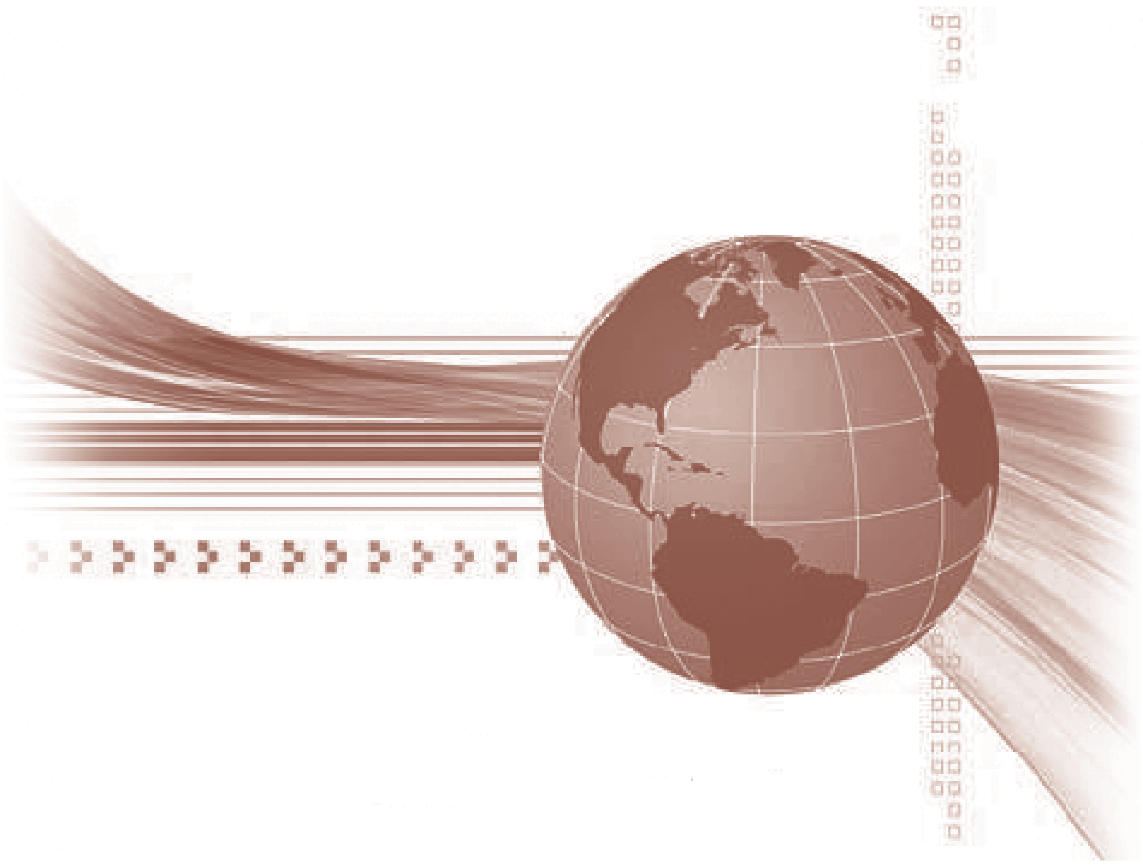
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

3/2010

Anul LV **2010**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

3

Redacţia: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

SOLVING OPTIMAL BROADCASTING STRATEGY IN METROPOLITAN MANETS USING MOCELL ALGORITHM

M. GHONAMY, A. BADR, AND ABD EL FATAH HEGAZY

Abstract. Mobile ad-hoc networks (MANETs) are a set of communicating devices that are able to spontaneously interconnect without any pre-existing infrastructure. In such a scenario, broadcasting becomes very important to the existence and the operation of this network. The process of optimizing the broadcast strategy of MANETs is a multi-objective problem with three objectives: (1) reaching as many stations as possible, (2) minimizing the network utilization and (3) reducing the broadcasting duration. The main contribution of this paper is that it tackles this problem by using multi-objective cellular genetic algorithm that is called MOCELL. MOCELL computes a Pareto front of solutions to empower a human designer with the ability to choose the preferred configuration for the network. Our results are compared with those obtained from the previous proposals used for solving the problem, a cellular multi-objective genetic algorithm which called cMOGA (the old version of MOCELL). We conclude that MOCELL outperforms cMOGA with respect to set coverage metric.

1. Introduction

Mobile ad-hoc Networks (MANETs) are composed of a set of communicating devices that are able to spontaneously interconnect without any pre-existence or operation of the network. There is no such an organization responsible for this kind of networks. Bluetooth and wifi are the most popular wireless networking technologies available. In MANET, devices communicate in a short limit and they can move while communicating. One of the main obstacles for performing efficient communication is that the topology may change quickly and unpredictably.

Received by the editors: June 2, 2010.

2010 *Mathematics Subject Classification.* 78M50.

1998 *CR Categories and Descriptors.* G.1.6 [**Mathematics of Computing**]: Numerical Analysis – *Optimization.*

Key words and phrases. Cellular Genetic Algorithm, Multi-Objective Optimization, Mobile Ad-hoc networks, Broadcasting.

The considered problem in this paper is broadcasting on a Metropolitan MANETs. Metropolitan MANETs is a subclass of MANETs which have some specific properties: Their density is heterogeneous and it is also dynamic (high density regions don't remain active full time). The considered broadcasting strategy in this work is Delay Flooding with Cumulative Neighborhood protocol (DFCN) [19].The considered three real world examples of such a network are mall environment, Metropolitan area, and highway environment. We took the previous environments into account so, instead of providing a special-purpose protocol for each environment, our suggestion lies in tuning the broadcasting process to adapt with each environment. The optimization of broadcasting process needs multi-goals to be satisfied at the same time by: (1) maximizing the number of reached devices (coverage) (2) minimizing the network usage (bandwidth) and (3) minimizing the duration of the process. This means that we are facing multi-objective optimization [5] [16].

The intended result of multi-objective optimization is not a single solution as the single-objective optimization. Rather, the goal is a set of solutions called Pareto optimal set (section 2). The goal is a set of solutions because one solution can provide the best result in one objective but another solution can provide the better results in another objective i.e: in our MOP, one solution can provide the best result in term of coverage but other solution can provide the best result in term of duration. These solutions are called non-dominated solution (Pareto optimal). When Pareto optimal plotted in the objective space, it is called Pareto front. Then, the role of the decision maker comes by choosing the most suitable solution from the Pareto front. In this paper, we investigate solving the problem of tuning some broadcasting strategy for metropolitan MANETs by using multi-objective optimization evolutionary algorithm (MOCELL).

Many evolutionary algorithms are used to solve multi-objective optimization problems. Although cellular genetic algorithm (cGA) has proved high efficiency and accuracy in solving single-objective optimization problems, a few works used genetic algorithm based on cellular population structure [20] in solving multi-objective optimization problem. The algorithm we propose is MOCELL which is presented in [2] as a new version of cMOGA (cellular Multi-Objective Genetic Algorithm).Our contribution lies in modifying MOCELL to be adapted with the nature of our problem (multiple decision variables with different data types). To the best of our knowledge, this is the first attempt to solve the broadcasting problem on MANETs by using MOCELL and the second with structured multi-objective EVs.

In order to verify the obtained results of the aforementioned algorithm MOCELL, we compared MOCELL results against CMOGA (the previous proposal used for solving our problem). But we needed to re-implement CMOGA in order to avoid the influence of the differences between the programming techniques used in this and the previous study.

The rest of the paper is organized as follows: section 2 presents a brief survey on multi-objective optimization and in section 3 we describe our problem and how the broadcasting protocol works. We present the chosen algorithm MOCELL in detail in section 4 and in section 5, we present our experiment in terms of the simulator configuration, parameters used with MOCELL, and the obtained results. We present and analyze the obtained result of comparing MOCELL results against cMOGA results in section 6.Finally; we summarize and suggest some topics for future research.

2. Multi-objective optimization

In this section we will revise some multi-objective optimization background. The concepts of multi-objective optimization, feasible region, Pareto optimality, Pareto dominance, Pareto optimal set, Pareto front, Pareto set approximation and Pareto front approximation are defined in the following subsections.

The scenario considered in this section involves an arbitrary optimization problem with p number of constrains and m objective functions which are (without loss of generality) to be maximized.All the objectives have equal weight.

Multi-objective optimization problem (MOP):
MOP can be defined as finding the vector $\overrightarrow{X^*} = [x_1^*, x_2^*, ..., x_n^*]$ which maximizes the vector function $\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_m(\overrightarrow{x})]$ where $\overrightarrow{x} = [x_1, x_2, ..., x_n]$ is the vector of decision variables. It also must satisfy the p constrains $h_i(\overrightarrow{x}), i = 1, 2, ..., p$

Feasible region
Feasible region Ω can be defined as the set of all vectors which satisfy all the constrains. Any point that belongs to the feasible region $\overrightarrow{x} \in \Omega$ is called *feasible solution*

Pareto dominance:
An objective vector $\overrightarrow{a} = (a_1, a_2, ..., a_n)$ is said to dominate $\overrightarrow{b} = (b_1, b_2, ..., b_n)$ (denoted by $\overrightarrow{a} \succ \overrightarrow{b}$) if and only if $\overrightarrow{b}$ is partially less than $\overrightarrow{a}$ i.e, $\forall i \in \{1, ..., n\}, a_i \geq b_i \wedge \exists i \in \{1, ..., n\} : a_i > b_i$ .In another word an objective vector $\overrightarrow{a}$ dominate $\overrightarrow{b}$ if no component of $\overrightarrow{a}$ is smaller than the corresponding component of $\overrightarrow{b}$ and at least one component is greater.

Pareto Optimality:
A point $\overrightarrow{x}' \in \Omega$ is Pareto optimal only if $\neg \exists \overrightarrow{x} \in X, \overrightarrow{f}(\overrightarrow{x}') \prec \overrightarrow{f}(\overrightarrow{x})$, for all $\overrightarrow{x}$ which belong to the *decision space* X, such a $\overrightarrow{x}$ that dominates $\overrightarrow{x}'$ does not exist.

Pareto optimal set:
The Pareto optimal set for a given MOP $\vec{f}(\vec{x})$ can be defined as $P^* = \{\vec{x} \in \Omega | \neg\exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \succ \vec{f}(\vec{x})\}$. In another word, none of the elements of the Pareto optimal set is dominated by others which belongs to the feasible region.
Pareto front:
The Pareto front for a given MOP $\vec{f}(\vec{x})$and its Pareto optima set P^* can be defined as $PF^* = \{\vec{f}(\vec{x}), \vec{x} \in P^*\}$.
Pareto set approximation:
Most work in the area of evolutionary multi-objective optimization has focused on the approximation of the Pareto optimal set. So we consider the outcome of our algorithm as mutually nondominated solutions, or for short Pareto set approximation.
Pareto front approximation:
To sum up, we can describe Pareto front approximation as the front of Pareto set approximation.

3. The problem

The considered problem consists of finding the most adequate parameters for DFCN broadcasting algorithm. This section is arranged as follows; in section 3.1 we describe the considered network in our work. We describe in section 3.2 the target broadcasting algorithm DFCN which should be tuned and in section 3.3 we present the Multi-objective optimization problem of our work.

3.1. **Metropolitan mobile ad hoc networks (MANET).**

Metropolitan mobile ad hoc network is MANET with the following properties. The first property is the high density areas where the nodes density is higher than the average i.e. school, airport, or supermarket. High density areas don't remain active all the time, they may appear or disappear from the system at any time i.e. school working hours from 8:00am to 5:00pm and the density of this school area outside this period is very low.

We needed a software simulator to represent such a network which allows us to tackle our problem. The chosen software simulator is Madhoc, a metropolitan MANET simulator [18]. Madhoc works as a tool to simulate different scenarios and environments based on some parameters.

There are a number of topological configurations such as people moving in a gallery place, airport place, and shopping center. The previous scenarios have different characteristics such as the size of the area, the mobility, the density of devices, the existence of walls (which has an effect on both the mobility and the signal strength), and other characteristics. We used three different scenarios implemented by Madhoc. The chosen scenarios are real world scenarios that model metropolitan area, shopping mall and a highway scenario.

- **Metropolitan environment** The metropolitan environment simulates MANETs in a metropolitan area. In this environment, we located a set of spots (crossroads) and connect them by streets. We model both human and vehicles, and they are continuously moving from one crossroad to another through streets. It is obvious that devices need to reduce their speed while attempting to cross a crossroad (like in the real world).
- **Mall environment** The mall environment is used to simulate MANETs in commercial shopping center. In this environment, the shops are located together in the corridors. The people move from one shop to another through corridors, and sometimes they stop to watch some shop window. These malls are very crowded (the density of devices is high).The behavior of people in shops is different from their behavior out of those shops (in term of mobility). There is a high density of shops in this environment. At the end, the walls of building restrict the mobility of devices and their signal propagation.
- **Highway environment** The highway environment simulates MANETs outside cities. This environment is characterized by the large surface with roads, and people travelling by car. Therefore, the density of this environment is very low since all devices are located in the roads moving in a high speed (in term of mobility). The obstacles that attenuate the signal strength and devices movement do not exist.

3.2. **Delayed flooding with cumulative neighborhood (DFCN).** The broadcasting protocols can be classified according to their algorithmic nature by the following criteria: determinism, reliability, or the information required by their execution such that the content of the hello messages. The deterministic algorithms do not use any randomness while the reliable algorithms guarantee the full coverage of the network [12]. In another work [14] the protocols are categorized as centralized and localized. Centralized protocols [1] need a global or semi-global knowledge of the network. So they are not scalable. On the other hand, the local protocols need some knowledge about one or two hops in the network.

According to the classification presented earlier, DFCN is a deterministic algorithm. It is a local protocol which works with 1-hop knowledge that permits DFCN to achieve great scalability. In DFCN, the "hello" messages do not carry any additional information but the broadcasting messages embed the list of node's neighbors.

Here is some additional information about DFCN.

- DFNC requires 1-hope neighborhood information like many other neighborhood knowledge based broadcasting protocols. DFNC obtains the required information through "hello" packets which work on network layer. The set of neighbors of device x is called N(x).
- The set of IDs of the 1-hop neighbors of every broadcasted message m is embedded in the header of m.
- Each device records local information about all the received messages. The single record of this local information consists of:
  - The received message ID.
  - The set of IDs of the devices that receive the message.
  - The decision of whether or not the message should be forwarded.
- Random Assessment Delay (RAD) is a random delay used by DFCN before re-forwarding a broadcast message m. It is used to prevent the collisions. In another word, while a device x forwards a message m, all the devices in N(x) receive it in the same time. Then all of them will re-forward the message m simultaneously and this causes network collisions. The goal of using RAD is delaying the process of re-forwarding the message m for each device in N(x) with a random value. Therefore, the risk of collisions is significantly reduced.

DFCN algorithm can be divided into three parts. The first two parts are responsible for dealing with outcoming events. The first part is responsible for dealing with new message reception, while the second is responsible for detecting a new neighbor. The third part is responsible for re-forwarding the received messages or detecting new neighbor during the follow-up of one of the previous parts. Reactive behavior is the behavior resulting from a message reception. Proactive behavior is the behavior resulting when a new neighbor is discovered.

Let x_1 , x_2 are two neighbor devices. When x_1 sends a message m to x_2 , the list of $N(x_1)$ are embedded in the sent message m. After x_2 receives the message m, it knows the set of all the message m recipients $N(x_1)$. Therefore, $N(x_2) - N(x_1)$ are the set of devices that have not received the message m yet. If x_2 re-forwards the message m, the number of devices that receives m for the first time is maximized through the following equation: $h(x_2, x_1) = |N(x_2) - N(x_1)|$.

The received message m is re-forwarded only if the number of neighbors who have not received the message m yet is greater than a given threshold to reduce the usage of network bandwidth. The threshold is a function of the neighbor devices for the receptor x_2 and it is written as threshold $(|N(x_2)|)$.

The device x_2 uses a Boolean function $B(x_2, x_1)$ to decide whether to re-forward the message m or not. The Boolean function $B(x_2, x_1)$ is defined as:

$$B(x_1, x_2) = \begin{cases} true, & h(x_1,x_2) \geq threshold(|N(x_2)|) \\ false, & otherwise \end{cases} \tag{1}$$

The recipient devise x_2 re-forwards the message m only if the threshold is exceeded. After the random delay defined by RAD is finished, the message m is re-forwarded. The threshold function allows DFCN to facilitate the message re-forward when the connectivity is low. It takes the recipient device x_2 neighbors number as a parameter and it is defined as:

$$threshold(n) = \begin{cases} 1, & n \leq safeDensity \\ minGain*n, & otherwise \end{cases} \tag{2}$$

DFCN always re-forwards while the density is below the maximum safe density called safe Density. DFCN uses minGain parameter to compute the minimum threshold for forwarding a message.

When the device x discovers a new neighbor, it forwards this discovery if $N(x)$ is lower than the given threshold called $proID$, otherwise this behavior is disabled, which means that there is no action taken in case of the new neighbor discovery.

3.3. **DFCNT (DFCN Tuning) as MOP.**

In this subsection, we present the Tuning of DFCN as a multi-objective optimization problem that we call DFCNT. The following are the five parameters that must be tuned with the role and range of each parameter in the DFCN.

- *minGain* is the minimum gain from the re-broadcasting process. Since minimizing the bandwidth should be highly dependent on the network density, minGain is the most important parameter for tuning DFCN. It ranges from 0.0 to 1.0
- *lowerBoundRAD* parameter is used for defining the lower bound of RAD value (random delay in re-broadcasting in milliseconds). This parameter takes values in the interval [0.0, 10.0] ms.
- *upperBoundRAD* parameter is used for defining the upper bound of RAD value. The parameter takes values in the interval [0.0, 10.0]ms.
- *proD* parameter is used for setting the maximum density to enable the proactive behavior (reacting to new neighbor). The parameter takes values in the interval [0, 100].
- *safeDensity* parameter is used for defining a maximum safe density of the threshold. This parameter takes values in the interval [0, 100].

The previous five parameters are considered as decision variables that characterized the search space. The chosen intervals are wide enough to include all the reasonable values that can be found in real scenarios. The three objective functions are defined as follows: the first objective function is minimizing the duration of the broadcasting process, the second is maximizing the network coverage and the third is minimizing the number of transmission (reduce bandwidth usage).Since we have three different real world Metropolitan MANETs scenarios, three instances of DFCNT have to be solved: DFCNT, Meropolitan, DFCNT, Mall and DFCNT, Highway.

4. The algorithm

Using EAs (Evolutionary Algorithms) in solving optimization problem has been very intense during the last decade [22]. It is possible to find this kind of algorithms tackling complex problems like constrained optimization task. These algorithms work on a set (population) of solution (individuals) by applying some stochastic operator on them to search for the best solution. Most EAs use a single population of individuals. They also apply their stochastic operator on the whole population as illustrated in figure [1]. On the other hand, there are other EVs that use structured population. In that case, the population is somehow decentralized. Structured EVs most suited to parallel implementation. The EAs that use decentralized population provide a sampling of the search space which improves both numerical behavior and execution time better than those that use single population. Distributed and cellular EVs are the most popular among many types of structured EAs as illustrated in figure [1] [4][6][7][11].We focus in this work on Cellular Genetic Algorithms (CGAs). CGAs use a small neighborhood concept, which mean that individual can only interact with his neighbors [4]. The overlapped small neighborhoods of CGAs help with exploring the space because the induced slow diffusion of solutions through the population provides a kind of exploration while exploitation takes place inside each neighborhood by genetic operations. Although CGAs were initially designed to parallel processors machines, they were adapted to suit mono-processor machines and accomplish good results. The neighborhood definition (during the CGA execution) did not depend on the graphical neighborhood definition in the problem space.

4.1. **Cellular genetic algorithm.** In this subsection, we present the canonical CGA in detail as published on [9]. CGA pseudo-code is presented in Algorithm 1. Since CGA is a structured EA, its population is structured as follows: it is usually structured in a regular grid of d dimensions with the neighborhood defined on it. The algorithm works on each individual in the population according to its place orderly (Algorithm1 line 5). The current

individual can only interact with his or her neighbors (Algorithm 1line 6). The current individual parents are chosen from the neighbors by using some selection technique (Algorithm 1 line 7). In line 8 and 9, crossover and mutation operators are applied to the current individual with probabilities P_c, P_m respectively. After that, the algorithm computes the fitness values of the offsprings (line 10) then, inserts them or one of them instead of the current individual either in the current population or in a new one according to the chosen replacement policy (line 11).

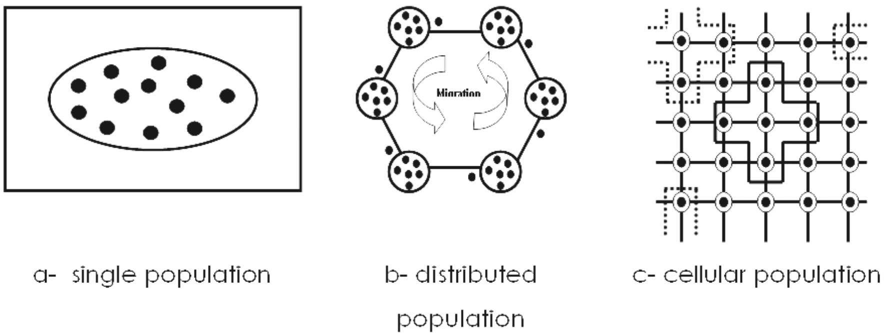


FIGURE 1. single (a), distributed (b), and cellular (c) EAs

After finishing the previous cycle for all individuals, we get a new population for the next generation (line 13). The loop continues until termination condition is met (line 4). The termination condition is met either by finding the optimum solution or exceeding the maximum number of calling the evaluation function or composed of both.

4.2. **Multi-objective cellular GA: MOCELL.** In this subsection, we present MOCELL, a multi-objective optimization algorithm based on a cGA model as presented in [2][3]. But we needed to modify it in order to tackle our problem in terms of dealing with multiple non-heterogeneous decision variables .We observed that Algorithms 1 and 2 were very similar. One of the main differences between the two algorithms is the existence of a Pareto front (see section 2) in the MOCELL algorithm. The Pareto front is just an additional population (the external archive) composed of a number of the non-dominated solutions found since it has a maximum size. In order to manage the insertion of solutions in the Pareto front with the goal of obtaining a diverse set, a density estimator based on the crowding distance proposed for NSGA-II [17] has been used. This measure is also used to remove solutions from the archive when it is full.

MOCell starts by creating an empty Pareto front (line 2 in Algorithm 2). Individuals are arranged in a 2-dimensional grid and the genetic operators were successively applied on them (lines 9 and 10) until the termination condition was met (line 5). Hence, the algorithm for each individual consists of two parents from their neighborhood, recombining them in order to obtain an offspring, mutating it, evaluating the resulting individual and inserting it in both the auxiliary population (if it is not dominated by the current individual) and the Pareto front. Finally, after each generation, the auxiliary one replaces the old population and a feedback procedure is invoked to replace a fixed number of randomly chosen individuals of the population by solutions from the archive.

Algorithm 1 Pseudo-code of a canonical cGA

```
1:Proc Evolve(cga)
2:GenerateInitialPopulation(cga.pop);
3:Evaluation(cga.pop);
4:While ! StopCondition() do
5: for  individual = 1 to cga.popSize do
6:   neighbors =calculateNeighborhood(cga, position(individual));
7:   parents   =selection(neighbors);
8:   offspring =recombination(cga.Pc, parents);
9:   offspring =mutation(cga.Pm,offspring);
10:  evaluation(offspring);
11:  replacement(position(individual), auxiliary_pop,offspring);
12: End for
13: Cga.pop     =auxiliary_pop;
14:end while
15:end proc Evolve
```

5. Experiments

In this section, we first describe the configuration of the network simulator (MadHoc). Next, we present the parameterization used by MOCELL. Finally, we present the analysis of the obtained results for DFCNT.

MOCELL has been implemented in Java and tested on a PC with a 2.8 GHz (dual-core) processor with 2GB of RAM memory and running windows XP service back 3. The java version used is 1.7.0. Although cMOGA was used in previous research to tackle our problem, we re-implemented it in order to avoid the influence of the differences between the programming techniques used in this and the previous study.

Algorithm 2 Pseudo-code of MOCELL

```
Proc Evolve(mocell)
Pareto_front = createPFront();
GenerateInitialPopulation(mocell.pop);
Evaluation(mocell.pop);
while ! StopCondition() do
 for  individual = 1 to mocell.popSize do
   neighbors = getNeighborhood(mocell, position(individual));
   parents   = selection(neighbors);
   offspring = recombination(mocell.Pc, parents);
   offspring = mutation(mocell.Pm,offspring);
   evaluation(offspring);
   Insert(position(individual),offspring,mocell, auxiliary_pop);
   InsertInParetoFront(individual,Pareto_front);
 end for
mocell.pop = auxiliary_pop;
mocell.pop = Feedback(mocell,Pareto_Front);
end while
end proc Evolve
```

5.1. **Madhoc Configuration.** There are three different environments for MANETs that Model three possible real-world scenarios. The main features of these environments are explained in this chapter and they are summarized in table [1]. In figure [2], we show an example for each environment. The examples in figure [2] are obtained by using the graphical user interface of Madhoc simulator by using the proposed configurations summarized in table [1].The broadcasting process is considered to be completed when either the coverage is 100% or it does not vary for 1.5 second. The broadcasting process termination is truly important since improper termination condition can lead to bad results or slow simulation.

Table 1. Main features of Madhoc environment

		Metropolitan	**Mall**	**Highway**
Surface (m^2)		160,000	40,000	1,000,000
Density of spots		50 (crossroad/km^2)	800 (store/km^2)	3 (joints/km^2)
Spots radius (m)		3 - 15	1 - 10	50 - 20
	Speed out of spots (m/s)	1 - 25	0.3 - 1	30 - 50
Devices	Speed in spots (m/s)	0.3 - 10	0.3 - 0.8	20 - 30
	Density(dev./km^2)	500	2000	50
Wall obstruction (%)		90	70	0

5.1.1. *The Metropolitan Environment.* In this section, we study the behavior of DFCN in the Metropolitan environment. In this environment modulation, we set the surface as 400 * 400 square meters. The density of spots (crossroads) is 50 per square kilometer. Each spot has a circle surface of radius between 3 and 15 meters. In this scenario, the wall obstruction (penalty of signal) is up to 90%. The density of the devices is 500 elements per square kilometer. While setting the speed parameter, we should consider the cases when people or cars move, so the value of movement speed in crossroads area ranges between 0.3 and 10 m/s, and between 1 and 25 m/s in other cases (streets). This kind of environment consists of a few numbers of sub-networks that are connected to each other by few links, one or two or even zero in case of unconnected subnetworks. Isolated nodes are those devices that are not connected to any subnetworks as illustrated in figure [2]. The topology of this environment can vary in a very fast way since the devices can move through cars. All of the previous properties show us how hard is the broadcasting process through this network and this was what made this scenario challenging for us.

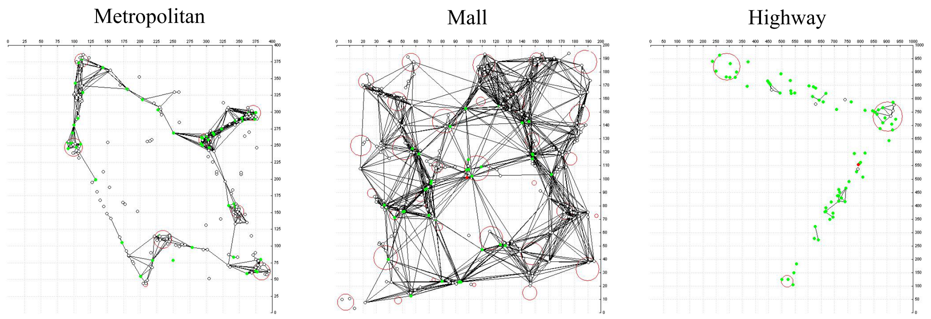


Figure 2. MANET scenario

5.1.2. *The Mall Environment.* In this section, we show the parameter of Madhoc configuration of the mall environment. In this scenario, the number of both shops (spots) and devices is very high (density). There are walls that have two roles, the first is to attenuate the signals and the second is to slow down the speed of devices that is already slow since we are modeling people walking. The surface of this environment is defined as 200 * 200 square meters. The number of devices per kilometer is 2000. The number of stores (spots) per kilometer is 800. Each store (spots) has a circle of radius ranging between 1 and 10 meters. The obstruction of the wall is measured by 70% attenuation of the signal strength. At the end, the speed of the devices range between 0.3

and 1 m/s inside the corridors (speed out of spots) and between 0.3 and 0.8 m/s inside stores (speed in spots).

In figure [2], we can notice that the mall environment diagram is a very condensed graph. The graph is condensed because the mobile devices coverage ranges between 40 and 80 meters. Therefore, the Mall environment problem is the hardest problem because of the broadcast storm [21].

5.1.3. *The Highway Environment.* The highway environment consists of a small number of devices moving in a high-speed manner. In this environment, there is no wall obstruction. The signal attenuation is set to 0%. The surface in this environment is 1000 * 1000. The number of devices is 50 devices per square kilometer. There are three spots (highway entrances or exits) in this scenario. The speed of devices outside the spots ranges from 30 to 50 m/s. The speed of the devices inside the spots ranges from 20 and 50 m/s. The radius of each spot ranges from 25 to 100 meter.

In figure [2], we can notice that the highway environment consists of a number of subnetworks usually unconnected. Each subnetwork is composed of a small number of devices. The main challenge in this scenario is how fast the topology changes because of the speed of the devices in the highway. The faster change the topology makes, the harder the broadcast process becomes.

5.2. **Parameterization of MOCELL.** In this section, we explain the parameter used by MOCELL in our experiment. The population consists of 100 individuals formed as square toroidal grid. We used C9 (compact nine) neighborhood composed of 9 individuals; the selected one and all adjacent individuals as illustrated in figure [3]. Per each evaluation function calling, we call the madhoc simulator five times because of the stochastic nature of the simulator. The objectives (time, coverage, and bandwidth) are calculated as an average of the five returned values through the five simulator calling. Calling the simulator five times per function has a great effect on our experiment time. The previous details show us why the number of algorithm is just 30 times.

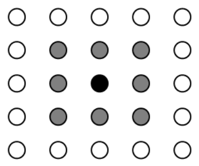


FIGURE 3. C9 neighborhood

Table 2. MOCELL Parameters

Parameter Name	**Parameter value**
Population size	100 individuals
Stop condition	25000 function evaluations
Neighborhood	C9
Parent selection	Binary Tournament + Binary Tournament
Recombination	Simulated Binary probability = 1.0
Mutation	Polynomial probability = 1.0/L
Replacement	Rep_If_Better
Archive size	100
Density estimator	Crowding distance
Feedback	20 individuals

Simulated Binary operator (SBX) [15] is used in the recombination phase with probability $p_c = 1.0$ since we deal with continuous decision variables. SBX simulates the behavior of the single point binary crossover on double individuals. We used polynomial operator [15] as a mutation operator with probability $p_m = 1.0/L$ for every allele (where L is the length of individual). We chose both parents by using Binary Tournament. The resulting offspring replaces the current individual if it dominates the current individual. We used adaptive grid algorithm to insert the individuals into the Pareto Front [13]. This algorithm divides the objective space into hypercubes that lead to the balance of the density of the non-dominated solutions in these cubes. In the case of inserting a new non-dominated solution into the Pareto Front, the grid location of the solution is determined. If the Pareto Front is already full and the new non-dominated solution does not belong to the most crowded hypercube then one of the solutions that belongs to that hypercube is removed to leave a space for the new non-dominated solution.

Using the try and error technique, we conclude that the previous MOCELL parameters are considered the best parameters for MOCELL in solving the aforementioned problem.

5.3. **Results for DFCNT.** In this section, we analyze the result of DFCNT in the three different environments. The DFCNT problem is composed of five decision variables and three objective functions. The experiment consists of 30 independent runs for each problem environment. The experiment execution time is almost 2 months.

We show the mean and standard deviation of both time (in hours) and number of Pareto optima obtained by MOCELL for the three different instances of the DFCNT problem (metropolitan, mall, and highway) in table

[3]. As we can see, the single execution run is 23 hours for Metropolitan and 16 hours for mall and 10 hours for highway. The complexity of the evaluation function, since we call the simulator five times, is the only reason for the long time of our experiment. The average of the number of Pareto optima obtained is 98.9 for Metropolitan, 99.6 for Mall, and 97.4 for highway where the maximum is 100 solutions per run. This result is very satisfying for the three instances of the problem since we provide the decision makers with a wide range of solutions.

In Figure [4], we show the diversity of MOCELL result for each of the three instances of the DFCNT problem. Best solutions are those that satisfy the following objective functions (maximize the coverage, minimize bandwidth and minimize the duration of the broadcasting process). From the obtained results, the solutions that cover over 95% in the broadcasting process need in average 720.8 ms and 69.91 messages (bandwidth usage) for the Metropolitan scenario. In addition, they need 163 ms and 22.45 messages for the mall scenario and 827.1 ms and 71.61 messages for the highway scenario. In fact, only 11% from the Pareto optima solutions reach 95% coverage for the metropolitan environment while 39% and 6% for mall and highway in consecutive. The previous results reflect the importance of the coverage and how hard it is to satisfy this objective function.

By looking to figure [4], we can note that in the case of the mall scenario, the duration is less than 250 ms, bandwidth usage is less than 30 messages and the coverage is always more than 0.4. Therefore, it is so obvious that the broadcasting process in the mall scenario is better than the other scenarios. In figure [4], the Duration axis (time in milliseconds) shows that the broadcasting process in both metropolitan and highway scenarios takes longer time than mall scenario. The bandwidth axis (number of sent messages) shows us that the broadcasting process in both metropolitan and highway scenarios take longer time than the mall scenario. The coverage axis (percentage of all devices in the network) shows us that there are some solutions with coverage less than 10% in both metropolitan and highway scenarios.

TABLE 3. experiment's time, and number of Pareto optimal for each problem

Environment	Time (h)	Number of Pareto optima
DFCNT.Metropolitan	$23.09 \pm_{0.998}$	$98.9 \pm_{1.45}$
DFCNT.Mall	$15.87 \pm_{0.368}$	$99.6 \pm_{0.966}$
DFCNT.Highway	$9.852 \pm_{0.181}$	$97.4 \pm_{5.235}$

The previous coverage results are expected because they depend on the difference between the scenarios. The probability of having isolated sub-networks (consists of one or two devices) increases with the decrease in devices density (increase simulation area and decrease devices number). Since the mall scenario has the highest connectivity (highest devices density), it has the best coverage results. However, the high density has its drawback because it increases the risk of broadcast storm which makes solving DFCNT.mall very hard. Based on these results, we note that MOCELL succeeded in dealing with this problem.

The Pareto fronts illustrated in figure 4 achieves the designs objectives of the DFCN protocol, since most of the plots are distributed on a wide range that provides a decision maker with a wide variety of solutions. Our results also have a set of solutions that allow DFCN to achieve a coverage rate close to 100%, while keeping the network throughput very low.

6. Comparing MOCELL against CMOGA

In this section, we compare our study with those that used CMOGA on DFCNT problem. The three instances of DFCNT problem (metropolitan, mall, and highway) are solved with CMOGA to make this case study. Although the DFCNT problem has previously been solved by CMOGA algorithm in [8], we re-implemented CMOGA algorithm in order to insure high accuracy in our comparative study by avoiding implementing differences effect.

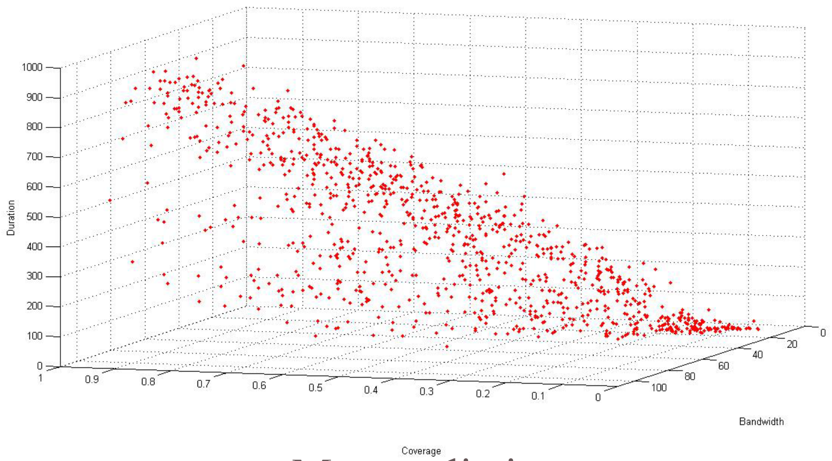
6.1. **Parameterization of CMOGA.**

In this section, we show the CMOGA algorithm parameters. The algorithm population size is 100 individuals. It stops when 25000 evaluation functions have been made. We chose C9 as a neighborhood operator described in section 2 and illustrated in figure [3]. Both the parents are chosen by Binary tournament operator. In the Recombination step, we used simulated binary crossover [15] with probability = 1. In the Mutation step, we used polynomial [15] with probability $p_m = 1.0/L$ for every allele (where L is the length of individual). The offspring replaces the current individual only if the former dominates the latter. The maximum archive size is 100 individuals. We used adaptive grid algorithm to insert the individuals into the Pareto Front [13]. As you notice, we used almost the same parameters of MOCELL for CMOGA algorithm to insure the accuracy in our comparative study. For evaluating each individual, we had to call the simulator five times as in the case of MOCELL. Therefore, in each single run of CMOGA algorithm, we had called the simulator 125,000 times.

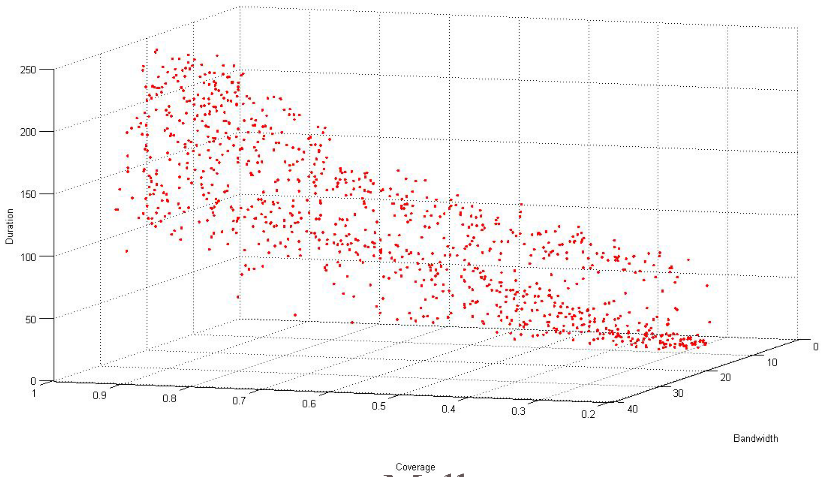
We used the same parameters previously used with MOCELL in order to insure the precision of our comparative study.

6.2. **Evaluation of the Results.**

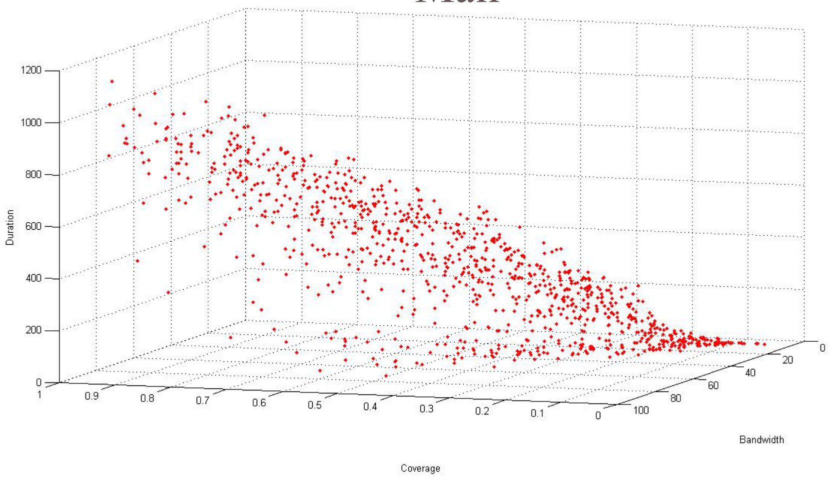
As Cellular Genetic Algorithms belong to meta-heuristic algorithms, it is considered as a non- deterministic technique



Metropolitain



Mall



Highway

Table 4. CMOGA Parameters

Parameter Name	**Parameter value**
Population size	100 individuals
Stop condition	25000 function evaluations
Neighborhood	C9
Parent selection	Binary Tournament + Binary Tournament
Recombination	Simulated Binary probability = 1.0
Mutation	Polynomial probability = 1.0/L
Replacement	Rep_If_Better
Archive size	100
Density estimator	Crowding distance

and this means that different solutions can be reached by using the same algorithm twice on the same problem. The previous detail makes a serious problem for the researchers in evaluating their results and in comparing their algorithms results to existing algorithms.

The studied algorithms are applied to 3 scenarios of real-world problems to insure that the proposed algorithms are capable of tackling such problems.

In our case of multiobjective optimization algorithms, we have to use metrics to compare the quality of the obtained solutions. However, until now there is no single metric that proves its superiority to the other metrics. So, we need to use more than one metric to insure the accuracy in our comparative study. The chosen metrics are a number of Pareto Optima, hypervolume, and set coverage [10]. Once we apply any of the previous metrics on our obtained pareto front, we get a single value.

Table 5. MOCELL vs. CMOGA Number of Pareto optima

	Algorithm	X	Max	Min	Test
DFCNT.Metropolitan	MOCELL	$98.9\pm_{1.45}$	100	96	-
	CMOGA	$99.7\pm_{0.675}$	100	98	
DFCNT.Mall	MOCELL	$99.6\pm_{0.966}$	100	97	-
	CMOGA	$99.9\pm_{0.316}$	100	99	
DFCNT.Highway	MOCELL	$97.4\pm_{5.23}$	100	84	-
	CMOGA	$94.7\pm_{8.68}$	100	72	

Since the proposed algorithms are non-deterministic, the comparison of a single execution is inconsistent. So, the comparison must be applied on a large set of results obtained after a high number of independent executions (in our case 30 independent runs) for the algorithms on a given problem. We used a statistics function in order to make the comparisons between the obtained results. Our statistics reflect the significance of the obtained results and the comparisons as shown in the Test column.

We applied Kruskal-Wallis test on our results. Kruskal-Wallis function allows us to determine whether the effects observed in our results are significant or it appeared because of error in the collected samples. We chose this statistics function since we have non-normal data distribution. We used Kolmogorov-Smirnov test to check if our data distribution is gaussian or not. We considered a confidence level of 95% in our comparison study and this means that we can guarantee that the differences of the compared algorithms are significant or not with a probability of 95% or with the p-value less than 0.05.

TABLE 6. MOCELL vs. CMOGA for Hyper volume metric

	Algorithm	X	Max	Min	Test
DFCNT.Metropolitan	MOCELL	$0.9998\pm_{5.32E-04}$	1	0.998	-
	CMOGA	$0.9996\pm_{1.22E-03}$	1	0.996	
DFCNT.Mall	MOCELL	$0.9965\pm_{3.77E-03}$	1	0.989	-
	CMOGA	$0.9964\pm_{3.91E-03}$	1	0.99	
DFCNT.Highway	MOCELL	$0.9998\pm_{6.71E-04}$	1	0.998	-
	CMOGA	$0.9999\pm_{1.46E-04}$	1	0.999	

6.3. **Discussion.** In this section, we made the comparison between MOCELL and CMOGA algorithms. As previously mentioned, the results are obtained after making 30 independent runs of every experiment for each algorithm and the used metrics are number of non-dominated solutions found in the Pareto Front, Hypervolum, and Set Coverage.

The obtained results are shown in tables [5], [6], and [7]. The previous tables include x (the mean) and the standard-deviation of our results. They also include the maximum and minimum obtained values for each metric.

In table [5], although the obtained results are not statistically significant, we can notice that both algorithms MOCELL and CMOGA reached a high number of Pareto Optima since the maximum number of Pareto Optima is 100 solutions. In table [6], MOCELL improves CMOGA in metropolitan, and mall scenarios in terms of the hypervolume metric but the difference is not statistically significant. But CMOGA improves MOCELL in the mall scenario in terms of the hypervolume metric without statistical significance. We can notice that both of algorithms have reached high level of Hypervolume metric since the maximum value is 1. The result of the set coverage metric is shown in table [7]. The MOCELL outperforms CMOGA in two of the studied problems (metropolitan, and highway scenarios) with statistical significance in terms of the set coverage metric. In contrast to the previous scenarios, CMOGA outperforms MOCELL in the mall scenario with statistical significant in terms of the set coverage metric.

To sum up, there is no algorithms better than the others. But MOCELL seems to be better than CMOGA in terms of hypervolume and set coverage. On the other hand, CMOGA outperforms MOCELL in the case of number of pareto optima. The differences between the two algorithms are statistically significant for the set coverage metric. On the other hand, we did not find any important differences in the other two metrics (number of pareto optima, and hypervolume).

Table 7. MOCELL vs. CMOGA for Set Coverage metric

	C(A,B)					
	A	B	X	Max	Min	Test
DFCNT.	MOCELL	CMOGA	$0.3501 \pm_{9.42E-02}$	0.6	0.122449	+
Metropolitan	CMOGA	MOCELL	$0.3209 \pm_{8.14E-02}$	0.51	0.15625	
DFCNT.	MOCELL	CMOGA	$0.2841 \pm_{6.91E-02}$	0.4848	0.16	+
Mall	CMOGA	MOCELL	$0.3322 \pm_{7.56E-02}$	0.51	0.175258	
DFCNT.	MOCELL	CMOGA	$0.3704 \pm_{9.60E-02}$	0.6	0.180556	+
Highway	CMOGA	MOCELL	$0.3577 \pm_{9.88E-02}$	0.6071	0.113402	

7. Conclusions and future works

In this paper we present the problem of optimally tuning DFCN (broadcasting protocol) which works on MANET (Mobile Ad-hoc wireless Network), by using MOCELL (Multi-objective optimization algorithm). DFCNT is defined as a three objectives MOP, with the goals of minimizing the network usage, maximizing network coverage and minimizing the duration of broadcasting.

We used three different realistic scenarios. Three different instances of MOP have been solved. They are city's streets (DFCNT.Metropolitan), mall center (DFCNT.mall) and Highway streets (DFCNT.Highway). we can conclude that solving DFCNT by MOCELL provides a Pareto front set that consists of more than 95 points in the case of the highway scenario and more than 99 points in the case of the other two scenarios.

In the second part of this paper, we compared our chosen algorithm MOCELL versus cMOGA (cellular Multi-Objective Genetic Algorithm) for the three proposed problems. Three different metrics were used in order to compare the algorithms: The number of Pareto optima, the hypervolume, and the set coverage metrics. We observed that MOCELL seemed to be better than CMOGA in terms of hypervolume and set coverage. On the other hand, CMOGA outperformed MOCELL in the case of number of pareto optima. Although the differences between the two algorithms in hypervolume and number of pareto optima metrics are not statically significant, both of them reach a high pareto optima result (since the maximum Pareto front is 100) and a high hypervolume results (since the maximum value is 1.0). Regardless the hypervolume and the number of Pareto optima metrics, MOCELL won. From these results, a clear conclusion can be drawn: MOCELL is a promising approach for solving DFCNT with advantage over the existing one.

Future research is needed to tackle the MOPs with MOCELL. In addition, research that parallels MOCELL to reduce the execution time is needed because reducing time will enable us to study other real-world scenarios that are larger and have bigger number of devices.

References

[1] A. Pelc, Handbook of Wireless Networks and Mobile Computing, Wiley, 2002, Ch. Broadcasting In Wireless Networks, pp. 509-528.

[2] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In D.A. Pelta and N. Krasnogor, editors, Proceedings of the NICSO, pages 25-36, Granada, Spain, 2006.

[3] A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. MOCell: A cellular genetic algorithm for multiobjective optimization. International Journal of Intelligent Systems, 2007.

[4] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithm. In J.D. Schaffer, editor, Proc. of the Third International Conference on Genetic Algorithms (ICGA), pages 428-433. Morgan Kaufmann, 1989.

[5] C.A. Coello, D.A. Van Veldhuizen, G.B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, 2002.

[6] D. Whitley. Cellular genetic algorithms. In S. Forrest, editor, Proc. of the Fifth International Conference on Genetic Algorithms (ICGA), page 658, California, CA, USA, 1993. Morgan Kaufmann.

[7] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 6(5):443-462, October 2002.

[8] E. Alba, B. Dorronsoro, F. Luna, A.J. Nebro, P. Bouvry, and L. Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. Computer Communications, 30(4):685-697, 2007.

[9] E. Alba, B.Dorronsoro, Handbook of Cellular Genetic Algorithms, 2008, Ch. Introduction to Cellular Genetic Algorithms, pp. 3-20.

[10] E. Alba, B.Dorronsoro, Handbook of Cellular Genetic Algorithms, 2008, Ch. Algorithmic and Experimental Design, pp. 73-82.

[11] E. Cantu-Paz. Efficient and Accurate Parallel Genetic Algorithms, volume 1 of Book Series on Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, 2nd edition, 2000.

[12] I. Stojmenovic, J. Wu, Broadcasting and activity scheduling in ad hoc networks, in: Mobile Ad Hoc Networking, IEEE/Wiley, 2004, pp. 205-229.

[13] J. Knowles and D. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation, 8(2):149-172, 2001.

[14] J. Wu,W. Lou, Forward-node-set-based broadcast in clustered mobile ad hoc networks, Wirel. Commun. Mobile Comput. 3 (2) (2003) 155.

[15] K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. Complex Systems, 9:115-148, 1995.

[16] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms,John Wiley & Sons, London, 2001.

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elist Multi- objective Genetic Algorithm: NSGA-II. IEEE TEC, 6(2):182197, 2002.

[18] L. Hogie, F. Guinand, P. Bouvry, The Madhoc Metropolitan AdhocNetwork Simulator, Universite du Luxembourg and Universite du Havre, France, available at http://www-lih.univ-lehavre.fr/∼hogie/madhoc/.

[19] L. Hogie, M. Seredynski, F. Guinand, P. Bouvry, A bandwidthefficient broadcasting protocol for mobile multi-hop ad hoc networks, in: ICN'06, 5th International Conference on Networking (to appear), IEEE, 2006.

[20] M. Laumanns, G. Rudolph, H.P. Schwefel, A spatial predator-prey approach to multi-objective optimization: a preliminary study, in:PPSN V, 1998, pp. 241-249.

[21] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In Proc. of the Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 151-162, 1999.

[22] T. Back, D.B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Oxford University Press, 1997.

Misr University for Science and Technology, Faculty of Information Technology, Al-Motamayez District, 6th of October City, Egypt
E-mail address: MS_Ghonamy@hotmail.com

Cairo University, Faculty of Computers and Information, 5 Dr. Ahmed Zewail Street, Giza city, Egypt
E-mail address: a.badr@fci-cu.edu.eg

Arab Academy for Science and Technology, College of Computing & information Technology, El Moshir Ismail Street, Misr El Gedida, Cairo, Egypt
E-mail address: abdheg@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

ALIGNMENT OF CUSTOM STANDARDS BY MACHINE LEARNING ALGORITHMS

ADELA SÎRBU[1,2], LAURA DIOŞAN[1,2],
ALEXANDRINA ROGOZAN[1], AND JEAN-PIERRE PÉCUCHET[1]

Abstract. Building an efficient model for automatic alignment of terminologies would bring a significant improvement to the information retrieval process. We have developed and compared two machine learning based algorithms whose aim is to align 2 custom standards built on a 3 level taxonomy, using kNN and SVM classifiers that work on a vector representation consisting of several similarity measures. The weights utilized by the kNN were optimized with an evolutionary algorithm, while the SVM classifier's hyper-parameters were optimized with a grid search algorithm. The database used for train was semi automatically obtained by using the Coma++ tool. The performance of our aligners is shown by the results obtained on the test set.

1. Introduction

The need for terminology integration has been widely recognized in different areas (economical, custom etc) leading to a number of efforts for defining standardized and complete terminologies. It is acknowledged by literature that the creation of a single universal terminology for a particular domain is neither possible nor beneficial because different tasks and viewpoints require different, often incompatible conceptual choices. Considering these aspects, an important research direction is searching for an automatic recognition of concepts with the same meaning, even though they have not the same syntactic representation.

The main aim of our work is to design an algorithm in order to perform alignments between two terminologies. The goal of the alignments is to put in correspondence concepts that refer to the same thing, but which appear under different forms. In other words, alignment problem consists in finding the

Received by the editors: July 26, 2010.

2000 *Mathematics Subject Classification.* 68T05, 91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*.

Key words and phrases. Concept alignment, Machine Learning, Binary Classification, Support Vector Machine.

correspondences between the definitions of different dictionaries (standards) that refer to the same concept. This alignment of definitions, which is one of the goals of ASICOM project[1] as well, has certainly to improve the fusion between the business models of different companies.

In our case, the concepts we are trying to align are belonging to the custom area and they are defined in two standards: CCL08A (Centre for Trade Facilitation and Electronic Business standard or shorter CCL) and Customs WCO (World Customs Organization or shorter WCO). In order to automatically perform this alignment, several definitions are considered from the two dictionaries. Pairs of two definitions (that can define the same concept - and in this case we deal with two aligned definitions - or different concepts - unaligned definitions in this case) are formed. Thus, the alignment problem could be considered as a binary classification problem: the inputs of the classifier are the pairs (of two definitions) and the outputs are the labels "aligned" or "unaligned" corresponding to each pair.

As we have mentioned, aligning two definitions means actually to solve a binary classification problem. Several couples of definitions, which could be aligned or unaligned, are required, so that the classifier could learn to discriminate correctly such relationships. In order to perform this alignment as a classification approach, all the possible couples of definitions are considered from the mentioned dictionaries (CCL and WCO). In this way, if a dictionary contains n_1 definitions and the other dictionary contains n_2 definitions, then we will be obtained $n_1 * n_2$ couples of definitions (some of them are aligned couples, while others are unaligned couples). Taking into account that we deal with a classification problem, a Machine Learning algorithm could be used.

For our task of solving the terminology alignment problem two Machine Learning based algorithms were chosen: a k Nearest Neighbour (kNN) algorithm [2] and a Support Vector Machine (SVM) [22] that work by using a particular representation based on the similarities between two definitions. Even if the kNN is a simple and fast algorithm, it requires to establish a threshold value and to specify the alignment cardinality. Therefore, we decided to try an SVM-based approach also, since it can learn the optimal value of the threshold and is able to produce any alignment type. Several performance measures, borrowed from the information retrieval domain, are used in order to evaluate the quality of the automatic produced alignments: the accuracy, the precision, the recall and the F-measure of alignments.

The remaining of the paper is structured as follows: Section 2 gives a short review of different alignment models, Sections 3 and 4 presents our solution to

[1]ASICOM – Architecture de Système d'information Interopérable pour les industries du Commerce

the alignment problem and several numerical results, while Section 5 concludes the results.

2. Related work

To our knowledge, the alignment problem has been intensively studied in order to achieve an automatic translation of terminologies, providing us several alignment models. The alignment problem has to be considered from two important points of view:

- the structures that must be aligned and
- the manner in which this alignment is actually performed (the matching algoritm).

2.1. **The structures that must be aligned.**

Regarding the structure of the alignment, two important levels of a dictionary could be identified: the sentence level and the ontology level.

2.1.1. *The sentence level.*

The sentence level refers to the bag of words of that dictionary, but it is actually a special bag of words where not only the frequency of a word is important, but also other linguistic information about these words. The richness of human language allows people to express the same idea in many different ways; they may use different words of the same language to refer to the same entity or employ different phrases to describe the same concept. Furthermore, the same idea could be expressed in different languages. Sentence-aligned bilingual corpora are a crucial resource for training statistical machine translation systems. In this context, we discuss about multilingual alignment. Therefore, the problem of sentence alignment for monolingual corpora is a phenomenon distinct from alignment in parallel, but multilingual corpora.

Monolingual alignment – Sentence-aligned bilingual corpora are a crucial resource for training statistical machine translation systems. Several authors have suggested that large-scale aligned monolingual corpora could by similarly used to improve the performance of monolingual text-to-text rewriting systems, for tasks including summarization [15, 16] and paraphrasing [1, 20]. Most of the work in monolingual corpus alignment is in the context of summarization. In a single document summarization alignment between full documents and summaries written by human is used to learn rules from text compression. Marcu [16] computes sentence similarity using a cosine-based metric. Jing [15] identifies phrases that were cut and pasted together using a Hidden Markov Model with features incorporated word identity and positioning within sentences, by providing an alignment of a document and its summary. In the context of multi document summarization, SimFinder [13] identifies sentences that convey similar information across input documents to select the summary content.

Multilingual alignment – To our knowledge, the problem of aligning sentences from parallel corpora has been intensively studied for automated translation. While much of the research has focused on the unsupervised models [3, 5, 11], a number of supervised discriminatory approaches have been recently proposed for automatic alignment [4, 18, 21]. Related to the use of linguistic information more recent work [19] shows the benefit of combining multilevel linguistic representations (enriching query terms with their morphological variants). By coupling Natural Language Processing and Information Retrieval (IR) the language is enriched by combining several levels of linguistic information through morphological (lemma, stem), syntactic (bigrams, trigrams) and semantic (terms and their morphological and/or semantic variants) analyses. Moreover, data fusion has been exhaustively investigated in the literature, especially in the framework of IR [8, 19]. The difficulty is to find a way to combine results of multiple searches conducted in parallel on a common data set for a given query in order to obtain higher performances than each individual search.

2.1.2. *The ontology level.* The ontology level is actually a generalisation of the first level that take into account for a given dictionary not only the words and their order in a sentence (when the words are in fact considered isolated elements), but also the relationships (syntactic, semantic or other relationship types) establish among the words/concepts. In other words, at this level a concept (or its definition) could be represented as a bag of concepts (by concept being understood the corresponding word and its relationships with other words/concepts).

At present, there exists a line of semi-automated schema matching and ontology integration systems, see for instance [12, 17]. Most of them implement syntactic matching. The idea of generic (syntactic) matching was first proposed by Phil Bernstein and implemented in the Cupid system [17], but COMA [12] is a generic schema-matching tool, which seems to be a more flexible architecture. COMA provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for evaluating of the effectiveness of different matchers.

2.2. **Matching algorithms.** We distinguish between matching algorithms at element-level and structure-level.

The element-level matching techniques focus on the entities of a dictionary (words or concepts).They compute matching elements by analyzing entities in isolation, ignoring their relations with other entities: string-based techniques (normalization techniques, substring or subsequence techniques, path comparison), language-based techniques (based on NLP techniques using morphological properties of the input words) – these methods may be either intrinsic (using the internal linguistic properties of the instances, such as morphological and syntactic properties: tokenization, elimination or filtration, lemmatization, stemming, weighting) or extrinsic (requiring the use of external resources, e.g. lexicon-based and multilingual methods: synonymy or semantic similarity).

The structure-level techniques focus on the structure of the ontology. It computes mapping elements by analyzing how entities appear together in the structure corresponding to a dictionary: graph-based techniques (graph algorithms which consider the input as labelled graphs) and taxonomy-based techniques (are also graph algorithms which consider only the specialization relation; the intuition behind taxonomic techniques is that is-a links connect terms that are already similar, therefore their neighbours may be also somehow similar).

3. Alignment methods

In order to obtain an automatic alignment two methods were utilized: the k Nearest Neighbour algorithm [2] and the Support Vector Machine [22].

3.1. **k Nearest Neighbour.**

kNN is a method of classifying objects based on closest training examples in a feature space. kNN is a type of instance based learning, or lazy learning where the function is only approximated locally and all the computation is deferred until classification. The kNN algorithm is amongst the simplest of all machine learning algorithms and is also very fast, two important reasons to utilize it for the alignment task. An object is classified by a majority vote of its neighbours, with the object being assigned to its class most common amongst its k nearest neighbours; k is a positive integer, typical small. If $k = 1$, then the object is simply assigned to its closest neighbour. In binary classification problems it's helpful to choose k to be an odd number as this avoids tied votes.

In our case an object corresponds to a definition (or to its representation as bag of ”special” words at different syntactic levels). The distance between two objects (definitions) is computed by using a similarity measure. The smallest distance (or the largest similarity measure) between two definitions (from all the possible combinations) will indicate that the two definitions are aligned. The main idea of the kNN classifier in this case is to search amongst the couples of two definitions those with similarity greater than a given threshold and to select from the found couples the first k couples with the largest similarity. Such algorithm is able to produce two types of alignments:

- one-to-one alignments when $k = 1$ or

- one-to-many alignments when $k > 1$.

Even if kNN's methodology seems to be very simple, the usage of this algorithm in order to reach our purpose has determined several important questions:

- Which is the optimal value for k?
- Which is the optimal value of similarity threshold?

In order to solve these problems an SVM algorithm instead of the kNN classifier was used, since SVM can learn the optimal value for the threshold and is able to produce any alignment type:

- one-to-one – a concept of a dictionary corresponds to one concept of the other dictionary (*equivalence* relation);
- one-to-many – a concept of a dictionary corresponds to many concepts of the other dictionary (*type of* relation);
- many-to-many – more concepts of a dictionary corresponds to many concepts of the other dictionary.

3.2. **Support Vector Machine.** SVMs are a set of related supervised learning methods used for classification and regression. Generally, classification is defined for the situation when there are more objects, each one belonging to one of several classes, and a classification task would be to assign the belonging class to a new given object. In the case of binary classification using SVM, being given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM model is a representation of the examples as points in space, mapped by a kernel so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In our case it is a binary classification problem, if the SVM is given a pair of definitions it will decide if they are aligned or not aligned.

An SVM algorithm has two phases: a training phase and a test phase. In the training phase the SVM model is learned starting from labelled examples (in our case, couples of definitions) and the hyper parameters are optimized and in the test phase the unseen definitions couples are labelled as aligned or unaligned.

Therefore, each data set has been divided in two: a part for training and a part for testing. The training part has been dived again in a learning subset, used by the SVM algorithm in order to find the hyper plane that makes the class separation and a validation set, used in order to optimize the values of the hyper parameters. The SVM model, which is learned in this manner, classifies (labels) the definitions couples from the test set.

4. Numerical experiments

4.1. **Data representation.** Our input data is represented by definitions stored in the two dictionaries (WCO Dictionary and CCL Dictionary). These dictionaries are provided by the team of ASICOM project in a form of 3 levels taxonomies: a concept is represented by the text of the definition, the path in the hierarchy of concepts and the father of the concept. In order to use a Machine Learning algorithm, the natural language of definitions has to be processed. First, each sentence was turned into a bag of words and then followed these steps: normalisation, filtering of the stop words and lemmatization. Furthermore, each couple of definitions is represented by a set of similarity measures.

4.1.1. *First set of similarities.* In order to perform automatic alignments using the kNN algorithm several similarities computed at different levels of the ontology were used :

- Name: Synonyms and TriGrams similarity applied on concept;
- NameType: Name and Data Type similarity, applied on the concept;
- Path: Name similarity, applied on path of the concept in the given ontology;
- Leaves: Name Type applied on leaves of the concept in the given ontology;
- Children: Name Type applied on children of the concept in the given ontology;
- Comment: TriGrams similarity, applied on comment (definition) tokens.

These six distances were computed using the Coma++ tool [10]. The similarity is represented by a weighted sum of these six distances whose weights are optimized on the training set using an evolutionary algorithm.

4.1.2. *Second set of similarities.* An important step in using the SVM is building the input vector. For each couple of definitions there is built a vector that contains five similarity measures:

- the Match coefficient [7] that counts the common element for two definitions,
- the Dice coefficient [9] that is defined as twice the number of common words, divided by the total number of words from two definitions,
- the Jaccard [14] coefficient, which is defined as the number of common words, divided by the total number of words from two definitions,

- the Overlap [6] coefficient, which is defined as the number of common words, divided by the minimum of the number of words from two definitions,
- the Cosine measure that is defined as cosine similarity of the product between term frequency of a term in a definition and the inverse definition frequency:

$$cosine(def_i, def_j) = \frac{def_i * def_j}{||def_i|| * ||def_j||}.$$

In addition, this vector contains also the lengths of the definitions and the label (aligned or not-aligned).

Since Match, Dice, Jaccard, Overlap similarities are based on reunion (the total number of words from two definitions) and intersection (the common words of two definitions) a special way of computing this reunion and intersection is actually used in the numerical experiments. This special computation is adapted to the bag representation (instead of set-based representation). Considering the following definitions and the corresponding bag of words:

- def_i = (a, b, b, c, c, d),
- def_j = (a, a, b, b, b, c, e).

In this case:

- $def_i \cup def_j$ = (a, b, b, b, c, c, d, e),
- $def_i \cap def_j$ = (a, b, b, c)

As shown in the example, a word may appear several times in a definition and from this reason when we calculate the cardinal of intersection of two definitions we take the minimum occurrence of a word in the two definitions and while on calculating reunion, the maximum occurrence.

4.2. **Construction of the database.** Taking into account that both Machine Learning's algorithms require a training set and a test set, in the absence of a human expert, a database of 180 aligned couples of definitions was semi automatic created using the Coma++ tool [10]. Regarding the unaligned couples were selected 395 couples. The selection procedure was based on a normal distribution of the average distance over the unaligned couples. The training set contains 290 couples of definitions (90 aligned and 200 unaligned) and the test set contains 285 couples of definitions (90 aligned and 195 unaligned). This database was created by the ASICOM's team.

4.2.1. *k Nearest Neighbour.* kNN receives as input the similarities between two definitions by using the weighted sum of six similarity measures and outputs the performed alignments. In order to select the aligned couples we identify:

- those with similarity > threshold (many-to-many alignments) and

- the best alignment from those with similarity $>$ threshold (one-to-many alignments).

The threshold was fixed 0.2 (default value from Coma++). The similarity value is represented by the weighted sum of the 6 distances presented in the first set of similarities. The weights are optimized on the training set, then validated on the test set by applying kNN with k=1.

4.2.2. *Support Vector Machine.* SVM receives as input the similarities between two definitions and outputs the label 1 if they are aligned and 0 if they are not aligned. The training set is built without mixing the couples of definitions, so first in the training set are taken the couples of aligned definitions and then the couples of unaligned definitions.

4.3. **Weights and parameters' optimization.** As we have previously mentioned, in the first experiment using kNN algorithm, the similarity is computed as a weighted sum of 6 distances. The weights are optimized using an evolutionary approach based on F-measure of aligned couples in the following manner: we consider a chromosome which codes the weights associated to the six distances. Its fitness is computed basing on the training set. Each definition from the first dictionary was compared with all the definitions from the second dictionary and was chosen the first couple with the greatest similarity value. If this similarity is greater than the threshold value, the couple is labelled as aligned, otherwise unaligned. In this manner 290 labelled couples are obtained. The fitness of the chromosome is represented by the F-measure of the aligned couples. The aligned couples are the ones that were considered aligned from the comparison with the threshold value and were initially labelled as aligned by Coma++, too. The best weights are the ones indicated by the chromosome with the greatest fitness value (F-measure). The best weights are learned on the training set and tested on the test set.

In the second experiment the SVM algorithm with an RBF kernel was used:

$$K(x, y) = exp(-\sigma|x - y|^2). \tag{1}$$

Cross Validation was made on the training set with different combinations of C and σ, C in range $[10^{-2}, 10^3]$ and σ in range $[2^{-5}, 2^2]$ by using a grid search algorithm and then the values from the combination (C and σ) with the best performances were chosen.

4.4. **Performance measures.** In order to measure the quality of the alignments the following performance measures were used: accuracy, recall, precision and F-measure, which are calculated for the aligned class (1) and the accuracy for both classes (1 and 0).

- the accuracy of alignments represents the percent of correct alignments from the total number of alignments;
- the precision of alignments represents the percent of relevant alignments among the proposed alignments;
- the recall of alignments represents the percent of relevant alignments that are effectively retrieved;
- F-measure represents the weighted harmonic mean of precision and recall. Greater F-measure (the maximal value being 1) signifies a correct and complete alignment.

4.5. **Numerical results.** In Table 1 the performances of the kNN and SVM based aligners are presented. The SVM classifier was applied at 3 different taxonomy levels while the kNN one utilizes a combination of them.

Table 1. The performance of the kNN and SVM-based aligners, using different similarity measures.

	Accuracy	Precision	Recall	F-measure
kNN	95%	87%	97%	92%
SVM on Explanation	32%	32%	100%	48%
SVM on Explanation & Path	90%	87%	81%	84%
SVM on Explanation & Path & Father	91%	87%	84%	86%

As we can notice from the results, by using the path and father of a concept it is possible to improve the performance of the classification process. The correct alignments were well recognized, but there are several not aligned definitions labelled as aligned by the SVM leading to a low precision, caused by: the couple type problems from our database like definitions with strong syntactic similarity and with different meaning (e.g. *Code specifying the type of package of an item* and *A code specifying a type of transport means*), the general-particular problem (e.g. *Means and mode of transport used for the carriage of the goods at departure, coded* and *A code specifying a type of transport means*) or words that appear in many definitions, but having different meaning.

The results using the kNN classifier on the definitions from this corpus are better than the ones using SVM, but, on the other hand, the utilization of the SVM classifier instead of the kNN one is more appropriate for solving the definition alignment problem taking into account that it can learn the optimal value of the threshold and it is able to produce any alignment type.

5. Conclusions

In this paper we presented our models for the automatic alignment of two terminologies (in fact, two real custom standards). These terminologies were reduced to several definitions taken from two dictionaries (CCL and WCO). The alignment issue was considered as a classification problem and solved by using two Machine Learning algorithms: kNN and SVM.

Even if the numerical results indicate that the SVM algorithm reaches weaker performances than the kNN method (from the F-measure performance point of view), the SVM is more helpful in order to align the given terminologies because it does not require fixing the values of the parameters. Furthermore, the SVM-based approach allows providing any type of alignments (one-to-one, one-to-many, many-to-many), which are very useful in the real world.

Future work will be focused on experiments considering a multi class problem. In this case we will deal with more types of definition couples: couples aligned and couples unaligned (straight unaligned, medium unaligned and weak unaligned).

References

[1] Barzilay, R., and Elhadad, N. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing* (2003).

[2] Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., and Toussaint, G. Output-sensitive algorithms for computing nearest-neighbor decision boundaries. *Discrete and Computational Geometry 4*, 33 (2005), 593–604.

[3] Brown, P. F., Pietra, S. D., Pietra, V. J. D., and Mercer, R. L. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics 19*, 2 (1994), 263–311.

[4] Ceausu, A., Stefanescu, D., and Tufis, D. Acquis communautaire sentence alignment using Support Vector Machines. In *Proceedings of the 5th LREC Conference* (2006), pp. 2134–2137.

[5] Chen, S. F. Aligning sentences in bilingual corpora using lexical information. In *Meeting of the Association for Computational Linguistics* (1993), ACL, pp. 9–16.

[6] Clemons, T. E., and Bradley, E. L. A nonparametric measure of the overlapping coefficient. *Comput. Stat. Data Anal. 34* (2000), 51–61.

[7] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms.* MIT Press, 1990.

[8] Croft, W. B. *Combining approaches to information retrieval.* Springer, 2000, ch. 1, pp. 1–36.

[9] Dice, L. Measures of the amount of ecologic association between species. *Ecology 26*, 3 (1945), 297–302.

[10] Do, H.-H., and Rahm, E. Coma++ (combination of schema matching approaches), 2010.

[11] Gale, W. A., and Church, K. W. A program for aligning sentences in bilingual corpora. In *ACL* (1991), MIT Press, pp. 177–184.

[12] Hai-Do, H., and Rahm, E. COMA - A system for flexible combination of schema matching approaches. In *VLDB* (2002), P. A. Bernstein, Ed., Morgan Kaufmann, pp. 610–621.

[13] Hatzivassiloglou, V., Klavans, J., and Eskin, E. Detecting text similarity over short passages: Exploring linguistic feature combination via machine learning. In *Proceedings of the joint SIGDAT Conference on EMNLP/VLC-1999* (1999).

[14] Jaccard, P. The distribution of the flora of the alpine zone. *New Phytologist 11* (1912), 37–50.

[15] Jing, H. Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics 28*, 4 (2002), 527–543.

[16] Knight, K., and Marcu, D. Statistics-based summarization - step one: Sentence compression. In *AAAI/IAAI* (2000), AAAI Press / The MIT Press, pp. 703–710.

[17] Madhavan, J., Bernstein, P. A., and Rahm, E. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases(VLDB '01)* (2001), P. M. G. Apers et al., Ed., Morgan Kaufmann, pp. 49–58.

[18] Moore, R. Fast and accurate sentence alignment of bilingual corpora. In *AMTA '02* (2002), S. D. Richardson, Ed., Springer, pp. 135–144.

[19] Moreau, F., Claveau, V., and Sébillot, P. Automatic morphological query expansion using analogy-based machine learning. In *ECIR 2007* (2007), G.Amati, C. Carpineto, and G. Romano, Eds., vol. 4425 of *LNCS*, Springer, pp. 222–233.

[20] Quirk, C., and Menezes, A. Dependency treelet translation: the convergence of statistical and example-based machine-translation? *Machine Translation 20*, 1 (2006), 43–65.

[21] Taskar, B., Lacoste, S., and Klein, D. A discriminative matching approach to word alignment. In *HLT '05* (2005), Association for Computational Linguistics, pp. 73–80.

[22] Vapnik, V. *The Nature of Statistical Learning Theory.* Springer, 1995.

[1]LITIS, EA - 4108, INSA, Rouen, France,

[2]Computer Science Department, Babeş Bolyai University, Cluj Napoca, Romania

E-mail address: `adela_sarbu25@yahoo.com, lauras@cs.ubbcluj.ro`

E-mail address: `arogozan@insa-rouen.fr, pecuchet@insa-rouen.fr`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

A FRAMEWORK FOR ACTIVE OBJECTS IN .NET

DAN MIRCEA SUCIU AND ALINA CUT

Abstract. Nowadays, the concern of computer science is to find new methodologies that help decomposing large programs and run them efficiently onto new parallel machines. Thus, the popularity of concurrent object-oriented programming has increased proportionally with the market requirements of concurrent and distributed systems that meet simplicity, modularity and code reusability. The purpose of this paper is to define a class library based on Active Object pattern introduced in [3], which has a high level of extensibility. Class library's main objective is to help in building concurrent object-oriented applications with a minimum effort and using a significant amount of already existing code. This approach addresses the problem of integrating concurrency with object-oriented programming and respects the principles imposed by them. In order to present the main features of our model a sample application is presented.

1. Introduction

An important motivation behind *concurrent object-oriented programming* (COOP) is to exploit the software reuse potential of object-oriented features in the development of concurrent systems [4]. *Object-oriented programming* (OOP) and *concurrent programming* (CP) unification seems natural if we think that real-world objects are indeed concurrent. On one hand, OOP has been developed having as a model our environment (seen as a set of objects among which several relationships exist and which communicate between them by message transmission). On the other hand, the concurrency between objects led to the normal trend of transposing this into programming. However, the integration of concurrency into OOP languages is not an easy task. The concurrent features of a language may interfere with its object-oriented features making them hard to integrate in a single language or cause many of

Received by the editors: August 6, 2010.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.3 [**Software**]: Software Engineering – *Coding, Tools and Techniques*; D.2.7 [**Software**]: Software Engineering – *Distribution, Maintenance and Enhancements* .

Key words and phrases. object-oriented concurrent programming, active objects.

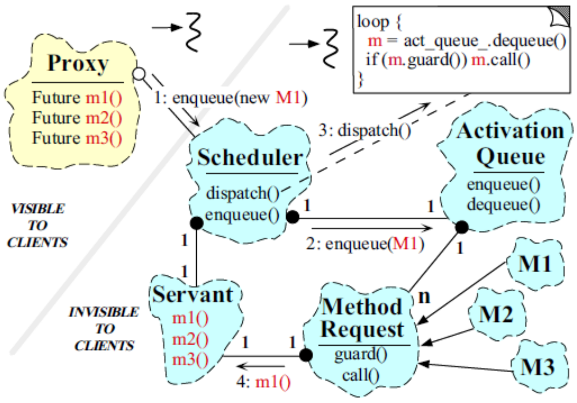
their benefits to be lost [4]. For this reason, we should carefully choose a proper mechanism for synchronization of concurrent objects.

Active object pattern is a high-level abstraction that simplifies CP and works on the object level. This paradigm proposes a new style of programming by decoupling method execution from method invocation in order to simplify synchronized access to an object that resides in its own thread of control [3]. Taking advantages of this pattern, we propose a general active object model which combines the reusability with the elegancy of integrating concurrency into OOP. Reusability of code is an important advantage of OOP that simplifies the development process by reducing the design and the coding. Later, in this paper, we present a sample that uses this specific active objects model for generating code from scalable statecharts [6].

2. Active Object model

Active object pattern comes to simplify the synchronization access to an object that is running in its own thread. The major difference imposed by this pattern is that it works on the object level not on an object hierarchy like most design patterns. This implies modeling classes as active classes with the implication that their operations are processed asynchronously and inherently thread-safe with respect to each other by processing at most one operation at any given time [2].

An active object has two important particularities: it runs in its own thread of control and the invoked methods don't block the caller but are executed asynchronously. Figure 1 illustrates the components of an active object as they are presented in [3].



```
loop {
  m = act_queue_.dequeue()
  if (m.guard()) m.call()
}
```

Figure 1. The components of the Active Object pattern [3]

We can see that an active object has two important parts according to the visibility property. The first part, which is visible to the client, contains a *proxy* which represents the public interface of an active object. It is responsible for accepting method calls or requests from clients (other objects that use an active object) and convert them into messages (*method requests*) that will be added into a *message queue*. The second part will contain the components that are hidden from the client. A *scheduler* is a special object which runs in the context of the active object thread. It maintains an *activation queue* with incoming messages. The scheduler, based on some criteria (the order in which the methods are inserted into the activation queue, some *guards*), will decide which message to dequeue in order to process it. After it processes the message, the scheduler will invoke the actual methods of the *servant*. The *servant* represents the private implementation of the active object. It encapsulates the data and defines the behavior and the state of the active object ([2]). In this manner, method invocation and method execution are decoupled and concurrency between objects is introduced.

Our proposed abstraction meets the properties of *active object pattern*: message-based property, asynchronous execution property and thread-safe property. The major concern was to develop a more general active object that allows us reuse as much code as possible and simplifies the development process. Figure 2 presents the components of our active object model and the relationships between them.

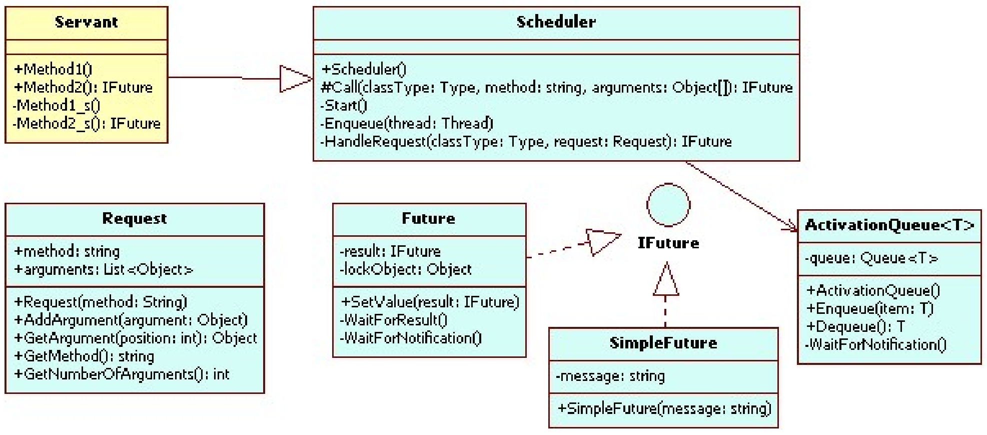


FIGURE 2. Extensible active object model

In order to obtain a model with a high extensibility we have decided to modify the structure of *active object pattern*. The first step in achieving our goal was to unify the *proxy* and the *servant*. In this case the *servant*, besides its regular activity, will also serve as a public interface for the client. It contains public methods that can be accessed by the clients, but these methods will only be used to forward a request to the scheduler. In this manner, the *servant* covers the behavior of a *proxy* presented in [3]. But, our *servant* also has some private methods (corresponding to the public methods) that represent the actual implementation of the services offered by this active object.

The second step to obtain a high extesibility was to define standard classes for the other components proposed in active object pattern (*scheduler*, *request*, *activation queue* and *future*) that offer us the possibility of reusing the code as many times as we need without any modification inside those classes. Our *scheduler*, as presented in figure 2, contains only four methods (three of them are private and one is protected) plus a public constructor. Although we have modified the structure of the *scheduler* to achieve our aim, it still meets all the functionalities of a scheduler presented in [3]. The protected method Call is the one that the *servant* calls each time a client makes a method call. This call is possible because the *servant* is a subclass of *scheduler*. Otherwise, the *servant* wouldn't be able to access a protected method of *scheduler*. The task of Call is to accept method calls, transform them into requests and add them, using Enqueue, into an activation queue. It will return an object IFuture that represents the place from where the client can read the result of his call. We already know that a *scheduler* runs in the context of the active object thread. The private method Start creates and starts the thread of the scheduler, a thread that will always try to handle the requests from the activation queue. HandleRequest is in fact the private method that will manage the requests. But how can it call the actual methods of the servant and still respect the standard that we want to obtain? Well, it uses reflection in order to call the proper methods of the servant when it knows the names of the methods (as strings) and the list of formal parameters.

Our proposed abstraction model takes into consideration the scenario when a client calls a method and waits a response from that method. We have modeled this by using future objects. A future object is a place where the active methods put their possible results. It can be considered a *rendezvous* for the caller and the active object ([2]). Once the result of a method is computed it will be stored in a future object and the caller can access the result from there. In the case that the caller tries to access the result before the method has computed it, the caller automatically blocks until the result is stored in the related future.

3. Sample - Robots Application

In order to prove the efficiency of our active object model, we developed a sample application implementing the behavior of a robot object, which is searching the exit of a maze using *the left-hand rule*. All objects (the robots and the maze components) used in the sample application are active objects: they have their own thread awaiting to receive and process method calls.

The robots are placed in the same table (representing the maze) and are sharing the same tracks. An important remark is that some tracks are blocked, meaning that there is a wall and the robot cannot access that tracks. When a robot meets a blocked track it should bypass it taking into consideration the *left hand side rule*. This rule assures that a robot tries to make left each time it meets an obstacle (maze margins or blocked tracks).

The directions of the robot are the ones corresponding to the four cardinal directions: *south* will be codified as direction 1, *east* will be direction 2, *north* will be codified with direction 3 and *west* will represent direction 4. So, each robot moves in the maze taking into consideration only these four directions.

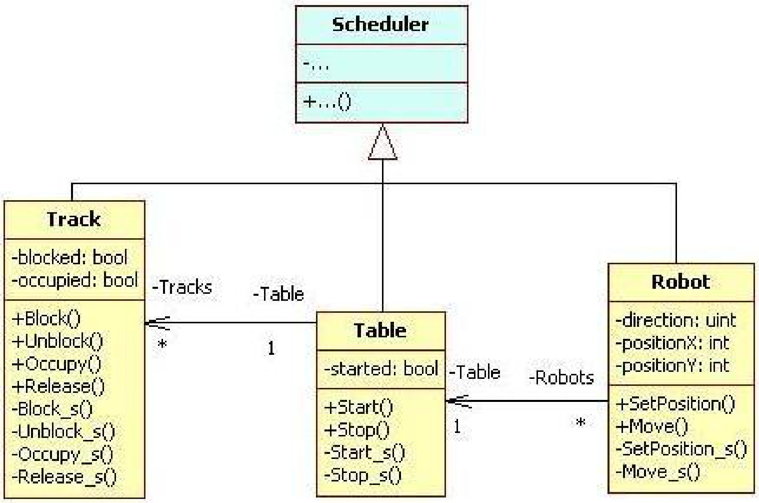


Figure 3. *Robots* application class diagram

Figure 3 contains the class diagram describing the structure of classes *Robot*, *Track* and *Table*. All these classes are derived from *Scheduler*, so they implement three distinct types of active objects. An important aspect is that no method implemented at Scheduler class level needs to be rewritten in its descendants. In other words, Track, Table and Robot objects are active objects (have their own execution thread, method queue and synchronization protocol) without implementing any element specific to concurrent programming, everything being inherited from Scheduler class. The only additions are referring to the implementation of their particular behavior.

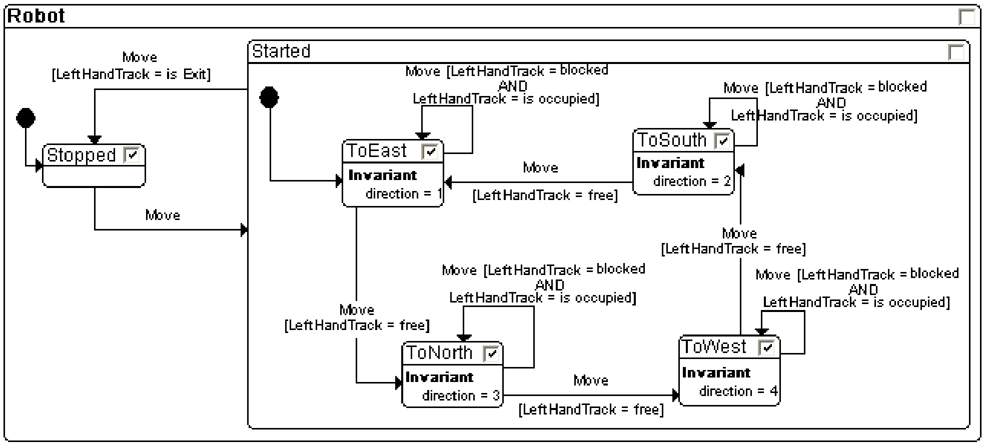


FIGURE 4. *Robot* class behaviour description using scalable statecharts

A robot keeps a direction and its coordinates on the table. It always makes a move to the right when the track is trying to conquer is blocked. In the case that a track is free then when a robot reaches it the track will become occupied.

For situations when two robots try to reach the same track we have considered that each time a robot occupies a cell, that track will also become blocked. From the robot point of view there is no difference between a blocked cell and an occupied cell. Of course that an occupied track will become free as soon as the robot leaves that track while a blocked track will remain blocked forever. When a robot tries to walk into an occupied track its move fails. In this situation the robot will make a move to the right from its position similar to the case in which that track is blocked.

Figure 4 shows the behavior model of robot objects defined with scalable statecharts created using *ActiveCASE* tool ([5]). Moreover, *ActiveCASE* tool was modified accordingly to support the proposed active object model and to generate source code based on it.

Figure 5 shows a screen shot of the *Robots* application.

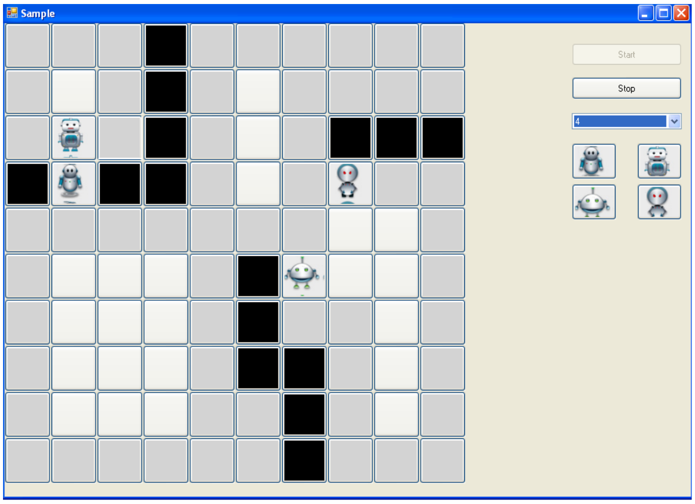


FIGURE 5. *Robots* application screen shot

4. Conclusions

The popularity of COOP is increasing as the concurrency is becoming a required component of ever more types of systems. *Active object pattern* offers us an elegant way of decoupling method invocation from method execution. Based on this pattern we have obtained a more general model by trying to reorganize the structure of a regular active object and keep the functionalities of its components. This type of active object is characterized by a higher extensibility and a larger amount of reusable code while meeting the base properties of the *active object pattern* proposed in .

Starting from the original model of an active object presented in [3] we have focused on developing an abstract C# library for active objects. Our aim was to reduce the number of classes that should be implemented for an active object, in order to simplify the work with this type of objects. For achieving this goal we have made some modifications in the structure of *active object pattern*.

Some important features of our library are:

- the concurrent programming aspects are handled exclusively at framework level. All future descendant classes will take care only about logic implementation of their behavior, without taking into account parallel execution or synchronization of methods,
- the higher extensibility offered by the abstraction of the classes,

- the reusability of code possible because of the inheritance relation introduced between the servant and the scheduler
- the flexibility provided by the easy adaption to the external changes in the implementation of the services.

The features mentioned above help to reduce the effects of inheritance anomalies that characterize COOPL, as described in [1]. We have also introduced the possibility of working with guards when trying to execute a method. A guard is a constraint that should be satisfied before its associated method can be executed.

We have decided to demonstrate the applicability and the efficiency of our framework by developing a sample application based on our model. The sample aplication presents the moves of some robots in a labyrinth with obstacles.

Future improvements of our library may introduce priorities at the level of the services offered by an active object. Taking into consideration this idea, a scheduler will execute the methods according to their priorities. This implies an improvement at the level of the activation queue. In this manner, instead of a simple queue a priority queue may be used.

References

[1] Jean-Pierre Briot , Akinori Yonezawa, "Inheritance and Synchronization in Concurrent OOP", European Conference on Object-Oriented Programming (ECOOP87), LNCS 276, pp. 3240, 1987.

[2] Tobias Gurok, "Active Objects and Futures: A Concurrency Abstraction Implemented for C# and .NET", Fakultat fur Elektrotechnik, Informatik und Mathematik, Universitat Paderborn, Bachelor Thesis, 2007.

[3] R. Greg Lavender, Douglas C. Schmidt, "Active Object: an object behavioral pattern for concurrent programming", Pattern languages of program design, Addison-Wesley Longman Publishing Co, Boston, USA, pp. 483-499, 1996

[4] Michael Papathomas, "Concurrency Issues in Object-Oriented Programming Languages", in Object Oriented Development, ed. D. Tsichritzis, Centre Universitaire dInformatique, University of Geneva, pp. 207-245, 1989

[5] Dan Mircea Suciu, "ActiveCASE Tool for Design and Simulation of Concurrent Object-Oriented Applications", Studia Universitatis Babes Bolyai, Informatica, Vol. XLVI, No. 2, 2001, pp. 73-80

[6] Dan Mircea Suciu, "Reverse Engineering and Simulation of Active Objects Behavior", Knowledge Engineering, Principles and Techniques - KEPT-2009 Selected Papers, Babes-Bolyai University of Cluj-Napoca, July 2-4 2009, pp. 283-290

Department of Computer Science, "Babeş-Bolyai" University, 1 M. Kogălniceanu St., RO-400084 Cluj-Napoca, Romania

E-mail address: tzutzu@cs.ubbcluj.ro, alina_cut@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

TEACHING MODEL CHECKING TO UNDERGRADUATES

A.VESCAN AND M. FRENŢIU

Abstract. The way program verification is taught in our faculty is firstly described. One of the verification methods is model checking, shortly presented to the students in one lecture. One laboratory project consists in using the SPIN tool. The difficulties encountered by students with this project are presented in this paper.

1. Introduction

The need for more reliable software [1, 9, 16, 6], and the role of Formal Methods [9, 3, 14, 19, 7] are well known. During the last two decades, research in formal methods has led to the development of some very promising techniques that facilitate the early detection of defects. These techniques are accompanied by powerful software tools that can be used to automate various verification steps. Investigations have shown that formal verification procedures would have revealed the exposed defect in, e. g., the Ariane-5 missile, Mars Pathfinder, Intel's Pentium II processor, or the Therac-25 therapy radiation machine [4].

Today, the computers are used in all fields of human activities. More and more programs are needed, and software critical systems require error-free programming [1].

If we want to build software of good quality, to increase its reliability, we have to contribute to better education of human resources. It is considered that the main barrier against the usage of Formal Methods is the lack of good professionals, able to use such methods.

Each university must educate better software engineers capable to use the newest methods, which increase the quality of software products and improves the software processes. In this direction, one of the important subject that must be taught to undergraduates is Verification and Validation.

Received by the editors: July 18,2010.

2010 *Mathematics Subject Classification.* 68Q60, 68N30.

1998 *CR Categories and Descriptors.* D.2.4 [**Software**]: Software engineering – *Software/Program Verification*; D.2.5 [**Software**]: Software engineering – *Testing and Debugging*.

Key words and phrases. verification and validation, model checking, tools, education.

2. The role of Verification and Validation (V&V)

One means to Software Quality Assurance is V&V. The course V&V is one of the important courses that contributes to obtain well-educated practitioners. We teach such a course to the third year undergraduates. The theoretical basis for building reliable software products is given here. The course consists of three main parts:

- the theory of program correctness;
- the methods of verification and validation;
- the consequences on software engineering practice.

The entire curricula of this course, and also, the undergraduate study program may be seen at [20].

One cannot understand V&V if he does not know the concept of program correctness. The first part of the course presents this concept and gives methods to prove correctness. More important, the accent is put on the methods to achieve this correctness.

The methods discussed in the second part are: proving correctness, testing, inspection, symbolic execution, and model checking. It is underlined that all of them must be practiced during the software process [8]. Their usage may be informal, or more formal, complete or only some of them, depending on the type of the system which is built. For safety-critical systems all of the above mentioned methods must be used. We consider that all future software engineers should be aware of all verification methods.

The third part of the course presents the Cleanroom methodology [17], the role of V&V for Software Quality Assurance and Software Process Improvement, and the consequences of correctness theory on software engineering practice [5, 6, 8].

At the undergraduate level we cannot afford to teach the mathematical basis of model checking, the theory that lies at the basis of model checking theory. Instead, the main theoretical aspects are presented in a two hours lecture, and the existence of tools and examples of such tools are shortly presented.

Model checking [12] is a verification technique that explores all possible system states in a brute-force manner. In this way, it can be shown that a given system model truly satisfies a certain property. The property specification prescribes what the system should do, or what it should not do, whereas the model description addresses how the system behaves. The model checker examines all relevant system states to check whether they satisfy the desired property. To make a rigorous verification possible, properties should be described in a precise and unambiguous manner. A temporal logic, which is a form of modal logic that is appropriate to specify relevant properties, is used as a property specification language. In term of mathematical logic, one checks that the system description is a model of a temporal logic formula. Temporal logic is basically an extension of traditional propositional logic with operators that refer to the behavior of systems over time. It allows for the specification of a broad range of relevant system properties [12] such as functional correctness (does the system do what it is supposed to do?), reachability (is it possible to end up in a deadlock state?), safety ("something bad never happens"), liveness ("something good will eventually happen"), fairness (does, under certain conditions, an event occur repeatedly?), and real-time properties (is the system acting in time?). Also, model checking may be used to check the conformance of design with the requirements [2].

Teaching model checking to undergraduates was already proposed by others [13, 18]. The undergraduate curricula cannot contain an entire course about model checking, but we consider that it is an important verification method and must be present in such a course.

3. Problems with teaching model checking

As a laboratory project one tool was presented to the students and this tool was Spin [10].

The language Promela was presented and introduced to students during the first part of the laboratory. Several examples [15] were presented and explained to better understand the syntax of the language but more than that, the semantics of the structures were also explained. The non determinacy was explained by using several examples. The notion of a process in Promela was presented and discussed. The concurrency, interference and interleaving between processes were described. The students played with the provided examples [15] and experienced the concurrency and interleaving. Several ways/methods for deterministic steps/atomic executions of statements of different processes were explained. So, one of the difficulties encountered with teaching model checking was the unfamiliarity with concurrent systems.

The structure of the laboratory consisting in model checking may be found here [21]. The laboratory consists in two hours and for the assignment the students had two problems: in class assignment problem and homework assignment problem.

The properties to be checked were first expressed using assertions. As in class assignment the students had to implement in Promela a process to compute a given value and then to use assertions to establish the correctness of the computation. They also had to use assertions to express preconditions and postconditions. The majority of them were able to work alone, without any additional help. This was a simple exercise to help student create a process and use assertions for correctness, and also to prepare them for the homework assignment problem. They have already used assertions (preconditions, post-conditions, invariants) in a previous laboratory [21] when they use ESCJAVA and JML [11].

Another problem faced by some of the students was to make them model a system in class, immediately after the presentation of similar examples. Obviously, they need more time to process and understand modeling using Promela and thinking about verification of a system/model using a model checker.

For the second part of the class we have discussed the use of LTL formula to express the properties that the model should have. They have run the prepared examples [15]: about critical section in two processes, about deadlock and starvation. They have used LTL formula to express these properties and used the JSpin tool to verify them. During the use of JSpin tool with LTL formula the students were very enthusiastic about the "power" of the tool, especially about the checking process.

For homework assignment they have received the following problem: Consider the frog pond shown in Figure 1. Three female frogs are on the three stones on the left and three male frogs are on the three stones on the right. Find a way to exchange the positions of the male and female frogs, so that the male frogs are all on the left and the females are all on the right. The constraints that your solution must satisfy are as follows: frogs can only jump in the direction they are facing. They can either jump one rock forward if the next rock is empty or they can jump over a frog if the next rock has a frog on it and the rock after it is empty. Model the above system using a Spin model, and show that it is possible to reach the desired end state.

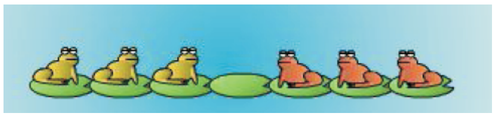


FIGURE 1. The Frog Pond Puzzle

The students played the game and tried to find a solution using [22]. Some of them found the solutin quickly and were able to sketch an algorithm for the solution.

Only a few of the students were able to model the system and check for solution, some of them used assertions and others used a LTL formula. Only a few of them understood that the property is checked in all states of the system model. They were very satisfied about the outcome of their finding.

Other students didn't understand correctly what they should do and they modeled the solution of the problem and not the system and the rules. They were very excited that the JSpin model checker always "gave" them the solution, and each time they run (randomly execution) the created processes they reached the solution!

Other students found the solution of the problem in the JSpin example directory but when asked about the model, the way that the model should be used in JSpin, they didn't know what to answer. They were not able to explain (even explained during the previous laboratory) how the model is used and how the model checker verify the LTL formula.

4. Conclusions

Verification by Model Checking had motivated the students, although they met the above difficulties. Since the allocated time for this subject was small, the examples were small and they could be considered as toys examples. Nevertheless, they offer to the students the possibility to acquire this new method of verification. They saw that Model Checking is a good mean to catch errors earlier in the model, to eliminate the rework, and to improve the quality of the product and the software process.

Nevertheless, we can improve this part of the course by choosing more suitable examples and give them to the students as homework projects.

Also, we must insist on improving the (first) model chosen by students. And, as well, we must insist that the students should pay attention to a broadly verification of a system, at least in the following three directions:

- to use all verification methods;
- to carefully design the testing cases according to the chosen criteria;
- to verify the robustness of the system.

References

[1] R. W. Butler, S. C. Johnson, *Formal Methods for Life-Critical Software*, in Computing in Aerospace 9 Conference, San Diego, California, 1993, pp. 319–329.

[2] M. Chechik, J. Gannon, *Automating Analysis of Consistency between Requirements and Designs*, IEEE Transactions on Software Engineering, 27(2001), no.7, pp.1–21.

[3] E. M. Clarke, J.M. Wing, *Formal Methods: State of the Art and Future Directions*, ACM Computing Surveys, 28 (1996), no. 4, pp. 626–643.

[4] N. Dershowitz, *Software Horror Stories*, `www.cs.tau.ac.il/~nachumd/horror.html`.

[5] M. Frentiu, *On Program Correctness and Teaching Programming*, Computer Science Journal of Moldova, 5(1997), no.3, pp. 250–260.

[6] M. Frentiu, *Correctness, a very important quality factor in programming*, Studia Univ. "Babe-Bolyai", Seria Informatica, L(2005), no.1, pp. 12–21.

[7] M. Frentiu, *The Need to Teach Formal Methods*, Analele Universităţii Bucureşti, LV, 2006.

[8] M. Frentiu, *Verificarea si Validarea Sistemelor*, Ed. Presa Universitara Clujeana, Cluj-Napoca, 2010, pp. 232, ISBN 978-973-610-979-9.

[9] C. M. Holloway, *Why Engineers Should Consider Formal Methods*, in 16th AIAA/IEEE Digital Avionics Systems Conference, Volume 1, 1997, pp. 1.3-16 – 1.3-22.

[10] G. J. Holzman, *The Model Checker SPIN*, IEEE Transactions on Software Engineering, 23(1997), no.5, pp. 279-295.

[11] JML, Java Modeling Language Home Page, `http://www.eecs.ucf.edu/~leavens/JML/`

[12] J. P. Katoen, *Principles of Model Checking*, MIT Press, 2008, pp. 995.

[13] H. Liu, D.P. Gluch, *A proposal for introducing model checking into an undergraduate software engineering curriculum*, Journal of Computing Sciences in Colleges, 18(2002), no. 2, pp.259–270.

[14] M. J. Lutz, *Alloy, Software Engineering, and Undergraduate Education*, in First Alloy Workshop, colocated with the Fourteenth ACM SIGSOFT Symposium on Foundations of Software Engineering, Portland, 2006, pp. 96–97.

[15] B. R. Mordechai, *Principles of the Spin Model Checker*, ISBN: 978-1-84628-769-5, 2008, pp. 216.

[16] B. Meyer,*Software Engineering in the Academy*, IEEE Computer, 34 (2001), pp. 28–35.

[17] H. Mills, M. Dyer, R.Linger, *Cleanroom Software Engineering*, IEEE Software, 4(1987), no.5, pp.19–25.

[18] H. Nishihara, K. Shinozaki, K. Hayamizu, T. Aoki, K. Taguchi, F. Kumeno, *Model Checking education for Software Engineering in Japan*, ACM SIGCSE Bulletin - COLUMN: Special section on formal methods education and training, 41(2009), no. 2, pp. 45–50.

[19] L. Yilmaz, S. Wang, *Integrating Model-Based Verification into Software Design Education*, Journal of STEM Education, vol.6 (2005), no.3–4, pp.29–34.

[20] Undergraduate study program - Babes-Bolyai University, `www.cs.ubbcluj.ro`

[21] A. Vescan, Software Systems Verification and Validation - course, seminar, laboratory - `http://www.cs.ubbcluj.ro/~avescan/`

[22] The Frog Pond Puzzle, `http://www.hellam.net/maths2000/frogs.html`

Department of Computer Science, Faculty of Mathematics and Computer Science,, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: {`avescan,mfrentiu`}`@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

CURRENT EXTENSIONS ON PULSE

SANDA DRAGOS

Abstract. Using a learning management system (LMS) is a common practise nowadays. Such instruments are used in educational institutions to enhance and support the teaching act as well as in industry for training purposes. In a computer science department of an university such instrument tends to be a basic requirement. That is because not only it allows a better management of courses and a better communication between students and professors, but can also serve as a perfect instrument for presenting teaching related materials for computer science subjects.

During the years I have created and used several such instruments: a **S**ystem with **I**nteractive ack**N**owledgement and **E**valuation of students work during laboratory sessions (SINE) [8, 9], a **P**hp **U**tility used in **L**aboratories for **S**tudent **E**valuation (PULSE) [6], and PULSE Extended [7]. The aim of this paper is to present the current enhancements of PULSE.

1. Introduction

Most learning management systems [1, 2, 5, 10, 11, 13] come as a substitute of a human tutor being very well suited for individual study and on-line auto-evaluation based on test-quizzes. Within laboratory sessions, however, the work is more consistent and aims a complex comprehension of new concepts. The evaluation aims to quantify the level of understanding gained by a student as a result of solving the assigned problem. Thus, the evaluation cannot be done on-line but as a discussion with each student, during which the misunderstood or unacknowledged concepts can still be clarified. Thus our systems comes as a complementary tool to the tutor's work.

There is a large number (both open source and commercial) of learning management systems (LMSs). A comprehensive list of them is presented in [10].

Received by the editors: August 6, 2010.

2010 *Mathematics Subject Classification.* 68N99, 68P30.

1998 *CR Categories and Descriptors.* K.3.1 [**Computing Milieux**]: Computer and Education – *Computer Uses in Education*; H.3.5 [**Information Systems**]: Information Storage and Retrieval – *On-line Information Services*.

Key words and phrases. Learning Management Systems, Web Programming.

Most open source LMSs are web-based, multi-language, cross-platform applications which support learning content management and tools for collaboration, communication, evaluation and assessment. Amongst them are OLAT (Online Learning And Training) [13] which is a Java-based component oriented application developed by the University of Zurich; Dokeos [5] which is based on per group management; and ILIAS [12] which is a complex LMS with multiple features: personal desktop, course management, cooperation (group management, file sharing), communication (internal messages, chat, forum, podcasting), test/assessment (quizzes), evaluation (survey, reporting and analysis) and administration.

Examples of commercial LMSs are Apex Learning [1], which offers courses in mathematics, science, English studies, social studies, foreign languages (e.g. French, Spanish), and Advanced Placement; Blackboard [2], which is mainly focused on academic e-learning; and CLIX [11], which offers different packages for enterprizes, universities and schools.

All these LMSs are complex applications which contain many features, some of which (e.g. work sharing) do not serve our purpose.

Similar applications to the one described in this paper and to its predecessors[1] have been developed and are used in our department. One such application is called *Assignment Management System (AMS)* [3, 4].

2. Background

SINE (a System with Interactive ackNowledgement and Evaluation of students work during laboratory sessions) [8, 9] was my previous instrument created for acknowledging and evaluating student work during laboratory sessions. It was a Linux-based instrument that recorded all e-mails sent by students (using procmail) and sending back automatic notification e-mails and changing correspondingly a web interface. Private information such as marks, observations and access to assignment sources were restricted by one password (the tutor's password).

This instrument was used, between 2005 and 2006, for four academic semesters during *Operating Systems*[2] and *Computer Architecture*[3] laboratories, at the Faculty of Mathematics and Computer Science of "Babes-Bolyai" University, Cluj-Napoca, Romania.

PULSE (a **P**hp **U**tility used in **L**aboratories for **S**tudent **E**valuation) [6] was the next instrument that replaced SINE. Its main feature consist in per

[1]The predecessors of the application described in this paper are presented in Section 2.

[2]The number of students that used SINE between 2005 and 2006 during *Operating Systems* laboratories were around 220.

[3]Around 80 students used SINE between 2005 and 2006 during *Computer Architecture* laboratories.

user authentication. The student interface was the main focus of this first version of PULSE, offering details such as: marks, attendances, assignments, theoretical support for new concepts and applicability examples. Tutors interface was the list of students with their marks, attendances and final marks. These lists could be sorted alphabetically, by the average of laboratory marks, by marks of practical exam or by final mark. The tutor's web interface still lacked the administrative part of marking attendances and assignments, which were done by Linux shell scripts.

PULSE was successfully used, between 2007 and 2008, for three academic semesters during *Operating Systems*[4], *Collective Programming*[5] and *Computer Architecture*[6] laboratories and seminars, at the Faculty of Mathematics and Computer Science of "Babes-Bolyai" University, Cluj-Napoca, Romania.

PULSE extended [7] came as an "extension" to the first version of PULSE, focusing on the administrative part of it. Thus, the admin interface contains now a form that allows marking assignment solutions, and another form for marking student attendances to a laboratory session. It also allows creating custom average mark calculation and the input of final exam marks. All these are used in the automatic computation of the final mark. The admin can also post announcements/news in a RSS format in English and/or in Romanian.

Some improvements were done in the student interface as well. Breadcrumbs were added for better navigation. More details are presented in assignment and laboratory session. News are managed as RSS 2.0 feeds which can be accessed by all PULSE actors by viewing them within the PULSE environment, or subscribing to them by using any convenient reader.

This new version of PULSE was used between 2008 and 2009 during *Web Programming*[7] and *Operating Systems*[8] laboratories, seminars and lectures, at the Faculty of Mathematics and Computer Science of "Babes-Bolyai" University, Cluj-Napoca, Romania.

This paper presents the new improvements to this instrument (that we continue to call PULSE) which were made in the course of the last academic year (i.e., 2009-2010).

[4]The number of students that used PULSE between 2007 and 2008 during *Operating Systems* laboratories were around 60.

[5]A subgroup (i.e., around 15 students) used PULSE between 2007 and 2008 during *Collective Programming* laboratories.

[6]Around 60 students used PULSE between 2007 and 2008 during *Computer Architecture* laboratories and seminars.

[7]The number of students that used PULSE extended between 2008 and 2009 during *Web Programming* laboratories were around 80.

[8]Both sections of *Mathematics and Computer Science* and *Applied Mathematics* (i.e., around 60 students) used PULSE extended between 2008 and 2009 for *Operating Systems* laboratories, seminars and lectures.

3. New improvements on PULSE

Along with the new layout and the new graphical design, PULSE got improved on all three interfaces corresponding to the tree types of actors. As in the previous versions, there are three types of actors using PULSE:

Students to attend the lab: They are the students that are assigned to attend the specific lecture, seminar and/or laboratory.

Tutors: Persons that are related to that specific lecture, seminar and/or laboratory but they are not students or the lecturer and/or the lab instructor

The admin: Is the lecturer and/or the lab instructor, or the person conducting the lecture, seminar and/or laboratory.

3.1. **The student interface of PULSE.** As depicted in Figure 1, the new layout of PULSE is more compact in design than the previous versions being better viewed on any web browser and any devices from a desktop computer to a mobile device such as a smartphone or tablet PC.

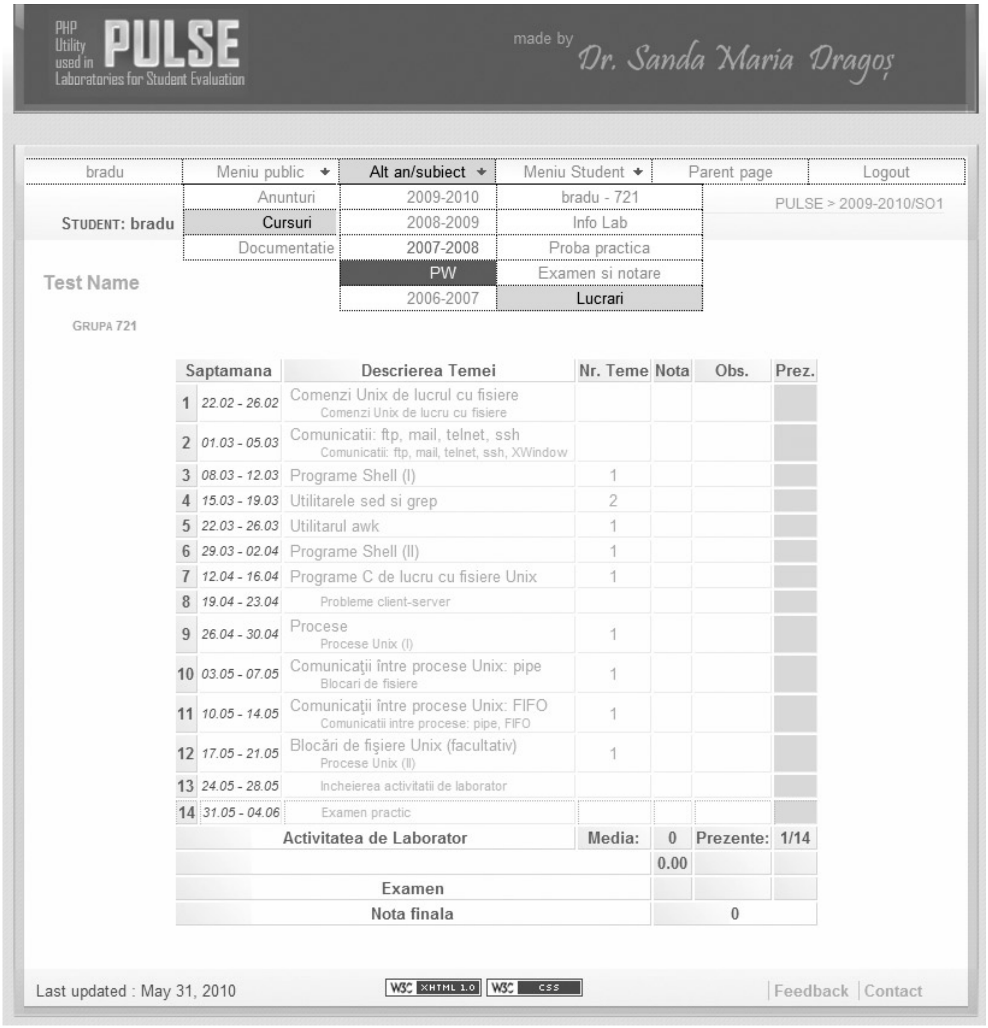
This new layout also contains a 2-layer menu. The first entry on the menu is a shortcut to the student information (e.g., marks, assignments, attendances). The second entry groups the additional information regarding the current subject. The current subject can be changed by using the next entry of the main menu. The forth entry contains specific information regarding that student. The last two entries are a link to the parent page and the logout button.

The new additions in the student interface, highlighted in Figure 1, are:

- Lectures. The second entry on the first submenu leads to a page that contains information and theoretical support for all lectures (held or to be held).
- Subject change. The third entry in the main menu contains a submenu automatically generated by the system which contains all subjects that the current student attended. After selecting one of the academic years, this submenu expands with the actual subjects studied by him/her that year. Selecting one of those subjects results in accessing the information related only to that specific year/subject.
- Lecture papers. The last entry in the last submenu leads to a page containing information about the test papers given during lecture sessions.

Part of the "Lecture" page is presented in Figure 2 and contains in this case the theoretical support for that lecture:

(1) the slides that can be viewed as a .pdf file;
(2) the slides that can be viewed as a webpage;



Saptamana		Descrierea Temei	Nr. Teme	Nota	Obs.	Prez.
1	*22.02 - 26.02*	Comenzi Unix de lucrul cu fisiere Comenzi Unix de lucru cu fisiere				
2	*01.03 - 05.03*	Comunicatii: ftp, mail, telnet, ssh Comunicatii: ftp, mail, telnet, ssh, XWindow				
3	*08.03 - 12.03*	Programe Shell (I)	1			
4	*15.03 - 19.03*	Utilitarele sed si grep	2			
5	*22.03 - 26.03*	Utilitarul awk	1			
6	*29.03 - 02.04*	Programe Shell (II)	1			
7	*12.04 - 16.04*	Programe C de lucru cu fisiere Unix	1			
8	*19.04 - 23.04*	Probleme client-server				
9	*26.04 - 30.04*	Procese Procese Unix (I)	1			
10	*03.05 - 07.05*	Comunicaţii între procese Unix: pipe Blocari de fisiere	1			
11	*10.05 - 14.05*	Comunicaţii între procese Unix: FIFO Comunicatii intre procese: pipe, FIFO	1			
12	*17.05 - 21.05*	Blocări de fişiere Unix (facultativ) Procese Unix (II)	1			
13	*24.05 - 28.05*	Incheierea activitatii de laborator				
14	*31.05 - 04.06*	Examen practic				
		Activitatea de Laborator	**Media:**	**0**	**Prezente:**	**1/14**
				0.00		
		Examen				
		Nota finala			**0**	

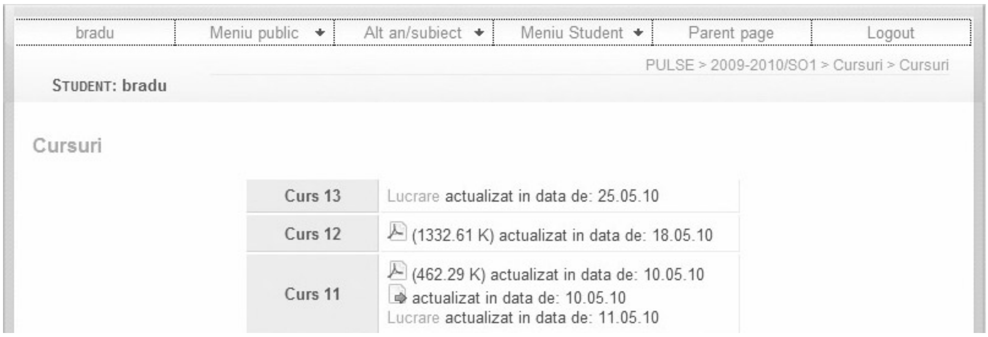
Last updated : May 31, 2010

FIGURE 1. The student interface of PULSE

(3) another link that leads to a webpage containing lecture paper requirements and solutions if any such test was given during that lecture.

The "Lecture paper" page is a webpage containing information about lecture papers results/marks for the current student. First, there are presented general information regarding all given lecture papers, such as:

- the number of lecture papers that were given up to this moment,
- the number of lecture papers that were taken by the current student,
- the average mark obtained by the student,



STUDENT: bradu

Cursuri

Curs 13	Lucrare actualizat in data de: 25.05.10
Curs 12	(1332.61 K) actualizat in data de: 18.05.10
Curs 11	(462.29 K) actualizat in data de: 10.05.10 actualizat in data de: 10.05.10 Lucrare actualizat in data de: 11.05.10

FIGURE 2. The page containing information about lectures

- how this can help him to the final mark.

Next, for each lecture paper, the student is presented with more information:

- the date in which the lecture paper was given;
- a link to the webpage containing lecture paper requirements and solutions;
- his/her mark, or the message “You did not take this lecture paper!” if no mark exists;
- a *graphical distribution*[9] of marks obtained by students which took the lecture paper.

The graphical distribution of marks helps the student to observe were he/she stands in the case in which he/she took the lecture paper. Otherwise, he/she can see how the other students were able to solve the issues that they were presented with.

3.2. The tutor interface of PULSE.

Figure 3 presents the menu in the tutor interface of PULSE. The main additions here are somehow similar to the ones in the student interface: the lecture button, the capability of changing the subject tutored by the logged-in person, and a “list” of lecture papers.

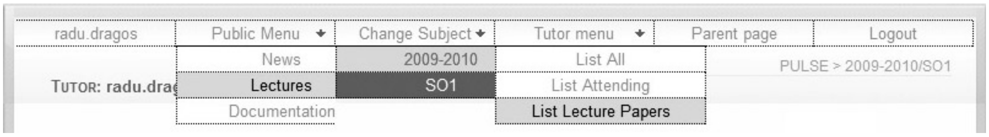
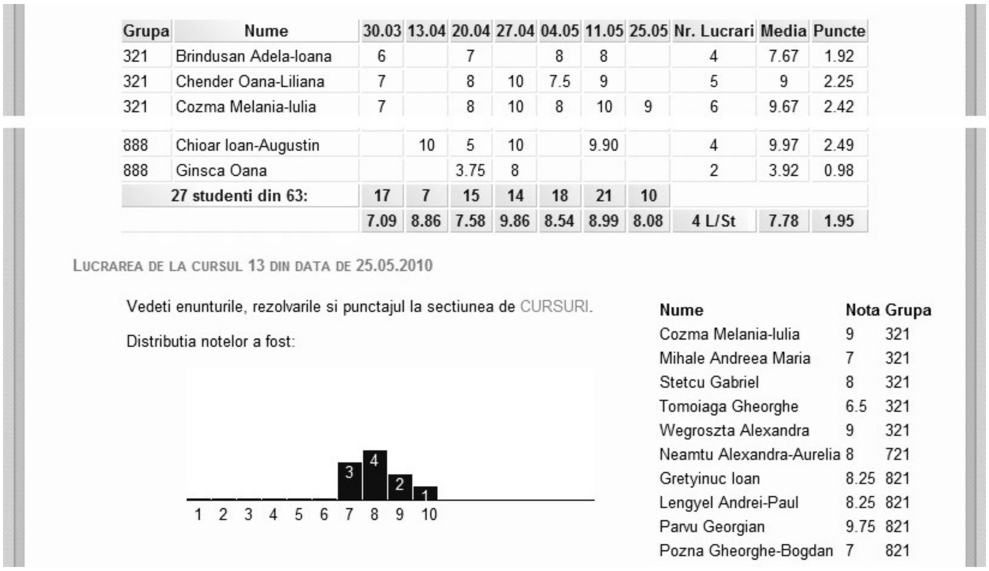


FIGURE 3. The menu in the tutor interface of PULSE

[9]This *graphical distribution* is similar with the one presented at the bottom of Figure 4.

The most important improvement here is the page containing the list of lecture papers as presented in Figure 4.



Grupa	Nume	30.03	13.04	20.04	27.04	04.05	11.05	25.05	Nr. Lucrari	Media	Puncte
321	Brindusan Adela-Ioana	6		7		8	8		4	7.67	1.92
321	Chender Oana-Liliana	7		8	10	7.5	9		5	9	2.25
321	Cozma Melania-Iulia	7		8	10	8	10	9	6	9.67	2.42
888	Chioar Ioan-Augustin		10	5	10		9.90		4	9.97	2.49
888	Ginsca Oana			3.75	8				2	3.92	0.98
27 studenti din 63:		17	7	15	14	18	21	10			
		7.09	8.86	7.58	9.86	8.54	8.99	8.08	4 L/St	7.78	1.95

LUCRAREA DE LA CURSUL 13 DIN DATA DE 25.05.2010

Vedeti enunturile, rezolvarile si punctajul la sectiunea de CURSURI.

Distributia notelor a fost:

Nume	Nota	Grupa
Cozma Melania-Iulia	9	321
Mihale Andreea Maria	7	321
Stetcu Gabriel	8	321
Tomoiaga Gheorghe	6.5	321
Wegroszta Alexandra	9	321
Neamtu Alexandra-Aurelia	8	721
Gretyinuc Ioan	8.25	821
Lengyel Andrei-Paul	8.25	821
Parvu Georgian	9.75	821
Pozna Gheorghe-Bogdan	7	821

FIGURE 4. Sections from the webpage containing details about the lecture papers

The first element on this webpage is a table containing all the students that took at least one lecture paper. The information in the columns of this table are: the group, the name of the student, the mark for each[10] lecture paper given by that student, the number of lecture papers taken by the student, the average mark, and the points for final mark. The last two lines on this table contain:

- the number of students that took at least one lecture paper and the number of all students registered for the current subject,
- the number of students and the average mark for each lecture paper,
- the average number of lecture papers taken per student,
- the average of students average marks,
- and the average of the points obtained.

Then, for each lecture paper, there is presented the same graphical distribution of marks obtained by students which took the lecture paper as in the student case, and the list of students that took that lecture paper along with their marks.

[10]There is a column for each lecture paper. Each cell of those columns contains the corresponding mark.

3.3. **The admin interface of PULSE.** The admin interface of PULSE contains all the information in the tutor interface. In addition it also contains the administrative capabilities. As shown in Figure 5, the main difference from the two previous interface is the "Add Assignments" entry from the last submenu.

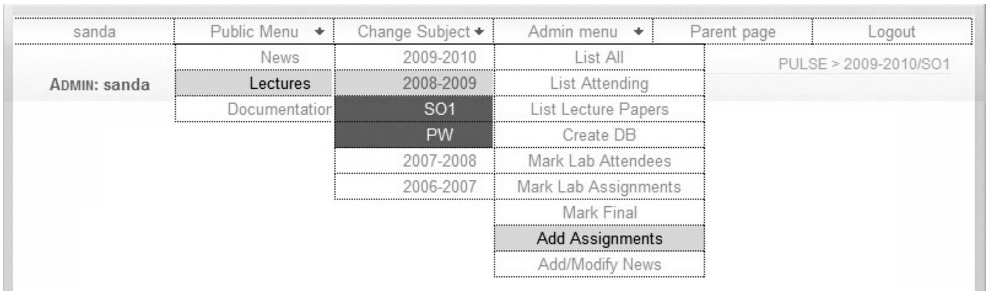


FIGURE 5. The menu in the admin interface of PULSE

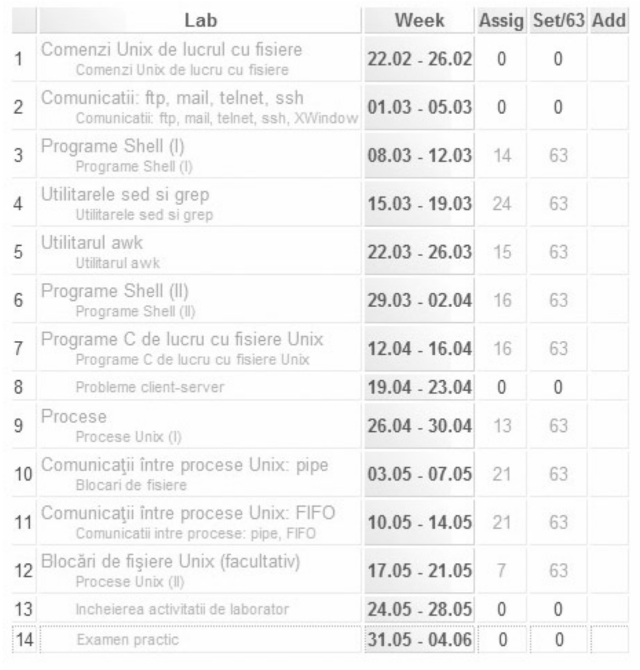
In each laboratory session a new concept can be introduced. In order to assimilate the new concepts, students are given assignments. Each concept has a pool of assignments from which one (or more) are randomly chosen for each student. The pool of assignments for each new concept are stored in distinct XML files. PULSE detects the existence of such file and creates the page presented in Figure 6, where are listed all laboratory sessions along with the new concepts they introduce and the number of assignments existing in the pool for that concept.

The penultimate column of the table presented here contains the number of students to which an assignment was already assigned. The actual assignment is performed when there is a pool of assignments[11] and not all student are assigned with one[12]. In this situation the last column contains a '+' sign. If this sign is selected the students that do not have an assignment are automatically assigned a randomly chosen one from the pool.

The content in the previous two columns can also be selected if that number is greater than zero. The links in the "Assig" column lead to webpages that contain corresponding assignment pools (i.e., the requirements). The links in the penultimate column lead to webpages that present the actual assignments of elements from the corresponding pool to each student.

[11]If the number in the "Assig" column is greater than zero.

[12]This is the case when the number in the penultimate column is less than *the total number of students* registered for the current subject. *The total number of students* registered for the current subject is presented in the table head of this column.



	Lab	Week	Assig	Set/63	Add
1	Comenzi Unix de lucrul cu fisiere Comenzi Unix de lucru cu fisiere	22.02 - 26.02	0	0	
2	Comunicatii: ftp, mail, telnet, ssh Comunicatii: ftp, mail, telnet, ssh, XWindow	01.03 - 05.03	0	0	
3	Programe Shell (I) Programe Shell (I)	08.03 - 12.03	14	63	
4	Utilitarele sed si grep Utilitarele sed si grep	15.03 - 19.03	24	63	
5	Utilitarul awk Utilitarul awk	22.03 - 26.03	15	63	
6	Programe Shell (II) Programe Shell (II)	29.03 - 02.04	16	63	
7	Programe C de lucru cu fisiere Unix Programe C de lucru cu fisiere Unix	12.04 - 16.04	16	63	
8	Probleme client-server	19.04 - 23.04	0	0	
9	Procese Procese Unix (I)	26.04 - 30.04	13	63	
10	Comunicaţii între procese Unix: pipe Blocari de fisiere	03.05 - 07.05	21	63	
11	Comunicaţii între procese Unix: FIFO Comunicatii intre procese: pipe, FIFO	10.05 - 14.05	21	63	
12	Blocări de fişiere Unix (facultativ) Procese Unix (II)	17.05 - 21.05	7	63	
13	Incheierea activitatii de laborator	24.05 - 28.05	0	0	
14	Examen practic	31.05 - 04.06	0	0	

FIGURE 6. Add assignments in the admin interface

3.4. **The feedback interface of PULSE.** Another new element in PULSE is its feedback interface which can be accessed before and/or after login. General or specific information about PULSE can be inserted here.

General information refers to the level of satisfaction that PULSE offers to its users, while specific information refers to the notification of a bug, observations regarding the content of the site, suggestions, and so on. Sending the feedback generates an e-mail that is sent to the administrator while also retaining that information in a database for further processing.

4. Conclusions

The current version of PULSE as well as its predecessors are created with the aim to support any professor that uses them to provide high quality education for a large number of students[13] in a field of rapid changes and practical aspects. Therefore these instruments were successfully used but constantly monitored and adapted to the needs of their users.

[13]"Babes-Bolyai" University (BBU) is offering more that hundred majors and has a student population of over 50 000.

The future work on PULSE consists in extending its capabilities by implementing an editor for new concepts and editing and running test-quizzes. Moreover, due to the fact that we used in parallel more such applications (e.g., PULSE and AMS) in our department, a new such instrument is developed. It is built by students and professors and intends to have all advantages of existing tools.

ACKNOWLEDGEMENT

This material is partially supported by the Romanian National University Research Council under award PN-II IDEI 2412/2009.

References

[1] Apex Learning . http://www.apexlearning.com/, 2007.
[2] Blackboard. http://www.blackboard.com/, 2007.
[3] Florian Mircea Boian, Rares Boian, and Alexandru Vancea. AMS: An Assignment Management System for Professors and Students. In *The Symposium "Colocviul Academic Clujean de Informatică"*, pages 137–142, Cluj, Romania, 2006.
[4] Florian Mircea Boian, Rares Boian, Alexandru Vancea, and Horia F. Pop. Distance Learning and Supporting Tools at Babes-Bolyai University. In *International Conference on Informatics Education Europe II (IEEII)*, pages 332–340, Thesaloniki, Greece, November 2007.
[5] Dokeos. Dokeos Open Source e-Learning . http://www.dokeos.com/, May 2007.
[6] Sanda Dragos. PULSE - a PHP Utility used in Laboratories for Student Evaluation. In *International Conference on Informatics Education Europe II (IEEII)*, pages 306–314, Thessaloniki, Greece, November 2007.
[7] Sanda Dragos. PULSE Extended. In *The Fourth International Conference on Internet and Web Applications and Services*, pages 510–515, Venice/Mestre, Italy, May 2009. IEEE Computer Society.
[8] Sanda Dragos and Radu Dragos. SINE - Sistem Informatic de Notare si Evaluare a Activitatii de Laborator. In *Conferinta Nationala Didactica Matematicii*, pages 11–21, Oradea, Romania, May 2006.
[9] Sanda Dragos and Radu Dragos. SINE - a System with Interactive ackNowledgement and Evaluation of students work during laboratory sessions. *Didactica Mathematica*, 25(1):31–39, 2007.
[10] Elearning India. Learning Management Systems - LMS. http://elearning-india.com/content/blogcategory/19/38/, 2006.
[11] IMC AG. eLearning Suite CLIX. http://www.im-c.de/Products/eLearning-Suite/, 2007.
[12] University of Cologne. ILIAS Learning Management. http://www.ilias.de/, August 2007.
[13] University of Zurich in association with the community. OLAT - Open Source LMS. http://www.olat.org/website/en/html/index.html, May 2007.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogălniceanu Street, Cluj-Napoca, Romania
E-mail address: `sanda@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

TOWARDS IMPROVING THE STATIC SEMANTICS OF XCORE

VLADIELA PETRAŞCU AND DAN CHIOREAN

Abstract. Throughout this paper, we analyse the state of facts concerning the static semantics of the XCore meta-metamodel and we propose improvements in formalizing it.

1. Introduction

Nowadays, the model-driven techniques in their various flavors (Model Driven Architecture (MDA) [8], Model Driven Engineering (MDE) [11], Language Driven Development (LDD) [5]) promise to revolutionize software development, by automating a major part of the process. Metamodeling languages stand at the core of this novel paradigm. Therefore, a complete and formal definition of these languages (including their abstract syntax, concrete syntax, and semantics) is vital to the success of the model-driven approach.

However, a study that we have carried out on three of the best known meta-metamodels, namely MOF (Meta Object Facility) [7], Ecore [12], and XCore [5], has revealed that the problem of formalizing the static semantics of these languages is far from being a solved issue. Within this paper, we focus on the state of facts regarding the XCore meta-metamodel and we propose improvements in defining its static semantics.

The rest of the paper is organized as follows. Section 2 provides some background on (meta)modeling and identifies the key requirements in defining the static semantics of a (meta)modeling language. A brief overview of

Received by the editors: August 5, 2010.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.4 [**SOFTWARE ENGINEERING**]: Software/Program Verification – *Programming by contract, Class invariants, Validation*; D.2.11 [**SOFTWARE ENGINEERING**]: Software Architectures – *Languages (e.g., description, interconnection, definition)* .

Key words and phrases. metamodeling, XCore, static semantics, Well Formedness Rules (WFRs).

XCore, the current approach in describing its static semantics, as well as details regarding our contribution are given in Section 3. The paper ends with conclusions in Section 4.

2. Backgound on (meta)modeling

Similar to any other language, a modeling language can be defined as a 3-tuple of the form (abstract syntax, concrete syntax, semantics) [5]. The abstract syntax defines the vocabulary of concepts employed by the language, together with their structural inter-relationships. It has been traditionally given in terms of a class model - called the metamodel[1], which can be visualized by means of class diagrams. Defining a metamodel is an analogous process to defining the BNF (Backus-Naur Form) grammar of a programming language. Moreover, similar to an ordinary context free grammar which is unable to capture static semantics' rules concerning typing or scoping, the graphical class diagram notation lacks the expressive power needed to lay down complex constraints that rule out illegal combinations of concepts in the language. Such constraints, known as Well Formedness Rules (WFRs) define the static semantics of a modeling language. They are usually formalized as invariants on the metamodel, using OCL [9] or an OCL dialect.

As they complement the class diagram descriptions, the major value of WFRs resides in the fact that they ensure a better understanding of the modeling concepts' semantics. The existence of an informal (natural language) equivalent of each WFR is mandatory in this respect. Regarding the formal specification, its role is twofold. On the one side, it increases rigor and helps preventing potential ambiguities. On the other side, it lays the basis of automatic model validation. Assuming that the language is tool-supported, the formalized WFRs allow checking whether models are correct/compilable or not with respect to their representation language. No model transformation task, such as code generation or XMI serialization, should be allowed on non-compilable models.

The arguments above are even stronger when the language is a metamodeling language. Metamodeling languages are the "languages used to define other languages". They stand at the top of the metamodeling architectures proposed by all model-driven approaches. In case of a metamodeling language, its abstract syntax is represented by means of a meta-metamodel (a model which is its own metamodel). Having its static semantics appropriately specified (by means of explicit and formal WFRs associated to the meta-metamodel) is thus

[1]Here, we use the term *metamodel* in its general acceptance, as denoting the abstract syntax model. According to the LDD vision however, a metamodel should capture the entire model of a language, covering also its concrete syntax and semantics.

highly important, since it enables checking the compilability of all metamodels instantiating it.

The research carried on the three previously metioned meta-metamodels has allowed us to identify the following general requirements in defining the static semantics of a metamodeling language:

- All WFRs should be stated in both an informal (human-understandable) and formal (machine-readable) style. The informal specification should come prior to the formal one and be as detailed and precise as possible.
- Each informal definition of a WFR should be accompanied by meaningful model test cases, promoting a test-driven WFR specification style.
- The formal specifications should be stated in a manner that allows getting the maximum amount of useful debugging hints in case of assertion failures (specifications should be testing-oriented). The use of OCL specification patterns, as proposed in [4], is highly recommended.
- The OCL specification style should ensure identical WFRs' evaluation results after translation in a programming language.

3. XCore static semantics

XCore is the bootstraping kernel of XMF (eXecutable Metamodeling Facility) [1, 5], a MOF-like metamodeling facility focused on capturing all aspects of a language definition - abstract syntax, concrete syntax and semantics. Unlike MOF though, XMF is completely self-defined and provides platform-independent executability support by means of an executable OCL dialect named XOCL.

3.1. **State of facts.** The official XMF reference [5] acknowledges the value of WFRs and promotes their use in defining the abstract syntax of modeling languages. Still, the document does not describe (neither informally, nor formally) any WFR for the XCore meta-metamodel. As regarding the XMF implementation, this does only include two explicit XOCL constraints, specified in the context of the `Element` and `Object` classes, respectively. Apart from these, there seems to be also a number of other constraints which are only inferable from the XOCL code corresponding to the XCore modifiers.

The XMF approach, that omits the explicit definition of WFRs, trying to preserve model consistency only by means of a suitable implementation of modifiers, has a number of drawbacks.

- In case of an executable language such as XMF, which also provides an interpreter console, one can never assure that model changes will be performed exclusively by calling the corresponding modifiers in the prescribed order. Direct assignments or different call sequences are also possible, leading to potentially invalid models.

- As emphasized by [6], this approach may be seen as an alternative to the use of preconditions. As opposed to preconditions however, it induces an increased code complexity, with a negative effect on reliability.
- Complex constraints generally involve multiple classes and the necessity of "adjusting" the code of several modifiers. Overlooking to check for the rule in any of these modifiers may lead to incorrect models. Instead, writing explicit WFRs is simpler, more clear, and less error-prone.
- Trying to preserve model consistency at all stable times may not be the best solution always. Underspecification, for instance, may be desirable in particular circumstances.

Writing explicit WFRs is a prerequisite in enforcing them. Even with the approach taken, the XMF implementation does not cover some of the elementary WFRs that are compulsory for object-oriented concepts, such as avoiding name conflincts among features of the same class/classifier or the proper management of contained-container dependencies.

3.2. **Proposed improvements.** As a solution to the above mentioned problems, we have proposed a set of XOCL WFRs for the XCore meta-metamodel, which we have tested on relevant model examples. The entire set of rules, together with the corresponding tests, can be consulted at [2]. Below, we only discuss three relevant examples.

3.2.1. *Name conflicts among owned and inherited members.* As previously stated, one of the WFRs not covered by the XMF implementation concerns the name conflict among an attribute owned by the current class and attributes inherited from its ancestors. This is a fundamental object oriented modeling constraint, being enforced by object oriented programming languages as well. Such a conflict should arise in case of class `D` from Figure 2, which defines the attributes `b` and `r`, having identical names with an inherited attribute and reference, respectively.

The XOCL constraint that we propose for the above mentioned WFR is given below. Figure 1 illustrates the corresponding part of the XCore meametamodel.

[WFR1] There should not be any name conflicts among the attributes owned and inherited by a class.

```
context Attribute @Constraint uniqueName
 let allAtts = self.owner.allAttributes() then
```

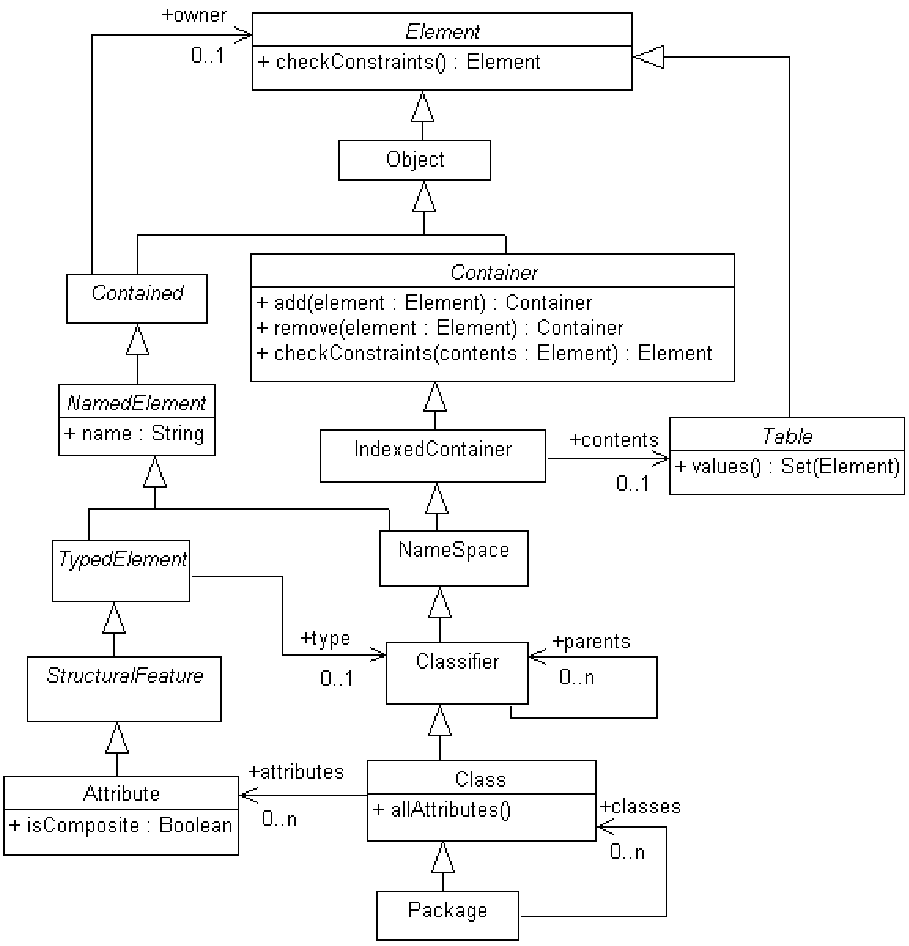


FIGURE 1. An excerpt of the XCore meta-metamodel

```
     sameNameAtts = allAtts->excluding(self)->select(att |
                       att.name.asSymbol() = self.name.asSymbol())
 in sameNameAtts->isEmpty()
 end

fail
  let sameNameAtts = self.owner.allAttributes()->excluding(self)->
        select(att | att.name.asSymbol() = self.name.asSymbol()) then
      msg = "Attribute name duplication! " +
            "Inherited/owned attributes of " + self.owner.toString() +
            " with the same name: "
  in @While not sameNameAtts->isEmpty() do
       let att = sameNameAtts->sel
```

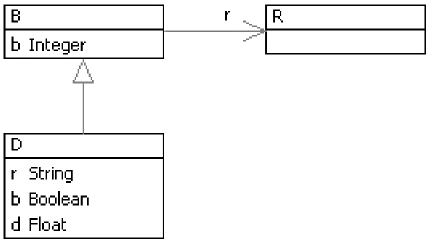


FIGURE 2. A non-valid model example

```
        in msg := msg + att.owner.toString() + "::" + att.toString() + "; ";
           sameNameAtts := sameNameAtts->excluding(att)
        end
      end;
      msg
   end
end
```

Apart from the constraint itself, XMF allows the specification of a fail clause, whose body is intended to provide valuable model debugging information in case of assertion failure. This facility is in accordance to the testing-oriented specification style that we promote in [4].

3.2.2. *Containment relationships.* The proper management of containment (composition) relationships is a fundamental issue in metamodeling. This subject has been also approached by us in [3], in the context of UML 2.3 [10]. As shown by the metamodel excerpt in Figure 1, XCore represents containments explicitely, by providing the `Contained` and `Container` abstract metaclasses in this purpose. Below, we give their description, as taken from the XMF documentation.

> "A *container* has a slot contents that is a table[2]. The table maintains the contained elements indexed by keys. By default the keys for the elements in the table are the elements themselves, but sub-classes of container will modify this feature accordingly. Container provides operations for accessing and managing its contents."
>
> "A *contained* element has an owner. The owner is set when the contained element is added to a container. Removing an owned

[2]According to the metamodel, this rather seems to be the description for `IndexedContainer`, due probably to a lack of synchronization between metamodel and documentation.

element from a container and adding it to another container will change the value of owner in the contained element."

According to the commonly-agreed semantics of containments, we claim that there are two fundamental rules that any model should fulfil in this respect.

(1) A part should belong to a single container at a given time.
(2) A container cannot be itself contained by one of its parts.

As in case of other constraints, the enforcement of the ones above was meant to be covered in XMF by an appropriate implementation of operations in the descendants of `Container` and `Contained`. Moreover, in order to preserve models' validity, these operations are expected to be called in a particular sequence during model editing tasks. As a consequence, the models created using the model/diagram editors of the XMF tool (XMF-Mosaic) are correct with respect to these rules. However, the models edited using the interpreter console (where there is freedom with respect to the type and sequencing of the editing operations) may reach invalid states, which are impossible to detect in the absence of explicitly stated WFRs.

In order to exemplify this for the rule (1), let us start from a sample XCore model containing an empty package named `Test1` (which has been assigned to a global variable `t1`), and the following sequence of XOCL commands executed within the XMF interpreter console.

```
p1 := Package("P1");
t1.add(p1);
p2 := Package("P2");
t1.add(p2);
c := Class("C");
p1.add(c);
```

The lines above modify our initial model by creating two new packages, `P1` and `P2`, which are added as subpackages of `Test1`, and a class, `C`, which is added to package `P1`. As a consequence, class `C` will have `P1` as its `owner`, while `P1` will have `C` as the only element witin its `contents` table.

Suppose `C` has been mistakely added to `P1`, when it should have been, in fact, added to `P2`. Issuing the following command in the console

```
p2.add(c);
```

apparently solves the problem, since the owner of `C` is changed to `P2`, and `C` is added to the contents table of `P2`. However, `C` still belongs to the contents table of `P1`, from which it should have been removed prior to its addition to `P2`. Therefore, in the current state, the model is invalid with respect to rule (1), as `C` simultaneously belongs to two different containers (`P1` and `P2`). A visual proof of this is given by the model browser on the left of the XMF-Mosaic screenshot from Figure 3, illustrating the state of the model as reached after the execution of the above commands.

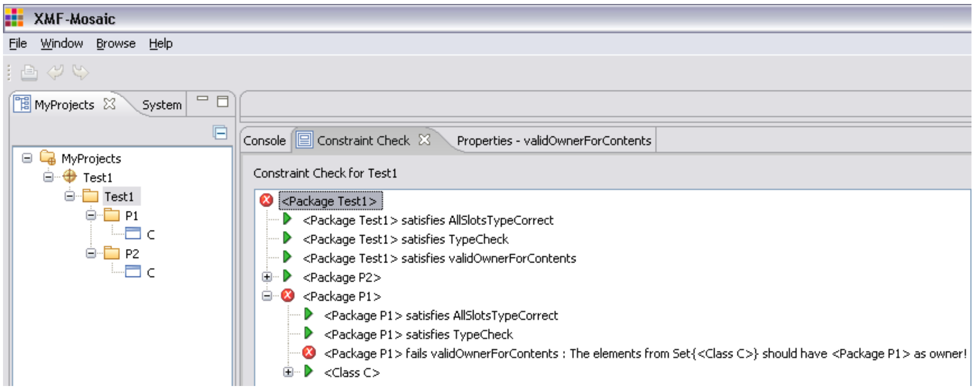
Still, even if the model is obviously wrong, the lack of an appropriate WFR makes the call to `checkConstraints()` on `Test1` report this package and its entire contents as valid. The XCore WFR that we propose below offers a solution to this problem.

[WFR2] All Contained instances that belong to the contents table of an IndexedContainer should have that container as owner.

```
context IndexedContainer
 @Constraint validOwnerForContents
  self.contents.values()->select(v | v.oclIsKindOf(Contained) and
               v <> null)->select(v | v.owner <> self)->isEmpty()

  fail "The elements from " +
       self.contents.values()->select(v | v.oclIsKindOf(Contained) and
                      v <> null)->select(v | v.owner <> self).toString() +
       " should have " + self.toString() + " as the owner!"
 end
```

As shown by the right-hand side of the screenshot in Figure 3, the model checking performed after the addition of the above constraint to `IndexedContainer` reports the `P1` package as invalid with respect to this particular constraint. In fact, the proposed constraint captures anomalies of a more general nature than just parts simultaneously belonging to at least two different containers (e.g. parts belonging to the `contents` table of a container and having no `owner` set at all).



Constraint Check for Test1

<Package Test1>
<Package Test1> satisfies AllSlotsTypeCorrect
<Package Test1> satisfies TypeCheck
<Package Test1> satisfies validOwnerForContents
<Package P2>
<Package P1>
<Package P1> satisfies AllSlotsTypeCorrect
<Package P1> satisfies TypeCheck
<Package P1> fails validOwnerForContents : The elements from Set{<Class C>} should have <Package P1> as owner!
<Class C>

FIGURE 3. XMF-Mosaic screenshot

Regarding the rule (2) above, the neccessity of introducing a corresponding explicit WFR can be argued by means of the following example. Let us assume the existence of an XCore model consisting of a single empty package named `Test2` (that has been assigned to a global variable `t2`). Furthermore, assume that there is the neccessity of creating under `Test2` a hierarchy of three subpackages, say `P1`, `P2`, and `P3`, each included in the previous one. This basic model editing task can be accomplished in XMF by means of the following sequence of commands.

```
p1 := Package("P1");
t2.add(p1);
p2 := Package("P2");
p1.add(p2);
p3 := Package("P3");
p2.add(p3);
```

However, the misuse of `p1` instead of `p3` as the argument of the latter call above has the effect of creating a circular containment between packages `P1` and `P2`, each of them becoming the `owner` of the other. Yet, in the absence of an explicit WFR prohibiting this, a call to `Element::checkConstraints()` on any of them reports no problem at all.

As a solution to this, we propose the XOCL WFR below, which applies to all indexed containers, except for the `Root` namespace (in XMF, `Root` is the global namespace in which everything is contained, itself included).

[WFR3] No IndexedContainer different from the Root namespace can be owned by one of its parts.

```
context IndexedContainer
 @Constraint notOwnedByPart
  (self <> Root and self.oclIsKindOf(Contained)) implies
  self.contents.values()->select(v | self.owner = v)->isEmpty()

  fail "This container is owned by each of its parts from " +
        self.contents.values()->select(v | self.owner = v).toString()
 end
```

4. Conclusions

In view of the goals pursued by the model-driven approaches, formalizing the static semantics of any metamodeling language is a must. Within this paper, we have analyzed the approach taken in case of the XCore metametamodel and we have identified its shortcomings. The proposed solution consists in the definition of a set of XOCL constraints, that have been validated on relevant model examples. Each WFR is stated both informally and formally and is accompanied by meaningful test cases.

Acknowledgements

This work was supported by CNCSIS-UEFISCSU, project number PNII-IDEI 2049/2008.

References

[1] eXecutable Metamodeling Facility (XMF) homepage. `http://itcentre.tvu.ac.uk/~clark/xmf.html`.

[2] Frame Based on the Extensive Use of Metamodeling for the Specification, Implementation and Validation of Languages and Applications (EMF_SIVLA) homepage. `http://www.cs.ubbcluj.ro/~chiorean/CUEM_SIVLA`.

[3] Dan Chiorean and Vladiela Petraşcu. Specification and Evaluation of Constraints in MOF-based Metamodels. In *ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS'10), Workshop on OCL and Textual Modeling*, October 2010. (accepted).

[4] Dan Chiorean, Vladiela Petraşcu, and Ileana Ober. Testing-Oriented Improvements of OCL Specification Patterns. In *Proceedings of 2010 IEEE International Conference on Automation, Quality and Testing, Robotics AQTR 2010, Tome II*, pages 143–148. IEEE Computer Society, 2010.

[5] Tony Clark, Paul Sammut, and James Willans. *Applied Metamodeling. A Foundation for Language Driven Development (second edition)*. Ceteva, 2008.

[6] Bertrand Meyer. *Object-Oriented Software Construction (second edition)*. Prentice Hall, 1997.

[7] Object Management Group (OMG). Meta Object Facility (MOF) Core Specification, Version 2.0. `http://www.omg.org/spec/MOF/2.0/PDF`.

[8] Object Management Group (OMG). Model Driven Architecture (MDA) Guide, Version 1.0.1. `http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf`.

[9] Object Management Group (OMG). Object Constraint Language (OCL), Version 2.2. `http://www.omg.org/spec/OCL/2.2/PDF`.

[10] Object Management Group (OMG). Unified Modeling Language (UML) Infrastructure, Version 2.3. `http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/`.

[11] Douglas C. Schmidt. Model-Driven Engineering. *Computer*, 39(2):25–31, 2006.

[12] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework (second edition)*. Addison-Wesley Professional, December 2008.

Babeş-Bolyai University of Cluj-Napoca, Mihail Kogălniceanu nr. 1, Cluj-Napoca, Romania

E-mail address: `{vladi,chiorean}@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

SOCIAL NETWORKS FOR EPIDEMIC SPREADING: A CASE STUDY

SILVIA RAUSANU AND CRINA GROSAN

Abstract. The study of social networks and the spread of disease on networks has recently attracted considerable attention and there is a huge amount of work dealing with this matter. The current work presents and develops a social network model based on the multigraph idea in order to simulate the spread of an epidemic. The paper deals with a real example which refers to the spread and evolution of A/H1N1 virus in Romania. The simulations obtained by applying the proposed model approximate very well the real evolution of the virus studied over a period of 10 months.

1. Introduction

The epidemiologists have developed mathematical models to estimate the size and other key features of an epidemic and social networks seem to be a very good candidate in simulating the spread of such epidemics [4, 9]. There are several approaches which simulate how a disease is transmitted among the nodes of a network which has the proprieties of a social network [3, 11, 10].

In our paper we develop a very general model which uses more than a simple social network but a set of connected social networks in order to simulate the spread of a well know epidemic of the past year and the begging of the current one: the A/H1N1 virus.

The appearance of the virus A/ H1N1 was dated at the end of April 2009 in Mexico, ever since this virus has spread around the globe with an amazing speed. Romania was reached by the virus only on the 23^{rd} of May 2009. As this virus can be contacted by air, the spread among the countries was facilitated by the multiple means of transportation, but not only among the countries, but also inside them.

The case of Romania did not make any exception: the virus was contacted from abroad, firstly in the main cities of the country, then, following the same

Received by the editors: August 13, 2010.

2010 *Mathematics Subject Classification.* 91D30.

1998 *CR Categories and Descriptors.* code [**I6.3**]: Computing Methodologies – *Simulation and Modeling*; code [**G2.2**]: Mathematics of Computing – *Graph Theory*.

Key words and phrases. social networks, epidemics spread modeling.

rule of moving masses of population, in almost each county. In addition to these factors, the time in which the spread got a higher speed was in the months when the likelihood of flu infection was greater and the crowding in different areas was much more common.

These factors can form a pattern in the spread of the virus, at any level of community: inside a county or inside a country. However, in Romania the spread did not respect entirely the patterns, nor the expectations of the Health Minister of Romania, as some counties , during the entire epidemic development, did not even had a case of infection with the virus A/H1N1.

The paper is organized as follows: Section 2 presents the basic social networks notions used in our model, in Section 3 some of the existing work in epidemics spreading is analyzed, in Section 4 the proposed model is presented in detail. Section 5 presents details about the simulations followed by experiments. Section 6 contains conclusions and future work ideas.

2. Social Networks

The notion of social network and the methods of social network analysis have attracted considerable interest and curiosity both from the social-behavioral community and from the computer science and mathematical community in the recent decades. Research in the social studies is mostly interested in the reason behind the network connection rather than the properties of the network structure itself. On the other side, the research in graph-theory has provided a wealth of quantitative tools and mechanism for describing networks mostly by analyzing the type of relations and the general structure.

Along with the growing interest and amount of research – in distinct domains – over the use and properties of social networks, it has been established a general set of principles for the social network analysis suitable for any domain [9].

During the studies of real-world social networks, some research was oriented towards the statistical properties of these networks. One important and fundamental result that has emerged from these studies concerns the numbers of ties that actors have to other actors, their so-called “degrees.” It has been found that in many networks, the distribution of actors’ degrees is highly skewed, with a small number of actors having an unusually large number of ties. Simulations and analytic work have suggested that this skewness could have an impact on the way in which communities operate, including the way information travels through the network and the robustness of networks to removal of actors. Among the developed models for networks, three of them will be recalled in what follows.

We first give some basic definitions which will be further used.

Consider a *graph* = G (V, E) is a plotting structure for the set of vertices V (nodes) and the set of edges E for the connection of pairs of vertices. The cardinality of the set V is denoted by n and is known to be the *order* of the graph, while the cardinality of E, m¸ is called the *size* of the graph. Two joined vertices by an edge are called *end-vertices.* Any two vertices connected by an edge are *adjacent* or *neighbors.*

Definition 1 (multigraph)

A *Multigraph* is a directed or undirected graph in which is allowed to have in the set of edges some edges contained more than once. If the edge e occurs several times in E, then its copies are called *parallel edges.*

Definition 2 (degree of a vertex)

The number of edges having as end-vertex the node $v1$ in the graph is called the *degree of vertex v1.*

Remark 1:

If the graph is a multigraph, then the parallel edges are counted according to their multiplicity.

Centrality indices exist for the quantification of an intuitive feeling that in most networks some vertices or edges are more central, important, than others. The computations of such indices provide new methods for the analysis of the social network.

Definition 3 (degree centrality)

Degree centrality is a characteristic of the node with the highest degree – the connector/hub of the network.

Definition 4 (Betweeness centrality)

Betweeness centrality characterizes a node that influences the flow of the network but not the network itself, in graph theory being a critical point for connectivity of the graph.

Definition 5 (closeness centrality)

Closeness centrality is the quality of a node that has the shortest path to every other node.

Network centralization is the property of keeping the connectivity of the graph around one or few vertices. High centralization on a node may lead to fragmentation of the network, as there is one critical node, while low centralization guarantees smaller changes of such thing to happen.

Definition 6 (clustering coefficient)

The *clustering coefficient* [3] refers to the tendency (observed in many natural networks) of forming *cliques* in the neighborhood of one given vertex. It measures the average probability that two neighbors of vertex i are also connected.

The clustering coefficient is computed as follows: let i be a vertex, with degree k and e the number of edges formed among its neighbors, then the clustering coefficient is the ratio between the actual number of edges among neighbors and the maximum number of possible edges between neighbors:

$$c = \frac{e}{\frac{k \cdot (k+1)}{2}}$$

Definition 7 (degree distribution)

Degree distribution is a statistical characterization of the graph given by the sequence of degrees or by the relative probability distribution of degrees.

2.1. **Random networks – Erdos-Renyi model.** The random network model of *Erdos* and *Renyi* (ER) is considered as the first model able to coherently describe networks of arbitrary size [3]. The rules to create a random graph according to *Erdos* and *Renyi* prescriptions are simple. The network is a set N of n different vertices, and m edges (E) joining the nodes, such that between any two vertices i, j exists a connecting edge with the independent probability p. In these conditions, the maximum number of edges in the network is:

$$\max(m) = \frac{n \cdot (n-1)}{2} \cdot p$$

and the maximum vertex degree is:
$\max(d(i)) = n$-1, $\forall \quad i \in N$.

The average degree is easily computed:
$k_{avarage} = (n\text{-}1) \cdot p$

The clustering coefficient is $c = p$, since it is simply the probability that two nearest neighbors of a vertex of degree k have an edge between them. The probability that a randomly chosen node has exactly the degree k (that is that only k of its possible edges are present and $n - 1 - k$ are absent) is given by the *binomial distribution*:

$$P(k) = \left(\begin{array}{c} n-1 \\ k \end{array} \right) \cdot p^k (1-p)^{n-1-k}$$

However, as a model of real-world social network, it has some serious shortcomings. Perhaps the most serious is its degree distribution, which is quite unlike for those seen in most real-world networks. The approximation to the real-world model is poor as the degree distribution is highly skewed. On the other hand, the random graph has desirable properties, particularly the fact that many features of its behavior can be calculated exactly.

Some algorithms might work in creating a more appropriate approximation to real-world networks. Starting from knowing only the degree distribution of the desired network, the probabilities for a random vector to have degree k, must be normalized. Take a number of N vertices and assign to each a number of k ends of edges, where k is a random number drawn independently from the distribution p_k for each vertex. Then we take the ends of edges randomly in pairs and join them up to form edges between vertices. This procedure will produce a graph with exactly the desired degree distribution, but which is in all other respects random. To put it another way, we have generated a graph that is drawn uniformly at random from the set of graphs with the given degree distribution.

2.2. **Scale-free model.** Scale-free networks are characterized by the power law distribution in degree distribution, which can be expressed mathematically as $P(k) \approx k^{-\gamma}$. From the form of the distribution it is clearly that when γ < 2, the average degree diverges and when $\gamma < 3$, the standard deviation of the degree diverges [8]. It has been found that most scale-free networks have exponents between 2 and 3, thus, they lack a characteristic degree or scale, and therefore their name.

The clustering coefficient distribution is in a strong reverse connection with the degree distribution, as it decreases with the increase of the degree of the node.

The *Erdos-Renyi* model is less likely to produce a scale-free network as their properties are not consistent with those of the ER model; consequently a model for growth process is needed. The most widely generative model is the one of *Barabási* and *Albert's ('rich and get richer')* in which each new vertex connects to another one with a probability that is not uniform, but proportional with the in-degree of the old-vertex. Later development led to respecting the power-law distribution by making a supposition according to which a vertex with high in-degree will tend to attract more vertices than the other ones.

2.3. **Small world model.** The small-world property refers to the fact that in many large scale networks the average distance between vertices is very small compared to the size of the graph. The distance between two vertices in a graph is measured as the shortest path length l among them. The small-world property is present when l scales logarithmically (or even slower) with the number of vertices.

Inheriting from sociology, this model is known as well for *"six degree separation",* stating that a short number of acquaintances (on average six) is enough to create a connection between any two 'actors' chosen randomly.

From the mathematical point of view, the small-world effect describes those graphs whose diameter and average path-length grow much more slowly than the number of nodes n,typically, *ln n,* just as in random ER model. Yet, a random graph has a very small local interconnectedness, captured by the clustering index.

Many scale-free networks are embracing the small-world model [3]: the average path length and the clustering coefficient of the network are compared with the same quantities calculated over random graphs with the same number of nodes and vertices. However, the reverse is not true: small-world networks are not scale-free networks.

3. Related work on epidemic spreading

In [1], the impact of the clustering coefficient and of the assortativity coefficient in a randomly generated network over epidemic behavior is analyzed. The authors start with a generation algorithm for the social network, emphasizing the positions of the already infected nodes, by placing them on an imaginary ring. For the simulation, it is used the SIR epidemic model which is run for distinct configuration of the network: different indices or different infected population size. The results of the experiment prove that the infection size, after a number of simulations, is negatively proportional with the values of the clustering and assortativity indices. The greater is the clustering index of the network, the smaller is the size of the newly infected population, this statement being true for the assortativity index, as well.

A project closer to the theme discussed in this paper is [2] which propose a model for the progression of pandemic influenza A (H1N1) in Vietnam. The main difference which arises from this paper is the used structure, distinct from social networks. It was developed an age- and spatially-structured mathematical model in order to estimate the potential impact of pandemic H1N1 in Vietnam and the opportunities for reassortment with animal influenza viruses. The model tracks human infection among domestic animal owners and non-owners and also estimates the numbers of animals may be exposed to infected humans.

In [6], the simulation of the pandemic influenza is performed using graphical representation on the maps. Maps are built by stitching the counties that contain cities and localities, the encoding of these elements using different colors on the map and the generation of the neighbor relationship.

4. Model description

The case of spreading the virus A/H1N1 in Romania might be modeled at the level of counties taking into account their connections with the other counties in Romania, but also the connections with other countries affected by the this epidemic. However, this approach would not have been enough as in the high peek of spreading, most of the infection were occurring only inside a county due to some extra crowding coefficients.

This is why the epidemics must be modeled as well at the level of a county by tracking (or assuming) the connections between the locators of the county.

The system will be represented as a network of networks.

The main network has as nodes the counties of Romania and some other important countries of the world with which Romania has a stronger connection. The edges are the connections between the counties created on the basis of neighborhoodness, collegial nodes, railroad nodes, important airports, tourism attractions. The influence of these connections is underlined by some probability of infection in one current node with the help of the number of infection cases in the counties at the other end of the edge. The networks inside the counties are advisable in simulating the epidemic because are nearer to the real situation and, furthermore, are a must in obtaining some results closer to reality. These inner-networks are impossible to be created based to reality, as tracking millions of connections is way out of any league, so they will be randomly generated based on a known model of networks: *Erdos-Renyi.* According to this model, in a network of n nodes, any two vertices are connected with an independent probability p which means that the network has as clustering index that initial probability p since it is simply the probability that two nearest neighbors of a vertex of degree k have an edge between them.

In the real situation, a county of m individuals with a population density d, is manipulated according to the *Erdos-Renyi* model [5] by generating a random network with m' nodes and clustering index d'. m' is a scale of the size of population m according to the maximum size of the social network permitted by the implementation. d' is also a scale of the density but according to the domain values of the clustering index : $[0, 1]$, which means that the county with the highest population density value will have the highest clustering index; although this would imply that the maximum clustering index between all the counties is 1, the maximum clustering index will actually be around 0.6 as any higher value will transform the network into an almost complete graph, situation which is mostly uncommon for large real-world social network (like a county, in this situation). In this way some connection between individuals will be assumed in the scope of gaining the clustering index. No further properties are attached to the network during its construction as the purpose of the network is oriented towards the direct connections between nodes (not by paths, centrality properties, etc).

When a county has some cases of infection, it appears the issues of positioning these infected nodes in the network. In order to gain some local results, the infected node must have an important position in the network, one idea could be selecting the first i nodes with the highest degree. This decision is taken on the basis that in reality the spread of the virus is done by air, usually in crowded places, which in network-terms is translated as clusters.

Usually in clusters there are some nodes with high degree which are important to the network, consequently, for the current case, there should be placed the already infected individuals. After setting the infected nodes, their position is no longer of interest, but only their 'identifier' in the entire network.

The spreading simulation is performed at two levels, one at a level of counties and one at the level of individuals. The individuals are tightly connected to the county they belong to, their individual evolution being influenced by the same characteristics the county is, but in different proportions. One individual can be connected to one county only, in this way a much complex network is created, although the homogeneity of the nodes' types and relevance lacks completely.

The factors of influence in the spreading are transmitted from the county in general, to each individual located in it, in particular. This transmission order classifies the network in two hierarchies, corresponding to the levels of spreading simulations, in this way the network for spreading fits the model of the hierarchical social networks. However, the flow of transmission is bidirectional, not only a county distributes its characteristics, but also the mass of individuals contribute to the final statistics computing for a county. Having the interdependence described above, the two hierarchies can be isolated and separated physically, but sharing the context of spreading, keeping the exchange of information during the simulation.

The spreading simulation is performed at two levels, one at a level of counties - the outer network - and one at the level of individuals - the inner network.

4.1. **The outer network.** The *outer network* is the network composed of the counties of Romania, some type of map of the country. Although a social network has usually as nodes sole individuals, a group or an organized formation of individuals can be at the basis of a network, in the current case, the mass of inhabitants of a certain country. The connections between these nodes are created on many criteria fact that change the structure of the network from a simple graph - directed or undirected - to a directed multigraph as between two nodes exists more than one type of links.

The outer network suits the scale-free model as it gathers its most important features. Firstly, the existence of hubs is underlined by some counties that tend to have lots of connections with the other nodes; these nodes have actually a high importance not only in the network but also in the country (collegial centers, main city, tourism nodes, etc). Secondly, the connectivity of the network is easily assured only by one type of connection: the neighborhood between counties, which makes the multigraph underlying it strongly connected. The multigraph structure is defending the network from fragmentation as the important counties do not keep in their links towards others the key of connectivity.

The size of of the outer network is 41 (the number of counties in Romania).

The links between the vertices are formed on some pre-established conditions which imply only characteristics taken from reality. The most important reasons for putting an edge between two nodes are: geographical closeness; collegial surroundings; nodes of means of transportation - railway, airport; tourism attractions; poverty level.

4.2. **The inner network.** The "inner network" is developed from one internal node of the outer network, inheriting some computed or native attributes from it.

The nodes of the inner network (corresponding to each county) will represent the individuals of a county without containing any extra information. The edges will be simple connections between individuals, generated according to the chosen model.

The two parameters required for the construction of the network are received from the corresponding node in the outer network and they are:

- the number of nodes in the network and
- the probability that any two nodes are connected.

Seeing clusters as crowds of individuals makes a logical connection with the population density which is known from the very beginning.

the number of nodes in the network, in an ideal programming environment, could have been taken raw as the size of the population in the current county. Unfortunately, the designed implementation does not permit such a vast memory usage; thus this parameter should be scaled as well according to the maximum size allowed and the maximum size of population among all counties. Applying this theory, an individual will actually represent n individuals which could cause problems during simulations when scaling the number of already infected individuals, for example: there is one infected individual, but following the scaling rule, one real individual is 0.2 of one individual in the used network, therefore there will be 0.2 individuals infected.

Due to the memory limitation, each inner network will be split in m smaller inner network which can be considered to be independent communities or clusters of individuals inside a county. The number of communities in a county will be equivalent with the population density in that county.

5. Simulations

An epidemic is characterized firstly by the way the virus can be contacted; the easier the virus is contacted, the more factors encourage the epidemics.

In the case of the virus A/ H1N1, the spreading is done by air, a common and successful medium for an epidemic to pass to a pandemic spreading. The influence of the continuous movement of masses of population is sustained by the multiple means of transportation between communities of individuals and, moreover, by the increase of the crowding coefficient that traveling with most of those means presumes. The considered situations were explained in the previous chapter by describing the way they were modeled and selected from the multitude existing.

The types of edges are the form chosen for codifying the way individuals change their current node location to another, carrying along the virus from one infected community to another one still healthy. Among the percentage of individuals that move from one node to another there exists the possibility of existing individuals that are infected, and as the virus is transmitted by air, any short or long contact of that individual can add a new victim to the general statistics. However, this theory is not totally real, depending actually on the particular characteristics of the individual, e.g. the power of its immunity system.

The state of the weather is another positive factor of influence for the spreading. In the cold months when it is often raining, snowing of wind blowing, the human immunity system fails to keep the same properties as it used to in the warmer months, so the probability of viral infection is increased for the majority of the population.

One factor that independently rises from the context is the apparition of a vaccine against this virus. The factors of influence remain valid for all the cases, but the number of susceptible individuals to the disease decreases drastically as an enormous part of the population of Romania has taken this vaccine. Consequently, in the final months of the epidemic, the spreading has reached the lowers level of activity and finally became inactive.

5.1. **Numerical experiments.** Numerical experiments are performed in Romania, over a hierarchical network with two layers corresponding to the country level (this network has 41 nodes corresponding to the 41 counties) and the county level (for each of the 41 counties, the network has a variable number of nodes according to the population size of each of them).

The results are simulated over 9 months, between May 2009 (the starting months which is not taken into consideration for simulations) and February 2010.

From the input set it is generated, for each test of the application (each input month), another set of data, used in the following computations. Besides the data gathered from different sources, there are other parameters which influence the spreading simulations, parameters which are established from the very start as constants for the entire application.

5.2. **Data.** The collected data includes the monthly situation of newly infected population size of each county from Romania and of the countries included in the network with which our country has stronger connections.

Tables 1 and 2 present the data concerning each node of the outer network during the studied period of time: May 2009 – February 2010.

The data organized in these tables was connected from the Romanian Health Minister official site, the section of press communicates [7].

Another set of data which remains constant during all the executions are the characteristics of an internal node in the outer network. These data consist of the population size and population density of each county.

Table 2 does not contain population information about the countries with which Romania has connections as the size and density of population corresponding to their country does not arise any interest as the simulation is not performed on those nodes, it are only used for helping the simulation inside Romania.

5.3. **Results of the simulations.** We present the results of the simulations for two of the nine months: August and October 2009. Simulations are performed in an identical manner for the other remaining 7 months.

Experiment 1 – Month August

The first experiment has as purpose to simulate the evolution of the epidemic over the counties of Romania during the month August 2009. The month which will be given as input will be *July.* The data for input, besides the one given by the user, is read from the local database and will be used for the construction of the network.

The construction of the network will proceed as follows: the nodes – counties of Romania and some countries – will be taken with the attached information: the name, population size and density (the data from the Table 2 will be loaded). To the loaded nodes there will be added some extra information: the situation for the month given as input: the infected population size of each node, the corresponding column for the month July 2009 from the Tables 1 and 2; after the nodes are set in the network, the connections are loaded.

At this moment, the network is loaded into memory and it is ready for starting the simulations over it. However, the simulation influence factors must be taken into account: the "temporal constants". The month August is known to be in Romania as the month when most of the people go in a holiday, consequently there will be slight modifications on the computations.

Table 1. Infection size on nodes - counties

County	**Infection population size**									
	May 2009	June 2009	July 2009	Aug. 2009	Sept. 2009	Oct. 2009	Nov. 2009	Dec. 2009	Jan. 2010	Feb. 2010
Alba	0	0	0	0	0	0	22	44	13	0
Arad	0	0	0	0	0	0	80	50	10	0
Arges	0	0	0	1	1	0	29	54	23	1
Bacau	0	0	0	2	0	0	130	111	29	1
Bihor	0	0	0	0	0	1	10	4	2	0
Bistrita-Nasaud	0	0	0	0	0	0	0	1	3	3
Botosani	0	0	0	0	0	0	269	133	111	0
Brasov	0	0	27	9	3	1	29	43	14	4
Braila	0	0	0	6	0	0	2	11	14	1
Buzau	0	0	0	0	0	0	25	121	5	0
Caras-Severin	0	0	0	0	0	0	19	63	14	1
Calarasi	0	0	1	2	0	0	3	0	5	0
Cluj	0	0	0	6	1	3	58	84	45	1
Constanta	0	0	5	2	1	0	21	55	48	4
Covasna	0	0	0	0	0	0	5	22	11	4
Dambovita	0	0	0	2	0	0	94	107	52	1
Dolj	0	0	8	0	0	1	148	98	8	0
Galati	0	0	1	4	0	0	85	33	22	1
Giurgiu	0	0	0	0	0	0	20	16	8	0
Gorj	0	0	0	0	0	0	0	0	0	0
Harghita	0	0	0	0	0	0	52	39	23	0
Hunedoara	0	0	0	8	0	0	94	40	27	0
Ialomita	0	0	0	0	4	0	16	22	5	0
Iasi	0	7	8	6	8	40	171	58	39	3
Ilfov	5	15	62	63	9	12	725	612	260	15
Maramures	0	0	0	0	0	0	18	19	3	0
Mehedinti	0	0	2	1	1	0	11	12	1	0
Mures	0	0	7	7	0	1	27	66	56	3
Neamt	0	0	0	0	0	0	56	33	24	1
Olt	0	0	0	0	0	0	10	24	40	0
Prahova	0	0	1	4	0	41	51	83	72	0
Satu Mare	0	0	0	0	0	0	0	0	3	1
Salaj	0	0	0	0	0	0	3	3	1	0
Sibiu	0	0	1	3	0	1	24	79	16	0
Suceava	0	0	0	2	0	0	10	70	30	1
Teleorman	0	0	0	3	1	0	10	10	3	0
Timis	0	5	2	8	0	0	23	62	38	3
Tulcea	0	0	0	0	0	0	6	23	5	1
Vaslui	0	0	0	0	0	0	45	27	4	0
Vâlcea	0	0	0	3	3	0	22	11	6	0
Vrancea	0	0	0	0	0	0	62	43	4	0

TABLE 2. Infection size on nodes - connected countries

.

Country	Infection population size									
	May 2009	June 2009	July 2009	Aug. 2009	Sept. 2009	Oct. 2009	Nov. 2009	Dec. 2009	Jan. 2010	Feb. 2010
Ukraine	0	1	1	0	6250	850000	11005	1230	301	22
Hungary	0	7	11	138	1250	1877	1107	203	70	3
Bulgaria	0	5	10	47	470	100000	2307	967	111	25
SUA	2254	20000	33902	6700	3200	1050	320	115	67	5
Canada	280	5438	7983	2060	986	320	98	20	0	0
UK	40	1540	7447	5957	8960	17325	6015	2000	200	0
Spain	93	430	760	838	2600	17303	1230	700	183	7
Mexico	1626	4957	10262	2350	1739	600	121	67	21	0
France	12	171	300	825	3024	7017	659	226	105	0
Turkey	0	27	40	50	180	625	303	29	15	0
Greece	0	58	109	1340	2506	2030	270	37	25	0
Germany	11	291	470	12320	1445	4445	750	217	32	12
Italy	9	86	130	1138	7213	21207	3070	375	93	31
Portugal	1	6	27	1960	1530	1248	625	123	75	1
Netherlands	3	100	134	1368	1020	2364	950	99	65	5

After the application has been executed, the values of the simulation will be as displayed in Figure 1 (there are displayed also the real results for a more obvious comparison).

The error expresses the number of extra or missing cases from the simulated results, compared with the real results. The value of 2.07 is an acceptable one as for the entire country there have been a mistake of this size which reported to the population of the country is meaningless.

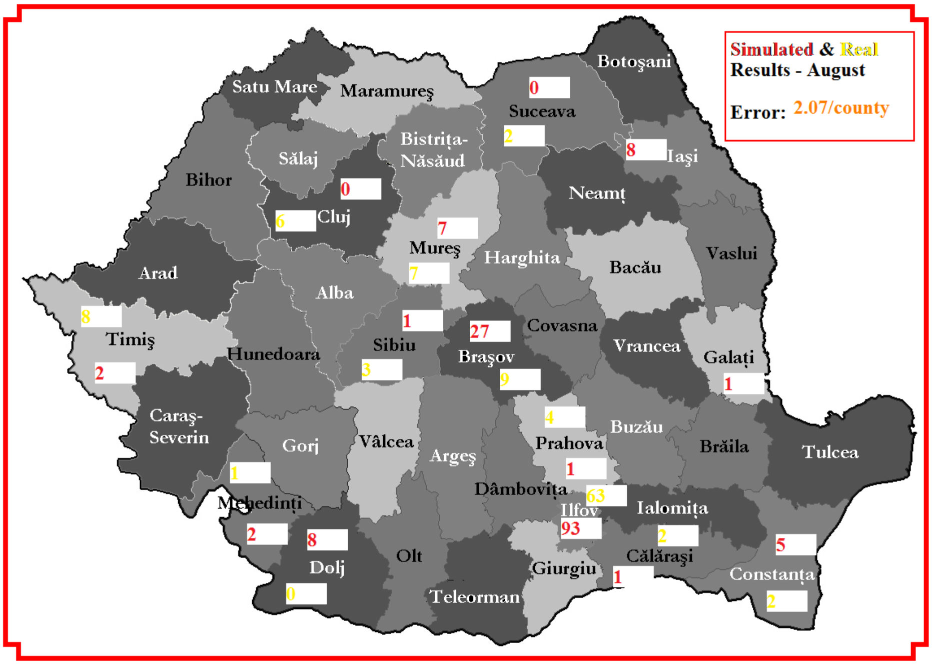
The arguments for the appearance of this error have a vast range of foundering reasons; however the simplest one is that the evolution of a virus on a real network is rather unpredictable. Still, one of the explications is that during the month of August it is possible that the theory of moving is not sustained as people can make their holidays abroad, not only in the country.

Experiment 2

The second experiment has as purpose to simulate the evolution of the epidemic over the counties of Romania during the month October 2009. The month which will be given as input will be *September.* The data for input, besides the one given by the user, is read from the local database and will be used for the construction of the network.

The network is constructed in a similar way to the one explained for the previous experiment, except for the fact that the size of the infected population

FIGURE 1. Results for the month August.



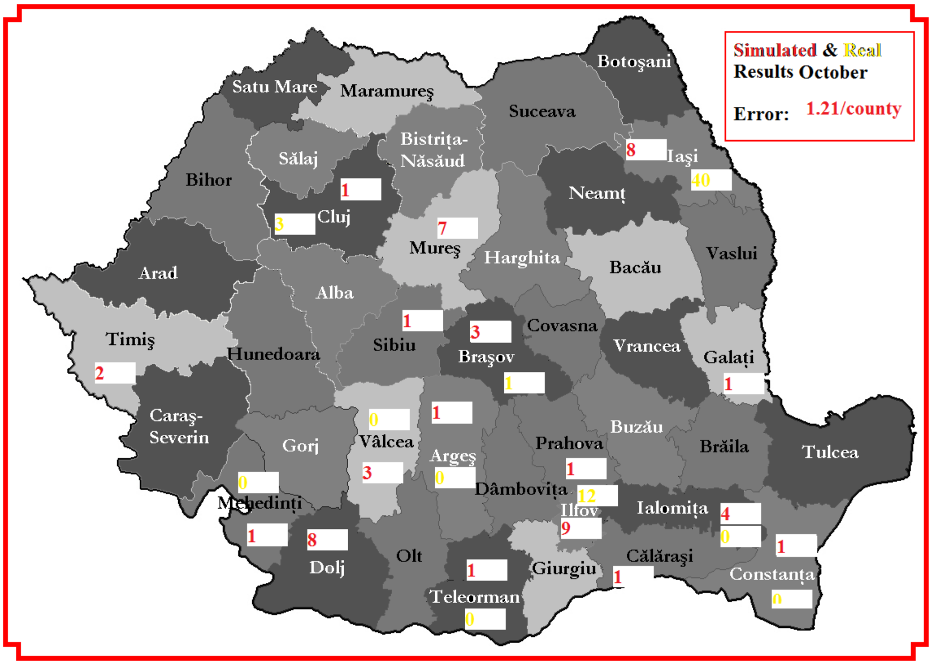
is taken from the column corresponding to the month September 2009 of the Tables 1 and 2.

The month October imposes some other limitations over the computations for the epidemic simulation. This month is known to be the month when the collegial year begins. The map for the month of October 2009 is depicted in Figure 2.

For this experiment there has been obtained a better results in means of distributing the bad-placed cases: 1.21 cases/county. In comparison with the country population, the difference is hardly noticeable.

One of the most relevant reasons for having these errors is the fact that October is an autumn month when the temperatures decrease and the infection likelihood increases, but slowly than in the succeeding months. In this way one of the draw-backs of the algorithm is that the “temporal constants” range are far too restrictive. At the moment a month has or not a property, it can not have it in a certain proportion, as it might be case for the month October.

FIGURE 2. Results for the month October.



6. CONCLUSIONS

The paper proposes a new approach for analyzing epidemic spreading over socials networks by introducing a new model tested against real-world results. The model is based on intensive research in social networks and epidemic spreading, viewed from different aspects: the mathematical way and the sociologic-statistical way. The selected data for taking part in the model has been restricted to a number of characteristics and support further extension.

The developed model is general and can be applied to any hierarchical structure similar to the one of Romania and it is valid for the simulation of the spreading of any other virus. The simulation algorithm can support modifications to fit any other epidemiological model. The application, although it is presented as a case of study, can be modified to have a general character: it can suit any country of the world, only with the change of data from the database and of the characteristics deduced from the time of year, which rather seem to be specific to Romania.

Acknowledgement. The second author acknowledges the support from the CNCSIS grand IDEI 2412/2009.

References

[1] J. Badham, R. Stocker, *The impact of network clustering and assortativity on epidemic behaviour*, Theoretical Population Biology, 77 (2010), pp. 71-75.

[2] M.F. Boni, B.U. Manh, P.Q. Thai, J. Farrar, T.T Hien, N.T. Hien N. Van Kinh, P. Horby, *Modelling the progression of pandemic influenza A (H1N1) in Vietnam and the opportunities for reassortment with other influenza viruses*, BMC Medicine, 7 (2009), pp. 43-47.

[3] G. Caldarelli, A. Vespignani, *Large Scale Structure and Dynamics of Complex Networks from Information Technology and Natural Science,* World Scientific, London, 2007.

[4] A. Degenne, M. Forsé, *Introducing social networks,* Sage, London, 1999.

[5] M.J. Keeling, K.T.D. Eames, *Networks and epidemic models,* J. R. Soc Interface, 2, (2005), pp. 295–307.

[6] V. Prejmerean, M. Frentiu, V. Cioban, and O. Ghiran, *Graphical representation of the pandemic spreading*, First International Conference on Complexity and Intelligence of the Artficial and Natural Complex Systems, Medical Applications of the Complex Systems, Biomedical Computing, Targu Mures, Romania, 2008, pp. 197-202.

[7] Romanian Health Ministry, Press release, http://www.ms.ro/?pag=62

[8] *Scale-free Networks,* http://www.scholarpedia.org/article/Scale-free_networks

[9] B. Wellman, *Structural Analysis: From Method and Metaphor to Theory and Substance* Cambridge: Cambridge University Press, London, 1988.

[10] S. Wasserman, K. Faust, *Social network analysis: methods and applications,* Cambridge University Press, 1994.

[11] D.J.Watts, S.H. Strogtz, *Collective dynamics of 'small-world' networks,* Nature, 393 (1998), pp. 440-442

Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, 400084, Cluj-Napoca, Romania

E-mail address: `silvia.rausanu@gmail.com, cgrosan@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

INTRODUCING VAST: A VIDEO-AUDIO STREAMING TESTER

ADRIAN STERCA AND CLAUDIU COBARZAN

Abstract. We present a testing package aimed at video and audio streaming across best-effort networks like the Internet. VAST is intended to be a testing framework for protocols transporting audio-video streams across IP networks. It offers the simplicity and predictability of deterministic simulators like ns-2 combined with the testing power of real-world experiments.

1. Introduction

The proliferation of multimedia data, especially the video and audio components, across the Internet has been exponential in recent years. As a result, the stress on the delivery infrastructure is increasing, as do the efforts to find and test audio-video streaming solutions in either simulated or real test environments. One of the most popular tools used for testing network transport protocols is the ns-2 simulator [8]. Although, simulation packages are in general easy to use and the obtained results are simple to assess due to their deterministic context, they lack the ability to capture the true randomness from real-world experiments, and thus they do not have the testing power of real-world experiments. On the other hand, because of all the randomness involved in real-world experiments and because of many noise sources, the results from those experiments are usually difficult to assess, as they reflect so many interconnections within the components of the experiment.

Our aim is a testing framework which does not rely on simulations, but borrows some of its deterministic approach and simplicity in assessing the test

Received by the editors: August 15, 2010.

2010 *Mathematics Subject Classification.* 90B18, 68M20.

1998 *CR Categories and Descriptors.* C.4 [**Computer Systems Organization**]: Performance of Systems – *Measurement techniques*; I.6.3 [**Computing Methodologies**]: Simulation and Modeling – *Applications*.

Key words and phrases. video-audio streaming, testing, rate control, congestion control, FFMPEG.

results and it also tries to capture true randomness of real-world experiments by using the available networking infrastructure as test environment, not a simulated network one. The name of this framework is **VAST**, which is an acronym for **Video-Audio Streaming Tester**.

The paper is structured as follows. In Section 2 we present other testing tools available on the Internet and related to VAST. Section 3 underlines the goals of VAST while the next section introduces the architecture of the proposed package. Section 5 deals with the pluggable structure of a VAST experiment. Section 6 outlines VASTs strengths and weaknesses compared to the testing tools presented in Section 2 while Section 7 presents our conclusions and intended future work.

2. Related Work

Ns-2 [8] is a well known network simulation package which is focused on congestion control. Although it is very powerful and has implementations for a wide set of transport protocols, it does not capture the possible benefic influence of real-world audio-video data on the network transport control algorithms and, conversely, it does not reflect the effect of network transport control algorithms on the audio-video stream perceived by the end user.

Another testing package for audio-video streaming is ViTooKi [10] which is an open source tool for developing real world audio-video streaming applications. One of the disadvantages of ViTooKi is that it has very low support for media-friendly and TCP-friendly congestion control and congestion control in general.

VLC [11] is yet another open source streaming project similar to ViTooKi which also implements limited congestion control (besides the one implemented by TCP in the OS kernel).

3. Package Goals

By using the VAST package, we want to be able to test in real network conditions (not in simulated environments) the performance of various congestion control as well as stream rate control algorithms for audio-video streaming in best-effort networks. Their performance will be measured by objective metrics (e.g. number of lost packets, throughput, buffer fill level etc.) but also by pseudo-subjective metrics (artifacts on the rendered video etc.).

Another goal of VAST is testing the performance of caching strategies for multimedia (audio-video) objects in audio-video proxy-caches [1].

Finally, we want to have the possibility to test in real-world conditions, algorithms for adapting multimedia streams at application level (e.g. temporal adaptation, grayscale reduction, spatial reduction, transcoding etc.).

The main advantages of the proposed package over other network simulators (e.g. ns-2) are the following:

- the data source is a real audio-video stream and is not random data from memory; because of this the distribution of the audio-video data can influence the semantics of the rate control algorithms;
- the performance of the transmission rate control and stream rate control algorithms can be visually assessed/quantified.

Because it uses real-world conditions in experiments, VAST has also some drawbacks compared to network simulators and the most important is the difficulty to assess the test results due to many noise sources generated by randomness in the experiments conditions (which is a characteristic of all real-word experiment packages). This difficulty is counteracted by the pluggable architecture of VAST which is described in the next section.

4. The Vast Architecture

Generally speaking, there are two approaches in network testing: *the deterministic approach – simulations* and *real-world experiments*.

In network simulations, it is easier to measure, predict and interpret results and no costly network infrastructure is needed for deployment. The main weakness of network simulations is the lack of the ability to capture true randomness and transient events. The exponent of network simulators is represented by ns-2 [8].

In real-world experiments, it is harder to measure and interpret results (i.e. difficult to isolate causes) and an elaborate network infrastructure is required for deployment. The advantage of real-world experiments is the possibility to capture the randomness in the real-world and transient events. Examples of tools useful for audio-video streaming testing are ViTooKi (The Video Toolkit) [10] and VideoLan VLC[11].

VAST tries to combine the advantages of both approaches mentioned above. On one hand, it lies in the real-world experiments category because it relies on real-world network infrastructure and network conditions, thus having a testing power larger than a simulator. On the other hand, through its flexible pluggable architecture, VAST provides an easy way of assessing test results, as do simulators. This is because in each experiment, many components of VAST can be *plugged-in* or *plugged-out* from the experiment, thus adding or removing a noise/randomness source from the experiment. By subsequently eliminating a noise source from the experiment, the test results can be better separated and more easily assessed.

The intended VAST testing architecture is shown in Figure 1. The VAST package can be deployed on a LAN which is linked by several routers. VAST has different data flow sources which can be monitored when competing for network resources: TCP flow source, fixed (parametrized) transmission rate UDP source and audio-video streaming sources/servers. Each VAST source type has a corresponding receiver. Also an audio-video proxy/cache module can be deployed inside the VAST network.

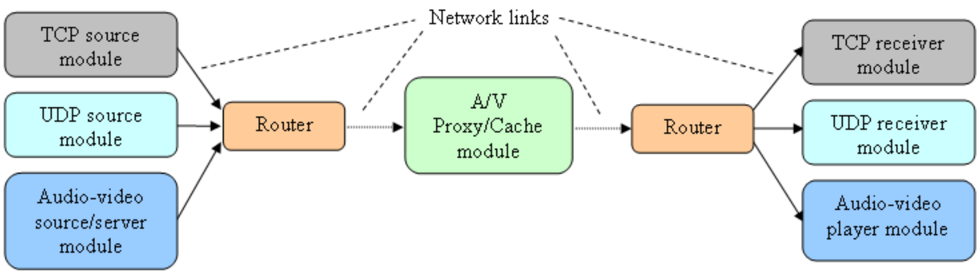


FIGURE 1. The VAST testing architecture

The VAST A/V player architecture is depicted in Figure 2 and has the following major components: *IO System* for interfacing with the network, *Session Management* for controlling the streaming session using standard streaming protocols and also our experimental streaming protocol (SMSP – Simple Multimedia Streaming Protocol), a *Codec Module* based on FFMPEG [7] for decoding audio-video data and a *Rendering Module* based on the SDL library [9].

The *IO System* is the flow control component. On the client side (i.e. A/V player) the *IO System* consists of feedback producers for various flow control protocols. On the server side, the *IO System* contains flow control components for computing/updating the transmission rate and enforcing the computed transmission rate. The A/V streaming session can use several flow control protocols: DCCP [4], TCP and newer flow control algorithms build on top of UDP: NoCC (No Congestion Control - an algorithm for sending the

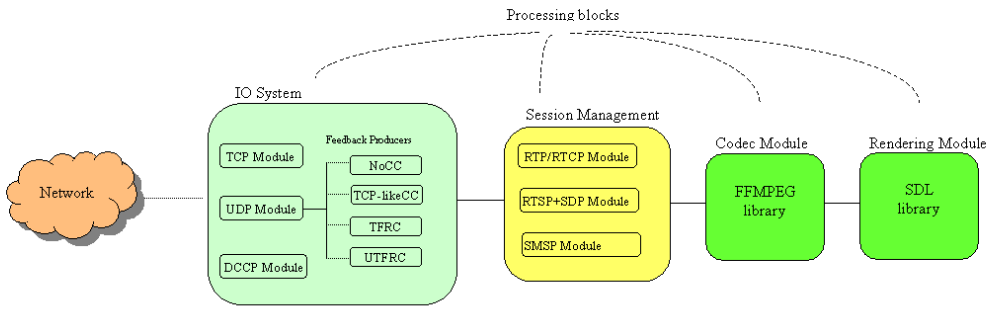


Figure 2. The VAST Audio/Video Player architecture

audio-video data at constant rate), TCP-like congestion control, TFRC [3], UTFRC [5, 6].

The *Session Management* component provides implementations for standardized streaming protocols (RTP, RTCP) and streaming control protocols (RTSP, SDP) and also our experimental SMSP – Simple Multimedia Streaming Protocol, whose header consists only of a sequence number and a timestamp.

In Figure 3 the architecture of the A/V streaming server is depicted. It has similar components as the A/V player.

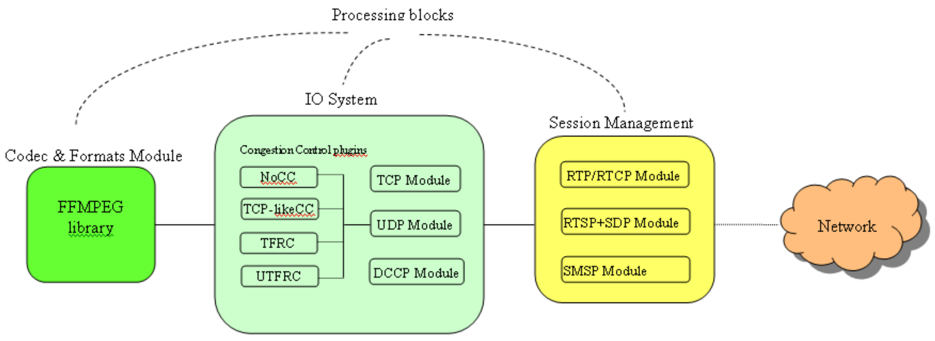


Figure 3. The VAST Audio/Video Server architecture

5. The pluggable architecture of a VAST experiment

It is important to note that almost each component of the server or client can be disabled in a VAST experiment setup which offers flexibility in isolating causes (noise sources). This is very useful in experiments in which the obtained results can not be explained due to the interconnections of the experiments components and other noise sources. In such a case each component can be independently and subsequently disabled (actually, it is not fully disabled, but replaced with a dummy one which does not generate too much noise because of its simplicity – it leaves data unchanged), until each noise source is identified. For example, if we do not want to consume CPU for the actual rendering of the video or we want to reduce the concurrency randomness at the client, we can disable the SDL-based *Render Module* and the *Codec Module* from the Player and no decoding or display will happen at the Player. Another example is that full fledged session management protocols like RTP/RTC, RTSP and SDP can be disabled and a simple custom header (SMSP) can be used instead, thus reducing the degree of concurrency and complexity of the data flow. Also, the congestion control module can be disabled at the server and replaced with a NoCC, one which transmits multimedia data at a constant rate, thus eliminating variations (randomness) in the transmission rate. To reduce the computing load at the server, the *Codec Module* can be disabled and random data from the memory can be sent instead to the client.

In essence, the architecture of a VAST streaming server or player is a highly configurable and a pluggable one.

6. VASTs advantages and disadvantages

Because of its pluggable and flexible architecture, the VAST package has several advantages over existing audio-video streaming simulators or streaming solutions:

- the ability to test new media-friendly congestion control algorithms which are influenced by the data distribution in the audio-video stream; this is because an audio-video source uses a real audio-video stream instead of random data from memory;
- the performance of the congestion control and stream rate control algorithms can be visually assessed (in the quality of the stream perceived by the client);
- testing specific proxy-cache operators can be done in real-word network setups/conditions;
- the possibility to isolate noise sources in the experiment results by enabling/disabling various components in the streaming server or client by taking advantage of the pluggable architecture of VAST.

Due to the fact that it is still in the development state, VAST has the disadvantage that is not as stable as a classic simulator like ns-2 which has been used for a number of years. Also because it is a real-world experiment tool, it does not have the deterministic properties of a network simulator, thus the results can not be as easily assessed as in the case of a simulator. And because real network infrastructures are used in experiments, the setup for the experiment can take longer than in the case of simulators.

7. Conclusions and Future work

We presented in this paper VAST, a video-audio streaming tester package based on FFMPEG. The strength of VAST comes from its highly pluggable and configurable architecture.

The current state of the implementation is:

- functional local video player;
- simple video streaming server;
- partial congestion control implementation (TFRC/UTFRC);
- naive TCP/UDP sources/receivers implemented.

As future work, we intend to complete the implementation of VASTs modules and compare the VAST package with the previously developed *vpcSim* [2] tool.

References

[1] Claudiu Cobarzan, *Distributed Video Proxy-Caching in High-Bandwidth Networks*, PhD thesis, 2008

[2] Claudiu Cobarzan, Doriana Dorutiu, *vpcSim: A Video Proxy-Cache Simulator*, In Proceedings of the Symposium Colocviul Academic Clujean de Informatica, pp. 135-140, 2005

[3] Sally Floyd, Mark Handley, Jitendra Padhye, Joerg Widmer, *TCP Friendly Rate Control*, RFC 3448, January 2003.

[4] Eddie Kohler, Mark Handley, Sally Floyd, *Datagram Congestion Control Protocol (DCCP)*, RFC4340, 2006

[5] Adrian Sterca, *Congestion Control in Streaming Protocols*, PhD thesis, 2008

[6] Adrian Sterca, *UTFRC - Utility-driven TCP-Friendly Rate Control for Multimedia Streams*, in Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, IEEE Computer Society, Germany, February 2009.

[7] *** FFMPEG library, `http://ffmpeg.org/`

[8] *** The ns-2 simulator, `http://www.isi.edu/nsnam/ns/`

[9] *** SDL library – Simple DirectMedia Layer, `http://www.libsdl.org`

[10] *** ViTooKi – The Video Tool Kit, `http://vitooki.sourceforge.net/`
[11] *** VLC, `http://www.videolan.org/vlc/`

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084-Cluj-Napoca, Romania
E-mail address: {`forest,claudiu`}`@cs.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

JAVASCRIPT GENERATORS

GHIŢĂ DANIEL-CONSTANTIN

Abstract. The JavaScript language is used for client-side programing in web applications. JavaScript is sent as source code from the web server to the user's browser and executed locally. Because not all current browsers support the latest version of JavaScript, developers are limited to use only the features supported by the lowest common denominator. I propose an approach to solve this problem by using an intermediary compile step which transforms the code written by the user into code supported by all browsers. This allows web developers to use the latest version of JavaScript today, without having to wait until all major browsers implement all the features.

1. Introduction

JavaScript is commonly known as the language of the web. This object-oriented scripting language is implemented in all modern browsers and it is the only cross-platform language available for client-side programming in web application. It is a dialect of the ECMAScript standard [6].

JavaScript is rapidly evolving. Because some implementations are lagging behind, developers cannot take advantage of all the features which have been added to the language. This is further amplified by the presence in the wild of browsers which are not up to date.

We can address this issue using static compiling. Using a compiler we can transform code which is written to use the latest version of JavaScript into code which uses only the basic syntactic constructs and so is (syntactically) compatible with all implementations in the wild. Using a library which will be loaded at run-time the program can add any missing methods to the core objects of the language. We will show how to address each issue individually with examples and offer an implementation of a compiler which integrates all the pieces. The result is a translator which takes as input code written in

Received by the editors: June 10, 2010.

2010 *Mathematics Subject Classification.* 68N20.

1998 *CR Categories and Descriptors.* D.3.4 [**Software**]: PROGRAMMING LANGUAGES – *Processors*.

Key words and phrases. JavaScript, ECMAScript, web applications, compiler, program analysis.

the latest version of JavaScript and outputs code which works on all current JavaScript platforms.

This rest of the paper is structured as follows. Section 2 is structured in two subsections: the first one offers an overview of JavaScript, while the second section takes a look at what benefits compiling has offered other languages. Section 2.3 presents the current status of methods applied for JavaScript source code. Section 3 presents the goal of this paper in more details. Section 4 details each step of the path in achieving said goal, and in the end we give some conclusion that evaluates the results which have been obtained and the roads opened for future work.

2. Background

2.1. **A Brief Overview of JavaScript.**

2.1.1. *Where JavaScript is today.* There are many different implementations of ECMAScript dialects and a few call themselves JavaScript. Every major browser has its own ECMAScript implementation and there are even compilers and interpreters for ECMAScript as a systems language. ECMAScript covers only the core language and many extensions, including "host objects", are left to the implementation:

> ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific *host* objects, whose description and behavior are beyond the scope of this specification[6]

2.1.2. *Basic building blocks.* Often described as "Lisp in C's clothing"[4,5], the JavaScript language is dynamic and weakly typed. It has first-class functions and prototype-based object inheritance. It has closures, which means functions have lexical scope and they can access the variables from the context in which they were created. Functions can be called with any arguments (and any number of arguments). All objects are hash tables and objects cannot be created *ex-nihilo.* Objects and arrays can be written as literals and the language has built-in support for regular expressions. The logical OR (`||`) and logical AND (`&&`) are short-circuiting operators. The ternary operator is similar to the one found in C.

2.2. **Benefits of compiling.** Compiling is the process of transforming code from one computer language into another—called the *target* language. Usually compilers target machine code which can be directly executed by a computer or virtual machine. Historically, compilers have been a major factor in the advance of programming by allowing the same code to be used on different architectures and by performing tedious tasks on the behalf of the programmer, such as checking type correctness, optimizations, relaxing the limitations of the machine (e.g. aligning data at certain memory locations), supporting high-level constructs not available on the underlying hardware, linking (distributing a program across more than one source file) and more. In short, compiling enables easier programming and portability.

2.2.1. *Source-to-source compilers.* While most compilers output machine code or bytecode, *source-to-source compilers* use a high-level language for both the input and the output. Source-to-source compilers can be used, for instance, for adding parallel code annotations (e.g. OpenMP) or for porting code from an old version of the language into a new version.

2.3. **Review of the State of the Art.** In this section we will look at what methods have been previously applied to JavaScript source code before.

2.3.1. *Minification.* Minification [5]is the process in which an input source code is transformed into an output source which, when executed, results in the same operations being performed on the same data. The program which performs this transformation on the code is called a minifier. The goal is usually to obtain a program which works identically to the input code but which has a smaller size. In order to achieve this goal, the minifier employs more than one technique:

- Whitespace removal. Indentation, line feeds and any other whitespace is deleted because it does not affect the semantics of the program.
- Comments are eliminated, for the same reason as whitespace. Usually JavaScript minifiers understand and preserve conditional comments. Also, some minifiers preserve comments which begin with '`!`' or which contain '`@license`' or '`@preserve`'. This is used in order to preserve copyright information.
- Redundant characters with syntactic but not semantic meaning are removed. For instance, many semicolons can be deleted without affecting the instructions because JavaScript allows the programmers to omit semicolons in some places.
- Numbers are written in the shortest form possible. For instance, `1000` could be transformed into `1e3` and `1e1` could be transformed into `10`, each modification saving one character.

- Local variables and function arguments are renamed to use less characters. This can be particularly problematic in dynamic languages which support `eval(...)` such as JavaScript. Because the argument to be evaluated is a string and is usually composed dynamically or received from an external source (e.g. a response from the server), any references to local variables inside the string cannot be updated by the minifier. This means that executing `print(eval("currentIndex + pageSize"));` after minification will result in two undefined references[1].
- Transform conditionals where possible from `if`-statements using short-circuiting operators. A short-circuiting boolean AND operator does not evaluate the second argument if the first is false. Some tools take advantage of JavaScript's boolean operators and transform statements such as `if (a) b();` into `a && b();`.

Minification is usually safe, unless local variables are obfuscated and `eval(...)` is used to reference local variables.

2.3.2. *Packing.* Packing[5] is the process in which the identifiers, keywords and constants are separated from the code and moved into a separate list. Duplicates are discarded and placeholders are used in the code to know where to insert each item. Two strings are obtained: the first contains the code without the removed parts and the second contains all the removed parts with a separator. The final file has the two strings and a few instructions which reassemble the original file (as a string) and feed it to `eval(...)`. Packing is usually used after the source has already been minified.

Tools which implement minification of JavaScript sources include Closure Compiler, JSMin, Microsoft AJAX Minifier, Packer, ShrinkSafe and Yahoo! UI Compressor. Packer is the only one which implements packing.

2.3.3. *Function inlining. Function inlining* or *inline expansion* is the process in which a call to a particular function is replaced with the body of the function. While this increases the size of the code if the function is called from more than one place, it optimizes the execution speed of the program if the function is called often because it eliminates the overhead caused by a function call.

2.3.4. *Aliasing. Aliasing* is the situation in which a data location can be accessed under different names called aliases. In JavaScript aliasing is often used in conjunction with calling the same method on the same object repeatedly.

[1]Local variables and function arguments are not renamed in functions which use a `with` instruction anywhere in the source because declaring variables inside a `with` statement creates properties inside the object instead of local variables.

For instance, creating an alias for the `appendChild` method of the <body> element in an HTML document:

```
function add(newNode) {
    document.body.appendChild(newNode);
}
```

Function inlining is implemented in Closure Compiler.

2.3.5. *Turning Java into JavaScript.* A different approach which has been tried is to give programmers a language with static typing (specifically Java) and then translate that language's syntax into JavaScript constructs. This has the advantage of static type checking and support from the IDEs which target the guest language[2]

The collection of tools which compiles Java into JavaScript is Google Web Toolkit.

2.3.6. *Asynchronous JavaScript.* While JavaScript has features which enable it to work asynchronously (setTimeout, setInterval, asynchronous connections to the server) it is single-threaded and execution itself is synchronous: once a function begins executing, it cannot sleep or pause execution in anyway; it will either terminate or enter an infinite cycle (in which case it will ultimately be killed by the browser). Two techniques have been used to bring this functionality into JavaScript using libraries.

The first method involves using a loader which will scan the JavaScript source before it is loaded into the browser and modify it before it is compiled. This works in all browsers but is very slow.

The second method consists of using the `yield` instruction (which is designed for generators)[7,8]. This takes advantage of a pause-and-resume support built into generator and thus it is more efficient than the first method. The major downside is that it requires JavaScript 1.7 which is currently only available in Mozilla implementations.

2.3.7. *ECMAScript 5 strict subsets.* There has been at least one attempt to implement a strict subset of ES5 strict mode on top of ES3[2]. This has been done in order to limit the flexibility of the language such that modules written using the allowed subset of the language are easier to isolate. This does not actually add anything to language, it just removes some building blocks. Google's Caja project implements two languages: *Cajita* is a strict subset of ES5 strict mode with many features removed (e.g. `eval`, `with`, the `Function` constructor, monkey patching, etc.[3]) and *Valija* is an extension of

[2]JavaScript IDEs are rather poor compared to C++, C# and Java IDEs. For instance, Microsoft Visual Studio 2010 is still unable to understand regular expression literals in JavaScript and tries to auto-complete the text inside and the flags, although the syntax has been in the language since the beginning.

Cajita which adds back some of the elements which have been removed, but with a limited scope (e.g. `eval`).

3. Problem Statement

3.1. **Existing browsers.**

JavaScript, as shown in section 1 and section 2, is a fast-evolving language with a huge installed base. Except C and maybe C++, it is a challenge to find one language which can be executed on so many devices. It is more widespread than even Java—every device which can run Java can also execute JavaScript using Rhino (a Java library) and there are many mobile devices which come with a browser but no Java runtime. Not only are there JavaScript shells and cross-platform FastCGI modules(also `#/bin/v8cgi`[1]), but the most popular desktop operating system, Microsoft Windows, comes with a built-in JavaScript environment capable of interacting with the system tools (and no Java runtime unless the user explicitly installs it).

Together with the size of the installed base also comes a disadvantage: fragmentation. Some implementations are lagging behind and, when writing web applications, generic libraries, or other tools, developers are forced to code for the smallest common denominator. This considerably limits the capabilities which can be used and denies the improvements which could be gained: faster development, less error-prone code, better performance and more.

3.2. **Need for compatibility.**

While there are many tools which process JavaScript source code, they do not extend it with the latest constructs. The tools either translate other languages to JavaScript constructs[3]. No tool has attempted to implement the new functionality offered by newer versions on top of the common baseline and the topic has not even been formally analyzed before.

Additionally, none of the tools available are written in JavaScript. This is important because JavaScript's `Function` constructor and `eval` function means code will be loaded from external sources, sometimes even on-demand. Unless the tool is implemented in JavaScript, it cannot catch calls to load extra code at runtime and handle the new code.

3.3. **Target and impact.**

Today, usually only baseline constructs can be used when developing JavaScript applications[4]. The ones most affected by this are, of course, developers working on JavaScript frameworks. Such frameworks can

[3]If the code must be written in a language other than JavaScript, this usually results in rendering the developer unable to use certain high-level features which are not available in the original language; this may include prototype inheritance, variadic functions, duck typing, iterators, generators, loading code on-demand, etc.

[4]The most notable exception consists of Firefox add-ons. Obviously, they can take advantage of the latest JavaScript implementation which is used by the browser.

also be used outside of the web and into the realm of gadgets, where JavaScript is probably the most popular language[5]. As expected, JavaScript is also used when writing add-ons for browsers[6]. JavaScript is also popular in other circles, as a plugin language for many environments (PDF files, Nokia's Qt framework, OpenOffice, Google Docs, Microsoft Office and more). JavaScript is also used as a general-purpose application programming language (Palm's webOS which is used for phones and other platforms, the GNOME Shell, KDE's kjs, etc).

4. Implementing JavaScript generators in ECMAScript 3

In order to address the issues presented we have built a compiling framework called *Alkaline*[7]. The framework is written in JavaScript in order to provide support for re-entry during runtime when the `Function` constructor or `eval` function are invoked.

4.1. **Architecture overview.** Alkaline is composed of a few modules and some glue which sends data from one module to the next. Between each stage, an Abstract Syntax Tree (AST) is used as the intermediary representation for the program. Four modules are provided:

- A parsing module which takes JavaScript source code and builds an AST. This module is built using the ANTLR parser generator and includes two distinct parts, a lexer and a parser.
- *Analyzer* annotates the AST (for instance, this resolves variable references).
- *5to3* transforms AST structures from the latest JavaScript specification into structures supported by ECMAScript 3.
- *Printer* transforms the AST back into source code.

4.2. **Generators.** Generators have been introduced into the JavaScript language from version 1.7[9] and they have been described as "a better way to build Iterators"[10]. There are two main differences between a normal function and a generator:

- generators persist the state of local variables after they *yield* each result
- on each subsequent invocation generators continue execution from the `yield` statement which generated the last item

[5]The following environments use JavaScript: Apple's Dashboard Widgets, Microsoft's Gadgets, Yahoo! Widgets, Google Desktop Gadgets and Serence Klipfolio.

[6]To be more specific, this applies to all the major browsers except Internet Explorer: Firefox, Chrome, Opera, Safari and maybe others.

[7]Alkaline is open source and available at `https://code.google.com/p/alkaline-js/`.

4.2.1. *A simple example.* Listing 1 illustrates a simple generator which yields the next odd number each time it is invoked. The generator is invoked five times in order to return the first five odd numbers.

Listing 1. A simple generator

```
function oddNumber() {
    var i = 0;
    while (true) {
        if (i % 2)
            yield i;
        i++;
    }
}

function printOdd(count) {
    var iter = oddNumber();
    while (count--)
        print(iter.next());
}

printOdd(5);
```

In a way, the generator itself works somewhat similar to a separate thread: once created, it has its own state and even though it never ends, it does not block the main thread.

We can take advantage of the fact that JavaScript offers first-class functions and translate generators into regular functions, and use `yield` statement. In order to demonstrate how the method described affects the code, in Listing 2 we implement the previous simple generator using only ES3 constructs.

The code in Listing 2 takes advantage of the fact that in JavaScript functions are used as constructors and transforms the generator-function `oddNumbers` into both a class and a function which returns a new instance of the class (line 2).

In order to simulate the resume behavior, all blocks of code leading up to the `yield` statement are placed behind `if` statements which test `_continuation` such that when "resuming" execution, they are skipped. All loops and conditionals which gate the execution path toward the `yield` instruction are injected code such that they always take the correct branch when `_continuation` is `true`.

Listing 2. A simple generator modified to use only ECMAScript 3 constructs

```
function oddNumber() {
    if (!(this instanceof oddNumber))
        return new oddNumber();
}

oddNumber.prototype.next = function() {
    if (!this._continuation) {
        this._local_i = 0;
    }
    while (this._continuation || true) {
        if (this._continuation || this._local_i % 2) {
            if (!this._continuation) {
                this._continuation = true;
                return this._local_i;
            } else
                this._continuation = false;
        }
        this._local_i++;
    }
}

function printOdd(count) {
    var iter = oddNumber();
    while (count--)
        print(iter.next());
}

printOdd(5);
```

4.2.2. *Multiple `yield` statements, `StopIteration`.* A natural extension of the simple generator previously presented is the support for multiple `yield` instructions. This increases the flexibility of the method but also complicates the implementation which must support it. Additionally, if after a call to `next()` the generator "terminates" without yielding anything, it automatically throw an error which inherits `StopIteration`. Listing 3 demonstrates all these mechanisms at work. The exception `StopIteration` is easy to solve here by simply adding a `throw` statement at the end of the generator body.

The conditions which gate execution when resuming after a `yield` will need to be updated. Basically, at any point in the source, if we are resuming to a `yield` which is placed after the instruction at the current position we must skip executing the instruction we are considering. If, on the other hand, the `yield` to which we are resuming is inside one of the branches which belongs to the current instruction then we must guide execution toward it. This means that if the instruction gates the third `yield` we must skip it completely when `this._continuation > 3` and we must enter one of the branches when `this._continuation == 3`. Listing 4 applies this approach to the generator from listing 3 mentioned previously. If the resume flag indicates a `yield` which is placed before the current instruction, the code should behave as if the flag is cleared (i.e. normal execution).

Listing 3. A generator with multiple yield statements using only classic constructs (simplified)

```
function multipleGenerator() {
    if (!(this instanceof multipleGenerator))
        return new multipleGenerator();
    this._continuation = 0;
}
multipleGenerator.prototype.next = function() {
    if (this._continuation < 1) {
        this._continuation = 1;
        return "first";
    }
    if (this._continuation < 2) {
        this._continuation = 2;
        return "second";
    }
    if (this._continuation < 3) {
        this._continuation = 3;
        return "third";
    }
    throw new StopIteration();
}

var g = multipleGenerator();
print(g.next()); // prints "first"
print(g.next()); // prints "second"
print(g.next()); // prints "third"
print(g.next()); // throws an error which is an instance of↩
    StopIteration
```

4.2.3. *The `send(...)` method, generator arguments, exceptions.* An important feature of generators is the ability to interact with a generator which has been started. The `yield` statement does more than just pause execution and wait for it to be resumed later. It can return a value and sometimes even throw an exception. By default, it returns undefined.

When `generatorInstance.send(something)` is called, the generator will be resumed (similar to calling `next()`) and the previously paused `yield` instruction will return `something`. This makes it possible to send new information into the generator when it is resuming. By calling `generatorInstance.throw(exception)` the generator will be resumed and the previously suspended `yield` will throw the given exception. Additionally, similar to functions, generators can take arguments. Listing 4 shows a generator for the set of natural numbers. The generator receives an optional starting value—which defaults to 0—and return a new number each time `next()` is invoked. By calling `send(number)` a value can be fed into the generator in order to reset its position to an arbitrary point. This listing demonstrates the use of arguments for generators and the `send(...)` method.

Listing 4. A generator which counts to +Infinity

```
function counter(start) {
    if (!start)
        start = 0;
    while (true) {
        var restart = yield start++;
        if (!isNaN(restart))
            start = restart;
    }
}

var f = counter();
print(f.next()); // prints 0
print(f.next()); // prints 1

f = counter(3);
print(f.next());  // prints 3
print(f.send(8)); // prints 8
print(f.next());  // prints 9
```

It should be noted that both `send(...)` and `throw(...)` wake up the generator. Of course, calling `send(undefined)` is the same as calling `next()`. Calling `send(something)` before the generator yields the first value will throw a TypeError.

Adding support for generator arguments is easy: the names of the arguments are added to the list of local variables and the arguments received in the generator constructor are saved just like local variables. In order to support the `send(...)` method, the generator must take the new value when resuming.

Listing 5 translates the generator `counter` into basic constructs with support for `send(...)`, `throw(...)` and generator arguments.

LISTING 5. A generator which counts to +Infinity using only ES3 constructs

```
function counter(start) {
    if (!(this instanceof counter))
        return new counter(start);
    this._continuation = 0;
    // the unlikely name is used to obtain a reference to ↩
       the primitive <undefined>
    this._yield_value = this.thisPropertyDoesNotExist;
    this._yield_exception = null;
    this._local_start = start;
}
counter.prototype._next = function() {
    var restart;
    if (!this._local_start)
        this._local_start = 0;
    while (true) {
        return this._local_start++;
        restart = this._yieldReturn();
        this._local_restart = restart;
        if (!isNaN(this._local_restart))
            this._local_start = restart;
    }
}
counter.prototype.next = function() {
    var nextValue = this._next();
    if (this._continuation < 0)
        throw new StopIteration();
    return nextValue;
}
counter.prototype._yieldReturn = function() {
    var exception;
    if (exception = this._yield_exception) {
        this._yield_exception = null;
        throw exception;
    } else {
        var value = this._yield_value;
        this._yield_value = this.thisPropertyDoesNotExist;
        return value;
    }
}
counter.prototype.send = function(yieldValue) {
    if (typeof yieldValue != 'undefined') {
        if (this._continuation == 0)
            throw new TypeError();
        this._yield_value = yieldValue;
    }
    return this.next();
}

var f = counter();
```

```
print(f.next()); // prints 0
print(f.next()); // prints 1

f = counter(3);
print(f.next());  // prints 3
print(f.send(8));    // prints 8
print(f.next());  // prints 9
```

4.2.4. *The `close()` method and `finally` blocks.* To quote from a previous paragraph:

> If, on the other hand, the `yield` to which we are resuming is inside one of the branches which belongs to the current instruction then we must guide execution toward it.

The `close` method is the opposite of this: when the user invokes it, the generator must return to the point in the source where it last left off, but instead of starting to execute instructions it must execute all the `finally` clauses and nothing else. This brings into discussion another aspect which was neglected until now: `finally` clauses. Whenever a JavaScript function returns, any finally clauses which are active are executed. This happens only for `return` and not `yield` because the latter is a pausing mechanism and execution is expected to resume later. By replacing `yield` statements with `return` equivalents we are changing the behavior: the `finally` clauses will be executed for each ”yield” and—depending on what code is inside them—this may cause problems. The solution is simple, all we have to do is gate the content of the `finally` clause so it is only executed when the generator is closing and when it is not paused (e.g. `finally { ... }` becomes `finally { if (normal execution or closing) { ... }}`).

4.2.5. *When the generator uses `eval(...)`.* If the body of the generator calls the function `eval` then additional steps must be performed. Because the code compiled and executed by `eval` may reference local variables and we have moved all of them into properties of the generator instance, we must either update the code which eval will compile so all the references are resolved, or make sure the variables exist and are up to date before calling `eval`. The latter option is simpler than parsing, analyzing and changing the code which will be executed. For instance, if a generator uses two variables, `a` and `b`, and the first one is the name of a function, the line `b = eval(a+ "()")` would be changed into `var a = this._local_a, b = this._local_b; this._local_b = eval(a + "()")` which works as expected.

4.2.6. *Putting it all together.* In this section we have built a solution for translating generators for JavaScript into a structure which uses only ECMAScript version 3 constructs and is therefore supported in all ECMAScript environments. No code outside of the generator has to be changed which makes this a local solution. Listing 6 show the template for implementing a generator. The steps are:

(1) If the class `StopIteration` does not exist, create it and make it inherit `Error`.
(2) Create the class for the generator. Inside the constructor, create the start state with the arguments. Save all the arguments in local properties of the generator instance. Move the generator-function to the `_next` method.
(3) Redirect all references to arguments and local variables into properties of the generator instance.
(4) Implement `_continuation` and transform `yield` statements into `return` statements which also push the new state to the list of states. Update `finally` clauses so they don't do anything when the function returns.
(5) If the function `eval` is used, make all the variables reference the current (saved) state before invoking `eval`.
(6) Add the static methods `next()`, `_yieldReturn()`, `send(yieldReturn)`, `"throw"(exception)` and `close()`. It is also possible to place these methods inside a single object and inherit it using the `prototype` chain (which would generate less code when there is more than one generator).

Listing 6. The template used to translate a generator into ECMAScript 3 constructs

```
if (typeof StopIteration == 'undefined') {
    var StopIteration = function(){};
    StopIteration.prototype = new Error();
}

function generatorName(arg1, arg2, ...) {
    if (!(this instanceof generatorName))
        return new generatorName(arg1, arg2, ...);
    this._continuation = 0;
    this._yield_value = this.thisPropertyDoesNotExist;
    this._yield_exception = null;
    this._closing = false;
    this._local_arg1 = arg1;
    this._local_arg2 = arg2;
    ...
}
generatorName.prototype._next = function() {
    ... actual generator code ...
}
generatorName.prototype.next = function() {
    var nextValue = this._next();
    if (this._continuation < 0)
        throw new StopIteration();
    return nextValue;
}
```

```
generatorName.prototype._yieldReturn = function() {
    var exception;
    if (exception = this._yield_exception) {
        this._yield_exception = null;
        throw exception;
    } else {
        var value = this._yield_value;
        // the unlikely name is used to obtain a reference ↩
            to the primitive <undefined>
        this._yield_value = this.thisPropertyDoesNotExist;
        return value;
    }
}
generatorName.prototype['throw'] = function(exception) {
    this._yield_exception = exception;
    return this.next();
}
generatorName.prototype.send = function(yieldValue) {
    if (typeof yieldValue != 'undefined') {
        if (this._continuation == 0)
            throw new TypeError();
        this._yield_value = yieldValue;
    }
    return this.next();
}
generatorName.prototype.close = function() {
    this._closing = {}; // create a new object
    while (this._closing)
        try {
            this['throw'](this._closing);
        } catch(e) {
            if (e == this.closing || (e instanceof ↩
                StopIteration))
                this._closing = false;
            else
                // a different error was thrown
                throw e;
        }
}
```

5. Conclusion

JavaScript generators from language 1.7 can be successfully implemented on top of the baseline ECMAScript 3 using a source-to-source compiler.

The proposed method is complete and does not sacrifice any feature which JavaScript brings to the table in order to bring this support to legacy environments.

A compiler toolkit has been implemented which allows compiling new versions of JavaScript and targeting legacy platforms.

5.1. **Future Research.**

(1) The idea presented here can be taken further and support can be implemented for all the features added by recent versions of JavaScript.
(2) A block of code with a very small footprint can be used to decide at runtime whether to load the original JavaScript source or the translated one. This would eliminate any small speed-bumps which the translator may add as long as the user is using an environment which supports the complete set of JavaScript instructions while still supporting everyone with an outdated environment.
(3) Additionally, the Abstract Syntax Tree generated by the parser can be used in order to perform optimizations.
(4) The output module can be replaced with one which generates C++ code or LLVM intructions.

References

[1] Apache configuration for v8cgi. http://code.google.com/p/v8cgi/wiki/ApacheConfiguration, last updated May 12, 2010.

[2] Google Caja, a source-to-source translator for securing Javascript-based web content. http://code.google.com/p/google-caja/, retrieved Jun 9, 2010.

[3] Overview of the Caja system. http://code.google.com/p/google-caja/wiki/CajaOverview#Cajita, last updated Jul 17, 2009.

[4] Douglas Crockford. JavaScript: The world's most misunderstood programming language, 2001. http://www.crockford.com/javascript/javascript.html.

[5] Douglas Crockford. JavaScript: The Good Parts. O'Reilly, May 2008.

[6] Ecma International. ECMA-262, 5 edition, December 2009. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf

[7] Eli Grey. Pausing JavaScript with async.js. Eli Grey's Blog, January 2010. http://eligrey.com/blog/post/pausing-javascript-with-async-js.

[8] Neil Mix. Threading in JavaScript 1.7. Neil Mix's blog, January 2007. http://www.neilmix.com/2007/02/07/threading-in-javascript-17/.

[9] Mozilla Developer Center. New in JavaScript 1.7, 2006. https://developer.mozilla.org/en/New_in_JavaScript_1.7 (revision 144).

[10] Mozilla Developer Center. Iterators and Generators, 2007. https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Iterators_and_Generators (revision 13).

Master student, Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: `bluepx@gmail.com`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

COMPUTER VISION AIDED MEASUREMENT OF MORPHOLOGICAL FEATURES IN MEDICAL OPTICS

BOGDANA BOLOGA AND ADRIAN SERGIU DARABANT

Abstract. This paper presents a computer vision aided method for non invasive interupupillary(IPD) distance measurement. IPD is a morphological feature requirement in any oftalmological frame prescription. A good frame prescription is highly dependent nowadays on accurate IPD estimation in order for the lenses to be eye strain free. The idea is to replace the ruler or the pupilometer with a more accurate method while keeping the patient eye free from any moving or gaze restrictions. The method proposed in this paper uses a video camera and a punctual light source in order to determine the IPD with under millimeter error. The results are compared against standard eye and object detection routines from literature.

1. Introduction

Research in this domain has a powerful motivation because of its significant contribution to the real world. It has a major importance to industry, community (helping the visually impaired) and, one of the most prominent application fields, medical computer vision or medical image processing [1].

Object Recognition raises many problems, the most important being that human visual recognition is not yet fully understood [1] and this makes it difficult to provide exact mathematical and algorithmic descriptions of the processes undertaken by the human brain. Human vision is invariant to the rotation, size or position of the objects and its attention is guided by an early processing of the image. These are problems that recognition systems, including the one presented in this paper, attempt to solve.

Received by the editors: June 28, 2010.

2010 *Mathematics Subject Classification.* 68T45, 68U10.

1998 *CR Categories and Descriptors.* I.4.7 [**Computing Methodologies**]: IMAGE PROCESSING AND COMPUTER VISION – *Feature Measurement*; I.4.6 [**Computing Methodologies**]: IMAGE PROCESSING AND COMPUTER VISION – *Segmentation.*

Key words and phrases. computer vision, object recognition, stereo vision, medical applications.

This work is supported by the Romanian Ministry of Education in the frame of PN2 ID-550/2007.

Object Recognition is dependent on the process that captures the visual image as well as on the preprocessing of the input. A typical vision system is composed of several modules before the actual Object Recognition step [1, 2]: visual image capturing, noise reduction and enhancement of details and dividing the image into regions of interest (segmentation). The results of all the intermediary stages are essential to the correctness and robustness of the recognition algorithms. Post-processing algorithms can be afterwards used to obtain higher localization accuracy.

This paper proposes a non invasive method for precise measurement of the interpupillary distance (IPD). In order to achieve this, the system uses a camera, light reflexions in the center of the pupils and an auxiliary object containing markers for precise eye localization and pixel to millimeters conversion. The high accuracy of the detection as well as the low computational cost are essential to the project.

The recognition system described in this paper is divided into three main parts: the first handles capturing the video stream and the calibration of the camera(s); the second deals with the preprocessing of the frames in order to obtain the final image where the detection algorithm is applied, and the last part computes the ophthalmological measurements. The detection stage is divided into: precise circle center detection (for the markers) and precise pupil center detection.

2. Prerequisites

For pupil recognition the system uses a light source directed towards the face of the patient. This light is reflected in the center of each pupil (Figure 1) allowing precise localization.

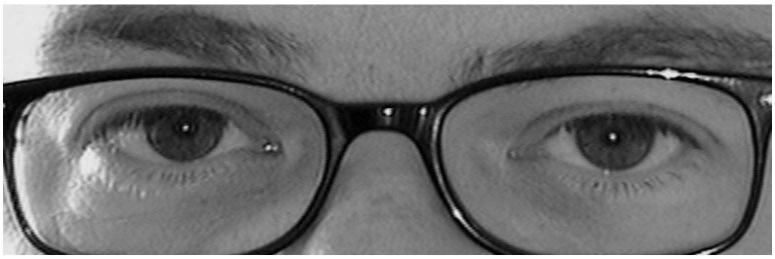


Figure 1. Illuminated center of pupils

In order to convert between the pixel and the metric system we use an auxiliary object with markers. The object has three white circles with a black dot inside, as shown in Figure 2. The markers are well defined shapes with high contrast colors in order to simplify detection.

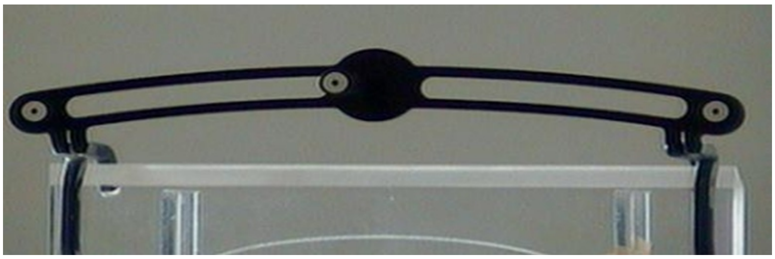


Figure 2. Auxiliary object with markers

In terms of illumination, the recognition system is designed to work under changeable visual illumination conditions. The problem of multiple light sources and multiple reflections around the eye zone is addressed as presented in Figure 3. The algorithm performs a selection of the pupil candidates. However, in some cases, these can interfere with the accuracy of the method.

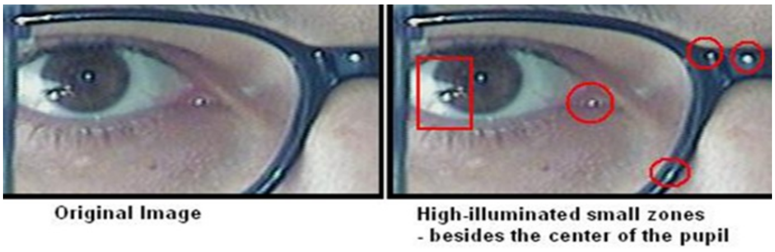

Original Image
High-illuminated small zones
- besides the center of the pupil

Figure 3. Multiple reflections issue

Another important hardware aspect is the capturing device. Higher resolutions can significantly increase the detection accuracy while smaller resolutions improve the speed of the process. The algorithm is designed to work on changeable resolution values.

3. Marker Detection (Circle Detection)

The detection of well defined shapes, such as circles and ellipses, can be achieved by using several standard algorithms. The following approaches have been researched in order to test for best accuracy and robustness: Hough Circle detection [1, 2, 4] and an Ellipse Fitting method [4, 8]. In both cases, the processing time was fitted for real time processing. However, under normal testing conditions, the number of omissions and false positives was high (around 25%-50%) and the precision of the localization was not satisfactory.

The results of the Hough Circles Detection algorithm massively differed with respect to the input parameters and image parameters. The Ellipse Fitting method has proven to be relatively more stable and was incorporated as part of the detection system. The preprocessing stage is composed of:

- Image enhancement algorithms: opening and closing - morphological filters used for eliminating the noise;
- Canny edge detection - the contours are given as input to the Ellipse Fitting method
- Image enhancement: closing filter - in order to create more connected contours (Figure 4)

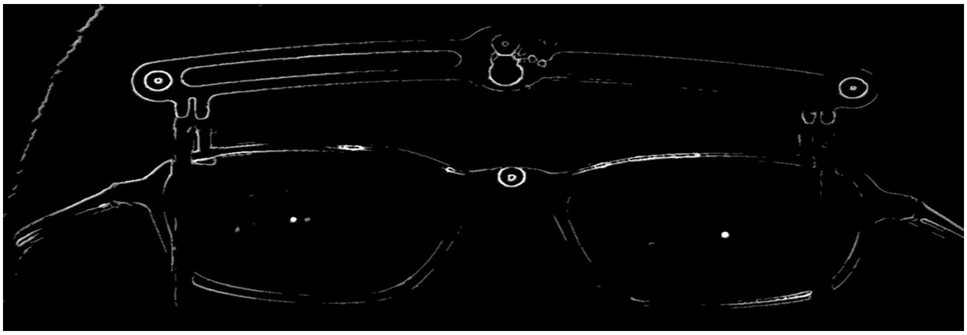


FIGURE 4. Final preprocessing step: Canny Edge Detection and Closing

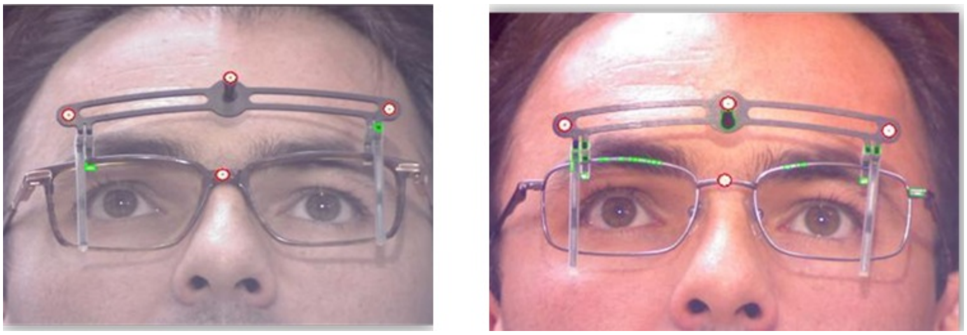


FIGURE 5. Ellipse Fitting results

After applying this technique we have the approximate location of some of the circles (it is not necessary that all markers are found) and certain false positives (Figure 5). Further analysis is needed in order to achieve better results. First, the false positives are eliminated by several methods:

- The preponderant values of the R, G and B channels inside the circles must be approximatively equal (shades of gray). An intensity threshold is not used so as not to impose any restraints on the luminosity of the scene.
- Color segmentation of the surrounding zone is performed in order to obtain the real object's dimensions. They must not highly differ from the candidate circle dimensions (in certain cases ellipsoid patters appear after the edge detection step).
- False positives are detected by comparing the circles' areas with the median area value.

In case at this point we have not selected three or more valid marker zones we apply a Fast Template Matching algorithm. The template image passed on to the method is chosen from the previously found markers in order to enhance localization. False positive elimination is performed after this step as well.

A final adjustment is necessary to increase precision, as illustrated in Figure 6. We position each of the found markers inside the center of the black dot:

- The marker image zone is binarized using and adaptive threshold
- Color segmentation is used to detect the black zone in the center of the white circle
- The enclosing square of the black circle is computed; the center of the circle is considered to be the center of the square; for higher precision, the result of this step is returned as a floating point number.

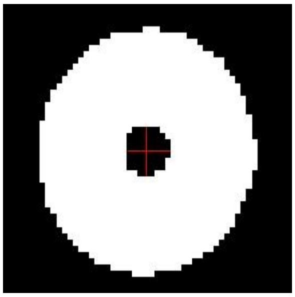


FIGURE 6. Center detection on binarized image of circle

Knowing the distance between the left and the right circle in pixels and dividing it to the real distance in millimeters (110mm in our case) gives us the ratio pixel to millimeters of the image.

4. Pupil Detection

There is much prior work on detecting and localizing facial features [1, 4], however most of these methods approach a general solution with approximative results while we are interested in extremely precise localization.

In this paper we present a fast and robust method for precise eye localization starting from the implementation of the Paul-Viola algorithm [10] for object detection using a boosted cascade of Haar-like features [2, 4]. The results of this method applied with an eye trained classifier are a sequence of rectangles representing the candidate eye zones. This includes many false positives and redundant information.

In terms of execution time, the algorithm performs well on low resolution images but it proved not to be suitable for a real time usage, especially when working with high resolution images. Figure 7 illustrates the results of the standard method compared to the final results.

Our detection system makes use of the Haar-like features in an efficient way that allows real-time processing. The source image is down sampled several times until we reach a minimum predefined resolution. The eye detection technique is applied starting from the smallest resolution image to the highest resolution image until two valid eye zones are selected. At each step the following selection methods are applied:

- Eye zones are separated into left and right eye candidates by taking as vertical axis the mean value of the centers of all detected rectangle zones. The vertical axis is dynamically chosen so as not to impose any restraints on the position of the face in the image.
- False positives are eliminated by comparing the eye zone areas with the median area value. We do not impose a predefined eye dimension.
- All eye pairs of the remaining candidates are computed and inadequate pairs are eliminated based on angle and distance measurements. The minimum angle and distance parameters are highly tolerant. However it is presumed that two eye zones cannot considerably overlap or have an angle larger than 60 degrees.

Generally the algorithm detects two valid eye zones after the first or the second iteration. The processed image is 4 times to 8 times smaller than the original which considerably reduces the execution time.

After choosing the left and right eye pair, further processing is needed in order to achieve high precision. The bright reflections in the pupils are searched inside the eye rectangles. The existence of multiple reflections in the eyes and eye glasses frames has a negative effect on the reflection candidates. This problem is addressed by taking into consideration all bright spots found in the surrounding zone and choosing the best fit pair.

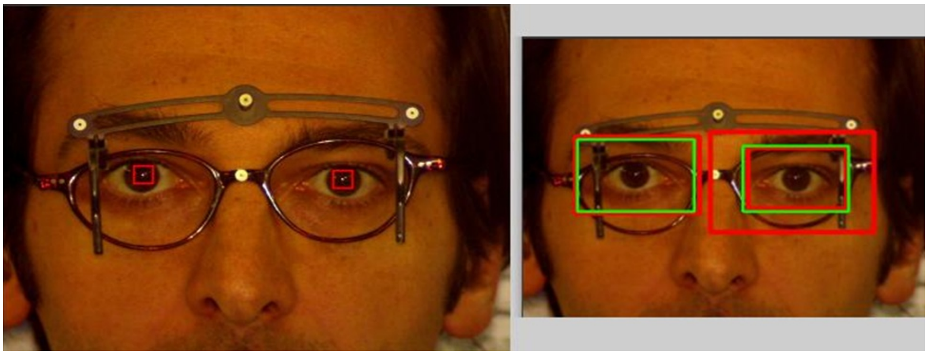


FIGURE 7. Left: Final post-processing results; Right: Initial result of detection;

The detection of the *bright pupil effect* has been used for eye detection and tracking [11, 12]. This effect takes place in near infrared illumination and the result is similar to the reflections in our images. This type of approach uses a set of two images with brightened respectively dark pupils. The difference image is used in order to obtain the reflections. Haro et al. [12] captured the two images for each frame using a system of two lightning sources. Zhao and Grigat [11] captured only one image per frame containing the brightened pupils and removed the bright spots by applying morphological opening operation [1], yet they applied another more complex algorithm for some special cases (where the pupils were not bright enough).

We use a similar technique for detecting the bright spots without the use of infrared lightning. By performing opening operation on the Red channel (where we observed that white light is less visible) we obtain the dark image. The difference between this image and one of the other color channels (where the white light is more visible) gives us the illuminated spots (Figure 8). Histogram equalization [1] is previously applied on both channels and allows us to correctly localize the reflection even when it is not clearly visible (as shown in Figure 8 and Figure 9).

Choosing the best pupil pair is done by heuristically selecting the brightest points closest to the center of the eye rectangle. Shape and intensity resemblance is also taken into consideration.

5. Experiments

In order to evaluate the presented detection system we collected two test sets, shown in Table 1. The images in the first set come from a normal usage and are obtained from different subjects under different conditions. The

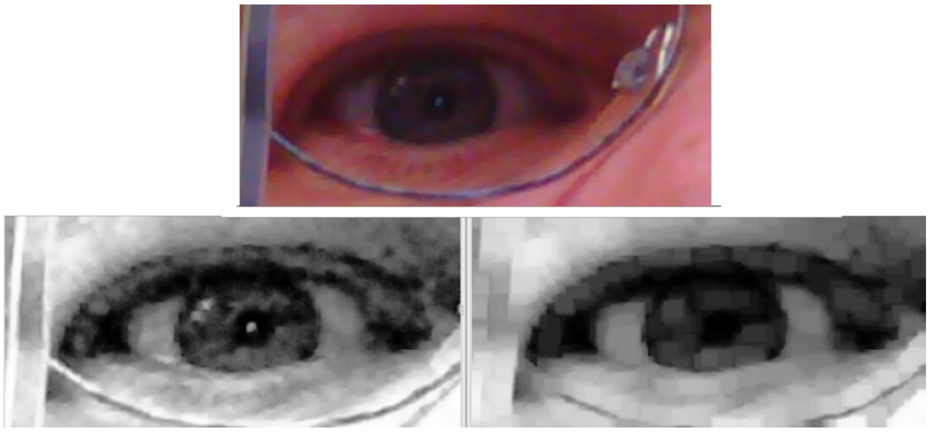


Figure 8. Original image with corresponding bright and darkened channels

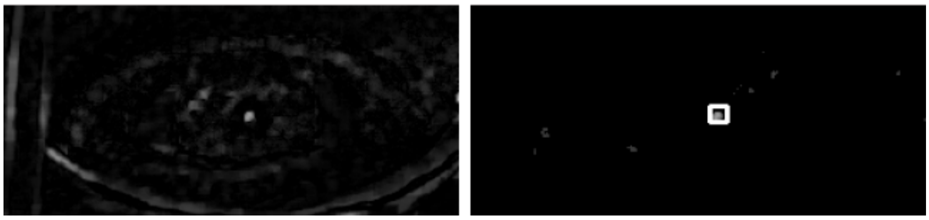


Figure 9. Left: Channels difference; Right: Final selection

Test Set	**Size**	**Contents**
Set 1	159	1600x1200 Images from normal usage
Set 2	30	Various Resolutions - special cases (low illumination, multiple light sources, etc.)

Table 1. Experimental data-photos taken on patients.

images were taken using a high resolution camera. The distance and the angle between the subjects and the camera differ slightly.

The second data set contains images specially captured in highly changing conditions. Multiple light sources were tested as well as low contrast images. The distance between the subjects and the cameras differ greatly as well as the angle of the camera and the rotation of the face.

Test Set	**MARKER Localization**	**Hit Rate (%)**	**Mean error (pixels)**
Set 1 (159 images)	Ellipse Fitting	74.2%	23.25
	EyeIPD	94%	6.2
Set 2 (30 images)	Ellipse Fitting	61.5%	32.4
	EyeIPD	94%	3.12

Table 2. Hit rate - markers detected and pixels mean error estimation.

Test Set	**EYE Localization**	**Hit Rate (%)**	**Mean error (pixels)**
Set 1 (159 images)	Haar Features	85%	24.4
	EyeIPD	94%	3
Set 2 (30 images)	Haar Features	93%	26.5
	EyeIPD	96%	2.8

Table 3. Hit rate - eyes detected and pixels mean error estimation.

We compared the results of the *ellipse fitting method*, *Haar detection* and the *detection algorithm (EyeIPD)* presented here with the manually chosen positions. For each image the manual locations of the eyes and markers were selected in a specially created application. These values were considered to have an accuracy of 100% (a 0 pixel error). Table 2 and 3 show the hit factor (percent of correctly detected markers/eyes) and the average measured error in pixels.

Considering that the average pixel to millimeter ratio is around 9 pixels/mm we get an 4-8mm error in computing the IPD using *Ellipse Fitting* and *Haar eye detection.* Using the *EyeIPD* method the maximum error is around 0.73mm. This yields an improvement in accuracy of four to eight times.

6. Conclusions

In this paper we propose a detection system that can be used for precise measurements in Medical Optics. The algorithm is designed and optimized in order to allow working on video capture and multiple resolution images. The method is flexible with respect to changes in illumination and contrast, camera angle, distance between the face and the camera and face rotations.

The recognition system developed does eye detection and localization: the position of the center of the eye is detected with an accuracy of more than

98% in more than 94% of the testing images. The location of the center of the eyes is then used for computing the exact interpupillary distance of the patient. For the support localization, the markers are circles that simplify detection and are attached to the eye-glasses for providing more information about the real distances. The three support circles centers were detected with an accuracy of over 99% in more than 90% of the images.

References

References

[1] R. Gonzales, R. Woods, *Digital Image Processing*, New Jersey: Pearson Education, Inc., 2008.
[2] M. Bennamoun, G. J. Mamic, *Object Recognition Fundamentals and Case Studies*, Trownbridge : Springer, 2002.
[3] I. Pitas, *Digital Image Processing Algorithms*, New York : John Wiley & Sons, Inc., 2000.
[4] G. Bradski, A. Kaehler, *Learning OpenCV*, s.1. : O'Reilly Media, 2008.
[5] A. Fitzgibbon, M. Pilu, R. B. Fisher, Direct Least Square Fitting of Ellipses, *Pattern Analysis and Machine Intelligence*, 1999, Vol. 21,5.
[6] Wu Chenyu et al., *Automatic Eyeglasses Removal from Face Images*, Melbourne : s.n., 2002, Vol. The 5th Asian Conference on Computer Vision.
[7] Hough Transform, *Planet Math* [Online], http://planetmath.org/encyclopedia/HoughTransform.html.
[8] R. Halir, J. Flusser, *Numerically Stable Direct Least Squares Fitting of Ellipses*. s.1 : Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic.
[9] X. Jiang et al., *Towards Detection of Glasses in Facial Images*, 1998.
[10] P. Viola, M. Jones, *Rapid Object Detection Using a Boosted Cascade of Simple Features*, Mitsubishi Electric Research Labs, 2001.
[11] S. Zhao, R. Grigat, *Robust Eye Detection for Cockpit Access Control Using an Appearance Model and Radial Symmetry Transform Under Active Near Infrared Illumination*, Hamburg University of Technology, Germany, 2007
[12] A. Haro, M. Flickner, I. Essa, *Detecting and Tracking Eyes by Using their Physiological Properties, Dynamics and Appearance*, in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2000.

Babes Bolyai University, Faculty of Mathematics and Computer Science, Cluj Napoca, Romania
E-mail address: bbsd0068@scs.ubbcluj.ro, dadi@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

A COMPUTER VISION APPROACH TO OBJECT TRACKING AND COUNTING

SERGIU MEZEI AND ADRIAN SERGIU DARABANT

Abstract. This paper, introduces a new method for counting people or more generally objects that enter or exit a certain area/building or perimeter. We propose an algorithm (method) that analyzes a video sequence, detects moving objects and their moving direction and filters them according to some criteria (ex only humans). As result one obtains *in* and *out* counters for objects passing the defined perimeter. Automatic object counting is a growing size application in many industry/commerce areas. Counting can be used in statistical analysis and optimal activity scheduling methods. One of the main applications is the approximation of the number of persons passing trough, or reaching a certain area: airports (customs), shopping centers and malls and sports or cultural activities with high attendance. The main purpose is to offer an accurate estimation while still keeping the anonymity of the visitors.

1. Introduction

Researches in video processing and implicitly image processing have been very active in the last years and the result is the abundance of new techniques that have been successfully adopted in various branches of science, industry, medicine and security systems. High performance computers have become widely available at relatively low costs and this makes it possible for many businesses to track the number of people that walk over their door step using a simple camera and a PC. Counting people gives retailers knowledge about their visitors. How they can be manipulated by advertisements, weather, time of day etc. This is extremely valuable and helpful information which can be used to, for example, plan staffing and optimum times for cleaning and maintenance,

Received by the editors: June 28, 2010.

2010 *Mathematics Subject Classification.* 68T45, 68U10.

1998 *CR Categories and Descriptors.* I.5.4 [**Computing Methodologies**]: PATTERN RECOGNITION – *Applications*; I.4.6 [**Computing Methodologies**]: IMAGE PROCESSING AND COMPUTER VISION – *Segmentation.*

Key words and phrases. computer vision, object recognition, blob tracking,counting.

This work is supported by the Romanian Ministry of Education in the frame of PN2 ID-550/2007.

determine opening hours and preferred advertisements, and so forth. The application itself is based on analyzing video streams from cameras in real time or from pre-recorded video files. Earlier people counting methods are the laser beam method which is basically a counter that increments the number of people every time the beam is interrupted; the ultrasound method, which sends an ultrasound beam that bounces back of the obstacles and determines the distance from the source to them in order to determine if the people have entered a certain area or not. These methods, although they are still used in many department stores, are highly inexact in their calculations because of the number of situations in which two persons enter the store next to each other or the fact that by using the above mentioned methods you can count only the total number of people and not the number of *ins* and *outs*. The main advantages of the method proposed in this paper, over a regular counting system are the following:

- Bidirectional counting
- Flexibility of the algorithm when unexpected situations occur
- Surveillance of large entrances or large spaces
- Surveillance of high traffic areas
- Easy integration with databases or online systems
- Easy installation process that does not restrict access to the store or building

The application is structured in five main parts: the first handles the capturing of the video (reading the video stream); the second represents the processing of the individual frames in order to obtain the binary image that contains the people blobs (binary large objects); the third part handles the blob detection and tracking part; the fourth part is the actual algorithm that counts the number of people that enter or exit and the last part, the fifth, handles the actual output of the application and the data being written on the video stream.

2. Object Tracking and Counting

This section presents the methods we used to reach the final results as well as the reasons for which they are used. They are divided into three main categories:

- Image Enhancement
- Blob Detection
- Blob Tracking

All algorithms are presented using sample images that illustrate the results. Figure 1(b) shows an example of an original frame from a video. The frame shows the normal placement of the camera - usually placed on ceilings and oriented towards the floor in order to allow object detection and to limit the field of vision such that identification of the visitors could not be possible. The counting approach is based in detecting moving objects in the camera field of vision, identify only human profiles and count them.

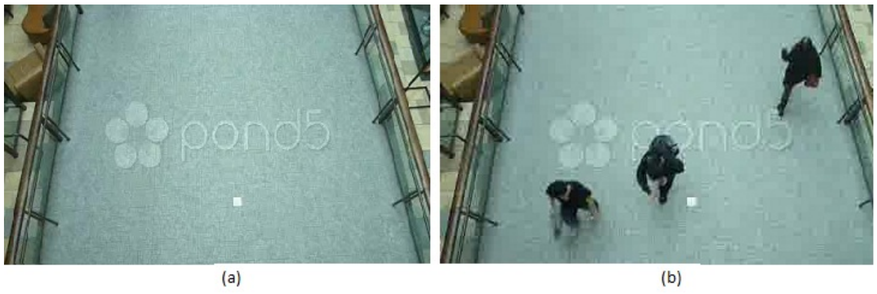


FIGURE 1. Original video frame from the input video.

The first problem that has to be faced here is to correctly identify only human profiles out of all items passing in front of the cameras using some kind of blob detection algorithm. The second problem to be dealt with is to correctly track the detected blobs so that we can keep trajectory information from one frame to the next and record their direction of movement. This would allow the identification and counting of objects *entering* and *getting out* of the controlled area. All these should be handled in an environment where lighting conditions are not always ideal. As the detection/measurement/counting process is usually continuous, the method should take care of variations in lighting during daylight to artificial lighting. Finally, often people tend to arrive in groups when interest area is highly crowded, thus partially covering each other even for a bird's eye view camera perspective. The following paragraphs describe the operations carried on the video flow, frame by frame in order to solve the counting problem.

2.1. **Image Enhancement Algorithms.** These algorithms represent the first step in the detection process.

2.1.1. *Background subtraction.* In order to obtain the foreground objects (in our case the people that enter the area) we need to do a background subtraction which means to subtract two images [6]. Figure 1(b) shows a video frame with people walking in the camera field of view, while 1(a) shows a frame with only static background. This method can be used when we are certain that we can get the first frame on the video to contain the static background and use that frame as a base frame and compare the real video frames against this base frame. It is however improper to assume that static background frames with the same lighting conditions as for day, night and in-between times of the day could be obtained. Light is different according to the seasons and artificial lighting conditions, so a more complex approach is needed. The solution should not need to stop the flow of people who enter the measurement area. The adopted idea is to construct an image object that will contain an accumulation of frames over time which will represent a mean of those accumulated images. This *accumulator image* is subtracted from the current frame and thus we obtain the foreground objects. Figure 2 represents the foreground objects obtained from the above explained method.

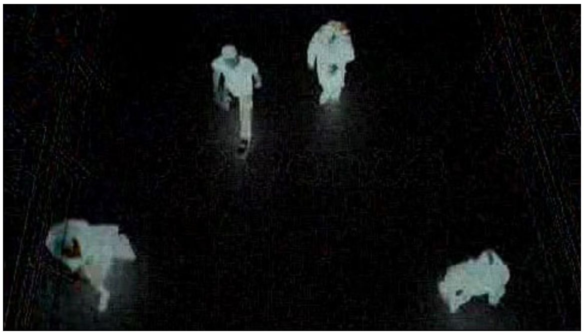


FIGURE 2. Foreground objects after subtraction.

The image still contains a lot of noise requiring thus further processing. The subtracted resulting frame gives a pretty good approximation of the informations we are looking for. Noise should be eliminated in order to avoid its interference with the blob detection algorithm.

2.1.2. *Gray-scaling.* In order to convert any color to its approximate level of gray, one must obtain the values of its red, green and blue (RGB) primaries, in linear intensity encoding. Then, a percentage of these values are added, resulting the desired gray value between 0 and 255 (0 is black and 255 is white). These percentages are chosen due to the different relative sensitivity of the normal human eye to each of the primary colors (less sensitive to blue, more to green). More than color conversion, a gray level image only contains a lot less data to be processed -one byte per pixel as opposed to four bytes per pixel for true color images. One-channel color images are easier to be dealt with in computations. They also contain all the information required to solve the problem and any not needed data is already discarded.

2.1.3. *Binary Thresholding (Binarization).* The next steps for optimizing blobs detection is to threshold the image and transform it into a binary one. A binary image still contains all needed information four our algorithm while further reducing data. Binarization is performed on gray images by applying a threshold and splitting the pixels into only black or white according to their value compared to the threshold. A good binarization is an essential and difficult step. One alternative is to choose a default threshold value that would work well for most videos. Statistical studies on the videos and their frames revealed that the value of 70 would be sufficient for most images. But there are special cases where the brightness highly differs from the average value such that no default threshold could be used. Leaving the task of choosing the right value to the user is not convenient in this case, as the high level of automation for the application would be damaged. Basically, after binarization we get a frame that only has two colors: black and white. The gray disappears because after binarization, depending on the thresholding value (in our case 70), everything that is under that value becomes black and everything over that value becomes white. Figure 3(a) represents the frame from Figure2 after thresholding is applied. At a closer look one can still observe some minor noise in the image.

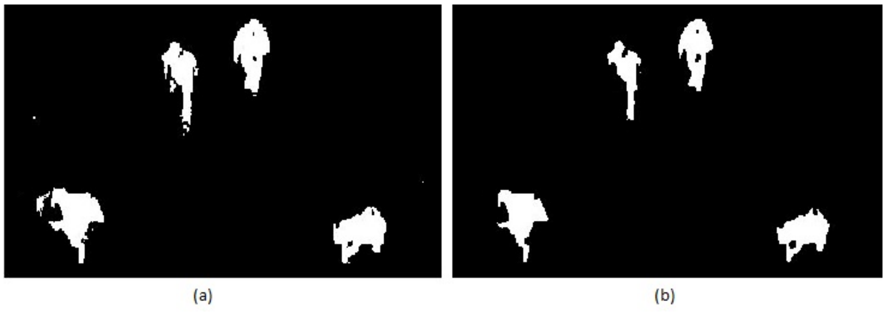


FIGURE 3. Binary image(a), Eroded image(b).

2.1.4. *Erosion and Dilation.* Erosion is used in order to remove the noise in images and make the people blobs more *exact*. It applies a kernel to the image that erodes the borders of the connected components in a binary image based on the values from the neighborhood. Erosion is an iterative method, that can be applied multiple times on an image. Figure 3(b) describes the frame from Figure 3(a) on which erosion was applied. Erosion is applied in order to separate lightly connected blobs that are probably part of different blobs connected by noise or some other overlap or link situations. Dilation applies a kernel on the image in order to correct irregular pixel-based shapes. The dialtion is necessary because we want to fill the black regions within the blobs and make them contiguous. Like erosion, dilation is also an iterative method, so it can be applied as many times as the user specifies. Figure 4 represents the frame from Figure 3(b) on which dilation was applied.

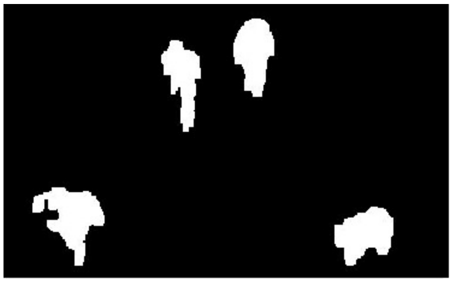


FIGURE 4. Dilated image-blobs are well separated and compact.

2.2. **Blob Detection.** Blob detection (blob analysis) involves examining the blob image and finding each individual foreground object [6, 3, 4]. This may in some special cases become a rather difficult task, even for the human eye. Figure 5(a) represents two blobs, which are easily distinguished from one another. When the moving persons have similar colors as the background, the blobs become rather distorted [1]. Some abnormal situations appear when blobs contain holes as shown in figure 5(b). Dilation was not sufficient in this case to completely compact the blob. Blobs are found by finding all adjacent pixels and putting them together so that the pixels form the blob object [1]. Analyzing contiguous blobs is a straightforward process. Fragmented blobs, like the one shown in figure 5(c), are harder to analyze since it is difficult to see if the blob is in fact a blob composed from multiple people. It becomes impossible to detect that in crowded areas. Still, large area blobs are likely to be groups of either people or something else that is moving. People standing in groups are hard to deal with, even for the human eye. When they stand close together or hold hands they form one big blob, like the one shown in Figure 6. It is hard to determine how many blobs there actually are in the image.

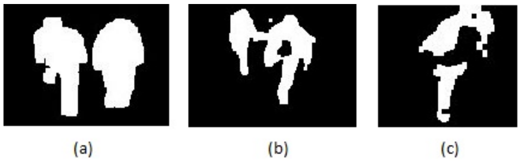


FIGURE 5. Normal blobs(a), Blobs with holes(b), Fragmented blobs(c).

FIGURE 6. Multiple people form a larger blob.

It may seem relatively straightforward to divide the larger blobs into smaller blobs by splitting the blobs with respect to height and width. This is not as easy, especially in an environment where not only people pass in the front of the camera (ex: in a mall or shop). People coming from grocery stores or other similar shops often push trolleys ahead of them. These trolleys usually appear to be of the same size as a human being. This makes splitting blobs by height or width a hard task, which must take movement into account. Two blobs moving horizontally appear larger in height, while moving vertically has larger width. In the current approach these are not dealt with yet. Blob analysis is all about identifying the blobs, grouping together adjacent pixels and representing them using a good data structure [5]. The size of the blob is not the issue. Even though the blob consists of numerous people it will still be represented as one blob. This can give rise to other problems such as with the fragmented blobs which will make one blob appear as many [7]. If the background subtraction is configured correctly, it should be relatively rare that a blob will be very fragmented. Therefore, all individual blobs, consisting of more persons will be represented as one blob in an appropriate data structure. Simple properties of a blob can be its position, width and height. A good way to combine position with width and height is to keep information about the coordinates of the smallest X and Y value, as well as the largest X and Y values [8]. In this way, one can easily know the position of the blob. It can find the width and height by subtracting the smallest X coordinate from the largest and similarly for the Y coordinates. The limitation of this approach is that the blob's shape will be known only by a rectangular approximation of the area surrounding it. This is shown in Figure 7. The point of origin for the blob is the smallest (X,Y) coordinate.

When blobs can easily be recognized one needs some sort of a data structure to hold all of the blob instances [9]. Trying to organize them so that one will quickly match the blobs from older frames to newer ones is a good idea. To do this in a fast and effective way one could use some advanced structure like a skip list to categorize them by, for example area or direction. The overhead of this method and the varying sizes of blobs on each location

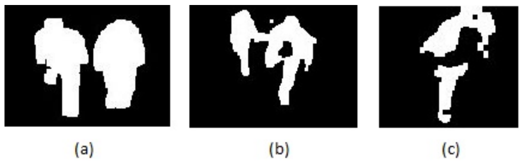


FIGURE 7. Bounding rectangle on a detected person in the video.

make this an ineffective way to compare blobs quickly. The fastest way to process the blobs is to simply put them into a list as they appear. This means actually categorizing them by their X and Y coordinates since the blob image is systematically processed starting at coordinates (0,0). We use an easy manageable list so that blobs could be quickly added, merged and deleted. Since the amount of blobs varies with time a dynamic list is required. With these reasons in mind, we choose a linked list in order to keep track of all the blobs.

2.3. **Blob Tracking and Counting.** The last step in order to count persons moving underneath the camera is to see whether a blob passes over a previously defined *boundary* that is basically a one pixel wide horizontal line across the middle of the video frame. The idea is to count a person who travels from the top towards the bottom of the frame as *IN* if and only if it passes over that line and count a person who travels from the bottom towards the top as *OUT*, again, only if it passes over that line. Some blobs only move under the camera and never enter or exit the store. These are in most cases not counted. The application also retains the track described by a blob in order to identify each unique blob across successive frames. A frame with blobs in a previous frame will indicate the blobs movement [6, 4]. If we were to count people going through a large entrance at the top of the image, knowing the path of the blobs will show that the middle blob can be counted as entering, the rightmost blob as exiting and the leftmost blob should not be counted. Analyzing the path of the blobs is the goal of blob tracking [2]. Blob tracking was performed with the help of the cvBlob library [10] which labels detected blobs in order to help tracking them from one frame to another. Figure 8 shows the algorithm applied on a video sequence.

3. Results and Conclusions

We used two video sets in order to statistically determine some of the parameters for the detection process. Then we applied heuristics for improving these parameters on video sequences of a few hours. This resulted in an accuracy of over 90% for the blob detection process. In the two initial video sets we detected 12 out of 13 people blobs. From the 20 people blobs that appear

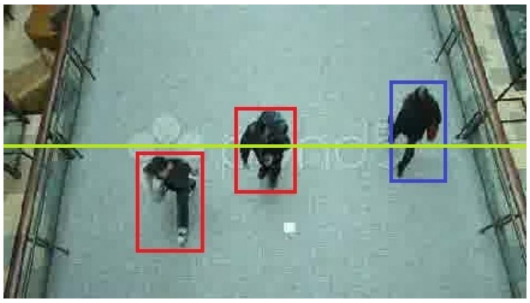


FIGURE 8. People detected as blobs, tracked and counted as *in* and *out*.

in the second video sample, only 1 remains undetected. On the experimental video sequences the counted blobs reached an accuracy of 92%. Out of 3000 people passing only 40 where incorrectly detected - mostly because they passed in very compact groups in front of the camera. The results also showed that segmentation can be the most difficult stage in blob detection because its success depends on the quality and brightness of the image generated by the illumination conditions.

In this paper we propose an algorithm for real-time detection of objects and their moving direction in a camera video sequence. Out method allows for bidirectional counting of moving objects as compared to other exiting methods. For this current version of the application, the blob detection process is done by scanning, in a binary image, pixel by pixel until a white pixel is found. When the white pixel is discovered, we search the area around it in all four directions in order to find other white connected pixels. This method can be very resource intensive. A future version of the application will scan areas instead of pixels. Dividing the frame in a number of areas, like a grid, and perform the search on the grid areas should allow for an important reduction in the CPU workload.

References

[1] Araujo, H., Mendonca, A., M., Pinho, A., J., Torres, M. I.: Pattern Recognition and Image Analysis, Springer, Portugal, 2009

[2] Comaniciu, D., Kanatani, K., Mester R., Suter, D.: Statistical methods in video processing, Springer, Prague, Czech Republic, 2004

[3] Huang, D., Jo, K., H., Lee, H., H., Kang, H., J., Bevilacqua, V.: Emerging Intelligent Computing Technology and Applications, Springer, Ulsan, South Korea, 2009

[4] Lindeberg, T.: Scale-space theory in computer vision, Kluwer Academic Publishers, Netherlands, 1994

[5] Niels, H., Niels da Vitoria L.: Visual event detection, Kluwer Academic Publishers, USA, 2001

[6] Langdon, P., Clarkson, J., Robinson, P.: Designing Inclusive Futures, Springer, London, UK, 2008
[7] Louban, R.: Image processing of edge and surface defects, Springer, Berlin, Germany, 2009
[8] Stiefelhagen, R., Garofolo, J.: Multimodal Technologies for Perception of Humans, Springer, Berlin, Germany, 2006
[9] Sunderam, V., S., Dick van Albada, G., Sloot, P., M., A., Dongarra, J., J.: Computational Science ICCS 2005 Part One, Springer, Atlanta, USA, 2005
[10] http://code.google.com/p/cvblob/

Babes Bolyai University, Faculty of Mathematics and Computer Science, Cluj Napoca, Romania
E-mail address: `dadi@cs.ubbcluj.ro`