual region
Examples (in pseudo concrete syntax)			

5 Time-related Foundation Elements

5.1 Time

Time serves as the super type of the different concrete time concepts.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: N/A

Time			
Attributes			
Name	Type	Cardinality	Description
timeZone	TimeZone	1	The time zone in which specific points in time are given
description	Element Description	0..1	A description of the time concept

5.2 UnitOfTimeType

UnitOfTimeType is used to express units of time, i.e. standardized time spans (durations).

- Ecore Type: EEnum

UnitOfTimeType			
Items			
Name	Literal	Value	Description
year	year	0	unit representing years
month	month	1	unit representing months
week	week	2	unit representing weeks
day	day	3	unit representing days
hour	hour	4	unit representing hours
minute	minute	5	unit representing minutes
second	second	6	unit representing seconds
millisecond	millisecond	7	unit representing milliseconds

5.3 TimeEntity

TimeEntity represents a time span (duration) of a certain finite length. As opposed to a time interval, it is not anchored at a specific point in time, but is used to specify the length of a time interval.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: N/A

TimeEntity			
Attributes			
Name	Type	Cardinality	Description
value	EFloat	1	The quantity of units of time to specify the entities length
type	UnitOfTime Type	1	The unit of time to be used to compute the entities length
Examples (in pseudo concrete syntax)			

```
<service>
…
 <requestTime xsi:type=”foundation:DurationInterval”>
 …
  <intervalDuration>
   <type>year</type>
   <value>3.0</value>
  </intervalDuration>
 <requestTime>
…
</service>
```

5.4 TimeInstant

TimeInstant represents a specific point in time. It serves as the super type of two concrete classes, each defining an instant in a different way.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: Time

5.5 AbsolutePointInTime

AbsolutePointInTime represents a specific point in type specified by a date-time instance.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: TimeInstant

AbsolutePointInTime			
Attributes			
Name	Type	Cardinality	Description
value	EDate	1	The date-time that marks the time instant

Examples (in pseudo concrete syntax)
<service> … <requestTime xsi:type=”foundation:DurationInterval”> … <start xsi:type=”foundation:AbsolutePointInTime”> <value> 2010-01-01T02:00:00.000-0600 </value> <description> <name> Roll-out time </name> </description> </start> … </requestTime> … </service>

5.6 RelativePointInTime

RelativePointInTime represents a date-time instant that is relative to some event, such as request date, delivery date, etc. This date-time instant can either lie in a certain distance, such as "payment is due in 3 days", or the distance can differ, such as in "payment is due next Monday" which can be 7 days from now on if today is already a Monday or it can be 3 days from now on if today is a Friday.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: TimeInstant

RelativePointInTime			
Attributes			
Name	Type	Cardinality	Description
relativeTo	Reference Event	1	The reference event to which the point in time is relative to
Relations			
Name	Type	Cardinality	Description
relativeDuration	TimeEntity	0..*	Time span components, which when added up specify when the point in time is reached
nextTimePoint	TimePattern	0..1	A single value time pattern that denotes a time concept, whose next occurrence marks the time instant; ***example:*** *next May (month of year range: 5)*
Examples (in pseudo concrete syntax)			

5.7 ReferenceEvent

ReferenceEvent is used to express distinct events in the provisioning of services or service bundles that serve as the reference (anchor) to relative points in time.

- Ecore Type: EEnum

ReferenceEvent			
Items			
Name	Literal	Value	Description
Other	Other	0	Reference event not captured by one of the events listed below
ServiceRequest	Service Request	1	Reference event is the requesting of a service or service bundle
ServiceDeliveryStart	Service Delivery Start	2	Reference event is the start of the delivery of a service or service bundle
ServiceDeliveryEnd	Service Delivery End	3	Reference event is the completion of the delivery of a service or service bundle

5.8 TimeInterval

TimeInterval represents a continuous time span with defined start and end points in time, i.e. a certain finite duration. It serves as the super type of two concrete classes, each defining an interval in a different way.

- Ecore Type: Abstract EClass
- Interfaces: N/A
- Superclass: Time

TimeInterval			
Relations			
Name	Type	Cardinality	Description
start	TimeInstant	1	The point in time when the interval starts
Examples (in pseudo concrete syntax)			

5.9 StartToEndInterval

StartToEndInterval is a time interval delimited by a specific start and end point in time.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: TimeInterval

StartToEndInterval			
Relations			
Name	Type	Cardinality	Description
end	TimeInstant	1	The point in time when the interval ends
Examples (in pseudo concrete syntax)			

5.10 DurationInterval

DurationInterval is a time interval defined by a specific starting point in time and a time span indicating its length. The time span can be composed out of arbitrary time entities of different length, e.g. 3 weeks and 2 days.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: TimeInterval

DurationInterval			
Relations			
Name	Type	Cardinality	Description
intervalDuration	TimeEntity	1..*	Time span components, which when added up specify the length of the duration interval

Examples (in pseudo concrete syntax)

```
<service>
…
 <requestTime xsi:type=”foundation:DurationInterval”>
  <start xsi:type=”foundation:AbsolutePointInTime”>
   <value> 2010-01-01T02:00:00.000-0600 </value>
   <description>
    <name> Roll-out time </name>
   </description>
  </start>
  <intervalDuration>
   <type>year</type>
   <value>3.0</value>
  </intervalDuration>
  <description>
   <name> Service Validity Period </name>
   <shortDescription> The service will be available from 1st of January, 2010, 2:00am EST </shortDescription>
  </description>
 </requestTime>
…
</service>
```

5.11 RecurrentTime

RecurrentTime describes a time pattern that is repeated in a specific scheme, for example repeat every 3 months and two weeks. The pattern is repeated either for the duration of a valid time interval OR for a number of times starting at a specific point in time.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Time

RecurrentTime			
Attributes			
Name	Type	Cardinality	Description
numberOfRecurrences	EInt	0..1	Indicates how many times the pattern will recur; has to be set in combination with *validFromStartTime*; is alternative to *validDuringTimeInterval*

Relations			
Name	Type	Cardinality	Description
validFromStartTime	TimeInstant	0..1	Specifies the point in time when the sequence of recurrences starts; has to be set in combination with *numberOfRecurrences*; is alternative to *validDuringTimeInterval*
validDuringTimeInterval	TimeInterval	0..1	Specifies the anchored time span during which the pattern is repeated; ; is alternative to *validFromStartTime* and *numberOfRecurrences*
repeatEvery	TimeEntity	1..*	Time span components, which when added up specify the length of the interval after which a recurrence takes place
timePatternToRecur	TimePattern	1	Reference to a time pattern that is recurred
Examples (in pseudo concrete syntax)			

5.12 TimePattern

TimePattern is defined by a set of time ranges. A time range can be a range of years, months, weeks (year-based and month-based), days (year-, month-, and week-based), day times, hours, minutes, or seconds.

Examples include:

- *a single year, month, time of day, etc., e.g.: range of days of the week: 6 (Saturday)*
- *a list of several years, months, etc. separated by a comma, e.g. range of months: 1,3,5 (January, March, May)*
- *a range of years, months, etc. connected by a minus, e.g. range of years: 2010-2020*

Note that this concept is still under exploration. Ranges are therefore encoded as a single expression using an expression language that is not defined here.

- Ecore Type: EClass
- Interfaces: N/A
- Superclass: Time

TimePattern			
Relations			
Name	Type	Cardinality	Description
timeRanges	Expression Element	1	The expression that encodes the time ranges that make up the time pattern
Examples (in pseudo concrete syntax)			