Appendix 2

National Research Program

"Cyber-physical systems, ontologies and biophotonics for safe&smart city and society" (SOPHIS)

Project No.2
"Ontology-based knowledge engineering technologies suitable for web environment"

Scientific report

Period 3

2016

Introduction

The goal of the project No.2 is to develop the scientific expertise of the next generation IT systems by researching and further developing novel competitive model-based information technologies and their applications in modern web environment and to transfer the created expertise and technologies to concrete domains of Latvia's economics, as well as introducing them into the higher education study process.

The main tasks of the stage 3 are:

- Further development of the ontology- and web technologies-based fast query language and its efficient implementation by introducing the view-based extension mechanism.
- Development of data access control mechanism based on data ontologies and web technologies, to be used for the implementation of the Ad-hoc query language
- Development of metamodel specialization methods and their application to building of domain specific language tools for web environment.
- Participation in SemEval-2016 competition with an improved version of the C6.0 classification algorithm adapted for Abstract Meaning Representation (AMR) information retrieval
- Further enhancement of the functionality of intelligent structural modeling tool I4S for multicriterial estimation of the importance of elements in knowledge structures of different types and granularity
- Development, merging and application of knowledge structure models in ontology- and rule-based decision making
- Detailed development of the approach for aligning requirements/systems engineering based knowledge flows (structures)/information artifacts
- Development and integration of different services in Web portal using open Semantic Web resources and enhancement of software configuration management methods
- Development of technologies for large scale NoSQL data base exploration and visualization
- Business process runtime verification

The goal of this report is to describe the main scientific and practical results during the reporting period. The parts of scientific results which are adequately presented in the corresponding publications will not be described in detail in this report; direct links to the corresponding publications will be provided instead.

2.1. Ontology based tools for knowledge analysis and mining semantics of natural language

2.1.1. Further development of the ontology- and web technologies-based fast query language and its efficient implementation by introducing the view-based extension mechanism.

2.1.1.1. Basic Ideas

The query language we propose here is to be used for formulation of ad-hoc queries, meaning queries that can be formulated in one sentence (perhaps together with some subordinate clauses) in natural language, so that the end-user can still understand it very well. The language will work on semistar ontologies (being ontologies that have only one type of relations between any two basic classes – the *has* relation). The simplicity of the "has" relation is the main factor, which allows query language to be simple and similar to a natural language. It is therefore convenient to build queries in a controlled natural language. This feature allows the language to be easily perceptible by non-IT specialists.

The semistar data ontologies-based fast query language is efficiently implemented, and its performance have been testes on real CCUH data and typical queries. We have reached the query execution performance of about 0.3 seconds on average per one query. This would match the performance of about 1 second per query if we took data from all the hospitals in Latvia (and take into account the potential to execute the query calculation process in parallel). These developments and results have been described in two publications: "J.Barzdins, M.Grasmanis, E.Rencis, A.Sostaks, A.Steinsbekk, Towards a More Effective Hospital: Helping Health Professionals to Learn from their Own Practice by Developing an Easy to use Clinical Processes Querying Language. // J.E.Q. Varajão et. al. (Eds.), Procedia Computer Science, Vol. 100, Elsevier, pp. 498-506, 2016" and "J.Barzdins, M.Grasmanis, E.Rencis, A.Sostaks, J.Barzdins, Ad-Hoc Querying of Semistar Data Ontologies Using Controlled Natural Language. // In: G.Arnicans, V.Arnicane, J.Borzovs, L.Niedrite (Eds.), Frontiers of AI and Applications, Vol. 291, Databases and Information Systems IX, IOS Press, pp. 3-16, 2016. (to be indexed SCOPUS), http://ebooks.iospress.com/volumearticle/45695".

Let us introduce an example query that will be exploited further in this section – count Patients, who have at least one HospitalEpisode, which has Manipulation with manipul.code=02078. This natural language sentence is understandable by the domain expert. Let us now inspect a bit more complicated query: count Patients, who have at least one HospitalEpisode, which has at least one TreatmentWard, which has at least one Manipulation with manipul.code=02078. This sentence may cause a certain ambiguity as it is not clear whether the asked *Manipulation* refers to *HospitalEpisode* or to *TreatmentWard*. It could be used in both meanings. In other words, relative pronouns such as "who" and "which" not always give us accurate understanding of what we relate to. To cope with such situations we introduce a concept of so called short name in our controlled natural language. Formally, the short name is a variable

over instances of the given class – count Patients p, where exists p.HospitalEpisode e, where exists e.TreatmentWard t, where exists t.Manipulation m, where m.manipul.code=02078. Now we are able to specify also the second of abovementioned meanings – count Patients p, where exists p.HospitalEpisode e, where exists e.TreatmentWard t, where exists e.Manipulation m, where m.manipul.code=02078. We have also unified other components of the natural language, e.g. we use the keyword "where" instead of "who", "which" and "with", and the keyword "exists" instead of "have/has at least one". The dot notation after the short name must be perceived as the "of" relation – count Patient p, where exists HospitalEpisode e of Patient p, where exists TreatmentWard t of HospitalEpisode e, where exists Manipulation m of TreatmentWard t, where manipul.code of Manipulation m equals 02078.

Formally speaking, the short name must be used before every attribute name to get rid of ambiguities. However, in cases when it is clear to which class the particular attribute refers the short name can be omitted. We also allow omitting other features of the language that are not critical for understanding of queries (e.g. one can omit the empty parentheses after the unary Date and DateTime operations like *year()* or *minute()*).

Let us now introduce some basic notations that we will use to describe the query language. We will use the terms *parent class* and *child class* to refer to classes that are higher or lower in the "have" hierarchy. For example, the class "TreatmentWard" has two parent classes – "HospitalEpisode" (direct parent) and "Patient" (further ancestor) and one child class "Manipulation". If x is an instance of the class "TreatmentWard", then its parent instances will be denoted as *x.HospitalEpisode* and *x.Patient*. In both cases they denote exactly one instance, i.e. that of the class "HospitalEpisode" and of the class "Patient", respectively. We use the same dot notation also for accessing instances of child classes, but in this case we obtain a set of instances. For example, *x.Manipulation* would be a set of manipulations reachable from the given treatment ward x.

In more complicated cases another concept of brother class is important. If x is an instance of the class "HospitalEpisode", then by *x.HospitalEpisode* we understand *y.HospitalEpisode*, where *y=x.Patient* (i.e. y is the closest parent of *x*, which is also parent class of the given class "HospitalEpisode"). Similarly, if *x* is an instance of the class "TreatmentWard", then *x.OutpatientEpisode=y.OutpatientEpisode*, where *y=x.Patient*.

If *AClass* is an arbitrary class of the ontology, we will use the term *AClass attribute expression* to denote attribute expressions of both *AClass* itself and all of its parent classes (we assume here that parents and children share no common attribute names). We cannot use attribute expressions of child classes here, because there can be many children instances for the given *AClass* instance. We will be able to access these instances by introducing quantors *exists/notexists* later.

2.1.1.2. Syntax and Semantics of the Query Language

Queries are to be written in a controlled natural language and are based on seven sentence templates. The main part of the templates is the so called *selection condition*,

which is a selection condition over instances of the given class. We assume that selection conditions are to be written in a natural language. We describe the used language constructs more formally at the end of this section. However, the sentence templates described in this section can be understood without knowing the precise syntax of selection conditions.

T1. COUNT AClass [x] WHERE <selection condition>

Semantics: counts instances of AClass, which satisfy the selection condition.

Examples:

- *COUNT Patients, WHERE EXISTS HospitalEpisode, WHERE referringPhysician=familyDoctor* (count of patients who have been referred to hospital by their family doctors);
- *COUNT HospitalEpisodes, WHERE dischargeTime-admissionTime>15d* (how many episodes have lasted longer than 15 days);
- *COUNT HospitalEpisodes e1, WHERE EXISTS HospitalEpisode e2, WHERE e1<>e2 AND e2.admissionTime>e1.dischargeTime AND e2.admissionTime-e1.dischargeTime<30d* (how many there have been such episodes, after which the patient has returned to hospital in less than 30 days).

T2. {SUM/MAX/MIN/AVG/MOST} <attribute expression> FROM AClass [x] WHERE <selection condition>

Semantics: selects instances of AClass, which satisfy the selection condition, calculates the attribute expression for each of these instances obtaining a list to which the specified aggregate function is then applied.

Examples:

- *SUM totalCost FROM HospitalEpisodes, WHERE dischargeReason=cured AND birthDate.year()=2012* (how much successful treatments of patients born in 2012 have cost);
- *MOST diagnosis.code FROM DischargeDiagnoses, WHERE nr=1 AND dischargeReason=deceased* (get the most frequent main (nr=1) death diagnosis).

T3. SELECT FROM AClass [x] WHERE <selection condition> ATTRIBUTE <attribute expression> ALL DISTINCT VALUES

Semantics is obvious.

Examples:

- SELECT FROM HospitalEpisodes, WHERE dischargeReason=deceased, ATTRIBUTE responsiblePhysician.surname ALL DISTINCT VALUES;
- SELECT FROM DischargeDiagnoses, WHERE nr=1 AND dischargeReason=deceased, ATTRIBUTE diagnosis.code ALL DISTINCT VALUES.

T4. SHOW [n/ALL] AClass WHERE <selection condition>

Semantics: shows n or all instances of AClass which satisfy the selection condition.

T5. FULLSHOW [n/all] AClass WHERE <selection condition>

Semantics: the same as “show”, but shows also the child class instances attached to the selected AClass instances.

T6. SELECT AClass x WHERE <selection condition>, DEFINE TABLE <x-expr'1> [(COLUMN C1], …, <x-expr'n> [(COLUMN Cn)] [, KEEP ROWS WHERE <Ci selection condition>] [, SORT [ASCENDING/DESCENDING] BY COLUMN Ci] [, LEAVE [FIRST/LAST] n ROWS]

Semantics: selects all instances of AClass, which satisfy the selection condition, then makes a table with columns C1 to Cn, which for every selected AClass instance x contains an individual row, which in column C1 contains the value of the <x-expr'1>, …, in column Cn contains the value of the <x-expr'n>. Then it is possible to perform some basic operations with the table like filtering out unnecessary rows, sorting the rows by values of some column and then taking just the first or the last n rows from the table.

Examples:

- SELECT HospitalEpisodes x, WHERE dischargeReason=deceased, DEFINE TABLE x.surname (COLUMN Surname), x.dischargeTime.date() (COLUMN Dying_date), (COUNT x.Manipulation, WHERE manipul.code=02078) (COLLUMN Count_02078), (SUM manipul.cost FROM x.Manipulation, WHERE manipul.code=02078) (COLUMN cost_02078);
- SELECT CPhysicians k, WHERE name=Gatis AND EXISTS HospitalEpisode, WHERE responsiblePhysician=k, DEFINE TABLE surname (COLUMN Physician_surname), (COUNT HospitalEpisodes, WHERE responsiblePhysician=k) (COLUMN Episode_count), (MOST diagnosis.code FROM AdmissionDiagnoses, WHERE nr=1 AND responsiblePhysician=k) (COLUMN Most_frequent_main_diagnosis), KEEP ROWS WHERE Episode_count>5, SORT DESCCENDING BY COLUMN Episode_count, LEAVE FIRST 10 ROWS.

Let us now talk a bit more precisely about the means for defining columns. The expression <x-expr'i> defines the value of column Ci in the row that corresponds to the AClass instance x. This expression can be defined in one of four ways:

<x-expr'i> ::= <x-dependent attribute expression> | <x-dependent count expression> | <x-dependent {SUM/MAX/MIN/MOST] expression> | <x-dependent child attribute selector expression>

- <x-dependent attribute expression> examples: x.surname, x.dischargeTime.date(). Prefix “x.” can be used before attributes of both x and its parents. Semantics is obvious.

- <x-dependent count expression> examples: (COUNT x.Manipulations, WHERE manipul.code=02078), (COUNT HospitalEpisodes, WHERE responsiblePhysician=x). In the first example we use the prefix x in "x.Manipulations" to denote that we do not select from the whole set of manipulations, but only from those that are reachable from x.
- <x-dependent {SUM/MAX/MIN/MOST} expression> examples: (SUM manipul.cost FROM x.Manipulations, WHERE manipul.code=02078), (MOST diagnosis.code FROM AdmissionDiagnoses, WHERE nr=1 AND responsiblePhysician=x).
- <x-dependent child attribute selector expression> examples: (x.DischargeDiagnosis, WHERE nr=1).diagnosis.code, (x.TreatmentWard, WHERE nr=*).ward (by * we denote the number of the last instance of TreatmentWard connected to the given HospitalEpisode x). This is a new kind of construction whose general form is as follows: (x.<name of x children class A>, WHERE <selection condition>).<name of attribute a of class A>. Its value is defined in the following way – we start by taking all instances of class A that are reachable from x, then select of them those instances that satisfy the selection condition and then create a list of values of the attribute a of the selected instances. The most important case here is the one where this list contains only one instance, e.g. in the following table definition
  example: *SELECT HospitalEpisodes x, WHERE dischargeReason=deceased, DEFINE TABLE x.surname (COLUMN Surname), x.dischargeTime.date() (COLUMN Dying_date), (x.DischargeDiagnosis, WHERE nr=1).diagnosis.code (COLUMN main_diagnosis), (x.TreatmentWard, WHERE nr=*).ward (COLUMN last_ward).*

T7. There are two more cases in the definition of the table, where table rows come from other source, not being instances of some class. Being very similar these two cases form two subtemplates of the last template:

a. SELECT FROM AClass [a] WHERE <selection condition> ATTRIBUTE <attribute expression> ALL DISTINCT VALUES x, DEFINE TABLE…
b. SELECT FROM INTERVAL (start-end) ALL DISTINCT VALUES x, DEFINE TABLE&helip;

Semantics of both cases is obvious.

Examples:

- SELECT FROM TreatmentWards ATTRIBUTE ward ALL DISTINCT VALUES x, CEATE TABLE x (COLLUMN Ward), (SUM manipul.cost FROM Manipulations, WHERE ward=x) (COLUMN Cost);
- SELECT FROM INTERVAL (1-12) ALL DISTINCT VALUES x, DEFINE TABLE x (COLUMN Month), (COUNT HospitalEpisodes, WHERE admissionTime.month()=x) (COLUMN Episode_count) (MOST diagnosis.code FROM AdmissionDiagnoses, WHERE nr=1 AND admissionTime.month()=x) (COLUMN Most_frequent_main_diagnosis).

Let us conclude this section by defining more formally the constructs of a controlled natural language allowed in the selection conditions. They can, of course, be guessed from the examples given above.

<AClass selection expression> ::= AClass [<short name>] [WHERE <selection condition>]

<selection condition> ::= <attribute condition> | <quantor condition> | (<selection condition> {AND|OR} <attribute condition>) | (<selection condition> {AND|OR} <quantor condition>)

<quantor condition> ::= {[NOT]EXISTS | FORALL} [short name.] AClass [short name] [WHERE <selection condition>]

Short name provides a name for the given object and can be any string different from the class and attribute names. The abovementioned grammar provides a formal language (for formulating selection expressions) that is close to a natural language and therefore easily perceptible. From a natural language's point of view selection expressions are sentences in a controlled natural language that exploit both words of a natural language (like AND, OR, WHERE, EXISTS, NOTEXISTS) and "foreign" words – attribute expressions whose syntax and semantics were described above. The grammar is only needed as a guide how to build the selection expressions.

2.1.1.3. The View Definition Mechanism

Additionally to the fast ad-hoc query language we have introduced a new construct – the view definition mechanism –, and we have implemented it efficiently. This new feature allows end-users to create new subclasses of ontology classes by defining them using only constructs of the query language. Therefore end-users are not obliged to learn any new languages. An example of the subclass definition:

```
DEFINE SuccessfulPatient = Patient WHERE FORALL
HospitalEpisodes HOLDS dischargeReason=healthy
```

The newly defined subclasses are stored within the same structure as other classes, so they can at once be used in queries in the same way the basic class from the ontology are. For example, one can now formulate a query using the class name "SuccessfulPatient":

```
COUNT SuccessfulPatients
```

or

```
SELECT 10 SuccessfulPatients, DEFINE TABLE name, surname
```

This allowance of instant exploitability of subclasses hugely improves the practical usability of the query language. The developed view definition mechanism does not decrease the performance of the system (i.e. the time needed for obtaining answer to queries that exploit views) significantly.

2.1.2. Development of data access control mechanism based on data ontologies and web technologies, to be used for the implementation of the Ad-hoc query language

Data access control is a widely investigated topic in the last decades. The most used methods for data access control are:

- Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC)
- eXtensible Access Control Markup Language (XACML), which includes also the above ones as profiles
- these methods are universal, but very complicated for practical usage

Therefore an important question arises whether in the case of semistar data ontologies and the Ad-hoc Query Language a simpler approach can be found which still preserves some universality.

This document briefly describes the main principles of user access control for the Ad-hoc Query Language described in Section 2.1.1. The main idea is that the same query language can be adapted for defining access rules for user categories. The same semistar data ontology on which the queries are based can be reused for access control as well. The proposed access control facilities are smart enough to define adequately the access rights in such a sensitive domain as a complete internal information of a hospital. The medicine domain is the main application area for the Ad-hoc query language.

The basic principles of the proposed access control are fairly classic. A system *user* is assumed to be a human being (not another software system). To identify a user some *authentication* mechanism is used. The simplest kind of authentication is described here, but more advanced facilities could be used as well – the authentication is not the main topic of the document. The other aspect of access control – the *authorization* (what an authenticated user can do with data) is described in detail since there are the main innovative aspects of the approach. Namely this is the aspect where the query language can be reused. Only the read access to data is analyzed here because the query language is meant for easy access to data and their analysis and visualization. In a sense, the approach uses the Role-based access control (RBAC), but in an extended way – roles can have parameters which determine the access details. They are defined via access rules.

In this approach the access to a data domain, e.g. to a hospital information system, is being defined via an *access control schema*. This schema defines how each user category – *role* – has to be authenticated and to what part of data it is authorized to have a read access. Schema is defined as a table containing a row for each role. See an example of such schema definition table in Fig. 2.1.2.1, with five roles defined. This example is based on a simplified version of the semistar data ontology for Riga Children's Clinical University Hospital – the main example used for Ad-hoc Query Language definition. See this ontology example in Fig. 2.1.2.2. There is one more table in the approach for the given schema – the *access table*, to be used during the query system runtime. This table is managed by the system administrator and can change in time. It contains rows for all registered user – role combinations. A row in

this table must be in complete accordance with the row in the schema definition table where the referenced role is defined. The values of both parameters (if required by the definition row) must have fixed values in the access row – they must be set by the administrator.

The first two columns of the schema definition table are related to user authentication and subsequent local identification. The authentication column defines the authentication facility to be used. The only fully explained case in this document is the simplest one – the Password option. It means that the user has to enter his registered (in the access table) password upon login to the query system. If the person code parameter C (without brackets) is present in the definition row, it also has to be entered by the user. The other options in this column are only briefly sketched here – External authentication means the usage of some existing facility for such action, e.g. via an internet banking, it must be further detailed in a definition row. The Open access option means no user authentication at all.

The Person code column defines how the given user is identified in in the system. In the proposed simple case this identification must be based on the user's *Person code* – a unique identifier assigned to all citizens of Latvia ("personas kods" in Latvian, used English synonyms are Personal code, Personal identity number, Personal number …). The usage of code is denoted by the parameter C, C without brackets means that the code value must be mandatory supplied at login – either by the user or by some support service (in the case of external authentication). Code in brackets means that the presence of code can be determined by the system administrator – it is not used as a parameter in the corresponding access rule. The usage of this code in access rules is based on the fact that the data ontology uses this attribute (personCode) for the classes Patient and CPhysician (see Fig. 2.1.2.2), these classes represent both the data stored in the system and potential users of the system.

The role name is a unique name per definition table – thus each definition row defines one role. A role can have a parameter – here denoted by n. The parameter value can be used in access rules, but it is not directly entered by the user. The administrator is responsible for creating access table rows so that the required login parameters (password and possibly person code) uniquely determine a row with the given value of n. The last column – the Access rule requires a more detailed explanation since it is the main innovative aspect of the approach

Authentication	Person code C	Role	Role parameter	Access rule
Password	C	Responsible Physician	–	FULLSELECT HospitalEpisode WHERE responsiblePhysician.personCode = C WITHOUT Patient.personCode
Password	[C]	Ward manager	n	FULLSELECT HospitalEpisode WHERE EXISTS TreatmentWard WHERE ward = n WITHOUT Patient.personCode
Password	[C]	Hospital CEO	–	FULLSELECT Patient WITHOUT Patient.personCode
External authentication (via Swedbank)	C	Patient	–	FULLSELECT Patient WHERE personCode = C
Open access	–	Journalist	–	FULLSELECT Patient WITHONLY

				HospitalEpisode.totalCost WITHOUT Patient.ALL, TreatmentWard.ALL, Manipulation.ALL, OutPatientEpisode.ALL, AdmissionDiagnosis, DischargeDiagnosis

Fig. 2.1.2.1 Example of a schema definition table.

The Access rule determines the subset of data which a user can see after a successful login. The rule is formulated in a simple extension of the Ad-hoc query language, by preserving its syntax and semantics as far as possible. Each row in the schema definition table contains just one access rule for the role defined in this row. The access rule defines a view on the original data model (ontology), specifying which classes and which attributes the user can use in his queries and which data instances (data rows) are included in the query result. The access rule essentially relies on the semistar structure of the data both in syntax and semantics – similar to the basic query language.

The access rule language contains just one kind of query statements, defined via the keyword FULLSELECT. The main structure of this statement is similar to the SELECT statement in the query language. It also contains a class reference and a WHERE-condition, with literally the same syntax and semantics as in the SELECT. The resulting view defined by a FULLSELECT statement bears the closest similarity to the FULLSHOW statement in the query language. Only the class instances which would be returned by a FULLSHOW with the same class parameter and selection condition are included in the resulting view (instance tree). It should be noted that here the selection condition can contain only the parameters C and n from the corresponding schema definition row in attribute value comparisons (if the parameters are present).

But FULLSELECT statement has one more part – the WITHOUT clause. This clause specifies which attributes and classes must be hidden in the resulting view (instance subtree). The classes and attributes are given as comma-separated list, using the standard notation (as in the select condition part). If a class is included in the list, nothing of it can be seen. An attribute inclusion means that only this attribute is hidden. A specific situation is when a class is not hidden but all its attribute are. This means that a user can include in the query the instance count for this class, but cannot see any attribute values or use them for selection. Since this situation is really used, a shortcut notation with the keyword ALL instead of attribute can be applied. Another shortcut notation can be used for the case when only few attributes from a large number of them are not hidden for a class – then the class (with the remaining attributes) can be placed under the "opposite" keyword WITHONLY (see the last rule in Fig. 2.1.2.1).

Now some comments on Access rules really used in the example in Fig. 2.1.2.1. We remind that the data ontology for this example is given in Fig. 2.1.2.2.

The access rule for the Responsible physician role asserts that he can see all the information on hospital episodes for which he has been responsible – the involved person code must be his own code supplied at login. He can also see the information about the patient involved in the episode, except the patient's person code.

The rule for ward manager can see the episodes where at least one treatment part was performed in the ward managed by him. The specification of the ward number is indirect for the user – he enters the appropriate password at login, and it is the responsibility of the administrator to configure the access table rows so that the manager has access to the correct ward (the value of n is set correctly). The only hidden information again is the patient's person code.

The hospital CEO can see all information in the hospital, except the patient's person code.

A patient can see information only about himself. It is specified that his authentication must be performed via internet banking in the Latvian Swedbank – it is one of the Latvian banks providing such services for access to personal medical data. The authentication procedure in the bank returns the relevant person code to the service requester – the query system. Thus we can guarantee that only the data subtree for the given patient is visible for him.

Finally, the only example of public access to data – for journalists – specifies in the access rule, that in fact no attributes are visible in this case. The only visible attribute is the totalCost for HospitalEpisode class (this attribute is in the list under the WITHONLY keyword). All attributes of other classes are hidden via the ALL construct, the classes AdmissionDiagnosis and DischargeDiagnosis are hidden completely. Thus a journalist can see how many patients have been treated in the hospital (count them), count the episodes and see e.g., the sum of total costs.

These are more or less typical examples of sensitive medical data hiding, but much more complicated situation could also be described by the proposed approach. Thus a fine-grained access control to data can be defined in a relatively simple way.

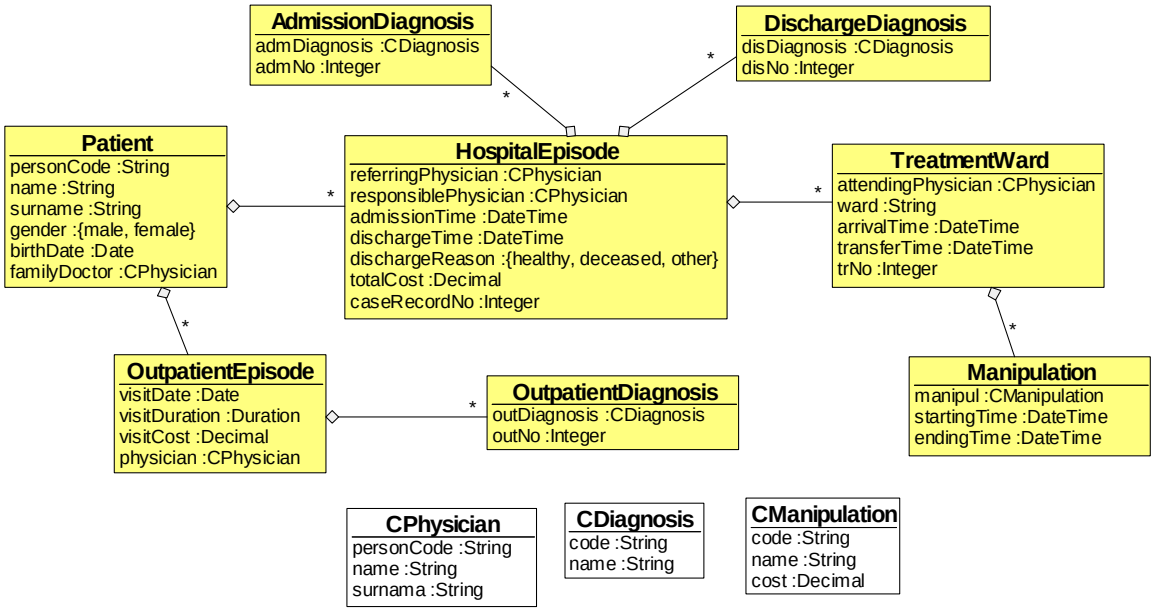


Fig. 2.1.2.2. The data ontology for the access rule examoples

We provide also a small example in Fig. 2.1.2.3 of a filled Access table for the schema definition table in Fig. 2.1.2.1. Passwords are fixed for all included users. However, there may be several rows containing the same password value – a user can

have several roles. The person code parameter value is specified here for the Responsible physician role where it is mandatory. The authentication for other roles here is based only on the password for the user. The ward number parameter value is present for the Ward manager role. Thus the corresponding access rule for all these roles can be evaluated after the user has successfully logged in – all required parameter values are present (several access rules must be evaluated if the logged-in user has several roles). No rows in this table must be created by administrator for External authentication or Open access case – there are no explicit parameters to be set in these cases. The example table contains two rows with the password "DmnPlm03". The corresponding user has thus the role of both the Responsible physician and Ward manager, accordingly for him the set of available hospital episodes (and related class instances) is the union of sets returned by both access rules. In this case the access rules are fully compatible, but it is the responsibility of administrator to define the rules so that no possible conflicts can occur for possible multi-role users (this may happen only in the WITHOUT clauses). The illustrated feature permits to emulate to a degree the role hierarchy present in RBAC models.

Authentication	Person code C	Role	Role parameter
ArpCrdV01	011040-11111	Responsible Physician	–
BrpHrtD02	301191-99999	Responsible Physician	–
CmnEmT01	–	Ward manager	12
DmnPlm03	–	Ward manager	07
DmnPlm03	280296-55555	Responsible Physician	–
EceDrBKS0	–	Hospital CEO	–

Fig. 2.1.2.3 An example of a filled access table

The extension of the current query language support system for access control seems also to be not very complicated. This is because the implementation in fact is based on a custom internal storage of the original relational data. This storage can be extended to include a user view specification facilities by adding the visibility attribute (relevant for the current query) to internal data classes and attributes.

Experiments on the existing query language implementation system without any access control have been performed, Fig. 2.1.2.4 shows the distribution of query answer times in the experiment on real hospital data. The vast majority of all queries executes in less than 0.3 seconds. Certainly, the inclusion of access rules in the implementation will slow down the execution. Experiments for the FULLSHOW option of the query language were also performed, in typical cases the execution time was less than 0.5 seconds. The actions to be performed for a FULLSHOW query are very similar to those to be performed for a FULLSELECT access rule evaluation. Therefore it is reasonable to assume, that the additional delay would also not exceed 0.5 seconds – that would be a completely acceptable performance.

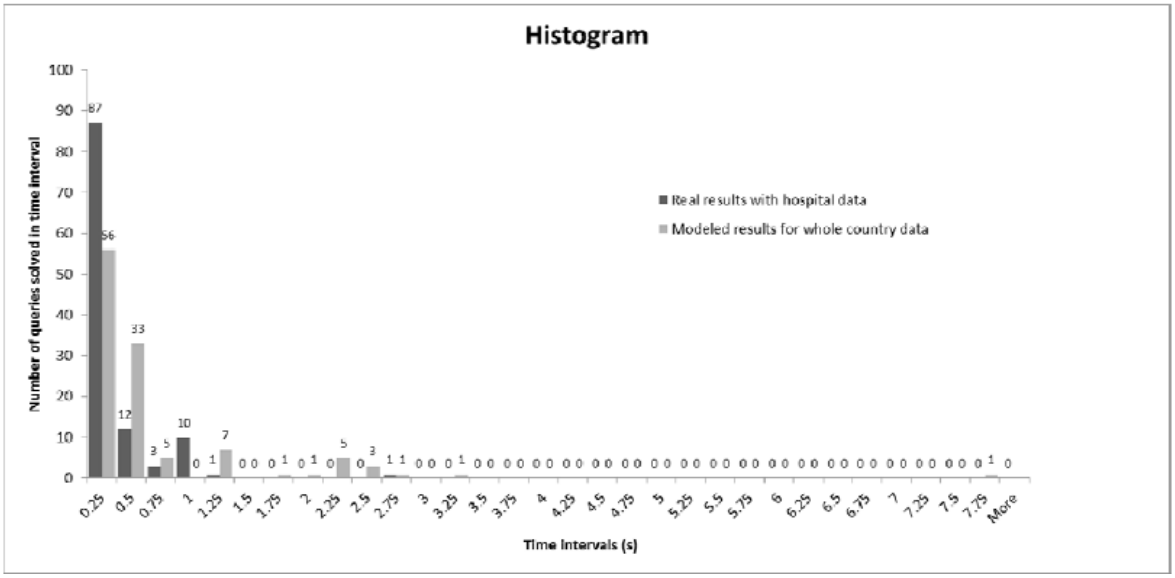


Fig. 2.1.2.4. Query execution times in an experiment

However these assumptions require an experimental check, which is planned for the next stage of the project. We expect that the delay caused by access control indeed will be small. It should be noted that in many practical data access systems the execution slowdown induced by access control is significant for more complicated access rules, frequently due to the fact that many access rules have to be evaluated to find out the access rights for a given user. Therefore our approach based on local rules is quite perspective.

2.1.3. Development of metamodel specialization methods and their application to building of domain specific language tools for web environment.

A new metamodeling method – the metamodel specialization method – has been developed. This method is based on standard UML facilities – class diagrams, class and association specialization and OCL constraints. An application of metamodel specialization method to building graphical DSL tools has been developed. This application results in a new kind of a platform for building DSL tools. In this platform at first a Universal metamodel for the chosen DSL tool domain (e.g., for graphical DSL modeling tools) is being created. A Universal engine for this metamodel is also built. Then any specific DSL tool from the chosen domain can be obtained by the metamodel specialization method. The proposed new approach differs from the existing traditional DSL tool building platforms by the feature, that instead of traditional metamodel instantiation we use the specialization of the universal metamodel. This permits to build a complete definition of the chosen DSL tool by adding appropriate OCL constraints. To compare, for traditional metamodel instantiation applications when building a more complicated DSL tool, as a rule it is necessary to dive into the internal implementation model of the corresponding universal engine, thus making the platform usage much more complicated. The main research results have been published in "A.Kalnins, J.Barzdins, Metamodel specialization for graphical modeling language support. // In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. ACM, pp.103-112, 2016." – in proceedings of the most prominent international conference in the area – Models 2016 (with paper acceptance rate 23.7%). Additional results have been published also in "A.Kalnins, J.Barzdins, Metamodel Specialization for DSL Tool Building. // In: G.Arnicans, V.Arnicane, J.Borzovs, L.Niedrite (Eds.), Databases and Information Systems, 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016, Proceedings, Communications in Computer and Information Science Vol. 615, Springer, pp.68-82, 2016. (indexed by SCOPUS)" and "A.Kalnins, J.Barzdins, Metamodel Specialization for Diagram Editor Building. // In: G.Arnicans, V.Arnicane, J.Borzovs, L.Niedrite (Eds.), Frontiers of AI and Applications, Vol. 291, Databases and Information Systems IX, IOS Press, pp. 87-100, 2016. (to be indexed SCOPUS), http://ebooks.iospress.com/volumearticle/45702". A graphical tool building platform for web environment, to a great degree based on the above-mentioned ideas has been developed as well and published in "A.Sprogis, ajoo: WEB Based Framework for Domain Specific Modeling Tools. // In: G.Arnicans, V.Arnicane, J.Borzovs, L.Niedrite (Eds.), Frontiers of AI and Applications, Vol. 291, Databases and Information Systems IX, IOS Press, pp. 115-126, 2016. (to be indexed SCOPUS), http://ebooks.iospress.com/volumearticle/45704"

2.1.4. Participation in SemEval-2016 competition with an improved version of the C6.0 classification algorithm adapted for Abstract Meaning Representation (AMR) information retrieval

We continued research in information retrieval and semantic parsing with an application of our earlier Semeval-2015 approach [1] to the formalism of Abstract Meaning Representation (AMR) [2]. AMR parsing extends the FrameNet micro-relations concept and attempts to build a semantic graph of all relations within a sentence.

We managed to achieve excellent AMR parsing accuracy, resulting in the first place in Task 8 of Semeval-2016 shared task competition [3]. In addition to integration of the C6.0 0 classifier with the AMR SMATCH [4] scoring tool to improve accuracy of the CAMR parser [5, 6], we implemented an ensemble with a character level sequence-to-sequence neural network model for semantic parsing with methods inspired by neural machine translation.

Exploration of these technologies also resulted in publications about applications of AMR in text summarization [7] and deep neural networks for Latvian tagging [8], and the development of three master's thesis on these technologies [9,10,11].

SMATCH extensions

We describe two extensions[1] to the original AMR smatch scoring script. These extensions do not change the smatch algorithm or scores produced, but they extract additional statistical information helpful for improving results of any AMR parser, as will be illustrated further.

Visual Smatch with C6.0 Classifier Rules

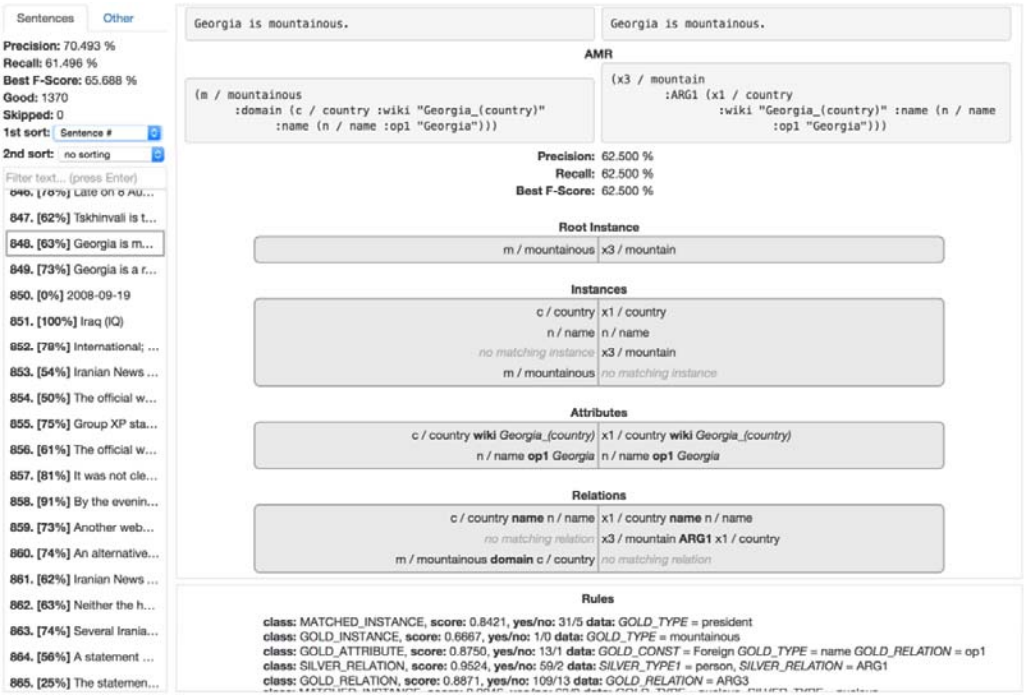
The original AMR smatch scoring metric produces as output only three numbers: precision, recall and F1. When developing an AMR parser, these three numbers alone do not reveal the actual mistakes in the AMR parser output (we call it *silver* AMR) when compared to the human-annotated *gold* AMR.

The first step in alleviating this problem is visualizing the mappings produced by the smatch algorithm as part of the scoring process. Figure 1 shows such smatch alignment visualization where gold and silver AMR graphs are first split into the edges, which are further aligned through variable mapping. The smatch metric measures success of such alignment – perfect alignment results in F1 score 100% while incomplete alignment produces lower scores.

The visualization in Figure 2.1.4.1 is good for manual inspection of incomplete AMR alignments in individual sentences. But it still is only marginally helpful for AMR parser debugging, because the data-driven parsers are expected to make occasional mistakes due to the training data incompleteness rather than due to a bug in the parser.

[1] Available at https://github.com/didzis/smatchTools

***Fig.* 2.1.4.1.** *Visual smatch with Rules. Left pane shows the document content and statistics. Right pane shows single sentence gold AMR (left) and silver AMR (right) along with smatch aligned instance, attribute, relation AMR graph edges. The bottom pane shows C6.0 classifier generated rules describing the common error patterns found in the document.*



Telling apart the repetitive parser bugs from the occasional training data incompleteness induced errors is not easy and to invoke the required statistical mechanisms we resorted to a rule-based C6.0 classifier [12,1], a modification of the legacy C4.5 classifier [13]. The classifier is asked to find most common patterns (rules) leading to some AMR graph edges to appear mostly in the gold, silver, or matched class after the smatch alignment. The bottom part of Figure 1 illustrates few such rules found by C6.0. For example, the second rule relates to the visualized sentence and should be read as "if the instance has type *mountainous*, then it appears 1 time in the gold graphs and 0 times in the silver graphs of the entire document". Similarly the third rule should be read as "word *Foreign* appears 13 times as *:op1* of *name* in the gold graphs, but only 1 time in the silver graphs of the entire document" – such 13 to 1 ratio likely points to some capitalization error in the parser pipeline. The generated rules can be sorted by their statistical impact score calculated as Laplace ratio $(p+1)/(p+n+2)$ from the number of correct p and wrong n predictions made by this rule.

Classifier generated rules were the key instrument we used to create a bug-fixing wrapper for the CAMR parser, described in section "CAMR Parser with Wrapper". We fixed only bugs triggering error-indicating-rules with the impact scores above 0.8, because Laplace ratio strongly correlates with the smatch score impact of the particular error.

Smatch Extension for Ensemble Voting

The original smatch algorithm is designed to compare only two AMR documents. Meanwhile CAMR parser is slightly non-deterministic in the sense that it produces different AMRs for the same test sentence, if trained repeatedly. Randomly choosing one of the generated AMRs is a suboptimal strategy. A better strategy is to use an ensemble voting inspired approach: among all AMRs generated for the given test sentence, choose the AMR which achieves the highest average pairwise smatch score

with all the other AMRs generated for the same test sentence. Intuitively it means that among the non-deterministic options we choose the "prevalent" AMR.

Multiple AMRs for the same test sentence can be generated also from different AMR parsers with substantially different average smatch accuracy. In this case all AMRs still can participate in the scoring, but weights need to be assigned to ensure that only AMRs from the high-accuracy parser may win.

AMR Parsers

We applied the smatch extensions described in the previous section to two very different AMR parsers.

CAMR Parser with Wrapper

We applied the debugging techniques from Section 2.1 to the best available open-source AMR parser CAMR[2]. The identified bug-fixes were almost entirely implemented as a CAMR parser wrapper[3] that runs extra pre-processing (normalization) step on input data and extra post-processing step on output data. Only minor modifications to CAMR code itself were made[4] to improve the performance on multi-core systems and to fix date normalization problems.

Our CAMR wrapper tries to normalize the input data to the format recognized well by CAMR and to fix some systematic discrepancies of annotation style between the actual CAMR output and the expected gold AMRs. The overall gain from our wrapper is about 4%.

The following normalization actions are taken during pre-processing step, together accounting for about 2% gain:

1. number normalization from a lexical (e.g. "seventy-eight"), semi-lexical (e.g. "5 million") or multi-token digital (e.g. "100,000" or "100 000") format to a single token digital format (e.g. "100000");

2. currency normalization from a number (any format mentioned in previous step) together with a currency symbol (e.g. "$ 100") to a single token digital number with the lexical currency name (e.g. "100 dollars");

3. date normalization from any number and lexical mix to an explicit eight-digit dash separated format "yyyy-mm-dd".

Small modifications had to be made to the baseline JAMR (Flanigan et al., 2014) aligner used by CAMR to reliably recognize the "yyyy-mm-dd" date format and to correctly align the date tokens to the graph entries (by default JAMR uses "yymmdd" date format that is ambiguous regarding century and furthermore can be misinterpreted as a number).

[2] https://github.com/Juicechuan/AMRParsing
[3] https://github.com/didzis/CAMR/tree/wrapper
[4] Modified CAMR at https://github.com/didzis/CAMR

The rules for date normalization were extracted from the training set semi-automatically using C6.0 classifier by mapping date-entities in the gold AMR graphs and corresponding fragments in input sentences.

Additionally, all wiki edges were removed from the AMR graphs prior to training, because CAMR does not handle them well; this step ensures that CAMR is trained without wiki edges and therefore will not insert any wiki entries in the generated AMR. Instead, we insert wiki links deterministically during the post-processing step.

During post-processing step the following modifications are applied to the CAMR parser generated AMR graphs, together accounting for about 2% gain:

1. nationalities are normalized (e.g. “Italian” to “Italy”);

2. some redundant graph leafs not carrying any semantic value are removed (e.g. “null-edge”);

3. wiki links are inserted deterministically next to “name” edges using gazetteer extracted from the training data and extended with the complete list of countries and nationalities (wiki value is selected based on the parent concept and content of the “name” instance);

About 1% additional gain comes from the observation that CAMR parser suffers from overfitting: it achieves optimal results when trained for only 2 iterations and with empty validation set.

Neural AMR Parser

For neural translation based AMR parsing we used simplified AMRs without wiki links and variables. Prior to deleting variables, AMR graphs were converted into trees by duplicating the co-referring nodes. Such AMR simplification turned out to be nearly loss-less, as a simple co-reference resolving script restores the variables with average F1=0.98 smatch accuracy.

We trained a modified TensorFlow seq2seq neural translation model[5] with attention [14,15,16] to “translate” plain English sentences into simplified AMRs. For the test sentence in Figure 1 it gives following parsing result:

```
(mountain-01
   :ARG1 (country
     :name (name :op1 "Georgia"))
```

Apart from a missing bracket this is a valid (although slightly incorrect) simplified AMR. Note that this translation has been learned in “closed task” and “end-to-end” manner only from the provided AMR training set without any external knowledge source. This explains overall lower accuracy of the neural AMR parser compared to CAMR, which uses external knowledge from wide coverage parsing models of BLLIP[6] parser [17]. The neural AMR parser accuracy is close to CAMR accuracy for short sentences up to 100 characters, but degrades considerably for longer sentences.

[5] https://github.com/didzis/tensorflowAMR
[6] https://github.com/BLLIP/bllip-parser

We optimized TensorFlow seq2seq model hyper-parameters within the constraints of the available GPU memory: 1 layer or 400 neurons, single bucket of size 480, each input and output character tokenized as a single "word", vocabulary size 120 (number of distinct characters), batch size 4, trained for 30 epochs (4 days on TitanX GPU).

Operating seq2seq model on the character-level [18,19,20] rather than standard word-level improved smatch F1 by notable 7%. Follow-up tests [21] revealed that character-level translation with attention improves results only if the output is a syntactic variation of the input (as is the case for AMR parsing), but for e.g. English-Latvian translation gives inferior results due to attention mechanism failing to establish character-level mappings between the English and Latvian words.

Results

Table 2.1.4.1 shows smatch scores for various combinations of parsers and thus quantifies the contributions of all methods described in this paper. We improved upon CAMR rather than JAMR parser due to better baseline performance of CAMR, likely due to its reliance on the wide coverage BLLIP parser.

Parser	**F1 on LDC2015E86 test set**	**F1 on the official eval set**
JAMR (baseline)	0.576	
CAMR (baseline)	0.617	
CAMR (no valid.set, 2 iter.)	0.630	
Neural AMR (word-level)	0.365	
Neural AMR (char-level)	0.433	0.376
CAMR+ wrapper	0.667	0.616
Ensemble of CAMR+ wrapper and NeuralAMR (char-level)	0.672	**0.620**

The CAMR parser wrapper is the largest contributor to our results. The weighed ensemble of 3 runs of CAMR+wrapper and 1 run of neural AMR parser) gave an additional boost to the results. Including the neural AMR parser in the ensemble doubled the gain – apparently it acted as an efficient tiebreaker between the similar CAMR+wrapper outputs.

Table 2.1.4.1: Smatch scores.

References

1. Guntis Barzdins, Peteris Paikens, Didzis Gosko. Riga: from FrameNet to Semantic Frames with C6.0 Rules. Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 959–963, Denver, Colorado, June 4-5, 2015, Association for Computational Linguistics.

2. Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider, 2013. Abstract Meaning Representation for Sembanking. Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pages 178–186. Association for Computational Linguistics.

3. Guntis Barzdins, Didzis Gosko. RIGA at SemEval-2016 Task 8: Impact of Smatch Extensions and Character-Level Neural Translation on AMR Parsing Accuracy. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, pp. 1143-1147. URL https://aclweb.org/anthology/S/S16/S16-1176.pdf

4. Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2013, pages 748–752. Association for Computational Linguistics.

5. Chuan Wang, Nianwen Xue, and Sameer Pradhan. A transition-based algorithm for AMR parsing. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pages 366–375. Association for Computational Linguistics.

6. Chuan Wang, Nianwen Xue, and Sameer Pradhan. Boosting Transition-based AMR Parsing with Refined Actions and Auxiliary Analyzers. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), 2015, pages 857-862. Association for Computational Linguistics.

7. N. Gruzitis and G. Barzdins. The role of CNL and AMR in scalable abstractive summarization for multilingual media monitoring. Controlled Natural Language, Controlled Natural Language 5th International Workshop, CNL 2016, Davis, Brian, Pace, Gordon J., Wyner, Adam (Eds.), LNAI, Volume 9767, pp. 127-130, Springer 2016. doi = "10.1007/978-3-319-41498-0"

8. Peteris Paikens. Deep Neural Learning Approaches for Latvian Morphological Tagging. Frontiers in Artificial Intelligence and Applications, Volume 289: Human Language Technologies – The Baltic Perspective, I. Skadiņa and R. Rozis (Eds.). IOS Press, 2016, pp 160-166. DOI 10.3233/978-1-61499-701-6-160 URL http://ebooks.iospress.nl/volumearticle/45531

9. Artūrs Znotiņš. Jēdzientelpas un to pielietojumi. 2016. Master's thesis, University of Latvia.

10. Roberts Darģis. Liela apjoma datu kopu klasterēšanas algoritmi. 2016, Master's thesis, University of Latvia.

11. Reinholds Pīrāgs. Automātiska teksta konspektēšana izmantojot jēdzientelpu. 2016, Master's thesis, University of Latvia.

12. Guntis Barzdins, Didzis Gosko, Laura Rituma, and Peteris Paikens. Using C5.0 and Exhaustive Search for Boosting Frame-Semantic Parsing Accuracy. In: Proceedings of the 9th Language Resources and Evaluation Conference (LREC-2014), 2014, pages 4476-4482.

13. Ross J. Quinlan. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

14. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael

Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. *Google Research whitepaper*. Software available from tensorflow.org.

15. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In NIPS, 2014.
16. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR'2015.
17. Eugene Charniak and Mark Johnson. 2005. Coarse- to-fine n-best parsing and maxent discriminative reranking. In Proceedings of the 43rd Annual Meet- ing on Association for Computational Linguistics, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.
18. Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015.
19. Junyoung Chung, Kyunghyun Cho, Yoshua Bengio. 2016. A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation. arXiv preprint arXiv:1603:06147
20. Minh-Thang Luong and Christopher D. Manning. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. arXiv preprint arXiv:1604.00788
21. Guntis Barzdins, Steve Renals and Didzis Gosko. 2016. Character-Level Neural Translation for Multilingual Media Monitoring in the SUMMA Project. In: Proceedings of the 10th Language Resources and Evaluation Conference (LREC-2016)

2.2. Development of approaches, methods and algorithms for knowledge structure transformations and analysis, and design methodology of semantic network services

2.2.1. Further improvement of functionality of intelligent structural modeling tool I4S for multicriterial assessment of importance of elements for knowledge structures of different types and granularities

2.2.1.1. Description of method for multicriterial assessment of importance of elements of knowledge structures of different types and granularities

Introduction

The functionality of the tool I4S is extended by applying the results of research done during this stage of the project. The objective is to investigate whether the importance of structural elements is invariant if changes of granularity level of knowledge representation take place.

Let $G(V, Q)$ be a system's aggregated knowledge structure which is represented by model of morphological structure (directed graph). Each node corresponds to one subsystem, for example, a subsystem of lubrication or a subsystem of cooling in case of internal combustion engine, or a particular study course in case of study programme.

The initial aggregated model is transformed into a model which has deeper level of granularity. Transformation is based on graph homomorphism (see Fig. 2.2.1.1).

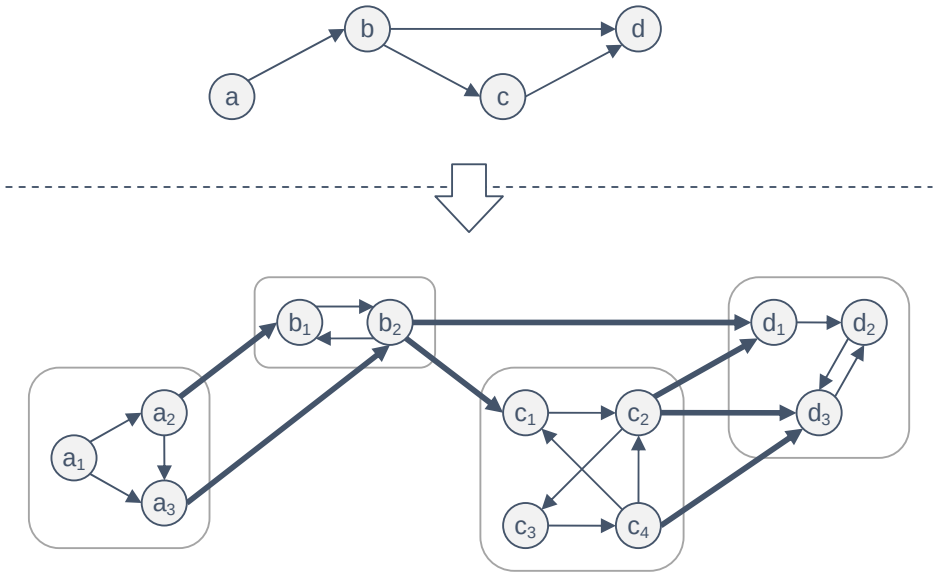


Fig. 2.2.1.1. Transformation of aggregated initial graph into homomorphic graph with deeper level of granularity

The result of transformation is a digraph $G_g(V_g, Q_g)$ where each node of the initial graph is replaced by a subset of nodes representing knowledge of deeper granularity.

It is worth to point that homomorphism of graphs maintains the basic structural characteristics (see the bold arcs in Fig. 2.2.1.1).

Fundamental principles for development of method

Analysis of importance of elements of different knowledge structures is rather widespread task nowadays. As a rule, this task belongs to problems of structural analysis of graphs because graph theory is the traditional language of structures. Analysis of related works in this field manifests that especially in case of large graphs, for example, graphs representing the structure of links between homepages or the structure of social networks, the importance of elements is assessed only on the basis of local connectivity, that is, using only in-degrees and out-degrees of nodes, or, in other words, direct relationships. As a consequence, a significant part of information is lost. In structural modelling approach [1], [2] the importance of structural elements (graph nodes) is weighted using three criteria: the local connectedness (in- and out-degrees of nodes of digraphs), the global connectedness (the number of paths between inputs and outputs of structure as well as the number of cycles which contain a node), and the causal connectedness – the number of elements in the reachability components of nodes which represent the consequences caused if some characteristics of nodes are missing.

The abovementioned criteria are used for ranking each structural element (node of digraph).

The value of the first parameter $p^1(x_i)$ of each node x_i is computed using the following equation:

$$p^1(x_i) = \deg^-(x_i) + \deg^+(x_i) \tag{1}$$

where $\deg^-(x_i)$ and $\deg^+(x_i)$ are in-degree and out-degree, correspondingly.

All nodes are ranked in accordance with the rule: the greater is the value of $p^1(x_i)$, the higher rank (place) R^1 has the node, that is, the node (nodes) with maximum value of $p^1(x_i)$ takes the first place (is ranked as the first).

The value of the second parameter $p^2(x_i)$ is computed using the following equation:

$$p^2(x_i) = \sum_{i=1}^{p}\sum_{j=1}^{q} P_{ij} + \sum_{x_i \in C_r} C_r \tag{2}$$

where P_{ij} is a path between input i and output j containing the node x_i, and C_r is a cycle containing the node x_i.

The principle of ordering of nodes is the same as for parameter $p^1(x_i)$, and the node with the maximum value of $p^2(x_i)$ has the highest rank R^2.

The value of the third parameter $p^3(x_i)$ is computed using the following equation:

$$p^3(x_i) = |ReC(x_i)| \tag{3}$$

where $ReC(x_i)$ is the reachability component of node x_i, and $|ReC(x_i)|$ denotes the number of its nodes (the selected node and all its descendants).

The principle of ordering of nodes in accordance with $p^3(x_i)$ is the same as for $p^1(x_i)$ and $p^2(x_i)$, that is, the node with the maximum value of $p^3(x_i)$ has the highest rank R^3.

The next step is to calculate the sum of ranks for each node x_i:

$$R_\Sigma(x_i) = R^1(x_i) + R^2(x_i) + R^3(x_i)\ . \tag{4}$$

In essence, the $R_\Sigma(x_i)$ is the sum of places taken by the node x_i according to parameter values $p^1(x_i)$, $p^2(x_i)$, and $p^3(x_i)$. That is the reason why the total rank $R_{\text{total}}(x_i)$ of the node x_i is obtained following the principle that the less is the value of $R_\Sigma(x_i)$, the higher is $R_{\text{total}}(x_i)$.

The final step which is needed to compute the importance of structural element (node) x_i is usage of the following equation:

$$SI(x_i) = 1 + \frac{1 - R_{\text{total}}(x_i)}{R_{\max}} \tag{5}$$

where $SI(x_i)$ is the structural importance of the node x_i, $R_{\text{total}}(x_i)$ is the total rank of the node x_i and $R_{\max}$ is the maximum value of R_{total} in the given graph.

Description of method for evaluation of importance of elements of knowledge structures at deeper level of granularity

The developed method is modification of the method described in the previous section. The modified method is appropriate for evaluation of importance of elements of knowledge structures which represent systems at a deeper level of granularity. In general, when more knowledge about the system is acquired, it is possible to transform the aggregated model into hierarchically ordered models by replacing a node at the previous hierarchical level with a subset of interrelated nodes at the next hierarchical level. The method will be described using only two models of morphological structure represented by a digraph $G(V, Q)$, which is called an aggregated model, and by a digraph $G_g(V_g, Q_g)$, which is a model representing the system at deeper level of granularity (see Fig. 2.2.1.1).

The developed method consists of five stages:

Stage 1. For each node x_i of the graph $G(V, Q)$ compute the values of parameters p^1, p^2, and p^3, determine the ranks R^1, R^2, R^3, R_Σ, and R_{total} according to equations (1–4). Use R_{total} to compute the structural importance (SI) of each element of aggregated structure according to equation (5).

Stage 2. For each node of the graph $G_g(V_g, Q_g)$, compute all values which are listed in Stage 1.

Stage 3. Define the mapping of each node of graph $G(V, Q)$ into a set of nodes of graph $G_g(V_g, Q_g)$ using the notions of boundary and interior [1] of each corresponding subset of nodes.

Stage 4. Sum up values of SI of nodes which belong to each particular subset of nodes and divide it by the number of nodes in the subset.

As a result, the values of SI characterize the importance of subset of nodes of a knowledge structure representing a system at a deeper level of granularity. It is straightforward that the values of importance of subsets of nodes found at this stage differ from the values of importance of nodes of aggregated knowledge structure. At the same time it is worth to point that the relative ordering is the same due to the fact of homomorphism of digraphs $G(V, Q)$ and $G_g(V_g, Q_g)$.

Stage 5. If more detailed analysis is needed, the importance of nodes within each of subsets may be carried out using computational operations described at Stage 1.

Description of algorithm

In the developed algorithm, in fact, few additional steps are added in comparison with the algorithm implemented in the tool I4S.

Algorithm:

1. Using the tool I4S compute the values of parameters p^1, p^2, and p^3, determine the ranks R^1, R^2, R^3, R_Σ, and R_{total} according to equations (1–4). Use R_{total} to compute the structural importance of each element of aggregated knowledge structure (a digraph $G(V, Q)$) according to equation (5).
2. Using the tool I4S compute the same values for each element of knowledge structure at deeper level of granularity (a digraph $G_g(V_g, Q_g)$).
3. Define the mapping for each node $x_i \in V$ into a set of nodes of digraph $G_g(V_g, Q_g)$. Let denote such set as $S^i(x_j)$, $j = 1, \ldots, m$.
4. Compute the importance of each subset of nodes, that is, find

$$SI_{S^i} = \sum_{j=1}^{m} \frac{SI(x_j)}{|n_i|} \tag{6}$$

   where $|n_i|$ is the number of nodes in the subset $S^i(x_j)$. The value of $SI_S{}^i$ corresponds to the importance of subset $S^i(x_j)$ in the knowledge structure of deeper granularity.
5. For all subsets $S^i(x_j)$, where $i = 1, \ldots, n$; n – the number of nodes of $G(V, Q)$, repeat the Step 1.

2.2.1.2. Development of formal method for evaluation of knowledge structure complexity from the systems viewpoint

The research is focused on concept maps as one kind of knowledge structures. The work extends the approach, the development of which started during the second stage of the project. The objective is to develop a formal method for evaluation of concept map complexity. The latter is necessary for more discriminative estimation of task difficulty and development of more accurate scoring system used in already developed intelligent concept map-based knowledge assessment system IKAS (an overview of the system can be found in "Grundspenkis J. Historical Retrospection on Success and Failures during the Development of Concept Map Based System IKAS. Proceedings of the 7th International Conference on Concept Mapping, Tallinn, Estonia, September 5-9, 2016, Vol. 2, pp. 113-119").

For evaluation of complexity of concept maps, it is proposed to use four aspects applied for estimation of systems complexity – the number of system's elements and relationships between them, the attributes of systems, their elements, and relationships, and the organizational degree of systems. During the current stage of research, the correspondence of complexity criteria between systems and concept maps is defined more precisely and the set of criteria is expanded. For estimation of concept map complexity for attribute graphs, a new formula is given. A modified equation is proposed to evaluate the organizational degree of a concept map by calculation of the complexity of structure of the concept map. New criteria, namely, the degree of centralization of structure and the relative weights of hierarchy levels are introduced, and corresponding equations are proposed. Besides, two novel methods are developed, namely, ordering of concept maps according to their complexity and determination of structural importance of concepts are worked out and validated for two kinds of incoming trees. More information on achieved results may be found in "Grundspenkis J. Towards the Formal Method for Evaluation of

Concept Map Complexity from the Systems Viewpoint. Databases and Information Systems IX, G. Arnicans et al. (Eds.), Frontiers in Artificial Intelligence and Applications, Vol. 291. IOS Press, 2016, pp. 341-354".

2.2.1.3. Assessment of element importance in knowledge structures of different types and granularity using various criteria

During this stage the research defined in the title of this subsection has been started. The task is to verify the hypothesis given by some researchers [3], [4] that ranks of elements are the same regardless of changes of lengths of outgoing paths from nodes of digraphs. Moreover, the hypothesis declares that there is no need to analyse outgoing paths with length greater than 4. Testing of this hypothesis is done using the developed repository of knowledge structure models. First results obtained from testing of very small number (only 10) of knowledge structures contradict the abovementioned hypothesis, showing that ranks of elements change significantly if paths of length 3 are counted comparing with results when paths of length 4 are considered. It is foreseen to continue this direction of research during the next stage of the project.

2.2.1.4. Possibilities to combine different knowledge structures used in distributed artificial intelligence

The objective of this research is to develop formal algorithms for effective mutual transformation of different knowledge schemes. This work is planned during the next stage of the research.

Introduction

In distributed artificial intelligence systems many different knowledge representation schemes can be used. It is well known that there are four knowledge representation schemes (based on logic, production rules, network, and structured ones) [5]. In literature devoted to artificial intelligence rather frequently there is a statement that for experienced expert it is easy to transform one knowledge representation scheme into another. It seems to be true if only logical schemes and production rules are considered. The problems arise when mutual transformations between all schemes are needed. Especially it is the case with network and structured schemes, which are the focus of research at the current stage.

Semantic networks

Semantic networks are one of declarative knowledge representation schemes. It is a graph which nodes represent facts, while the edges show the relationships between displayed facts. Semantic network edges are labeled and oriented what makes it easy to understand and define the relationships between the facts and objects. The semantic networks demonstrate concepts and links between them. The nodes of semantic network can represent not only the facts, but also objects, their properties, actions, and events – actually a semantic network can represent knowledge about any realities in the world. The semantic network can represent only first order logics; therefore it is unable to represent logical operations like conjunction, negation, disjunction [6].

Edges of the semantic network are able to represent different types of relations [6]:

- class–superclass – *Is a* relation;
- instance–class – *Is an instance* of relation;
- part–whole or *Is a part of* relation;
- object–its attribute – *Property* (*has)* relation;

- attribute–its value – *Value* relation;
- different linguistic relations.

Last group of relations shows that the semantic networks have not strictly defined formal semantics. Relation *Is* is related to the inheritance in semantic networks, which facilitates network design process, avoiding overloading the network with unnecessary relationships and helping to structure network, because this way all the objects of a particular class inherit its superclass information. Inheritance mechanism is related to one of the semantic network problems – concurrent inheritance from several classes – because class and its superclass may have conflicting values of the same property.

Below is an example (Fig. 2.2.1.2) of a semantic network that represents the fragment of dog show exhibition knowledge. Accountant Alina has a dog which breed is wire-haired dachshund and name Freud. It comes from Bergberry kennel, owned by Ieva. Freud is 4 years old; he performs in the champion class, because in the past he has already got Latvian Junior Champion title. From the same kennel comes another wire-haired dachshund from open class – Faust, who is Freud's brother, who is six years old, but his owner is a dentist Daiga.

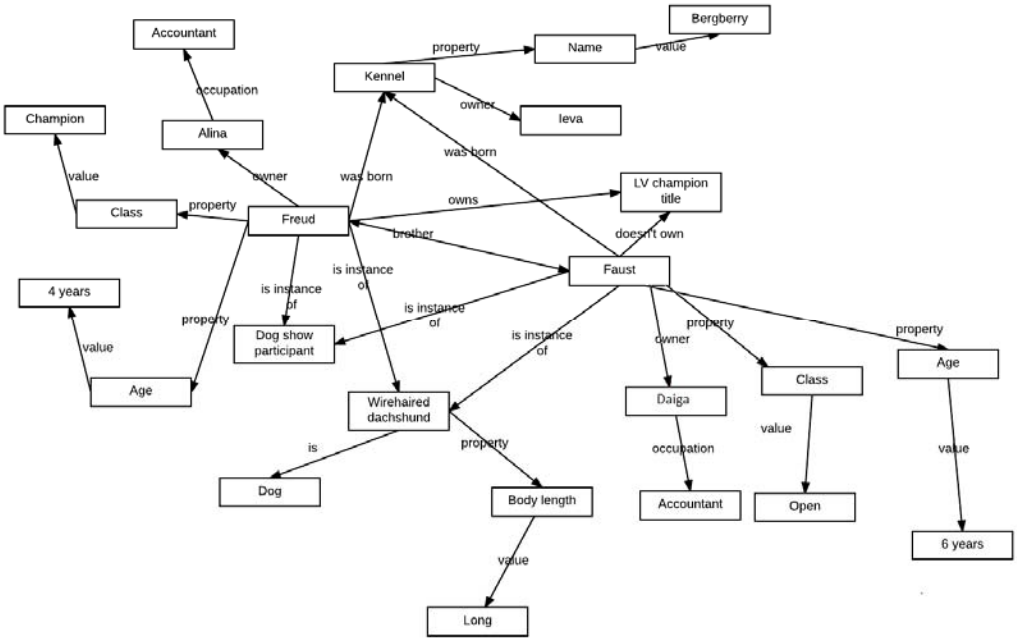


Fig. 2.2.1.2. Semantic network – dog show

Conceptual graphs

Another knowledge representation scheme, without the strictly formalized semantics are conceptual graphs. Conceptual graph is finite and bipartite connected graph, which represents one statement. In the bipartite graph all the nodes are divided into two distinct and non-empty sets and there are edges only between the nodes of different sets. In the context of conceptual graph, these two sets of nodes are the set of concepts and set of conceptual relations, so then edges are allowed only from the concept to the relation or in the opposite direction – from the relation to the concept. Relations are represented as the nodes, so that edges of the conceptual graph are not labeled, moreover conceptual relation can include several edges, the number of which is called the valence of relationship, so the conceptual graph allows you to represent

not only the first-order logics (first-order logics – one outgoing edge, second-order logics – one incoming and one outgoing edges, the third-order logics – two incoming and one outgoing). In each case outgoing edge can be only one. Conceptual graphs allow you to define the signature of the relationship, which defines restrictions on the types that can be connected by a given relation [7].

The concept node may contain information about the concept type and particular instance of this type (if necessary also quantifier can be included). If it is not possible to specify a particular instance, then the node can store only type name or unnamed individuals, whose names can be obtained by inference mechanism.

Conceptual graphs like semantic networks allow realizing the inheritance. It can be achieved by defining the type hierarchy and graph restriction and matching operations. In general, knowledge base based on the conceptual graphs, consists of set of graphs, the concept type hierarchy and catalog of instances [7].

Conceptual graphs allow to represent the conjunctions, disjunction and negation operations. To realize the negation operation it is necessary to group the conceptual graph, to which will be applied the negation, by the expression node. Then *No* conceptual relation is applied to this expression node. The conjunction is realized by grouping several conceptual graphs in the expression node. Disjunction is represented with the conjunction and negation.

To create a conceptual graph, based on previously developed semantic network, the network has to be divided into separate statements. To do this, it is necessary to analyze network nodes and associated edges, starting with the nodes without the outgoing edges.

First of all it is necessary to define referents and their classes (node, which follows after the *Is instance of* relation), because the concepts in the conceptual graphs contain both the instance and its type. It is one of the problems of semantic network transformation to the conceptual graph, because in the semantic network individuals can formally exist without their types. For example, in the semantic network that was described above nodes Alina, Daiga and Ieva have not any related types, but despite this, they are supposed to be individuals, not the attributes or any other class. In order to avoid such a situation, it would be better to define type for each individual in the semantic network.

It has been already described that some difficulties can arise while trying to represent inheritance from several types at the same time in the semantic networks. But formally semantic networks allow to represent such situations when one object is an instance of more than one class. For example, Freud is the instance of a wirehaired dachshund class and the instance of show participant class. So it is not clear which class should be shown in a concept node (Fig. 2.2.1.3).

Fig. 2.2.1.3. Choosing type of object

It would be logical to choose a class depending on the context. For example, while dealing with a statement *Freud is 4 years old*, it is more important to know that Freud is a dachshund than the fact that he is a participant in a dog show. On the other hand,

if the statement is *Freud's class is a champion class* (Fig. 2.2.1.4) then it would be logical to note that Freud is a participant in a dog show. Computers, unlike people, have difficulties with analyzing the context of a statement.

Fig. 2.2.1.4. Choosing type of object

To perform conceptual graph actions, a type hierarchy must be defined first. That's why after finding a semantic network node that corresponds to a certain class (a node that follows an *is an instance of* relation), it is necessary to check whether or not it is followed by an *is a* relation. For example, previously developed semantic network's nodes Freud and Wirehaired dachshund are connected with an *is an instance of* relation, while the node Wirehaired dachshund is followed by an *is a* relation connecting this node to a Dog class. Type hierarchy Wirehaired dachshund <= Dog can be derived from his chain.

Another big problem is to distinguish objects from relations and properties. Defining which node from a semantic graph becomes a separate concept, and which – a concept type, is another problem. If a node in a semantic network is preceded by a *property* relation, it will be combined with a node preceded with a *value* relation.

For example, it would be reasonable to transform a statement *Wirehaired dachshund Freud is 4 years old* into the conceptual graph shown on the Fig. 2.2.1.5.

Fig. 2.2.1.5. Property and its value in a conceptual graph

Slightly more complicated situation occurs while trying to depict the information about the kennel and its name. Similarly to the previous example it can be done in a way shown in a Fig. 2.2.1.6. But in this case the kennel is perceived as a class, and not as a specific kennel. So, it would be logical always to combine a name with a kennel, thus getting the specific object.

Fig. 2.2.1.6. Name attribute in a conceptual graph

Problems arise while transforming this conceptual graph back into a semantic network, because the result of this transformation (shown in a Fig. 2.2.1.7) does not correspond to the original semantic network and causes multiple inaccuracies.

Fig. 2.2.1.7. Instance in a conceptual graph

Adding the information about Freud being born in a kennel complicates the problem further, because it is not clear that Freud was born specifically in the Bergberry kennel (Fig. 2.2.1.8).

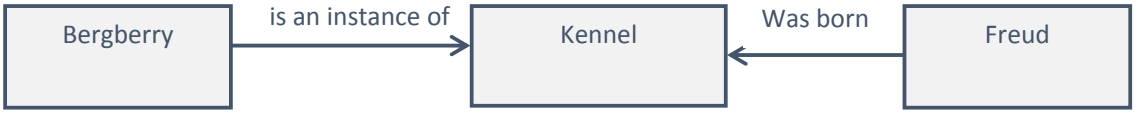


Fig. 2.2.1.8. Inaccuracies that have occured

While transforming conceptual graphs into semantic networks, one should take into account that only first-order relations can be represented in a semantic network, but logical operations, like conjunction, disjunction and negation, can't be represented in a semantic network at all. To represent a negation in a semantic network, negation should be added to the relation itself. For example, in order to transform the graph shown in a picture below, negation should be added to the *owns* relation, making it a *doesn't own* relation (Fig. 2.2.1.9).

While transforming semantic network to conceptual graph, negation should be excluded from negative relation and moved to a separate node. This means that linguistic analysis should be performed – *does not* has to be separated from verb.

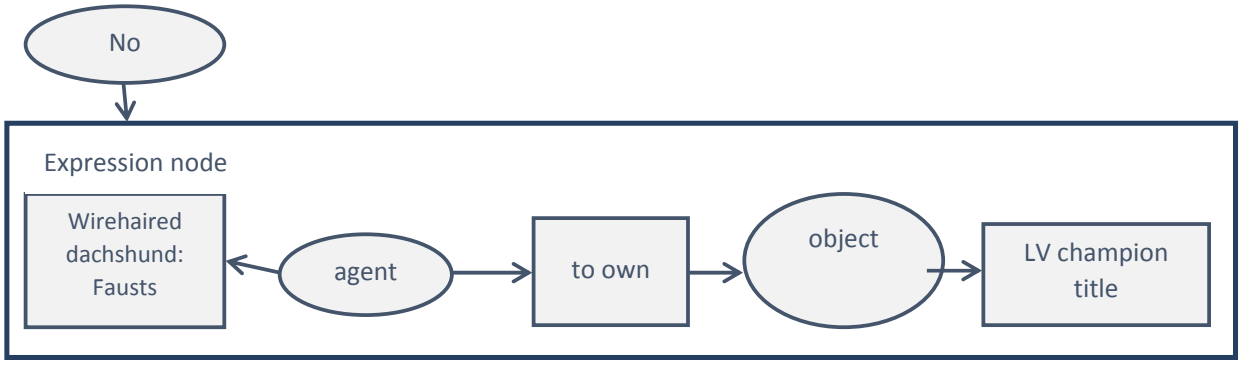


Fig. 2.2.1.9. Negation in a conceptual graph

Frames

Frames are static data structure that allows to represent typical knowledge about the object in a structured way. Frames have slots that contain data about the properties of frame and its relations to other frames. Inheritance is widely used in frames – it is realized by the frames of three types – class frame, subclass frame and instance frames. It is possible to define relationships of three different types between frames – generalization (*Is instance of* relation), aggregation (*A part of* relation) and the association. Frames can be expanded by adding facets, which allow you to define additional constraints (value type restriction or other restrictions) and adding the actions that can be performed. Actions can be divided into procedures and methods. Methods are actions that are automatically triggered by accessing frame and its slots [8].

Performing transformation of semantic network to the system of frames should start with defining classes and subclasses of an object (by identifying nodes that follow after relation *Is instance of*) and checking *Is* relation after the node that was found. This way it is possible to define classes and subclasses, each of them is then

implemented by separate frame. Then the *Property* nodes of the corresponding class node have to be analyzed and added as frame slots (Fig. 2.2.1.10).

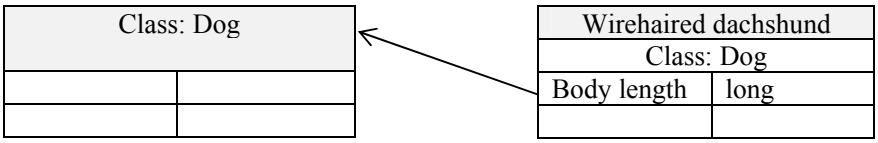


Fig. 2.2.1.10. Class and subclass frame

After classes and subclasses with their properties were identified, instance frames and their slots have to be defined. Then similar to the case of classes, properties have to be added as slots (Fig. 2.2.1.11).

Freud	
Class: Wirehaired dachshund	
Body length	long
Age	4 years
Class	champion

Fig. 2.2.1.11. Instance frame

After properties are added, other relations of an instance have to be analyzed. Some of them can be added to an instance frame as slots, too; for example, the owner of dog and kennel (Fig. 2.2.1.12).

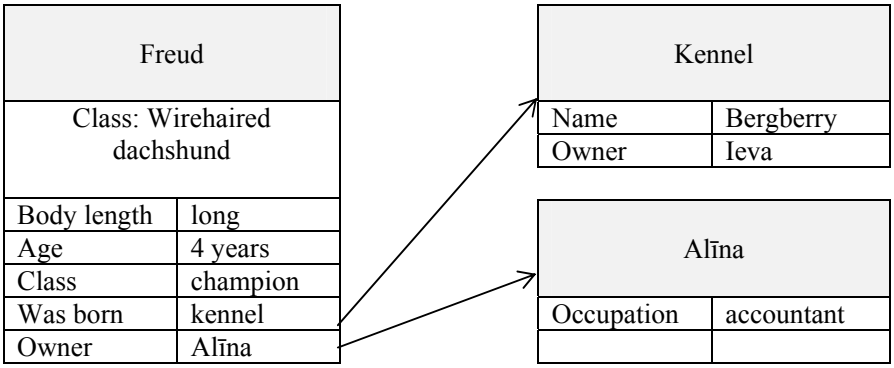


Fig. 2.2.1.12. Slots examples

In case of using frames there is also a problem if one object belongs to several classes at the same time.

References

1. Grundspenkis J. Structural Modelling of Complex Technical Systems in Conditions of Incomplete Information: A Review. In: Modern Aspects of Management Science, No 1. Riga, Latvia, 1997, pp. 111-135.
2. Grundspenkis J. Reasoning Supported by Structural Modelling. Intelligent Design, Intelligent Manufacturing and Intelligent Management. K. Wang and H. Pranevicius (Eds.), Lecture notes of the Nordic-Baltic Summer School on

Applications of AI to Production Engineering. Kaunas University of Technology Press, Technologija, 1999, pp. 57-100.
3. Nechiporenko V.I. Strukturnyj Analiz Sistem (in Russian). Moscow, Sovetskoe radio, 1977.
4. Nikolayev V.I., Bruk V.M. Sistemotehnika: Metodi i Prilozeniya (in Russian). Masinostroyeniye, Leningrad, 1985.
5. Luger G.F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison Wesley, 2005.
6. John F. Sowa. Semantic Networks. Stuart C Shapiro (Ed.), Encyclopedia of Artificial Intelligence. Wiley, 1992.
7. Chein M., Mugnier M.-L. Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer, 2009.
8. Marvin Minsky. A Framework for Representing Knowledge. MIT-AI Laboratory Memo 306, June 1974.

2.2.2. Development, merging and application of knowledge structure models in ontology- and rule-based decision making

During accomplishment of this research activity several studies have been done on knowledge structure models and their applications in the decision making, including affective tutoring systems, intelligent agents and multi-agent systems, as well as development of a repository for knowledge structure models that stores developed knowledge structures.

Development and analysis of different types of systems is essential to the creation of classification of knowledge structures. Qualitatively described and represented knowledge structure can be considered as a system model. System model describes the system from different viewpoints, allows understand and analyse its structure, functioning and behaviour, as well as to assess and carry out appropriate solutions regarding real-world system and its operations. One of the approaches which supports domain based, partly formal system representation is Structural Modelling [1]. Structural modelling studies the topology of models and defines a continuous and unified view on the research system where element and model visualization is designed taking into account such aspects as structure, functions, and behaviour, as well as deep causal knowledge and modelling purpose.

2.2.2.1. Building a repository of knowledge structures (models of systems)

Within the current step of the task a repository of knowledge structures – models was built. This repository was made using developed intelligent system I4S (also IFS) which supports knowledge acquisition and representation based on SM approach. The functionality of I4S ensures that knowledge structures (models) of systems can be changed, shared and reused. A repository serves as a central part where these models (knowledge structures) of different types of systems are stored (see Fig. 2.2.2.1).

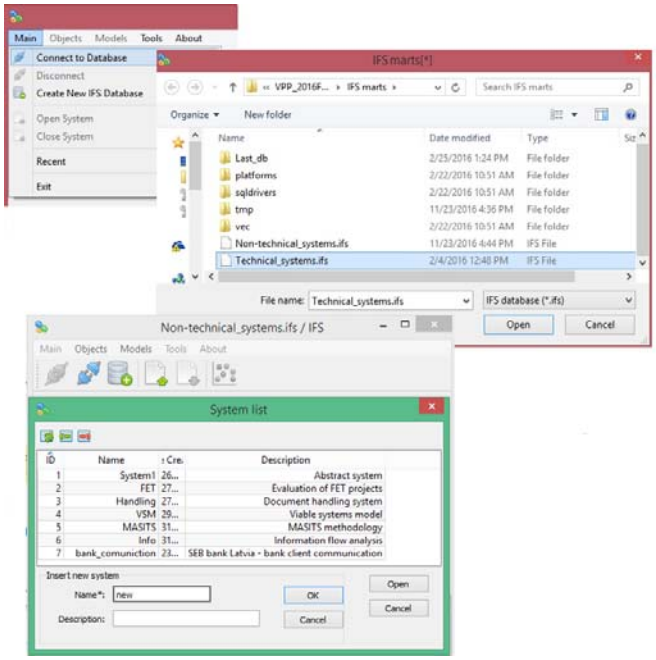


Fig. 2.2.2.1. A repository of knowledge structures (models) of different types of systems

The input of knowledge structures is made according to the developed methodology [2]. On the bases of once entered knowledge there is a possibility to generate different models in I4S (there is a submodule "Models") [3], which is important for qualitative and quantitative analysis of systems. Working with the system I4S, as during the input of knowledge as also during analysis of systems, expert's main task is to create a formal representation of the research system, including aspects that are relevant to structural modelling [1]: what system is viewed, what objects and relationships exist in the system, what are the properties and behaviour of the system and objects. Any system and its parts has a definite structure, that characterise its composition [4, Ch. 1]. Parts of the system may be in different sizes and can be either homogeneous – those who do not have different characteristics and heterogeneous – with different elements and/or structural properties. The structure is the relationship between the parts that together with identity of parts form a whole, taking into account the fact that between parts exists a certain order. Interactions and relationships between parts of the system are as important as the parts themselves. Interactions form a certain organization in the system. The organization is defined as a system feature, which is characterized by a structure that is purposefully created to perform certain functionality. It is worth to note that the structure of the system remains relatively stable over time; here is meant the structure that complies with the system organization. If the organization of a system stays invariant, while the structure of the system changes, then system remains the same and doesn't lose its identity. The organization of a system defines it as a unity in any space, while its structure constitutes it as a concrete entity in the space of its components.

System I4S can acquire and represent expert's knowledge about research system in various ways [2]:

- describing system's objects, relationships between objects, properties (e.g. colour, weight) and alternatives;
- describing relationships in more detailed way – representing flow name and type, functions name, behaviour, possible flow combinations, as well as flow properties (for example, type of matter: oil);
- specifying parameters and relationships between them for each behaviour state;
- creating and representing system's object, function and property hierarchies;
- transforming the knowledge acquired from expert in structure models.

System I4S allows create connections only between objects that are represented in a hierarchy, therefore all system parts must be described. Investigating systems in conditions of incomplete information, their models must meet following criteria [5]:

- It must be possible to create model, using only available knowledge;
- Model must describe all system, regardless of element heterogeneity;
- Model must be easy adjustable, when the system is changed;
- Model must "work" in conditions of incomplete information and must give new knowledge about the research system.

Structural modelling approach implemented in I4S allows structure model construction, considering before mentioned criteria.

On the basis of acquired and described knowledge about a system the functionality of software ensures a convenient way for transition between different type of models, which, in turn, is essential for carrying out appropriate research models and deep causal analysis. The example of knowledge representation view of the system under investigation is given in Fig. 2.2.2.2.

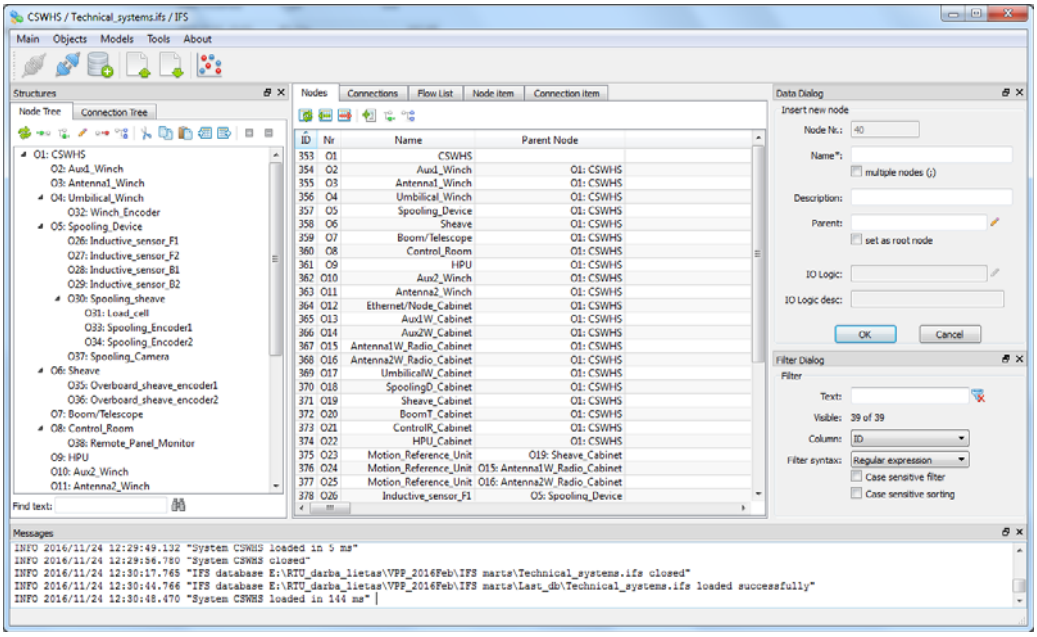


Fig. 2.2.2.2. Example of knowledge acquisition and representation in I4S

System I4S allows manipulate with represented knowledge about objects, relationships, properties and behaviour. Expert can add not only object description, but also to change and delete it, and also to reorganize system objects. Reorganization here means that a chosen system object (also with all descendants) can be relocated in other place within the hierarchy, changing objects predecessor. It is relevant not only to change system representation in the case when system changes, but also if an expert has created inaccurate system representation. System I4S performs representation of various hierarchies: system's parts, functions and properties. System creates object and function hierarchies also in corresponding models. When in the system I4S the description of research system is created and are represented all systems parts, relationships, behaviour and parameters as well as logical operators, then the construction of different structure models in different decomposition levels is performed [2].

There are about 20 knowledge structures from different domains (for example Evaluation of FET projects, Viable systems model, Control System for Winch Handling System, Language interconnection, Roads connection of Latvia, Bayesian network example etc.) entered in a repository. At present, a repository of models of systems is built taking into account their affiliation to technical or non-technical systems (see Fig. 2.2.2.1). Such a division of systems under investigation is not enough, namely, in the future another more detailed classification of systems must be developed and more knowledge structures must be added. As well as a qualitative and quantitative analysis of models stored in a repository must be made and appropriate functionality for I4S software must be added.

2.2.2.2. Emotion modelling for simulation of affective student–tutor interaction

Besides the creation of a repository, a research related to the development of knowledge structure for the dynamic adaptation of emotion-based tutoring process started on the previous stage of the project has been continued [6]. Various knowledge structures, e.g. knowledge structure for the representation of pedagogical knowledge and for the communication of agent's emotional state are developed during this research stage. Both mentioned knowledge structures are also included as part of the above mentioned repository (see Fig. 2.2.2.3).

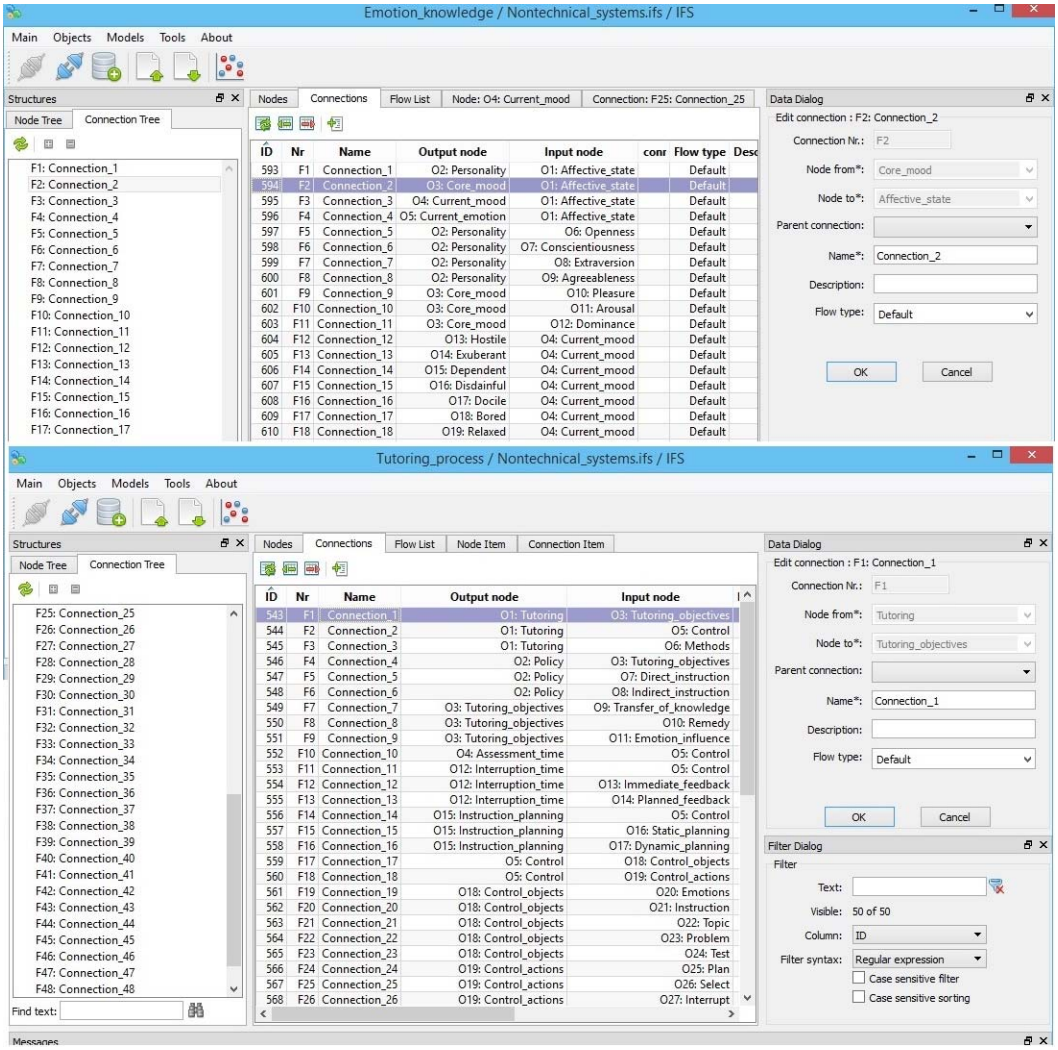


Fig. 2.2.2.3. Structure models representing an agent's emotional state and pedagogical knowledge

Furthermore, studies related to the implementation of dynamic adaptation of tutoring process are carried out by adopting a multi-agent system as a basic approach to enable a simulation of affective student-tutor interaction. The architecture of a pedagogical agent is designed supporting not only the usage of pedagogical knowledge (e.g., tutoring strategies including game-based learning) but also emotion ontology, which includes both descriptions of emotions and their possible causes. The communication of multi-agent system is enabled by using fragments of previously researched ontologies. The knowledge of personality is included in the model by transforming it

into core mood thus ensuring communication of emotional state among agents. Furthermore, the reasoning mechanism of a pedagogical agent utilizes knowledge about student's personality that serves for various purposes, e.g., for the prediction of student's emotions and behaviour, for the generation of appropriate tutor's personality and teaching actions, as well as for the selection of suitable teaching methods. Described results are represented in a journal paper "Petrovica S., Pudane M. Emotion Modeling for Simulation of Affective Student-Tutor Interaction: Personality Matching. International Journal of Education and Information Technologies, Vol. 10, 2016, pp. 159-167". Future work is related to the further development of student and tutor emotion models to include several functions, such as emotion generation and simulation of an emotional behaviour.

2.2.2.3. Development of ontologies- and rules-based multi-agent system management tool

During the previous stages of the project, research on knowledge structures used in intelligent agents was started. The mechanism for introduction of knowledge structure changes was developed. This included the general conceptual mechanism for introduction of changes, the ontology for experimental needs and the mechanism for adaptation of existing rules to new/incomplete data. For achievement of research goals and providing experiments the prototype of room cleaning multi-robot system simulator which supports manual introduction of changes of knowledge structure into ontologies and rule bases of agents that simulate robots was developed during the first stage. Fig. 2.2.2.4 shows the way how the agents are modelling cleaning robots.

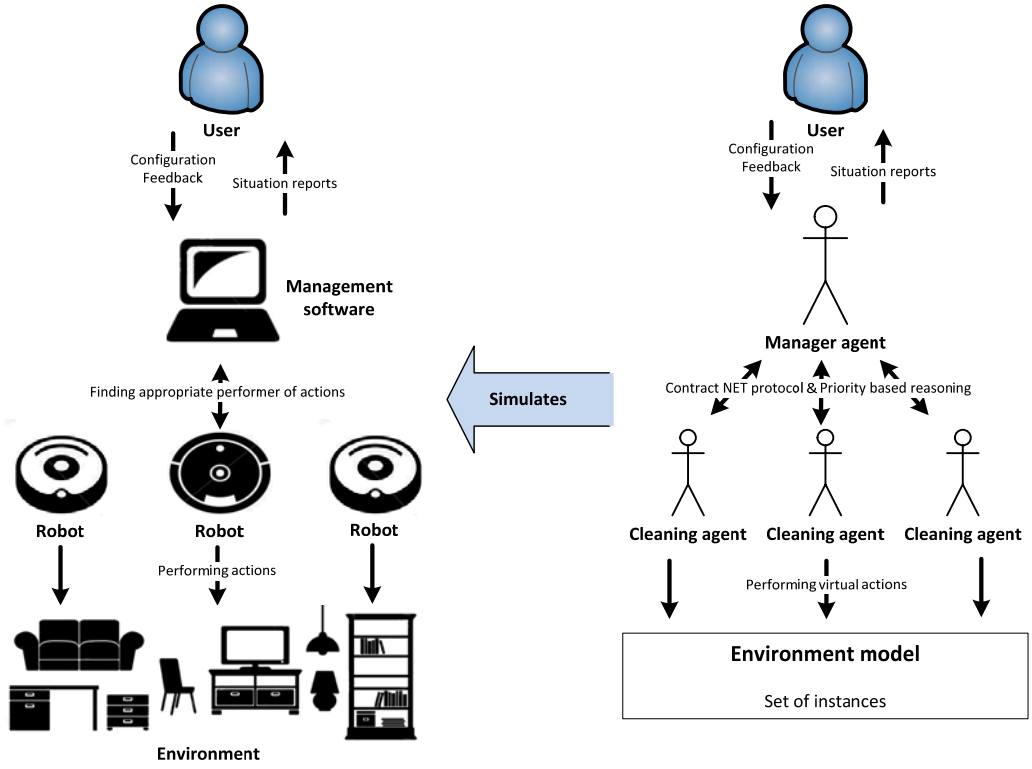


Fig. 2.2.2.4. The use of the multi-agent system to simulate a cleaning multi-robot system

Detailed architecture of the knowledge representation and learning framework were developed during the stage 2 of the project. The rule based learning approach was added to the previously created ontology based knowledge structure. A concept of a multi-agent system management tool was introduced in the system architecture. As a result, the system is composed of the following main components: a cloud that implements the knowledge base and machine learning algorithms, manager agent with its interfaces and the set of agents (see Fig. 2.2.2.5). The knowledge base and machine learning cloud are centralised for all agents to enable synchronous update of knowledge that is used by all agents. The cloud based knowledge base consists of the ontology, rule base and environment model. The latter two are based on the class hierarchies defined in the ontology, i.e., they are using classes from these hierarchies to define their instances and additional knowledge about them in the form of rules.

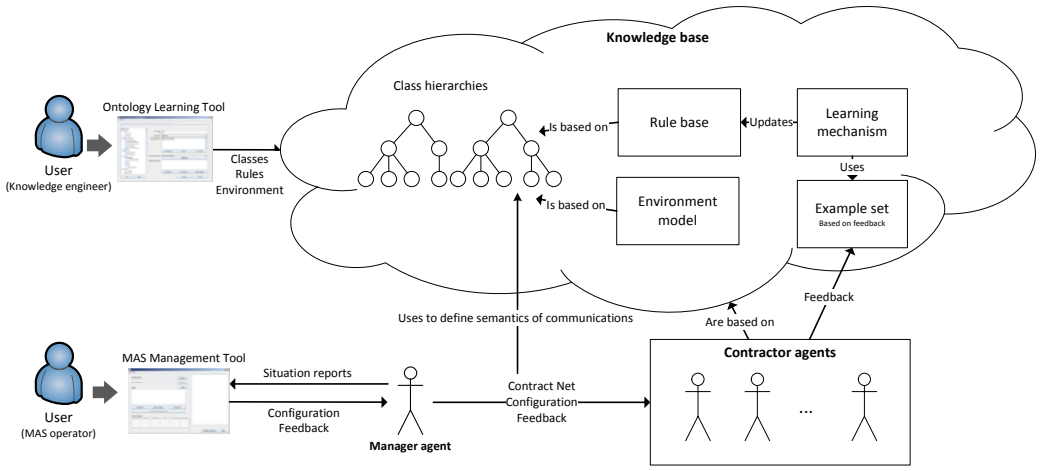


Fig. 2.2.2.5. The architecture of the cloud based multi-agent system

The relationships among the mentioned higher level components are the following. The agents use the knowledge base for their priority based decision making. They also use the ontology to define the semantics of the communications according to the JADE Ontology support. Finally, agents collect the example set for the learning mechanism. The manager agent updates the knowledge base and ensures the fulfilment of user's requests by contracting the appropriate agent to do the particular task.

The multi-agent system (MAS) consists of two types of agents. The manager agent represents the user and acts on behalf of him/her to find the most appropriate performer of each action. A Contract Net protocol is used to allocate tasks to the most appropriate agent. Another type of agents is the ones doing actual actions in the environment. These agents are named contractor agents. The set of these agents is heterogeneous in the sense that each agent can have different capabilities. Each agent knows its capabilities and can use the common knowledge base to calculate its appropriateness to the particular action and to choose the particular method to execute it. In the particular cleaning scenario these are the agents that represent the cleaning robots.

The manager agent's purpose is to serve as an interface between the system and its users and to organise the work of the multi-agent system. It is an agent with two user interfaces, namely the user interface for knowledge engineer and the user interface for the operator of the system.

As can be seen in the architecture, the developed management tool is complementary to the previously developed ontology learning tool. The ontology learning tool is used by the systems developer during the design phase of the system and later on to update the knowledge structures whenever necessary. The ontology learning tool has the following functionality:

- Defining the class hierarchies to be included in the ontology;
- Defining the environment model consisting of the class instances;
- Specifying the user defined rules.

The user interface of the tool is given in Fig. 2.2.2.6.

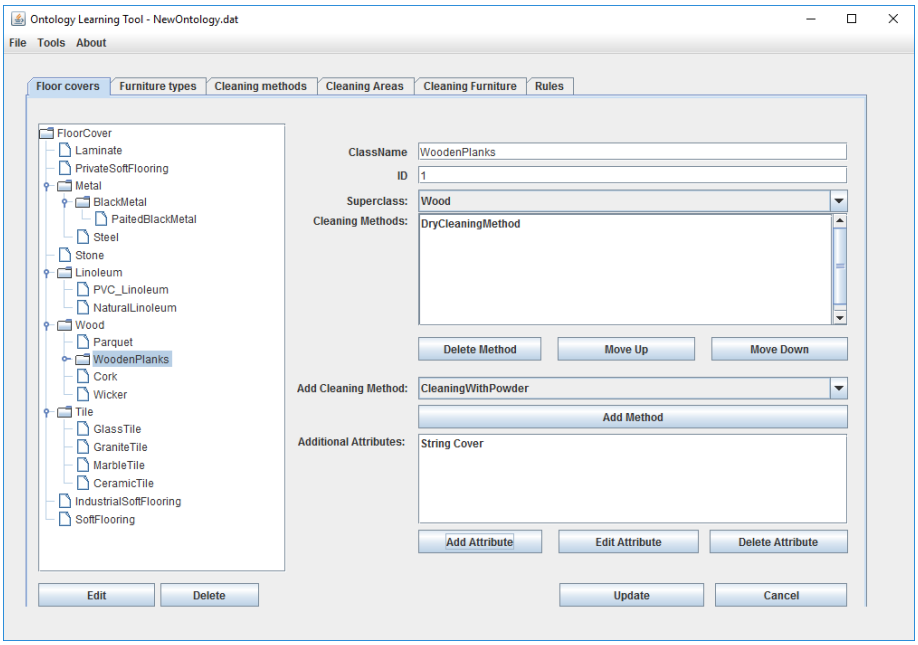


Fig. 2.2.2.6. Ontology learning tool

During the third stage of the project the conceptual framework has been implemented into ontologies and rules based multi-agent system management tool. The rule based priority mechanism is implemented into the tool and is applied to the task allocation in multi-robot systems (currently working in a simulated environment). Description of the developed tool has been published in a journal paper "Lavendelis E. A Cloud Based Knowledge Structure Update and Machine Learning Framework for Heterogeneous Multi-Agent Systems. International Journal of Artificial Intelligence, Vol. 14 (2), October 2016. CESER Publishing, pp. 157-170 (indexed in Scopus, SNIP: 1.159)".

Currently the developed management tool has the following functionality:

- Choice of ontology and knowledge base as well as loading the environment. These things can be read from files that have been created by the OntologyLearning tool. The ontology is stored as java classes while the knowledge base and the environment is stored in a knowledge base file. The area of the user interface denoted by 1 in Fig. 2.2.2.7 is used for this purpose.
- Configuration of the system by defining the agents and their capabilities in terms of cleaning methods. The area of the user interface denoted with 2 in Fig. 2.2.2.7 is used for the configuration purposes.

- After defining the configuration of the system, the agents are started and the area denoted by 3 in Fig. 2.2.2.7 becomes active and shows the state of each part of the environment.
- Additionally the area denoted by 4 in Fig. 2.2.2.7 becomes active and provides the log information and enables the user to give the feedback about the cleaning results.

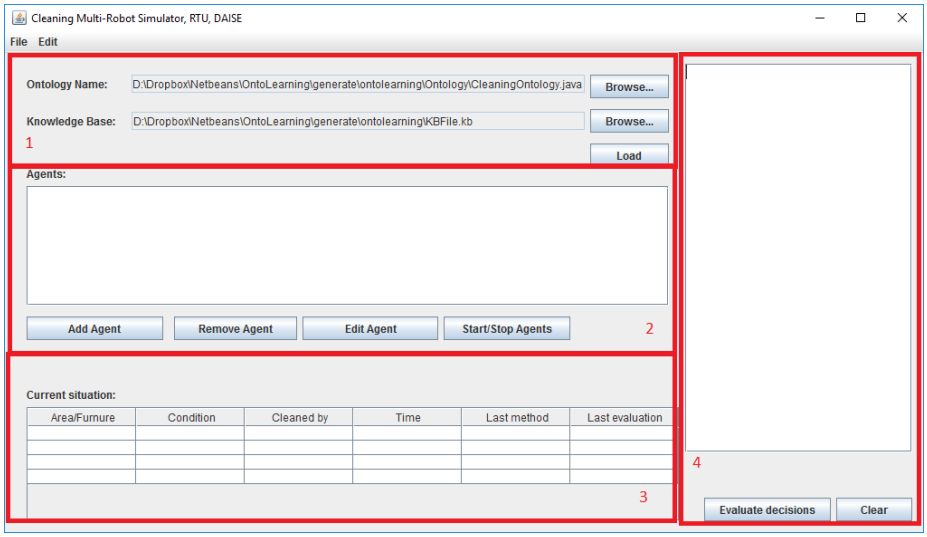


Fig. 2.2.2.7. User interface of the ontology and rules based multi-agent management tool

A scenario of a typical experiment in the simulated environment is the following:

- The user loads the ontology, knowledge base and environment details by specifying the path to the ontology and knowledge base file and clicks Load.
- The user chooses the configuration of the system by specifying the agents and the cleaning methods that they are capable to execute. After the configuration is complete the user clicks "*Start/Stop Agents*". The simulation starts at that moment. A screenshot of the tool just after the start of the simulation is given in Fig. 2.2.2.8.
- The manager agent autonomously checks the environment and in case some area becomes dirty, it finds the most appropriate agent to clean it. A screenshot with the result of the first cleaning task is given in Fig. 2.2.2.9, while a screenshot with more cleaning results is given in Fig. 2.2.2.10. The simulation is speeded up to shorten the time needed for simulations.

The Fig. 2.2.2.10 shows a situation when rule base is not perfect and the chosen capabilities are far from the optimal ones, for example, high pressure washing is bad for the Living room that has Laminate as a floor cover. Of course this particular situation can be solved by adding one particular rule. Still it does not solve the general problem. To get a general solution, future research during the next stage of the project is to implement the machine learning mechanism to enable the system to learn from the experience. Implementation of such a mechanism will make the tool fully functional and enable fully autonomous updates in multi-agent systems. The approach and tool after finalizing the implementation will give a significant contribution to the research of autonomous machines (for example, robots), by enabling autonomous changes in knowledge structures and knowledge itself after the system is deployed.

Lack of such mechanisms is one of the obstacles that currently hinders the development of fully autonomous robotic systems.

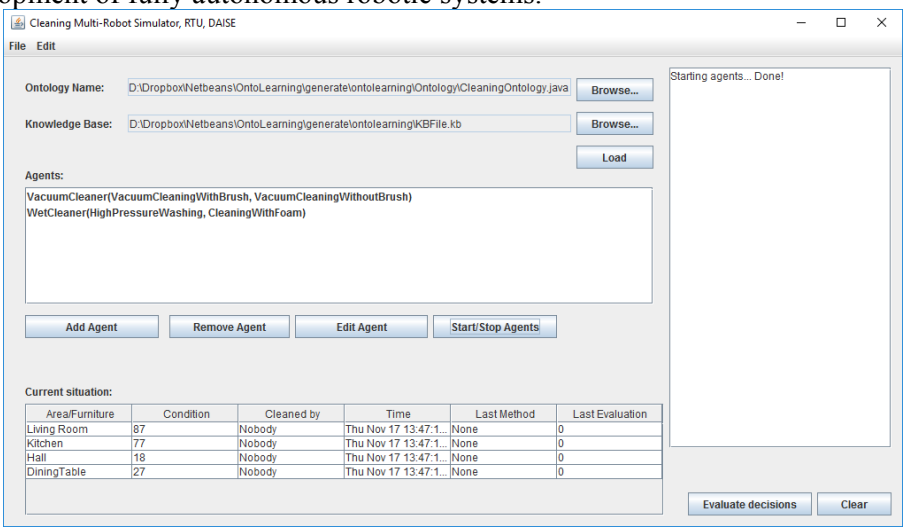


Fig. 2.2.2.8. Multi-agent system management tool just after the start of the simulation

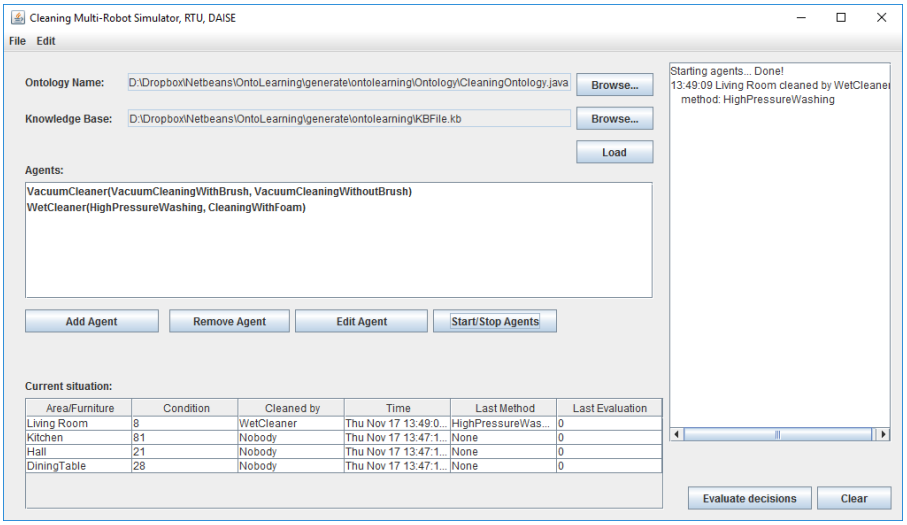


Fig. 2.2.2.9. The first cleaning results

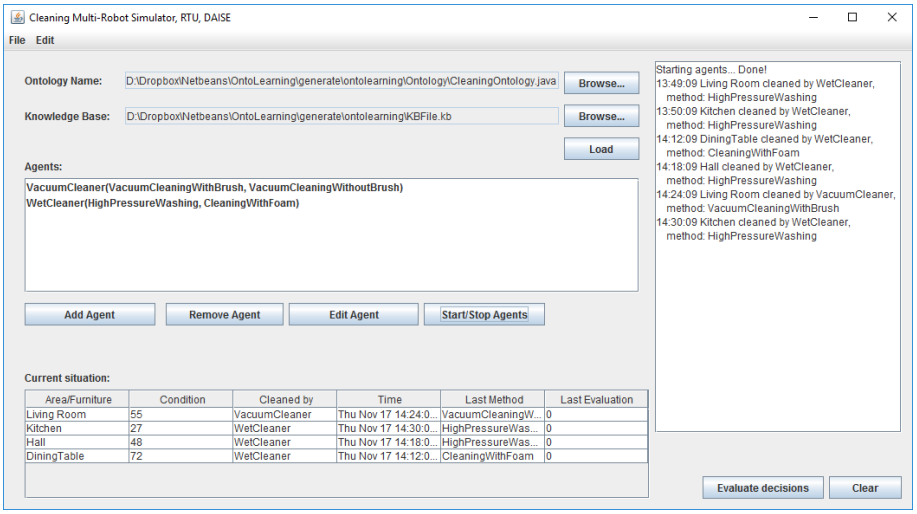


Fig. 2.2.2.10. Further cleaning results

References

1. Grundspenkis J. Reasoning Supported by Structural Modelling. Intelligent Design, Intelligent Manufacturing and Intelligent Management. Lecturer notes of the Nordic-Baltic Summer School on Applications of AI to Production Engineering, K.Wang and H.Pranevicius (Eds.). Kaunas University of Technology Press, Technologija, 1999, pp. 57-100.
2. Zeltmate I. Development of intelligent system for structural modelling of complex systems. Doctoral Thesis, Riga Technical University, 2012.
3. IFS User Manual v.1.0.
4. Young B.J., Booch G., Conallen J., Engel M.W., Houston K.A., Maksimchuk R.A. Object-Oriented Analysis and Design with Applications, 3rd Edition. Addison-Wesley, 2007.
5. Grundspenkis J. Structural Modelling of Complex Technical Systems in Conditions of Incomplete Information: A Review. In: Modern Aspects of Management Science, No 1. Riga, Latvia, 1997, pp. 111-135.
6. Petrovica S., Pudane M. Simulation of Affective Student–Tutor Interaction for Affective Tutoring Systems: Design of Knowledge Structure. International Journal of Education and Learning Systems, 2016, No.1, pp. 99-108.

2.2.3. Detailed development of the approach for aligning requirements/systems engineering based knowledge flows (structures)/information artifacts

The proposed approach for aligning requirements engineering based knowledge flows (structures)/information artifacts is based on the continuous requirements/systems engineering framework FREEDOM that was developed in the previous phase of the project. In this phase, the details of the framework, which influence the knowledge/information flow in the framework, were addressed. The results are described in the following publications:

- Businska L., Kirikova M. The Goal-Based Selection of the Business Process Modeling Language. 9th IFIP WG 8.1 Working Conference, PoEM 2016, Skövde, Sweden, November 8-10, 2016, Vol. 267 of the series Lecture Notes in Business Information Processing. Springer, pp. 307-316.
- Kirikova M., Matulevičius R., Sandkuhl K. The Enterprise Model Frame for Supporting Security Requirement Elicitation from Business Processes. Proceedings of 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016, Communications in Computer and Information Science, Vol. 615. Springer, pp. 229-241.
- Kirikova M., Matulevičius R., Sandkuhl K. Application of the Enterprise Model Frame for Security Requirements and Control Identification. Databases and Information Systems IX, G. Arnicans et al. (Eds.), 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016. IOS Press, pp. 129-142.
- Kozlovs D., Kirikova M. Auditing Security of Information Flows. Proceedings Perspectives in Business Informatics Research. 15th International Conference, BIR 2016, Prague, Czech Republic, September 15–16, 2016, Vol. 261 of the series Lecture Notes in Business Information Processing. Springer, pp.204-219.

In parallel the FREEDOM framework was compared to other frameworks developed for similar purposes; and the variants of the FREEDOM framework were identified and illustrated to demonstrate its flexibility. More details can be found in "Kirikova M. Towards Framing the Continuous Information Systems Engineering. Joint Proceedings of the BIR 2016 Workshops and Doctoral Consortium, co-located with 15th International Conference on Perspectives in Business Informatics Research (BIR 2016), September 14-16, 2016, Prague, Czech Republic, B. Johansson and F. Vencovský (Eds.), Managed Complexity. CEUR-WS.org, Vol. 1684".

Aligning requirements/systems engineering based knowledge flows (structures)/ information artifacts requires addressing the existing gap between business process models and states of business objects. Therefore an approach was developed for explicit definition of states of business objects, automatic generation of conceivable state space at a process model design-time, automatic generation of lawful state space, and compliance checking at a process run-time, which is described in "Peņicina L. Controlling Business Object States in Business Process Models to Support Compliance. PoEM 2016, Doctoral Consortium, Skövde, Sweden, November 8-10, 2016 (will be published in Ceur-ws.org)".

In this regard time aspects play an important role; therefore, in this phase of the project, also time aspects were analyzed with respect to the FREEDOM framework. Below the results of the analysis are briefly described.

2.2.3.1. Time dimension with respect to the FREEDOM framework

FREEDOM framework [1] describes two states "R" and "F". "R" – reality, artifacts that represent present condition or state, also called as-is situation. "F" – future representation, in this representation there are artifacts that describes system's future: e.g., models, states, designs, predictive analysis results. Similarity in R and F representations is noticed, as many enterprise architecture frameworks describes as-is state as present and to-be state as future representation of an enterprise and project enterprise development as transition from present as-is state to to-be future state [2]. The FREEDOM framework exposes a similar approach.

During *E_1 – Requirements Engineering* function system requirements are formulated in cooperation with stakeholders. At the end of E_1 stage – the list of requirements is obtained (includes also time restrictions on system implementation or system life cycle [3]). *E_2 – Fulfillment Engineering* defines detailed project plan. We have to consider detailed and deep planning of jobs and resources during E_2, also new technical details that comes up can change requirements. This means that the list of requirements can change very frequently and the date of transition from E_1 to E_2 is difficult to define. After the list of requirements and project plan is finalized, the agreement between developer and customer is to be signed. Formally, agreement's date is the end of E_2 at a particular iteration and the start of D function. *D – Design and implementation* function is usually managed based on project plan. Project plan includes time and other resource restrictions for all stakeholders. The end of D stage can be strictly defined, as acceptance act is signed. The date of the act formally is the end of D stage and start of O function/stage. *O – Operations* function may have specific time restrictions, but those depend on the particular enterprise and systems. If system life cycle is defined, the end of O stage ere is known. Also there is difficulty to define *M – Management* stage time frames. FREEDOM permits to shows O and M stages sequentially as well as to consider them in parallel.

The example of different time aspects is shown in Fig. 2.2.3.1.

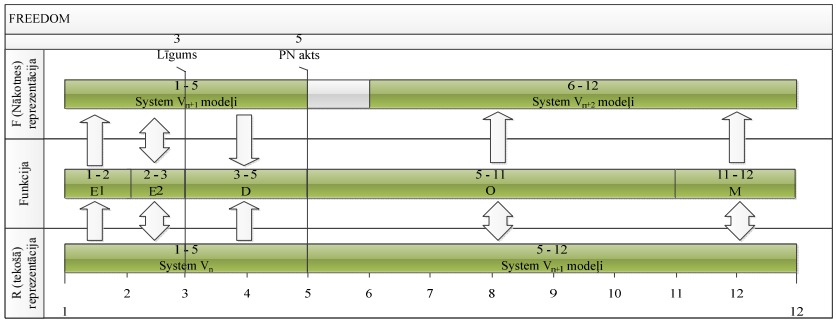


Fig. 2.2.3.1. Time issues in FREEDOM framework (an example)

Taking in to consideration above mentioned time aspects that above were discussed for the base representation of FREEDOM framework, it is clear that the complexity and variability of time aspects for variations of the FREEDOM framework will require a separate time model for each variant of the framework.

The above discussed developments mainly address internal artefacts of the FREEDOM framework. However the framework prescribes also consideration of the external artefacts. Therefore additionally an approach on the identification of system's external knowledge structures/information artefacts and the changes of their state has been proposed. The approach is rooted in the analysis and the representation of the structure and content of the documents capturing valuable knowledge for the enterprises. Below, the brief description of the approach is given.

2.2.3.2. An approach on the identification of system's external knowledge structures/information artefacts and the changes of their state.

Growing amount of information and its diversity pose challenges to process it. Both in the World Wide Web and in specific data bases inside the organizations there exist information sources containing valuable and reusable information. Nowadays there are trends to use this existing information in automated information processing solutions, for example, in gathering of Web content, recommender systems, data mining and other solutions [4]. Information processing usually consists of activities, which can be classified in four main groups, namely, information acquisition, analysis, decision making and decision implementation [5]. The development of automated information processing solutions fosters the execution of these activities.

There are several groups of information artefacts that are produced within the enterprise or outside the enterprise and are relevant for the enterprise. For example, course descriptions in the educational institutions, job descriptions in industrial organizations.

The research in the project was done in the context of education demand and offer information monitoring, which is well known by the researchers, therefore relevant information sources and knowledge structures were identified in this context.

Education demand and offer information monitoring is perceived as service system [6]. Service systems are focused on the interaction between people, technology, and other internal and external service systems, as well as on the exchange of shared information between the stakeholders of service system to achieve common goal. In this regard, for identification of relevant external information, any enterprise actually can be considered as a service system.

An important issue of analysis of external information is identification of information sources. Information source is regarded here as anything what gives information about anything. It is assumed that information source is constituted by document or the set of documents. Document is the representation of something. For example, curriculum gives information about courses included, thus we say that curriculum is constituted by the set of courses; whereas the course description is the document representing, for example, obtainable knowledge. Similarly we can say about job advertisements of some company, they are the edu d/o information source constituted by the set of job advertisements (i.e., documents).

In the context of monitoring of education demand and offer the following sources were identified: in the job market – job advertisements, job descriptions, surveys; in the education institutions – study and certification course description; in governmental institutions, international institutions and from the industry associations – profession standards, curriculum recommendations, professional bodies of knowledge. For business companies there will other relevant sources, such as industrial standards,

legal information, as well as information in social networks. Regardless of the sources the way how the sources are handled is similar; therefore, further the illustrations will be given in the context of education demand and offer monitoring.

To be able to process the external information it is necessary to identify knowledge structures that give an opportunity to establish the conceptual base (as simple as the list of concepts or more complex knowledge structure). For instance, the following knowledge structures for systemizing knowledge, skills, and competences were identified in the educational demand and offer monitoring context:

- Taxonomy provided by Association for Computing Machinery (ACM) [7];
- European Dictionary of Skills and Competencies (DISCO) [8], [9];
- European e-Competence framework (e-CF) [10];
- Skills Framework for Information Age (SFIA) [11];
- Classification of European Skills, Competences, Qualifications and Occupations (ESCO) [12].

From the listed knowledge structures, ACM, e-CF and SFIA are specific for the field of computer science and information technology, whereas DISCO besides this field includes also the skills of other fields (e.g., business administration). General purpose knowledge structure with respect to knowledge and competences is ESCO.

Knowledge structures can serve as a "unified language" [13], which is understood by stakeholders. It is important in external information monitoring to ensure unified and structured way for the representation and processing of the information sources.

In information sources not only their amount of information can be huge, but it can also continuously change (e.g., changing course descriptions, changing standards etc.), it can be distributed across different locations (e.g., World Wide Web, databases, printed documents) and can be in different formats (various unstructured, semi-structured or structured textual documents) [14]. Also the presence of service systems and the necessity to gain mutual benefit raises the challenge to process the information that is mutually important for all involved parties.

The proposed approach is based on the following results obtained in information demand and offer information monitoring context:

- Identified general approach on change management of the information artefacts (documents);
- Identified different types of changes in information artefacts (documents).

Further the proposed approach on the state identification of system's external knowledge structures/information artefacts is described in detail. It is rooted in the analysis and structured representation of information artefact (i.e., document) structure and its content.

The main steps of the approach are the following:

1. Knowledge structure based document transformation ;
2. Identification of changes;
3. Trigger based notification.

By performing these steps, it is possible to capture the information artefact state, how it has changed, how it has changed in comparison with mapped documents and ensure the facilities of the notification in order to perform further actions, e.g., start

discussion with participants within service system, reflect changes in the operation of organization. Table 2.2.3.1 briefly describes the steps of the approach.

Table 2.2.3.1. Brief description of the steps of the approach

Knowledge structure based document transformation	Identification of changes	Trigger based notification
Knowledge structure based document transformation is used to ease the mapping of different documents and to foster the understanding among involved parties.	Changes can be of different types, namely: • Structure changes • Extracted concept changes • Correspondence (mapping) changes between different information artefacts: o Mapping removed o New unmapped elements o New mapped elements	Trigger based notification about the 3 different types of changes

It is important to ensure continuous information artefact change management process to be able to perform informed activities within the service system. General change management process is presented in Fig. 2.2.3.2.

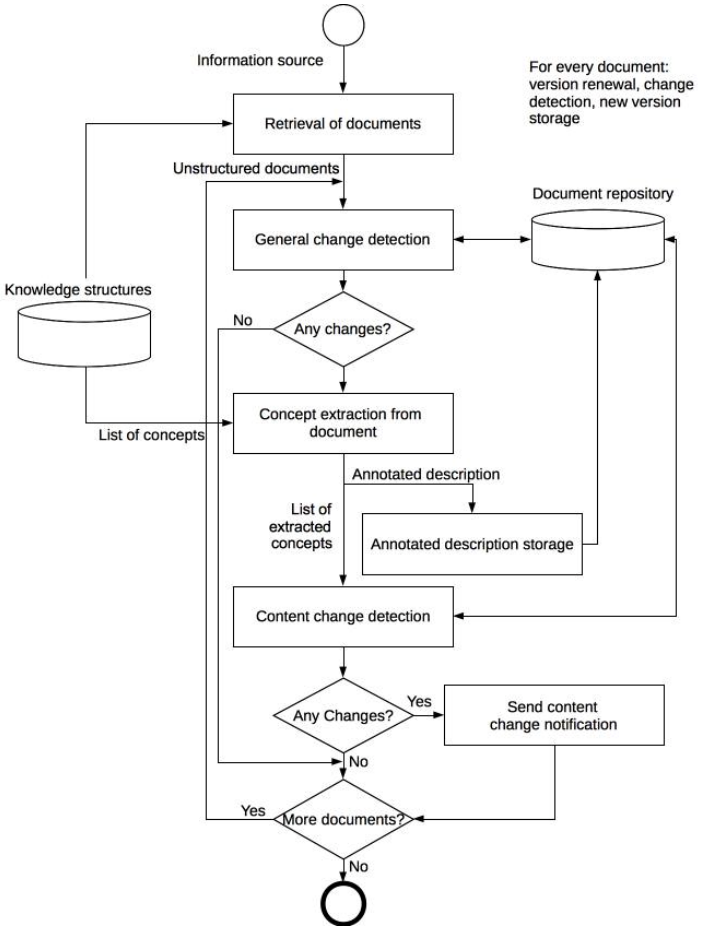


Fig. 2.2.3.2. Information artifact change management process

Two types of change detection can be distinguished, namely, general change detection and content change detection. *General change detection* renews the version of source document in the repository or in the database and also identifies if previous document version differs from the current one. In case, the changes are not identified, next steps should not be performed, because it is known that the document has not been changed. *Content change detection* includes the detection of changes in the concepts identified in the document (e.g., new concepts identified, concept deleted etc.).

To keep a reusable history of considered artefacts, the following general information artefact change management process is proposed:

1. Information source is selected.
2. Based on the available knowledge structures, relevant documents are retrieved.
3. Retrieved documents are checked against the document version already available in the system. If the changes are detected, then the new unstructured document version is saved. After the saving next action can be performed. If no changes are detected, then the next document is taken for processing and change identification.
4. By using knowledge structures, the relevant concepts are identified in the document. As a result, an annotated document is created. Annotations are the concepts identified in the document.
5. Detected concept changes in the document are saved in database. Several changes of documents are aggregated and special notification triggers may send notifications to involved parties to take further actions with respect to the identified changes.

Taking into consideration that large amount of requirements are elicited and analyzed during information systems development process and that those requirements can be processed, stored, and managed in various tools and communicated via various channels; the appropriate requirement distribution approach was developed. This approach is the basis for seeing the constraints on the flexibility of FREEDOM framework. Below the approach is briefly described.

2.2.3.3. Requirements distribution approach

Large amount of requirements are elicited and analyzed during Information Systems development process. And those requirements can be processed, stored and managed in various tools, for example MS Word or MS Excel, e-mails, JIRA system etc. and communicated in various channels. The more tools and channels we have the more effort we need for requirements management to keep all requirement versions up to date. In a situation when we have many tools, channels and involved persons in IS development and even in all IS life cycle, some issues of requirements versions and availability may arise.

Requirement distribution approach describes key principles for effective requirements distribution during the IS life cycle.

Requirements are information about desired IS or expected results. Depending on selected approaches (Requirements Engineering approach, Project Management, Development etc.), various ways for requirements documentation and analysis can be used, for example – user stories, notes, specifications, models, tasks in development systems etc. If we take a situation when many sides are involved in IS development like client's project manager and analysts, development team (project manager,

analysts, testers, programmers etc.) and other stakeholders and stakeholder groups, then a question arises – how to manage requirements in a way, that allows all involved persons to have access to up to date requirements.

The following requirements are considered as important for requirement distribution:

- Requirements distribution approach must be effective – this approach should provide requirements to right person in the right time and the right place;
- Approach must be sustainable – with possibility to provide access to requirements in future.

As we know from BABOK 3 [15] – Requirement Engineering includes activities like Requirements elicitation and collaboration, Strategy Analysis, Analysis planning and monitoring, Requirement Analysis and Design, Solution Evaluation and Requirements Life Cycle Management. All these activities can be involved in each IS life cycle phases. And during the IS life cycle phases information and knowledge about IS is critical for effective work results.

The following factors have been taken into consideration to identify challenges in requirements distribution in the context of continuous requirements/system engineering:

- The IS life cycle contains several projects (more than one), like the development of new IS module;
- The Following groups are involved in IS development:
    - Clients project team (project manager, analysts, testers);
    - Development project team (project manager, analysts, testers, developers and other specialists);
    - Client (like a person or group of persons);
    - Stakeholders;
    - Third party – quality control, auditors etc.
- More than one tool is used (e.g., MS Word/Excel, e-mail, JIRA, notes and other tools) for capturing requirements and more than one channel is used to communicate requirements.

Aforementioned facts are very important and in the same time, if we use more than two tools (MS Word, JIRA, e-mail, Share point and other tools), it can be difficult to make it work. The possible solutions in this situation can be:

- Analyst or other specialist repeats each modification in requirements in each tool – but this is time consuming;
- We can develop integration platforms that make online changes automatically, if we make changes in one of used tools – this approach probably will be expensive.

As in above cases, most probably, the effectiveness and efficiency of requirements distribution will be low; we need to find other alternatives.

From this point we can point out difficulties or challenges in requirement distribution:

- Optional amount of time resources for requirement management and distribution – more tools, more operations to keep up to date requirements in all tools;
- Best alternative for requirement management tool – more involved groups and persons, more rules and principles for requirement management tools

selection. It is important to take into account specifics or each group and possibilities. There will be groups of limited financial resources that will not acquire expensive tools;

- Other challenges.

Many approaches and possibilities in knowledge management and distribution are described in literature. For example requirement distribution can be equivalent to knowledge management, because it includes a whole set of methods for knowledge distribution [16]. There are some assumptions about recommendation tools and their effort in requirement engineering and distribution [17].

The proposed approach describes the key principle that can help make requirements distribution more effective and even make requirements management simpler and more effective and save resources. Approaches testing will be held during the next phase of the project.

The basic idea of the approach is – by reducing requirements management tools and channels we can reduce resources for requirement management and simplify communication management.

This simple idea can be shown as Architecture of Requirement distribution approach (see Fig. 2.2.3.3).

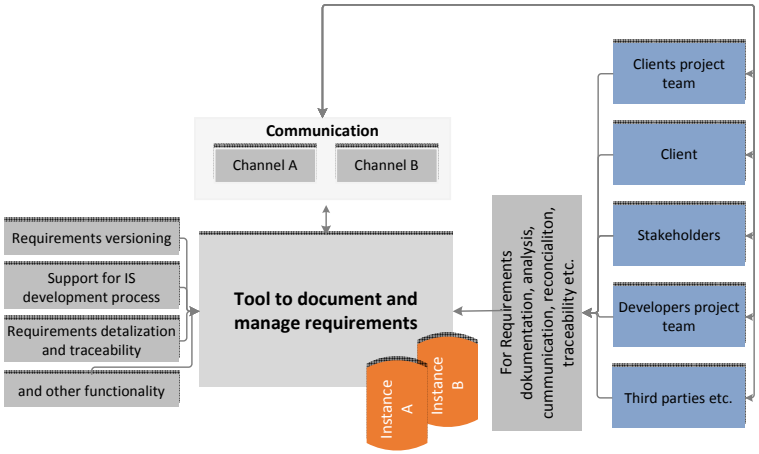


Fig. 2.2.3.3. Architecture of requirements distributions approach

Requirements distribution approach recommends:

- To agree on requirement management tool at the start of the project (better in the first project of IS development);
- In tools selection take into account the following criteria:
  - Possibilities to share information and requirements and to work with requirements in online mode;
  - Possibilities to document requirements in textual format, models etc. and to manage requirements traceability;
  - Possibilities to link related information like testing scenarios, testing tasks, testing results etc.;
  - Possibilities to make reports and analyze information in various dimensions;
  - Possibilities to use results of one project in another project;
  - The tool can be used as a communication channel.

- To keep to following conditions:
  - The tool needs to be used to document requirements and these requirements are kept up to date;
  - All involved groups use the tool;
  - Historical information is available, etc.

Mostly IS development projects involve more than one project team and involved person. If it is so, there are challenges in requirement management and distribution because in most cases each project team uses their own tool and communication channel. Proposed approach points out the need for small amount of requirement management tools and sets rules for effective requirements management and communication.

To establish the background for unification of abovementioned aspects of artifact alignment in the next phase of the project, the possibility to use graph algorithms in information flow analysis was analyzed.

2.2.3.4. Use of graph algorithms for analysis of information artefact flow

Today through the information systems (IS), which operate in a specific enterprise can flow very large amount of information. Also, the rapid development of IT in recent years has created some specific problems, such as data storage and processing. Therefore, it is necessary the methods that can handle the flow of information [18], [19].

The information flow concept appeared because amount and speed of transmitted information every time rises up. Information flow has to comprise the minimum amount of information that is necessary for company's functionality.

Information flow description using graph is one of the most popular methods that exist today. Using this approach we can graphically represent the management system and the information flow functionality [20].

To establish the graph suitable for information flow analysis the following approach is proposed:

1. Build the BPMN model (s) for the specific scope of enterprise activities;
2. Cross-check the built models for completeness using ArchiMate language;
3. Transfer the developed models into the information flow graph.

The information flow graph is constructed using the elements depicted in Fig. 2.2.3.4.

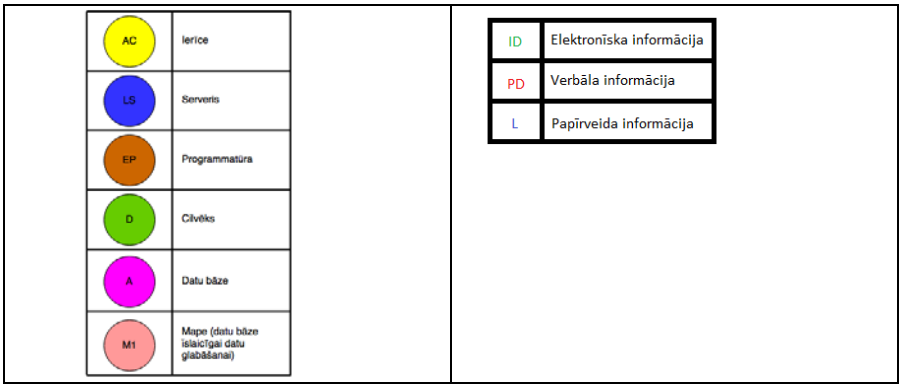


Fig. 2.2.3.4. AC – device, LS –server, EP –Software, D – human, A – data base, M1 – Temporal data storage; ID – electronic information, PD – verbal information, L – paper based information.

The example of the application of the approach is illustrated here with the automatic control signing case. In Fig. 2.2.3.5 the source BPMN and ArchiMate models are shown, the obtained information flow graph that is applicable for the use of graph algorithms is illustrated in Fig. 2.2.3.6. The abbreviations used in Fig. 2.2.3.6 are described in Table 2.2.3.2. The arrows in the graph reflected in Fig. 2.2.3.6 show the direction of the information flow.

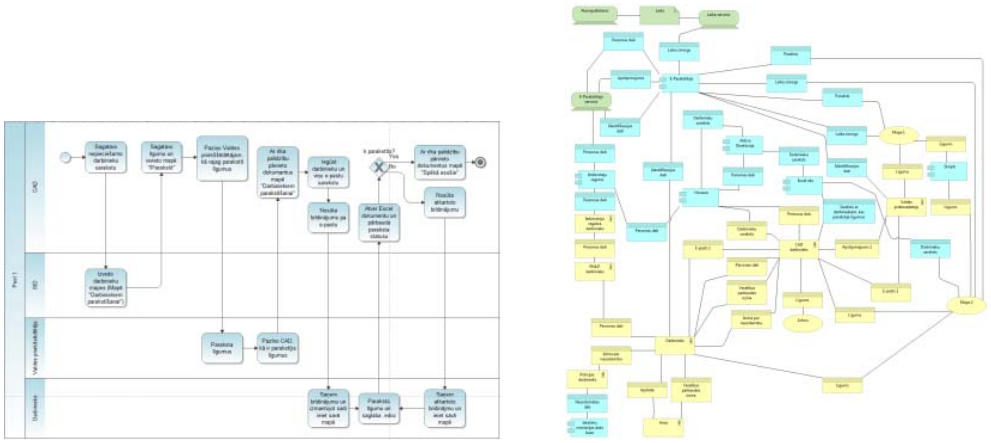


Fig. 2.2.3.5. BPMN and ArchiMate models of company employer automatic contract signing case

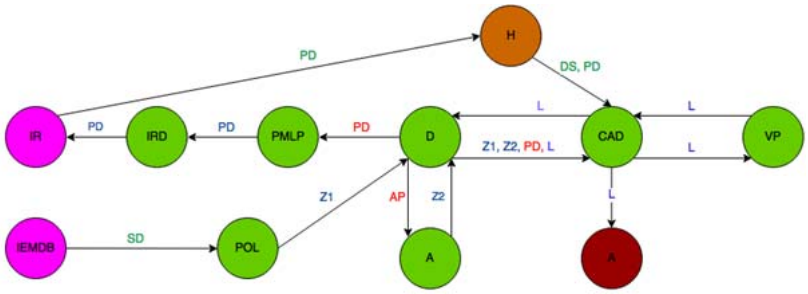


Fig. 2.2.3.6. Company employer automatic contract signing information flow graph

Table 2.2.3.2. Designation of the abbreviations used in the graph

Abbreviation	Transcript
D	Employer
PMLP	Migration service employer
IR	Citizens register
VP	Company head director
CAD	Work resources department employer
A	Archive
IEMDB	Interior ministry database
POL	Policeman
A	Doctor
H	Horizon application
IRD	Citizens register employer
PD	Person data

Z1, Z2	Certificate
AP	Examination
SD	Non conviction data
L	Contract
DS	Employer list

The colors of the graph reflected in the Fig. 2.2.3.6 show that the graph algorithms for information flow analysis have to distinguish between different types of information and different types of information handlers. Modification of existing algorithms for graph analysis (such as identification of cut-sets of the graphs, etc.) is intended to be performed in the next phase of the project.

References

1. Kirikova M. Continuous requirements engineering in the FREEDOM framework: A position paper. CEUR Workshop Proc., vol. 1564, 2016.
2. Kirikova M., Penicina L., Gaidukovs A. Ontology based linkage between enterprise architecture, processes, and time. Commun. Comput. Inf. Sci., vol. 539, pp. 382-391, 2015.
3. Finke A. Requirements inheritance in continuous requirements engineering: A position paper. CEUR Workshop Proc., vol. 1564, 2016.
4. Rubin S.H., Chen S.-C. The 2013 IEEE International conference on information reuse and integration: Forward. In 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI), 2013, pp. xii–xii.
5. Parasuraman R., Sheridan T.B., Wickens C.D. A model for types and levels of human interaction with automation. IEEE transactions on systems, man, and cybernetics. Part A, Systems and humans: a publication of the IEEE Systems, Man, and Cybernetics Society, vol. 30, no. 3, pp. 286-297, May 2000.
6. Spohrer J., Maglio P., Bailey J., Gruhl D. Steps Toward a Science of Service Systems. Computer, no. January, pp. 71-77, 2007.
7. Association for Computing Machinery (ACM) taxonomy, 2012. [Online]. Available: http://www.acm.org/about/class/2012.
8. 3s Unternehmensberatung. European Dictionary of Skills and Competencies - DISCO II, 2012. [Online]. Available: http://disco-tools.eu/.
9. Müller-Riedlhuber H. The European dictionary of skills and competences (DISCO): An example of usage scenarios for ontologie, in Proceedings of I-KNOW 2009 - 9th International Conference on Knowledge Management and Knowledge Technologies and Proceedings of I-SEMANTICS 2009 – 5th International Conference on Semantic Systems, 2009, pp. 467-479.
10. European Committee for Standardization, European e-Competence Framework 2.0, 2010. [Online]. Available: http://ecompetences.eu/.
11. SFIA FOUNDATION, Framework referece SFIA version 5, 2011. [Online]. Available: http://www.sfia.org.uk/v5/en/.
12. Vrang M., Papantoniou A., Pauwels E., Fannes P., Vandensteen D., De Smedt J. ESCO: Boosting Job Matching in Europe with Semantic. Computer, vol. 47, no. 10, pp. 57-64, 2014.
13. Asgrahani M., Shankararaman V. Skills frameworks: A tool for reform in Information Technology higher education. In 2014 9th International Conference on Computer Science & Education, 2014, pp. 81-88.
14. Rudzajs P. and Kirikova M. IT Knowledge Requirements Identification In Organizational Networks□: Cooperation Between Industrial Organizations And Universities. In Information Systems Development: Asian Expierences,

W. W. Song, S. Xu, C. Wan, Y. Zhong, W. Wojtkowski, G. Wojtkowski, and H. Linger, Eds. Springer New York, 2011, pp. 187-199.
15. International Institute of Business Analysis. A Guide to the Business Analysis Body of Knowledge v3, 2015.
16. Mahdi Owayid A., Alrawi K., Shaalan K. Chapter 45. Effectiveness of Information Systems Infrastructure and Team Learning in Integration Knowledge Management and e-Learing Technologies. The 8th International Conference on Knowledge Management in Organizations, 2014.
17. Maalej W., Kumar Thurimella A. Towards a Research Agenda for Recommendation Systems in Requirements Engineering. 2009 Second International Workshop on Managing Requirements Knowledge (MaRK'09), 2010.
18. Hammer C., Snelting G. Flow-sensitive, context-sensitive, and object-sensitive information flow control based on program dependence graphs. International Journal of Information Security, December 2009, Volume 8, Issue 6, pp. 399-422.
19. Durugbo C., Tiwari A., Alcock J.R. A review of information flow diagrammatic models for product–service systems. The International Journal of Advanced Manufacturing Technology, February 2011, Volume 52, Issue 9-12, pp. 1193-1208.
20. ISO/IEC 2382-1:1993, Information technology – Vocabulary – Part 1: Fundamental terms. 01.01.01: knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts that within a certain context have a particular meaning.

2.2.4. Development and integration of different services in Web portal using open Semantic Web resources and enhancement of software configuration management methods

The EAF methodology, which is based on MDA approach and is aimed at software configuration management and automation of IT operations for software practical application, has been further improved. A survey has been implemented to determine the main continuous processes that ICT companies are going to automate, the current automation level of these processes, and the main challenges in the automation field. The survey had 42 respondents from more than 35 ICT companies in Latvia. The results of survey will be used for further enhancing the EAF. More details can be found in scientific papers "Bartusevics A. Automation of Continuous Services: What Companies of Latvia says about it? In: Procedia Computer Science 2016, ICTE 2016, December (accepted)" and "Novickis L., Mitasiunas A., Ponomarenko V. Information Technology Transfer Model as a Bridge between Science and Business Sector. In: Procedia Computer Science 2016, ICTE 2016, December (accepted)".

The semantic services development and integration methodology utilizes model-driven approach to design, develop and maintain services with reusability in mind. One of the key elements of the approach – the Reusable Functions Library – was developed. This library makes it possible to reuse implemented automation functions in different projects and different workflows. Development of the library is described in "Bartusevics A., Novickis L., Lesovskis A. An Approach for Development of Reusable Function Library for Automation of Continuous Processes. In: Procedia Computer Science 2016, ICTE 2016, December (accepted)".

The methodology for systematic development of RESTful Semantic Web services using SADI (Semantic Automated Discovery and Integration) framework has been further developed. The methodology was used to develop a set of Semantic Web services that are used in eLOGMAR logistic portal (available at http://www.elogmar.eu). The validation was implemented in collaboration with industrial partner – company Logitrans Consult. For scientific results related to this direction of research, see the following publications: "Grocevs A., Prokofjeva N. The Capabilities of Automated Functional Testing of Programming Assignments. In: Procedia – Social and Behavioral Sciences, Volume 228, 2016, pp. 457-461", "Jurenoks A., Novickis L. Adaptive Method for Assessing the Life Expectancy of a Wireless Sensor Network in Smart Environments Applications. In: Proceedings of 14th IFAC International Conference on Programmable Devices and Embedded Systems, PDES 2016. Brno University of Technology: 2016, pp. 93-98", and "Jurenoks A., Novickis L. Simulation-Based Experimental Research of Wireless Sensor Network Life Expectancy Reconfiguration Method in Transport Logistics Area. In: 2nd International Conference on Systems Informatics, Modelling and Simulation, Riga: 2016, pp. 135-140".

2.2.4.1. Validation of algorithms for services integration in Web portal

Service development and integration methodology utilizes model-driven approach to design, develop and maintain services with reusability in mind. One of the key elements of the approach is the development of library of reusable functions. This library makes it possible to reuse implemented automation functions in the different

projects and in the different workflows. The approach for development of RFL (*Reusable Functions Library*) contains three mandatory conditions for developers:

- All automation functions should be developed and stored in the scripts by particular pre-defined template. The template contains special requirements for naming conventions and formatting to be able determine where is description of current function, input and output parameters etc.
- All functions should be parameterized and independent on each other. Revisions should be passed to function as an input parameter.
- Each function should have at least one output parameter – function exit code. The value could be 0 (in case of success) and 1 (in case of fail).

The functions that fulfill these conditions are called Actions. The main attributes of an Action are the input parameters, unique name and output parameters. Two or more Actions could create a workflow called ActionFlow. Each Action in an ActionFlow takes into account the output parameters of the previous Actions in this particular ActionFlow. The architecture of the approach consists of two main parts (Fig. 2.2.4.1).

The first part is a physical directory in the particular file system that stores the source codes of reusable automation functions. The second part is an RFL manager that parses and transforms functions to make them reusable, manageable and human readable.

Semantic web services which semantically annotate information received from traditional web services that are included in eLOGMAR portal (www.elogmar.eu) have been developed at the 2nd project's period. Processed information is related to the logistics domain (available routes and their types, cargo expenses etc.) [1].

The process of validation of services integration into Elogmar Web portal has been performed in close cooperation with industrial enterprises Logitrans Consult Ltd. and UIFA. Some results of Web portal validation are presented in Fig. 2.2.4.2 and Fig. 2.2.4.3. The results of users' feedback show that in general the quality of portal functionality as well as user interface has been estimated positively.

Semantic web technologies have been integrated also into model driven software configuration management approach [2]. It makes easier to implement repeated usage and integration of Software Configuration Management data.

The process of validation of has been performed in close cooperation with industrial enterprises, including Tieto Latvia.

Besides that, the approach has been tested in the set of experiments in five different software development projects. In all of the mentioned projects the experiments revolved around the continuous software delivery to a test environment.

Results of experiments shows, that until Reusable Function Library is empty, implementation of automation by new approach is not rational – it takes quite more time. Most of this time is required for development and testing of reusable functions by pre-defined template. However, after Reusable Function Library contains all necessary Functions, implementation of automation by new approach helps to save up time ("project 2" – 36 hours, "project 3" – 52 hours, "project 4" – 33 hours, "project 5" – 44 hours). To increase benefits from provided approach, it is necessary to find how to fulfil the Library as soon as possible. It will help to reduce implementation time at "project 1".

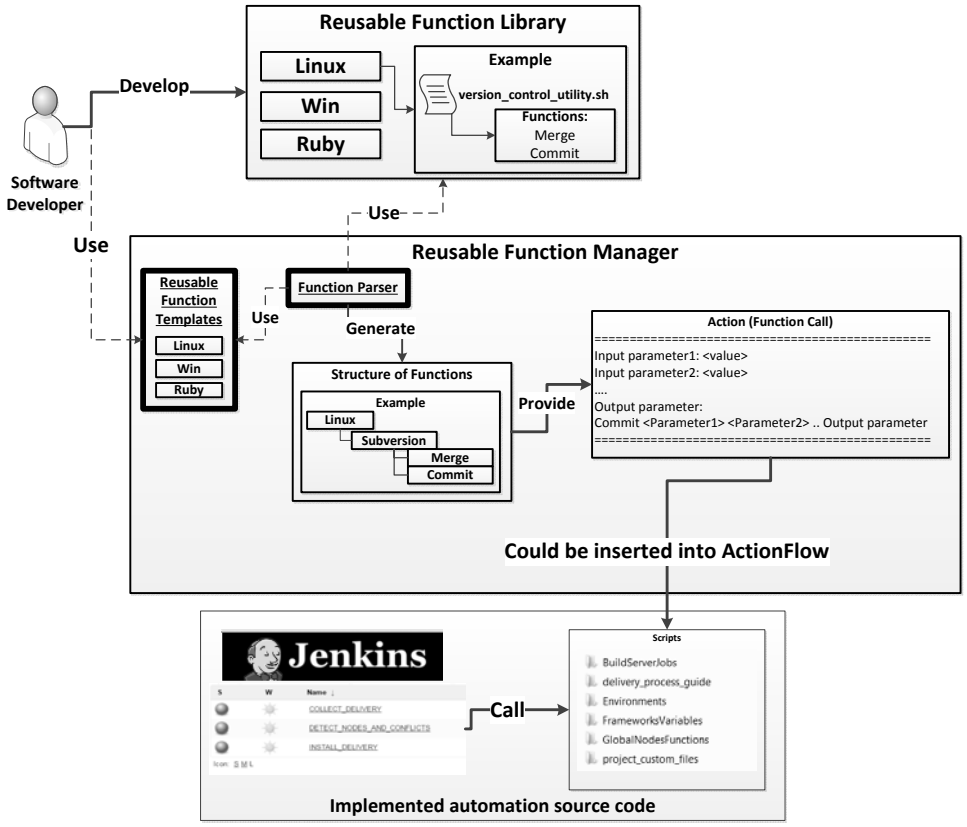


Fig. 2.2.4.1. Approach for development and use reusable function library

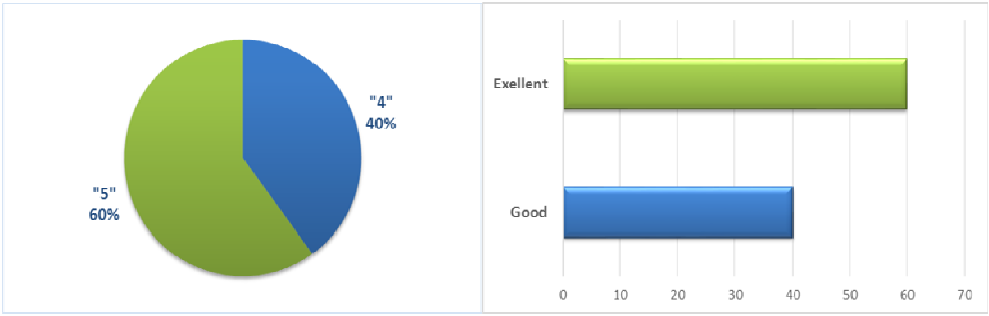


Fig. 2.2.4.2. Estimation of Web portal functionality

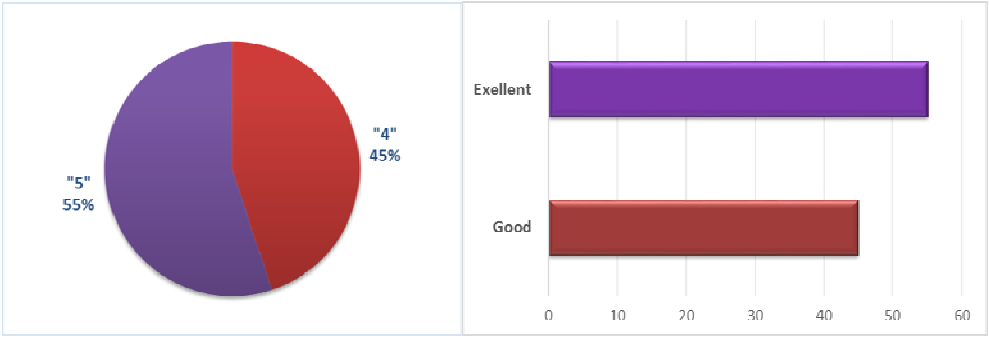


Fig. 2.2.4.3. Estimation of user interface

SCM ontology written in OWL language is used to semantically annotate configuration files. This semantic data (metadata) can be used to as input data for the Semantic Web services and also can be used to in the tasks like data integration, searching, and indexing.

2.2.4.2. Further adaptation of open Semantic Web resources (Linked Open Data, Open Calais, DB Pedia etc.) for integration of different services in Web portal

This section presents a methodology for the systematic development of RESTful Semantic Web services using SADI (Semantic Automated Discovery and Integration) framework. SADI framework was chosen after an exhaustive analysis of the available solution for Semantic Web service development.

This methodology consists of the following four general activities:

1. Define and host domain and service (input and output classes) ontologies;
2. Generate a service skeleton (i.e., Java class stub and other supporting files) using a SADI service generator;
3. Add business logic (Java classes and necessary supporting files) to the service
4. Build service and deploy produced WAR (Web application ARchive) file to an application server.

Ontology definition aims at defining concepts and associated relationships that describe the knowledge domain to which the service under development belongs. Since domain ontologies describe well-established knowledge domains, they are usually stable and can be used in the development of a number of services pertaining to the domain. During this activity, we initially search for a suitable ontology. However, in case no suitable domain ontology is identified, appropriate domain ontology is developed, preferably by reusing the existing ontologies.

The methodology was used to develop a set of Semantic Web services that are used in eLOGMAR logistic portal (available at http://www.elogmar.eu). The services use modified logistic domain ontologies LogiCO and LogiServ from TNO.nl (Fig. 2.2.4.4).

The services receive logistic-related data from usual Web services available at eLOGMAR portal and return semantically annotated data. Output of one service can be used as input in the other SADI services (within and outside of eLOGMAR portal) – this potentially allows chaining of multiple SADI services.

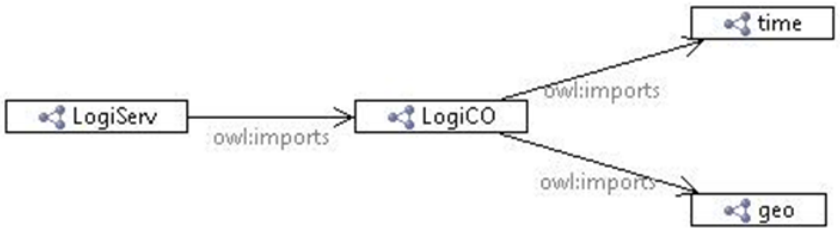


Fig. 2.2.4.4. Schematic depiction of LogiServ and LogiCO ontologies

Another developed Web service is a normal RESTful service that is used to submit textual data it receives as input to OpenCalais platform that processes the text using NLP (Natural Language Processing) methods and returns RDF formatted results identifying entities, facts and events within the text. This service's output can be used as input for other SADI services.

2.2.4.3. Research on enhancing and application of reusable source code base in Latvian IT companies. Enhancing of EAF methods

Current version of EAF approach contains the following parts:

- Platform Independent Environment Meta-model – a modelling language implemented on MetaCase platform. The scope of this language is modelling of continuous process, which should be automated later.
- Reusable Function Library – a repository with scripts and tools for automation of particular parts of continuous processes. Really, this is a physical directory in file system where reusable scripts and tools are stored under version control.
- Automation Framework – a tool, which connects together a model of continuous process, designed using Platform Independent Environment Meta-model and Reusable Function Library. The framework analyses and parses Reusable Function Library. During the next step, other parsers get structure of Platform Independent Environment Model. When all parsing are completed, user is able to choose particular automation solutions from library. Finally, source code for automation of process described in model could be generated as well representation of automation buttons.

To improve EAF and make it demanded and user-friendly, the survey has been implemented. The first goal of survey is detection of most popular tools for automation to detect scripting languages for Reusable Function Library. The second goal is determining the main continuous processes ICT companies are going to automate, current automation level of these processes and main challenges in the automation field. However, the most important question of provided survey is: "Why continuous processes are not fully automated however automation market has many modern solutions for automation?" Answer for this question could provide a vision how to make EAF approach demanded and popular for ICT companies, taking into account practical need.

The survey has 42 respondents from more than 35 ICT companies of Latvia. 60% of all respondents are members of ICT companies with 200 and more employees. It means that survey has opinion of large companies with many different software development projects and results could be believable, however 42 respondents is not a big amount for statistical analysis. The Fig. 2.2.4.5 provides overview of popular scripting languages and technologies using by respondents.

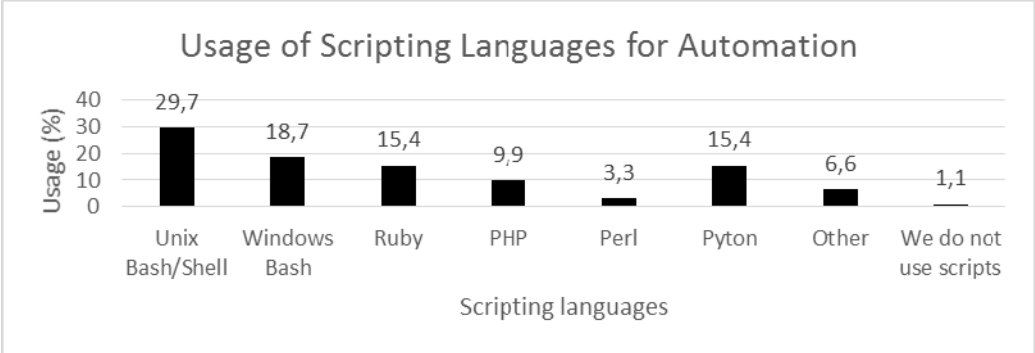


Fig. 2.2.4.5. Scripting languages for automation

The most popular scripting languages are Unix Bash/Shell, Windows Bash, Ruby and Python. It means that Reusable Function Library could be quite usable if functions for these scripting languages will be stored.

The next part of survey related to particular continuous processes. We asked to evaluate automation level for most important continuous processes:

- *Bug Tracking* – process related to updates of statuses of tickets (For example, from 'Ready Solution' to 'In Testing' after new release for test environment);
- *Version Control* – process related to operations with version control repositories to prepare baselines for software builds (merge, branch, tag, commit etc.);
- *Software Build* – process related to build software from source code;
- *Software Deployment* – process related to deploy software builds to test and production environments during new releases;
- *Software Delivery* – process related to preparing software release for production environment. This process could include such steps as creation of installation package, preparing installation guides and release notes, updates of bug tracking items, generate formal emails and notifications about new release etc.

To evaluate automation level of mentioned processes, we introduced 4 automation levels:

- *Not automated* – process is fully manual and requires human efforts for all steps of it;
- *Partially automated* – at least one step of process is automated;
- *Fully automated* – all steps of process are automated, no human efforts are required to complete process;
- *Will be automated* – process is not fully automated but company understands it and going to automate this process in future.

Evaluations of automation level are provided in Fig. 2.2.4.6.

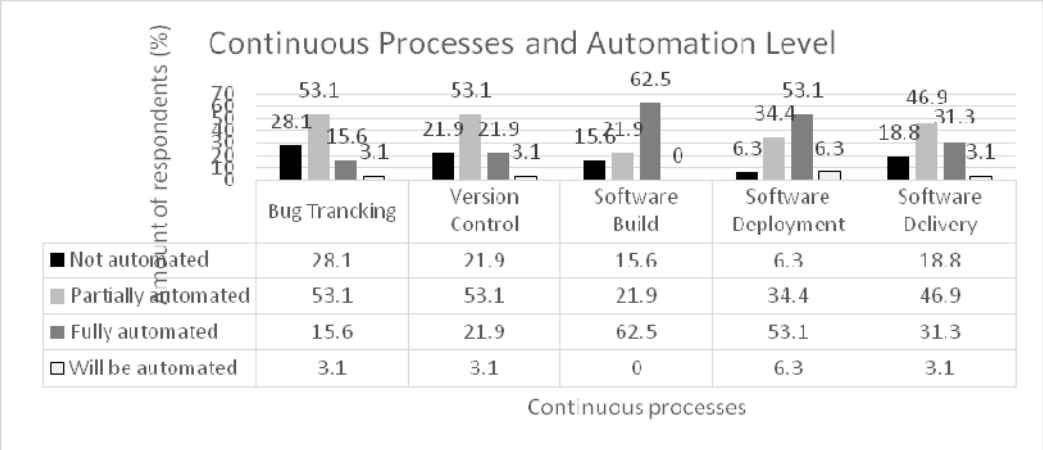


Fig. 2.2.4.6. Automation of continuous processes evaluated by respondents

After analysing of provided data, some important conclusions have been made:

- Continuous processes like “Software Builds” and “Software Deployment” have the best automation level compared with other. Only 6% of respondents still have not automated deployment process and near 16% has not automated deployment process.
- Bug Tracking has poor automation level; only 16% of respondents think that bug tracking is fully automated. The most popular bug tracking systems like JIRA and Redmine provide API interface and could be easily managed from scripts, however there are so many different build and deployment tools and we have integration problem: build and deployment tools cannot “tell” to bug tracking system which tickets should be updated and when.

The general conclusion is that companies know the own software very well and could automate at least most important steps of build and deployment. However, many respondents still have not automated such processes like bug tracking, version control and software delivery.

The next part of survey related to detect challenges in the automation field as well detects the actual problems. We could try to answer mentioned question “Why processes still not fully automated, however automation market has such tools like OpenMake, Serena, Jenkins, Puppet, Chef, Docker etc.”

To understand better actual problems and challenges in the automation field, we have asked two questions:

- What are actual problems in the automation field and how they could be fixed?
- What is a modern automation solution of second half of 21 century?

Here are some interesting answers from survey:

- “Copying source code and creation of modifications without saving integration with code base”
- “Changes in one particular module or system in context of different projects raise merge conflicts and manual merges does not allow to achieve full automated level”
- “Dependencies from 3-rd party vendors”

- "New tools and approaches related to automation are not trustable and are risky. Companies going to apply minimal requirements without including additional financial and human resources"
- "Too many definitions regarding the same things: for example, DevOps, software configuration management, continuous processes etc. Companies do not see themselves in the new tools and approaches, they do not see how new tools could improve existing process and where benefits are"
- "There is no way how to see benefits from new automation solution. For example, if I will pay something, how much money and resources I will save up in future"
- "Automation solutions in the future will be reusable and easy integrated with other solutions/tools. New integrations and components will be easy included and managed without manual code writing and some additional efforts"
- "There are going to be introduced more abstraction levels"
- "Source code will be generated automatically when machine learning will achieve new improvement level"
- "People going to improve automation by own trusted tools. They do not like revolution, they do not like to buy some unknown and not trusted, but they would like to improve process by reusing existing and trusted tools"
- "Usually companies have many different software development projects for different customers. They are going to use the same automation solutions in all projects but they do not like to share automation source code or implementation for customers or 3-rd party vendors. There is a challenge to make some automation solution reusable for different projects and customers without sharing implementation details and without additional requirements for infrastructure on the customer side"

In addition, we have asked to evaluate mentioned problems in the automation field. After analysing of this evaluation, we have found that there is no one TOP actual problem. It means that if some particular problem is actual for one respondent, it is not actual for other etc. We have summarized all answers and data of particular survey and designed a vision of modern and demanded automation solution in context of previously developed EAF approach.

Vision of modern and demanded automation solution based on results of survey

To understand better a vision of automation solution, let us challenges in the automation field:

- Why continuous processes are not fully automated, however so many automation tools exist? Because companies love theirs processes. They do not think that processes could be better and faster and do not see any benefits from new automation tools. Some error or not automated step of particular process sometimes is so usual and process owner does not think that it is error or not automated step, but something "normal and usual". For example, we can imagine technical specialist who is responsible to deploy new version of software to test environment. During each deployment, he copying 1000 files from workstation to remote server manually using SSH tools. Copying of file takes 40 minutes every day, because specialist does not use a script what can do the same by one click. If some other specialist, who can do the same by one click, will looks for this manual copying, he could tell that process is not automated and it could be absolutely. However, for technical specialist, who

do this manual copying during last 10 years every day, it is something usual and normal. He does not see any problems, he could not imagine that process could be faster and he do not like to study some new automation tools, which "maybe will improve the process".

- Let us describe a case when technical specialist knows that his own process could be better and more automated. It is a good, but how to choose correct automation solution. Sometimes presentations of new automation tools contains many definitions instead of demonstration of benefits. In addition, potential user cannot see how new solution could improve his process. Presentations of modern automation tools contains many different concepts like DevOps tools, release management tools, software deployment tools, configuration management tools etc. For example, I need a tool, which could prepare build for database delta. How I can search needed tool? I could do it by keywords like "database delta installation", "database change migration", "DevOps for database deployment", "continuous integration for database development" etc. Because no one standard exist for definitions of continuous processes automation field. It is a reason why sometimes it is a challenge to find in the huge sea of automation tools only one what actually needed. In other words, sometimes two companies could do the same thing and could call this thing differently.

Sometimes modern tools for automation suggest making changes in the usual process. Changes could be like revolution, for example installation of new unknown tools, buy new licences, refactoring of software architecture, changing structure of application and database servers and many other changes. However, this suggestion looks too risky from company side, because companies like and trust own tools and processes.

They do not like to make huge changes, install unknown tools and especially, spend resources for such refactoring. Instead of mentioned, they would like to try new automation solutions without some installations, revolutions etc. They would like to try some new features free and they would like to have rollback option in case when new automation solution will not give expected benefits.

Based on mentioned summary, new vision of EAF approach has been designed. The vision has 3 main concepts and improvement topics. The first concept is "mirror effect". The Fig. 2.2.4.7 provides the principles of "mirror effect".

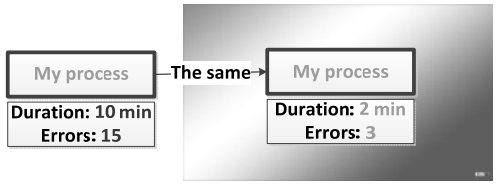


Fig. 2.2.4.7. Main principles of "mirror effect"

EAF approach should provide a service to create a model of the same process. Model should takes a meta-data from particular process, for example statistics from bug tracking system, history of version control repositories, statistics of continuous integration builds etc. Model should have information about tools, which support current process, for example JIRA, Subversion, Git, Jenkins, OpenMake Build Meister etc. The service should draw a model of the same process, which contains the same workflows and the same tools, so it called "mirror effect". User of EAF approach should see identical process but with improved characteristics and

measurements (Fig. 2.2.4.7). To demonstrate benefits of each automated solution, this solution should be applied for the equivalent process, which contains the same workflows and tools. If user will see benefits of automation solution, which is applied for his own process, these automation solution has a good chance to become trusted and implemented.

The next concept of new vision of EAF approach is the “backup mode” and cloud-oriented implementation. Cloud oriented implementation means that EAF users could call particular automation solution or framework from cloud without some additional installations on local infrastructure. The “backup mode” is a service of EAF approach, which allows making a backup of current state of all artefacts and data of process. In case when new automation solution become not acceptable for particular project, the new service provides an option to restore all artefacts. For example, if company would like to try automation solution for Git merging and branching workflows, the service allows saving Git repository state before trying of automation solution and restoring it at any time if required.

The last concept of new EAF vision is “infrastructure independent service”. It means that company could design and save each automation solution in cloud and could call it from many different infrastructures. It means that automation solutions will be reusable and implemented once, could be used anywhere. The Fig. 2.2.4.8 provides visual representation of infrastructure independent service.

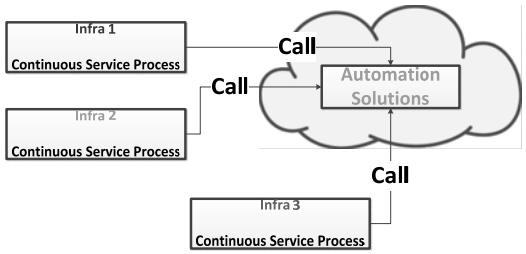


Fig. 2.2.4.8. The vision of infrastructure independent service

References

1. Novickis L., Vinichenko S., Sotnichoks M., Lesovskis A. Graph Models and GeoData Based Web Portal in Cargo Transportation. In: Scientific Journal of RTU, Applied Computer Systems, 2015/17. RTU Press, 2015, pp. 35-39.
2. Bartusevics A., Novickis L. Models for Implementation of Software Configuration Management. In: “Procedia Computer Science”, ICT in Regional Development, Vol. 43, 2015. Elsevier, pp. 3-10.

2.3. Model based data visualization and real-time verification of business processes

2.3.1. Development of technologies for large scale NoSQL data base exploration and visualization

The existing research in the field of large-scale high-resolution display wall sys-tems has mainly been focused on specific needs and environments. In many cas-es, the solutions built for large-scale high-resolution visualization are only valid for a specific use case (e.g. distributed 3D rendering, presentation of HTML con-tent, stereoscopic projection). However, such solutions can soon become outdat-ed and be left unmaintained (as seen with several distributed OpenGL library im-plementations). This happens due to the appearance of new technologies that solve the existing tasks in a more effective manner. A possible solution is to pre-vent most limitations on the set of technologies that can be used when the soft-ware is running on a display wall system. One way of achieving this is to use vir-tualization. This chapter presents Infiniviz - a virtual machine based high-resolution display wall system. Infiniviz approaches the visualization task in a seamless manner. The main aim of Infiniviz is to be able running any common desktop operating system software on a large scale high-resolution display wall without any modifications. Infiniviz achieves this by running a headless virtual machine with the required operating system backed by a custom software stack that handles the actual visualization. The authors have performed performance evaluations, virtualization environment comparisons and comparisons among other display wall architectures. This work along with key conclusions has been summarized in this chapter.

2.3.1.1. Introduction

The traditional display system of desktop PCs (a single or multiple GPUs connected to a single or multiple physical displays) often suffers from scalability limitations. These limitations have created a distinct field of research that is devoted to the construction of large-scale high-resolution display wall systems. Such display wall systems are required in environments that require a display surface with characteristics that cannot be achieved by a single physical display - either a very large physical size or a very high resolution or both.

Consider the following use case. There is an ongoing telescope image archive digitalization process at the Baldone observatory in Latvia. Old telescope images stored on plates are being scanned and stored in a digital format. The resolution of the resulting image is around 500 megapixels. Such images cannot be fully viewed on a single PC system, not even with multiple monitors. A single display with 4K resolution would allow viewing only 1.6% of the whole image. Even if higher level hardware is considered — e.g. a system of 4 GPUs with 6 outputs each at a resolution of 2560×1440 — this is still only 88 megapixels in total (17% of the whole image). Thus a simple conclusion can be derived — a pure hardware solution without any software middleware that would union multiple hardware systems in a unified display surface is simply not enough in the case of such a task.

The Reality Deck display wall [1] demonstrates a display wall system that combines hardware nodes similar to the ones described above (4 GPUs connected to 6 physical displays) to drive a unified 1.5 gigapixel display surface through a custom

software stack. The capabilities of the Reality Deck display wall system can already match the given requirements of displaying a 500-megapixel image. Thus this example demonstrates how software solutions used in the display wall systems can leverage the hardware limitations. One of the conclusions derived by the authors of the Reality Deck was that even though PowerWall, one of the first high-resolution visualization systems, was developed 20 years ago, the development process of such systems has still much to improve upon.

However, the custom software stack of the display wall system introduces compatibility issues. If the presented content is static (e.g. image or video files), it must be stored in a format supported by the software stack. In most cases, this is not an issue. However, if the content is generated in real-time by some client software issues can arise. If the client software is already developed to use an internal API of the software stack that drives the display wall, this software cannot be run on a desktop operating system or another display wall system with a different API. If the client software is developed based upon any on the standardized visualization APIs present in the desktop operating systems (e.g. OpenGL, Direct3D, Direct2D, DirectDraw, GDI), the software stack must also support that API. There are distinct display wall systems targeted to support one or another of these APIs with the most common example being OpenGL, but still, it only allows running client software based on that API. Moreover, even the OpenGL implementations expose limitations. The supported OpenGL versions are often not up to date. Moreover, it is not known whether the supported versions will ever be increased to the up-to-date ones. This clearly shows that the display wall system should not bother about this at all. In an ideal case, the display wall system should only be exposing a set of display/rendering entities that have some known capabilities and supported APIs to the client software in the same way as an operating system would expose a GPU with the same capabilities and APIs and manage these entities instead of implementing their functionality. With such a solution a maximum software compatibility and portability between desktop operating systems and display wall systems could be achieved.

After performing a survey on the field of displays walls [2], the authors of this chapter tried to solve this situation with a different approach. The existing solutions can be generalized as a software component that interacts with the client software and exposes the display wall to them through some specific APIs. Instead, why not expose the display wall to the client software through the standard means of the operating system as a large monitor? In such scenario the client software would be able to use all the standard APIs exposed by the operating system and no modifications would be needed. With this though in mind the authors of this chapter designed a virtual machine based high-resolution display wall architecture and developed a prototype. By using virtualization, this display wall system can run the client software on a virtualized instance of the operating system that the client software was developed for and provide a presentation of the content that would normally appear on the attached physical display device on the display wall instead. The virtual machine exposes a virtual display with the resolution of the physical tiled display wall surface and a virtual GPU backed by physical hardware in the host system. The actual visualization is done by GPUs present in the hardware system that hosts the virtual machine. However, instead of being displayed on an attached physical display, the content is rendered in the memory of the GPUs and then encoded in a video stream which is transmitted to client nodes that display it. In such a way the requirement for the display wall system to implement any of the visualization APIs has been lifted. The virtual machine directly exposes the capabilities and supported APIs of the hardware

present on the host system. All of the drawing and rendering calls made from the client software running on the virtual machine are directly executed on the actual hardware.

This chapter is an improved version of the authors' previous publication [3], and it is organized into six sections. The first section contains a short introduction to the chapter. The second section gives a brief description of display wall architectures and the issues present in them. The third section describes the proposed architecture of virtual machine based high-resolution display wall system, the fourth section describes Infiniviz — the display wall system based on this architecture. The fifth and sixth sections summarize the lessons that authors learned throughout the development and testing of the resulting system, and conclusions upon which to direct the further work.

2.3.1.2. Display Wall Architectures and Their Problems

To understand the factors that drive the research related to the construction of display walls one must be introduced the most important limitations in the classical PC display system. For that reason, the authors need to introduce a few terms and their explanations.

The terms *display wall* and *video wall* are used interchangeably in this and many of the related works. Both of the terms are used to describe a large sized tiled display surface. Each tile can be either a single physical display or an image from a projector. The purpose of a display wall is to create a single continuous display surface that is superior when compared to a single display or projector image regarding maximal size, resolution or both.

Visualization software denotes the software that performs either a 2D drawing or 3D rendering work the outcome of which is an image that must be represented to the user. Visualization software is the factor that creates the need for display wall whenever the produced image cannot be presented on a single display. Visualization software is not always a single user space process; a whole operating system can be perceived as visualization software since it provides a way for the processes running inside it to present visual data to the user.

In the ideal case, the visualization software should not be aware of the display environment. It should not care whether it needs to use a specific 2D drawing or 3D rendering API or another traditional API when running on the display wall or running on a traditional PC system. In the context of this chapter, *software awareness* is understood as the need for the visualization software to be aware of the display environment and change its behavior accordingly. A display wall is called *software agnostic* if it does not enforce software awareness.

Framebuffer is a continuous array of all pixels that represent the contents of an image. Framebuffer is the source of the image that is presented on a display or projector image. At the same time, it is the target of all the rendering and drawing actions performed by visualization software. Framebuffer implementations may differ, and this chapter covers research that has tried very diverse solutions.

In general, the intuitive approach to the building of a display wall is quite trivial – one would simply to create a large display surface by physically tiling multiple smaller displays or images of multiple projectors. However, that is only a part of the solution. A system that provides the signals for the tiles at one end and exposes the display wall to the visualization software that provides the content at the other end is required. Such systems can be constructed in various ways – from purely hardware

based ones that involve simple signal cascading and scaling to complex software systems involving distributed computing.

The traditional display system used in desktop PC systems consists of four components (see Figure 2.3.1.1):

1 A *visualization software* that creates the visual data to be presented to the user and interacts with the GPU or video adapter by executing 2D drawing and 3D rendering commands;

2 A *GPU* or video adapter – a hardware peripheral in a computer system that exposes and implements standardized 2D drawing and 3D rendering APIs (e.g., OpenGL, Direct3D), implements them, renders the results to an internal framebuffer and provides the display signal to a display;

3 A *display* – a device that visualizes the signal from the GPU;

4 A *medium* that connects the GPU and the display.

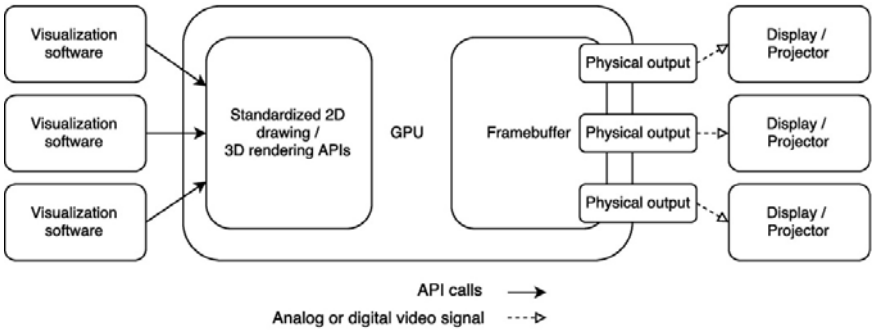


Fig. 2.3.1.1. The traditional PC display system.

Each of these four components has some limitations in regards to the resolution of the visualization on display. The GPU has a limited maximum resolution that is enforced by the memory available on the GPU and the medium used for signal delivery. The medium which is used to interconnect the GPU and the display has some throughput limitation which limits the amount of data that can pass through in a single unit of time. The display as well has a limited maximum resolution depending on the limits of the chosen visualization technology (e.g. the maximum resolution of the matrix) and the medium that delivers the signal.

Such dependencies among the components hold back the general progress — e.g. why would a hardware vendor create a GPU with a resolution that no medium could deliver or no display could visualize? Thus due to these limitations domains that require large high-resolution display surfaces created the need to tackle this issue. That was done by creating hardware peripherals and software systems that leverage and combine the possibilities of the existing GPUs and untraditional mediums.

The mentioned limitations regarding resolution and size are a direct problem for static content like images. However, another aspect that introduced the need for an alternative to the traditional PC display system was the computing power difference between CPUs and GPUs — a mismatch between the speed at which visualization software generates the image and the speed at which the GPU can render it. For example, graphical representations of fluid flow simulations done by supercomputers could not be visualized by a single GPU at a real time. Thus again a scalable rendering solution that could be expanded to meet the needs to visualize the generated graphical data at the real time was needed.

Even though the hardware peripherals evolve they still impose some kind of maximum limit of displays that can be connected. Moreover, removing constraints by

providing flexible and scalable solutions is one of the general aims of any research on display walls. A good example to back this statement is the Reality Deck [1] — the world's currently largest display wall that has a resolution of 1.5 gigapixels. Pure hardware could not achieve such resolution — the Reality Deck is built upon hardware nodes that reach the available hardware constraints and driven by scientifically developed software that composes these nodes in a unified framebuffer.

One of the main characteristics of a display wall system is how the display surface is exposed to the visualization software. To achieve the task of being software agnostic the display wall system must expose itself as similar as possible to the classical PC display system. There have been several approaches to this task:

- Exposing the display wall through a custom implementation of widely used 3D rendering libraries, most often OpenGL;
- Exposing the display wall as additional displays by hooking in the GPU driver. This is a path chosen by many software products that allow devices like phones, tablets, and TVs to be used as additional displays for a PC system. However, this requires a different implementation on each operating system and leaves rarely used operating systems unsupported. Moreover, since the operating system vendors can make arbitrary changes to the device driver architecture, such solutions may be rendered dysfunctional after releases of newer versions of the operating system. A good example in this case is Microsoft Windows. Microsoft Windows XP had a driver model name XPDM that supported using mirror drivers. A mirror driver allowed duplicating the output of the primary display driver to a custom memory region. Many software products were developed to support attachment of custom display nodes by using this functionality. However with Windows Vista a new driver model called WDM was introduced that added new requirements for the mirror driver implementations and thus the software products developed for XPDM had to be modified to be still supported;
- Providing a custom library that exposes API for the visualization software to interact with the display wall.

The intuitive assumption that could arise is why not simply provide a fully virtual GPU and hide all the display wall related internals inside the driver? This is a valid approach to the issue but again becomes complicated due to need to handle different operating system architectures. Operating systems expect some amount of hardware presence (interrupt handling, DMA, etc.) from a GPU and implementing this could be problematic due to the existence of closed source operating systems. However, it must be noted that such a solution could fully mimic the traditional PC display system.

Most of the previously developed display wall systems try to solve the mentioned hardware scalability limitations by separating the many functions of the GPU among several components that are usually implemented as separate computer systems (see Figure 2.3.1.2). For more scalability clusters of such systems can be used for each task instead of a single instance. For example, to increase the rendering power the display wall systems either expose custom 2D drawing or 3D rendering APIs or provide their implementations for the standardized ones like OpenGL. Then the display wall software stack forwards these calls to distributed rendering nodes thus increasing the rendering power. Further down the pipeline the pixel regions produced by the rendering nodes are reassembled and synchronized by one or many framebuffer components. The framebuffer components then present the complete image on the display wall.

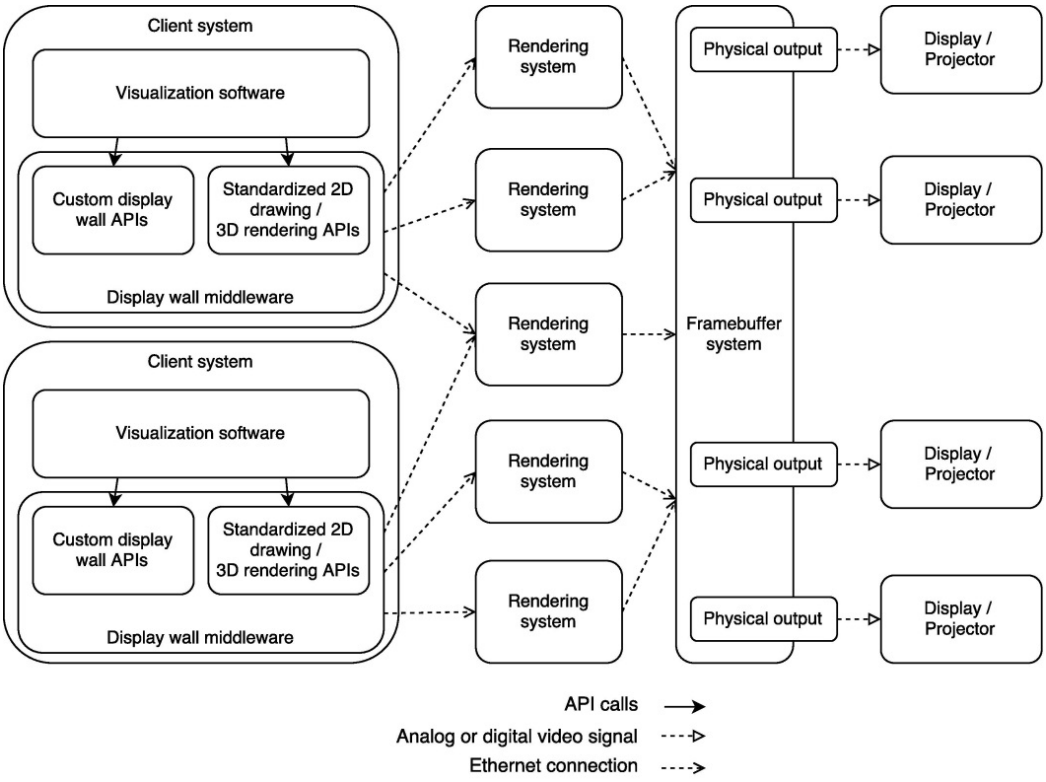


Fig. 2.3.1.2. Generalization of the existing display wall system architectures.

The research in the display wall construction domain has given birth to numerous display wall systems like Reality Deck [1], SAGE [4], SAGE2 [5], DisplayCluster [6], Chromium [7], WireGL [8], Equalizer [9], CGLX [10], XMegaWall [11], SGE [12], and others.

A detailed analysis of all these display wall solutions is outside the scope of this chapter. However to provide a ground for comparison with the display wall architecture proposed by the authors of this chapter a brief introduction is needed.

In general, the existing solutions can be divided in two groups:

- Ones that expose themselves as OpenGL implementations (WireGL, Chromium, Equalizer, CGLX);
- Ones that perform pixel streaming from client workstations, host their applications that can accept external input and provide custom or standardized frameworks for the adoption of existing software (SAGE/SAGE2, DisplayCluster, XMegaWall, Reality Deck). SAGE2 that offers to host HTML5 content directly on the display wall system is a good example of using standardized technologies other than OpenGL for software adoption on a display wall.

2.3.1.3 Virtual Machine Based High-Resolution Display Wall Architecture

The authors of this chapter propose a new display wall architecture (see Figure 2.3.1.3) [13]. This architecture in contrast to most of the current systems does not host

the framebuffer outside of the client system. Instead, it uses virtualization for the

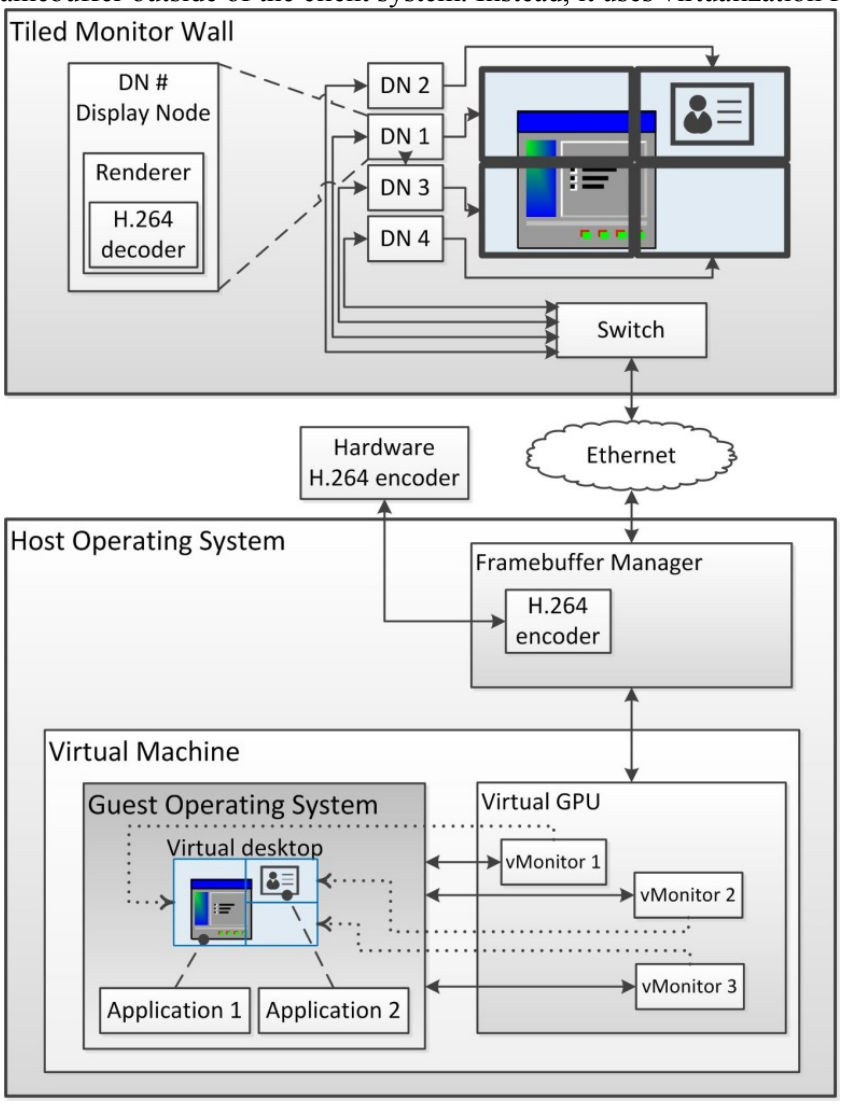


Fig. 2.3.1.3. The proposed display wall architecture.

framebuffer implementation. It implements a virtualized GPU that works on top of one or many physical GPUs. This approach allows removing a hard dependency among physical outputs on the physical GPUs and the size of the display surface available. The proposed architecture does this in a manner which hides this fact from the visualization software and sets no specific requirements on it. Any visualization software running in the client system interacts with a virtualized GPU that works just like a normal GPU and exposes all the standard 2D drawing and 3D rendering APIs. Underneath a custom display wall software stack implements these calls by using the physically available GPUs in the system thus allowing efficient scaling by adding more GPUs to the system in the case if the previously described gap between the computing power and rendering power is encountered. The rendered data is then encoded in a video stream that is transmitted over Ethernet to a display endpoint system where it is decompressed and displayed on a connected display or projector. To satisfy the needs of a multi-client environment, the display endpoint can receive different independent streams and display them in a layered mode.

GPU virtualization has become a trending technique in the virtualization technologies. Leading solution providers like VMware and VirtualBox have implemented a way to provide the acceleration features like Direct3D and OpenGL support of the host GPU directly to the virtualized guest operating system. Similar technology is provided for XEN by NVIDIA vGPU and RemoteFX for Microsoft Hyper-V. The GPU virtualization technology has successfully helped to utilize a single GPU in a multi-operating system environment.

This approach can be applied to solve the issues that exist in the field of display walls. If the virtualization technology provides the guest system with a purely simulated and freely configurable GPU that supports hardware acceleration for 3D APIs like OpenGL and Direct 3D by using the physically available GPUs this can solve both problems — remove the dependencies on the physically available video outputs and increase hardware utilization.

Let us look at the architecture in more detail. The physical host system runs a software stack currently denoted as Framebuffer Manager and some kind of virtualization platform which in turn hosts the guest operating system that is running the content that needs to be visualized on the display wall. The virtualization platform simulates a virtual GPU that can be freely configured in terms of virtual monitors and resolutions to exactly match each desired use case depending on the amount of data that needs to be visualized. The virtualization platform interacts with the Framebuffer Manager software stack by providing notifications about drawing operations on the guest operating system and access to the video memory contents of the virtual GPU. The Framebuffer Manager itself performs event-driven management of the framebuffers and handles (crops/scales) the mapping of image data from the virtual monitors to the display nodes in the monitor wall. After the logical partitioning of the image, the Framebuffer Manager uses hardware-based video encoding capabilities in the host system to encode the image and provide an encoded video stream to each display node. The mapping between display nodes and virtual monitors is flexible and there are no hard constraints. As seen in the schematic of the architecture (see Figure 3), a virtual monitor can be mapped to a single or multiple display nodes. The given example shows that virtual monitor vMonitor 1 actually takes up two physical display nodes DN 1, DN 3, while virtual monitors vMonitor 2 and vMonitor 3 are each displayed on a single display node DN 2 and DN 4 respectively.

The proposed architecture removes direct dependencies between the needed monitor setup and presence of physical GPUs — all of this is taken care by the Framebuffer Manager and virtualization platform. The Framebuffer Manager takes care of using the present hardware for video encoding and 3D acceleration support for the virtualization platform, while the virtualization platform provides unconstrained display and resolution configuration options.

2.3.1.4. Infiniviz - Virtual Machine Based High Resolution Display Wall System

The next logical step for the authors of this chapter was to create a working display wall system based on the virtual machine based high-resolution display wall architecture [14]. This lead to the creation of Infiniviz. *Infiniviz* is the Framebuffer Manager software stack, built upon VirtualBox for reasons explained further on. Authors of this chapter deployed Infiniviz on a physical tiled display wall consisting of 5x5 22” monitors and two different host systems:

- one with moderate hardware — Gigabyte Brix Pro mini PC (Intel Core i7 4770R CPU (4 physical cores, 8 virtual cores at 3.2 GHz), Intel Iris 5200 Pro GPU, 12 GB of RAM and Windows 8.1);
- one with high-end hardware — 2 Intel Xeon e5-2630 v2 CPUs (12 physical cores, 24 virtual cores at 2.60 GHz) , NVIDIA Quadro K4200, Windows 8.1.

Both host systems run VirtualBox as the virtualization platform with both Windows and Linux guests. VirtualBox was chosen for two reasons. First, it was one of the few open source virtualization systems that had support for a configurable virtualized GPU. Second, it provided the best scalability options in terms of the total resolution while other virtualization systems were capped at lower limits. Table **2.3.1.**1 describes these limitations in more detail.

Table 2.3.1.2. Resolution limitations of virtualization systems.

Vendor	**Maximum resolution for a homogenous surface**	**Comments**
NVIDIA vGPU	16 megapixels (8 displays at 2560×1600)	The mentioned results are based on the NVIDIA GRID K260Q card, other cards provide lower capabilities
RemoteFX (Microsoft Hyper-V)	10 megapixels (8 displays at 1280×1024)	Windows Server 2012 R2 host operat-ing system and Windows 8/8.1 guest operating system
VMWare vSGA	4 megapixels (2 displays at 1920×1200)	Only Windows guest operating systems
Oracle VirtualBox	Any configuration that can fit in the video memory of the virtualized	The video memory of the virtualized GPU currently cannot exceed 256MB

The authors have developed a Framebuffer Manager implementation that runs alongside VirtualBox on the host operating system and collects the image data from the framebuffers of the simulated GPU, encodes them into a H.264 stream with either the Intel Iris 5200 Pro or the NVIDIA Quadro K4200 GPU. After the video has been encoded the Framebuffer Manager streams it over a Gigabit Ethernet to the monitor wall.

The monitor wall itself consists of 25 22” DELL displays, each of which is driven by a Raspberry Pi A model device. The Raspberry Pi devices were chosen to implement the role of the display node because of the low cost, efficient power usage and ability to decode a 1920×1000 H.264 at acceptable frame rates for live streaming. The Raspberry Pi units are poorest regarding scalability in this prototype since they support H.264 decoding only up to the resolution of 1920×1080 meaning using displays with higher resolution is not possible in the current prototype.

5

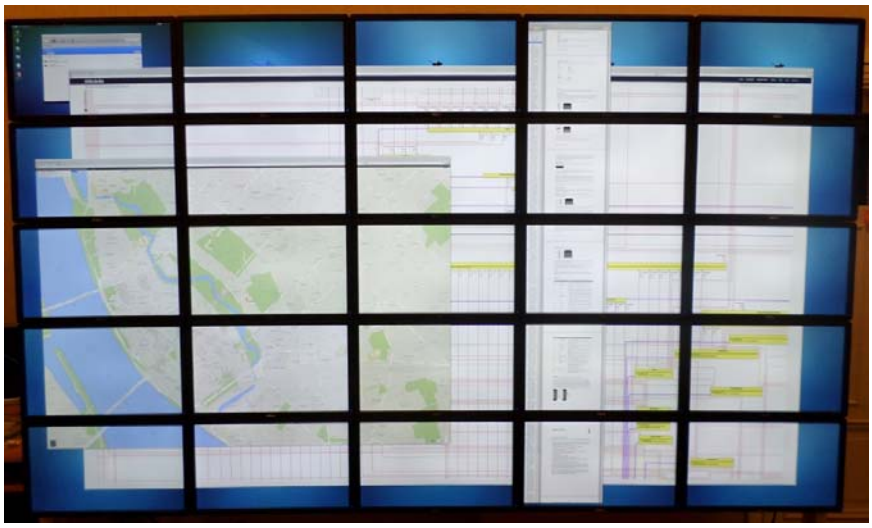


Fig. 2.3.1.4. The prototype running Xubuntu 14.04.

6

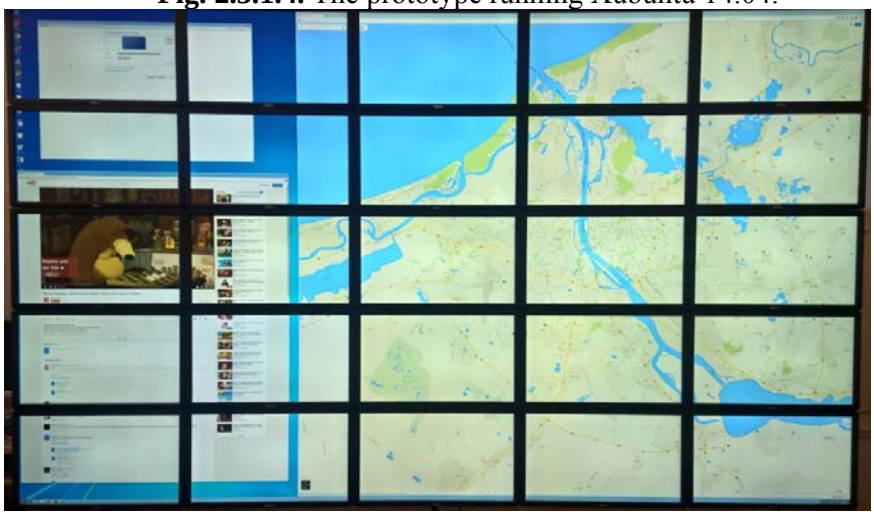


Fig. 2.3.1.5. The prototype running Windows 7.

The Infiniviz software stack is developed in a platform independent manner - it can run on both Windows and Linux host systems. However, due to security reasons, only VirtualBox versions older than 4.3.8 can be used on Windows without compilation from sources. Since there is no such limitation on Linux the authors mainly favour Linux as the host operating system. The guest operating systems had been run in two modes - simulating 25 independent displays 1920×1080 and one large 9600×5400 display. The Figure 2.3.1.4 demonstrates Xubuntu 14.04 guest operating system running common applications with static content like Google Maps and SVG based graphs in Firefox and a PDF viewer. All these applications seemed to work without issues regarding the high display area. The Figure 2.3.1.5 demonstrates Windows 7 operating system running Chrome with Google Maps and Youtube. Also, there is an open Control Panel instance showing that the system is being told that it is using a single display with the resolution of 9600×5400 pixels.

2.3.1.5. Lessons Learned

All the described development and evaluation process lead the authors to several conclusions and observations.

Software agnostic display wall environment is hard to achieve due to the multiple versions of the 3D rendering APIs that need to be supported. Initially, VirtualBox seemed to be a great choice regarding support for the most popular 3D rendering APIs — OpenGL and Direct3D. For OpenGL, VirtualBox has a code layer that dispatches the OpenGL calls made in the guest operating system to the hardware on the host operating system. The results are rendered into off-screen pixel buffers that are passed back to the guest. Such approach has the benefit of exposing the correct OpenGL version and capabilities of the hardware present on the host system. However, the performance suffers from the fact that the rendered frames are pushed from GPU memory to RAM after each rendering operation. This step cannot be omitted if the produced image needs to be displayed on the host system. However, in a display wall environment, VirtualBox code could be patched so that the rendered frames are left in OpenGL surfaces on the GPU and the video encoding is performed directly on these surfaces. Direct3D, on the other hand, has a much more complex implementation. To provide support for Windows guests on all types of host systems, VirtualBox uses Wine to map the incoming Direct3D calls to OpenGL calls and then execute them on the host system. This is where the limitations come in. Currently, the latest supported version in VirtualBox is Direct3D 9 while most software already requires Direct3D 10 and newer. Moreover, there are no guarantees that the newer versions of Direct3D can be mapped to OpenGL in a fully functional way. Again as stated for OpenGL — the display wall environment could introduce some restrictions that could allow easier solutions. For example, allowing only *Windows guest / Windows host* combination for Windows guests. In such case, a direct Direct3D call dispatch mechanism could be implemented similarly to OpenGL.

Video compression codec choice for real-time video streaming is not as obvious as it may seem. Currently, most of the multimedia content available either in a stored or live format is encoded with the leading inter-frame ISO/IEC Moving Pictures Experts Group (MPEG) codecs — H.264 and H.265. Both of the codecs are so widespread because they offer better compression ratios compared to older intra-frame compression methods like JPEG [15]. Since size is the most important factor for media that is streamed over the Internet the ability to sustain real-time encoding and decoding is not prioritized. However, in the case of desktop capture, these factors switch places. VNC systems still use JPEG for transmitting the content since JPEG performs better in the scenario where only small regions in the picture change from frame to frame. Both H.264 and H.265 encode the full frame each time (there are possible future improvements here since NVIDIA NVENC supports region of interest encoding which has not been yet implemented in the virtual machine based display wall system). JPEG can only encode the dirty regions of a frame. Moreover, in the case where the only thing moving in the picture is the mouse pointer, this makes a great difference. However, the content of the desktop can swiftly change from static to very dynamic — and with dynamic content JPEG produces bitstreams with much higher size than H.264 or H.265. Thus, the lesson here is that for optimum performance the display wall software stack must use a hybrid encoding mode that is able to use the best codec for each type of content and perform real-time switching among them.

The hardware acceleration on GPUs will continue to evolve making the proposed display wall architecture more and more efficient. Currently the hardware accelerated video encoding available in the evaluated GPUs (Intel Quick Sync in Intel Iris Pro

5200 and NVIDIA NVENC in NVIDIA Quadro K4200) is not powerful enough to provide real-time encoding for large-scale display wall systems. However, according to the roadmaps of both hardware vendors, the performance of the GPUs will increase very fast and will eventually exceed the CPU based video encoding technologies. For example, the official NVIDIA statistics state that the number of 1920×1080 frames encoded per second on the fastest encoding mode has increased from 227 to 648 during four generations of GPUs (from Kepler to Pascal).

The main limit of the proposed architecture is the amount of virtual GPU memory in VirtualBox which can be lifted or at least increased to a sufficient amount. The achieved results regarding the total display resolution from the evaluated prototype may not be impressive if compared to the Reality Deck. However, the only limitation here is the hardcoded virtual GPU memory limit in VirtualBox. Thus, by removing that and providing sufficient amounts of RAM, this display wall architecture can provide higher resolution with only one GPU than a single node used in the Reality Deck.

The modern operating systems are still somewhat incapable of handling large amounts of displays with the built-in mechanisms. As seen on the prototype, both Linux and Windows are not yet capable to flawlessly support a large number of displays. Windows 7 could not identify more than 16 displays. Even though using a single display with a large resolution is a better choice at least from the ergonomic point of view there may be scenarios where using many separate virtual displays is required. The maximum limits of the operating systems must be determined to understand when the operating system itself becomes the bottleneck.

Even running common software like web browsers demonstrated issues in an environment with such a high display resolution. For example, Chrome which seems to have an internal limitation of the maximum memory it allocates on the GPU stopped rendering content when the window was resized big enough that the contents could not fit in this memory limitation. Also, Javascript based drawing started to get quite CPU intensive at such a resolution.

Display wall systems are more vulnerable to synchronization problems among display nodes. GPUs and screen splitters can easily synchronize all of the attached physical screens at hardware level — either by using a common clock on all of the outputs or by other means. If the display nodes of a display wall system are not interconnected, it is hard for them to guarantee a simultaneous display of the current frame. Even if a frame is divided and transmitted with minimum overhead on the host side, additional latency can be introduced on the network level and the processing side. The authors have considered using some kind of shared hardware clock among the Raspberry Pi endpoint devices that would work as a common frame display refresh trigger. However, such a mechanism has not been implemented yet.

2.3.1.6. Conclusion

The display wall domain is very heterogeneous. There are several solutions to the same general problem but each targeted at some specific use case or the environment. Due to the fact that the use of display walls has not been widespread — in most cases, they are used in scientific or research facilities to either view some static high-resolution content or display real-time simulations — the requirement to use custom crafted client software to generate the visual content has not been a limiting factor.

The authors of this chapter have proposed a software agnostic display wall architecture. An implementation of this architecture is able to run and visualize the binary version of any client software that runs on a given operating system without any modifications. This is achieved by running the corresponding client software in a virtualized instance of the required operating system. The virtualized operating system exposes a virtual GPU and a set of virtual displays. The virtual GPU matches the capabilities of the actual host hardware except for maximum resolution, which is defined as needed by the given physical display configuration. Any calls made to the virtual GPU are actually handled on the physical GPUs. The result is properly divided into regions corresponding to the physical display configuration, encoded into a live video stream and transmitted to the physical display nodes. Moreover, the authors have developed Infiniviz — a virtual machine based display wall system based on this architecture and proven it to work.

This chapter contains evidence gathered from the prototype of this architecture that the given premise is valid — the authors were able to virtualize Linux and Windows operating systems, simulate a virtual GPU with a 52-megapixel resolution and present the contents of this virtualized desktop on a physical display wall. No software incompatibilities were encountered while running both casual desktop software (web browsers and office applications) and domain specific software (CCTV surveillance system).

The issues that were encountered are only relevant to performance — the limitation of the maximum resolution is capped by the limit of the virtual GPU video memory in VirtualBox and the flawlessness of the interaction is limited by the real-time video encoding performance of the physical GPU on the host system. Both of these will decrease with further releases of GPU architectures and VirtualBox.

If compared to the existing solutions the proposed architecture reduces the complexity of the hardware — no distributed systems are involved — just one host system with a set of attached display nodes. This eases deployment and maintenance. The virtualization allows running any client software opposed to the limitations of using OpenGL or custom APIs imposed by other display wall systems.

References

1. C. Papadopoulos, K. Petkov, A. E. Kaufman, and K. Mueller, “The Reality Deck — an immersive gigapixel display.” *IEEE computer graphics and applications*, vol. 35, no. 1, pp. 33–45, 2015.
2. R. Bundulis and G. Arnicans, “Architectural and technological issues in the field of multiple monitor display technologies,” in *Databases and information systems vii: Selected papers from the tenth international baltic conference, db&IS 2012*, 2013, vol. 249, p. 317.
3. R. Bundulis and G. Arnicans, “Conclusions from the evaluation of virtual machine based high resolution display wall system,” in *Databases and information systems: 12th international baltic conference, db&IS 2016, riga, latvia, july 4-6, 2016, proceedings*, G. Arnicans, V. Arnicane, J. Borzovs, and L. Niedrite, Eds. Cham: Springer International Publishing, 2016, pp. 211–225.
4. L. Renambot, A. Rao,. Singh, B. Jeong,. Krishnaprasad, V. Vishwanath, V. Chandrasekhar, N. Schwarz, A. Spale, C. Zhang, and others, “Sage: The scalable adaptive graphics environment,” in *Proceedings of wace*, 2004, vol. 9, pp. 2004–09.
5. L. Renambot, T. Marrinan, J. Aurisano, A. Nishimoto, V. Mateevitsi, K. Bharadwaj, L. Long, A. Johnson, M. Brown, and J. Leigh, “SAGE2: A

collaboration portal for scalable resolution displays," *Future Generation Computer Systems*, vol. 54, pp. 296–305, 2016.
6. G. P. Johnson, G. D. Abram, B. Westing, P. Navr'til, and K. Gaither, "Displaycluster: An interactive visualization environment for tiled displays," in *2012 ieee international conference on cluster computing*, 2012, pp. 239–247.
7. G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski, "Chromium: A stream-processing framework for interactive rendering on clusters," *ACM transactions on graphics (TOG)*, vol. 21, no. 3, pp. 693–702, 2002.
8. G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan, "WireGL: A scalable graphics system for clusters," in *Proceedings of the 28th annual conference on computer graphics and interactive techniques*, 2001, pp. 129–140.
9. S. Eilemann, M. Makhinya, and R. Pajarola, "Equalizer: A scalable parallel rendering framework," *IEEE transactions on visualization and computer graphics*, vol. 15, no. 3, pp. 436–452, 2009.
10. K.-U. Doerr and F. Kuester, "CGLX: A scalable, high-performance visualization framework for networked display environments," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 3, pp. 320–332, 2011.
11. Y.-B. Kang and K.-J. Chae, "XMegaWall: A super high-resolution tiled display using a PC cluster," in *Proceedings of computer graphics international*, 2007, pp. 29–36.
12. K. A. Perrine and D. R. Jones, "Parallel graphics and interactivity with the scaleable graphics engine," in *Proceedings of the 2001 acm/ieee conference on supercomputing*, 2001, pp. 5–5.
13. R. Bundulis and. Arnicans, "Concept of virtual machine based high resolution display wall," in *Information, electronic and electrical engineering (aieee), 2014 ieee 2nd workshop on advances in*, 2014, pp. 1–6.
14. R. Bundulis and G. Arnicans, "Virtual machine based high resolution display wall: Experiments on proof of concept," in *Proceedings of the international conference on systems, computing sciences and software engineering (scss 14)*, 2014.

R. Bundulis and G. Arnicans, "Use of H.264 real-time video encoding to reduce display wall system bandwidth consumption," in *Information, electronic and electrical engineering (aieee), 2015 ieee 3rd workshop on advances in*, 2015, pp. 1–6.

2.3.2. Business process runtime verification

The chapter is devoted to the topics of self-management and its implementation. Self-management features are intended to support the usage and maintenance processes in information systems life cycle. Four self-management types are analysed in the paper. Run-time verification and environment testing can be implemented without any intervention in base business processes, while self-testing and business process incorporation in system require an instrumentation of the base business processes. The approach is applied in praxis and shows that the implementation of self-management features requires relatively modest resources.

2.3.2.1. Introduction

The man-kind's progress of information technologies has brought up the complexity of computing systems. The IBM autonomic computing manifesto [1] claims: “It’s time to design and build computing systems capable to manage themselves, adjusting to varying circumstances, and preparing their resources to handle most efficiently the workloads we put upon them”.

One of the possible solutions of this problem is to entrust at least some of complex IT supervisory processes to the systems themselves. IBM autonomic computing manifesto defines the self-management by four fundamental self–* features: self–configuration, self–healing, self–optimization and self–protection. Later [2] the “self–chop” was extended to eight self–management properties. Today the number of identified self–management properties reaches 20 and more [3].

There are two possible ways trying to implement the self-management: (1) to construct an “autonomous supervisor” – to develop autonomous information system for supervision of other computer systems, or (2) to implement the properties by adding “independent” components (or add-ins) to the system.

The autonomous supervisor idea is consistent with the nature of the autonomic computing – independent autonomic components like natural live organisms solve self–management problems even without knowing about existence of other components. Researchers [4] have tried to create a universal supervising component allowing the implementation of almost all self-properties. Such a universal approach unfortunately is for the time being faced with serious difficulties - each individual self-property requires a specific approach, and therefore the individual implementations are rather preferable. It also meets the strategy of continuous step-by-step software development without attempting to construct the entire system as a whole at the very beginning

The implementation of self-management properties by adding independent components to information systems is a rather practically oriented approach. A group of specific solutions should be developed, partially built-in into the target systems, to enhance systems with self-management properties.

There are two bigger groups of self–management properties, referred to as smart technologies in [5]:

- self–management properties to support information system operation: (1) run-time verification - control whether internal processes executing is compliance with business measures, (2) environment testing - control of interaction with the external environment;

- self–management properties for maintenance support (3) business model incorporation – built-in business process model lets to change the functionality of the information system by updating the business process descriptions, (4) self-testing – built-in testing support component for internal whether system process control usable also in a productive mode.

Since each self–management property is simple to implement, the approach of smart technologies becomes rather applicable and it can be used even by rather small team of developers or companies (40-50 employees). This conclusion is based on the authors' subjective experience and the research [6]. However, resource assessments by other companies may be radically different from those as the implementation of smart technologies requires deep understanding of software engineering methods and the architecture of the specific software solution.

At the same time, it is the weak point of this approach: self–management properties may be developed practically from the scratch for every information system. To minimize this impact, the authors, suggest developing self–management properties as ideas applicable for wide spectra of systems. Thereby this paper discusses smart technologies which bring software development towards the objectives of IBM autonomic computing manifesto.

This chapter is a continuation of the research described in [7] and contains more detailed characteristics of implementation of smart technologies. Structure of paper – the second section deals with related research; the third section describes smart technologies for systems' operation; the fourth section describes smart technologies for systems' maintenance.

2.3.2.2 Related work

Autonomic computing and smart technologies have a similar goal – to reduce the complexity of system use by delegating some part of user support functions to the information system itself. The autonomic computing manifesto declares a vision of fully independent computer systems that are able to self-management. The closest from the perspective of this paper is self–diagnosis – a system's ability to analyze itself in order to identify existing problems or to anticipate potential issues. Some of self-properties discussed in this paper are contained by self–diagnosis: run-time verification and testing of execution environment. These properties support system users during its runtime: at first users can verify if the system is running correctly in changing environment and then – whether all of business tasks are accomplished correctly.

Other two of smart technologies are aimed to the support of system maintenance process. The first of them is self–testing: it intends to include testing components into system itself. Practical experience shows that this solution helps to verify software correctness not only during software development likewise it would be done by traditional testing tools but also after the system is taken in production and the maintenance has been begun. However, it helps to verify systems correctness in real productive environments.

The last one of smart technologies, business model incorporation, applies to the self–configuring: it is a system's ability to (re)configure itself by (re)setting its internal parameter values to achieve high-level policies or business goals [3]. The existence of this property potentiates implementing the principles of Model Driven Development (MDD). MDD provides business process model integration with

computer system, thus allowing updating systems functionality by changing business processes definitions.

As of now, manifesto's targets have been met only to some extent. Paradoxically, to solve the problem — make things simpler for administrators and users of IT — you need to create more complex systems. Continuing efforts on autonomic systems include both, theoretical research and practical implementation [8]. Although many of self–properties are introduced, there is still place for innovative implementations [9]. Many of these are provided as individual compact solutions like smart technologies.

2.3.2.3 Self-management for system operation

The studies of system life cycle [11] usually focus on the problems of system development. Usually software developer teams are IT professionals and experts therefore they have practically no problems with use of complex technologies. Many of them consider implementing of several non-functional features like self–management properties to be a waste of time and resources.

On the other hand the complex technologies of nowadays lead to complex solutions with awkward usability. Thereby information systems sooner or later are upgraded to improve their usability. The "ordinary" users of information systems become a target audience of self–management properties because they often have difficulties in overcoming of IT complexity. On this account the main focus of the next chapter discussing two smart technology features will be devoted to the support issues for better and easier information system's operation (exploitation) instead of software development phases.

2.3.2.3.1 Runtime Verification

Context

The business process runtime verification is the self-management property which allows verifying whether the business process is executed correctly and in compliance with all of time restrictions. This property is particularly useful when business process is supported by two or more loosely coupled information systems or some of business process steps are not automated at all.

Likewise, runtime verification is required to prevent conflicts of different processes and systems in collaboration, where one part of the process is done by people, and the other part is supported by software. The software can be designed to support particular processes in different environments at different time frames.

Runtime verification has been well known for years in the area of embedded systems. It is an approach to computing system analysis and execution based on extracting information from a running system and using this information to detect and possibly to react to observed behaviors satisfying or violating certain properties. Such defines mechanisms may be included in the system during its development or they may be included as independent controls from the base process. The independent character of such mechanism allows making later adjustments by adding or disabling the controlling component when a system is developed, and changes are made. These ideas can be applied in business process runtime verification, too.

Solution

Authors [11] propose a solution for business process runtime verification (see Figure 1) using three objects - verification model/description, agents and controller:

- a verification description contains instructions about the correct execution of the base process;
- an agent is a software module for registering of base process execution events;
- a controller compares the events received from agents with the permissible (“correct”) events described in the verification model. As a result, the controller may discover the incorrect behavior of base processes. If inconsistencies are detected, the controller sends messages to the user.

If the base process is already described by a graphical model, the verification process can be created from the base model by indicating those process steps which will be carried out in the runtime verification process.

Implementation

The developed runtime verification solution was piloted in bank's electronic clearing system (ECS). It identified file processing delays and bottlenecks. Authors propose to build runtime verification solution as asynchronous multi-agent mechanism: asynchronous – it runs in parallel with process being verified (base process), multi-agent – processes are observed by multiple agents, providing information for verification controller.

The proposed solution is intended to "track" verifiable processes by following their “footprints". For instance, if system exports some data, it will leave “footprints” in the filesystem, i.e., a new file will be created. From the verification perspective it is important to establish the fact of file creation, e.g., notice filesystem events "new file created". Similarly, other events can also be specified for verification purposes: New database record inserted, File is deleted, File is modified, E-mail is sent, Content of web page is changed, Etc.

These and other events may be used for process verification and in order to confirm process step execution. Therefore, verification could be done by keeping track of external events. Proposed verification mechanism contains two components:

- Controller. This component is centralized and provides verification mission it compares noticed events with process verification description, i.e., what could happen during the execution of the process instance. Controller, knowing the expected process events, requests agents to record the relevant process events.
- Agents. They are verification mechanism components, which observe process execution environment at the request of the controller and reports, if relevant events are detected. Asynchronous two-way agent and controller collaboration is effective, if process is running in eventful environment and only some of them are useful for verification (e.g., only few filesystem events will be relevant to one system). However, if all events could be useful for verification, simple agents with one-way collaboration can be used – these agents report about all detected events without controller’s request.

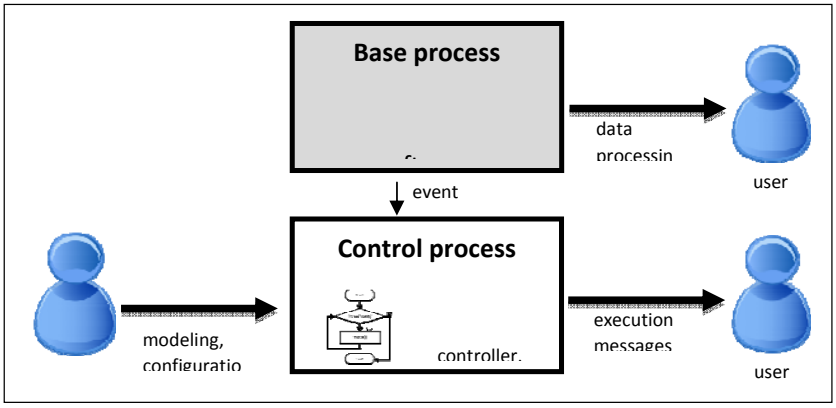


Figure 2.3.2.1. Smart technology component: Runtime verification [7]**.**

The verification solution is designed to support freely expandable list of agents: agents may be implemented for different environments according to the needs of verification. Authors have developed compact xml–based domain–specific language that enables specification of process verification. The language contains only concepts required by the verification with insignificant overhead for workflow definition (states and links). The process verification description that corresponds to a particular base process is represented by a directed graph where the vertices represent the events that approve execution of base process steps. The term "event" within the scope of verification description language implies changes in the system's "memory" (file system, e–mail, database etc.), which can be handled by agents. Arcs between events represent the order in which events are executed. The syntax and the semantics of the verification description language see [12]. Each event may have absolute time and relative time restrictions:

Each description of process verification has at least one start activity and one end activity. Likewise each description may contain the following attributes:

- A verification process instance load flag – whether the process instance is started by occurrence of the start event or by another verification process instance;
- Parameters and variables that are used in process verification events (these provide process instance identification);
- Report settings when errors occur or time limit warnings must be sent.

The implementation of the proposed approach requires programming of approximately 10000 LOC in C#, and it includes the implementation of the controller and two agents.

The runtime-solution was developed step-by-step. A simple process execution control language with synchronous event agents was implemented during the first phase. The user experience gathered using the prototype developed in the first phase showed the need for asynchronous agents. They were created in the second phase, and there was shown that the runtime verification process creates a negligible additional load for the execution of business processes.

Results

The piloting results lead to the three main conclusions – (1) the solution provides a convenient instrument for the tracking of business process execution, (2) the solution is able to detect business process execution defects, and (3) the data processing

system verification process creates a tiny extra load for the involved information systems infrastructure.

The solution provides a number of interesting possibilities, which bring us closer to the goal defined by ideas of autonomic computing:

- the verification process can be defined without modifying the base process - the base process can have more than one verification process so as to verify all of its various aspects;
- the verification process runs in parallel to a base process and does not interfere with it;
- the process verification can be added dynamically to legacy systems;
- the process verification does not depend on modeling language used for process description; it depends only on possibility of verification agents to identity events of the base process.

Likewise, some solution limitations must be taken into account: verification mechanism can detect only those base process steps which leave some modifications in the computer systems „memory" detected by agents.

2.3.2.3.2 Environment Testing

Context

The environment testing is the self–management feature for controlling and monitoring of operation (execution) environments to ascertain all involved operation environment fit to the requirements necessary for successful running of the information system. The requirements can relate to operating system, network characteristics, workstation parameters, etc. Discrepancy between the information systems requirements to external environment and the concrete execution environment may occur in several situations:

- Workstation may be incompatible in various means: insufficient memory or processor performance, inadequate network connection and other technical parameters. These are cases when execution environment verification may be done once.
- Workstation may use external resources and availability of these resources may vary during execution time. E.g., some of web services may be unavailable because of lost network connection on server downtime. Obviously in these cases execution environment should be verified continuously.
- Workstation settings do not comply with software requirements: directory structure does not contain all of required subfolders, decimal separator must be the symbol ",", data base server must be reachable, etc.
- Developers sometimes assume that software, which works in development environment, will keep working after it is deployed elsewhere, hence encoding some assumptions about the environment into the program. As a result, when the soft-ware is installed in other environment, which is different from the development environment, the software may fail or work only partially correct.

Practical use of information systems shows that many incidents and failures are not related to the functionality of the information system itself, but rather are caused by inadequate infrastructure and the execution environment. It means information systems must be accompanied by automatic means for external environmental testing.

Solution

Authors [13] propose a technology, which allows independent environment checks, performed by the software, named – "checker", in order to validate if the execution environment is suitable for normal execution (see Figure 2.3.2.2). The proposed solution implies gathering these requirements in a "software profile" to be able to validate the execution environment before program's starting. Only if the results of all checks are satisfactory, the program can be considered prepared for work at a given environment, otherwise the session is stopped, giving the user an explanation, why it is not possible to perform work.

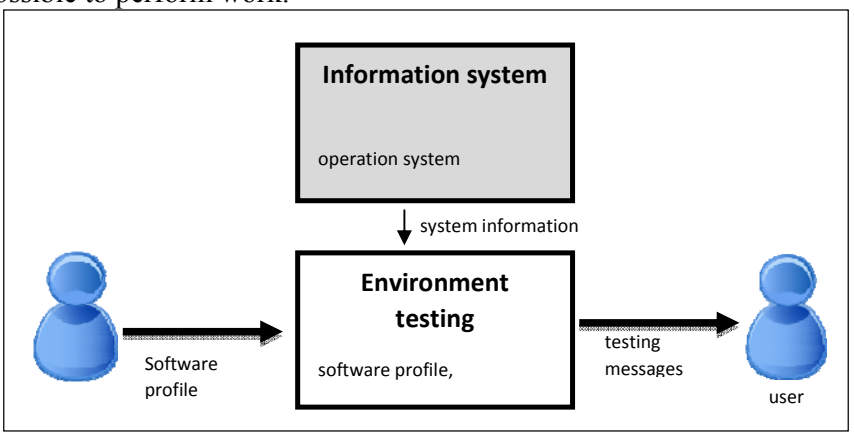


Figure 2.3.2.2. Smart technology component: Environment testing [7].

A program execution profile is a document achieved when all the requirement descriptions of software are combined together. The profile can be formalized as a separate document and supplemented to typical software deliverables such as code and documentation. The main, but not the only use of the profile is validation of execution environment during program use.

The practical environment testing task is carried out by "checker", which manages environment validation modules - drivers. Each driver is an atomic unit, which enforces validation of a single type of requirement; this is done by reading information from the environment and comparing it to reference values.

To be able to modify the set of checks to be performed without modifying the program code, information about the checks (both the algorithms and reference values) must be stored outside the code of base system – in the software profile. This concept differs from other approaches used in practice – both from the ones, which validate the environment straightaway after installation or updating, and from the others, which try to "hide" the checks in source code.

Implementation

There are four mutually independent logical layers necessary for implementation of the environment testing:

- Information system – the business software you need to run in the environment that should be checked. The mechanisms of the execution environment are independent from the business software, i.e., they may be used without changing the business software source code.
- Software profile – a list of requirements must be met to ensure appropriate business software running in an environment. The requirements can be described in a standardized XML-based language. The language can be

complicated by the many different parameter values related to the environment but unspecified during the development, for instance, the names of file folders or their location on hard drives.

- Testing modules (checkers) – programs for checking of fulfillment of certain requirements in a given environment. Each module is an atomic unit checking only one kind of requirements. The checking consists of reading the information from the environment and comparing it with the benchmark values.
- Testing coordinator – a component for monitoring of the environment testing processes modules. It analyses the software profile, runs the testing modules and processes the gained results. The implementation of this component must be flexible and open to be able to supplement it with new types of requirements and testing modules and this feature complicates the test coordinator structure significantly. The testing coordinator also must be compatible with the language which is not yet fully defined during the development of the coordinator. It can be solved by incorporating only a language syntax analyzer into the testing coordinator, therefore the semantic sense of the requirements is interpreted on the testing module level.

The practical implementation showed that development of the proposed approach requires relatively little programming resources (~4000 LOC in C#).

Results

The proposed solution since 2009 is used in a number of local information systems in Latvia. An execution environment testing was usually performed when supplying a new version of the information system. The new version was installed only after the current execution environment was checked for its ability to run the new version. Also, receiving alarms from users about the systems malfunctions there was first tested if the execution environment of the concrete workstation meets the environment requirements. In many cases, missing or wrong components of the execution environment were the reason for malfunctions.

2.3.2.4 Self-management for System Maintenance

This chapter is dedicated to the support, which may be provided to the information systems by self–properties during maintenance of these systems. After the first version of information system is deployed to the operation environment, it is will be updated or modified several times to comply with real user requirements. This leads to regular changes being introduced into the information systems.

In turn, this means that: (1) change requirements must be defined, (2) the software must be updated accordingly, (3) each of software versions must be tested (it should include regression tests and tests for new or updated functionality), (4) software should be deployed to the runtime environment and, if it requires, system data should be migrated. Furthermore usually system execution may be stopped just for rather limited period of time. The authors will discuss two self–properties introduced for support of software maintenance.

2.3.2.4.1 Business Model Incorporation

Context

Business model incorporation is a self-management feature allowing to adapt the functionality of information system without (or with minimal) coding effort, just by changing graphical business process descriptions. One of the implementation options is to apply Model Driven Development (MDD) principles for software development.

Model driven development provides a range of advantages for the system development, maintenance and execution [14]. Same of main advantages are: (1) MDD provides high level of business process abstraction thus providing less error-prone description, meaningful validation and exhaustive testing, (2) MDD bridges the gap between business and IT, (3) MDD captures domain knowledge, (4) MDD results in software being less sensitive to changes in business requirements, (5) MDD provides up-to-date documentation since the models describe the essential issues of the information system's usage. How should the system be developed to gain advantages provided by MDD?

Solution

At the beginning of information system development the business processes (see Figure 2.3.2.3) should be described as the information system will be designed to support them. A set of business process descriptions are created using DSL, and it serves as business process model. Graphical representations like diagrams can easily be understood and used by domain experts (as a rule, non-IT specialists) for the business process description. After the business process model is created, the information from the diagrams can be transferred to the database of an information system, and it is a task for IT professionals. The business process descriptions are embedded into the information system, and the engine of the information system can interpret information born from the diagrams. Embedded business processes ensure that the information system behaves according to the business process model.

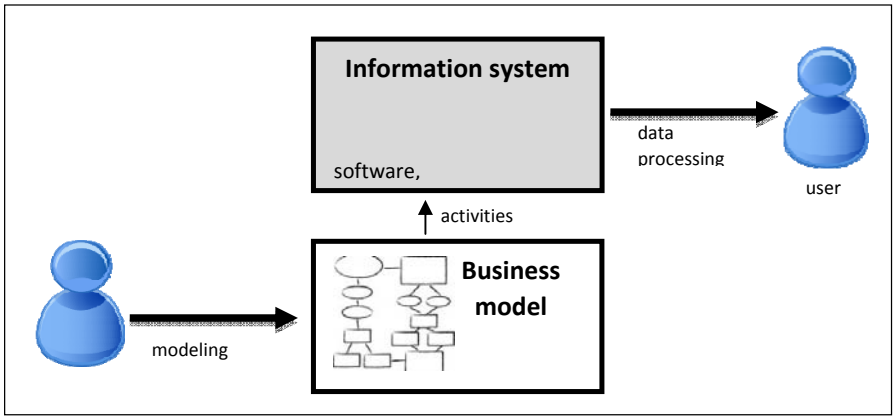


Figure 2.3.2.3. Smart technology component: Business process model incorporation [7].

However, the proposed business process incorporation approach differs radically from the model driven architecture (MDA): MDA offers a complete application generation using business process specification described in unified modeling language (UML). If the business processes are changed, the changes must be implemented in the software specification and then new software should be generated.

The proposed approach of smart technologies provides business process execution engine running according business process definition (domain specific language or

DSL is used for process description). It provides the possibility to develop flexible applications with user-friendly interfaces which can be implemented for each of systems independently. Furthermore, business processes can be updated without software modifications, and the functioning of the information system can be updated by changing of business process descriptions, without programming.

Implementation

The smart technology described in the chapter was developed and used in a number of national information systems in Latvia for many years. Therefore, it is problematic to give an accurate assessment of the resources spent for the implementation. However, it should be noted that the information systems (including the appropriate smart technology features) were implemented by teams of 4–6 IT specialists.

The implementation of the described approach was done in four consecutive, steps:

- Defining of DSL and design of graphical editor. In contradiction to the traditional software development approach, a specific, the user needs adequate DSL was used, not one, all modeling capabilities covering modeling language. The graphical editor DIMOD [6] for graphical definition of DSL syntax was designed using the tool building platform GrTP [15, 16]. A set of business process descriptions (business process model) was created using the DSL.
- Business process modeling. Graphical representations like diagrams can easily be understood and used by domain experts (as a rule, non-IT specialists) for the business process description. Both domain experts and IT professionals were involved in the business process modeling: IT professionals advised domain experts in usage of DSL and tools as well as helped to describe the most complex processes, whereas domain experts were able to model and update the existing processes independently since mastered the tools and the DSL.
- Business process transfer to the database. After the business process model is created, the information from the diagrams may be transferred to the database of an information system, and it is a task for IT professionals. An API is created which allows you to access the IS software model artifacts repository and write them into information system database in the required format.
- Built-in business process model. The business process descriptions are embedded into the information system, and the engine of the information system can interpret information born from the diagrams. Embedded business processes ensure that the information system behaves according to the business process model. It was implemented as an interpreter for execution of created business models by joining the business process diagram objects like events, actions etc. with the information systems' elements like screen forms, data base operations etc.

Unfortunately, the business process incorporation into information system can't be created as a completely independent component being able to interact with the base process without any modifications of it. This is a substantial deviation from the principles of autonomic computing described in the previous chapters.

Results

The solution described in this chapter leverages use of DSL as language for business process definition providing user – friendly method for process description.

As practice shows [17], it is possible to create a special tool for transfer of model's data to executable application relatively quickly. The API of the graphical editor can be used to access the model's repository, to gather the information and to transfer it to applications database. When business model is added to the system database, there is no more need for DSL editor repository in the system's execution environment. In turn, it provides numerous advantages for system performance and usability. Thus guaranties that the application operates according to the model developed in a graphical DSL. And the overall quality of the application – usability, reliability, security, performance etc. – is dependent on the application itself, not on the hypothetical ability of a code generator to create an application in the desired quality.

2.3.2.4.2 Self-testing

Context

Self-testing is the self-property providing the software with a feature to test itself automatically prior to operation. There is similarity between self-testing and hardware self-checking where computer tests its own readiness for operation when it is just turned on. The purpose of self-testing is to use a built-in support component for automated execution of previously accumulated tests cases not only in test environment, but also in operation (production) environment.

Solution

Self-testing contains two components:

- Test cases of system critical functionality that check the set of functions without which the software could not be used. Identification of critical functionality and designing of tests for it is a part of the requirement analysis and testing process.
- A built-in automated testing mechanism (regression testing) provides automatic execution of tests and comparison of results with benchmark values.

The automated testing mechanism provides four execution modes (see Figure 2.3.2.4): (1) Test capture mode - new test cases are captured or existing tests are edited/ deleted. (2) Self-testing mode - automated self-testing of software is done by automated execution of stored test cases. (3) Use mode - there are no testing activities – a user simply uses the system. The built-in self-testing mechanism can be used in emergency situations to find out the internal state of the system, which may facilitate the analysis of the causes and consequences of the emergency situation. (4) Demonstration mode. The demonstration mode can be used to demonstrate system's functionality. User can perform system demonstrations using use cases stored in storage files. These features are typically a part of traditional testing tools.

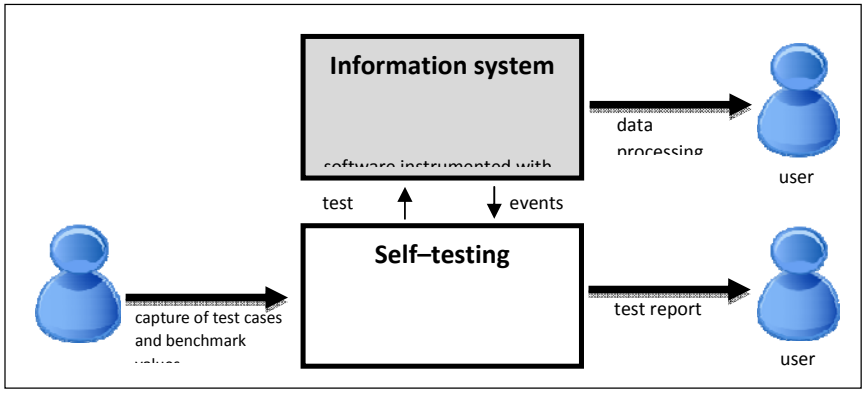


Figure 2.3.2.4. Smart technology component: Self-testing [7].

Implementation mechanism of self-testing approach [18] uses an idea and means of the software instrumentation, which is already known from the 70-ies. Testing operations are put by programmers into certain places of the source code; named - test points. Testing operations allow to track the changing values and to compare them with a benchmark. Thus it is possible to check the correctness of the information system. Unfortunately, this solution is usable only for that software whose development is in the user's influence sphere.

Implementation

The self-testing implementation includes two main steps:

- Instrumentation of software. The information system must be supplemented by test-points covering the essential ("critical") functionality of the system. The test-points are built into the information system's source code, and they ensure gathering, storing and using of test data in different self-testing modes. The instrumentation can be done only by developers of the particular information system.
- Test capturing. The data for testing of the critical functionality should be captured by using the instrumented information system in the test capture mode. Self-testing components are a part of the software to be tested. And this means that software developers, customers and users do not need to install additional testing tools to perform system testing.
- The implementation of self-testing feature required 10000 LOC in C#.
- The self-testing tool, which began in 2010, was divided into two modules:
- Self-testing module – C# .NET library (*.dll file) ensuring the interface (API) for building of test-points into the information system
- Self-testing management module – C# .NET application ensuring test capturing, creating, editing, execution, comparing with benchmark values, configuring of the tool etc.

Additionally, the source code of the particular system should be instrumented with testing activities like accumulating of test cases and executing of them. These investments are justified when the system is designed and developed for long-term use.

Like the business model incorporation feature, the self-testing also requires modifications of the base process.

Results

It should be noted that the idea of built-in support for program testing has been offered quite a while ago [19] and it has been implemented in some projects. Practice shows that the most important self-testing benefits are gained by using a single testing support for both development and maintenance phases.

2.3.2.5 Conclusions

There are four smart technology features provided and described in this paper. Authors are not discussing the research directions where no practical implementations of technologies are achieved yet. These extends variety of software self–properties and allow to achieve goals similar to autonomic computing – facilitating the use and maintenance of systems by including support components in them:

- building of smart technologies into information systems requires additional work; the proposed smart technologies have advantages when the information systems are used by many users without profound IT knowledge and the cooperation between the customer and the supplier is long-term;
- according to authors' experience smart technologies can be used even in a small to medium size IT company with 40-50 employees.

The smart technologies which are described in this paper achieve the autonomic computing initiative goals only partially. There may be still a vast variety of smart technologies which would be useful to explore and implement practical systems. For instance, these would include – data quality control, access control, performance monitoring, availability monitoring which are easy enough to implement for a small/medium size organization.

References

[1] P. Horn, Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM, (2001) http://libra.msra.cn/Publication/2764258/autonomic-computing-ibm-s-perspective-on-the-state-of-information-technology

[2] J. Kephart, D. Chess, The Vision of Autonomic Computing, IEEE Computer Magazine 36 (2003), 41-52, doi:10.1109/MC.2003.11600552003.

[3] P. Lalanda, JA. McCann, A. Diaconescu, Autonomic Computing: Principles, Design and Implementation, Springer-Verlag, London, 2013.

[4] R. Sterritt, D. Bustard, Towards an autonomic computing environment, Proceedings of 14th International Workshop on Database and Expert Systems Applications, Prague, (2003), 694 – 698.

[5] Z. Bičevska, J. Bičevskis, Smart Technologies in Software Life Cycle, Proceedings of 8th International Conference on Product-Focused Software Process Improvement (PROFES 2007), Springer-Verlag, Berlin, Heidelberg, (2007), 262-272.

[6] J. Bicevskis, Z. Bicevska, Business Process Models and Information System Usability, Procedia Computer Science 77 (2015), 72 – 79.

[7] J. Bicevskis, Z. Bicevska, I. Oditis, Self-management of information systems, Communications in Computer and Information Science 615 (2016), Springer-Verlag , 167-180.

[8] J. Bicevskis, Z. Bicevska, K. Rauhvargers, E. Diebelis, I. Oditis, J. Borzovs, A Practitioner's Approach to Achieve Autonomic Computing Goals Baltic J. Modern Computing 4 (2015), 273-296.

[9] J. Kephart, Autonomic computing: the first decade. Proceedings of 8th IEEE/ACM International Conference on Autonomic Computing (ICAC), (2011), 1-2.

[10] Roger S. Pressman, Software Engineering, The McGraw-Hill Comp., Inc., 2010.

[11] I. Oditis, J. Bicevskis, Asynchronous Runtime Verification of Business Processes, International Journal of Simulation: Systems, Science and Technology 16 (6), (2015) 6.1-6.11.

[12] I. Oditis, Runtime Verification of Business processes, PhD thesis, University of Latvia, 2016.

[13] K. Rauhvargers, J. Bicevskis, Environment Testing Enabled Software – a Step Towards Execution Context Awareness, Selected Papers from the 8th International Baltic Conference Databases and Information Systems, IOS Press, 187, (2009), 169–179.

[14] Johan Den Haan, 15 reasons why you should start using Model Driven Development, (2009), http://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/ .

[15] J. Barzdins, A. Zarins, K. Cerans, A. Kalnins, E. Rencis, L. Lace, R. Liepins, A. Sprogis GrTP: Transformation Based Graphical Tool Building Platform, (2014), http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-297/

[16] A. Sprogis, Configuration Language for Domain Specific Modeling Tools and Its Implementation, Baltic J. Modern Computing, Vol. 2, No. 2, (2014), 56-74.

[17] J. Cerina-Berzina, J. Bicevskis, G. Karnitis, Information systems development based on visual Domain Specific Language BiLingva, Lecture Notes in Computer Science, LNCS 7054 (2012), Springer-Verlag, 124-135.

[18] E. Diebelis, J. Bicevskis, Software Self-Testing. Proceedings of the 10th International Baltic Conference on Databases and Information Systems, IOS Press, 249, (2013), 249 – 262.

[19] M. Chengying, I. Yansheng, Z. Jinlong, Regression testing for component-based software via built-in test design. Proceedings of the ACM symposium on Applied computing, Seoul, (2007). 1416-1421.