e-Informatica

software engineering journal

2012 | volume 6 | issue 1

e-Informatica

e-Informatica

software engineering journal

2012 volume 6 issue 1

e-Informatica

Wrocław University of Technology

Editors

Zbigniew Huzar (*Zbigniew.Huzar@pwr.wroc.pl*)
Lech Madeyski (*Lech.Madeyski@pwr.wroc.pl, http://madeyski.e-informatyka.pl/*)

Institute of Informatics
Wrocław University of Technology, 50-370 Wrocław, Poland

e-Informatica Software Engineering Journal
www.e-informatyka.pl/wiki/e-Informatica/, DOI: 10.5277/e-informatica

Wojciech Thomas (Editorial Office Manager).

Typeset by Wojciech Myszka with the LaTeX 2ε Documentation Preparation System

Printed in the camera ready form

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.wroc.pl;
e-mail: oficwyd@pwr.wroc.pl; zamawianie.ksiazek@pwr.wroc.pl

ISSN 1897-7979

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej.

Editorial Board

Contents

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 7–26, DOI 10.5277/e-Inf120101

Time Coordination of Heterogeneous Distance Protections Using a Domain Specific Language

Marcin Kowalski*, Jan Magott*
Institute of Computer Engineering, Automatics and Robotics, Wroclaw University of Technology
marcin.kowalski@pwr.wroc.pl, jan.magott@pwr.wroc.pl

Abstract

BACKGROUND: Distance protections are widely used in protection of energy transmission lines, but their time coordination is still an important and difficult problem. Inappropriate configuration leads to a hazard event: remote circuit breaker tripping provided the local circuit breaker can be opened, which severely impairs power system operation.
OBJECTIVE: To describe a method and provide software tools to alleviate the hazard in power systems.
METHODS: A domain specific language (DSL) for representation of a transmission line with its distance protection schema, and a translation algorithm from the DSL to probabilistic fault trees with time dependencies (PFTTDs) are employed.
RESULTS: The paper presents software tools that can support power protection experts in time coordination of distance protections. The tools are based upon abstract and concrete syntax of the DSL designed specifically for the purpose of the distance protection time coordination problem. In order to render creation of power line and its protection schema models easier, a DSL-dedicated editor supporting syntax and semantic aspects of the DSL has been developed. Additionally, a translator from the DSL into PFTTD language has been implemented.
CONCLUSIONS: Power system experts are enabled to perform hazard probability assessment and sensitivity analysis.
LIMITATIONS: Translation supports two types of distance protections, which are: single-system relays with starting elements as well as multi-system relays without starting elements. For the single-system relay, there is one timer per relay. For multi-system relays, there is one timer for each of possibly many protection zones. Other types of protections, e.g. overcurrent are not considered.

1. Introduction

Distance protections are widely used in protection of energy transmission lines. The transmission line is divided into sections, whose boundaries are defined by power stations. Because of important consequences of faults like short circuits in high voltage transmission lines, a schema with primary (local) and backup (remote) distance protections is applied. Section where fault occurs is called the local section. Backup protection should disconnect the transmission line in only when the local protection with its circuit breaker has not done it beforehand. However, if the backup protection reacts, then greater part of transmission network is isolated compared with the part disconnected by the local protection. The hazard is the event: remote circuit breaker tripping provided the local circuit breaker can be opened.

For each distance protection, a set of zones, e.g. Zone 1, Zone 2, Zone 3 is defined to approximately point out fault occurring place. The greater the zone number is, the greater part of the line it covers.

Time coordination of primary and backup protections is a significant and difficult problem. If the local protection has not interrupted the line, then the remote protection has to do it as soon as possible. However, the remote one has to wait for symptoms of line disconnection made by the local one. The remote protection waiting time (also known as time delay or tripping time) for symptoms of local protection activity is usually selected according to generally accepted rules that are not adapted to particular cases. These rules are based on the worst case analysis with safety margin. According to these rules for distance protections [1], time delays for Zone 2 are selected about 350 ms, and about 800 ms for Zone 3. Therefore, the selected settings are not optimal as far as the prompt fault clearance is concerned.

Time coordination of distance protections (relays) is a part of extensive research in the field of power systems. The approaches already used are: Petri nets [2], linear programming [3,4], evolutionary algorithms [5] and multi-agent systems [6]. In these papers, overcurrent protection schemes are mainly analyzed. Distance protection cooperation with overcurrent protection is considered in papers [4, 7, 8]. These papers do not concern cooperation of distance protections.

Linear programming [3,4] and soft-computing approaches [5, 7] are used in order to solve the relay coordination problem provided tripping times (coordination intervals) are known. The main difference between the above papers and our approach is as follows. In [3–5, 7] time delays are supposed to be input data selected accordingly to generally accepted rules. Our goal is such selection of time delays for zones which is based on parameters of: transmission line, source, load, protections, and time characteristics.

Papers [9] and [10] show that time coordination of distance protections can be achieved for significantly smaller values than the recommended ones by using respectively: fault trees with time dependencies (FTTDs) and probabilistic fault trees with time dependencies (PFTTDs). In FTTD [9] time parameters are specified in a non-deterministic way by their minimal and maximal values. On the contrary, in PFTTD [10] time parameters are characterized by random variables. Models of the following time parameters have to be found: entrance (exit) times of impedance into (from) characteristics of different zones of protections for different locations of fault, circuit breaker opening time. In the present paper the following distance protection coordination process is proposed.

1. Define scopes of distance protections zones of the power line in question.
2. Specify the power subsystem and its protections in a domain specific language (DSL).
3. Translate the DSL-based model into FTTD (or PFTTD).
4. Determine time parameters from the real system or simulation experiments involving e.g. the EMTP utility [11].
5. Find the time delay for each zone of each protection using analytic bounds for FTTD [9] or by simulation for PFTTD [10].

Structures of both trees obtained in point 3. are the same. Time parameters are different only.

When time parameters are calculated using simulation program, then the following power system features have to be taken into account in the evaluation: resistance and reactance of transmission line, source impedance, types of simulated faults, fault resistance, loads, fault locations over the line, impedance characteristics for protection zones. In the present paper two distance protection types are considered, namely:

– single-system relays with starting elements,
– multi-system relays without starting elements.

For the single-system relay there is one timer. For multi-system relays the format of distance relays is the full distance scheme without starting elements, i.e., delays are timed individually for each zone. Relatively long operation time in Zone 1, the fundamental one, is a severe disadvantage of single-system relays. Therefore, in high voltage networks, and extra high voltage networks in particular, multi-system protections are applied instead.

Time coordination of multi-system distance protections using PFTTD has been considered in [10], whereas using FTTD in [9]. Time coordination of single-system protections with starting elements using FTTD has been studied in [12].

For different protection types, PFTTD of different structure is constructed. The output

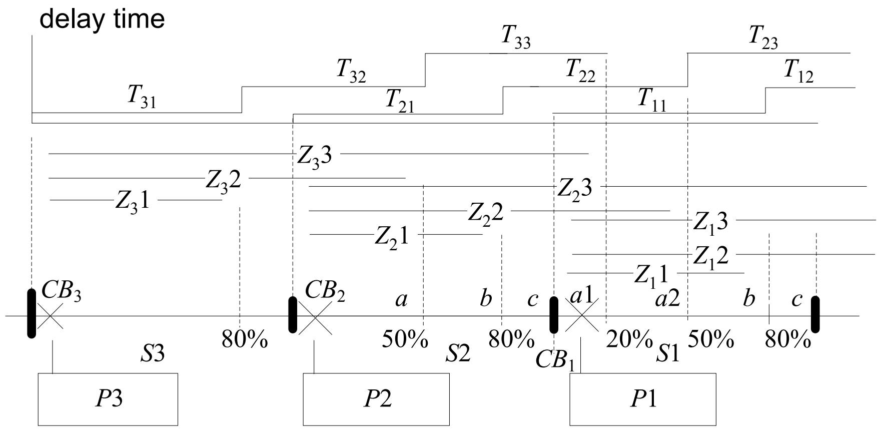

Figure 1. A three-section transmission line with protections, their zones and timing [10]

model differences arise from the aforementioned variety of distance protections supported, each of which finally could possibly disconnect the power line fragment. Since power lines with mixed distance protection types are common, the DSL to PFTTD translation should seamlessly integrate different model fragments to correctly capture the hazard. In the present paper, PFTTD models are considered. Although they have the same structure as FTTD models, they are probabilistic by nature, whereas the latter ones express time parameters in the non-deterministic way.

Although building a new domain language requires substantial development effort, the cost is quickly returned by providing work efficiency unapproachable to generic solutions. In fact, none of general purpose modeling languages may reach level of intuitiveness provided by a domain one, which has been designed specifically to support some particular domain, which in this case is time coordination of power system protections. So, by means of the DSL power system experts may build and optimize models using their own technical vocabulary without familiarizing themselves with computer engineering concepts.

Contrary to a general language, which inherently is a trade-off between requirements of various domains, taking precise semantics offered by a DSL for granted allows to build software engineering tools more effectively accomplishing their tasks, like the PFTTD generation. Had the goal been achieved with some generic solution, it would have involved not only a more verbose transformation, but also implicit restrictions of the generic language used. This is for reasons of intuitiveness and precision that domain languages win acceptance of engineers.

Structure of the paper is as follows. In Section 2, distance protection schema of a power transmission line composed of three sequential sections is outlined. In the next two sections, abstract and concrete syntax models of the DSL are presented. In Section 5, abstract syntax of PFTTD language is given. Next, PFTTD models for power lines with mixed protection types are presented. In Section 7, the transformation from DSL to PFTTD with few PFTTD models is described. The last section recapitulates the research.

2. Distance Protection

The ultimate goal of distance protection schema depicted in Fig. 1 is selectivity, i.e. only those Circuit Breakers (CBs) that are required to isolate a fault (short circuit) are opened.

A notation used in Fig. 1 is as follows: Si, where $i \in \{1, 2, 3\}$, is section,
Pi, where $i \in \{1, 2, 3\}$, is the protection placed on the left-hand side of section Si,
Z_ij, where $i \in \{1, 2, 3\}$ and $j \in \{1, 2, 3\}$, is j-th zone of protection Pi.

In the paper faults located by protections in section $S1$ are considered. According to the zones shown in Fig. 1, the $a1$, $a2$, b and c subsections can be distinguished for the $S1$ section, whereas a, b and c for $S2$.

Let us suppose that the fault is located by the $P1$ protection in the Z_11 zone, i.e. a part of $S1$ composed of subsections $a1, a2$, and b. In this case, the $P1$ protection that is close to the left-hand side of $S1$ should trip its CB_1. The $P1$ is a primary protection and triggering its breaker turns $S1$ off. It is possible, however, that a faulty operation of either $P1$ or CB_1 may occur. If the fault occurs, then the remote backup protections $P2$ and sometimes $P3$ should trip CB_2 and CB_3 respectively. In this case, however, the remote section $S2$ or even $S3$ are turned off.

The $P2$ protection operates in three zones. Zone Z_21 covers 80% of $S2$. Zone Z_22 contains $S2$ and half of $S1$. Zone Z_23 covers both sections and 20% of the section which is the right neighbor of $S1$. The $a1$ subsection is covered by the Z_33 zone of $P3$. This subsection is also covered by zones Z_11, Z_12, Z_13. Hence, this subsection is covered by six zones of three protections altogether. The c subsection, on the other hand, is covered by three zones Z_12, Z_13, and Z_23. These zones are used to roughly recognize a fault location. Finding the zone where fault has occurred is based on measurements of impedance of transmission line from the place of protection mounting to the fault location. Protections trip after T_{ij}, where $i \in \{1, 2, 3\}$ and $j \in \{1, 2, 3\}$, which is a time delay of j-th zone of protection Pi. If $P2$ recognizes a fault in the Z_2j zone, where $j \in \{1, 2, 3\}$, then after time delay T_{2j}, $P2$ sends signal to CB_2 in order to open it. Graded times of the tripping delays for zones of Pi, where $i \in \{1, 2, 3\}$, are used ($T_{i1} < T_{i2} < T_{i3}$), i.e. the greater number of the zone, the greater time delay. For the Z_i1 zone, the time delay of the start of the CB_i tripping is usually equal to zero.

Two distance protection types are considered, namely:

- single-system relays with starting elements and one timer,
- multi-system relays without starting elements and with one timer for each zone.

Let us explain protection schema assumed for single-system protection with starting element. In analysis performed in the paper, each distance protection with starting element has one timer for each protection. The following protection schema is assumed. There are impedance characteristics: Starting, Zone 1, Zone 2, and Zone 3. The Starting impedance characteristic contains Zone 3, and Zone i characteristic, where $i \in \{2, 3\}$, contains Zone $(i - 1)$ one, i.e. there is decreasing order of characteristic areas.

Let τ be time instant when the fault started. An algorithm for protection with starting elements is as follows:

```
if the starting element of the protection recognized that measured impedance entered starting impedance characteristic at instant $\tau$ then
    the timer is set to $\tau + T_1$;
end if
if impedance is in Zone 1 impedance characteristic at instant $\tau + T_1$ then
    tripping signal is sent to the $CB$ at instant $\tau + T_1$
else
    at instant $\tau + T_1$ the timer is set to $\tau + T_2$;
end if
if impedance is in Zone 2 characteristic at instant $\tau + T_2$ then
    at instant $\tau + T_2$ tripping signal is sent to the $CB$
else
    at instant $\tau + T_2$ the timer is set to $\tau + T_3$;
end if
if impedance is in Zone 3 characteristic at instant $\tau + T_3$ then
    at instant $\tau + T_3$ tripping signal is sent to the $CB$.
end if
```

The goal is to derive statistical relations between time settings for protection zones of protection and the hazard probability. To carry out investigations, information on operating time of distance relay with respect to fault occurring instant, as well as on dropout time with respect to fault clearance instant are required.

Probability distribution functions of entrance time to and exit time from all concerned zones for protections $P1$, $P2$, and $P3$, under assumption of faults located by relays in subsections $a1$, $a2$, b, c of section $S1$ and subsections a, b, c of section $S2$ have to be known (Fig. 1). They can be obtained by measurements of real system or simulation experiments using, e.g. EMTP. In the paper, these times are expressed by random variables denoted as $T_{iej|kf}$ entrance time to (exit time from) impedance characteristics of the Zone j for the protection Pi, where:

- $i \in \{1, 2, 3\}$ number of protection Pi,
- $e \in \{en, ex\}$ where en (or: ex) stands for the entrance time to (or: the exit time from) the impedance characteristics,
- $j \in \{1, 2, 3, S\}$ number of Zone j or S for starting zone of protection Pi, under assumption: the fault is located by relay (protection) in section k and its subsection f, where
- $k \in \{1, 2\}$ number of section,
- $f \in \{a, b, c, d\}$ name of subsection.

3. Abstract Syntax of the Domain Specific Language for Heterogeneous Power Line Protection Systems

The need to successfully conduct sensitivity analysis of the hazard accounts for development of a language capable of describing distance protections. Then, on the basis of models expressed in that language, PFTTD may be automatically produced and analyzed. To properly support the process, the language should consists of abstract and concrete syntax, which will be analyzed in the current and next section.

As Figure 2 indicates, these are sections and protections that comprise any power line. Sections are in turn further divided into subsections, and each of them has the *factor* attribute assigned that is determining its length in relation to the containing section. Therefore, the *PowerLine*, *Section* and *Subsection* classes along with their containment hierarchy lay the foundations of designing protections, which is performed using classes for the remaining metamodel.

Consequently, a number of protections and their circuit breakers (represented by the *Protection* and *CircuitBreaker* classes) are found in the metamodel. Since a breaker is triggered by the protection when a fault is found in one of its zones, the *cb* and *zones* associations have been added to specify these objects.

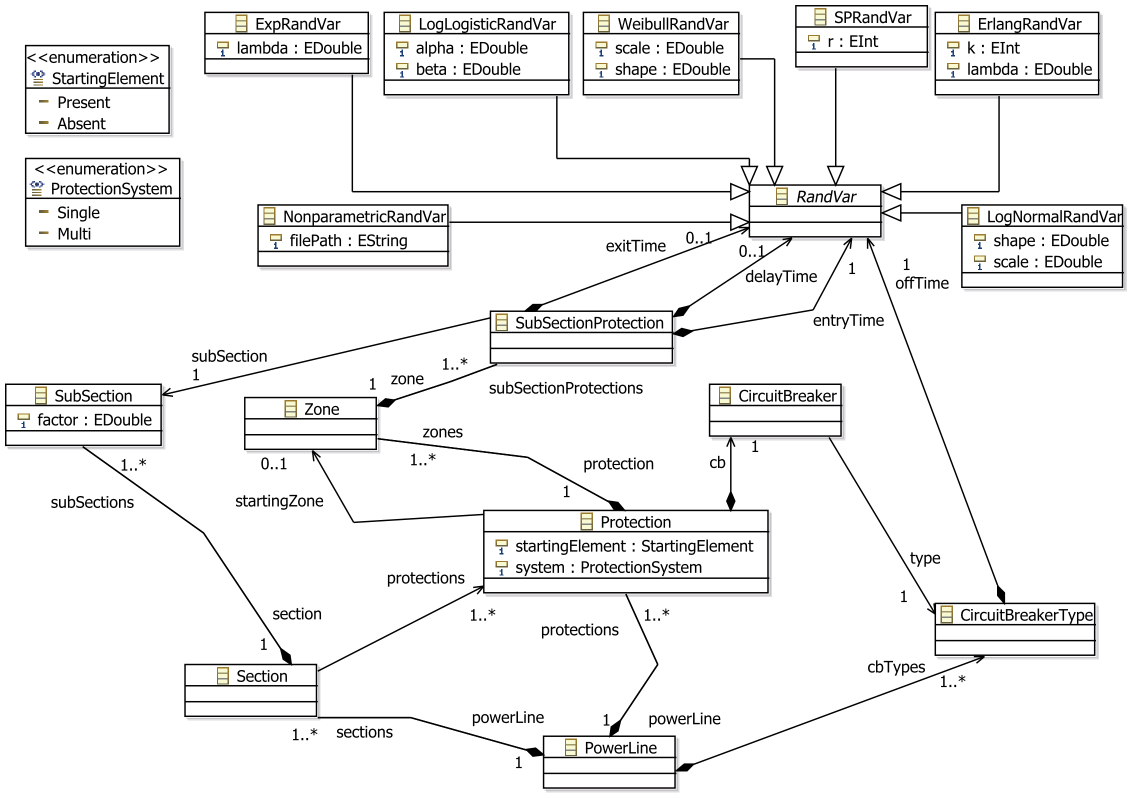

Figure 2. Abstract syntax of the domain specific language for heterogeneous power line protection systems

For the protection to trip the breaker, the timing constraints for a particular subsection must are met. This is why every zone works in the range of a few subsections specified by objects of the *SubSectionProtection* class and its *Subsection* association. A set of random variables is then necessary to perform hazard analysis. When some protection is local with respect to the subsection it protects (i.e. the subsection belongs to the same section the protection is located in), only *entryTime* and *delayTime* are needed to run the transformation and simulate the model (see Section 2). Otherwise, if the protection is remote, *exitTime* must be additionally defined to correctly capture hazard dependencies. The *RandVar* class abstracts the variety of probability distributions that random variables may conform to. Logarithmic-normal distribution (*LogNormal*) has been found to fit well when modeling entry times (denoted by the *entryTime* reference in Fig. 2) and circuit breaker opening times (*offTime*) [13].

Compared with [14], the language has been refactored to comply with requirements of the transformation supporting heterogeneous distance protections. As a result, using enumerations such as *StartingElement* or *ProtectionSystem*, power system experts may indicate presence or absence of a starting element, and whether the protection consists of one or many parts.

In order to investigate the hazard probability correctly when a starting element is on the run, a starting zone of a protection must be indicated. This is modeled as a non-containment reference, since all the zones are stored in the *zones* association.

4. Concrete Syntax of the Domain Specific Language for Heterogeneous Power Line Protection Systems

A language may be considered domain specific when domain experts are enabled to conveniently define systems they operate on in this language. Hence, the power system language abstract syntax should be accompanied with concrete syntax, from which an actual editor could be generated. This way, power system experts have to understand neither object oriented paradigms, nor internals of the language they use, which otherwise would limit intuitiveness of the approach.

Generally, there are two ways of combining abstract with concrete syntax [15]. First, the abstract syntax could be derived automatically from concrete syntax by means of fixed metamodel-level translation. However, keeping the distance protection to fault tree model transformation in mind, full control of the abstract syntax is retained in this paper by incorporating the second approach. Having defined the abstract syntax, concrete syntax is built by referencing objects and their relationships in the grammar rules. This way we managed to keep the translation robust and independent from the actual user representation of the model.

The *EMFText* [15] tool from the *Eclipse Platform* was used to design and produce concrete syntax, but feasibility study showed that the *Xtext* [16] tool would have been also useful. By turning the abstract syntax into a focal point of development, both tools could be even used simultaneously.

The grammar definition language is an extension of Backus-Naur Form consisting of rules, each starting with a rule name and ending with a semicolon (for example lines 2–3 from Listing 1). Each rule name refers to some concrete class called alike in the distance protection metamodel (Fig. 2). Moreover, rules are defined using tokens and class features (attributes or references from Fig. 2) from the metamodel. When a parser enters the rule, it creates a new object conforming to the proper class from the metamodel. It then fill values of attributes and references according to the following tokens.

For example, as line 26 in Listing 1 suggests, when a parser stumbles across the ‘Section’ token, it creates a new instance of the *Section* class. Next, the parser expects to find the *name* attribute value, followed by an opening curly brace. The *subSection* literal comes from the *Section* class drawn in Fig. 2. At that point, it notifies the parser to invoke (possibly many times due to the '+' character) the rule for *SubSection* and add newly created objects describing subsections to the aforementioned association. Once again for another rule, as line 28 indicates, the parser expects now the 'SubSection' token followed by the *name* attribute value, the 'Factor' token and the *factor* attribute value. There may be many *SubSection* objects, but once a closing curly brace is found, the Section is complete. When a similar analysis will be started from the 'PowerLine' rule to the last possible rule, the resulting hierarchy will constitute a tree spanning the metamodel. As a result, the parser will create a complete distance protection model designed by a power system expert.

To be more specific about rule definitions, when some feature is an attribute, its name is always followed by square brackets with an optional type attribute written inside the brackets. On the other hand, when some feature is a reference and its name is followed by brackets, the parser will assign to it some already existing object with the *name* attribute equal to the user typed token at that place. When some feature is a reference and is not followed by brackets, a new object will be created. Compare the *subSection* reference in the 'SubSectionProtection' rule (line 36 in Listing 1) with a $S1a1$ subsection protected by Zone Z_33 (line 28 in Listing 2). The latter listing shows also a definition of the concrete syntax for heterogeneous power line protection systems.

Listing 1. The DSL concrete syntax definition

```
1  RULES{
2   PowerLine::= "PowerLine" name[] "{" "CircuitBreakerTypes" "{" cbTypes+ "}"
3              "Sections" "{" sections+ "}" "Protections" "{" protections+ "}" "}";
4
5   CircuitBreakerType::= "CBType" name[] "OffTime" offTime;
6
7   WeibullRandVar::= "Weibull" "{" "scale" ":" scale[FLOAT]
8              "Shape" ":" shape[FLOAT] "}";
9
10  ExpRandVar::= "Exp" "{" "Lambda" ":" lambda[FLOAT] "}";
11
12  SPRandVar::= "SP" "{" "R" ":" r[INT] "}";
13
14  ErlangRandVar::= "Erlang" "{" "K" ":" k[INT]
15             "Lambda" ":" lambda[INT] "}";
16
17  LogNormalRandVar::= "LogNormal" "{" "Scale" ":" scale[FLOAT]
18             "Shape" ":" shape[FLOAT] "}";
19
20  LogLogisticRandVar::= "LogLogistic" "{" "Alpha" ":" alpha[FLOAT]
21             "Beta" ":" beta[FLOAT] "}";
22
23  NonparametricRandVar::= "Nonparametric" "{"
24             "Histogram" ":" filePath[] "}";
25
26  Section::= "Section" name[] "{" subSections+ "}";
27
28  SubSection::= "SubSection" name[] "Factor" factor[INT];
29
30  Protection::= "Protection" name[] "{" cb zones+
31             "System" system[] "StartingElement" startingElement[]
32             ("StartingZone" startingZone[])? "}" ;
33
34  Zone::= "Zone" name[] "{" subSectionProtections+ "}";
35
36  SubSectionProtection::= "SubSection" subSection[]
```

```
          "{" "EntryTime" entryTime ("ExitTime" exitTime)?
          ("DelayTime" delayTime)? "}" ;

CircuitBreaker::= "CircuitBreaker" name[] "Type" type[];
}
```

Let us consider $P1$, $P2$ and $P3$ protections of the $a1$ subsection contained in the $S1$ section in Fig. 1. Protections $P1$, $P3$ are multi-system, whereas $P2$ is one-system. A model defined in Listing 2 uses the language from Listing 1, so that the grammar parser can bind rules and eventually create the object model (an instance of Fig. 2) that will become subject of transformation described in the next section.

So, the *PowerLine* block is started first. Then come parts for types of circuit breakers (*CircuitBreakersTypes*), which are further referenced while describing protections. Power line structure (i.e. *Section* and *SubSection* rules) is defined next. Finally, the three protections are described inside the *Protections* block in the following way. For each zone controlled by the protection, a set of subsections is referenced to assign *EntryTime*, *DelayTime* and possibly *ExitTime* values. For the sake of simplicity, only parts of the $a1$ subsection are given in this example. Timing parameters are equal to 0, because EMTP simulator has not been run yet.

Listing 2. A sample model expressed in the DSL for subsection $a1$ of $S1$, where protections $P1$, $P3$ are multi-system, whereas $P2$ is one-system

```
PowerLine powerLine1 {
CircuitBreakerTypes {
   CircuitBreakerType typeA OffTime LogNormal { Scale: 0 Shape: 0 }
}
Sections {
   Section S3 {
            SubSection S3a Factor 80
            SubSection S3b Factor 20
   }
   Section S2 {
            SubSection S2a Factor 50
            SubSection S2b Factor 30
            SubSection S2c Factor 20
   }
   Section S1 {
            SubSection S1a1 Factor 20
            SubSection S1a2 Factor 30
            SubSection S1b Factor 30
            SubSection S1c Factor 20
   }
}
Protections {
   Protection P3 {
      CircuitBreaker CB3 Type typeA
      ...
      Zone Z_33{
      ...
      SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
         ExitTime Erlang { K: 0 Lambda: 0 } DelayTime SP {R: 0} }
      }
      System Multi
      StartingElement Absent
   }
```

```
    Protection P2 {
        CircuitBreaker CB2 Type typeA
        ...
        Zone Z_22{
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
            ExitTime Erlang { K: 0 Lambda: 0 } DelayTime SP {R: 0} }
        }
        Zone Z_23{
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
            ExitTime Erlang { K: 0 Lambda: 0 } DelayTime SP {R: 0} }
        }
        Zone Z_2S {
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 } }
        }
        System Single
        StartingElement Present
        StartingZone Z_2S
    }
    Protection P1 {
        CircuitBreaker CB1 Type typeA
        Zone Z11{
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
            DelayTime SP {R: 0}   }
        }
        Zone Z_12{
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
            DelayTime SP {R: 0}   }
        }
        Zone Z_13{
          ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 }
            DelayTime SP {R: 0}   }
        }
        System Multi
        StartingElement Absent
}}}
```

All in all, the $S1a1$ subsection is protected in 6 zones: Z_33, Z_22, Z_23, Z_11, Z_12 and Z_13. The Z_22 and Z_23 zones are protected by the $P2$ protection, so their work is driven by the starting element.

The second example discussed in the following sections differs from the first one in the $P3$ Protection configuration, so now it is one-system with starting element (Listing 3). First of all, a new zone Z_3S is added (lines 9–13). To indicate that the zone is starting line 16 has been added. Changes made in lines 14–15 notify the transformation to build a PFTTD for a single-system protection with a starting element.

Listing 3. The sample model with the $P3$ protection being one-system with starting element

```
1  Protection P3 {
2          CircuitBreaker CB3 Type typeA
3          ...
4          Zone Z33{
5          ...
```

```
        SubSection S1a1 { EntryTime Erlang { K: 0 Lambda: 0 }
           ExitTime Erlang { K: 0 Lambda: 0 } DelayTime SP {R: 0} }
        }
        Zone Z3S{
        ...
        SubSection S1a1 {EntryTime Erlang { K: 0 Lambda: 0 } }
        }
        System Single
        StartingElement Present
        StartingZone Z3S
}
```

One possible application of modeling the same line with two different protection schema is to evaluate impact of protection modernization on the hazard. Differences in the two hazard scenarios will be analyzed in the subsequent sections.

5. The Language of Probabilistic Fault Trees with Time Dependencies

s The PFTTDs (modeled by the *FaultTree* class in Fig. 3) are bipartite graphs with a single node denoted to be the root, which usually specifies the hazard event. An object of the *FaultTree* class contains objects of classes such as *Node* and *Edge*, which are both further specialized by the notions related to the fault tree language. These are *Event* and *Gate* which constitute the two types of bipartite graph nodes. Objects of those classes are connected through *EventOutput*- and *GateOutput*- edges. The first ones start with events and end with gates (the one drawn between E7 and G1 in Fig. 5 for example), whereas the latter ones start with gates and end with events (the one drawn between G1 and E1).

The second part of the language depicted in Fig. 4 refines the PFTTD gates, which can be causal or generalization. Names of gates consist of two parts. The first letter denotes a kind of gate (C for causal and G for generalization) and the remaining part defines preconditions for a gate to start its output event. The AND, OR and NOT types relate to the classical logic, whereas PAND generates output when both input event occur and the left one occurred as first. Delay gates, which are denoted by an hour-glass symbol, operate like CXOR gates with a single input by introducing random variable-based time delay between input and output events.

6. Discussion of a Generated Probabilistic Fault Trees with Time Dependencies

Although details of the DSL to PFTTD translation will be explained in the next section on the grounds of some specific model cases, in this section discussion of the output model (Fig. 5 from transformation of the $a1$ subsection of $S1$ section from Fig. 1) will follow.

Probabilistic Fault Tree with Time Dependencies (PFTTD) analysis starts with identifying hazards, which is the event: remote circuit breaker tripping provided the local circuit breaker can be opened and faults are located by relays in the $a1$ subsection. For each hazard, a PFTTD is created.

Let us make the following assumption regarding fault occurrence.

Assumption 1: At most one fault can occur during analysis interval, and once it happens, it is permanent.

Types of protections are as follows:

$P1$, $P3$ – multi-system without starting elements, and with one timer per each zone,

$P2$ – single-system with starting elements, and with one timer per one protection.

According to requirements specification, if there is a fault in $S1$, and additionally $P1$ and the local breaker CB_1 are operational, then only $S1$ should be disconnected. The hazard is event

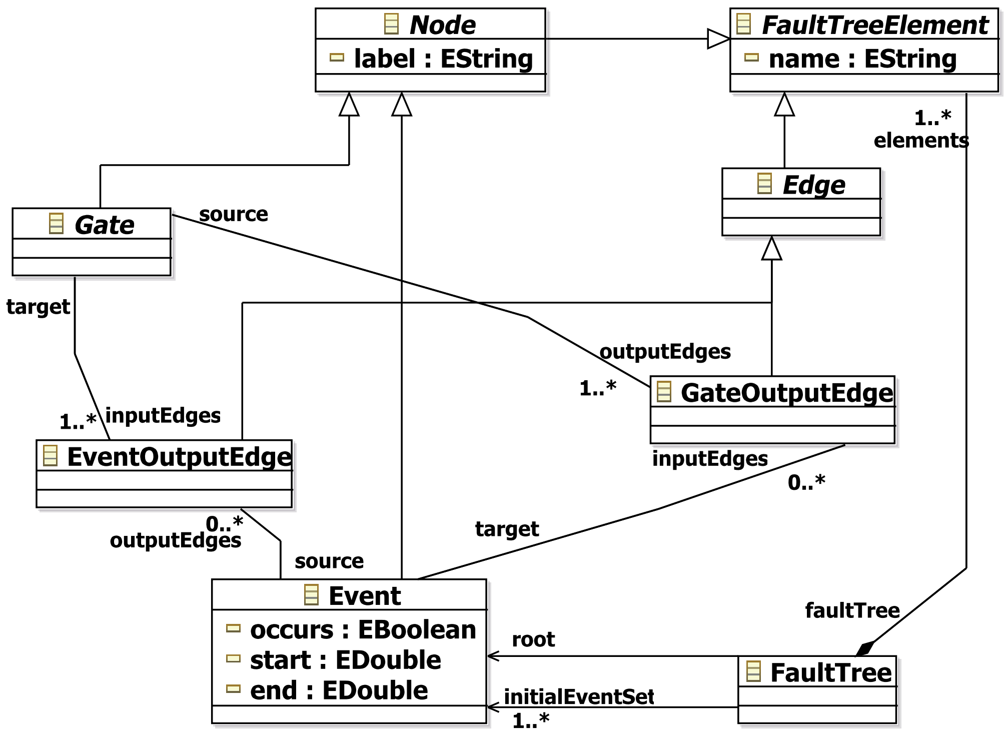

Figure 3. The core part of the PFTTD language

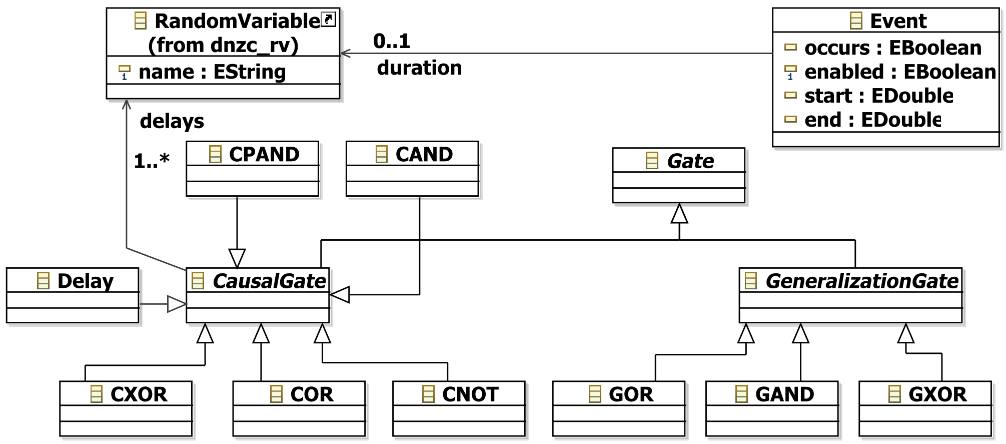

Figure 4. Causal and generalization gates of the PFTTD language

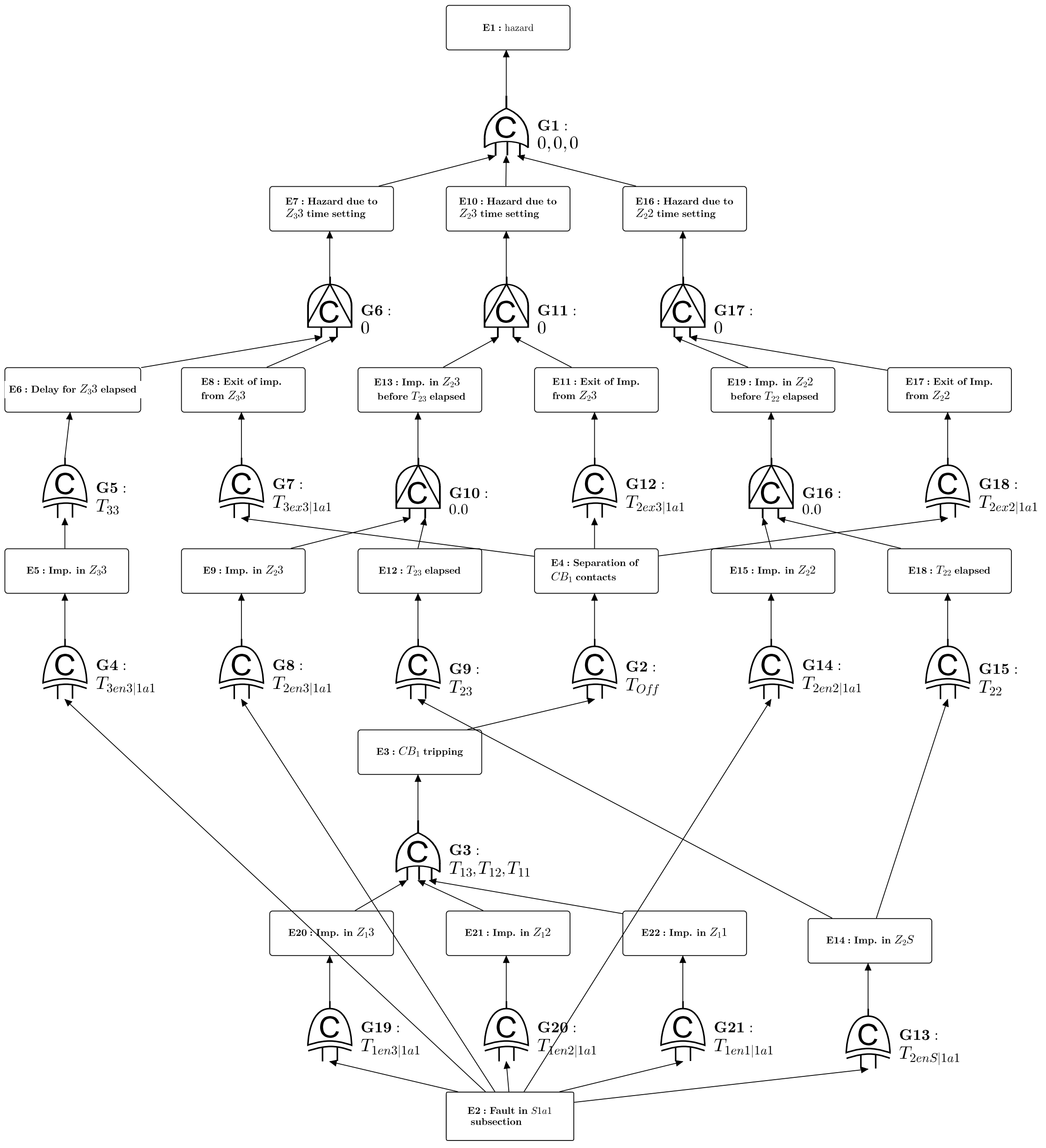

Figure 5. A PFTTD for the first example: fault located by relays in the $a1$ subsection of $S1$ section, which is guarded by $P1$, $P3$ – multi-system protections without starting elements, and $P2$ – single-system with starting element

E1: remote circuit breaker (CB_2 or CB_3) tripping provided the local CB_1 can be opened. Hence, the hazard happens when excessively large part of the power network is isolated. The PFTTD for this hazard and fault located by relays in section $S1$ and its subsection $a1$ generated by the translator is illustrated in Fig. 5. The tree contains parts that are similar to the ones for multi-system protection [10] and fragments similar to those for single-system protections [12].

Event E1 occurs if at least one delay time (T_{33}, T_{23}, T_{22}) is too small, i.e. at least one of the E7, E10 or E16 events has occurred. Hence, the hazard occurs, if at least one remote protection time delay for zones Z_22 or Z_23 of protection $P2$, or zone Z_33 of $P3$ has been set incorrectly. Time delay between start instant of event E7, E10 or E16 and start instant of event E1 is equal to 0. Hence, delays for all inputs of gate G7 are equal to 0.

Let τ(Eis) denote instant when event Ei has started. Some events may stop immediately, where others last longer. For G6, if event E7 has occurred then event E6 had occurred not later than the E8 event, i.e. the start of E6 had occurred not later than the start of E8, so $\tau(\text{E}6s)<\tau(\text{E}8s)$. In this case, $P3$ trips its CB_3. Turning off process will not be stopped, and CB_3 will be opened. In this case, tripping of CB_3 will be prior to detection of fault clearance symptoms. Hence, the $P3$ relay has reacted too early, what caused that the hazard event E7 has occurred. In order to avoid the hazard, the following condition: $\tau(\text{E}8s)<\tau(\text{E}6s)$ has to be satisfied. Additionally, if CB_1 of $P1$ is not opened, then the $P3$ will not detect that CB_1 is opened. Hence, the E8 event does not occur, and consequently neither does the hazard. On the other hand, let us suppose that the $P3$ had observed that the CB_1 is opened (event E8) before time delay T_{33} has passed, so $P3$ will not turn off its CB_3. Hence, event E7 does not occur (the hazard does not occur).

Let us consider a sub-tree with the E7 event as the root. In this sub-tree, the part with the E6 event concerns the $P3$ protection and its zone Z_33, while the right sub-tree with E8 concerns $P1$. In sub-tree with event E16 as the root, the part with the E19 event is related to $P2$ and its zone Z_22, while part with E17 is associated with opening activity of CB_1 by $P1$.

If there is a fault in subsection $a1$ of $S1$ then impedance seen by $P3$ can be inside operating characteristics of Z_33. In this case impedance seen by $P2$ can be inside characteristics of Z_22 or Z_23. Hence, tripping of CB_3 can be started after time delay T_{33} from instant the impedance entered characteristic of Z_33. Time T_{33} is the time from start instant of event E5 till start instant of event E6. The tripping of CB_2 can be started after time delays T_{22}, T_{23}, respectively, from instant the impedance entered impedance characteristic of the Starting zone Z_2S of $P2$, provided the impedance remains in characteristics of Z_22, Z_23. These times are given by real numbers, and are represented by delays of the G9, G15 gates associated with E14 event.

Impedance trajectory measured by $P2$ enters characteristics of Z_2S after time $T_{2enS|1a1}$ relatively to instant τ being start of the fault. This time is the parameter of the G13 delay gate. If the fault in subsection $a1$ of $S1$ occurred at time instant τ and it still lasts then the impedance seen by $P3$ enters characteristics of Z_33 at instant $\tau+T_{3en3|1a1}$. Time $T_{3en3|1a1}$ is the delay of delay gate G4. Times $T_{2en2|1a1}$, $T_{2en3|1a1}$ respectively, associated with $P2$ are the time delays of the gates G8, G14.

Trajectory of the impedance seen by $P3$ exits from characteristics of Z_33 after time $T_{3ex3|1a1}$ since separation of CB_1 contacts. This time is the delay of gate G7. CB_1 tripping lasts the time given by random variable T_{Off} (Fig. 5). Hence, delay of gate G2 (time between start instant of event E3 and start instant of event E4) is equal to T_{Off}.

If there is a fault in $a1$ of $S1$ then impedance seen by $P1$ can be inside operating characteristics of Z_11, Z_12 or Z_13. Hence, CB_1 tripping can be started either immediately, or after time T_{12}, or after T_{13}, relatively to the instant when the impedance seen by $P1$ entered characteristic for Z_11, Z_12 or Z_13 respectively. Therefore, three times, namely $T_{11}=0$, T_{12} or T_{13} are delays of the G3 COR gate. They are all equal to time intervals between start instants of input events E20, E21, E22 of this gate and start instant of the E3 output event. Entry times of impedance into characteristics for zones Z_11, Z_12 and Z_13 of $P1$ are random variables $T_{1en1|1a1}$, $T_{1en2|1a1}$ and $T_{1en3|1a1}$ respectively. These random variables characterize delays of the G19, G20, G21 delay gates.

Detailed explanation of fault trees with time dependencies for single-system protection with starting elements can be found in [12], while explanation of probabilistic fault trees with time dependencies for multi-system protection is given in [10].

7. A Power Line Protection DSL to PFTTD Transformation

According to the procedure proposed in the Introduction, the third step of hazard analysis is to translate a domain model into PFTTD. The translation in question is discussed below.

Generation of output models takes place in three phases. The first one produces a fault tree skeleton and is protection independent. In the second phase, the skeleton is supplemented with elements generated from local protections guarding a subsection. Finally, parts related to remote protections are created. The procedures described below (or mappings in the Query View Transformation parlance [17]) constitute the second and third phase and are run iteratively for each protection guarding the subsection.

The skeleton of every model consists of the E1, E2, E3 and E4 events along with G1, G2 and G3 gates (see Fig. 5). These parts are common among any produced models and describe the fault, hazard and breaker tripping performed by the local protection. Depending upon type and placement of subsection's protections, modifications will be applied to the resulting fault tree.

When a local protection without a starting element is employed, time to trigger the breaker depends on zone entry time as well as on a deliberately introduced delay (possibly 0). Hence, the missing part between the E2 event and G3 gate is composed of: a delay gate, event and delay variable assigned to the G3 gate. For example, G19, E20 and T_{13} in Fig. 5.

Presence of a starting element, however, substitutes that model fragment with the one presented in Fig. 6. The tripping process is altered in such a way that impedance must enter characteristic of a protecting zone (E23) before (E22), so it enters the starting zone (E25) and awaits the delay of the G22 gate. The translation process of this PFTTD model fragment will be discussed on the basis of code snippet presented in Listing 4.

As the name of the *localProtectionWithSe* mapping suggests (line 10), it produces a set of PFTTD elements related to the operation of a local protection driven by a starting element. They all fit between the line fault event (E2) and the COR gate (G3), hence the mapping's arguments. The class name after the *query* or *mapping* keywords (e.g. line 1 or 10) indicates the execution context, i.e. a class of the *self* local variable specific to a particular mapping invocation. Moreover, in this case the mapping can be executed only when the *isLocal* and *hasSe* queries both return the logical truth (lines 10–11), which is when the mapping is applicable. Otherwise, some other mappings (not listed) are executed.

Listing 4. A QVT mapping for translation of a single system protection with a starting element

```
1 query SubSectionProtection::hasSe() : Boolean {
2  return self.zone.protection.startingElement = StartingElement::Present;
3 }
4
5 query SubSectionProtection::isLocal() : Boolean {
6  return self.subSection.localProtection() = self.zone.protection;
7 }
8
9
10 mapping SubSectionProtection::localProtectionWithSe(in lineFault: Event,
11    inout cbTripping: COR) when {self.hasSe() and self.isLocal()}{
12
13  var delayElapsed:= new Event(stroke("T_{" +
14    self.zone.protection.protectionNo() +
15    self.zone.zoneNo() + "}") + latexText(" elapsed"));
16  var impdInZone:= new Event(latexText("Imp. in ") + stroke(self.zone.name));
17
18  var impInZoneBeforeDelay:= new Event(latexText("Imp. in ") + stroke(self.zone.
19    name) + newLine() + latexText(" before ") + stroke("T_{" + self.zone.
20    protection.protectionNo() + self.zone.zoneNo() + "}") + latexText(" elapsed"));
21
22  var entry:= new Delay();
23  var delay:= new Delay();
24  var order:= new CPAND();
25
26  new GateOutputEdge(delay, delayElapsed);
27  new GateOutputEdge(order, impInZoneBeforeDelay);
28  new EventOutputEdge(impInZoneBeforeDelay, cbTripping);
29
30  new EventOutputEdge(lineFault, entry);
31  new GateOutputEdge(entry, impdInZone);
32
33  new EventOutputEdge(impdInZone, order);
34  new EventOutputEdge(delayElapsed, order);
35
36  new EventOutputEdge(self.subSection.map
37    StartingElement(self.zone.protection, lineFault), delay);
38
39  cbTripping.delays += new RandomVariable("0");
40  entry.delays:= Sequence {self.map toEntryTime()};
41  delay.delays += new RandomVariable(stroke("T_{" + self.zone.protection.
42    protectionNo() + self.zone.zoneNo()+"}"));
43 }
44
45 mapping SubSection::StartingElement(in protection: Protection,
46    in lineFault: Event): Event {
47
48 init {
49   result := new Event(latexText("Imp. in ") + stroke("Z_" +
50     protection.protectionNo() + "S"));
51  }
52  var faultInZoneS := new Delay();
53  faultInZoneS.delays := Sequence{protection.startingZone.
54    subSectionProtections->select(e |e.subSection = self)->
55    first().map toSeEntryTime()};
56
57  new EventOutputEdge(lineFault, faultInZoneS);
58  new GateOutputEdge(faultInZoneS, result);
59 }
```

The first query (lines 1–3) traverses the distance protection model to find out whether the starting element has been indicated by a user. The second query checks if the protection that the subsection protection (*self*) belongs to (right-hand operand in line 9) is the same as the local protection of the subsection being analyzed (left-hand side operand).

Three events are created in lines 13–20. For example, the E22 event, *delayElapsed* was generated using the *delayElapsed* variable. Similarly, *impInZone* is E23 and *impInZoneBeforeDelay* is E24. Arguments of the event constructor (not shown) are expressions wrapped in some Latex tags using the *stroke* and *latexText* queries (not shown).

Next, the G21, G22 and G23 gates are created, in that order, in lines 22–24. Then, starting from line 26 up to 37 are all the aforementioned elements are bound together by the *GateOutputEdge* and *EventOutputEdge* objects. In each case, the first constructor argument is the source element, and the second is target. Finally, the starting element part, whose creation will be discussed shortly, is connected with the gate represented by *delay* variable (lines 36–37).

The final part of the mapping is the assignment of correct random variables. When the

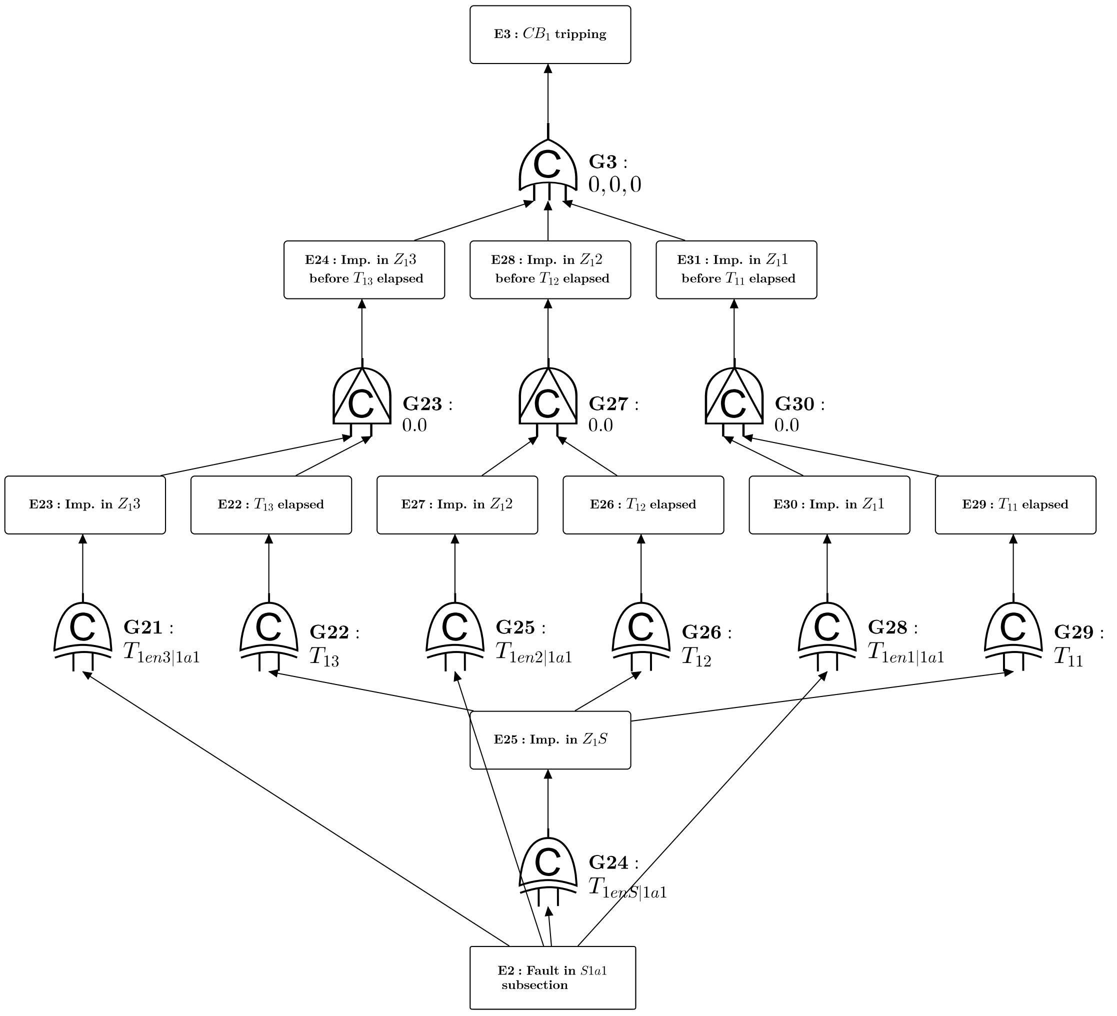

Figure 6. The local circuit breaker tripping by the single-system protection with starting element

starting element is present, all random variables of the COR gate should be equal 0 (line 39). Contrary, entry time of the subsection protection should be transferred from the DSL model, which is performed by the *toEntryTime* mapping (not shown). When it comes to delay values (lines 41–42), only a new random variable is created without assigning its distribution. It should be filled manually by a domain expert while analyzing the resulting PFTTD.

Special attention should be paid to the *StartingElement* mapping invoked in lines 36–37, whose code is listed in lines 45–59. It is responsible of creating the G24 and E25 elements, which specify how a starting element operates. This mapping has returned, for example, the E25 event initialized in lines 48–51. Next, the Delay gate is created and initialized. When assigning its timing parameters (lines 53–55), the respective starting element description is searched over the collection of entry times to the starting zone. Newly created elements are finally connected in lines 57 and 58.

Independently of the local protection type, the main goal of the second phase is to refine E3, being the circuit breaker tripping event, which is referred to in the last step.

The hazard results from competitions between the local breaker tripping and each of remote protections, which may interrupt too soon some excessively large line area before the local protection will disconnect the local breaker. The third phase binds E4 with E1 in a way dependent on presence of a starting element.

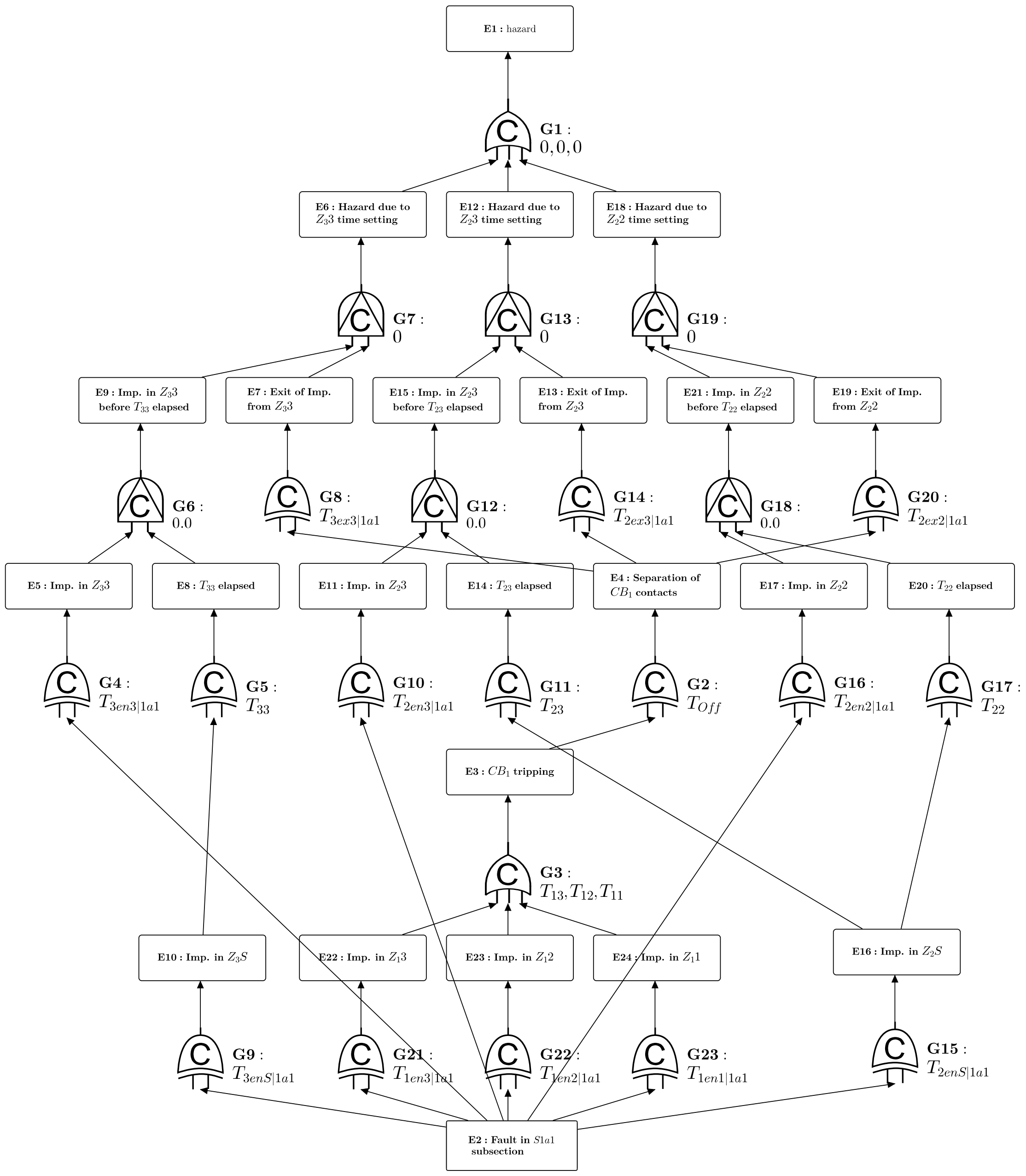

Figure 7. A PFTTD for the $a1$ subsection in $S1$ section, which is guarded by: $P1$ – multi-system protection without starting elements, $P2$, $P3$ – single-system with starting element

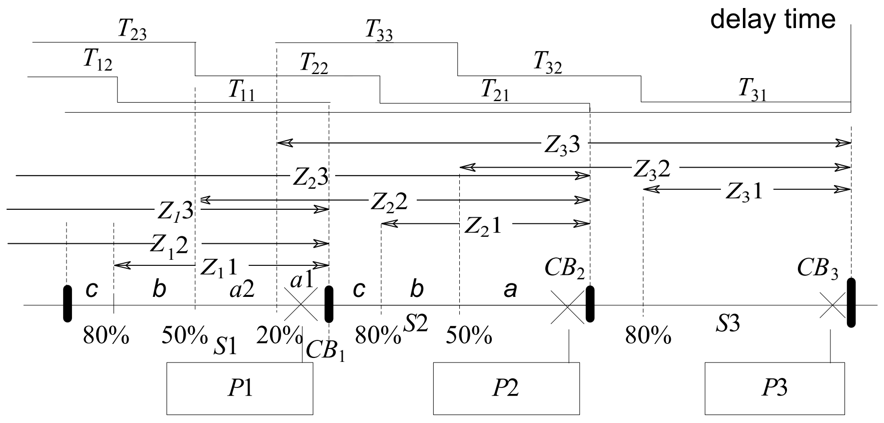

Figure 8. A three-section transmission line with protections placed at the sections' ends

When there is no starting element, the structure such as E5, E6, E7 and E8 is constructed similarly to Fig. 5. However, as described by the end of Section 4, in the second example, protection $P3$ was turned into a single-system with a starting element. The transformation correctly captured that change and produced a new model fragment with E5, E6, E7, E8 and E9 as shown in Fig. 7.

Furthermore, depending on the real system configuration, protections may be placed either at the beginning of the section (as in Fig. 1) or at its end (as in Fig. 8). Sometimes protections are located at both sides of a section. By analyzing the hazard probability for the model with reverted zones, power system experts may decide on redesigning the real system. Figure 9 depicts the results of transforming the system defined in Fig. 8 for the $a2$ subsection in the $S3$ section. Originally there were no remote protections for this subsection, so no redundant safety measures were taken in the case of fault. According to new protection schema, however, two zones of remote protection $P2$ additionally guard the area and hazard estimation is possible owing to model drawn in Fig. 9.

8. Final Remarks

The new domain specific language for representation of a transmission line with its distance protection schema accompanied by the translator to probabilistic fault trees with time dependencies were designed, implemented and verified. The verification was run for different protection schemes. Two types of distance protections: single-system relays with starting elements as well as multi-system relays without starting elements were analyzed. Additionally, protections were located at both ends of the section.

Since structures of FTTD models for distance protection are the same as those conforming to the PFTTD metamodel, the translator might be also employed to the non-deterministic models.

In the DSL, protection schema for line with transformer can be expressed according to paper [9], and the translator can be used for sections with transformer too.

The resulting fault tree proves its usefulness in:

- communication of the risk analysis outcome between power system experts,
- simulation,
- hazard sensitivity analysis by means of altering protection schema,
- hazard sensitivity analysis due to upgrade of protection type.

The last step of the distance protection coordination process proposed in Introduction, i.e. model simulation, requires respective tooling. Software capable of handling PFTTDs generated from the DSL was described in [18].

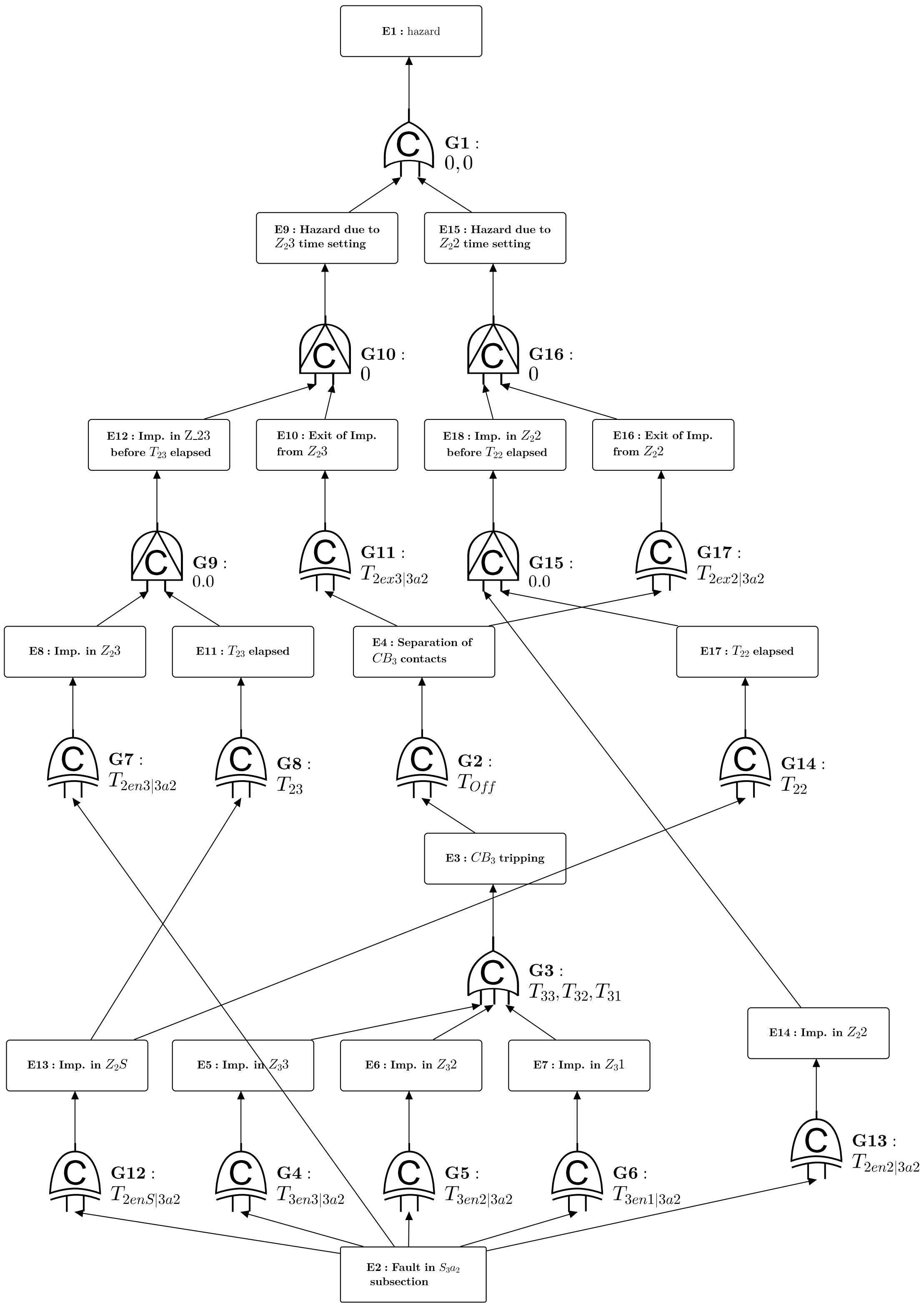

Figure 9. A PFTTD for the $a2$ subsection in $S3$ section when protections are placed at the end of sections: $P1$ – single-system with a starting element, $P2$ – single-system with a starting element, $P3$ – multi-system

Acknowledgment

The authors wish to express sincere gratitude to Prof. Leszek Trybus for his in-depth review.

References

[1] *Network Protection and Automation Guide*, ALSTOM T and D, 2002, version 1.

[2] L. Jenkins and H. P. Khincha, “Deterministic and stochastic Petri Net models of protection schemes,” *IEEE Transaction on Power Delivery*, Vol. 7, No. 1, July 1992, pp. 84–90.

[3] L. G. Perez and A. J. Urdaneta, “Optimal coordination of directional overcurrent relays considering definite time back-up relaying,” *IEEE Transaction on Power Delivery*, Vol. 14, No. 4, October 1999, pp. 1276–1284.

[4] ——, “Optimal computations of distance relay second zone timing in a mixed protection scheme with directional overcurrent relays,” *IEEE Transaction on Power Delivery*, Vol. 6, No. 3, July 2001, pp. 385–388.

[5] C. W. So and K. K. Li, “Time coordination method for power system protection by evolutionary algorithm,” *IEEE Transactions on Industry Applications*, Vol. 36, No. 5, 2000, pp. 1235–1240.

[6] J. H. Chen, S. H. Chen, and Y. M. Yang, “Multi-agent based protection relay system for transmission network,” in *Proc. Second International Conference on Machine Learning and Cybernetics*, J. Smith, Ed. IEEE Press, November 2003, pp. 2251–2254.

[7] H. A. Abyaneh, S. Kamangar, F. Razavi, and R. M. Chabanloo, “A new genetic algorithm method for optimal coordination of overcurrent relays in a mixed protection scheme with distance relays,” in *43rd International Universities Power Engineering Conference UPEC 2008*, 2008, pp. 1–5.

[8] S. Jamali and M. Pourtandorost, “New approach to coordination of distance relay zone-2 with overcurrent protection using linear programming methods,” in *39th International Universities Power Engineering Conference*, 2004, pp. 827–831.

[9] M. Łukowicz, J. Magott, and P. Skrobanek, “Selection of minimal tripping times for distance protection using fault trees with time dependencies,” *Electric Power Systems Research*, Vol. 81, 2011, pp. 1556–1571.

[10] T. Babczyński, M. Łukowicz, and J. Magott, “Time coordination of distance protections using probabilistic fault trees with time dependencies,” *IEEE Transaction on Power Delivery*, Vol. 25, No. 3, July 2010, pp. 1402–1409.

[11] *Electromagnetic Transient Program EMTP*, Leuven Center, 1987.

[12] J. Magott and M. Łukowicz, *Reliability, risk and safety : back to the future*. London: Taylor, Francis, 2010, ch. Time coordination of primary and back-up distance protections with starting elements in electrical power systems, pp. 514–521.

[13] J. Zając, “Short circuit duration time in 110 kV electrical power networks in the light of statistical-probabilistic research (in Polish),” Ph.D. dissertation, Poznań University of Technology, Electrical Faculty, 2007.

[14] M. Kowalski and J. Magott, *Methods of development and application of real time systems (in Polish)*. Gdańsk: Pomorskie Wydawnictwo Naukowo-Techniczne, 2010, ch. A domain specific language for time coordination of distance protections in power systems, pp. 199–208.

[15] F. Heidenreich, J. Johannes, S. Karol, M. Seifert, and C. Wende, “Derivation and refinement of textual syntax for models,” in *Model Driven Architecture – Foundations and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2009, Vol. 5562, pp. 114–129.

[16] *Xtext Reference Documentation*, Itemis, www.eclipse.org/text/documentation.

[17] *MOF Query/Views/Transformations Specification*, Object Management Group, http://www.omg.org/spec/QVT/, Dec. 2009, version 1.1.

[18] M. Kowalski, *Software engineering in the process of information system integration (in Polish)*. Gdańsk: Pomorskie Wydawnictwo Naukowo-Techniczne, 2010, ch. A model driven tool for design and simulation of Probabilistic Faults with Time Dependencies, pp. 225–234.

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 27–37, DOI 10.5277/e-Inf120102

Supporting Applications Development and Operation Using IT Security and Audit Measures

Katalin Szenes[*]
[*]*Faculty John von Neumann, University Obuda*
szenes.katalin@nik.uni-obuda.hu

Abstract

The market success of the enterprises depends on the ability to support their business processes. This involves the requirement of a seamless, well-ordered operation of the whole company. Operation is greatly affected by the quality of its IT support. The information should be available, handled confidentially, preserving its integrity, have to be processed in a reliable, efficient, effective way, in compliance with the requirements of supervisory authorities. Extending the scope of these information criteria to criteria determining operations quality and adding two business-level requirements to them makes possible to find preventive, detective and corrective, originally information security control measures, raised to the level of operational quality, that support the market success of the institutions.

1. A Method Based on IT Security and Audit for Supporting Corporate Governance

The goal is to facilitate the use of the originally information security and information systems audit ideas and tools in the area of corporate governance. In the followings the criteria characterizing such a corporate IT functioning, that is able to contribute to the compliance to a widely accepted set of requirements, are extended to the area of corporate operations. To operations belong every area, that supports business. Corporate finance, controlling, human resource management, and the like all belong here. Without them no business could operate.

In order to improve IT processes ISACA (Information Systems Audit and Control Association) was probably the first organization, that collected all these criteria. If we extend the scope of the measures by which some of these criteria can be fulfilled, to other business-supporting areas, then these criteria can also be raised to the level op corporate operations. This possibility of discussing the problems in a greater arena then before, will be illustrated here on a special application, on the service-oriented architectures.

2. Business Goals and Information Security

Seamless operation is one of the basic factors of the corporate market success. Improvement of operational quality, and compliance to the requirements coming from government and other authorities are vital. IT applications are non-separably interwoven into the everyday and even into the strategic level activities of every company. Thus to the fulfillment of the strategic business goals, computer applications have to support the – often contradictory – aspects of operation and compliance.

An efficient IT of a professionally operating firm follows best practice methods. Good examples are the methodologies of such prominent organizations as ISACA, or the ISO standards. ISACA and ISO both require the availability, confidentiality and integrity of corporate data. In its methodology ISACA appends to these the requirements of effective, efficient, reliable processing, and compliance to the authorities' prescriptions [1].

To this set two business-level requirements are to be added, according to my experience. One is appropriate functionality of every IT system, meaning, that the business-, or any kind of end-users are asked to confirm, that the systems help them reaching their strategic and business goals. The other is keeping order in every aspect of the company life.

The functionality requirement, that means actually involving the end-users into the development process, can directly be translated to a lower level goal to be set to IT: the deliveries of every milestone of the systems development lifecycle should be approved by the responsible organizational unit.

One of the necessary conditions of maintaining order in a company is to do so in every department. Doing so, involves specifically, among other requirements, up-to-date documentation, and configuration & change management of the whole IT architecture. An important factor of order is, of course, planning the other supporting, and what is more important, the business activities, too, before acting [2].

If we extend to operations our seven criteria originally set by ISACA as a best practice for IT, and add to them IT systems functionality, and order in every corporate activity, then we get a list of conditions usable in the improvement of operational quality.

Applying these conditions to different targets taken from the company life we get a generalization of the notion of IT "control objective". Information systems auditors and security professionals refer to best practice management objectives set to IT activities as "control objectives". Let us call these as "IT control objectives", and extend this notion to such best practice management objectives, that the operational areas have to achieve. This way we get the "operational control objective" and we will call this as "control objective" in the followings. (This will not arise disturbance, as IT control objective is a special case of operational control objective.)

To reflect the intentions of the top management in devising (operational) control objectives this term was extended to mean any kind of goals that can be derived from the corporate strategy [2]. Actually the scope of the original control objective is extended from IT to the broader arena of corporate operations.

Using this terminology, the above considerations mean, in other words, that lower level operational control objectives help the company to achieve one of its most important, high level control objective: raising the level of company operation so that it supports corporate success as well as possible.

The weights of these often contradictory, even if perhaps not completely independent requirements are always to be balanced, of course, according to the requirements of the given situations. The actual weights to be assigned have to depend on the business requirements. To find an optimal balance, that suits to the business goals the best way, risk management methodologies can be used [3].

Methods taken from the knowledge base of information security and audit, will be shown here to be able to help a lot in satisfying these control objectives, in order to illustrate how information security and audit are able to serve directly corporate strategy through the improvement of the quality of operation. It should be noted, that for managing risks the same or similar information security & audit ideas and tools could be exploited, as the ones presented here [3].

Having chosen our control objectives, the next step is to find measures, so-called "control measures", that can help reaching them. If the control measures are categorized, then to find the appropriate one will be easier. As the goal is operational excellence, the proposed categories are based on the three basic pillars of corporate operations [2]:
– organization,
– regulational system,
– technics.

The control measures will be presented here together with the control objective they help achieving, or the problem they help solving. We must not forget, that all these control objectives – at least in a balanced way – are necessary to supporting the business, but they are not enough. Without them the business users will not have a clear and exact picture on the present state of their tasks, but reaching these control objectives is not enough, will not totally transform the company. The other value of information security and audit ideas will be just the control measures. All of them, by themselves, will help the company towards a better organized way of living. However, it should be noted, that the complex process of identifying those strategic goals that help best the company to market success can not be spared. There are systems analysis methods for this purpose, that we have no room to discuss here.

To illustrate how these measures support the business goals, such a practical example was chosen, as an extension of former information security considerations [4], that belongs to an emerging area of application development: the service oriented architecture, SOA.

3. Implementation of Business Intelligence Using Service Oriented Architectures

This already for years fashionable architecture can be considered as a set of business processes performing business functions. The processes are implemented by so-called services, programs usually written in Java. These processes are "loosely coupled" to each other. This relation means either direct communication or a kind of orchestration – cooperation, that provides for the scheduling of process execution. For implementing this loose coupling different, complex ready-made products are available.

The processes are known to each other or to the outer world only through their communications so newly built and old, legacy applications can be packed together into this architecture and then the individual applications will be reached through this common platform.

According to ISACA researchers choosing this type of architecture positively affects the return of IT portfolio because of its promising cost/efficiency of solution delivery [5]. The SOA system is stated to reduce systems complexity, implementation and maintenance costs, and to enhance test effectivity at the same time.

This architecture is not an off-the-self product, but rather an approach to problem solving that supports a new way of thinking which is very useful in building such complex structures as e.g. enterprise portals that collect information from various background information sources.

SOA is on the way to contribute to the alignment of IT to the business processes by the means of a transparent and integrated application, service and process landscape. The technical processes can directly be derived from the business process models by the means of an integrated enterprise-wide meta repository of the available components. This is a repository of such services from which a complete IT projection of a business model can be built.

On the level of the reference model, however, SOA is a collection of distributed capabilities, that are created by people or by organizations and are needed by somebody to solve a problem. SOA is said to be "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [6].

The idea originated in the middle nineties with the ambitious goal to share the business logic of an enterprise between its different computer applications and to facilitate a kind of multi-threaded execution of these applications, even if some of them operate on the same database.

The step that surely leads beyond the limits of the enterprise architecture integration is the spreading of the applications systems components all over the internet. The 21th century SOA consists also of loosely coupled, in a way individual parts, but these parts are now so-called web services, such services, that can be made available, or, in other words, can be invoked, either from the corporate intranet or from the internet and they use these two media for communication.

Not only the system components can reside on different nodes of the world wide web, but the users of the system, too. Nowadays when employees have to access the corporate applications practically from anywhere, the availability of an application system from the outskirts of the company is a very important point. Thus the service orientation turned into web service orientation both from the viewpoint of its build and that of its way of using.

4. SOA Main Features

The architecture of these new systems presents a unified surface to their user but their services might

– reside on different network nodes of the corporate network or even those of the internet,
– are diversified and run on different hardware, software – operating systems and database platforms,
– are developed by different vendors, using different methodologies.

To satisfy availability, confidentiality and integrity of the information, to process it effectively, efficiently, reliably, taking the requirements on compliance, order and functionality into consideration, is not at all trivial, with these complex applications, having parts spreading over the internet. To make matters even more difficult, when we pack new and old applications together as if they were individual services but called from a central entry point, this diversification of users and services, and the possibility of incorporation of the legacy systems into a brand new applications architecture at the same time, together with the loose coupling of so different components, by communication and scheduling, arouse new problems, preserving – due to the components – the traditional difficulties just as well.

These latter come from the legacy systems, that their users do not want to part with. These systems are independent islands in the enterprise information system. Their services are completely satisfactory to their users who are accustomed to them. Unfortunately, they frequently rely on obsolete databases, and are written in out-of date programming languages. Their documentation, if it ever existed, has been lost long ago. However, these drawbacks are the problem of the IT personnel while the end-users insist on preserving these systems. A solution is the wrapping of a legacy system in such a way as if it were a black box affecting the state of its environment only by its input/output. To find ways to implement this wrapping became a subject of interest already in the end of the last century [7].

Some experts consider the service oriented concept as a successor or an improvement of the idea of enterprise architecture integration. This integration wanted to provide for a common framework connecting every application of an enterprise [8]. This connection usually provides for a common entry point for the applications, too, so it can serve as a front-end system. One of the tasks of a front-end is to authenticate the users of the package of application systems behind it, then, according to their roles, the users are authorized. This authorization determines, how they will be able to use the systems of this package. As a next step, according to their authorized access rights the front-end offers the users the services of the systems. For the end-users this functionality looks like a menu system. This is the first thing they meet having authenticated themselves to the operating sytem of their computer.

The front-end systems of such heterogenous and giant corporate applications as the accounting systems of financial institutions are built quite frequently according to this structure. The users in the bank connect to the application portfolio – customer accounting systems, treasury, brokers' systems – through a menu system. At this menu the users have to be authenticated and then authorized to perform different functions – to invoke menu points – according to their work roles, that is defined by their job descriptions. Thus this is a point where confidential access of the employees to the set of applications behind the menu can be enforced. Besides confidentiality the fulfillment of other requirements can also be illustrated on front-end systems and service-oriented architectures.

The vulnerabilities and other issues concerning any architecture can be grouped in different ways. A possible classification of the SOA vulnerabilities can be, that to one group belong those, that are caused by the SOA architecture itself, for example by the difficulties involved in planning such a system, and the other group can be formed from those, that the operation of the SOA systems yields. The lack of considerate planning and/or that of the perfunctory implementation can undermine, of course, either the structure or the operation.

It will be marked here, which control objective, by what kind of control measure can be fulfilled, and to which – organizational, regulational, or technical – pillar does that control measure belong to. Our example will set mostly IT-related goals, but the measures will be on operational level, classified according to these three proposed pillars of operation.

5. Architectural Issues

A first step in finding the weak points of this architecture might be to explore, what is that mentioned loose coupling, that keeps its parts together. The parts are the so-called web services, that implement such activities, usually one service by one activity, that the business processes invoke. The business processes serve the business goals directly, the services perform usually one step that helps achieving the business goals. In order to make cooperation possible between these parts a kind of communication is necessary.

The services do not call each other in a subroutine-calling way. They communicate, using mostly XML, and the other connection between them is a kind of organization of their cooperation, the so-called orchestration of the web services, or rather the orchestration of their quite various functionalities. The orchestration and the communication together provides for the loose coupling, that makes a SOA from the components. The orchestration is to

– implement the business logic that connects the business processes to each other, and
– contribute to the building of such an application system from these various web services that is able to serve the current business goals set by the end-user.

Practically the execution of a SOA structure is based on an integrated handling of resources, together with such an administration of these resources that yields the satisfactory provisioning of the resources. This requires:

– choosing the appropriate web service, invoking it, and managing the passing of the control from one service or administration function to the other according to the needs of the user,
– the management of the communication between users, system, auxiliary components.

To achieve the high-level goals of the orchestration described above different solutions are available. The so-called enterprise service bus (ESB) was one of the most popular among them. It collected references of the available services into a kind of registry from where they could be chosen in case of need [9]. These ESBs became collections of such business service capabilities that could be invoked.

Compliance of the Application System to Business' and Authorities' Requirements. Served by: Regulational Pillar Type Control Measure – Involves Administering Order.

Authorities here mean those government and other institutions that have the authority to demand compliance to their requirements.

Such a compliance can only be based to have regulations on preliminary planning and on the continuous documentation of the satisfaction of both the users' and the compliance requirements at the different phases of development and throughout the whole life-cycle of the application. Planning before doing anything, and preparing documentation are both preventive control measures, they might parry quite a lot of problems.

Availability. Served by: Regulational Pillar Type Control Measures; Change Management, Configuration Management.

As it was already mentioned, documentation, change management and configuration management are vital information security measures both in developing and in operating any kind of applications [2, 10].

Changes of the application development projects, either shifting the goals, or adding/revoking resources, or any other event should be rigorously managed. This involves, among others, the documentation of the change requests, that of the permissions of the competent officers before the change is actually committed, etc. Otherwise sooner, than later the application becomes inconsistent with the information available about it. This results in chaos, in incompatibility of the running environment with the actual needs, in impossibility of administering any further corrections or impossibility of tuning the system to the business users' requirements, as nobody will know where is the point to be corrected, etc.

If we turn for advice to the COBIT methodology of ISACA, the description of the "Major Upgrades to Existing Systems" process of the domain Acquire and Implement says that if we carry out a major change to our application then we should "follow a similar development process as that used for the development of new systems" [1].

Without configuration management the current state and the whereabouts of the IT facilities will be unknown, and then the maintenance and other tasks to be executed can not be allocated. The instructions concerning documentation, change and, of course, release management should be part of the regulational system of every institution.

Availability of the Application. Served by: Technical and Regulational Pillar Type Control Measures.

The availability of the SOA architecture can be unpredictable when incompatibilities between the parts of the applications are realized too late. It can happen that the repositories used do not seem to be able to handle the services of other suppliers and then these services will be unreachable.

Even if the application doesn't always require the presence of every service at the same time, every service should be available. Some consider the asynchronous, publisher/subscriber way of communication to be a flexible possibility [11]. In this case the services are invoked in an event-driven way.

The ideas behind the SOA methodology and the building tools are changing, every day new issues arise. So many enthusiastic professionals began dealing with this new promising land that to track every direction would be hopeless. The variety of building blocks is very rich and these blocks are even developed according to different quality standards, if any are used at all.

In order to ensure the interoperability of these security solutions the Liberty Alliance [12] was founded by the suppliers. If a product complies with the requirement set of the generally accepted version of the Security Assertions Mark-up Language (SAML) then it is compatible with the products of other suppliers.

The other organization, where the security of communicating web services are widely discussed is XML Protocol Working Group of the W3C – World Wide Web Consortium [13].

Presently the service oriented architectures operate mostly in a client-server way so the services have to be present, too. For implementing the details of interaction a widely accepted standard should be chosen and then ordered to be followed. These are technical, and regulational control measures at the same time. Having them executed, we will have compliant products that are able to cooperate with each other.

6. Operational Issues

Here we follow an imaginary operation of a front-end system based on SOA technology. Looking for weaknesses in the execution of a front-end system, when we find one, then we look for appropriate control measures. Our palette of vulnerabilites to be cured will be here far from complete, of course, books could be written on this subject.

Preserving Confidentiality at the End/Abort of Service Execution – Locating Point of Termination. Served by: Technical Pillar Type Control Measure.

One vulnerable point when these program systems begin operating is surely common. This is the flexible way of calling this set of services by a simple click that incurs all of the threats that usually endanger a remote connection. This connection can be invoked either from the more or less defended corporate network or remotely from the outside and the point of termination can be anywhere in the internet.

It would be desirable, if the requestor of the service could decide, when and how is the requested service to be terminated. Without predefined plans and painstaking programming this is not possible. Should anything go wrong otherwise, then, besides doing something unplanned, the service might go astray, carrying along some valuable business/personal data or logic.

Balancing Between Control Objectives: Availability Versus Confidentiality Balancing between the requirements is very important as the SOA applications usually support rich and complex functionality.

In this case, against unathorized outsiders the sensitive data could be encrypted but then availability might suffer as encryption/decryption will decrease performance. Business requirements are to decide, which opportunity is to be chosen.

First Confidentiality Issue in Operation – Provisioning for the Users' Access Rights. Served by: Organizational, Regulational, and Technical Pillar Type Control Measures.

Some years ago the so-called middlewares began replacing the ESBs. These are able to extend the business support capability by a facility of access right management [14].

This means that here we can use an important organizational control measure: the tasks of the organizational units and those of the employees are to be clearly defined in the job descriptions in such a way that the duties are appropriately separated.

This organizational control measure should be written into a rulebook. Having put then this rule into effect we have built a regulational control measure.

If the access rights are assigned in such a way, that everybody is permitted to reach those and only those data that are necessary to perform their duties, then the application built on this middleware will support the confidentiality requirement.

Segregation or separation of duties is considered to be appropriate according to the best professional practice, if it satisfies at least the two most important basic requirements [1]. The first is, that there is no employee with too big power in modifying the corporate data, e.g. nobody has development and operation responsibility at the same time. The second is that there is no employee who has to supervise himself/herself. This way the business secrets and other, e.g. for privacy reasons sensitive data will have a chance to be confidentially handled.

Second Confidentiality Issue in Operation Identification and Then Complete Authentication of the User Who is Asking an Entry Permission. Served by: Technical and Regulational Pillar Type Control Measures.

When the application system is based on a SOA architecture then the user authentication process is even more important with all the internet connections involved. To one customer different companies might provide for web services that cooperate with each other and the user has to be known to every service.

As first step of the authentication, the user has to be identified by the means of a user identifier that is valid according to the records kept by the operating system. If this identity is accepted, then he/she has to be authenticated in order to ascertain if this identifier really belongs to the user who has given it. After the successful authentication the user will be authorized to go forward, according to the settings belonging to this user identifier.

The threats entail the necessity of a really rigorous identification – authentication – authorization process that is advised to be extended towards federated identity management if more than one companies are involved in the provisioning of the web services comprising the SOA. Federated is the identity management if it supports a check throughout different companies by the means of strong authentication tools.

Federated identity management raises the level of the user authentication from that of the individual web services to a level of a synergy of these services. Serving the end-user these services have to communicate and have to pass the control to each other. The user has to be identified by all of the services that have anything to do in fulfilling his/her needs. Federated identity provides for a single sign on facility at the entry point of the SOA. This is the control point where the access rights of the user are to be set according to his/her role in the company. Having the user authenticated the services can communicate with each other on behalf of the user.

Federated identity management is described by OASIS [6], a non-profit organization, that develops standards and specifications to support e-business.

The strong user authentication requires more information pertaining to the user than a simple user password. Biometrical tools can be used to provide some personal characteristics. Tokens, smart cards, and the like devices, that are based on possessing something can also be used to enhance security.

These technical control measures, of course, have to be described by regulations. The processes of authentication and authorization are to be defined. The requirements of a successful authentication have to be clearly stated.

Third Confidentiality Issue in Operation: Authorization of the Authenticated User. Served by: Technical, Regulational and Organizational Pillar Type Control Measures.

After the successful authentication the computer system has to authorize the user according to his/her organizational roles in the corporate. Having clicked onto the entry point of the SOA the user encounters a menu. This again is a possibility to administer defensive measures. After the successful identification and authentication of the user, the access right management system should authorize him/her exactly according to his/her role in the organization.

Some of the bases of authorization were already mentioned. Summarizing the most important ones:

- regulations concerning the enrolment, and
- termination of the employees,
- their job description,
- the process of asking for and then,
- confirming permissions,
- the revocation of the permissions.

The facilities of the system offered usually as menu points are to be just those options that he/she is permitted to use. The range can be properly set only if an exact job description is available which:

- is aligned to the organizational structure,
- defines the tasks to be performed,
- takes the segregation of duties principle into consideration.

The users should have access to

- those and only to those systems and within them,
- to those systems functionalities, and
- data, that are necessary in order to perform the duties given in their job description.

Devising organizational diagrams, defining the tasks of the organizational units and the employees, their job descriptions, in such a way, that their duties are properly segregated belong to the organizational type of the control measures. All of these are preconditions of a well-planned authorization process.

Fourth Confidentiality Issue in Operation: Defending Important Business Data. Served by: Technical, Organizational and Regulational Pillar Type Control Measures.

The data of the information systems are resources, necessary to perform that functionality of the SOA system which satisfies the user's request.

To illegal program modification more internal knowledge and skills are needed then to attack data directly. According to its function the data can be:

- applications data – these relate to the business of the institution,
- management data – needed to the administration of the information systems.

To the management data belong:

- the databases containing the user identification, authentication and authorization information – e.g. password tables, some of these might be embedded into different access control systems,

– the data supporting the operations of the SOA and that of the IT infrastructure.

Examples for management data are: the data that are necessary to the scheduling of the web services or to operating the network devices or managing intrusion detection systems. Lots of other data set are vital to a well-functioning operations support. To the user databases belong those that are needed for the entry to the corporate network. This user information, unfortunately, can not be stored in one central collection but is usually spread all over the corporate network. These data describe, among others:

– PC users who are permitted to connect to the corporate network – this is usually an operating system table,
– the users of the different applications – stored usually in the applications themselves,
– the users of the different devices and facilities, etc.

The applications user groups are normally part of the group of PC users. Those firms that are strong enough financially to melt these groups into a single – sign – on user community have a chance to strive for a central user administration.

All of these data have to be defended against stealing. Defense involves hiding the users' identification and authentication data. We can not detail here, how to choose a safe solution, but we mention that one of them is encryption. Encrypted data can, of course, be decrypted, so such algorithms have to be chosen that cost/effectively defend the data.

In Windows-based networks Microsoft Active Directory is rather frequently used for storing the authentication informations of users' groups. To its advantages belong the more or less ready availability of the systems engineers who are Windows experts. Their cost is usually less than that of a skilled Linux/Unix professional where the openness of these operating systems requires considerable inside knowledge besides management & maintenance experience. This wider requirement set might make the company quite dependant on these employees.

One of the most important drawbacks of the Active Directory is the lack of a facility to maintain the history of the access rights of the users from the point of time they were employed till the termination of their employment. Active Directory shows always the present state only.

The risk of this lack of control can be mitigated sometimes on application level. Enterprise integrated system SAP is a positive example. It is able to track its users' access right history throughout their life in the company from entering till termination. Without such an application the organized and regulated tracking and archiving of the changes in the access rights might be a feasible solution. The respective tasks should, of course, be allocated, thus this is both regulational and organizational control measure.

As far as the access control on database level is concerned a considerable improvement of some of the database systems seems to be necessary in the near future. In some cases there are ready solutions available.

If there is no such control of every field of a record that the system could log the employee who modified something then the suppliers of these database system and the customers have to find other solutions. Confidential data can be locked from trivial access, e.g. the data can be put in a kind of vault. Some of the database systems facilitate fine-tuning of access rights according to the roles in the organizational units and to the sensitivity classes defined for the data [14].

There is a possibility to control field level access in such a way that the database administrators do not have full access rights full time but they get the access right necessary to complete their work from a security administrator just for the time interval when they need it. Field level access might improve the data processing performance of the applications and facilitates the fine tuning of access rights at the same time.

It must be noted, that the control measures defending the data should usually be supplemented by application level control procedures. These latter depend partly on the specific features of the given database system [15]. If these control measures are still not enough then come the organizational level control measures that usually define rules concerning the personal behaviour of the employees. These measures should be explicitly described in procedural rulebooks.

Fifth Confidentiality Issue in Operation: Screening Users' Legal Activities; Tracking the Unauthorized Access Attempts. Served by: Technical, Organizational, and Regulational Pillar Type Control Measures.

Should an auditor want to ascertain if the data are safe or not then he /she might want to compare the actual activities to the documented permissions. Another important question is: what do the users do with their legal permissions?

The logging of the users' activities and those of the data base administrators is not only a detective control measure but might help these employees to prove their innocence in case of security incidents. Of course, the logs provide for authentic proofs only if they can not be tampered with from that point of time when they were created. The solution is to sign digitally the log records, and to stamp them with the point of time of their creation, and doing so immediately at creation time. Digital signature means – roughly speaking – the creation of a so-called hash code. This code is composed from the bytes of the record to be preserved intact in its original form.

To log the activities the logging facility has to be set on – if the target system has such a facility at all. But all of these efforts are worthless if the log records are not managed, that is they are not archived, handled, etc., and if the collection of logs of different systems is not analysed, taking into consideration, of course, their relations to each other. All of the log records should be introduced into a central log management system. These are called as SIEM – Security Information and Event Management Systems.

Besides the users' information other equally important data are the log records of the various IT infrastructural elements. Infrastructural elements are the different hardware, operating system, databases, or even computer applications, the network devices, the defense systems and other special facilities such as those that participate in providing for the internet service: the proxy servers, the web servers and the like. Some of these devices are able to give signs about their current, or sometimes even about their future state in the form of log records. (Some of them can "complain" that it will go wrong within a short time.) The appropriate use of this information should be included in the maintenance regulations.

All of the duties enumerated above have to be assigned to somebody – this is an organizational measure, and the measures are to be described and regulated, these are regulational measures. Making all this possible by the means of handling the log records is a set of technical measures.

7. On Other Issues to be Handled

Here we can only call the attention to some also very important SOA issues that are also to be taken into consideration. All the problems can not even be listed here, that are known to the professional community, and to which different departments of the company have to answer by detective, corrective or preventive control measures. Here we restrict ourselves to giving only a sample in the followings.

Managing the resources needed by the services to perform their business function arises the question of availability again. These resources are mostly data in databases but to the resources belong, in a broader sense, all of those infrastructural elements that support somehow the operation of the web services. There is a lot of type of them, that all have their identifiable role in the SOA infrastructure, just as in the case of any other program system architecture. The infrastructural elements are subjected to the usual threats characterized by the nature of the given element, thus the elements one-by-one, and the whole system too, has to be defended, as a complex structure. This defense involves physical and logical measures alike. To the latter belong numerous maintenance tasks for improving the availability, integrity, confidentiality of the information and the resources.

Besides the supporting architecture, problems can arise from using SOA, too. The communication of its components with each other, and with the user, the cooperation of the parts by a kind of deadlock-free scheduling have to be managed [4, 16].

The communication protocols used for these communications can be attacked. The lack of planning, or omitting systems analysis phase yield vulnerabilities in the production systems. Unfortunately, this organizational and regulational defect of the support of corporate strategy is quite frequent.

8. Conclusions

Informatin security and audit methodology used for many years successfully for IT Governance is being extended to the support of corporate governance [2]. As an illustration of this research ways of at least partially solving some formerly discussed problems arisen by the complexity of service oriented architectures [4] were discussed above.

To prevent, detect or correct such problems operational level organizational, regulational and IT technical level measures were suggested. Information criteria belonging to the toolkit of information security and audit were extended to the level of evaluation of corporate operations.

References

[1] "Cobit 4.1 framework, management guidelines, maturity models," 2007.

[2] K. Szenes, "IT GRC versus? enterprise GRC but: IT GRC is a basis of strategic governance," in *EuroCACS 2010 – Conf. on Computer Audit, Control and Security.* Budapest, Hungary: ISACA, Rolling Meadows, Illinois, USA, March 2010.

[3] ——, "Building a corporate risk management methodology and practice," in *EuroCACS 2002 – Conf. for IS Audit, Control and Security.* Budapest, Hungary: ISACA, Rolling Meadows, Illinois, USA, March 2002.

[4] ——, "On the intelligent and secure scheduling of web services in service oriented architectures – soas," in *Procds. of the 7th International Symposium of Hungarian Researchers on Computational Intelligence*, Budapest, Hungary, November 2006, pp. 473–482.

[5] P. Williams, J. Spangenberg, and S. Kovaleva, "It and shareholder return: Creating value in the shareholder industry," *Information Systems Control Journal*, Vol. 4, 2007, pp. 39–42.

[6] Oasis – organization for the advancement of structured information standards. http://www.oasis-open.org.

[7] M. Yoshioka, T. Sodo, A. Yoshikawa, and K. Sakata, "Legacy system integration technology for legacy application utilization from distributed object environment," *Hitachi Review*, Vol. 47, No. 6, 1998, pp. 284–290.

[8] S. Bennett, S. McRobb, and R. Farmer, *Object-Oriented Systems Analysis and Design Using UML.* McGraw-Hill Education, 2006, ch. System Architecture, pp. 338–370.

[9] C. Nelson, J. Miller, W. Farrell, R. Reinitz, and K. Brown, "Implementing a service – oriented architecture version 1.0," September 2005.

[10] D. Melancon, "Security controls that work," *IS Control Journal*, Vol. 4, 2007.

[11] J. van Hoof. Client server versus publish subscribe. http://soa-eda.blogspot.com/2010/09/clientserver-versus-publishsubscribe.html.

[12] Liberty alliance. http://www.projectliberty.org.

[13] W3C – world wide web consortium. http://www.w3.org.

[14] C. Everett, "Oracle spreads into the middle," *Infosecurity Today*, Vol. 3, No. 4, Jul. 2006, pp. 34–36. [Online]. http://www.sciencedirect.com/science/article/pii/S1742684706704359

[15] J. H. White, "Important but often dismissed: Internal control in a microsoft access database," *Information Systems Control Journal*, Vol. 6, 2006, pp. 30–34.

[16] D. Perelman-Hal, "Ajax and record locking," *Dr. Dobb's Journal*, October 2006, pp. 45–51.

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 39–45, DOI 10.5277/e-Inf120103

Middleware Architecture for the Interconnection of Distributed and Parallel Systems

Ovidiu Gherman*, Stefan Gheorghe Pentiuc*
**Electrical Engineering and Computer Sciences Faculty, "Stefan cel Mare" University*
`ovidiug@usv.ro, pentiuc@eed.usv.ro`

Abstract

Grid computing is a fast evolving technology, bringing more computing power to its users. Two main directions are observable: creating dedicated supercomputers for scientific and commercial tasks and creating distributed commodity-based systems. The first ones are usually much expensive, but have the advantage of performance, better control and uniformity in platforms. The second one is more affordable but lacks in flexibility and easy maintenance. The computing necessities that often require supplementary computing power for certain time periods are better satisfied by interconnecting available resources than buying new, expensive ones. But interconnecting platforms – sometimes radically different – can be a difficult task. The proliferation of hybrid parallel computing systems can be even more complicated because it puts in contact systems with various operating flows at the parallelism level. In this frame, the present article proposes a new middleware architecture that can connect multiple parallel or distributed resources, of different types, allowing unitary resource utilization and reservation for the user's jobs. The new architecture is described functionally and structurally.

1. Introduction

This article presents a middleware architecture that can connect a diversity of grid and parallel systems so that the users can run seamless jobs on multiple platforms, of different architectures, the compiling and executing part being sourced to the suitable resources assigned by a centralized (or decentralized) manager (broker). The main scope is interconnecting clusters of computers with different architectures and platforms (for example as MPI and hybrid Cell-based systems) so that the access will be transparent and uniform to the user, without the problems arising from the use of multiple computing clusters [1].

Certain acronyms will be used in the next pages: MPI – Message Passing Interface (communication protocol and specification set regarding communications between processes in parallel computers; popular software implementation are OpenMPI, MPICH1/2 and LAM-MPI), Cell and PowerXCell8i microprocessors manufactured by an alliance led by IBM (International Business Machines), hybrid systems (computing platforms that seamlessly integrates multiple CPU architectures), HPC – high performance computing, SSH/SFTP (Secure Shell and Secure File Transfer Protocol – widely used communication protocols in computers data transfers) SPF (single point of failure, showing a critical component that can disrupt or stop the entire system from normal work) and Beowulf systems (parallel computer systems built from inexpensive PCs).

Using a middleware (that offers a set of services) has the advantage of being easier to implement and can be installed on top of the already existent equipment (both hardware and software). Being message-oriented, can be easily extended and allows dynamic reconfiguration of the platform (for example when new resources are brought in the grid or crash and are removed from the available pool). The middleware can be extended so that new facilities can be attached. Once a new resource is added to the resource availability pool and properly set up (depending on the particular configuration of the software environment on that resource) it can be chosen to run certain jobs (in a generic way or a specific one – if it has a desired particularity).

2. Proposed Middleware Architecture

2.1. Introduction

The necessity of using large parallel and distributed computing systems required the creation of computing clusters – both homogeneous and heterogeneous regarding the distribution of hardware and software components. The most difficult step is to control and manage them efficiently and satisfactory for the user – goals that sometimes are opposite.

The proposed architecture wants to be a "glue" between parallel computing platforms that use the grid infrastructure already on the local resources (such as parallel clusters or Beowulf systems using MPI platforms – OpenMPI, MPICH1/2, LAM-MPI – and/or hybrid platforms like MPI on global level and PowerXCell locally [1]) and to extend the functionality of these systems, in the same time connecting them in an unitary fashion, transparently to the user. This middleware is built on top of the local operating systems in order to benefit from the security and optimization layout of the OS and parallel middleware, allowing fast development and optimization.

The middleware architecture provides an easy access method to numerous resources with different specifications, in which a set of services are made available to the users, services that allow to define a set of attributes required for the execution of the programs (and the programs themselves), with the actual process of allocation and reservation of certain resources being made automatically. Remote compilation and execution allows writing only one source code for a given project (respecting the characteristics of the desired target machine) and thus creating platform-agnostic programs.

2.2. Resource Classification and Performance Criteria

The proposed architecture uses a set of different resources that are allocated to the incoming jobs. For a more suitable planning, every resource can be scored [2,3] as to ascertain the trustfulness of the given resource regarding the online time ratio V and the success ratio G (and to quantify the level of QoS compliance). This way, the resources can be classified for better QoS compliance (the most reliable resources are the most used). Although there are more parameters that can be used to measure quantitative the QoS level [4], the most meaningful in this case are:

$$V = \frac{\text{time}_{\text{online}}}{\text{time}_{\text{total}}} \tag{1}$$

and:

$$G = \frac{\text{jobs}_{\text{succesful}}}{\text{jobs}_{\text{total}}} \tag{2}$$

2.3. Layered Architecture

The proposed middleware amalgamates multiple clusters and distributed general-use systems into a unitary platform, under a centralized or decentralized management system.

The parallel clusters have – usually – a local management system that is highly optimized [5]. When multiple such clusters are interconnected (even if those clusters are homogenous internally, having identical components in the nodes and a global parallel environment), the differences in the platform, operating systems, middleware for parallel applications or administration policies can generate difficulties in the competent and automatic allocation of the available resources and in the seamless running of the client's applications. This allows a better level of quality of service also by simplifying the overall architecture and hiding the QoS penalties induced by the computing systems themselves. Their optimization is executed apart, by the rightful administrators, the management system (including the resource broker) monitoring and scoring every resource accordingly. This makes easier to deploy the middleware across multiple parallel and distributed systems and to create a unique grid infrastructure. The scoring allows to employ specific selection algorithms that will reserve the most appropriate resource (depending on the user's requirements and his job's requirements), making possible to guarantee a certain level of QoS. Supplementary modules can be employed (future work) to obtain a greater redundancy in executing jobs (checkpoint, job migration, etc.) [6] and a superior QoS level, even for volatile grids.

The proposed architecture uses multiple layers to describe the level of operation in the system, as in Fig. 1.

The middleware uses the operating systems mechanism to operate with the user files and the communication systems, compiling and running them in the parallel application environments installed locally on every cluster (resource) (usually MPI or MPI/hybrid) [7, 8]. It also employs the security layer already in use at the cluster/node level. The communication protocols are SSH/SCP/SFTP, used by virtually every *nix OS used on the HPC (High Performance Computing) systems and distributed systems in use. Thus, no other security holes will be opened in the security infrastructure, the systems will be properly patched as part of the maintenance requirements and the deployment will benefit from a stable and tested protocol/security module. The middleware will use the well known and reliable facilities offered by the system itself.

Similar application are usually deployed for dedicated platforms – high throughput computing like Condor Project [9] or distribution for Beowulf clusters (such as Rocks Cluster). These solutions are dedicated to job scheduling or cluster management, lacking a lightweight approach that allows using volatile resources or diverse communities of parallel and distributed systems.

The deployment of the middleware will be on top of the local grid infrastructure, at three levels: user, resource manager (broker) and resource. It will use every resource in conformity with its particularities.

2.4. Architecture, Modules and Functioning

The architecture of the middleware has three main areas.

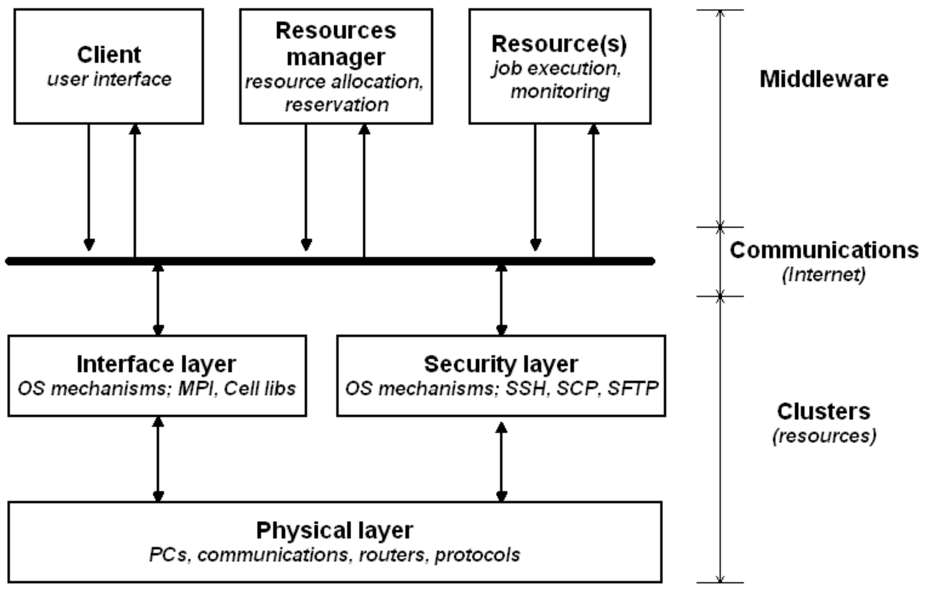
Client
user interface
Resources manager
resource allocation, reservation
Resource(s)
job execution, monitoring
Middleware
Communications
(Internet)
Interface layer
OS mechanisms; MPI, Cell libs
Security layer
OS mechanisms; SSH, SCP, SFTP
Clusters
(resources)
Physical layer
PCs, communications, routers, protocols

Figure 1. The layered architecture of the proposed middleware, built on top of the communication and security mechanisms offered by the underlying operating systems

The user (client) area, which represents the only method of interaction between an authorized user and the resources (without implicating a system administrator). Once the middleware is set and configured, the interface allows submitting jobs (along with the required attributes) and retrieving the result of the jobs either positive or negative if the job fails from user fault (wrong instructions or conditions, errors in programming, errors in algorithms, etc.) or system fault (node malfunction, environment problems, software failure, etc.) – all transparent to the user. The error messages produced by the operating system/middleware/software interfaces can be automatically analyzed or logged for further study (since the module operates on top of the software platform, it can record the activities associated with the client applications). If no suitable resource is found the job is aborted. The user must specify – along the source code of the program – a set of attributes that will define the requirements of the job (number of CPUs, special software requirements, and particular hardware architectures – such as the hybrid ones). These requirements, along with the user's profile (access rights, history, program's nature and execution length) will be factored in selecting a suitable resource (with immediate execution or advance reservation).

The resource manager (the resource broker) area must monitor the available resources (especially the volatiles ones – resources that often come online/offline, completely or partially, or those that have heavily variable performance) in order to make an accurate and valid selection (suitable to the user and efficient for the platform). Also, it must employ an algorithm to select the suitable resources (based on resource specifications, job requirements and user profile). Because of the modularity of the middleware, such multiple algorithms can be used (from the simple to the most complex ones), allowing even to benchmark those algorithms.

The resource manager must have a database with the resources' behaviour in order to make accurate prediction regarding the suitability of the resources. The monitoring module must be permanently online and in communication with the resources, activity that can lead to signification overhead at the manager level (in computation and communication alike). It is best to use a dedicated machine for this module. Since the middlware uses communication methods already present at the system-level, the informations can be send as commands or data strings.

Also, a QoS module must be used to ensure that the selected resource will run the job in time. If a better performance than best-effort is sought, a series of estimates must be generated for every job – the estimated computation length of the job and a programmed date for the start of the program (for advanced reservation). The job estimates can be generated by the user (and mediated by that user's history in giving accurate predictions) for example. If a job is not executed in the allotted time, the event is logged and used for scoring the resource/algorithm.

The resource area will receive user jobs (with the attributes and the broker's instructions) and will compile and run the requested jobs, in conformity with the local parallel environment. Because the compilation is local, the user has the flexibility in writing the source code (providing that he specifies correctly the attributes). Every resource (cluster or grid) will have its own environment, with specific commands, that will be given in the setup stage by the manager of that resource.

The resource area must also monitor the node behaviour and must report to the resource manager.

The proposed architecture of the middleware is described in Fig. 2.

The platform must be modular, flexible, in order to permit using multiple selection and planning algorithms to test their performance and the resource consumption on the given QoS specifications. This is more important for the volatile systems. If a parallel cluster is more reliable (it is homogenous and has a compact nature), the network outage (Internet) can influence greatly (and easily) the volatility of the system based on multiple parameters. There are 4 steps to be taken in order to work: user-resource manager communication, resource allocation, sending the job in accordance with the allocation scheme,

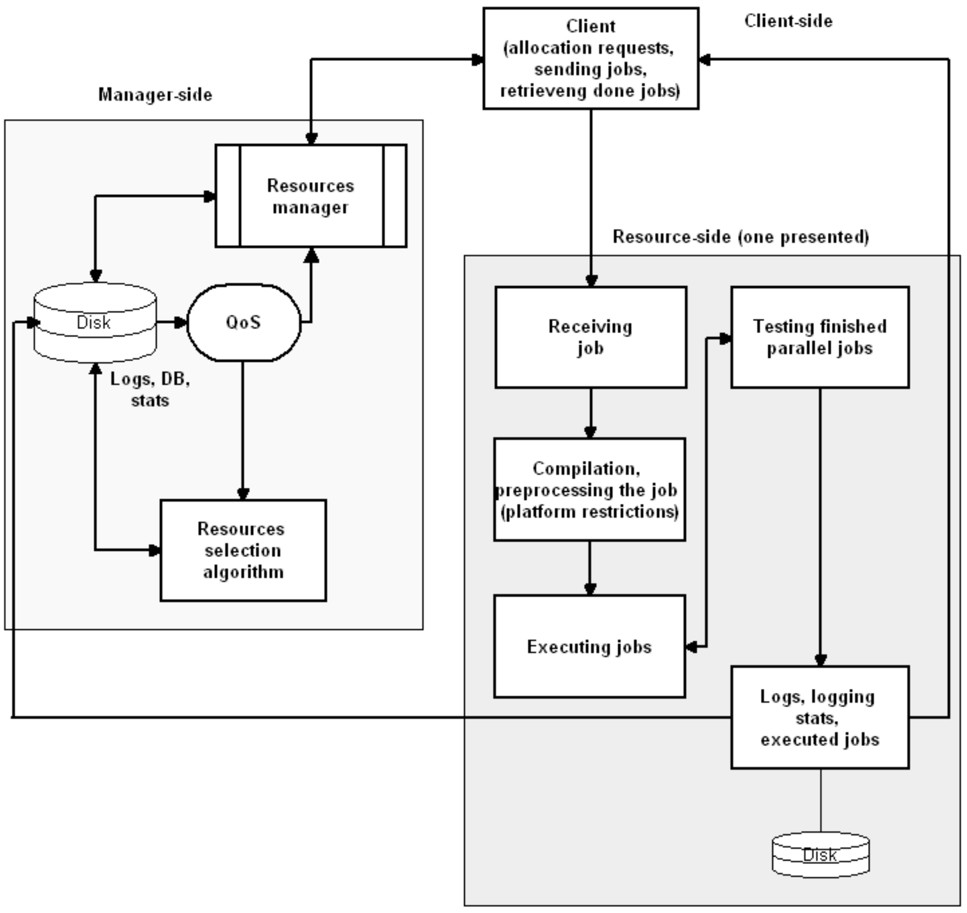
Client (allocation requests, sending jobs, retrieveng done jobs)
Client-side
Manager-side
Resources manager
Disk
QoS
Logs, DB, stats
Resources selection algorithm
Resource-side (one presented)
Receiving job
Testing finished parallel jobs
Compilation, preprocessing the job (platform restrictions)
Executing jobs
Logs, logging stats, executed jobs
Disk

Figure 2. The architecture of the proposed middleware (GLUE)

taking back the processed job. Each activity is logged and analyzed afterward.

One of the main vulnerability of this architecture is the use of a centralized unit to run the resource manager components. This inserts a single point of failure (SPF) vulnerability – in the event that the unit crashes or loses the Internet connection, the middleware becomes broken (requiring recovery from the previous states, possibly with loss of information). There are at least 2 possible methods to avoid this vulnerability:

– Using multiple resource managers that run synchronized; the loss of one unit is covered by one of the neighbours. Every job analysis and resource monitoring is repeated in all units (but only one is active), so that no information will be lost if the unit fails (Fig. 3). To avoid the waste in terms of bandwidth and CPU time (mainly as redundancy) if a replication service for multiple manager modules is employed, only one manager will be active, logging the results of different actions; in the event of a crash, every other instance (selected by a default order) can take its place, losing only the current action (if any). This way, a certain degree of redundancy is assured and the possibility of a SPF is avoided.
– The resource manager can be completely decentralized. Every instance is running a certain number of connections (the users select the managers in conformity with a default order or randomly) so that the grid is decongested (using different units and different communication routes). For efficient planning, at certain time moments the databases must be synchronized (for efficiency and consistency) between the managers. If a unit crashes, its jobs and data can fallback to a backup unit (losing only the current operation, if any) or can be taken by a neighbour

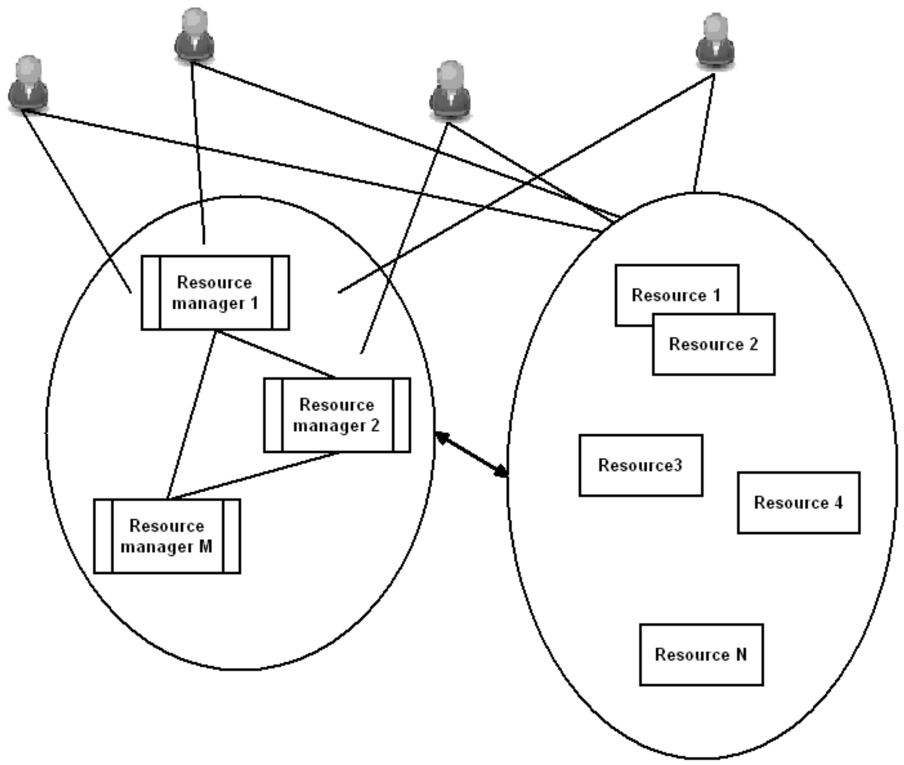

Figure 3. Schematic diagram using distributed (decentralized) resource managers

unit. The users can download actualized lists for the updated resource managers.

3. Conclusions

The proposed architecture brings advantages for interconnecting clusters systems: the architecture is modular (with modifiable modules and the possibility to insert new selection algorithms), flexible and portable, being written using Perl scripting language (scripting language is flexible, portable and powerful, allowing access to the underlying architecture below the middleware level). Also, the architecture provides increased security because it uses a reliable security mechanism that is already in place at the OS level and allows management of different clusters and systems and easy upgrade to the software. The middleware allows transparent, seamless application build and execution for the user. Users can specify certain particularities of the intended target machine to help selection (for example when having commodity hardware based clusters or parallel and hybrid clusters). The application itself is written in Perl, a known scripting language that offers flexibility to the platform and allows designing the work modules (and the addition of new ones).

Some observed disadvantages are: the possibility of SPFs at the resource management level avoidable if the system is decentralized, as been observed [10]; the resources crashing can – missing a recovery solution – lose jobs (worsening the QoS compliance of the system); it cannot allow execution of big jobs across multiple resources (but allows the simultaneous execution of multiple jobs on a resource, providing there are enough free CPUs); computing the planning for resource allocation can generate an overhead for complex algorithms and an increased number of users and resources (although the algorithm can be parallelised for increased performance). Finally, the middleware has a rather specific purpose and reduced applicability outside it (e.g. not fit for commercial domain).

4. Future Work

The middleware will be expanded (functionality-wise) in the future, including new selection algorithms (for the resource manager), preparing the source code to be error-tolerant and stable on the diversity of available grid platforms, improving resource monitoring (increasing number of relevant logged parameters), possibility to state advanced reservation for certain jobs and plans and others and – possibly – methods of increasing the quality of service offered by preventing the loss of jobs (for example by using job replication).

Acknowledgment

This paper was supported by the project "Doctoral Burses at USV", contract POSDRU/6/1.5/S/22, funded from European Social Fund for Operational Sectoral for Human Resources Development 2007–2013.

References

[1] O. Gherman, I. Ungurean, and S. G. Pentiuc, "Principles of interconnecting a hybrid cluster in a grid system," *7th edition of National Scientific Conference Distributed Systems*, No. 7, Dec. 2009.

[2] N. Fujimoto and K. Hagihara, "A comparison among grid scheduling algorithms for independent coarse-grained tasks," *International Symposium on Applications and Internet Workshps SAINT 2004*, 2004.

[3] I. Goiri, F. Julia, O. Fito, M. Macias, and J. Guitart, "Resource-level QoSmetric for CPU-based guarantees in cloud providers," *7th International Workshop on Economics of Grids, Clouds, Systems and Services GECON 2010*, No. 7, 2010.

[4] S. Kalepu, S. Krishnaswamy, and S. W. Loke, "Verity: a QoS metric for selecting web services and providers," *4th International Conference on Web Information Systems Engineering Workshops WISEW'03*, No. 4, 2003.

[5] O. Gherman, I. Ungurean, S. G. Pentiuc, and O. Vultur, "Data communication in a HPC hybrid cluster and performance evaluation," *10th International Conference on Development and Application Systems DAS 2010*, No. 10, May 2010.

[6] C. Dabrowski, "Reliability in grid computing systems," *The Special Issue of the Open Grid Forum (OGF) Journal – Concurrency and Computation: Practice and Experience*, Vol. 21, No. 8, 2009.

[7] K. Koch, "Roadrunner platform overview," Roadrunner Technical Seminar Series, Los Alamos National Laboratory, SUA, Technical report, 2008.

[8] C. Kessler, "Programming techniques for the cell processor," Multicore Day Seminar, Sweden, Technical report, 2009.

[9] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor – a distributed job scheduler, ch. 15, research project at the Unversity of Wisconsin-Madison. (2001). [Online]. http://www.cs.wisc.edu/condor/doc/beowulf-chapter-rev1.pdf

[10] R. Buyya, D. Abramson, and J. Giddy, "An economy driven resource management architecture for global computational power grids," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA 2000*, 2000.

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 47–59, DOI 10.5277/e-Inf120104

A View on a Successful International Educational Project in Software Engineering

Zoran Budimac*, Zoran Putnik*, Mirjana Ivanović*, Klaus Bothe**
*Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad
**Institute of Informatics, Humboldt University Berlin
zjb@dmi.uns.ac.rs, putnik@dmi.uns.ac.rs, mira@dmi.uns.ac.rs, bothe@informatik.hu-berlin.de

Abstract

In this paper, a successful and fruitful joint project will be presented. The project joins participants from 9 countries and from 15 universities. Since it started in 2001, this project entitled "Software Engineering: Computer Science Education and Research Cooperation" helped participants to gain excellent, up to date educational material, apply modern teaching methods, exchange experiences with other participants, and work jointly on the further development of lectures, case-studies, assignments, examination questions, and other necessary elements of a course. Project works under auspices of Stability Pact of South-Eastern Europe, and is supported by DAAD. The project started with the creation of a common beginning course in "Software Engineering", but over time it grew and the number of other courses was developed. Finished almost completely are the courses in "Object-oriented programming", "Software Project Management", "Advanced Compiler Construction", and "Data Structures and Algorithms", and some other courses are under development. Aside from the educational collaboration, project members also developed good scientific cooperation, and published several research papers.

1. Introduction

Since its beginning in 2001, an international project entitled "Software Engineering: Computer Science Education and Research Cooperation" assembles participants from nine countries, and from fifteen universities. Project exists under the sponsorship of "Stability Pact of South-Eastern Europe", and is financially supported by DAAD ("Deutscher Akademischer Austausch Diens", or "German Academic Exchange Service"). At the beginning, the project was concerned with the creation of a common course in the field of "Software Engineering", yet as it progressed, project dealt with the development of a number of other courses. Mostly finished so far are the courses in "Object-oriented programming", "Software Project Management", "Advanced Compiler Construction", and "Data Structures and Algorithms", while some other courses are still under the development. The most developed one is still the course in "Software Engineering". Aside from presentations of a theoretical material, a whole set of learning resources has been developed: e-Lessons, case-studies, team and individual assignments, or pool of questions, for example. Naturally, beside the educational purpose, members of a project later developed nice cooperation within the area of scientific research, and published several papers.

At first, cooperation within the project has been started by the group of researchers and educators that still make the core group of a project, and consists of members from: Germany, Serbia, FYR Macedonia, and Bulgaria. Head and the main coordinator of a project is Professor Klaus Bothe from the Humboldt University in Berlin. Over the years, project was enlarged through the inclusion of participants from other Balkan countries: Croatia, Romania, Bosnia and Herzegovina, Albania, and Montenegro. While originally project was granted for the period of three years, excellent results in cooperation and development of joint teaching resources induced project continuation and new grants year after year, so the project still lasts. DAAD foundation also reported about the successful results of a project in [1] and [2]. The basic information about the project, its participants, and achievements can be found on its home-page [3].

There are several other projects of a similar type and purpose, let us mention [4–8] MuSoft, ISEUC, Swenet, Ariadne, Merlot, where the first three are also dealing with the field of "Software Engineering". Still, we feel that there is a substantial difference between those and our project, mainly in the approach to the creation of teaching material. All of mentioned projects created a set of relatively independent modules that can be combined and used as lecturers decide. In the case of our project, the idea was to create a complete course and the whole teaching material, creating a unity consisting of a sequence of interconnected material, yet allowing the substantial level of parameterization. The scheme behind this concept is to make the whole material usable even to those lecturers for whom "Software Engineering" is not in the key focus of interest.

The main official aim of the project was "academic reconstruction of a South-Eastern Europe". Still, it had a whole list of basic and more down-to-earth aims, of which we list here the most important:

– Inclusion of the course "Software Engineering" into curricula of participating universities;
– Creation of a consensus about the common course in "Software Engineering", selection of topics it will cover, and creation of jointly created pool of presentations from which the participants can choose the most appropriate ones for their university;
– Creation and development of joint teaching and examination materials for selected topics: presentations, case-studies, team and individual assignments, pool of examination questions, adequate literature, lecture notes, etc;
– Forming of bases for the further scientific and educational cooperation in the field, so that the actuality and quality of the teaching material is preserved.

All of the participating countries, more or less, took their part in the development of certain topics or subtopics. Some of the particular activities were:

– Further development of the existing teaching resources;
– Usage of the course as a whole, or some of its parts within the appropriate courses;
– Creation of reports based on the experiences and surveys performed;
– Creation of new topics, case-studies, assignments, and so on;
– Liability analysis, suggestions for the further development paths, creation of re-sources for additional common courses.

In the beginning, the course of "Software Engineering" was based on teaching mate-rials used at the Humboldt University in Berlin, which are in turn based on the text-book on software engineering [9]. All of the most important suggestions from the significant world computer science associations were taken into account [10,11]. This way, all of the basic and introductory topics were created, but also a lot of advanced topics suggested by the ACM, IEEE, and other world expert bodies. Recommendation was also made, that the course should be conducted on final years of studies, after they cover all of the necessary basic notions indispensable for the field. So far, all of the participating universities, followed the recommendation, and the course in "Software Engineering" has been conducted for the students of the final year of studies everywhere.

The rest of the paper is organized as follows: Section 2 presents the contents of the "Software Engineering" course, with all of its basic components. In Section 3, some problems that project participants encounter in their work for the project are presented. Section 4 presents the other courses developed through joint work within a project, based on the positive experience with the first one. Section 5 brings some students' reactions and opinions. Finally, in the 6th Section, more general conclusions are given, and further development paths are considered for the project members.

2. The Development of the Project

Based on the first and the most developed course created within the project (course on "Software Engineering") we will present the current practice of course development and refinement. Over the years, this course went through three, very often overlapping phases:

– During the first phase, existing topics based on [9] were translated from German to English language, and then, through the participation of all project members, refined, polished, and further developed. The bases for the refinement were the experiences with the course presentation at the home university;
– In the second phase, new topics were created and developed, basic and advanced. Those new topics are also continuously refined and polished over the years. General rule is that those new topics are at first developed in English, as a universal, common language for all of the participants. After that, through participation of all interested members, presentations and materials are refined and improved. Only in the final phase, after all of the members are satisfied with the quality of the material, resources are translated back to local languages, if needed;
– During the third phase, the final forms of the teaching materials, agreed by all of the project participants, were translated to local languages [12]. For those purposes, a specialized tool has been created [13].

As the basic outcomes for the project, it has been defined that the teaching materials should help students to develop the following abilities:

1. To work in a team;
2. To have analytic and synthetic approach to decision making;
3. To apply gained knowledge on practical assignments in realistic surroundings;
4. To renew, expand, and continually improve their knowledge, and
5. To make the appropriate decisions during the software development cycle [14].

All of the abilities mentioned here are rather generic, not related only to software engineering, yet that doesn't diminish nor weakens their importance. And, as "basic outcomes", we can say that after 10 years experience, those are quite fulfilled, if we believe the reports we get from the industry.

2.1. Teaching Materials for the "Software Engineering" Topics

One of the basic components of the "Software Engineering" course are teaching materials organized in five parts, with altogether 28 topics covered. Each topic is presented primarily as PowerPoint presentation, enriched with the multiply useful "lecture notes" for the lecturer. Those serve as a starting point for the exchange of ideas between the users of teaching resources (lecturers and students); they contain the answers to the questions presented during the lecture, and they enable creation of printouts of the materials presented during the lectures. Topics are divided into parts and organized as presented in Table 1.

From this pool of topics for which teaching materials were developed, each of the lecturers is allowed to select those suitable for his view on the course, or suitable for the curriculum as it is defined at his/her university. A natural consequence of this agreement is a variety of methods for the usage of teaching resources within the "Software Engineering" course. For example:

– Humboldt University of Berlin, Germany and University "Paisi Hilendarski" from Plovdiv, Bulgaria, use all of 28 topics in their course. The course is conducted on the fourth, final year of the bachelor studies;
– University of Novi Sad, Serbia and University "St. Ciril and Methodius", Skopje, FYR Macedonia, are using almost all of the topics, except several from the last part, "Advanced problems". Some of those topics that are not used are covered within some other master courses, while for some of the others there

is simply not enough time. Also, it is worth mentioning that the course is lectured on the last year of studies for couple of different directions, so there are students from the third, and from the fourth year of studies;

– Universities from Belgrade and Kragujevac, Serbia; Zagreb and Rijeka, Croatia; Podgorica, Montenegro; Sarajevo and Banja Luka, Bosnia and Herzegovina; Tirana, Albania, and Timisoara, Romania selected a subset of topics. Depending on the university, the number of topics varies between 5 and 12, and those are incorporated successfully into already existing courses on "Software Engineering", becoming the integral part of those;
– A special case is Polytechnic University of Tirana, where the course is not conducted during the regular school-year, but instead as a one-week crash-course, when about 18 topics are presented. Lecturers are visiting professor from Berlin, Germany, and assistant from Novi Sad, Serbia, and the course is conducted as a part of master studies. After the four crash-courses, part of the topics is taken over by local assistants from Tirana, while the general plan is that the whole course will be once conducted by local lecturers.

With this variety of types of course conduction, it is quite likely that topics are continuously being developed and refined. Large number of lecturers, each one with his/her own teaching style, habits, and pedagogical principles, guarantees the actuality and quality of teaching resources. All of the new ideas, techniques, suggestions, and innovations are exchanged during the regular meetings of the project members, conducted each autumn at some of the participating countries.

One of the methodologies presented, certainly deserves greater attention. Since at the moment, the most of the development methodologies are built around UML, this methodology is represented within the course also. Methodology is not introduced formally, since it has been studied within other, previously taught courses in that manner. Still, because of this fact, the lecturer is in a position to introduce the methodology through examples.

Within the introductory topics, the importance of UML is explained, and so are the notations, being the part of it. The "body of knowledge" for it is not described deeper, because it was the part of several compulsory courses preceding the "Software Engineering" course. Later on, the methodology is used wherever it is needed, as convenient to the lecturer.

To confirm that the students covered this important methodology sufficiently, one of the obligatory assignments students have also requires knowledge and usage of it. We will not discuss it in more details here, since more about this will be given in a subsection dealing with the assignments.

2.2. Case-studies for the Course of "Software Engineering"

Second important course component, linked with both theoretical and practical exercises, are relatively complex case-studies. The main reason for usage of those case-studies is the need to illustrate theoretical concepts presented during lectures on some practical and realistic examples. The original course, used as a basic element for the development of the final project resulting course, used throughout the lectures two case-studies:

– "Seminar organization" – a software system, taken and adapted from [9], used to help running the company that deals with the organization of various educational seminars and their presentations to interested clients. The system is also supposed to help with: contact with clients and other companies, communication with the lecturers, students, hotels, travel agencies, and all other necessary users and services. This case-study is used within ten topics to illustrate theory presented during the lectures.
– "XCTL" – a real life software system, used to control the work of measuring instruments at the Institute of Physics, Humboldt University in Berlin, Germany [15]. System was analyzed, measured and enhanced using methods of re-engineering, software metrics, software testing, and some other fields, so the students

Table 1. Topics presented within a course

Part I: Introduction to software engineering 1. What software engineering is 2. Quality criteria for software products 3. Software process models 4. Basic concepts for software development documents	*Part III: Software Design* 15. Overview of design activities 16. Structured design 17. Object-oriented design
Part II: Requirements engineering 5. Results of the "analysis and definition" phase 6. Cost estimation 7. Function-oriented view	*Part IV: Implementation and testing* 18. Implementation 19. Systematic structured testing 20. Functional testing
8. Data-oriented view 9. Rule-oriented view 10. Structured analysis 11. State-oriented view 12. Scenario-oriented view 13. Object-oriented analysis 14. Formal software specification and program verification	*Part V: Advanced problems* 21. Software metrics 22. Maintenance 23. Reverse engineering 24. Quality of software development process and its standardization 25. Introduction to software ergonomics 26. User manuals 27. Project management 28. Configuration and version management

are faced with the realistic results of those analysis, within four topics. Being sufficiently big, system and the results of the mentioned measurements, present adequate and satisfactory base for the explanation of all needed methodologies.

During the years of usage, more case-studies have been developed. At Skopje, FYR Macedonia, case-study covering classical functions of a university library was developed. Currently, at the University of Novi Sad, Serbia, two case-studies are arising. One of them presents a system for agent selling of consumer products, while the other is again adapted from [9] and is dealing with the control console for a car. This last case-study is especially important and different from the others by being the only one from a technical domain.

The idea behind the existence of several case-studies is the wish that the lecturers have a possibility to interchange those examples, depending on their needs and wishes. This type of usage would require some deeper work by the lecturer, who would have to change presentations and examples, but is doable. Another, even more important moment, connected with the case-studies is the fact that they are used within the complex team assignments, used to assess the knowledge students gained during the lessons, and their ability to put it into practice. Within those assignments, it is often necessary to change the case-study used, in order to prevent students from cheating and taking the solution of previous generations. This requires almost no additional effort, since all of the case-studies contain all of the elements needed by all of the assignments.

2.3. Assignments for the Course "Software Engineering"

The third essential component of the course is the assignments, created and prepared for team solving. Over the course conduction students are obliged to solve certain number of team assignments. As a common practice for all universities, it has been accepted that the students have to achieve 50% of the points for the assignments, but how are those points used later, is different. At some universities, this is just a condition that students have to fulfill to be able to approach the exam. On other universities, besides being a condition for the exam, number of points gained for the assignments is used for calculation of the final grade.

With the assignments, the situation is the same as with the topics. Number of assignments created is much bigger than it is necessary for a successful realization of the practical part of the exam. This way, each of the lecturers has enough material to be able to choose those assignments that (s)he finds the most suitable compared to: the topics presented, quality and affinities of students, or compared to other courses available that semester at the university in question.

This gives the course flexibility, enabling usage of the assignments within crash-courses, one-semester course, and within longer, two-semester courses. Large number of assignments, and the fact that they are parameterized, gives lecturers also the possibility to exchange assignments over the years, so the plagiarism and copying of solutions is decreased to a bearable level. The pool of assignments consists of the following ones:

Assignment 1: Reading and reviewing of the preliminary requirements specification and requirements specification for a case-study "Seminar Organization" (or alternative). Students are supposed to find and correct errors, misunderstandings, and ambiguousness, and suggest the ways of improving the text. This assignment was generally created in order to test students' ability to present their ideas in a precise and concise manner;

Assignment 2: Application of the "Function point" method on the requirements specification, in order to calculate the price and the human resources needed for a chosen case-study. The purpose of this assignment was to create a habit for students to follow the rules and procedures they heard during classes;

Assignment 3: Analysis of a product model, resulting from the application of structure analysis. Again, "Seminar Organization" (or alternative) case-study is used, where students are faced with several data-flow diagrams (including some errors), and are required to notice those, and suggest the ways of improving the diagrams. Data-flow diagrams are taken from an important and distinguished book [9]. As a consequence, we hope to teach the students not to trust blindly to any authority, but to observe and check all of the information they reach;

Assignment 4: Development of a part of a static model by creation of class diagram and use-case diagram. Students are this time faced with a new, small problem, so their creativity is tested here.

This assignment was the one intended to check on students' ability to use UML methodology. While that seemed *not* to be the problem, a need for creativity that this assignment required, was one of the largest problems amongst the assignments, be-cause the usual fact was that the students were over-creative;

Assignment 5: Development of a formal specification for several new operations, based on formal specifications presented during lectures. With most of the students selecting this study direction because of their love for computers, this assignment has a purpose of showing them that they also need some knowledge in other, related fields, such as mathematics and formal logic;

Assignment 6: Analysis and review of another teams' solution of the fourth assignment "Development of the part of a static model". Students are here presented a different view on the same problem, a have to comment on it, and critic it. This gives students a chance not only to see the different view on the same problem, but also to try to assess the value of someone else's solution;

Assignment 7: Application of software metrics methods, through usage of a tool. This assignment faces students with the regular situation in a working life of a soft-ware developer, namely, with the need to find, install, learn, and use tool never seen before;

Assignment 8: Specification of a regression test. Students are required to define a set of test-cases that guarantees branch-coverage condition, using the tool for regression testing. This one, and the next one, introduces students with the most expensive, and probably the most important part of the software development life-cycle, the testing process, and

Assignment 9: Creation of "classification tree" for software testing. Again using the "Seminar Organization" case-study, this time in combination with the tool for functional testing, students are required to define a set of test cases, and check the correctness of a program.

In practice, for all of the participating universities, the same procedure is applied: teams get their assignment and a term of no less than two weeks, to submit a solution. Team members are required to read and review the assignment and given material, to contemplate about it, and to create their version of a solution before the team meeting. Over (usually) several meetings, a team discusses individual judgments, and creates a common solution.

Occasionally, but compulsory after the first assignment is submitted (but not yet graded) a class is organized where the team of students who submitted the most intriguing solution for the first assignment, present it to other students. Decision, and classification of assignments so that "the most intriguing" solution is found, is up to the assistant. While the experience helps for making the right choice, we think that the presentation of *any* solution would be interesting. Namely, solutions *are* different, so each one will have their opponents, students who would challenge and confront it, so the fruitful discussion would happen in any case.

The rest of the teams, confronted with a different view on the same problem con-template, analyze, discuss, and critic suggested solution. Here we can also mention the cooperation of assistants from Humboldt University Berlin, Germany and University of Novi Sad, Serbia, who jointly created the most logical and most appropriate "correct solutions" for all of the assignments, based on several years of experience with those submitted. Typical, common errors are then presented to other students, while this solution is also used as a model for checking other assignments. Naturally, every year this "correct solution" is tested and further developed, through fruitful discussions with students.

Because of the trend noticed that some of the students participate less, or do not participate at all in some of the assignment solving, while the other students cover for them, at some universities the "experiments" started with the usage of wiki as a tool for the purpose of assignment solving. The idea behind this is to recognize, by care-fully reading through the history log of a learning management system, how much each of the team members participated in creation of final document. Since the application of this technique is still new, it is too easy to comment on it more. Still, the first experiences show that the better students are quite satisfied with this methodology, while those inclining to "cheating" at the exam had a lot of objections. Still, it is worth mentioning that the whole idea aroused from the pleas of students given in the surveys about their satisfaction with the course. Namely, there have been several cases where students asked the lecturers, to find the way of either punishing students not participating in assignment solving, or rewarding those who did most of the hard work.

Another characteristic problem with the assignments is a universal one, noticed at each participating university. Students, who are less ambitious, abandon their team as soon as they achieve minimal number of points needed. As mentioned, assessing the assignments is different amongst universities. So at some this means that students achieved 50% of the points are allowed to approach the exam, and they *are* not interested to learn additional methods and techniques. On other universities it additionally means that they are satisfied with the lower grade. In any case, this puts additional burden on those students willing to continue with the assignments. Assignments are created for team solving, so when only one or two students approach the work, they are much more difficult! Adequate and fair solution for this problem has not been found yet, and participants from several universities are working on it.

As mentioned, not all of the assignments are used each year at all universities. That decision depends on some subjective factors - choice and ideas of a lecturer, but also, and mostly objective factors. Some of the assignments are based on usage of the tools that require significant financial investments and registration of the faculties for their usage, which is not always possible. At other faculties the course is shorter, so there is not enough time for all of the assignments to be conducted.

3. Difficulties and Peculiarities of the Course

Over the years, quantity of teaching material for the course grew to a significant size. There are 28 presentations with lecture notes included, 5 case-studies (more or less used and finished), 9 assignments, collection of around 500 examination questions, and many more. Also, most of those resources exist in several different languages, because interested lecturers were usually obliged by local laws to translate materials to local languages.

Under these circumstances, a natural problem arose. Modifications and improvements of the materials are hard to maintain, evolve, and spread throughout all of it. Even the corrections, improvements of style, grammar, typing errors, or occasional logical or material errors, are repeated over and over again. Between the team of core members of a project, a possibility to employ some kind of configuration management system is considered, but not yet utilized.

Even biggest problem is present at universities who use local, translated versions of the material, we must admit. Those lecturers improve and refine local versions, and hardly are able to find the time to send those refinements back to be used in English versions. So, two versions diverge from each other more and more each school-year, without realistic chance to become one again ever.

Discussing the above problem, the core members were able to recognize additional complication. Even if creators of the local versions find the time, collect all of the versions to send them back, the question would be – send them back to whom? Who is the one (or possibly more) person(s), who would be able to dedicate enough time to incorporate new ideas and findings, combine those coming from several different sources and in several different languages, and create a valid, refined new teaching material. The conclusion was that the solution would be if an employee could be found, permanently connected with the project, in charge of keeping the material up-to-date, of collecting and unifying changes made at different participating universities. Still and unfortunately, this idea is at the moment unsolvable, because there is no possibility for such a thing within a project.

A temporary solution for a problem of material unification occurs every now and then, in a form of an interested student! Several students at different universities, project members were employed to do certain tasks of common interest, as a part of their seminar papers, diploma, or master thesis. Not being a lasting solution, this option helps at least in a part, and decreases number of unsolved issues.

There is one addition to the grading process, as an experiment at some of the participating universities. Besides those generally agreed and used big team assignments, another type of “assignments” is also used. Namely, since “Bologna rules” of course conduction require regular students participation and course attendance, at some universities this was becoming a problem to some extent. In a situation when the course is conducted at master studies, the most of the students are regularly employed, and were unable to be present at all of the lectures. On the other hand, for some of those regular, undergraduate students, topics and lectures, but also the field in general, was not too interesting, while the course was obligatory. Their presence at the lectures was because of that more an annoyance to other students, than help to them, since they were not paying attention at all.

As a result, at some universities, a solution was found through a free interpretation of a notion of “course attendance”. Not *all* of the present students were given points for attendance, but only those who actively participated in the lectures, answering (and asking) questions and commenting on presented materials. For questions asked during lecture presentations (possibly and usually several during one lecture), students were able to earn so-called “bonus” points, and advance their grades. Those points could help a person to improve, but also make up for the points lost at tests, or within the assignments.

Very soon, only those interested in the field were present at the lectures, while the others were just involved in teamwork, and came to the final exam. This had good consequences on the class atmosphere and learning curve of those present. Still, in order to give the equal possibility to all of the students to earn points for participation, and interest them in the field, small assignments, requiring some thinking, searching, and researching were often given to students to be solved at home. The first person, who answers the given question by mail, or using the common "forum" of a learning management system used, would be awarded a bonus point.

4. The Other Courses Created Within the Project

Based on the good experiences and successful cooperation realized for the "Soft-ware Engineering" course, members of the project decided to extend their cooperation to other courses. So far, joint work has been conducted for the development of additional four courses, where some of them are already largely used, while the others are still partly in the development phase. All of those courses are related to the "Software engineering" course: either as a required pre-knowledge, further developed part of it, or simply belonging to the very close expert field.

- "Joint teaching materials on OOP using Java" [16] Java, is a subproject started very early, after the beginning of the project "Software Engineering: Computer Science Education and Research Cooperation", during the year 2004. The subproject is dealing with the development of joint teaching materials for several project participating institutions. Since this course already existed in curricula of all countries, the entire effort was invested to a pure educational and research cooperation, without triggering a complicated administrative procedure of introducing the new course, as it was the case with the course in "Software Engineering". Participants of this subproject are lecturers from six universities (project members). By joining their existing materials, refined and improved, but also by creating the new material, relatively fast a new course of a very high quality has been created. Such a new course is successfully conducted by six universities who contributed to its' development;
- "Software Project Management" is a subproject started in 2004, with the aim of development of additional material for this very important subfield of "Software Engineering". For this purpose, mostly participant from the University of Novi Sad, Serbia and Humboldt University of Berlin, Germany were active, with the partial help from the University "St. Ciril and Methodius", Skopje, FYR Macedonia. So far, course is successfully conducted only in Novi Sad, Serbia, since the year 2005;
- "Advanced Compiler Construction" is a subproject started in 2004. The important issue here is that courses with this name already existed at the universities in Novi Sad and Belgrade, Serbia, and Humboldt University in Berlin, Germany. The main purpose of a subproject was to make those courses compatible, and improve them through the exchange of the existing, and creation of new teaching resources. This cooperation was also successful and the new course has been conducted for five years now, at mentioned universities. Possibility to transfer this and other developed courses to other universities, project participants is also considered, and probable in the future;
- "Data Structures and algorithms" is a subproject started in 2006. Within this one, Universities from Novi Sad, Serbia, and Skopje, FYR Macedonia were largely involved. Since course under this name and with the similar contents exists at all other universities, project participants assisted and helped in the development and review of the course;
- For the last three mentioned subprojects, the development of specially dedicated web-pages is under construction, while the teaching materials that are developed so far, and are used in teaching, can be found at the local learn-

ing management systems of the participating universities. For Department of Mathematics and Informatics in Novi Sad, that page is available at [17].

5. Reactions and Opinions of Students

Almost everything we stated in this paper was the views from the position of lecturers. What about the other side? What students think about the course? For autumn workshops of the project participants, we have for years prepared reports with opinions and answers to the anonymous questionnaire we ask our students to fill. We will summarize those results here, and present part of the results.

For start, let us first recognize the character of our students. Even though their average grade is between 7 and 8 (60%) (on the scale from 6–10), only 32% of them between 8–9, and just a symbolical 8% over 9, their expectations stated before the course in "Software Engineering" were much higher. They stated that they will de-serve grade 10 (17%), or 9 (63%)! The rest of 20% said they will deserve the grade 8, and even that grade is above their average.

In reality, the problem with the course was *not* passing it, but grades were at the same level as for the other courses. Also, a general conclusion over the years was that the more students attended the lectures, their grades were higher. The more concrete questions and answers for bachelor students at the University of Novi Sad, Serbia, and master students of the Polytechnic University of Tirana, Albania were:

- Considering the question "Rate the amount of knowledge offered in the lectures" (where grades meant 5=too much, 1=too little): over the last five years, grades for the course were almost ideal, around 3. Grades given to the course by students of master studies in Tirana, were a little bit towards "too much", but we must admit here the existence of language problem, since the course is conducted in non-mother tongue;
- About the question "Rate the contents of the lecture" (5=too easy, 1=too difficult) in last five years, we received the following grades: 2.75, 2.78, 3.00, 3.00, and 3.04. Master student had on the average, almost the same opinion. Again, this gives the course almost ideally balanced difficulty of its' content;
- For the question "Is the course well structured", for the first time there is a signifi-cant difference in opinion. Students of undergraduate studies in Novi Sad rated the course on the scale 5=very well to 1=unstructured, with 3.4 on the average. Still, master students had a much higher opinion of a course, around 4.5;
- Similar difference was repeated with the question "Is the amount of information on slides adequate?" where undergraduate students rated the course with 3.3, while master students estimated it with 4.1. Without wishing to dispraise students of un-dergraduate studies, we estimate that master students have more pre-knowledge, and thus better chance to assess our course accordingly;
- The greatest difference in opinion was shown with the question "Are the slides well-structured and clearly arranged?" Grades from undergraduate students were between 3.4 and 3.7. At the same time, the lowest grade by master students was 4.4, while the other grades went up to 4.63;
- Both groups assessed very well knowledge of the lecturers (undergraduate around 4.3, masters around 4.7), their preparation and readiness for conducting the lectures (4.3 by undergraduate, almost 5 by masters), their engagement during the lectures (same as the previous question), and their willingness to answer questions (by both around 5, this time);
- Finally, when we consider some more general opinions about the course, situation is probably the best graded:
  - Did you learn a lot of new things (5=much, 1=not), grades were around 4.05 by the undergraduates, and around 4.20 by masters;
  - Do you think that the content of the lectures was useful: 4.1 by undergraduates and 4.4 by masters;
  - Overall rank of the course (5=very well, 1=bad) grades are 4 (with a very slight

margin over the years) by undergraduates, and 4.5 by masters.

What we feel that must be noted here is that for the master students, attendance of the lectures was obligatory. For undergraduate students in Novi Sad, it wasn't. Yet, even though undergraduate students estimated that they attended only about 40% of the lectures, on the average, and that this fact forced them to spend more time both on studying and assignment solving, they felt qualified to assess presentations, lecturers and the course in general. What can give us optimistic bust is the fact that even those lower grades were very good, and that they prove that joint creation of common courses worth the effort.

6. Conclusions

Experience we gained so far during the ten years of creation and usage of common course and teaching resources developed by the project participants, can be in short enumerated with several basic results [18]:

– Courses are developed as a whole set of resources, containing presentations, but also lecture notes, assignments, case-studies, and all other necessary materials. Still, our experience shows that such a course can be adjusted to local curriculums, and also to style and needs of a lecturer, and be taught in different ways, at different universities, and different countries;
– Courses have been taught at different universities in a different manners and using a diverse subset of teaching materials, yet in each case those resources proved to be extremely useful;
– Examination and practice assignments were also used in different ways, but they also proved to be developed in a satisfactory quality and quantity to fulfill all of the needs arising;
– Exchange of the teaching materials is worthwhile. The development time is greatly shortened, guarantee for actuality and quality of the material is largely increased, exchange of experiences in enabled, and so is the exchange of technical and educational findings;
– Development of "lecture notes" enables usage of the teaching material and lecture conduction even to lecturers with less experience in the field that is taught. On the other hand, for those closer to the field, preparation time for the lectures was largely shortened;
– The validity of the previous two claims we can illustrate and prove by actual situations at Skopje University in FYR Macedonia, and Rijeka University in Croatia. Since the course was already conducted at Humboldt University in Berlin, Germany, and University of Novi Sad, Serbia, all of the needed materials were developed and practically tested. As a consequence, the whole course was rapidly introduced at Skopje University, where the professor and her assistants needed only two months to introduce the course. Still, this was the extreme case, caused by the fact that both professor and the assistant were the long time members of the project, that they have heard lectures during the workshops, heard the experiences with the assignments and case-studies, and so on. The other mentioned University of Rijeka is a more natural case that even better proves that this joint preparation of a course was worthwhile. Rijeka University is a member of our project, but a professor that introduced the course there, never was. She just took over our joint course, including all of the presentations, case-studies, assignments, and everything else, and within six months, she started conducting it successfully. We are still waiting for the written, numerical results of a survey conducted on students, but verbally given opinions and experiences are highly positive;
– Existence and usage of the common material enabled also exchange of experiences between lecturers, conductions of surveys and application of students' wishes and suggestions, as much as the continual improvement of courses;
– The various experiences collected over the years have been described in several papers

published over the years, at several conferences and journals: [12–14, 18–24].

Since the starting idea of the project was the exchange of experiences, rising of the teaching quality, and decreasing the effort needed for a creation of new courses, the above conclusions clearly prove that these aims are not only fulfilled, but surpassed by far. Based on the experiences gained with the first course in "Software Engineering", collaboration was extended to the development of new courses, already used in practice, but still refining and developing.

Currently, most of the efforts in a process of further refinement of the course are aimed at the development of appropriate e-Learning support for the course. Depend-ing on the University, these activities are in different phases. Universities in Novi Sad, Serbia, Skopje, FYR Macedonia, and Rijeka, Croatia incorporated joint materials into their learning managements systems, and students are freely using them. Even more, in Novi Sad, e-Lessons, glossaries, and quizzes for knowledge self-testing based on original presentations were developed, so that the students can choose the type of study resources they prefer.

Lecturers gathered around this project didn't stopped just to deal with the educational elements – great cooperating experiences with the development of new teaching materials have been deepened with the research cooperation. This cooperation extended over the limits of the courses that started it. Autumn each year is the time when participants of the project gather to exchange ideas and experiences, and to communicate and consult about the further educational and research efforts. Each year, these workshops include young assistants, but also the best students from the participating universities. Over the ten years of project existence, among the students that participated in the workshops, more than 10 have been selected as new assistants at various participating universities.

References

[1] "Bringing curriculums and equipment up to date," DAAD, Sep 2002, issue 3.

[2] "DAAD newsletter," www.daad.de/imperia/md/content/hochschulen/stabilitaetspakt/newsletter/2009_1-en.pdf, 2009.

[3] Project home-page. (2011). [Online]. http://www2.informatik.hu-berlin.de/swt/intkoop/jcse/

[4] E.-E. Doberkat, C. Kopka, and G. Engels, "MuSofT – multimedia in der softwaretechnik," *Softwaretechnik-Trends*, Vol. 24, No. 1, 2004.

[5] K. Modesitt, "International software engineering university consortium (iseuc), a glimpse into the future of university and industry collaboration," in *Proceedings of 15th CSEET*, Covington, Kentucky, USA, 2002, pp. 32–41.

[6] T. Hilburn, G. Hislop, M. Lutz, S. Mengel, and M. Sebern, "Software engineering course materials workshop," in *Proceedings of 16th CSEET*, Madrid, Spain, 2003.

[7] Ariadne Project. (2011). [Online]. http://www.ariadne-eu.org

[8] Merlot project. (2011). [Online]. http://www.merlot.org

[9] H. Balzert, *Lehrbuch der Software-Technik.* Spektrum Akademischer Verlag, 1998, Vol. 1 and 2.

[10] Computing curricula 2001, ACM and the Computer Society of the IEEE. (2011). [Online]. http://www.acm.org

[11] P. Bourque and R. Dupuis, Eds., *Guide to the Software Engineering Body of Knowledge SWEBOK.* IEEE Computer Society, 2001.

[12] K. Bothe, K. Schuetzler, Z. Budimac, and K. Zdravkova, "Collaborative development of a multi-lingual software engineering course across countries," in *Proceedings of 35th ASEE/IEEE Frontiers in Education Conference*, Indianapolis, USA, 2005, pp. T1A–1–T1A–5.

[13] K. Bothe and S. Joachim, "Tool support for developing multi-lingual course materials," in *Proceedings of ONLINE EDUCA Berlin, 10th Intl. Conference on Technology Supported Learning & Training*, Berlin, Germany, 2004.

[14] Z. Budimac, Z. Putnik, M. Ivanovic, K. Bothe, and K. Schuetzler, "On the assessment and self-assessment in a students teamwork based course on software engineering," *Computer Applications in Engineering Education*, Vol. 19, No. 1, 2011, pp. 1–9.

[15] Behavioral specification (requirements) of XCTL-control program. (2011). [Online]. http://www2.informatik.hu-berlin.de/swt/intkoop/jcse/case_studies/xctl/XCTL-Man-Adj.html

[16] Java course home page. (2011). [Online]. http://perun.pmf.uns.ac.rs/java

[17] Course home page. (2011). [Online]. http://perun.pmf.uns.ac.rs/moodle

[18] K. Bothe, K. Schützler, Z. Budimac, Z. Putnik, M. Ivanovic, S. Stoyanov, A. Stoyanova-Doyceva, K. Zdravkova, B. Jakimovski, D. Bojic, I. Jurca, D. Kalpic, and B. Cico, "Experience with shared teaching materials for software engineering across countries," in *Proceedings of Informatics Education Europe IV*, Freiburg, Germany, 2009, pp. 57–62.

[19] K. Bothe, K. Schuetzler, Z. Budimac, K. Zdravkova, D. Bojic, and S. Stoyanov, "Technical and managerial principles of a distributed cooperative development of a multi-lingual educational course," in *Proceedings of the 1st Balkan Conference in Informatics*, Thessaloniki, Greece, 2003, pp. 112–120.

[20] K. Bothe and S. Joachim, "Interactive tool-based production of multilingual teaching and learning materials," in *Proceedings of the 5th IEEE International Conference on Advanced Learning Techniques*, Kaohsiung, Taiwan, 2005, pp. 516–518.

[21] Z. Budimac, Z. Putnik, M. Ivanovic, and K. Bothe, "Common software engineering course: Experiences from different countries," in *Proceedings of the 1st International Conference on Computer Supported Education*, Lisboa, Portugal, 2009, pp. 375–378.

[22] M. Ivanovic, Z. Budimac, Z. Putnik, and K. Bothe, "Short comparison of tasks and achievements of different groups of students with the common software engineering course," in *Proceedings of the International Conference on Software Engineering Theory and Practice*, Orlando, USA, 2009, pp. 84–91.

[23] K. Zdravkova, K. Bothe, and Z. Budimac, "SETT-net: A network for software engineering training and teaching," in *Proceedings of the Information Technology Interfaces*, Cavtat, Croatia, 2003, pp. 281–286.

[24] ——, "The structure of SETT-net," in *Proceedings of the Eurocon*, Ljubljana, Slovenia, 2003, pp. 126–129.

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 61–70, DOI 10.5277/e-Inf120105

Towards Automation Design Time Testing of Web Service Compositions

Dessislava Petrova-Antonova[*], Sylvia Ilieva[*], Ilina Manova[**], Denitsa Manova[**]
[*]*Sofia University, Faculty of Mathematics and Informatics*
[**]*Rila Solutions*
`d.petrova@fmi.uni-sofia.bg, sylvia@acad.bg, ilinam@rila.bg, denitsat@rila.bg`

Abstract

Service-Oriented Architectures (SOA) allows software applications to interoperate in a new way in distributed environment. Currently, web services are the most widely adopted technology for implementation of SOA. However, they bring a number of challenges to development as well as to testing. Testing web service compositions is one of the major problems in SOA domain that is due to the unknown context, absence of web service source code, multiple provider coordination, lack of tool support, etc. In such context, the paper proposes a framework, named Testing as a Service Software Architecture (TASSA), which aims to provide design time testing of both functional and nonfunctional behavior of web service compositions described with Business Process Execution Language (BPEL). TASSA consists of set of tools that can be used together with existing development environments of service based applications. The paper focuses on an approach for negative testing and unit testing of BPEL processes. The negative testing is supported by TASSA tool, called Fault Injector tool, which implements a fault injection technique providing message delays, wrong message data, etc. The goal of unit testing is to test a BPEL process in isolation from its dependent web services. The isolation technique is implemented in another TASSA tool, named Isolation tool.

1. Introduction

Service-Oriented Architecture (SOA) is a dominant paradigm for design and development of distributed and interoperable software applications. The most widely adopted approach to SOA implementation is web services based on standards such as SOAP, WSDL and UDDI. Testing such implementations is challenging for various reasons. It is difficult to simulate all possible configurations and loads during testing process due to dynamic nature of web services and their consumers, varying load on SOA infrastructure and underlying network [1]. In addition, web services are outside of the control of consumers, leading to potential misunderstandings between parties.

The need of automation of SOA testing process results in targeting of many research efforts to SOA domain. There are separate testing tools and complex proprietary frameworks that can be used for testing of web service compositions, but open, a complete solution that meets SOA testing challenges is still missing. This paper addresses this problem by proposing a framework, named Testing as a Service Software Architecture (TASSA). The main goal of TASSA is to support the testing, validation and verification of both functional and nonfunctional behavior of web service compositions at design time [2]. It consists of set of tools that can be used together with existing development environments of service based applications. This paper focuses on two of TASSA tools, namely Fault Injection tool and Isolation tool that respectively provide functionality for negative testing and unit testing of web service compositions described with

Business Process Execution Language (BPEL). The goal of the negative testing is to test the BPEL process in case of message delays, errors in message data, wrong business logic, etc. The unit testing aims to test the BPEL process in isolation from its partner web services. The essence of fault injection and isolation techniques as well as their automation and application to real scenario are presented in the paper.

The content of the paper from this point forward is organized as follows. Section 2 introduces current approaches for web service composition testing. Section 3 presents TASSA tools. Section 4 shows experimental results from execution negative test cases over a sample BPEL process in TASSA framework. Finally, section 5 concludes the paper.

2. Related Work

The BPEL inherently brings a challenge for testing due to its specific syntax, dynamic binding during execution and the fact that it integrates web services implemented by various providers. This section presents a review of various techniques, methods and tools that meet this challenge. The generation of the test suite for basis path testing of WS-BPEL and an accompanying tool that can be used by service testers are presented in [3]. The proposed testing tool does not support all XML schema data types in the generation of test data (only integer, float, boolean, and string are supported). Also, only sequence, condition, and repetition patterns of control are allowed. The tool does not consider infeasible paths that cannot be accessed. In [4] the authors propose a gray-box testing approach that has three key enablers: test-path exploration, trace analysis, and regression test selection. In order to improve the preciseness of the generated test paths IBM BPEL extensions, like Java snippets, need to be handled. The experimental results show that the test-generation time is linear to the number of test paths searched. Thus a more efficient generation algorithm is needed to avoid the performance problem for complex processes. In [5] a formal model for an abstract-based workflow framework that can be used to capture a composed web service under test is introduced. It is focusing on verifying, based on structural-based testing strategies that a composed web service can function correctly according to its semantic, activities and data dependencies. In [1] the authors use High-level Petri nets (HPNs) to model BPEL web service composition. The relationship between BPEL conceptions and HPNs is specified in four levels according to inter-service, intra-service, inter-activity, and intra-activity. In [6] a model-driven approach toward generating executable test cases for the given business process is presented. Its drawback is that the generated test cases still needs some effort to develop the adapter and codec to run. In [7] WSA is proposed to model concurrency, fault propagation, and interruption features of BPEL process. A model checking based test case generation framework for BPEL is implemented. An open issue is to prove the correctness of the model transformation. The approach in [10] is more applicable to programs without complex variable sharing or process interaction patterns. The messages' maximum enablement is limited to one time during the transformation of BPEL process into Extended Control Flow Graph XCFG. Also, the exception handling logic does not affect the other running threads, which run to undisturbed completion. An advantage of the approach is that it is modularized so that it can be used together with other testing technologies. It avoids the state space explosion problem and is applicable for programs in which concurrent computation units have only very few or no shared variables or other types of synchronization. In [8] an approach to unit testing of WS-BPEL and a tool prototype extending JUnit are presented. The proposed BPEL-Unit provides the following advantages: allow developers simulate partner processes easily, simplify test case writing, speed test case execution, and enable automatic regression testing. In [9] the authors propose a layer-based approach to creating frameworks for repeatable, white-box BPEL unit testing, which is applied to new testing framework. The framework does not provide much support in test case creation and the monitoring of the

Table 1. Comparison of BPEL testing approaches

Approach	EH	FH	A	ER	TCG	NT
Lertphumpanya [3]	no	no	yes	yes	yes	no
Li [4]	yes	yes	yes	yes	yes	no
Karam [5]	no	no	no	no	no	no
Yuan [6]	no	no	yes	yes	yes	no
Zheng [7]	yes	yes	yes	no	yes	yes
Li [8]	no	no	yes	yes	yes	no
Mayer [9]	no	no	yes	no	yes	yes
Dong [1]	no	no	yes	yes	yes	no
Yan [10]	yes	yes	no	yes	yes	no
Karam [5]	yes	yes	yes	yes	yes	yes

PUT. Developers have to manually prepare large amount of coherent XML data and XPath expression to compose a test case. This is a painstaking task considering the complex structure of involved XML data. The results from comparison analysis of the current BPEL testing approaches are shown in Table 1. More detailed results can be found in [11].

The most of the authors propose to transform the BPEL process into intermediary model using CFG, HPN, etc. in order to find the executable paths of the process and generate test cases. Some of the approaches do not cover all BPEL activities during transformation. That is why the table has columns that show which approaches consider event handling (EH) and fault handling (FH) BPEL activities. The forth column of the table, called Automated (A), shows which approaches are implemented as tools or frameworks that are ready to use by testers. The next table column, named Experimental results (ER) indicates which of the approaches are proved via case studies, experimental results, etc. Only one of the approaches does not provide test case generation (TCG). This can be seen from column before the last one. And finally, the last column shows which of the approaches supports negative testing (NT).

3. TASSA Tools

The TASSA framework consists of several tools that can be used jointly to achieve end-to-end testing of BPEL processes. The architecture of TASSA framework is presented on Figure 1.

In this section the cooperation of TASSA tools with the focus of *Fault Injection* and *Isolation tools* is presented. The main task of *Fault Injection tool*, called *faultInjector*, is to simulate faults during message exchange in order to generate negative test cases. The possible situations that are simulated are (1) overload of the communication channel that leads to delay of sending or receiving a message, (2) failure of the communication channel that leads to impossibility of sending or receiving a message, (3) noise in communication channel that leads to receiving a message with syntax and structure errors, and (4) wrong business logic of particular web service that leads to sending or receiving a message with syntax errors in its data. *faultInjector* takes as input a BPEL process under test, a list with failure parameters that describes the above situations and a string with values, which correspond to the arguments of the activity causing the failure. It returns a transformed BPEL process with simulated failure. The fault injection process consists of the following steps:

– identification of message exchanged when the failure is simulated,
– modification of communication channel, so that the failures expected by the tester occur,
– modification of an activity that corresponds to the message in order to send message to the proxy created between the message sender and receiver,
– serialization of input arguments of the real receiver (marshalling),
– invocation of the proxy,
– deserialization of output arguments and sending to the real receiver (unmarshalling).

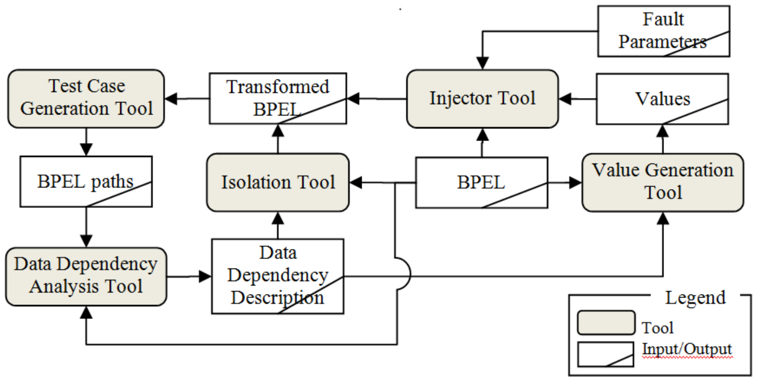

Figure 1. Architecture of TASSA framework

Similar steps are performed for the response of the invocation.

The formal representation of the process of marshalling and unmarshalling is as follows:

$o = \text{invoke}(i_1, i_2, ..., i_n)$
$o = \text{Unmarshal}(\text{ProxyInvoke}$
$(\text{Marshal}(i_1, i_2, ..., i_n), R))$

where $i_1, i_2, ..., i_n$ are the real arguments of the modified Invoke activity, o is the original output data, *Marshal* and *Unmarshal* are the embedded BPEL functions for marshalling and unmarshalling, and *ProxyInvoke* is the call of the proxy with failure parameters specified by R.

The proposed approach is applicable only to invoke activities because their corresponding exchange of the messages is initiated by the BPEL process. It is necessary condition for the realization of the approach because activities for marshalling and unmarshalling need to be placed round the initiator of the message exchange.

The values passed to faultInjector are generated from a tool, called *Value Generation Tool* (VGT). Its goal is to generate valid values for all field of a given variable defined with XML Schema Definition (XSD). The main functionality of VGT is provided by a tool, called WS-TAXI, which is developed by a research team of Software Engineering Research Laboratory at the ISTI - Istituto di Scienza e Tecnologie dell'Informazione A.Faedo in Pisa. WS-TAXI generates compliant XML instances from a given XML Schema by using well-known Category Partition technique [12]. VGT takes as input a BPEL process under test and an array with identifiers of variables, whose values need to be generated.

The array of variables is produced by a tool, called *Data Dependency Analysis Tool* (DDAT) [2]. For a given path of the BPEL process DDAT finds all conditional activities along the path and specifies which variables affect those conditional activities. It receives as input a BPEL process and an array of unique identifiers of activities, describing the path that the BPEL process needs to follow. The path is generated by a tool, called Test Case Generation Tool (TCGT). TCGT solves two tasks. Its first task is to identify all paths of a given BPEL process in order to assist the tester in the process of test case generation. The second task of TCGT is to ensure management capabilities and storage for test cases.

The output of DDAT is also needed for a tool, named *Isolation Tool* (IsT). IsT provides temporary removal of BPEL process dependencies from one or more external web services. This allows the tester to control the web service returned results and pre-determine the possible routines in the BPEL process, as well as to continue testing even if a particular web service is missing. The BPEL process's dependency upon external services can be described as follows:

- synchronous execution of operation provided by an external service (Invoke activity in the BPEL process description),
- asynchronous execution of operation provided by an external Service (combination of Invoke and Receive activity in the BPEL process description),
- unforced message receipt from external service (Pick activity),

Table 2. Replacement of the BPEL process activities

Original Activity	Replacement Activity
Synchronous Invoke	Assign
Asynchronous Invoke	Empty
Receive	Assign
Reply	Empty
Pick/OnAlarm	Wait and OnAlarm branch
Pick/OnMessage	Assign and OnMessage branch
HumanTask	Invoke

– sending message to external service (resulting from an ingoing message),
– HumanTask activity, which requires human intervention and which affects the application through its output data (operator-entered values).

Invoke activity is modeled with following expression:

$o = f(i_1, i_2, ..., i_n, R)$

Herein the letter f denotes the functionality of the operation provided by the external service, $i_1, i_2, ..., i_n$ are the input parameters of the operation, o is the returned result, and R is additional parameters of the activity not directly related to the operation execution.

To eliminate the dependency upon f the following modifications are necessary to isolate the BPEL process from operation 1:

– Modification of the process, where the relevant Invoke activity is replaced with Assign activity to assign the output variable o specific values set by the user;
– When isolating the process from one activity there is created a test artifact (a variant of the BPEL process, in which the Invoke activity is replaced by an Assign activity).

The other dependencies are handled in a similar way, e.g. in the asynchronous mode for operation call (Invoke and Receive), the Invoke activity is replaced by the Empty activity (as it does not influence it) and Receive activity is replaced by Assign activity. Table 2 illustrates the mechanisms for isolation of the process from the different dependencies.

Through the cooperation of the above described tools the automation of the functional testing is largely achieved. Furthermore, lack of functionality for automation of testing in conditions of poor or unavailable communication channels with remote services is to a great extent overcome. Automation of negative testing is also supported.

4. Application of Fault Injection and Isolation Techniques in TASSA Framework

This section presents the interoperability between faultInjector and IsT of TASSA framework. The tools are verified through testing of sample BPEL process, called Order Data Verifier Business Process (ODVBP). The process consists of four web services that are described in Table 3.

ODVBP uses the web services presented in Table 3 to validate the clients order data, namely email, credit card number and Zip code. It also retrieves the state abbreviation form Zip code and converts the total amount of the order into appropriate currency according to current rate.

Listing 1 shows an invoke activity, called CardValidatorInvoke, that is responsible for invocation of web service for validation of client credit card number.

```
<invoke name="CardValidatorInvoke"
    partnerLink="CardValidatorPartner"
    operation="Validate_CreditCard"
    xmlns:tns="http://www.Softwaremaker.Net/
    WebServices/" portType=
    "tns:ValidatorSoap"
    inputVariable="Validate_CreditCardIn"
    outputVariable="Validate_CreditCardOut">
</invoke>
```

Listing 2 shows transformation of the above activity after execution of faultInjector and IsT of TASSA framework.

```
<assign name="Assign1">
  <copy><from>
```

Table 3. Web services called from Order Data Verifier Business Process

Web service	Description
Email Validator	Validates email addresses for client applications
Credit Card Validator	Validated credit card number and type
Currency Convertor	Get conversion rate from one currency to another currency
Zip Code Validator	Validate Zip code and returns USA state abbreviation, latitude (decimal degrees) and longitude (decimal degrees)

```
      sxxf:doMarshal($Validate_CreditCardIn.parameters)
    </from><to>
      $ProxyInvokeOperationIn.operationIn/tassaP:part1
    </to></copy>
  <copy><from>
      'http://www.softwaremaker.net/webservices/
      swm/validator/validator.asmx?WSDL'
    </from><to>
      $ProxyInvokeOperationIn.operationIn/
      tassaP:endpoint
    </to></copy>
  <copy><from>20</from><to>
      $ProxyInvokeOperationIn.operationIn/tassaP:wait
    </to></copy>
  <copy><from>0</from><to>
      $ProxyInvokeOperationIn.operationIn/
      tassaP:errorsFactor
    </to></copy>
</assign>
<invoke xmlns:tns="http://www.rila.com/tassa/ProxyInvoke"
      inputVariable="ProxyInvokeOperationIn"
      name="ZipCodeInvoke"
      operation="ProxyInvokeOperation"
      outputVariable="ProxyInvokeOperationOut"
      partnerLink="PartnerLink1"
      portType="tns:ProxyInvokePortType"/>
<assign name="Assign2">
  <copy>
    <from>
      sxxf:doUnMarshal($ProxyInvokeOperationOut.part2)
    </from>
    <to part="parameters"
      variable="Validate_CreditCardOut"/>
  </copy>
</assign>
```

In order to generate transformed BPEL process *faultInjector* and *IsT* need a configuration information that describes the simulated failures as follows:

- Wait interval: an integer value that defines the delay of message seconds in seconds;
- Error factor: an integer value that that defines the kind of error will be injected (1–100: insert random errors in the data, which would possible break the XML structure; 0: usually used with Wait interval to delay the message; −1: replace the original values in the message; −2: interrupt the message);
- End point address: an end point address of the partner web service;
- Activity variables: input and output variables of the activity that will be injected.

As can be seen from Listing 1 and Listing 2, the Invoke activity, named CardValidatorInvoke, is enclosed with two additional Assign activities. The first Assign activity initializes the input parameters of ProxyInvoke operation of faultInjector. The parameters are as follows:

- Serialized input arguments of card validator operation of Credit Card Validator web service;
- End point address of the Credit Card Validator web service;
- Wait interval initialized with 20;
- Error factor initialized with 0.

The second Assign activity copies deserialized result from invocation of ProxyInvoke operation of faultInjector to the output variable of the Credit Card Validator web service. In addition, CardValidatorInvoke activity invokes ProxyInvoke operation instead actual Credit Card Validator web service.

faultInjector is validated through four test cases that correspond to its possibility of faults generation:

- Test Case 1: Message delay,
- Test Case 2: Interruption,
- Test Case 3: Noise in the message structure,
- Test Case 4: Noise in the message data.

To prove the fault injection against normal behavior of the process first the standard use case should be observed:

Table 4. Configuration data

Data	Description
wait=20	Error factor
error_ratio=0	Wait interval
http://www.softwaremaker.net/webservices/swm/validator/validator.asmx? WSDL	End point address
$Validate_CreditCardIn.parameters=$Validate_CreditCardOut.parameters	Activity variables

Table 5. Expected outputs from test cases

Test case	Description
Test case 0	Meaningful, well formed message that is executed in time interval t_i.
Test case 1	Meaningful, well formed message that is executed in time interval $t_i + T$, where T is the delay given as a failure parameter.
Test case 2	Error message, because of interruption
Test case 3	Error message, because of wrong structure
Test case 4	Well formed message with a random or invalid data

Table 6. Input data of Order Data Verifier Business Process

Test data	Test data values	Remark
Input1	<ord1:firstname>John</ord1:firstname> ... <ord1:currencycode>EUR</ord1:currencycode>	Valid data
Input2	<ord1:creditcardnumber>5374439468966228000 </ord1:creditcardnumber>	Invalid credit card number
Input3	<ord1:email>dessislava.g.petrovagmail.com</ord1:email>	Invalid email
Input4	<ord1:postalcode>070930</ord1:postalcode>	Invalid zip code

– Test Case 0: No fault injection (normal behavior).

The expected outputs from each test case are described in Table 5.

Test data are generated according to the XSD schema of ODVBP. They are presented in Table 6.

Table 7 presents the possible output results from execution of ODVBP.

Test case 0 includes tests representing normal behavior of the BPEL process. The tests correspond to the input data presented in Table 6, namely Input 1, Input 2, Input 3 and Input 4. The results form test executions are respectively Output 1, Output 2, Output 3 and Output 4. All tests are performed approximately for about 5.714 seconds.

The results from execution of the rest test case are presented in Table 8, Table 9, Table 10 and Table 11. As can be seen from the tables, all tests are passed, which means that the faultInjector successfully detects the faults injected in the business process.

In Test Case 1 (Message delay) the output data is the same as in Test Case 0 (Normal behavior), but the execution time is longer due to simulated delay. It is obvious that the execution time of Test Case1 differs from the execution time of Test Case 0 in the delay given as a failure parameter. For example, the Wait interval of the test for Validate_CreditCard operation is 10 s. Therefore, the execution time of this test is equal to the execution time of corresponding test in Test Case 0 plus 10 s.

During execution of Test Case 2 and Test Case 3, the BPEL process fails, due to impossibility of sending or receiving a message as well as corruption of message data. These faults are not handled by the sample BPEL process, so here the negative test cases catch a bug in the business logic, which is the main idea of testing BPEL processes against fault injections.

In Test Case 4 (Noise in the message data) faultInjector simulates wrong business logic of a web service by corruption of the received message with invalid data. The first test of this test case shows that this leads to wrong data values comparing with the expected ones in Test Case 0, or even to incorrect workflow.

Table 7. Output data of Order Data Verifier Business Process

Result data	Test data values
Output1	<ns0:firstname>John</ns0:firstname> . . . <ns0:total>141.74</ns0:total> <ns0: currencycode>EUR</ns0:currencycode> <ns0:ordererrors>Validation is successful.</ns0:ordererrors>
Output2	<ns0: ordererrors >ERROR: Invalid Credit Card</ns0:ordererrors>
Output3	<ns0: ordererrors >ERROR: Invalid email</ns0:ordererrors>
Output4	<ns0: ordererrors >ERROR: Invalid Zip Code</ns0:ordererrors>
Output5	exMessage: disconnected
Output6	BPCOR-6130: Activity Name is CardValidatorInvoke Caused by: javax.xml.soap.SOAPException: javax.xml.stream. . . Message: An invalid XML character was found in the element content of the document
Output7	BPCOR-6130: Activity Name is EmailInvoke Caused by: javax.xml.soap.SOAPException: javax.xml.stream. . . Message: An invalid XML character was found in the element content of the document
Output8	BPCOR-6130: Activity Name is ZipCodeInvoke Caused by: javax.xml.soap.SOAPException: javax.xml.stream. . . Message: An invalid XML character was found in the element content of the document

Table 8. Test Case 1 execution results

Wait interval	Error factor	Operation	Input	Output	Test result	Execution time (s)
wait=10	error_ratio=0	Validate_CreditCard	Input1	Output1	Passed	15.714 s
wait=20	error_ratio=0	ValidateEmail	Input2	Output2	Passed	25.714 s
wait=15	error_ratio=0	ConversionRate	Input3	Output3	Passed	20.714 s
wait=20	error_ratio=0	ValidateZip	Input4	Output4	Passed	25.714 s
wait=10, wait=10	error_ratio=0, error_ratio=0	ConversionRate, ValidateZip	Input1	Output1	Passed	25.714 s
wait=10, wait=15	error_ratio=0, error_ratio=0	Validate_CreditCard, ValidateEmail	Input1	Output1	Passed	30.714 s

Table 9. Test Case 2 execution results

Wait interval	Error factor	Operation	Input	Output	Test result	Execution time (s)
wait=0	error_ratio=-2	Validate_CreditCard	Input1	Output5	Passed	8.425 s
wait=0	error_ratio=-2	ValidateEmail	Input2	Output5	Passed	8.145 s
wait=0	error_ratio=-2	ConversionRate	Input3	Output5	Passed	8.412 s
wait=0	error_ratio=-2	ValidateZip	Input4	Output5	Passed	8.345 s

Table 10. Test Case 3 execution results

Wait interval	Error factor	Operation	Input	Output	Test result	Execution time (s)
wait=0	error_ratio=40	Validate_CreditCard	Input2	Output6	Passed	5.194 s
wait=0	error_ratio=40	ValidateEmail	Input3	Output7	Passed	6.594 s
wait=0	error_ratio=40	ValidateZip	Input4	Output8	Passed	7.194 s

Table 11. Test Case 4 execution results

Wait interval	Error factor	Operation	Input	Output	Test result	Execution time (s)
wait=0	error_ratio=-1	ValidateEmail	Input2	Output1	Passed	4.086 s
wait=0	error_ratio=-1	ConversionRate	Input2	Output2	Passed	6.411 s
wait=0	error_ratio=-1	ValidateZip	Input2	Output2	Passed	4.014 s

The obtained results show that faultInjector is suitable for generation of different types of faults, which leads to the possibility of using TASSA framework for negative testing of BPEL processes.

5. Conclusion

This paper presents TASSA Framework, which offers approach for web service compositions testing by automating the testing process. Currently, it supports only design time testing, which is provided by five tools, composed as services itself, which integrated together offer to developers and service integrators the complete environment for functional testing of BPEL processes. Through integration of faultInjector tool in TASSA framework the negative testing is also achieved. This allows testing of BPEL processes in conditions of poor or unavailable communication channels with remote services. Furthermore, the testing process can be performed in isolation of external partner web services of the BPEL process under test due to possibility of Isolation Tool to remove dependency from them. The experimental results from validation of the testing approach through simple business process are presented.

Our future work will concentrate on developing complete testing methodology and validation of all TASSA framework tools over more complex BPEL processes.

Acknowledgment

The authors acknowledge the financial support by the National Scientific Fund, BMEY, grant agreement no. DO02-182 and SISTER project, funded by the European Commission in FP7-SP4 Capacities via agreement no. 205030.

References

[1] L. Dong, H. Yu, and Y. Zhang, “Testing BPEL-based web service composition using high-level Petri Nets,” in *Proceedings of the IEEE International Enterprise Distributed Object Computing Conference*, 2006, pp. 441–444.

[2] I. Spassov, D. Petrova-Antonova, V. Pavlov, and S. Ilieva, “DDAT: Data dependency analysis tool for web service business processes,” in *Second International Workshop on Software Quality SQ*, June 2011, pp. 232–243.

[3] T. Lertphumpanya and T. Senivongse, “Basis path test suite and testing process for WS-BPEL,” *WSEAS Transactions on Computers*, Vol. 7, No. 5, 2008, pp. 483–496.

[4] J. Li, H. Tan, H. Liu, J. Zhu, and N. Mitsumori, “Business-process-driven gray-box SOA testing,” *IBM Systems Journal*, Vol. 47, 2008, pp. 457–472.

[5] M. Karam, H. Safa, and H. Artail, “An abstract workflow-based framework for testing composed web services,” in *International Conference on Computer Systems and Applications (AICCSA)*, 2007, pp. 901–908.

[6] Q. Yuan, J. Wu, C. Liu, and L. Zhang, “A model driven approach toward business process test case generation,” in *10th International Symposium on Web Site Evolution (WSE)*, 2008, pp. 41–44.

[7] Y. Zheng, J. Zhou, and P. Krause, “An automatic test case generation framework for web services,” *Journal of Software*, Vol. 2, No. 3, 2007, pp. 64–77.

[8] J. Li and W. Sun, “BPEL-Unit: JUnit for BPEL processes,” in *Service-Oriented Computing, ICSOC*, 2006, pp. 415–426.

[9] P. Mayer and D. Lubke, “Towards a BPEL unit testing framework,” in *Proceedings of the workshop on Testing, analysis, and verification of web services and applications*, 2006, pp. 33–42.

[10] J. Yan, Z. Li, Y. Yuan, W. Sun, and J. Zhang, “BPEL4WS unit testing: Test case generation using a concurrent path analysis approach,” in *Proc. of ISSRE. IEEE Computer Society*, 2006, pp. 75–84.

[11] D. Petrova-Antonova, I. Krasteva, and S. Ilieva, “Approaches facilitating WS-BPEL testing,” in *17th Conference on European Systems and Software Process Improvement and Innovation*, September 2010, pp. 5.1–5.17.

[12] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, “WS-TAXI: A WSDL-based testing tool for web services,” in *International Conference on Software Testing Verification and Validation*, 2009, pp. 326–335.

e-Informatica Software Engineering Journal, Volume 6, Issue 1, 2012, pages: 71–77, DOI 10.5277/e-Inf120106

On Principles of Software Engineering – Role of the Inductive Inference

Ladislav Samuelis*
*Faculty of Electrical Engineering and Informatics, Technical University of Košice
Ladislav.Samuelis@tuke.sk

Abstract

This paper highlights the role of the inductive inference principle in software engineering. It takes the challenge to settle differences and to confront the ideas behind the usual software engineering concepts. We focus on the inductive inference mechanism's role behind the automatic program construction activities and software evolution. We believe that the revision of rather ln old ideas in the new context of software engineering could enhance our endeavour and that is why deserves more attention.

1. Introduction – Beyond Software, Beyond Engineering

Before getting into details of the role of inductive inference in software engineering, we make some explanatory notes on the notions of *software* and *engineering* as it appears in the title of this contribution.

We can observe plenty of interpretations of the software and software engineering throughout the history of computing. Searching the origin of the word *software* F.R. Shapiro states that it appears for the first time in the work of John W. Tukey. He used that term in the context of computing in an article of the American Mathematical Monthly, in 1958! The quote is as follows [1, p. 1]:

> Today the 'software' comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its hardware of tubes, transistors, wires, tapes, and the like.

In 2008, the work of Leon J. Osterweil [2] appears. He suggests that there may exist other types of software besides computer software. He identifies parallels between computer software and other societal artefacts as laws, processes, recipes, instructions, and suggests that there are similar parallels in the ways, in which these artefacts are built and evolved.

There are 50 years between these two above-mentioned interpretations of the word *software*. The 'semantic gap' between the ideas that is behind this word has been widened enormously in time. The notion has gained more specific meanings during its life-span and it is expected to keep narrowing down. There are no signs that a rather 'calm' period of software 'evolution' would come. On the contrary, as we may conclude from the recent pervasively distributed and service-oriented software development.

Recently there has been an incredible increase in the performance of hardware. This increase itself is the reason for the incredible growth of software complexity. The Wirth's, or rather Reiser's 'law': *Software is getting slower, faster than hardware is getting faster* [3], allegorically points to the same fact.

After revealing partially the roots of the notion of software, we may focus on the notion of *engineering*. The question is as follows:

What feature of software enables us 'to engineer' it?

We can find a comprehensive answer to this question in the work of Wei-Lung Wang [4]:

> The key reason is that software is a tangible form of mathematics that lends itself to being engineered. At its core, a program is an abstract sequence of instructions that performs some computation. But when the program is realized on a computer, it becomes an information tool with its own use-feedback cycle. It changes from an ethereal entity to a tangible tool and its actions can be observed. Instead of mathematically proving the results of a program, we can simply run it on some sets of inputs and observe its behaviour. This tangibility (or 'executability') is both software's strength and Achilles heel.

We accept that this specific feature of *observability in the material world* is the essence of the software *engineering.* In other words, this *tangibility* or *executability* enables specific ways of experimentation, which is the basis for observing the behaviour of the software by perception. In this way, we may create highly complex models that can be fully mapped into a computer representation and this model can be 'executed'.

Experiences gained from these observations also trigger needs for deeper understanding of the implemented ideas. We may say that observation by perception supports the comprehension of the modelled reality. This opposite process is *program comprehension*; when the task is to 'understand' (or to gain a mental image of) the computer model from the implemented code.

Of course, there also exist attempts that search a single unified theory of software engineering. For example, the contribution of P. Johnson and M. Ekstedt, presented at the recent International Conference on Software Engineering Advances [5], outlines the requirements for such a unified theory.

The statement of M. Jackson [6] is against a unified theory. He says that software engineering is a clumsy notion. He supports this observation with the fact that software engineering is split into various topics (e.g, compiler engineering, operating systems, database engineering, etc.) and in this way it does not cover any knowledge gained from solved problems. In other words, software engineering is an abstraction and every successful area of software engineering immediately changes to set up an independent specialization.

A recent article from M.S. Mahoney [7] characterizes the history of software engineering in the following way:

> Historians and software engineers are both looking for a history of software engineering. For historians, it is a matter of finding a point of perspective from which to view an enterprise that is still in the process of defining itself. For software engineers, it is the question of finding a usable past, as they have sought to ground their vision of the enterprise on historical models taken from science, engineering, industry, and other professions.

These contradictory views and motivations of experts outline the broad diversity of research and ought to remind us to be careful with applying theory for building software.

2. The Role of the Inductive Inference in Automatic Program Construction

Automatic program construction has been a goal of the first programmer who faced with difficulties of programming. This activity has been spreading over the history of software engineering with various intensity and it acts like a moving target that constantly shifts in order to reflect changing requirements.

Much of what was originally conceived of as automatic programming was achieved a long time ago. In 1958 this term was mentioned for the first time in connection with the compiler construction. On the other hand, current expectations regarding its potential are often based on an idealized view of reality and some of them probably cannot be met. Nevertheless, a number of important developments are in progress in research efforts and in commercially available systems.

The term *automatic program construction* (or *program synthesis*) is used to refer to the study and implementation of methods for automating a significant part of the process of developing and maintaining software within the context of software life-cycle.

plus.15em A broader goal of this field is to make computer programs significantly easier by means of automation selected software development process. More specific goals include increasing software productivity, lowering costs, increasing reliability, making more complex systems tractable, and allowing users to focus more on solving problems rather than on the details of implementation.

The big challenges for automatic program construction are defined, for example, by L. McLaughlin [8] in the following manner:

– to produce good runtime performance;
– to produce code that someone can look at, deal with, and understand;
– to ensure the code that is provably correct.

Every point deserves its own 'science' and the emergent research fields on automatic program synthesis follow roughly these criteria too.

The freedom of language selection allows using more declarative and less procedural specification. In other words, the specification is closer to the *what* (end-user defined) and the implementation is closer to the *how* end of the spectrum. A more technical characterization of the 'gap' between specification and implementation is that there is less detailed information content in the specification than in the implementation. The program synthesis process consists of filling in this gap with details that are omitted from the specification.

The automated synthesis of programs has its roots in artificial intelligence too. It is interesting to observe the mutual influence and synergy of ideas stemming from the field of software engineering and artificial intelligence. In particular, the task of inference of grammars from pattern analyses triggered the research on programming by examples [9].

In general, we distinguish two main methods in software development. *Deductive* methods – deduction is basically a problem of searching for an inference path from some initial set of facts to a goal fact. This fundamental mechanism is behind the deductive approach to automatic programming. In principle any method of automated deduction can be used to support automatic programming. For example, programming language PROLOG [10] (in fact, its inference engine) represents a deductive system. Despite its limitations, PROLOG remains among the most popular languages today, with many free and commercial implementations available. One of the challenges in research is to combine automated deduction with other methods.

Inductive methods – inductive inference is based in building models on the basis of experienced facts. Let us suppose that we have a mental model at our disposal. In the next step we may apply the standard scientific approach, which consists of *finding metrics*, *finding other descriptions* (or refinement of the model) and based on these new models *to infer the future behaviour* of the observed object. These three mental activities are, in fact, the inductive inference activities. In other words, finding metrics (or measurable features, data, observable signs, etc. on the model or on its output data) is indispensable for the description (or modelling). This activity is based on collecting individual data in order to create a hypothesis (model) with the process of generalization. We create model not only for the description of the actual systems but also for prediction. It means that the inferred model can serve for the prediction the system's behaviour in the future. An example of the application of the inductive inference is the 'programming by examples' where the examples serve for building models or rules (in this specific case a grammar).

Many areas exist where demonstration of an example is a suitable tool for automating tasks. For example, paths of robots represent linear plans and the task is to construct program or the sequence of learning objects represent the progress of the student in the learning material and the task is to construct the navigation plan (learning by watching or incremental learning). Programming by examples roots in the incremental learning, which was elaborated in 1970s and

1980s [11]. The structures of the systems devoted to synthesis of programs by examples are similar to the structure of linguistic pattern recognition systems [11].

It is evident that during the construction of the final model, a new instruction of the example could completely modify the existing model (due to the inductive inference principle). This fact deserves special awareness in the application of inductive inference.

In summary, knowledge of the part, which may be represented at whatever level of granularity, serve for the development of rules. In fact these rules may serve as basis for a new deductive system. For example, knowledge condensed in software design patterns represent higher level granularity of knowledge and these artefacts may also serve for building new system.

We may conclude that automatic program construction during the 1980s was more or less about the optimization of loops. In other words, the discovery and the implementation of reusable code segments was in fact the discovery of loops.

3. Notes on Software Evolution

In the 1950s, the term *automatic computing* referred to almost any work related with a computer. We tend to forget that before the object-oriented approach the methodology of automatic program construction was also associated with the idea of 'construction of programs by examples'.

The research on automating programming before object-orientation was influenced to a great extent by results gained in artificial intelligence research ([12], and [13]).

There are plenty of articles and a special IEEE conference [14] devoted to software evolution. Research on software evolution is discussed in many software related disciplines. In any case, software evolution is equal to comprehension. This idea is briefly expressed by M. Jazayeri [15] as:

> Not the software itself evolves, but our understanding and the comprehension of the reality.

This is in compliance with the idea that our understanding of the domain problem incrementally evolves and learning is an indispensable part of program comprehension.

The complexity of understanding and maintaining a program is proportional to its size and complexity. F. Brooks in the well-known paper 'No Silver Bullet' [16], argues that programming is inherently complex. Whether we use a machine language or a high-level programming language, in general, we cannot simplify a program below a certain threshold that he calls an 'essential program complexity'. The two main factors that determine this complexity threshold are:

– the complexity of a problem and its solution at the conceptual level, and
– the complexity of the infrastructure and the environment which has to be taken into account when solving problems by a program at operational level.

These two factors cannot be clearly separated from each other in program components, which constrains our efforts in using conventional decomposition ('divide and conquer') in combating software complexity. Very often, we cannot create a software conform like LEGO models unlike hardware constructions [17]. This difficulty of achieving clean decomposability is an important difference between hardware and software. Another important difference is that we do not try to change hardware so often and in such radical ways as we do software. The difficulty of clearly separating concerns by means of decomposability necessarily hinders changeability.

The inductive inference (or incrementality) appears in various contexts in the relatively short history of software engineering. This fact seems natural because software development processes like comprehension, design, refinement and realization are done iteratively and incrementally in practice. Due to this common fact incrementality notion is applied superficially in software engineering literature. We note that programming cannot be *fully* automated, since a computer must at least be told what to do. Only a human being is able to create ideas and tell it what to do. It is hoped that as technology progresses, the required details on how to do the task will steadily decrease.

4. The Ubiquity of the Inductive Inference

The following remarks show the wide range of the usage of incrementality notion. A brief historical overview of the 'Incremental and Iterative Development' is presented in the work of C. Larman and V. Basili [18]. This work summarizes the role of the iterative and incremental software development through significant software projects since the mid of 1950s. It focuses on the incrementality utility, applied in software engineering processes, from the managerial point of view. It describes the driving thoughts and misbelieves, which were behind the practices applied in the past decades in the field of software engineering.

We accept in general that comprehension is also a continuous iterative and incremental process. The fact that problem solving does not progress in a linear manner from one activity to the next is highlighted as the conjecture:

Empirically based models mature from understanding to explaining and predicting capability.

This conjecture is explained in the handbook of authors A. Endres and D. Rombach [19, p.273], which is devoted to the empirical aspects of software engineering.

Inductive inference plays an important role in practical software engineering. At present time the incremental change in object-oriented programs are in focus (for example [20]). These activities investigate the impact of adding new functionalities into the code and finding the relevant program dependencies. Incrementality is important in software visualization too [21], where the aim is to get a better comprehension of the software behaviour by representing complex structures graphically.

The objective of the software development is to model a certain aspect or abstraction of reality as stated by B. Meyer in his work 'Reality: a cousin twice removed' [22]. Software engineering, as every engineering discipline, is characterized by trials and errors, which are necessary steps for clarifying the comprehension of the requirements, design and implementation. On the other hand we have to note that the incrementality principle has its mathematical roots and is explained in the theory of inductive inference [23]. Incremental software development is sometimes called 'build a little, test a little'. We can see the similarity between building concepts and models in software engineering and building hypotheses in mathematics. This process is very clearly highlighted in Pólya's classic work, 'How to Solve It' [24].

The empirical evidence from the real-world software suggests that learning or incremental program development is possible only when the data are presented incrementally. For instance, programming languages dispose with constructs, which help postpone solving some issues. A good example is the exception mechanism in object-oriented programming. This process makes, of course, the software more complex and drifts away from the original design. These facts may lower the quality of the software but it is the task of the validation and verification to ensure the formal quality software.

To sum up, the inductive inference is ubiquitous in software engineering. With each step we discover new requirements, analyse, plan, implement and test them. Every iteration adds new insights and the system grows or logically clarifies this way. In other words, software programs are too complex for getting correct details on any artefacts without some amount of experimentation. The software developers' ideas evolve as they progress.

5. Conclusion

This paper looks into the question: *What is the role of inductive inference in software engineering?* Its specific aim is to highlight the hidden role of inductive inference phenomenon behind the wide variety of software engineering concepts. It opens with automatic program construction and proceeds to the software evolution concept. Analysing the inductive inference in a sterile environment is not unusual. We take the challenge to settle out differences and confront the ideas behind the usual software engineering concepts in a turbulent and impure environment of recent software engineering activities. This approach might help in developing a more condensed foresight and provoke constructive thinking. We know that we did not invent a new solution to an existing problem; but we rather revised old ideas and analysed them in recently applied software engineering practices. Practice generates always new problems and the task of software engineers is permanent strive for the identification essential concepts as it is expressed in the Semat initiative [25].

Acknowledgement

This work is supported by the Slovak Scientific Grant Agency: KEGA 040TUKE-4/2011: Modern software engineering in education – structure and implementation of software engineering subjects for university informatics study programmes.

References

[1] F. S. Preston, F. R. Shapiro, and L. R. Johnson, "Comments, queries, and debate," *IEEE Annals of the History of Computing*, Vol. 22, No. 2, 2000, pp. 69–71.

[2] L. J. Osterweil, "What is software?" *Automated Software Engineering*, Vol. 15, No. 3, 2008, pp. 261–273.

[3] N. Wirth, "A plea for lean software," *IEEE Computer*, Vol. 28, No. 2, 2006, pp. 64–68.

[4] W. Wei-Lung, "Beware the engineering metaphor," *Commun. ACM*, Vol. 45, No. 5, 2002, pp. 27–29.

[5] P. Johnson and M. Eckstedt, "In search of a unified theory of software engineering," in *International Conference on Software Engineering Advances, ICSEA*, 2007, p. 5.

[6] M. Jackson, "Will there ever be software engineering?" *IEEE Software*, Vol. 15, No. 1, 1998, pp. 36–39.

[7] M. S. Mahoney, "Finding a history for software engineering," *IEEE Annals of the History of Computing*, Vol. 26, No. 1, 2004, pp. 8–19.

[8] L. McLaughlin, "Automated programming: The next wave of developer power tools," *IEEE Software*, Vol. 23, No. 3, 2006, pp. 91–93.

[9] H. Liebermann, Ed., *Your Wish is My Command- Programming by Example, Automatic programming, Encyclopedia of Artificial Intelligence.* Morgan Kaufmann/San Francisco, February 2001.

[10] R. A. Kowalski, "The early years of logic programming," *Commun. ACM*, Vol. 31, No. 1, 1988, pp. 38–43.

[11] A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, Eds., *Watch What I Do: Programming by Demonstration.* Cambridge University Press, 1993, ch. Programming by demonstration.

[12] J. S. Poulin, "Technical opinion: reuse: been there, done that," *Commun. ACM*, Vol. 42, No. 5, 1999, pp. 98–100.

[13] Z. Manna and R. J. Waldinger, "Toward automatic program synthesis," *Commun. ACM*, Vol. 14, No. 3, 1971, pp. 151–165.

[14] ACM, "IWPSE '01: Proceedings of the 4th international workshop on principles of software evolution," New York, NY, USA, 2001, conference Chair-Tamai, Tetsuo.

[15] M. Jazayeri, "Species evolve, individuals age," in *International Workshop on Principles of Software Evolution.* ACM, September 5–6, Lisbon 2005.

[16] J. F. P. Brooks, "No silver bullet essence and accidents of software engineering," *Computer*, Vol. 20, No. 4, April 1987, pp. 10–19.

[17] A. Ran, "Software isn't built from lego blocks," in *ACM Symposium on Software Reusability*, 1999, pp. 164–169.

[18] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*, Vol. 36, No. 6, June 2003, pp. 47–56.

[19] A. Endres and D. Rombach, *A Handbook of Software and Systems Engineering; Empirical observations, laws and theories.* Pearson, Addison Wesley, 2003.

[20] V. Rajlich and P. Gosavi, "Incremental change in object-oriented programming," *IEEE Software*, Vol. 21, No. 4, July/August 2004, pp. 62–69.

[21] C. Knight and M. Munro, "Visual information-amplifying and foraging," in *Proceedings of SPIE, San Jose, USA, volume 4032. International Society for Optical Engineering*, January 2001, pp. 88–98.

[22] B. Meyer, “Reality: A cousin twice removed,” *IEEE Computer*, Vol. 29, No. 7, July 1996, pp. 96–97.

[23] D. Angluin and C. H. Smith, “Inductive inference: Theory and methods,” *Computing Surveys*, Vol. 15, No. 3, September 1983, pp. 283–269.

[24] G. Pólya, *How to solve it: A New Aspect of Mathematical Method*, 2nd ed. Princeton University Press, 1957.

[25] I. Jacobson, “Discover the essence of software engineering,” *CSI Communications*, July 2011, pp. 12–14.

e-Informatica Software Engineering Journal (EISEJ) is an international, open access, peer-reviewed journal that concerns theoretical and practical issues pertaining development of software systems. Our aim is to focus on experimentation and data mining in software engineering.

The purpose of **e-Informatica Software Engineering Journal** is to publish original and significant results in all areas of software engineering research.

The scope of **e-Informatica Software Engineering Journal** includes methodologies, practices, architectures, technologies and tools used in processes along the software development lifecycle, but particular stress is laid on empirical evaluation.

e-Informatica Software Engineering Journal is published online and in hard copy form. The online version (which is our primary version) is open access, which means it is available at no charge to the public.

Topics of interest include, but are not restricted to:

— Software requirements engineering and modeling
— Software architectures and design
— Software components and reuse
— Software testing, analysis and verification
— Agile software development methodologies and practices
— Model driven development
— Software quality
— Software measurement and metrics
— Reverse engineering and software maintenance
— Empirical and experimental studies in software engineering (incl. replications)
— Evidence based software engineering
— Systematic reviews and mapping studies
— Meta-analyses
— Object-oriented software development
— Aspect-oriented software development
— Software tools, containers, frameworks and development environments
— Formal methods in software engineering.
— Internet software systems development
— Dependability of software systems
— Human-computer interaction
— AI and knowledge based software engineering
— Data mining in software engineering
— Prediction models in software engineering
— Tools for software researchers or practitioners
— Project management
— Software products and process improvement and measurement programs
— Process maturity models
— Search-based software engineering

Papers can be rejected administratively without undergoing review for a variety reasons, such as being out of scope, being badly presented to such an extent as to prevent review, missing some fundamental components of research such as the articulation of a research problem, a clear statement of the contribution and research methods via structured abstract or the evaluation of the proposed solution (empirical evaluation is strongly suggested).

The submissions will be accepted for publication on the base of positive reviews done by international Editorial Board and external reviewers.

English is the only accepted publication language. To submit an article please enter our online paper submission site.

Subsequent issues of the journal will appear continuously according to the reviewed and accepted submissions.

http://www.e-informatyka.pl/wiki/e-Informatica

e-Informatica

ISSN 1897-7979