9th SC@RUG 2012 proceedings

Smedinga, Rein; Biehl, Michael; Kramer, Femke

university of groningen / faculty of mathematics and natural sciences / computing science

SC@RUG 2012 proceedings

9th SC@RUG 2011-2012

Rein Smedinga, Michael Biehl
en Femke Kramer (editors)

www.rug.nl/informatica

SC@RUG 2012 proceedings

Rein Smedinga
Michael Biehl
Femke Kramer
editors

2012
Groningen

ISBN: 978-90-367-5631-0
Publisher: Bibliotheek der R.U.
Title: Proceedings 9th Student Colloquium 2011-2012
Computing Science, University of Groningen
NUR-code: 980

About SC@RUG 2012

Introduction

SC@RUG (or student colloquium in full) is a course that master students in computing science follow in the first year of their master study at the University of Groningen.

In the academic year 2011-2012 SC@RUG was organized as a conference for the ninth time. Students wrote a paper, participated in the review process, gave a presentation and were session chair during the conference.

The organizers Rein Smedinga, Michael Biehl and Femke Kramer would like to thank all colleagues, who cooperated in this SC@RUG by collecting sets of papers to be used by the students and by being an expert reviewer during the review process. They also would like to thank Janneke Geertsema for her workshops on presentation techniques and speech therapy.

Organizational matters

SC@RUG 2012 was organized as follows. Students were expected to work in teams of two. The student teams could choose between different sets of papers, that were made available through *Nestor*, the digital learning environment of the university. Each set of papers consisted of about three papers about the same subject (within Computing Science). Some sets of papers contained conflicting opinions. Students were instructed to write a survey paper about this subject including the different approaches in the given papers. The paper should compare the theory in each of the papers in the set and include own conclusions about the subject.
Five teams proposed their own subject.

After submission their papers, each student was assigned one paper to review using a standard review form. The staff member who had provided the set of papers was also asked to fill in such a form. Thus, each paper was reviewed three times (twice by peer reviewers and once by the expert reviewer). Each review form was made available to the authors of the paper through *Nestor*.

All papers could be rewritten and resubmitted, independent of the conclusions from the review. After resubmission each reviewer was asked to re-review the same paper and to conclude whether the paper had improved. Rereviewers could accept or reject a paper. All accepted papers can be found in these proceedings.

All students were asked to present their paper at the conference and act as a chair and discussion leader during one of the other presentations. Half of the participants were asked to organize one day of the conference (i.e., to make the time tables, invite people, look for sponsoring, etc.) The audience graded both the presentation and the chairing and leading the discussion.

In her lectures about communication in science, Femke Kramer explained how conferences work, how researchers review each other's papers, and how they communicate their findings with a compelling storyline and cleverly designed images.

Michael Biehl taught a workshop on writing a scientific paper and Janneke Geertsema gave workshops on presentation techniques and speech therapy that were very well appreciated by the participants.

Rein Smedinga did the overall coordination, administration and served as the main manager of *Nestor*.

Janneke Geertsema provided workshops on improving presentation skills.

Students were graded on the writing process, the review process and on the presentation. Writing and rewriting counted for 50% (here we used the grades given by the reviewers and the re-reviewers), the review process itself for 15% and the presentation for 35% (including 5% for the grading of being a chair or discussion leader during the conference). For the grading of the presentations we used the judgments from the audience and calculated the average of these.

In this edition of SC@RUG students were videotaped during their presentation using the new video recording facilities of the University and with thanks to the CIT crew and Albèrt Kerkhof and his video crew for handling the equipment. The recordings were published on *Nestor* for self reflection.

On 18 and 20 April 2012, the actual conference took place. Each paper was presented by both authors. On Wednesday 5 presentations were given and on Friday 6, each consisting of a total of 20 minutes for the introduction, the actual presentation and the discussion. On Wednesday Wico Mulder from Logica was the keynote speaker (he spoke about "Bridging academic and industrial worlds" and and on Friday Theodoor De Jonge of Numeriek Centrum Groningen gave a key note lecture on "NUPAS-CADMATIC 3D ship design software".

As mentioned before, each presenter also had to act as a chair and discussion leader for another presentation during that day. The audience was asked to fill in a questionnaire and grade the presentations, the chairing and leading the discussion. Participants not selected as chair were asked to organize the day.

Thanks

We could not have achieved the ambitious goal of this course without the invaluable help of the following expert reviewers:

- Michael Wilkinson,
- Michael Biehl,
- Marco Aiello,
- Tobias Isenberg,
- Jos Roerdink,
- Matthias Galster,
- Wim Hesselink,
- Heerko Groefsema
- Alexander Lazovik.

Also, the organizers would like to thank:

- the *School of Computing and Cognition* for making it possible to publish these proceedings and sponsoring the conference,
- *Numeriek Centrum Groningen* for sponsoring lunch on Friday and
- *Janneke Geertsema* for, again, providing excellent workshops on improving presentation skills.

Rein Smedinga
Michael Biehl
Femke Kramer

Contents

A comparison of particle-based deformable models for image segmentation

Zhe Sun and Steven Bouma

Abstract— The deformable model is an effective method for image segmentation and shape recovery. Recently, three novel physically motivated deformable models have been introduced. These methods include: the charged-particle model (CPM), the charged-contour model (CCM) and the charged fluid method (CFM). All these models are inspired by classical electrodynamics and simulate the physical movement in an electrostatic field for the purpose of image segmentation.
Our research investigates and compares the CPM, CCM and CFM models. The three models are all inspired on different physical phenomena (electrodynamics, fluid dynamics), yet there are many similarities in the different approaches. We investigate whether this theoretical similarity leads to indistinguishable or comparable results in image segmentation. We also discuss the best application scenario of each model and provide a usage guideline, pointing out advantages or disadvantages of the models.
We compare the three models based on: theoretical similarity, usability, accuracy of image segmentation, generalization and algorithm complexity. Besides discussing the various initialization procedures, we also look into the parameter configuration to show which approach uses a robust parameter set that is easily configurable.

Index Terms—Image segmentation, deformable model, charged-particle model (CPM), charged-contour model (CCM), charged fluid method (CFM)

1 Introduction

Segmentation is a process of partitioning the image into some non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous [15]. Applications of image segmentation include: machine vision, biomedical image analysis, pattern recognition and analysis of geographical images. Segmentation is an important step of many computerized analysis procedures, and the segmentation quality can determine the outcome of such a procedure to a large extent.

Image segmentation has proved to be a non-trivial problem that has kept researchers active for many years. Most segmentation methods are rather context specific and a method that is successful in a wide range of applications is yet to be developed [3]. Ideally a method would allow the segmentation of an image into regions with similar properties without requiring user input, however there are usually many ways to segment an image, and domain knowledge is often required to select the correct segmentation. This concept is shown in figure 1, where it is visible that the quality of the segmentation may depend on domain knowledge. In addition there are always elements in the data, such as artifacts and noise, which may interfere with the process of segmentation. Not all methods handle noise equally well in which case boundaries may become disconnected, a concept often described as *boundary leakage*. The challenge is to extract boundary elements belonging to the same structure and integrate these elements into a coherent and consistent model of the structure [3].

Methods of image segmentation can be roughly divided into several categories: pixel-based, edge-based, region-based, matching approaches and deformable models [15, 17, 5]. The deformable models have proven to be quite successful in shape recovery and medical image areas since they were introduced by Kass et al. in 1987 [10]. The most popular deformable model is the snake model. Some disadvantages of the snake model were pointed out in [1, 18]. Many articles introduced ways to improve the performances of the snake model [6, 11, 16, 18]. Among the aforementioned methods, the Gradient Vector Field (GVF) method introduced a new external force which has a large capture range and is able to move the snake into boundary concavities.

Recently, three novel particle-based deformable models were proposed by Jalba et al. [9], Chang et al. [3] and Yang et al. [19]. These methods include: the charged-particle model (CPM), the charged-contour model (CCM) and the charged fluid method (CFM). They are all inspired on classical physical phenomena, i.e., electrodynamic and fluid dynamics, and show great theoretical similarity. Our research compares the three methods and show that this theoretical similarity leads to comparable results.

In section 2 we list two classic deformable models, snake and Gradient Vector Flow (GVF), and discuss the defects of them. Then, we summarize the CPM, CCM and CFM models in section 3. Next, three methods are compared in section 4 from the aspects of parameter sensitivity, particle initialization, computation complexity, weak edge detection, generalization and performance with noisy image. This paper is finalized with a summary of the comparison results and a conclusion in section 5.

2 Related work

The active contour model, also known as the *snake model*, was initially introduced by Kass et al. [10]. The active contour model is a simple deformable model that works well for convex shapes. A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges [18]. A tradition snake is a classic active contour models: a spline $x(s) = [x(s), y(s)]$, $s \in [0,1]$, that moves through the spatial domain of an image to minimize the energy functional

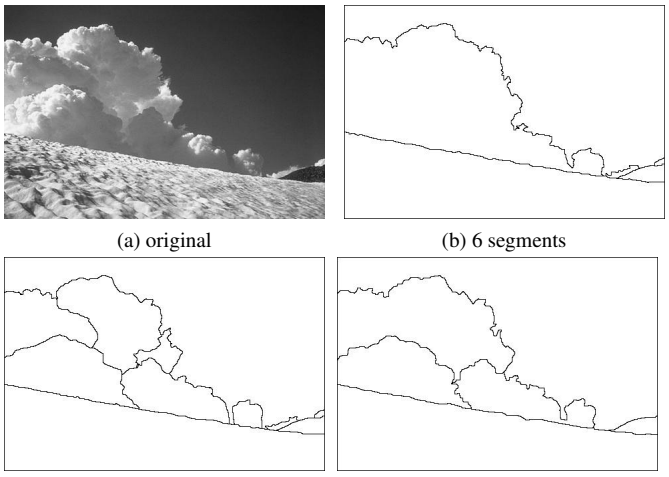
$$E = \int_0^1 \frac{1}{2}[\alpha|x'(s)|^2 + \beta|x''(s)|^2] + E_{ext}(x(s))ds \qquad (1)$$

where α and β are weighting parameters that control the snake's tension and rigidity, respectively, and $x'(s)$ and $x''(s)$ denote the first and second derivatives of $x(s)$ with respect to s. Snakes provide a unified treatment to a collection of visual problems [10], including detection of edges, lines, and subjective contours; motion tracking; and stereo matching.

The Snake method has been improved over time and knows many extensions [13, 14]. Ng et al. [14] have proposed this method in the context of medical image segmentation. McInerney et al. [13] provide a more general overview of deformable models as applied in medical

- *Zhe Sun is a MSc. student at the university of Groningen, E-mail: z.sun.2@student.rug.nl.*
- *Steven Bouma is a MSc. student at the university of Groningen, E-mail: s.bouma@rug.nl.*

SC@RUG 2012



(a) original (b) 6 segments

(c) 11 segments (d) 6 segments

Fig. 1: Multiple ways of segmenting an image as indicated by human subjects. [12]. (a) emphasis the difference between foreground and background, all the clouds are in one segmentation; (b) detailed segmentation of the clouds; (c) roughly segmented clouds

image analysis. However, there are two key difficulties when using the snake method. First, the initialization snake has to lock onto the true boundary, and locate the edge accurately, otherwise it will likely converge to the wrong result. The second problem is that active contours have difficulties progressing into boundary concavities [10].

The GVF snake model proposed by Xu et al [18], which is an extension of the original snake, has been a popular deformable model so far. Like the methods described in this paper, the GVF snake model also uses the concept of a vector-field to direct the deformable model. Xu et al. [18] introduced a new external force model and it is calculated as a diffusion of the gradient vectors of a gray-level or binary edge-map. The E_{ext} term in equation 1 is replaced by the gradient vector flow field (GVF) $v(x,y) = [u(x,y), v(x,y)]$ which minimizes the energy function

$$\varepsilon = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 dxdy. \qquad (2)$$

$f(x,y)$ is the edge map derived from the image $I(x,y)$ having the property that it is larger near the image edge. The parameter μ is a regularization parameter governing the trade-off between the first term and the second term in the integrand [18]. Compared with the classic snake, the GVF snake improves the performance in three aspects: first, the GVF field has a larger capture range and it is not as sensitive to the initial position of the snake as the classical version. Second, contrary to classical snakes, it is able to trace concavities in the boundaries of objects. Third, it reaches the desired shape in far few iterations.

3 PARTICLE-BASED DEFORMABLE MODELS

This section describes three image segmentation methods that are based on particle simulation, being CPM, CCM and CFM respectively. All three methods are inspired by classical physical phenomena. The CPM and CCM methods are both based on the trajectory of electrically charged particles (such as electrons or ions in a plasma) in an electrostatic field. The CFM method also uses the concept of moving particles in a vector field, but the concept of a charged fluid was used to derive the model instead. The methods address several issues regarding initialization and parameter sensitivity, that are encountered in traditional deformable methods such as the active contour model.

The common element of all methods is a vector-field that is initialized from the gradient magnitude of the image that is to be segmented (Fig. 2). Although the particle dynamics for all three methods are different, all methods acquire particle-directing forces from such a vector-field.

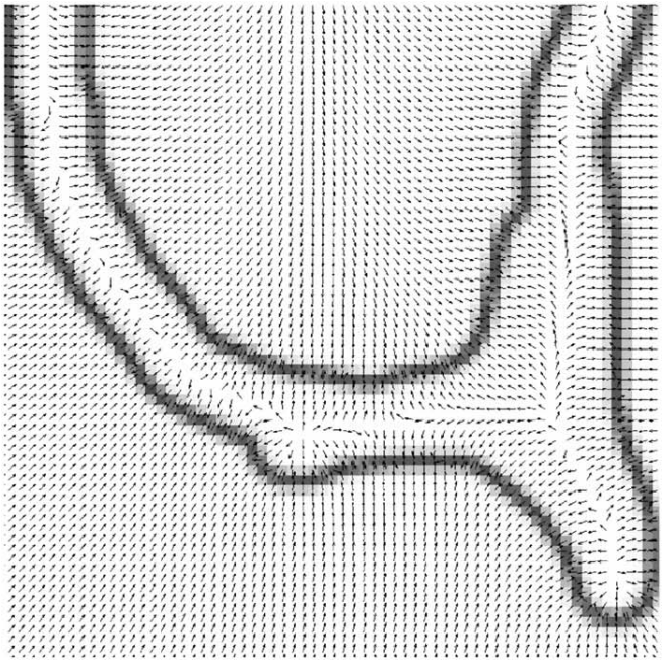


Fig. 2: Vector Field initialized from image gradient magnitude [9].

3.1 Charged Particle Model (CPM)

The charged-particle model proposed by [9] is based on particles that ultimately form the segmentation boundary (deformable model) that are inserted into a vector-field. The particles are updated in multiple iterations in which they move around contours of an image segment forming a distribution along the boundaries of a structure. When the particles have reached and equilibrium (i.e. a stable state) and formed a sufficiently dense distribution, a closed contour is constructed from the particles. The boundary reconstruction can be achieved using a reconstruction algorithm such as proposed by Amenta et al. [1]. This algorithm can be used to obtain a contour (2D) or a surface (3D) from a given set of points, in this case the particles.

One of the most notable capabilities of the CPM is that it works well for highly concave shapes. Such shapes are often encountered in for example biomedical imaging applications. This is opposed to methods such as the active contour model that only work well for convex shapes.

CPM Particle dynamics All particles in the CPM method are affected by two kinds of forces:

- Particle-attracting Lorentz forces. The Lorentz forces attract particles towards the contours of elements in the image.
- Particle-repelling Coulomb forces. The Coulomb force is a repelling force between particles. In the theory each particle affects every other particle.

The total force on particle p_i is the combination of the Coulomb and Lorentz components:

$$F(r_i) = F_{Coulomb}(r_i) + F_{Lorentz}(r_i). \qquad (3)$$

The vector field with particle-attracting Lorentz forces is initialized from the gradient magnitude of an image. In CPM, this is a fixed field for a particular image, that does not change over time. The gradient magnitude computed by $f(x,y) = |\nabla(G_\sigma(x,y) * I(x,y))|$, is used to determine the electric charge $e_k = -f(x,y)$, which has lower values (< 0) in areas of higher change.

The factor e_k is used to determine the Lorentz force in each location of the vector field. The Lorentz forces are computed by:

$$F_{Lorentz}(r_i) = q_i \sum_{k:R_k \neq r_i}^{M} \frac{e_k}{4\pi\varepsilon_0} \frac{r_i - R_k}{|r_i - R_k|^3}. \quad (4)$$

Inter-particle forces, so called Coulomb forces are defined by

$$F_{Coulomb}(r_i) = q_i \sum_{j \neq i}^{N} \frac{q_j}{4\pi\varepsilon_0} \frac{r_i - r_j}{|r_i - r_j|^3}, \quad (5)$$

and each time step the Coulomb forces are computed dynamically for every particle. Assuming each free particle has equal electric charge q, equation 4 and 5 can be substituted in 3, which has

$$F(r_i) = \frac{1}{4\pi\varepsilon_0} \left(q^2 \sum_{j \neq i}^{N} \frac{r_i - r_j}{|r_i - r_j|^3} + q \sum_{k:R_k \neq r_i}^{M} e_k \frac{r_i - R_k}{|r_i - R_k|^3} \right) \quad (6)$$

as a result.

In order to always achieve a state of equilibrium, a dampening factor is introduced to equation 3:

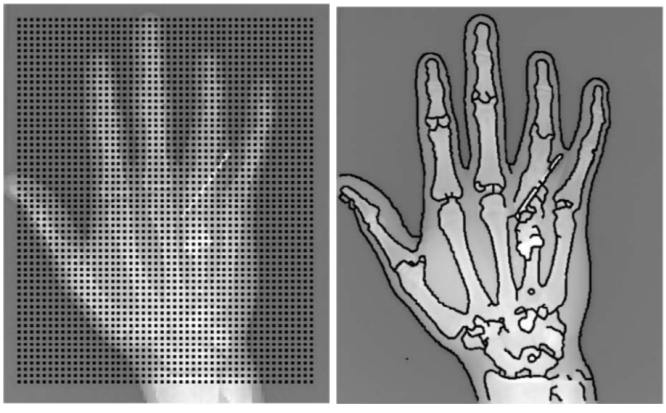
$$F(r_i) = w_1 F_{Coulomb}(r_i) + w_2 F_{Lorentz}(r_i) - F_{damp}(r_i). \quad (7)$$

Under the influence of the given forces, particles will be attracted towards high-frequency details of the image such as the contour of a shape. The computation of the Coulomb forces requires data from every other particle, this implies an update operation of $O(n^2)$ complexity. In practice a cutoff-radius can be used, allowing an accurate approximation by computing the force for K-nearest neighbors. This reduces the complexity of the update to $O(n\ log\ n)$.

The post-initialization (dynamics) part of the method is described in the following simplified pseudo code:

Algorithm 1 CPM Particle dynamics and reconstruction

repeat
  $F_{coulomb}$= EvaluateCoulombForce($p_{1..N}$)
  $F = w_1 * F_{coulomb} + w_2 * F_{lorentz}$
  InsertAndDeleteParticles
  UpdateParticles($p_{1..N}$,F,F_{damp},∇t)
  $t = t + \nabla t$
until Convergence($p_{1..N}$)
ReconstructBoundary($p_{1..N}$)



(a) particle initialization (b) result

Fig. 3: Segmentation of an X-ray image using the CPM [9].

As is visible in figure 3, the method works well for highly concave shapes, and is able to extract internal segments of a structure as well.

3.2 Charged Contour Model (CCM)

Motivated by CPM, Yang et al. [19] proposed a novel active contour model for the segmentation of cardiac SPECT images, named the charged contour model (CCM). This model detects objects starting with a positively charged active contour (in contrast to individual particles) that propagates in an image-base electrostatic field, distributed with negative fixed charges, under the influence of Lorentz forces.

CCM Particle dynamics The positively charged contour is placed in an image-based electrostatic field in which the fixed charge at every pixel position is computed as $e_k = -f(x_k, y_k) \leq 0$, for $k = 1, \ldots, M$, where $f(x,y)$ is the image edge map and M is now the size of image. The Lorentz force field is obtained using the equation for $\mathbf{F}_l$ in equation 4 (similar to CPM). As the Lorentz force decays with squared distance, it produces weak flows in homogeneous areas and stronger flows closer to edges. In the CCM a weighted normalization of the Lorentz force field is performed to speed up the model convergence, such that the normalized field has strong flows in homogeneous areas and weak flows (closer to zero) near edges which finally stop the contour. To achieve this, the edge-preserving weighted normalization process is built on the information supplied by the edge map:

$$\mathbf{F}'_l(\mathbf{r}_k) = \frac{\mathbf{F}_l(\mathbf{r}_k)}{|\mathbf{F}_l(\mathbf{r}_k)|} \exp(-|e_k|) \quad (8)$$

where e_k is the fixed charge located as $\mathbf{r}_k$ in the edge map.

Contour evolution The contour evolution function derived as

$$\frac{\partial C}{\partial t} = \alpha g(x,y)\kappa\mathbf{N} + (1-\alpha)(\mathbf{F}'_l \cdot \mathbf{N})\mathbf{N} \quad (9)$$

where κ denotes the curvature flow, $g(x,y) = (1 + f(x,y))^{-1}$ is a stopping term, $\mathbf{N}$ denotes the contour inward normal, and α is a positive constant that balances the contribution from curvature flow regularization and the Lorentz force attraction. The first term regulates the contour. The orientation of the Lorentz force field at a point is towards strong nearby edges and thus every point of the charged contour will be attracted towards these edges and stop there, whilst being maintained as a closed contour.

Computation of the Lorentz force field based on 4 and 5 is an intensive method which leads to $O(M^2)$ complexity when not using some form of approximation. Therefore, Yang et al. use the Particle-Particle Particle-Mesh method, originally proposed in [7], for fast and accurate evaluation of Lorentz forces.

Automatic Initialization A simple automatic initialization method is specifically designed for the cardiac SPECT application. The divergence of a 2D vector field $\mathbf{F}$ is:

$$Div(\mathbf{F}) \equiv \nabla \cdot F \equiv \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} \quad (10)$$

According to [19], the candidate points are in the brightest regions of the image on the premise that these belong either to the object or the background. The contours then segment the object from either the inside or the outside. It works very well in practice for the cardiac SPECT images and figure 4 shows an example of the automatic divergence point selection process, including a close-up of a divergence region and the entire divergence map.

The simplified pseudo code in Algorithm 2 describes the algorithm of the CCM.

Fig. 4: From left: automatic initialization, propagating CCM, final result [19].

Algorithm 2 Charged Contour Model (CCM)

$F_{Lorentz}$:= ComputeLorentzForces
WeightedNormalization($F_{Lorentz}$)
repeat
  UpdateParticles($p_{1..N}$,F,∇t)
  $t = t + \nabla t$
until Convergence($p_{1..N}$)
ReconstructBoundary($p_{1..N}$)

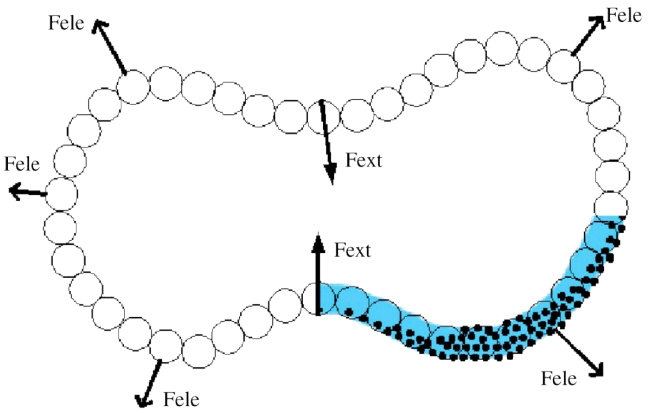


Fig. 5: Concept of a charged fluid. A charged fluid conceptually consists of charged elements (the large circles), each of which exerts a repelling electric force on the others. The fluid elements are modeled as charged particles (the solid dots). The liquid can be influenced by internal electric forces $\mathbf{F}_{ele}$ of repulsion as well as external forces $\mathbf{F}_{ext}$ from the image data. [3].

3.3 Charged Fluid Method (CFM)

Chang et al. [3] introduced the charged fluid method (CFM) by proposing a two-stage evolution framework for image segmentation. The basic concept of charged fluid is explained in figure 5 . The first step distributes the elements of the charged fluid along the propagating interface until an electrostatic equilibrium is achieved. During this procedure, the charge of the fluid elements is interpolated into the neighboring discrete points such that the fluid elements are constrained to move on the lattice. In this approach, the electric force is numerically calculated using the finite-size particle (FSP) method implemented via the fast Fourier transform (FFT) algorithm. After this procedure, the electric force is approximately normal to the propagating contour and can be used to guide curve evolution. The second step advances the propagating front of the charged fluid such that it deforms into a new shape in response to the equilibrium electric force and the image force, which is computed using the smoothed image gradient. The purpose of this procedure is to enable the charged fluid to detect object boundaries that change the shape of the propagating curve. The two-stage evolution is repeated until the propagating front resides on the boundary of objects being segmented.

Charge distribution When using multiple charged fluid, the one having the stronger electric potential dominates the behavior of the overall system. This can dramatically influence the contours of other charged fluids by repelling the fluid elements in the weaker charged fluids. One way to solve this problem is to normalize the electric potential for each charged fluid through Poisson's equation.

Front deformation After the charge distribution procedure, the charged fluid is still an isolated system without the interaction with external force. The front deformation enables the CFM to interact with the image data such that the 1-pixel-wide propagating front deforms the shape in response to the gradient of an image [3].

The complete algorithm is shown as pseudo code in Algorithm 3.

4 Method Comparison

In this section CPM, CCM and CFM are compared based on: parameter sensitivity, particle initialization procedure, computational complexity, performance in case of weak edges and generalization aspects.

4.1 Number of parameters

The number of parameters involved in an image segmentation method and sensitivity of these parameters is an important and decisive factor to estimate the usability and robustness of a method, especially for the users who are not a computer vision specialists. Ideally a method has clear and sensible parameters that are easily configurable, that on the other hand, are not too sensitive.

One of the design purposes of CPM is to reduce parameter-sensitivity as encountered in e.g. the snake method. The main parameters of CPM are time-step Δt, weight of Lorentz force and Coulomb force w_1 and w_2, The minimal and maximal distance of the charged particles d_{min} and d_{max}, a which weights the computation time and density of particle, the damping coefficient β which adjusts the boundary leakage. Yang et al. do not indicate the parameters of CCM algorithm explicitly in [19]. However, we can infer that (a) there is no d_{min} and d_{max} in CCM since dynamic addition and deletion of particles does not exist in CCM, (b) there are similar parameters for the force field because Lorentz force and curvature flow are used in CCM, Coulomb forces on the other hand are not used in CCM. Chang et al. do not give the explicit parameters of CFM either. We can infer [3] that equilibrium quality γ, which determines the degree of electrostatic equilibrium, and weighting factor β which adjusts the image gradient potential, are the two main parameters. The weighting factor β is sufficient in most of the segmentation task [3]. Hence, CFM has the least number of parameters and all of them have fair parameter sensitivity.

4.2 Particle initialization

CPM can segment the image by using automatic initialization with parameters: $w_1 = 0.6$, $w_2 = 0.7$ and $\beta = 0.4$. The initial free particles spread uniformly or are placed where an image has larger gradient magnitude. The performance of segmentation is very good and most of the interesting regions are detected, even in the case of complex images. CCM can initialize the particles automatically and generate good segmentation. CFM has the limitation that the initial particles have to be placed entirely inside regions of interest (ROI). The automatic initialization method of CCM as described in 3.2 is promising. The initialization points are placed in the top 20% brightest regions of the SPECT image, after which the particles will continue diverging until reaching the contour. In contrast, Chang et al. do not propose an automatic initialization method of CFM in [3] and CFM has the limitation that the particles must be initialized somewhere inside the boundary of a segment.

4.3 Computational complexity

The complexity of CPM algorithm is $O(N \log N)$, where N is the number of charged particles. Jalba et al. [9] used a k-d tree data structure to reduce the the computation of the coulomb force and the so called "particle-particle particle-mesh" (PPPM) method to get a fast and accurate evaluation of the electric force. Similarly, CCM used

Algorithm 3 Charged fluid method (CFM)

initialization
Parameter setting of β (image gradient weight) and γ (electrostatic equilibrium)
Image potential computation (weighted image gradient)

repeat
uniform charge distribution over fluid elements

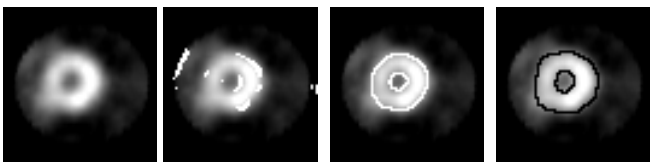
repeat **charge distribution procedure**
forward *FFT* of the charge array
inverse *FFT* computation of the potential array
electric field computation
advance distance computation
charge interpolation
until electrostatic equilibrium (0 net flow of electric charge)

1-pixel-wide front construction
front deformation procedure
effective field computation and
2-pixel-wide interface localization
mean potential computation and charge normalization
until no deformation in the charged fluid shape

the PPPM method as well to implement the computation of Lorentz forces. The CFM uses a FFT-based FSP algorithm to reduce the computational complexity to $O(M^2 \log M)$, in which M is the length of the square that is used for the electric potential computation. The FFT is a widely supported and highly optimized algorithm which contributes to the computational performance of CFM.

4.4 Weak edge detection

CPM can not guarantee closed contours, which inevitably results in gaps in the recovered object boundaries, particularly if the object is occluded or has weak edges [19]. This is because particles intent to push along the boundaries. But SPECT images usually have a lower resolution and largely diffused edge areas, which cannot be segmented by CPM perfectly. CFM has the same problem of leakage when dealing with weak edges, because CFM uses the same image gradient technique as the 'image potential' part of the computation. CCM on the other hand perform a weighted normalization of Lorentz force such that the flow of the particles can stop on weak edges. Figure 6 shows that the CCM outperforms the CPM in SPECT images with weak edges.



(a) original (b) CPM result (c) CCM result (d) hand-labelled

Fig. 6: CPM vs. CCM applied to a SPECT image [19].

4.5 Generalization

In [9], Jalba et al. show that CPM is apt to various applications, e.g. shape recovery, automatic segmentation, skeleton extraction, even extended in three dimensional image. The scenes include synthetic images, natural images, portrait, medical images. CPM generates good segmentation results in most of the situation and some decent results in some noisy or highly-textured images. CCM works perfectly for cardiac SPECT images, but Yang et al. do not give more results of CCM in other type of images in [19]. Some examples of using CFM are given in [3], but most of them are about medical images and simulated phantoms.

4.6 Performance with noisy image

Jalba et al. studied the behavior of the CPM in different types of noise like Gaussian noise, uniform noise and salt and pepper noise with various parameters. CPM can handle most of the cases and generates a recognizable contour, except for the cases with the highest extent of uniform noise and salt and pepper noise. The performance with noisy image of CCM is not given literally in [3]. In a noisy region, the propagating interface of CFM is particularly rough, with many sharp points.

5 Conclusion

We have described three particle-based deformable models: Charged Particle Model (CPM), Charged Contour Model (CCM) and Charged Fluid Method (CFM) and compared them in various aspects. The comparison results are summarized in table 1. All three methods yield very similar results and are an improvement over previously developed deformable models, such as the active contour model.

The results show clearly that CFM is by far the most complex method but it does not perform significantly better than CPM. CCM is a modified version of CPM that is fine-tuned specifically for a special kind of images. CCM primarily deals with the issue of weaker edges, but otherwise does not improve CPM. The charged particle model, CPM, is by far the most generally useful method and by far less complex that CFM.

Our comparison is based mainly on the articles [3], [9] and [19]. Further investigations could be retrieving or implementing the codes of three methods. Then we can estimate a preciser benchmark and test the three methods under the uniform condition. In this way, our results will be more accurate and less subjective.

References

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 415–421, New York, NY, USA, 1998. ACM.

[2] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(9):855 –867, sep 1990.

[3] H.-H. Chang and D. J. Valentino. Image segmentation using a charged fluid method. *J. Electronic Imaging*, 15(2):23011, 2006.

[4] M. de Bruijne and M. Nielsen. Image segmentation by shape particle filtering, 2004.

[5] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[6] R. Grzeszczuk and D. Levin. ldquo;brownian strings rdquo;: Segmenting images with stochastically deformable contours. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(10):1100 –1114, oct 1997.

[7] R. Hockney and J. Eastwood. Computer simulation using particles, 1988.

[8] F. Ibrahim, N. Osman, J. Usman, and N. Kadri, editors. *Spinal Curvature Determination from an X-Ray Image Using a Deformable Model*, volume 3rd Kuala Lumpur International Conference on Biomedical Engineering 2006: Biomed 2006, 11-14 December 2006, Kuala Lumpur, Malaysia of *IFMBE proceedings*. Springer, 2007.

[9] A. C. Jalba, M. H. F. Wilkinson, and J. Roerdink. CPM: a deformable model for shape recovery and segmentation based on charged particles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1320 –1335, oct. 2004.

Methods Overview

Sensitivity to initialization

Snake	CPM	CCM	CFM
Yes	No	No	Yes, must be within boundary

Automatic initialization

Snake	CPM	CCM	CFM
No	Yes	Yes	No

Computional complexity of particle update

Snake	CPM	CCM	CFM
N/A	$O(N^2)$, reduced to $O(NlogN)$	$O(N^2)$, reduced to $O(NlogN)$	$O(N^2)$ reduced to $M^2 \log M$ (where M is length of ROI)

Behaviour on noisy images

Snake	CPM	CCM	CFM
N/A	Good	OK	fair

Boundary cavity convergence

Snake	CPM	CCM	CFM
No	Good	Ok	Ok

Number of parameters

Snake	CPM	CCM	CFM
2 or 3	7	< 7	2, but often 1 is sufficient

Generalization

Snake	CPM	CCM	CFM
OK	Good	fair, specific to SPECT images	fair

Implementation challenges

Snake	CPM	CCM	CFM
fair	dynamic addition/removal of particles	fair	very complicated dynamics

Extendible to 3D

Snake	CPM	CCM	CFM
Possible	Yes	Yes	Yes

Table 1: Method comparison

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[11] F. Leymarie and M. Levine. Tracking deformable objects in the plane using an active contour model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(6):617 –634, jun 1993.

[12] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[13] T. Mcinerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1:91–108, 1996.

[14] H. P. Ng, K. C. Foong, S. H. Ong, P. S. Goh, and W. L. Nowinski. Medical image segmentation using feature-based gvf snake. *Conference Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, 2007:800–803.

[15] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[16] D. J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14 – 26, 1992.

[17] A. Wismuller, F. Vietze, J. Behrends, A. Meyer-Bse, M. Reiser, and H. Ritter. Fully automated biomedical image segmentation by self-organized model adaptation. *Neural Networks*, 17(8-9):1327–1344, 2004.

[18] C. Xu and J. Prince. Snakes, shapes, and gradient vector flow. *Image Processing, IEEE Transactions on*, 7(3):359 –369, mar 1998.

[19] R. Yang, M. Mirmehdi, and D. Hall. Charged contour model for cardiac spect segmentation. In *Medical Image Understanding and Analysis, MIUA*, pages 171–175. BMVA Press, July 2006.

Imbalanced class distribution in Receiver Operating Characteristic and Precision-Recall

Werner Buck, *Software Engineering & Distributed Systems*, and Jan Willem de Jonge *Intelligent Systems*

Abstract—In the field of machine learning and pattern recognition there is a need to evaluate classification methods (classifiers). This paper researches both classic statistical evaluation methods and visual evaluation techniques. Receiver Operating Characteristic (ROC) and Precision Recall (PR) are visual evaluation methods, which give a two-dimensional graphical representation of the classifiers. It is shown that classic evaluation methods can give misleading results regarding classifier performance when used with imbalanced datasets, and is investigated by performing a comparison test using artificial datasets with different class balance ratios. The test reveals that certain methods like *accuracy* are sensitive to class imbalanced datasets and can give misleading results. The effects of class imbalanced datasets is also tested using non-discrete classifiers and is followed by a conclusion about the effectiveness of ROC and PR regarding imbalanced dataset problems.

Index Terms—Receiver Operator Characteristics, Precision Recall, Area Under Curve, evaluation, pattern recognition, classification, classification evaluation

1 Introduction

In machine learning and pattern recognition, classification is a procedure for assigning given input data to a certain category. The algorithm that implements classification is called a classifier. Single input data are referred to as *instances*, and the categories to which instances belong are called *classes*. Classifiers can be evaluated by their comprehensibility, computational performance and predictive performance. When referring to the performance of a certain classifier, we refer to predictive performance.

There are a number of ways to evaluate performance of classifiers. The most commonly used metric is accuracy. Accuracy is used as a statistical measure to show how well a binary classification test correctly identifies or excludes a condition. However, Provost et al. have argued that simply using accuracy as a evaluation technique, can lead to misleading results [9]. Instead, the paper recommends using Receiver Operating Characteristics (ROC), which is a classification evaluation method that presents the quality of the classifier in the form of a curve in a graph.

Precision Recall (PR) is often used in the field of Information Retrieval [7] and is an alternative to ROC curves. PR is used in particular for tasks with an imbalanced class distribution [10].

A frequent problem in classification is that the distribution of the number of samples is not equal between the classes, this is called a biased or imbalanced class distribution. When evaluating classifiers on a dataset that has an imbalanced class distribution it is not always obvious which classifier performs better. In this paper we give an introduction to traditional non-visual evaluation metrics, as well as research the use of ROC and PR. This paper concludes by performing a test on ROC and PR, and show how different ratios of class distribution can affect the results.

2 Classifier evaluation

In binary classification, a classifier assigns an instance to either class 1 (negative) or class 2 (positive). The number of errors or positives the classifier makes can be represented in a *confusion matrix* also known as a contingency table. When using a supervised algorithm in classification, supervised means that the supervisor has knowledge regarding which instance belongs to class 1 or class 2. When a researcher uses a supervised algorithm he must be in possession of a dataset of which he knows the class for each instance.

For information on obtaining reprints of this article, please send e-mailto:wernerbuck@gmail.com.

Table 1. Confusion Matrix

	actual positive	actual negative
predicted positive	True Positives (TP)	False Positives (FP)
predicted negative	False Negatives (FN)	True Negatives (TN)

Cross-validation is used to ensure that the classifier performance can be generalized to an independent data set. A common method is k-fold cross-validation. In k-fold cross-validation the dataset is randomly partitioned in k disjoint sets. One of these subsets is retained to use as the validation data to test the classifier. The rest of the data is used to train the classifier. This process is repeated k times so that each instance is used one time as evaluation data. The results are usually combined by taking the average of the k results.

These results are then summarized in the confusion matrix. The confusion matrix has four categories: True positives (TP) are the number of instances correctly classified as positive. True negatives (TN) are the number of instances that are classified correctly as negative. False positives (FP) are instances that are classified as positive but are truly negative. Finally, False negatives (FN) are positive instances classified incorrectly as negative. A confusion matrix is shown in Table 1. The values along the major diagonal are the correct decisions made, and the values of the minor diagonal represents the errors (the confusion) between the various classes. The confusion matrix forms the basis of many common metrics which can be used to evaluate the performance.

For the next few examples we will consider as an example the diagnosis of a particular disease in patients. Each patient taking part of the study has or does not have the disease. Then the confusion matrix categories can be interpreted as:

- True positive: Ill patient correctly identified as ill
- False positive: Healthy patient incorrectly identified as ill
- True negative: Healthy patient correctly identified as healthy
- False negative: Ill patient incorrectly identified as healthy.

2.1 Accuracy and Precision

Accuracy shows how well a binary classifier correctly identifies or excludes a condition. An accuracy of 100% means that the classifier

is perfect in the sense that no instance is misclassified. Accuracy is defined as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

In terms of our example, it returns the percentage of data correctly diagnosed against the given population of patients. However, when the accuracy metric is used on highly imbalanced datasets, predictive models with low accuracy may have greater predictive power than models with higher accuracy. This is known as the accuracy paradox. [8] The problem lies in the fact that accuracy can be improved by always predicting negatively. Therefore it is noted that simply using accuracy as an evaluation is not enough.

Precision is defined as the number of true positives against the number of all positive results (both true positive and false positive). Precision is defined as follows:

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

In terms of the previous example, this formula returns the percentage of ill patients correctly diagnosed against the total number of patients identified as being ill.

Precision and Accuracy can best be explained using the dartboard analogy. Imagine a person throwing darts hitting the bullseye. The closer the dart hits to the bullseye, the more accurate the tosses are. However if a dart misses the bullseye with every throw but all darts end up together they can still be very precise, but not accurate.

2.2 Sensitivity and Specificity

Sensitivity (also called Recall in the field of Information Retrieval) gives feedback on how well the classifier identifies positive results.

$$Sensitivity = Recall = \frac{TP}{TP+FN} \quad (3)$$

In regard to our example, the sensitivity of the test is the percentage of people who have the disease and are tested positive for it. In other words, sensitivity is the probability of a positive test given that the patient is ill. If sensitivity is 100%, it means that the test recognizes all actual positives.

Specificity gives feedback on the ability of the classifier on how well it identifies negative results. It is defined as follows:

$$Specificity = \frac{TN}{TN+FP} \quad (4)$$

Considering the example, the specificity of a test is defined as the percentage of patients who do not have the disease and will test negative.

Precision and Recall is used in Information Retrieval to evaluate algorithms that retrieve a set of documents based on a search query. Both sensitivity and specificity are heavily used in medical tests [6].

3 Receiver Operator Characteristics

A ROC curve is a technique for visualizing classifier performance in a two-dimensional curve. The ROC curve was originally developed by electrical engineers and radar technicians in World War II. It was used in signal detection to show the hit/false alarm rates of classifiers [2][13]. From that point, ROC became popular in visualizing and analyzing the behavior in medical diagnostic tests [12]. ROC graphs were used in the field of machine learning as early as 1989 by Spackman [11], who used it to evaluate and compare machine learning algorithms. After an article in Scientific American written by Swets et al in 2000 [13] ROC became widely used.

A point in two-dimensional ROC space is constructed using information of the confusion matrix, namely the True Positive Rate (TPR) and the False Positive Rate (FPR). The TPR is the same as Sensitivity or Recall.

$$True\,Positive\,Rate = \frac{TP}{TP+FN} \quad (5)$$

$$False\,Positive\,Rate = 1 - Specificity = \frac{FP}{FP+TN} \quad (6)$$

The TPR is plotted on the Y axis and the FPR is plotted on the X axis. A classifier is a mapping from instances to predicted classes. There are two types of classifiers. The *discrete* classifier outputs only the predicted class label. There are also non-discrete classifiers such as Naive Bayes or neural network which return a probability or score that represents the degree to which that instance belongs to the class. In ROC space the performance of discrete classifiers is represented by a single point (FPR, TPR). Classifiers with a probability or score are represented by a curve generated by varying the decision threshold.

Figure 3 shows an ROC graph containing discrete classifiers. The position of a classifier in ROC space matters a great deal. The top-left point represents a perfect classifier as the classifier has no false positives and only true positives. If a point (classifier) is closer to the top-left, informally the closest one is better. Fawcett notes that classifiers that fall on the left side of the ROC graph near the X axis can be thought of as "conservative": they make positive classifications only with strong evidence so they make few false positive errors, but often have low true positive rates as well [3]. He completes this analogy by noting that classifiers in the top-right can be thought of as "liberal": they make positive classifications with weak evidence as they classify nearly all positives correctly, however they often have high false positive rates [3]. In line with this example, it can be seen that A is more conservative than B. Fawcett also notes that an ROC graph in general depicts relative trade-offs between benefits (true positives) and costs (false positives) [3].

The dividing line in the middle $x = y$ marks the random performance strategy. If a classifier falls on this line it means that the classifier is no better than the flip of a biased coin. Thus the performance of Classifier C at (0.7) on X and Y axis is guessing the positive class 70% of the time. Consequently if a classifier finds itself in the lower right corner like Classifier E it means that it intentionally classifies wrongly and the classifier can easily be inverted to be at the other side of the line.

Non-discrete classifiers like Naive Bayes classifier or neural networks return a continuous output, which can be used with a threshold to produce a discrete classifier. For example if the classifier value is above the threshold it returns positive, else negative. Each threshold makes up a new point in ROC space because with every threshold the confusion matrix is updated and the original non-discrete classifier are now many different binary classifiers. The total of points make up for a single a curve of a single non-discrete classifier. Figure 3 shows two algorithms (ROC-curves) acting on a dataset to predict if the breast cancer is malignant. Class distribution ratio is approximately 1:2 (241 malignant and 458 benign).

3.1 Area Under the Curve (AUC)

In machine learning, the Area Under the Curve (AUC) of the ROC-curve is often used as a single value metric of classifiers performance. Calculating the AUC is done by taking the integral of the convex hull of all discrete classifier points forming the non-discrete classifier. The convex hull in ROC is defined as a set of points with the following criteria:

- Linear interpolation between adjacent points.
- No points above the final curve.
- points connection lines equal or under convex hull.

AUC is a portion of the area of the unit square, hence its value will always be between zero and one. An AUC of 0.5 means the classification is no better than random and an AUC of 1 means perfect classification. Because AUC is a single scalar value, a multitude of non-discrete classifiers can be compared relative to each other in contrast to regular ROC, as there is a limit to the number of curves that can be displayed in one graph. For additional information and detailed discussion on the statistical interpretation of AUC, see Fawcett [3]. Also

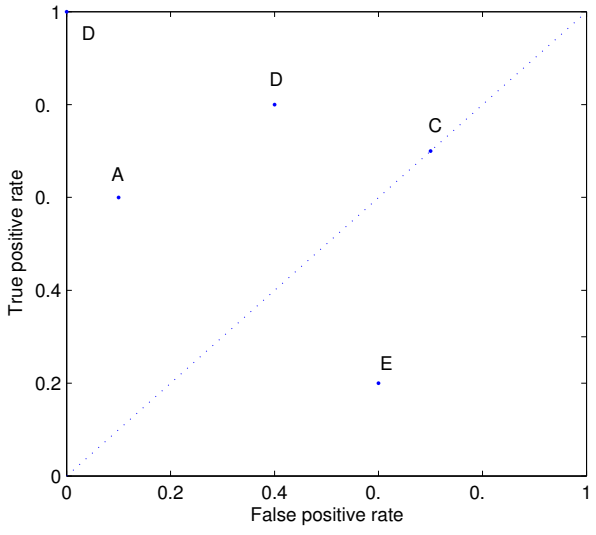


Fig. 1. Discrete ROC graph, as discussed in section 3.

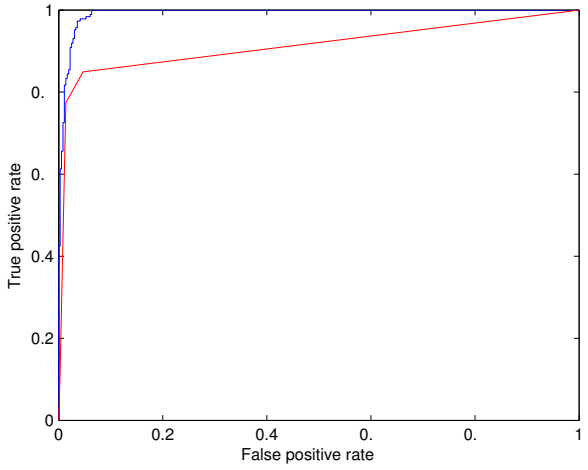


Fig. 2. Example ROC curve, as discussed in section 3.

Huang and Ling [5] showed both empirically and formally that AUC is a better measure than accuracy. However, AUC as a performance measure has been questioned because it has some significant problems [4] with certain classifiers but is still heavily used for benchmarking large numbers of classifiers.

4 Precision Recall (PR)

Precision-Recall (PR) curves are often used in Information Retrieval [7]. It is also known to be a good alternative to ROC when used with tasks with a large imbalance in class distribution. A PR curve is shown in Figure 3 and is plotted on the same dataset as with Figure 3.

The main difference between ROC and PR is the visual representation. In PR space, Recall (also known as true positive rate) is plotted on the x-axis. Precision is plotted on the y-axis. For ROC the objective is to be close to the top-left and for PR the goal is to be as close as possible to the top-right. When comparing the two figures in Figure 3 it can be seen that the red classifier dominates over the blue classifier. However the red classifier in the PR curve shows slightly better results than in the ROC curve.

In terms of the algorithm, PR works in the same way as ROC regarding non-discrete classifiers. It plots thresholds and creates discrete classifiers for each threshold which results in a curve.

The relationship between ROC and PR has been investigated by Davis and Goadrich [1]. The paper shows that for a given dataset a one-to-one correspondence exists between the ROC and PR curve and that one curve dominates another in ROC-space, if and only if the first dominates the second in PR-space. A curve dominates another curve in ROC space if all other ROC curves are beneath it or equal to it [**?**].

Calculating AUC under PR offers the same benefits as calculating AUC of an ROC curve. However with PR, calculating the AUC involves a more complicated interpolation and unfortunately falls outside the scope of this work. More detailed information on this subject can be viewed under [1].

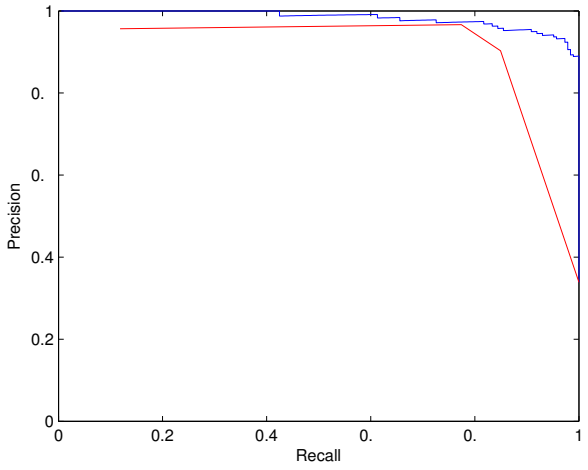


Fig. 3. Example PR curve, as discussed in section 4.

5 Class distribution imbalance

Our hypothesis is that "class distribution problems can lead to misleading or a wrong view of the results of classification evaluation metrics". We decided that the best method to investigate the possible influence of class distribution imbalance on classifier evaluation methods is to generate artificial classification results. We generated a data set of two partially overlapping clusters to form nearly one cloud. With logistic regression and a decision tree we generated two sets of classification results, each class containing 25000 instances. By then randomly dropping instances from a particular class we introduced an imbalance in class distribution.

In Figure 4 we show the results of our test. It shows the ROC and PR of two classifiers with three different class distribution ratios being: 1:1 (original), 1:10 and 1:100. The 1:1 ratio serves as the control reference that has no class distribution imbalance problem. The ratio of 1:10 implies that one class has ten times more instances then the other class, and the ratio 1:100 implies that one class has hundred times more instances than the other.

Remember when viewing the curves that in ROC and PR space the curve itself represents the non-discrete classifier and the points that form the line are all individual discrete classifiers with a certain threshold value.

5.1 Accuracy

First we look at how the traditional classification evaluation method, i.e. accuracy performs. The accuracy of a non-discrete classifier can be calculated when a threshold has been selected. For a point on the ROC curve the confusion matrix on a dataset is fixed, hence each accuracy metric in our experiments can be calculated for a given threshold meaning every point in the graph.

Table 2. Accuracy at recall of 50%, 90%and 99% for logistic regression (LR) and decision tree (DT).

Class distribution ratio	1:1	1:1	1:10	1:10	1:100	1:100
Algorithm	LR	DT	LR	DT	LR	DT
Recall = 50%	0.75	0.74	0.95	0.94	0.99	0.98
Recall = 90%	0.89	0.93	0.89	0.96	0.89	0.96
Recall = 99%	0.78	0.66	0.61	0.38	0.58	0.28

Table 3. ROC-AUC with different class distribution ratios

Class distribution ratio	logistic regression	decision tree
1:1	0.965	0.961
1:10	0.964	0.961
1:100	0.963	0.960

In Table 2 the accuracy values for each classifiers at a recall value (TPR) of 0.5, 0.9 and 0.99 is shown. As expected from earlier observations by other researchers, the results show that accuracy is significantly affected by class distribution imbalance. The severely imbalanced datasets score high when the desired recall is not too high, but this does remain valid when confronted with a high recall. The accuracy for the classifiers is different at a fixed recall.

Further analysis shows that it is not the same classifier with the best accuracy at different recall levels, in other words there is a better classifier. When considering the case when there is no imbalance, the accuracy is 0.5. When the positive class has a prior probability of 0.1 the same classification can result in an accuracy of 0.99 which is a significant increase. This shows that accuracy is not a good metric to evaluate classifier on a imbalanced dataset and should be used with caution and should never be used as the only evaluation metric.

5.2 ROC-AUC

To compare the curves in a more empiric way, instead of visually noting the difference between curves, we calculated the AUC values from the ROC curves shown in Figure 4. Table 3 lists these values. It can be seen that the AUC of both classifiers are only very slightly affected by the class distribution imbalance. Even at the ratio of 1:100 the AUC manages to stay almost the same. It seems that ROC is a far better measure to evaluate classifiers when the researcher does not care about class imbalance, as it remains unaffected.

5.3 ROC and PR performance

The resulting ROC curves for the different imbalanced sets give almost the same curve. This is because The ROC uses only values from the left column of the confusion matrix. If the class distribution balance changes only the total number of positive examples changes.

The PR curves in contrast to the ROC are very different. This is caused by the fact that the PR curve uses metrics from both columns of the confusion matrix therefore it is sensitive for class distribution imbalance. When comparing classifiers performance with ROC and PR curves, it can give more detail of the performance of a classifier than ROC-AUC. This is because a researcher can see exactly with which threshold value the classifier fails. ROC-AUC lacks this information.

When evaluating the curves it is clear that the differences in balance have a major impact on the PR curve. When the class distribution is imbalanced the precision drops for a fixed recall value.

6 Conclusion

Evaluating the performance of classifiers can be done by using different evaluation methods. In this paper we have discussed some of the single value metrics, the ROC curve and the PR curve. For non-discrete classifiers the single valued metrics (except AUC) can only be calculated after a threshold is selected. Also, when the class distribution is imbalanced the AUC of the ROC is not affected while the accuracy metric can become misleading. Therefore AUC of the ROC-curve is a better measure than accuracy when dealing with a imbalance in data distribution. Also we recommend that when evaluating classifiers to not restrict yourself to one method, but use both like PR and ROC together with a plot of Sentivitity and Specificity. Using more methods increases the chances of not being misled by class distribution problems while getting a better evaluation at the same time.

Acknowledgements

The authors wish to thank the staff and reviewers of the Student Colloquium 2012 of Computing Science RuG

References

[1] J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 233–240, New York, NY, USA, 2006. ACM.

[2] J. P. Egan. *Signal detection theory and ROC analysis*. Series in Cognition and Perception. Academic Press, New York, NY, 1975.

[3] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[4] D. J. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Mach. Learn.*, 77(1):103–123, Oct. 2009.

[5] J. Huang and C. X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17:299–310, 2005.

[6] A. G. Lalkhen and A. Mccluskey. Clinical tests: sensitivity and specificity. *Contin Educ Anaesth Crit Care Pain*, 8(6):221–223, Dec. 2008.

[7] C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1 edition, June 1999.

[8] F. Miller, A. Vandome, and J. McBrewster. *Accuracy Paradox*. VDM Verlag Dr. Mueller e.K., 2010.

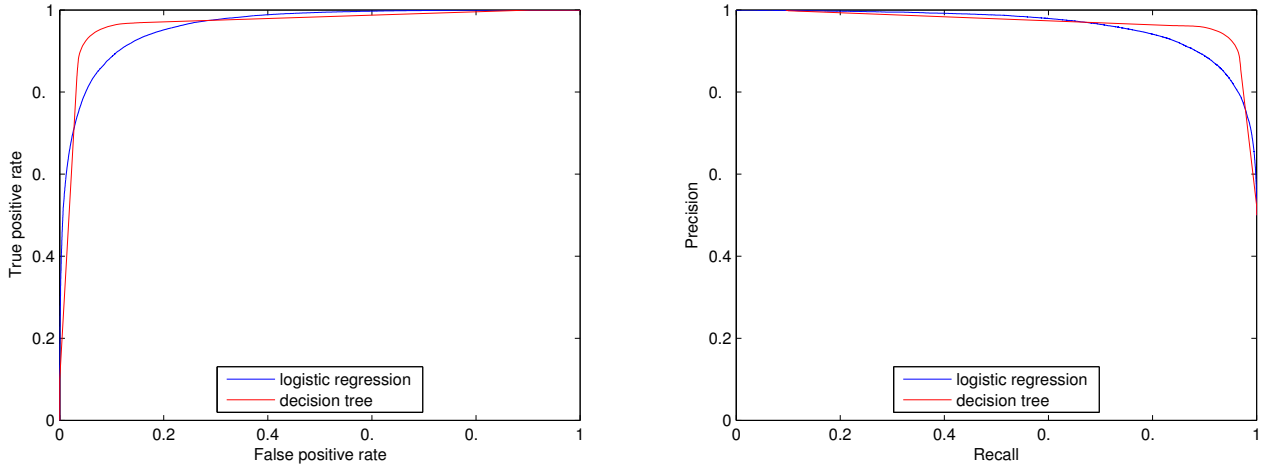
[9] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, 1997.

[10] V. V. Raghavan, G. S. Jung, and P. Bollmann. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229, 1989.

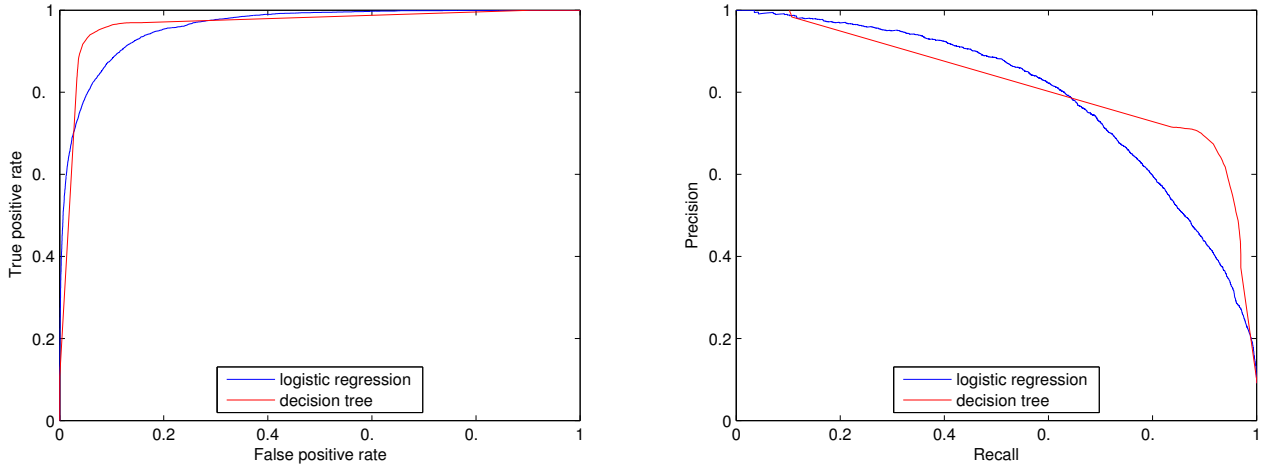
[11] K. A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, page 160163, San Mateo, CA, 1989. Morgan Kaufman.

[12] J. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:12851293, June 1988.

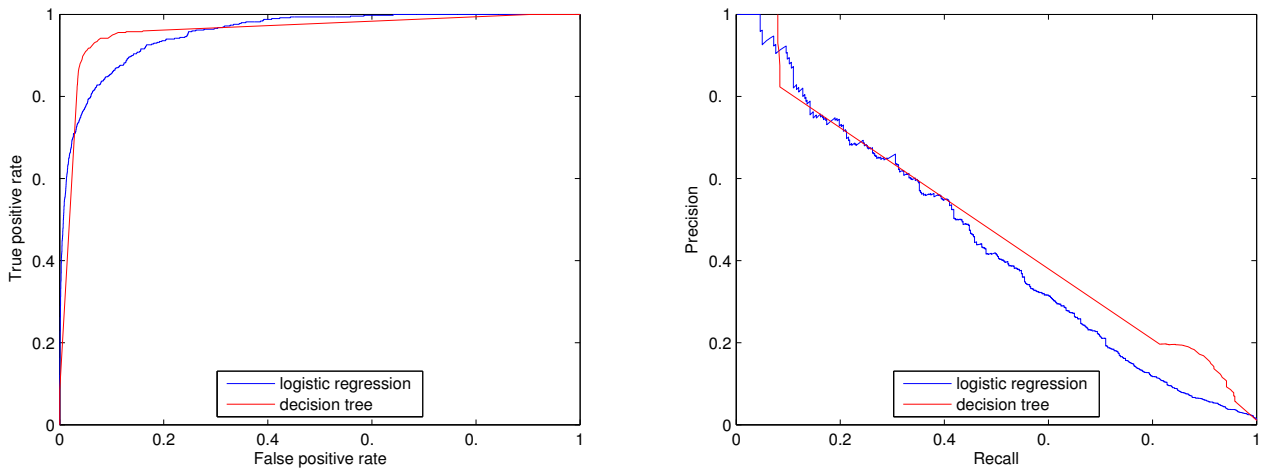
[13] J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283(4):82–87, Oct. 2000.



(A) Left ROC, right PR Class distribution ratio: 1:1 (balanced)



(B) Left ROC, right PR Class distribution ratio: 1:10 (imbalanced)



(C) Left ROC, right PR Class distribution ratio: 1:100 (imbalanced)

Fig. 4. Comparison of ROC and PR with different class distribution imbalances.

Cyber Security in the Smart Grid

Assel Bekbatyrova Sunna Björg Sigurjónsdóttir

Abstract— In order to improve efficiency, importance, reliability, economics, and sustainability of electricity services the Smart Grid is used to predict and intelligently respond to the behaviour and actions of all electric power users connected to it [1]. A secure Smart Grid is able to withstand physical attacks and cyberattacks without suffering enormous recovery costs and massive blackouts. With the growth of the Smart Grid several security issues are introduced. Security measures taken a few years ago may not be holding up with this rapid growth. We take a closer look at the current situation regarding security in the Smart Grid. We investigate what methods are used today, what type of security issues we can expect in the future and what type of measures we think need to be taken to secure the Smart Grid. Our research is based on several existing articles discussing security in the Smart Grid and their findings, where we focus on the work of A. Metke and R. Eki, "Security Technology for Smart Grid Networks" [10] and D. Wei et. al, "Protecting Smart Grid Automation Systems Against Cyberattacks " [9]. We discuss the proposed security solutions given by A. Metke et.al. and D. Wey et.al. and conclude that a solution based on a three layered architecture with a security service proxy is the best choice for the future of the Smart Grid. The integrated security framework is designed to protect the power grid automation system from cyberattacks, as well as from external and/or internal networked sources. It gives desirable results in terms of scalability, modularity and manageability.

Index Terms — Smart Grid, electrical grid, cyber security, cyberattacks, network security, public-key infrastructure (PKI).

1 Introduction

The vast black-out in the northern and eastern U.S. and Canada in 2003 caused a $6 billion loss [2]. It indicated that the state of the current electrical grid was outdated and requires an upgrade. There have been more similar cases.

In 2009 it was reported that software programs had been illegally supplanted in the US Smart Grid, believed to have originated in China, Russia and other countries [3]. These programs were believed to be designed to navigate the electrical system and its controls and could be used to damage the power grid. Therefore, besides the goals to become more reliable, efficient, environmentally friendly, and co-operate with alternative energy resources, it should resist cyberattacks. In other words, it has to be "smart".

The Smart Grid's goal is to integrate the benefits of information technology with current and new energy generations and storage technologies to provide consumers with a more reliable, efficient power infrastructure.

This paper examines the concerning cyber security in the Smart Grid and discusses a few current solutions, as well as proposed additions to those solutions, to help protect the electrical grid from cyber attacks.

This paper is organized as follows: Section 2 presents a conceptual model of Smart Grid and identifies the 7 major domains which make up the grid, as well as 3 main functions of the Smart Grid. Then we analyze in-depth three important working components of the Smart Grid. Section 3 identifies the major challenges of the Smart Grid and the requirements important to focus on when finding solutions to those challenges. In section 4 we talk about cyberattacks, cyber security and their impact to the Smart Grid. Section 5 introduces two existing security solutions available in the electrical grid. Section 6 and 7 discuss suggested solutions that are identified. We end this paper on our conclusion to the discussions above.

2 Background

When talking about the Smart Grid we identify seven major domains identified by NIST (National Institute of Standards and Technology), as seen in Fig. 1 [4]. The solid lines in the figure represent a secure communication interface and the dashed lines are electrical interface. The Bulk Generation generates electricity in bulk quantities from renewable and non-renewable energy sources, as well as having storage devices for later distribution. This is typically where the process of delivering electricity to customers begins. The transmission domain transfers the electrical power from the Bulk Generation domain to the Distribution domain via various substations. Its main responsibility is to maintain stability across the network. Electricity is distributed to and from customers in the Smart Grid via the Distribution domain. The Customer domain is the end user (home, commercial and industrial) who is connected to the Smart Grid via smart meters. The Service Provider is the organization which provides electric services to customers while the Operations domain performs the management functions needed for a smooth circulation in the Smart Grid. Finally, the Market domain balances supply and demand, as well as performs pricing.

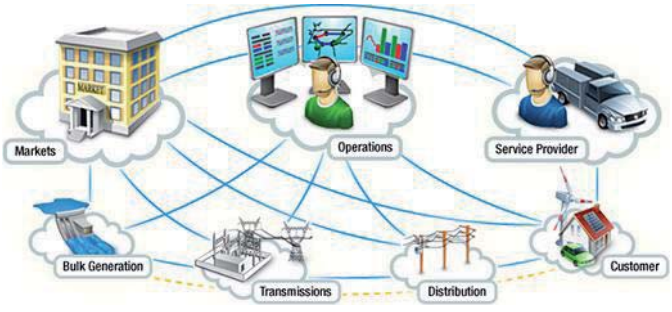


Fig. 1. Conceptual model of the Smart Grid [4]

In the following subsections we describe main components and functions of the smart grid

2.1 Working Components of a Smart Grid

A Smart Grid consists of three major components [5]:

A. Demand management

Demand management deals with reduction of electricity consumption in homes, offices and factories. The management's work includes monitoring electricity consumption and managing electricity demand for electrical gauges. It is comprised of demand-response programs, smart meters and variable electricity pricing, smart buildings with smart appliances and energy dashboards. The total of electrical gauge management applications allow utility companies and

- *Assel Bekbatyrova is a masters student at the University of Groningen E-Mail: a.bekbatyrova@student.rug.nl.*
- *Sunna Björg Sigurjónsdóttir is a masters student at the University of Groningen, E-Mail: s.b.sigurjonsdottir@student.rug.nl.*

consumers to effectively control and react to changes in their electricity consumption.

- **Demand - response**
  During peaks of energy consumption, consumers receive electronic alerts asking to reduce their energy. It can be done by turning off non-essential gauges. In the future, when Smart Grid is fully developed, gauges will receive alerts and need for manual intervention will be eliminated.
- **Smart meters and variable electricity pricing**
  Today's price of electricity is very fluctuating; it can be determined by supply and demand. Generation capacity and weather conditions can also be reasons for price changes. During off-peak hours prices can be reduced to 50% compared to peak hour pricing, and vice versa. Despite price changeability many retail consumers are charged for an electricity price which is regardless of the time of day. Therefore, a lot of consumers could regulate their use of electricity and reduce their electricity bill. To solve this issue, utility companies are replacing mechanical meters with smart ones. Smart meters help utility companies to monitor consumer's usage and consumers themselves are able to regulate their usage regarding time of day.
- **Smart buildings with smart appliances**
  In the past decades architects have constructed buildings with energy-efficient systems, such as multi-pane windows, better insulation, and appliances that use less energy and environmentally friendly. However, up-to-date technology innovations manage to do more; they monitor and reduce energy consumption by appliances, such as heating, ventilation, air conditioning and lighting. They are gathered into IT infrastructure, which allows them to communicate with each other. It helps to work more efficient and reduce electricity usage.
- **Energy dashboards and controllers**
  Already developed online energy dashboards and controllers will allow regulating turning on/off main appliances. Moreover, it helps to adjust thermostats to reduce energy consumption and CO2 emission by real-time monitoring into individual consumption and generation.

B. Distributed electricity generation

Generally electricity is generated in enormous centralized facilities, such as fossil fuel, nuclear, hydropower or solar power plants. Distributed generation, also called decentralized energy or distributed energy, generates electricity from many small energy sources. Distributed electricity generation allows collection of energy from many sources and may give lower environmental impacts and improved security of supply [16].

- **Renewable energy using micro-generation devices**
  Micro-generation devices, small-scale energy-generation equipment designed for home and office usage, such as rooftop solar panels, wind turbines, are getting more popular for individual, commercial and industrial customers.
- **Storage and PHEVs**
  Pumped water storage used to be beneficial in storing electricity on a large scale, but with appearing of PHEVs[1] and electric cars, new opportunities will come.

C. Transmission and distribution grid management

Utility companies are approaching IT to solve issues with controlling and monitoring electrical grid in real-time. By solving these issues, the lifetime of the current grid can be extended and reduce investments, which are needed for an upgraded infrastructure. New solutions could be used to monitor only high-voltage transmission grids, but nowadays due to increasing grid reliability it covers medium- and low-voltage distribution grids as well.

- **Grid monitoring and control**
  Utility companies install sensors to avoid expensive power stoppages. In addition to monitoring and controlling in almost real-time, those sensors can detect faults in time and aid to solve them.
- **Smart Grid security**
  Smart Grid security comprises three categories: physical security, data security and cyber security [8]. Physical security defines as protection of physical infrastructure of the Grid including advanced meter interface (AMI) hardware (smart meters, transmission lines, generating equipment and control room) from damage. This damage could be either from international attacks, which use weapons, such as magnetic pulses or from electrical storms. Data security relates to the privacy of information transforming via Smart Grid. In general, it is customers' information, which includes personal details, financial information and energy-usage patterns. Individuals could suffer from damaged data due to hacking. Cyber security refers to vulnerabilities of Smart Grid, which hackers could use for intentional infiltration by Internet or other digital-information management system and damage electricity delivery system.

2.2 Main functions of a Smart Grid

The power grid's main functions are divided in three levels:

A. Corporate

At the corporate level, major functions of business management and operational management are presented [9]:

- Planning - plan of equipment and line upgrades based on forecast of load and generation source, market conditions and system utilization;
- Accounting – manage contracts and bids with other market participants;
- Engineering – engineer and design system for distributions lines and automation systems;

B. Control center

At the control level, important real-time and non-real-time functions are presented:

- Forecast – forecast load and power generation sources;
- Monitoring – monitor state, activity, load and equipment and conditions of the system;
- Operation – change operation, start emergency procedure, perform system restoration, etc.;
- System analyzing – update model, analyze stability and power flow;
- Training – train operator;
- Logging – archive logs and reports;

C. Substation

At the substation level, major real-time and non-real-time functions are introduced:

- Normal operations – gather data and alarms, send them to control center, execute commands.
- Emergency operation - protect power system, load shedding, recovery from load shedding, shunt and compensation controls;
- Engineering – protection, automation and line engineering.

3 Major challenges and solutions for Smart Grid

Several users of the Smart Grid are using Internet technologies, broadband communication and nondeterministic communication environments in their deployments. Electric-grid communications have relied in the most part on serial communication environments to provide for controlling and monitoring. There is also an express growth of Smart Grid systems that are without acceptable security and reliability planning.

[1] PHEV Plug-in Hybrid Electric Vehicle

In addition, user and corporate privacy are becoming more of an issue with the increase of home automation and metering. Data about energy use and patterns could give indications of when homeowners are at work or travelling, as well as being an indicator of what type of appliances are available in the users home. Regarding the corporate sector, an increase in power draw could suggest an increase in business operations, which could provide business intelligence to competitors.

H. Khurana et. al. list six architecture-based requirements which are important to consider for effective cybersecurity solutions[6];

Availability and confidentiality:

Due to the fact that the cyber infrastructure of the Smart Grid manages steady power flow in the physical infrastructure, the requirement for availability is highly regarded. It is so important to most stakeholders that it is regarded as an acceptable trade-off with confidentiality.

Efficiency and scalability:

Different types of systems deployed to the Smart Grid, call for different types of real-time requirements, which calls for a close look at efficiency. Scalability is of great concern since the Smart Grid needs to be able to handle a fast growth of devices as well as the growing number of interactions between the entities on the grid.

Adaptability and evolvability:

Since devices can last for a long time, they could outlive cryptographic tool's lifetime, which calls for the importance of adaptability and evolvability.

4 Cyberattacks, cyber security and impact to Smart Grid

The implementation of the Smart Grid could bring benefits to electric utility industry, providing a more economic and efficient system. However, it is also adds a variety of security and reliability concerns, especially, in the cybersecurity field. Cybersecurity is heavily dependent on authentications, authorizations and privacy technologies. According to the report of Electric Power Research Institute (EPRI), "Cybersecurity is critical issue due to the increasing potential of cyber attacks and incidents against this critical sector as it becomes more and more interconnected. Cyber security must address not only deliberate attacks, such as from disgruntled employees, industrial espionage, and terrorists, but inadvertent compromises of the information infrastructure due to use errors, equipment failures, and natural disasters. Vulnerabilities might allow an attacker to penetrate a network, allow him to gain access to controlling software and alter load conditions, to destabilize the grid in unpredictable ways" [10].

4.1 Cyber security threats

In 2007, CIA Analyst Tom Donahue revealed a power stoppage in a number of countries worldwide caused by cyber attacks. In March 2007, Idaho National Laboratory of the Department of Energy (DOE) carried out a physical experiment, which known as "Aurora Generator Test". The resulting damage was caused to a diesel generator through exploitation of a security flaw in its SCADA system. The so-called "Aurora Vulnerability" is still a security concern today [18]. The physical damage was caused to a diesel generator through exploitation of a security flaw in its SCADA system. Nowadays it is still considered an issue. Years later, during Russian-Georgian war there were cyber attacks to Georgian electric grid. It was discontinued during Russian Army's advance through the country. In 2009, the Wall Street journal reported about "cyberspies", which penetrated the USA's electric grid and insert software that could bring huge damage for the system [8].

4.2 Cyber security Vulnerabilities

One of the major vulnerability in cyber security is the complicated communication network. It makes power grid automation system weak to cyber attacks, which could divide into three categories [9]:

Component-wise

Automation devices (RTU[2], HMI[3]) help user interface to perform configuration or diagnostic functionality from a remote location. Hacker can possess remote access and bring following harm [9]: 1) mislead data for control system operator, 2) loss of service (hacker turn off the device), 3) damage to field equipment (by inaccurate fields of data on which the operator performs supervisory control operations).

Protocol-wise

Virtually all modern data communication protocols hold to a messaging protocol, because it is well documented and available in the public domain. The DNP[4] documented protocol is very common in electric utilities, by using the protocol hackers can do reverse engineering of the data acquisition protocol and hack it by implementing attack, called "man-in-the-middle". The consequences of this attack could be receiving misleading data by control center operator and in results we will get following [9]: 1) financial loss (e.g. attack leads to excess generation output), 2) safety vulnerability (line is energized while linemen are in the field servicing the line), 3) equipment damage (control commands are sent to the field resulting in overload conditions).

Topology-wise

Hackers can exploit network topology vulnerability. For example, a denial-of-service attack, by flooding an RTU with valid protocol messages, saturates the CPU[5] computational power, memory, bandwidth. it brings delay or slow down in real-time data exchange. It can cause failure of control center operators by making wrong decision.

5 Current security solutions

Nowadays a "gateway" security solution (Fig. 2) is widely used to protect power automation system from cyberattacks.

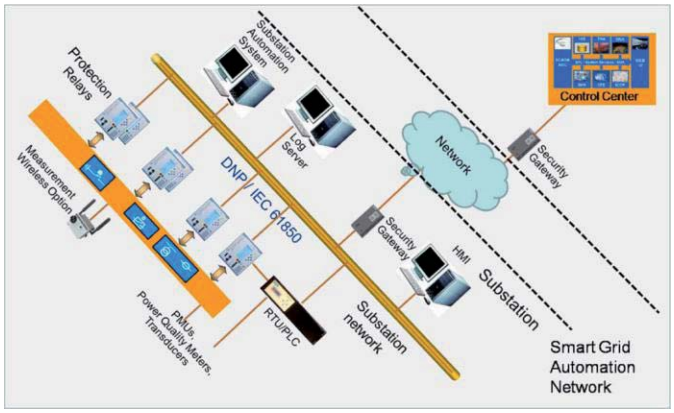


Fig. 2. Gateway security solution [9]

Two security gateways, one in a control center and the other in a substation, inspect their own incoming data packets. There are several security issues that could occur during normal operations of the automation network of the Smart Grid. For instance, an automation engineer, who needs to upgrade HMI or RTU, connects his laptop to the substation network; he could avoid the security gateway and change settings of protection relays, which he is not supposed to do. This could bring internal attack. Consider another example, substation holds thousands of electronic devices, the "white list" (who has access to which electronic devices), in the substation

[2] RTU Remote Terminal Unit
[3] HMI Human Machine System
[4] DNP Distributed Network Protocol
[5] CPU Central Processing Unit

gateway could be very long. It brings delay for some time-critical data packets [9].

Public-key infrastructure is another commonly used security solution in today's electrical grid. A public and a private key are issued by a certificate authority, who gives the private key to the owner and uses the public key as a part of a digital certificate that all others can access. The public key is used by others to encrypt a text, which is then decrypted by you with your private key. This way others are also able to authenticate themselves to you by using your private key to encrypt a digital certificate. This solution is widely used for e.g. Smart Metering devices/systems and home automation, authentication of users to applications such as smart card logon [17].

6 First solution: Use PKI

A. Metke and R. Eki [10] propose a PKI (public-key infrastructure) solution, but with some improvements.

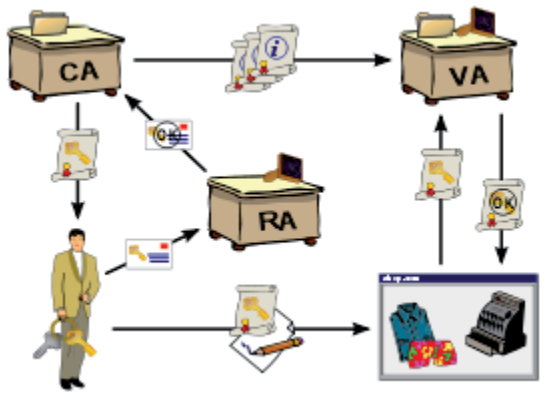


Fig. 3. Basic PKI procedures [10]

The basic procedures used in a public key infrastructure are shown in Fig. 3. The CS[6] sends a CSR[7] to the RA[8], who uses a vetting function to verify the validity of the CSR. If RA approves the request it is sent to the CA[9] with a stamp of approval. The CA issues a signed certificate and sends it to the CS, and the information about the certificate to a VA[10]. Next time that CS wants to access a secure resource, CS sends the certificate to the relying party (RP), who requests the status of the certificate from the VA.

A. Metke and R. Eki propose four technical elements added to the existing PKI:

- PKI standards
- automated trust anchor security
- certificate attributes
- Smart Grid PKI tools

Existing standards (such as x.509) provide a mechanism for defining naming conventions, certificate constraints, and certificate policies, leaving the level of detail on how to implement them to the organizations[11]. A. Metke and R. Eki suggest that standards should be added which determine security policies and PKI practices, such as how the CA is to be protected and how the CSR should be vetted. If these are determined by the industry they will be known to have levels of security that is desired.

CA's are able to provide CSR's to each others, which results in mutual trust or cross signing. Thus, an RP can trust a CA which is trusted by a CA that the RP already trusts. It is advised that each operator will keep its own hierarchy of CA's, and put their own TA[11] at the top (usually a top level CA), who will serve as the root of trust. In order for the RP to have a secure method to load and store this TA, A. Metke and R. Eki propose that each device manufacturer for the Smart Grid will supply each device with a factory preloaded manufactures certificate. This certificate constitutes make, model, serial number and a pre-provisioned TA certificate. When an operator purchases a device, the manufacturer will issue a TA transfer certificate, which will instruct the device to accept that operators root CA certificate as the new TA. The whole TA transfer process will all be automated by use of new Smart Grid PKI tools, in order to simplify the process.

When portions of the Smart Grid infrastructure become unreachable, it is essential for other portions to be able to function by giving the Smart Grid devices a means to authenticate and determine each other's authorization status. This is proposed to be performed by adding policy attributes to the certificates of each Smart Grid device. Each CS will be periodically supplied with the status of its own certificate, which it will provide to the RP when authentication is requested. The RP determines, based on local policy, if the status is new enough to accept. In addition, all CS's will be loaded with the chain of certificates between themselves and their TA. They will be able to select chains of certificates between the subjects' TA and other agencies TA which is trusted by means of cross signing. New Smart Grid tools would manage these chains of certificates as well as ensuring the proper sets are given to devices.

Trusted computing is the final step in this proposed solution. Two categories of devices, embedded systems (designed to perform a specific task or set of tasks, e.g. cable television set-top box) and general purpose computer systems (intended to support third party software, e.g. PC) are considered separately. In order to protect embedded systems against installation of malware, the manufacturer must implement one of many standard models available [12] to secure its software development. In addition a secure software upgrade solution must be provided, using PKI. To address the security concern for general purpose computer systems, strict code signing standards by Smart Grid suppliers and operators is proposed. The Trusted Computing Group [13] has put forth mechanisms for enforcing such standards.

7 Second solution: An integrated security framework

In this section we consider an integrated security framework, which is given by D. Wei et. al. [9], comprises of three layered architecture: power layer at the bottom, automation and control layer in the center, and the security layer on top. Power grid processes are controlled and monitored by automation and control layer, while the security layer provides security functionality. Moreover, the proposed security solution could replace current "gateway" security solution by a "security service proxy" solution, which shown in Fig. 4, has three key security subsystems: security agent, security switch and security manager. The first two are security enforcement devices and run as a security service proxies. Meanwhile, the security manager runs as a security management device in the substation or in the control center.

[6] CS Certificate Subject
[7] CSR Certificate Signing Request
[8] RA Registration Authority
[9] CA Certificate Authority
[10] VA Validation Authority
[11] TA Trust Anchor

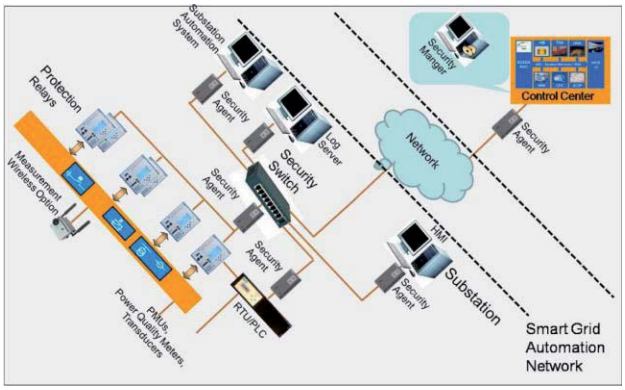


Fig. 4. Security service proxy solution [9]

The integrated security framework operates on three hierarchical levels: device level, network and operation levels (Fig. 5). In the device level the Security Agents protect electronic devices, such as RTU[12] and IED[13]. The Network level protects communication bandwidth, and in the operation level the Security Manager controls and operates security policies. The Security manager has multiple numbers of secure TCP/IP connections to each Security Agent and managed Security Switch.

The three key security subsystems are as follows:

- **Security Agent**
  The security agent provides security to the edges of system and applies to both wired and wireless devices, and depending on the layer of the control hierarchy they could be either firmware or software. The agents will be less intelligent in the device level, having responsibilities, such as to log and to report containing simple rules and decision making capabilities. On the contrary, they will be more intelligent in substation level, holding complex rules identifying and detecting intrusive events and activities.
  According to D. Wei et. al. [9] agents possess next functionalities:
  - to translate between different protocols;
  - to acquire and run the latest vulnerability patches from its security manager;
  - to collect data traffic patterns, system log data and report to the security manager;
  - to analyze traffic and access patterns with varying complexity depending on the hierarchical layer;
  - to run host-based intrusion detection;
  - to acquire access control policies from the security manager and enforce them;
  - to detect and send alarm messages to the security manager and designated devices, such as HMI;
  - to encrypt and decrypt exchanged data.
- **Security Switch**
  The bandwidth is protected by managed security switches, which are used across the automation network. The switches working as network devices prioritize data packets and connect controllers, servers in the control center and substation, also RTUs, HMIs.
  There are managed security switches' functionalities proposed by D. Wei et. al. [9]:
  - to separate external and internal networks, trusted and non-trusted domains;
  - to run as a Dynamic Host Configuration Protocol (DHCP) server;
  - to run network address translation and network port address translation (NAT/NPAT) and to hide the internal networks;
  - to acquire bandwidth allocation patterns and data prioritization patterns from the security manager;
  - to separate data according to prioritization patterns, including operation data, log data, trace data, and engineering data;
  - to run simple network-based intrusion detection.
- **Security Manager**
  Security managers, which have a graphical user interface (GUI), are located in the automation network and protected by existing IT security solutions. They can connect (directly and indirectly) to the security switches through automation networks. Security managers have following functionalities [9]:
  - to collect security agent information;
  - to acquire vulnerability patches from a vendor's server and download them to the corresponding agents;
  - to collect alarms and events;
  - to generate access control policies based on collected data and download them to agents;
  - to collect data traffic pattern and performance matrix from agents;
  - to work as an authentication, authorization, and accounting server, validating user identifications and passwords, authorizing user access rights (monitor, modify data), and recording user changes to controllers;
  - to run complex intrusion detection algorithms at control network levels.

  The security manager resides exactly in the center of automation network and controls how security functions are performed.

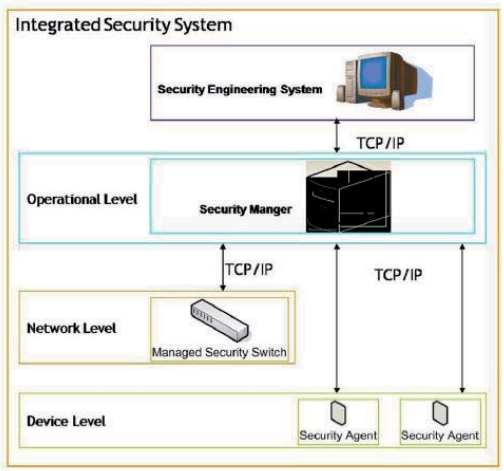


Fig. 5. Multilayered integrated security framework [9]

8 Conclusion

This paper first introduces the conceptual model of Smart Grid and considers 7 major domains identified by NIST. Moreover, it gives an explanation about important components, such as Demand management, Distributed Electricity Generation, Transmission and Distribution Grid Management. We discuss the major challenges and two chosen solutions to protect the Smart Grid against cyber attacks.

The first solution, PKI with trusted computing is rightfully criticized by H. Khurana et. al. [6] due to lack of scalability. They argue that an organization, which uses PKI system to provide X.509 certificates, will need one support staff for every 1,000 user

[12] RTU Remote Terminal Unit.
[13] IED Intelligent Electronic Device

certificates. If this ratio is transferred over to a utility company who has 5,5 million smart meters, that would mean that there would be a need for a staff of 500 people, just to keep up the certificates. They also mention time and processing required to update the certificates as a negative point to this solution. With the added standards, this cost could be lowered since each organization will not have to independently research PKI to determine appropriate policies and practices. Never the less, this solution is overall costly and involves many parties to work, in addition the human part regarding certificate signing and human error is not taken into account. Earlier this year the 2012 budget of the Dutch government was accidentally published on a webpage that was not encrypted [15].

Our choice is the second solution; it is based on a solution that is currently widely in use, with some changes. It is based on a three layered architecture with a security service proxy. The integrated security framework is designed to protect the power grid automation system from cyberattacks, as well as from external and/or internal networked sources. It gives desirable results in terms of scalability, modularity and manageability. The developed prototype of the security system has undergone during five-day uninterrupted testing at Idaho National Laboratory. It proved that the system manages to mitigate common vulnerability of automation system of power grid without compromising control and signal data in terms of timely availability. However, it is still needs testing in a real power grid environment with 24/7 operation. It is our recommendation that a closer look needs to be taken towards the human side of security, which is a very interesting subject in its own.

ACKNOWLEDGEMENTS

The authors would like to thank Professor Marco Aiello for his valuable reviews and inputs to our work. In addition we would like to extend our gratitude to Ruurd Moelker and Werner Buck, who reviewed the paper and gave us extremely productive criticism for rewrite.

REFERENCES

[1] "Smart Grid", Wikipedia, http://en.wikipedia.org/wiki/Smart_grid, [accessed 19 March 2012]

[2] A. Cavoukian, J. Polonetsky, C. Wolf, "Smart Privacy for the Smart Grid: embedding privacy into the design of electricity conservation", IDIS (2010) 3:275–294, DOI 10.1007/s12394-010-0046-y, 20 April 2010

[3] S. Gorman (8. April 2009) The Wall Street Journal, "Electricity Grid in U.S. Penetrated By Spies". http://online.wsj.com/article/SB123914805204099085.html, [accessed 19 March 2012]

[4] IEEE Smart Grid, Smart Grid conceptual model, http://smartgrid.ieee.org/ieee-smart-grid/smart-grid-conceptual-model [accessed 20 March 2012]

[5] W. Frye, "Transforming the Electricity System to Meet Future Demand and Reduce Greenhouse Gas Emissions", Cisco Internet Business Solutions Group, November 2008

[6] H. Khurana, M. Hadley, N. Lu, D. A. Frincke, "Smart-Grid Security Issues", IEEE Security and Privacy, January/February 2010

[7] "Cyberattack", IT Law, http://itlaw.wikia.com/wiki/Cyberattack, [accessed 17 March 2012]

[8] C. Ebinger, K.Massy, " Software and hard targets: Enhancing Smart Grid cyber security in the age of information warfare", Policy Brief 11-01, Februarry 2011

[9] Dong Wei, Yan Lu, M. Jafari, Member, P. Skare, K. Rohde (DECEMBER 2011), "Protecting Smart Grid Automation Systems Against Cyberattacks ", IEEE TRANSACTIONS ON SMART GRID, VOL. 2, NO. 4,

[10] A. R. Metke, Randy L. Eki, "Security Technology for Smart Grid Networks", IEEE Transactions on Smart Grid, vol. 1, no. 1, June 2010

[11] X.50, http://en.wikipedia.org/wiki/X.509, [accessed 20 April 2012]

[12] N. Davis, Secure software development life cycle processes Software Eng. Inst., Carnegie Mellon Univ., 2009.

[13] D. Challener et al., A Practical Guide to Trusted Computing . Upper Saddle River, NJ: IBM Press.

[14] Publick-key infrastructure, http://en.wikipedia.org/wiki/Public-key_infrastructure, [accessed 21 April 2012]

[15] ABC NEWS, "Dutch 2012 budget accidentally posted online", http://abcnews.go.com/Technology/dutch-2012-budget-accidentally-posted-online/comments?type=story&id=14527664#.T5QoPqtzUvk, [accessed 21 April 2012]

[16] "Distributed generation", http://en.wikipedia.org/wiki/Distributed_generation, [accessed 22 April 2012]

[17] Software and hard targets: Enhancing smart grid cyber security in the age of information warfare, http://www.brookings.edu/~/media/Files/rc/papers/2011/02_smart_grid_ebinger/02_smart_grid_ebinger.pdf , [accessed 22 May 2012]

HTML5 data visualization capabilities of mobile devices

Ruurd R. Moelker and Wilco E. Wijbrandi

Abstract— Data visualization allows scientists, engineers and medical personal to better understand datasets. With increasing popularity and performance of mobile devices, visualization on mobile devices is more interesting than ever. However, development for mobile devices is very resource consuming because of a heavily fragmented market. The development process is significantly simplified by abstracting from these differences by using web applications. And while data visualization on mobile devices is extensively studied, visualization in conjunction with web applications is a relatively unexplored aspect. Furthermore, while many mobile browsers support recent techniques like HTML5, WebGL and WebSockets, in practice integration is often still limited.
In our study we have investigated to which extent web applications are capable of 3D data visualization on mobile devices.
One aspect researched is the current support for HTML5 techniques required for data visualization. A benchmark is performed comparing the graphical performance between native applications built with OpenGL and web applications built with WebGL. Finally a comparison between the general performance of JavaScript and the mobile devices native languages is done.
Our study shows that the majority of the currently used mobile browsers do not yet support new techniques required for data visualization on mobile devices. Furthermore the performance of JavaScript and WebGL is significantly lower than their native counterparts. This makes computationally intense mobile data visualization using web applications challenging, since performance already is limited on mobile devices.

Index Terms—HTML5, WebGL, mobile data visualization, mobile web application.

1 Introduction

Mobile data visualization allows visualizations to be shown to a wide audience using a convenient platform. Data visualization applications vary between showing volumetric medical datasets, molecules and software visualization in an intuitive manner (e.g. figures 1 and 2). With increasing mobile use and performance, mobile application development is more interesting than ever. Visualizations can be more accessible and user friendly on mobile devices.

In the field of scientific data visualization often large datasets are displayed, such as 3D scans or graphs. While powerful desktop system generally can represent this data for interactive purposes, mobile devices (e.g. phones, tablets and netbooks) have trouble with this. Mobile systems have intrinsic limitations such as CPU and GPU performance, limited and varying screen sizes and limited bandwidth. Many of these problems have been solved by offloading computations to another device, compressing and layering communication [23; 18; 22] and easing navigation of large visualization [16].

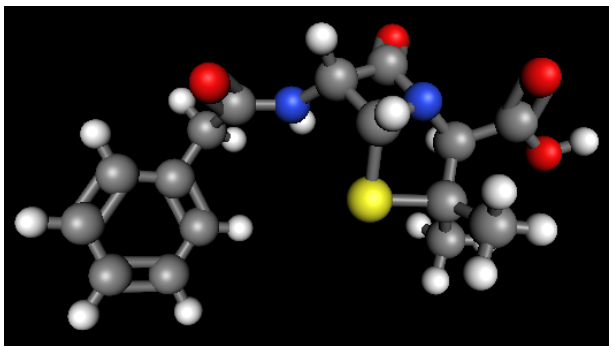


Figure 1. Example of the visualization of a molecule using a web application [8].

- *Ruurd R. Moelker is a Computing Science student at the University of Groningen, E-mail: r.r.moelker@student.rug.nl.*
- *Wilco E. Wijbrandi is a Computing Science student at the University of Groningen, E-mail: wilco@wijbrandi.nl.*

One of the drawbacks of developing applications for mobile devices is the wide diversity of mobile hardware and software. Therefore, reaching a large audience by developing an application for multiple mobile platforms requires significantly more development effort and money. With new standards like *HyperText Markup Language 5* (HTML5) [6] and *Cascading Style Sheets 3* (CSS3) [2] as well as *Web Graphics Library* (WebGL) [11] and WebSockets [12], rich web applications can be developed. Building web applications can reduce development time because the HTML5 techniques abstract from the underlying hardware and software. However, little research is done on the technological support for these techniques on mobile devices.

This study explores mobile support for these new standards. The performance impact of a web application versus a native application is investigated. Both 3D visualization as well as general processing performance with JavaScript is discussed.

1.1 Overview

First related work on mobile data visualization and web application technologies is discussed in section 2. Next the current development of web application techniques is discussed in section 3. Test setup and methods are presented in section 4. Results of the technologies supported, native versus web application benchmarks and finally JavaScript performance are presented in sections 5, 6 and 7. Finally conclusions are drawn and future work is discussed in sections 8 and 9.

2 Related work

Over the last years much progress has been made in the field of data visualization on mobile devices. The advances address many of the inherent constraints of mobile devices such as screen size, processing power and limited bandwidth. Previous research on web application use on mobile devices was done concerning mobile versus native applications, input management and even entire HTML5 based operating systems have emerged.

Data visualization Data visualization on mobile devices provide unique challenges. Zhou et al. have developed a method to visualize and transfer large volumetric datasets over a limited bandwidth connection [23]. The proposed method is able to provide a preview of the volume. As more data is transferred, rendering accuracy is increased. This is done by first preprocessing the volume data before sending it to the client mobile device. The client receives the smaller representations of the dataset first and with increasing detail until finally the entire voxel model has been transferred. Like the work by Luke and

Hansen [19], this method by Zhou et al. provides a faster way to visualize computational intense volumetric data by rendering either the point cloud, isosurfaces or voxel model based on the capabilities of the device.

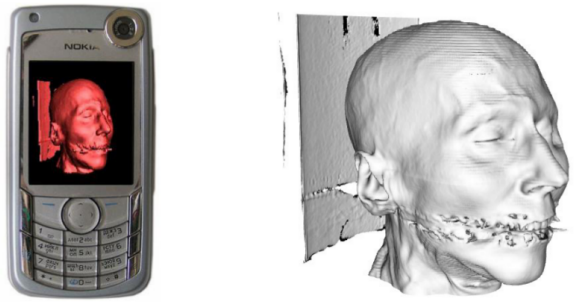


Figure 2. Data visualization on a mobile [23].

Luke and Hansen discuss a remote data rending method that also use multiple rendering steps [19]. Their multifaceted approach scales based on client capabilities. A low performance client and high bandwidth connection causes their framework to display renderings fully made on the server. If on the other hand the client has good rendering capabilities, stepwise rendering methods are done by the client similar to the ones described in the previous paragraph.

A method researched by Lippert and Gross allows for yet another method of volumetric data rendering in a client server setup [18]. Their work decomposes volumetric data to the frequency domain using wavelets. Transparent textures are then computed which at run time can quickly be accumulated by graphics hardware to render the volumetric data. Because the decomposition needs only to be done once, this can be done up front on a server machine. The client machine then only needs to be capable of accumulating transparent textures which should be possible with simple hardware. A downside of the wavelet decomposition method is that complex visualizations, such as self-occulsion and nonlinear shading models, are not directly available. The work of Westenberg and Roerdink extend on this work by providing a faster method at the expense of very little rendering accuracy and a heavier usage of memory [22].

An application of the discussed remote processing techniques is suggested by Granot et al. [15]. The authors argue that by only visualizing medical data on a mobile device and performing the processing of the data elsewhere, relatively cheap imaging tools can be provided to physicians in developing countries.

Web applications While most inherent problems to mobile devices have been addressed in recent years, a new issue has emerged. The mobile device market is highly fragmented with many different devices, specifications and operating systems. The use of web application as a solution to this hardware and software segmentation issue has been researched before. An article by Charland and Leroux underlines the time/cost effort and knowledge needs of developing for the fragmented mobile market [14]. The article also states the importance of application performance yet is limited in conclusions on performance of native versus web applications. One of the focuses of this study is the performance comparison of both 3D drawing and processing power of web applications.

Like the previously discussed remote data visualization methods a similar framework is researched for web applications [21]. Wessels et al. use HTML5 techniques to visualize large dataset on mobile devices. The work shows that a web application can handle large datasets, provide user interaction and is able to run on a range of mobile devices. WebSockets are utilized for asynchronous sustained connections for sending interaction commands and receive visualization data. Visualizations are drawn on a canvas HTML5 element. Our study also investigates the canvas element and WebSockets, as well as WebGL canvas elements which are needed for more intense visualizations.

A project which illustrates that HTML5 techniques are on the rise is Google's operating system *Chrome OS*, which exclusively works with web applications [3]. Chrome OS extends the cross platform capabilities of web technologies with the use of cloud storage in order to provide a more seamless experience between devices.

3 Web applications

In the last couple of years web applications have become very popular. A good example is Google Docs: an office suite which runs entirely inside the browser. Documents are stored on Google's servers, and you can use that application as well as access your own documents from anywhere where there is an Internet connection and a modern browser available.

Web applications are dynamic web pages which behave like a normal application. The application is a combination of code that runs on a web server and codes that runs inside the client's browser. This means that whenever a web application is opened, part of the application needs to be downloaded in advance. This could be a drawback for mobile devices which have limited bandwidth.

3.1 HTML5 technologies

Where in the early days of the Internet web pages only consisted of an HTML document, nowadays we use a whole *stack* of technologies.

HTML is the markup language for web pages. It describes the structure of the document. HTML describes elements of the document using tags, just like *Extensible Markup Language* (XML) does.

CSS is a style sheet language designed to describe the appearance and layout of document described in a markup language like HTML or XML. By separating document structure from presentation, it is easy to create a generic look and feel for multiple documents and to apply different look and feels for a single document. It is for example common to have a separate style sheet for displaying a web page on a computer screen and printing a web page.

JavaScript is a scripting language which is typically (but not exclusively) used to create interactive web pages. It is a multi paradigm language supporting object-oriented, imperative and functional programming styles. JavaScript is formalized and standardized as *ECMAScript* [5].

Scalable Vector Graphics (SVG) [10] is a XML-based format for describing vector images. SVG drawings can be modifiedor animated using JavaScript. It provides a rich set of event handlers which make it easy to make interactive graphics.

With the newest version of HTML, HTML5, the standard goes beyond the description of the structure of a document. It adds new features like dragging and dropping elements, working with the geolocation of the user, storing data in a key-value database at the client and adding native support for showing video's.

Strictly speaking, while HTML5 is merely a standard for describing documents, the term is often used to describe the whole stack of technologies. HTML5 connects all these technologies and allows the creation of functional web applications. Almost every current desktop or mobile application could be build as a web application, without the need for additional plug-ins.

3.2 WebGL

When we want to do data visualization on mobile devices through HTML5 techniques we need a fast performing, low-level API for rendering graphics on the Graphics Processing Unit. OpenGL ES 2.0 [9] currently seems to be the most widely used API on mobile devices, since it is supported by popular platforms like iOS, Android and Blackberry OS.

Web applications cannot access the low-level OpenGL ES 2.0 API directly. WebGL however is a relatively new API which is available for web applications. It is based on the OpenGL ES 2.0 standard. It can be accessed using JavaScript and it can use the HTML canvas element to draw graphics inside a website. Since WebGL is based on

OpenGL ES 2.0 it is relatively easy to port native applications to web applications and vica versa.

3.3 WebSockets

A technique which could be a key technology for client-server visualization on mobile devices is WebSockets. Normally, web applications can only communicate with a server through HTTP requests. HTTP is a stateless protocol which only allows a client to request documents from a server. A WebSocket provides a bi-directional full duplex communication between a server and a client using a standard TCP socket [17]. This way data can be streamed efficiently from a server to a client.

4 METHODOLOGY

To test the visualization capabilities of mobile devices several experiments are performed. First the method for creating a supported HTML5 feature list for different browsers and platforms is presented. Next, to test the graphics visualization capabilities of web applications a benchmark for three different devices is discussed. Finally, JavaScript performance is compared to native programming language performance.

4.1 Browser technology support

An overview of supported web application techniques for popular browsers is made for several platforms. For the two most dominant mobile operating systems, iOS and Android, all mobile browsers with a market share of more than 5% are tested. An exception to this rule is Firefox for Android, which only has a market share of 0.03%. The browser is included because it is the browser for Android supporting WebGL. While this study focuses on mobile devices, desktop and laptop systems are included as a frame of reference.

Global usage data for the month February 2012 is used throughout the tests. Mobile browser usage is derived from browser traffic data from `http://netmarketshare.com` as a substitute for installed browser percentages [7]. For desktop operating systems `http://www.w3schools.com` is used to retrieve browser traffic data [1]. Actual feature support data originate from all the HTML5 elements, WebGL and WebSockets links found on `http://caniuse.com/` [13]. Detailed feature support of Firefox for Android was not available and of is retrieved by testing support for HTML5 elements and WebGL on the browser on Android Samsung Galaxy S II (test system 3).

WebGL support is generally distinguished into either not supported, software emulated or hardware accelerated. For this test WebGL is considered to be supported when either software or hardware rendering is available.

Many rendering engines support HTML5 but do not fully support the entire set of components described by the still developing HTML5 standard. This research focuses on support for the elements: offline caching, the canvas element and the use of text in the canvas element. The canvas element could potentially be used as a fallback for when WebGL is not supported. This subset of HTML5 elements is needed to at least display visualization and text in the browser without a persistent Internet connection. WebSockets support is also investigated because it is useful for client server visualization.

4.2 Graphics performance comparison

In this test we make a comparison of the visualization performance between a native application and a web application. We did this for two separate platforms: a laptop computer and a smartphone. For reference we also included a desktop computer.

The benchmark visualizes a variable number of rotating cubes. This test stresses the graphical hardware and is relatively small and therefore easy to port to different platforms. Every face of each cube has a separate color. The cubes and colors are stored using *Vertex Buffer Objects*, so they can be rendered directly from the video device memory. The cubes are each rotated individually. The visualization can displays n cubes in a grid with $\sqrt{n}$ columns. The grid is scaled automatically in such a way that all cubes are visible. For consistency, we used a constant resolution of 480 by 762 pixels. This should have little consequences on the results since this benchmark does not stress fragment/pixel shading.

For every platform we have implemented the benchmark in the normal fashion for that particular platform. For the web application we have implemented this visualization using HTML5, JavaScript and WebGL. For the native application on the Linux computers we have used *GLUT* (the OpenGL Utility Toolkit), a lightweight cross-platform framework for developing OpenGL programs in the C programming language. For the Android platform we have developed a native application with the Android API in the Java Programming language. In the current version of Android, the application runs inside the Dalvik JVM, which is optimized for mobile devices and uses just-in-time compilation.

We run our benchmark on three different machines: A desktop computer as a reference, a MacBook as a mobile computer and an Android Smartphone. Unfortunately, we are not able to test an iOS device since it has no browser available which supports WebGL. In our opinion, these devices are representative for the platforms.

System 1: Desktop Debian Linux operating system, Intel Core 2 Duo E6550 2.3GHz CPU, GeForce 8400 GS GPU, OpenGL 2.1, Google Chrome 6.0 browser.

System 2: MacBook Ubuntu Linux operating system, Intel Core 2 Duo P8600 2.4GHz CPU, GeForce 320M GPU, OpenGL 3.2, Google Chrome 17 browser.

System 3: Samsung Galaxy S II Android 2.3.5 operating system, 1.2 GHz dual-core ARM CPU, ARM Mali 400 GPU, OpenGL ES 2.0, Firefox for Android 10 browser.

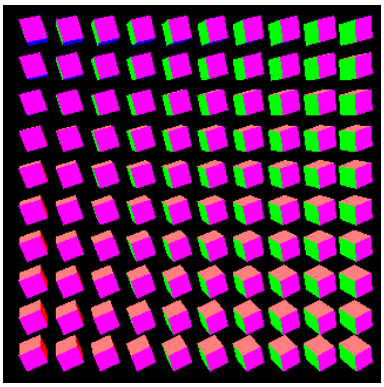


Figure 3. Benchmark application of 10x10 rotating cubes.

4.3 JavaScript performance comparison

Web application performance depends on a combination of several factors: the JavaScript engine, the operating system, hardware and the already discussed graphics capabilities. To compare performance between web applications and native applications this test was limited to the influence of the used programming language.

Web application programming JavaScript is compared to the most prevalent platform development languages in use, i.e. Objective-C for iOS and Java for Android. Benchmark data is retrieved from the open source benchmark project called “The Computer Language Benchmarks Game” [4]. The benchmarks available from this source do not provide Objective-C coding. As a replacement the C code benchmarks are used. Although Objective-C is based on C, it adds some features which decrease performance. It should be noted that Objective-C will in general be slower than C. All results are for a single test machine running Ubuntu x86 operating system on an Intel Q6600 (single core).

5 Browser technology support

Support for HTML5, WebGL and WebSockets varies between browsers. For web applications development to be a good alternative to native applications, a large portion of (mobile) devices must be able to run and display all the elements of web applications. In this section the features supported by recent mobile and desktop browsers are evaluated. Finally an exploration is made of the portion of devices that can run web applications with their currently installed software.

5.1 Current implemented features

Table 1 shows the supported techniques for the latest version of popular browsers and for several platforms. The feature support table shows that the latest browsers on desktop/laptop operating systems can handle fully fledged (HTML5 and WebGL) web applications.

The two most prevalent browsers for mobile systems, Safari for iOS and the Android browser also have limited support for the tested techniques. HTML5 elements are supported, but WebGL is not. WebGL is however available for iAd, Apple's platform for mobile advertising on iOS. There is a workaround to enable WebGL for web applications. This requires a developer to disguise the application as an advertisement [20]. This is simply too risky for developers in the authors view because an application may easily be rejected or removed from the application store.

5.2 Market adoption

All systems discussed, except iOS, support HTML5 and WebGL for at least one browser. It would be dangerous to assume that users always have the latest version of a browser installed. Therefore an understanding is needed of what portion of web applications will work out of the box on mobile devices with their current OS and browsers versions.

Table 2 shows the web technology (simple canvas and WebGL) support for both desktop and mobile operating systems. Wherever available browser traffic statistics of supporting versions have been used to only count traffic of versions which support the mentioned aspects. Unfortunately such version data was not found for the mobile operating systems. As mentioned before, the reader should pay attention that, because of the lack of currently installed browser data, browser Internet traffic data is used. This is only an estimation of actual installed browsers on mobile devices.

Browser\Element		Canvas&text	WebGL
Desktop	Firefox	35.6%	31.3%
	Chrome	36.3%	35.9%
	Opera	1.7%	0.0%
	IE	5.7%	0.0%
	Total	**78.3%**	**67.2%**
Mobile[1]	Android	18.62%	0.0%
	Firefox	0.03%	0.03%
	Safari	61.19%	0.0%
	Total	**79.84%**	**0.03%**

Table 2. Percentages of worldwide browser traffic that support at least the canvas and text element or those elements and WebGL. 1) No version data was available for mobile browsers so traffic of all versions is shown.

Table 2 provides an estimate of what percentages of desktop and mobile OS users can view web applications without needing to update software or install new applications. Basic support for canvas drawing is available in most mobile devices out there today with 79.84% of browser traffic supporting these features. This is a very wide coverage, as around the same percentage of desktop operating system users are able to run these basic visualization using web applications.

On the other hand, there is currently almost no support for WebGL on mobile devices. Only 0.03% of browser traffic accounts for WebGL supporting browsers which if directly linked to browser availability would mean almost no mobile device is able to display WebGL. A critical note here is that the WebGL supporting browser Firefox for Android is known to be a sluggish browser, so in this case Firefox for Android usage may poorly reflect whether it is available.

6 Graphics performance comparison

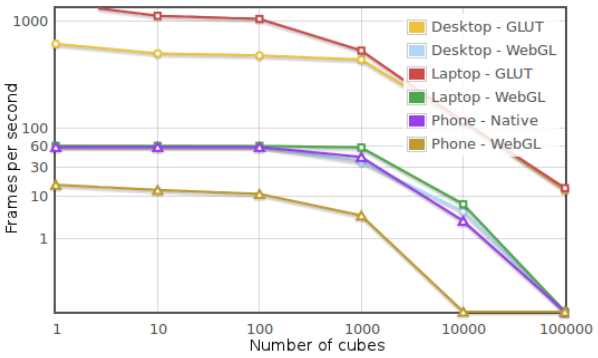


Figure 4. Graphical performance of native applications and web application on several platforms. Scales are non-linear.

Table 3 and Figure 4 show the result of the benchmark described in section 4.2. The first thing we notice is that the performance of the desktop computer and the laptop computer is very high compared to that of the smartphone.

The smartphone seems to be limited to 60 frames per second. This is probably done by the Android framework to save power. More frames per second will probably not be noticed by human eyes. Rendering less frames saves energy.

The Google Chrome browser's WebGL implementation also seems to be limited to 60 frames per second. Although power consumption and heat are less of an issue on desktop and laptop computers, it is still important. A quick test shows that the Firefox browser also limits WebGL to 60 frames per second.

As seen in Table 3, when ignoring the cases where a frame rate limit applies (1,000 cubes or more), the performance of the WebGL implementation is always lower than 10% of the native implementation.

7 JavaScript performance comparison

Table 4 shows several benchmarks for different programming languages. The table is an indicator for general processing performance of web language JavaScript versus native languages for the mobile platforms.

A first observation from the table is that both Java and C benchmarks perform significantly better than JavaScript benchmarks. On average Java is almost nine time faster and C is over fifteen times faster. More generally speaking and translating the results to practice, this means that JavaScript applications are on average ten times slower than their native coded counterparts. This is quite a large slowdown from native code and means that increased developer effort must be spent on program speed up, or that a web application simply will run/respond slower.

When interpreting the results one should take into account that JavaScript and Java are interpreted languages. One may notice that the V8 JavaScript engine compiles the script to native code before execution. However, because of the interpreted nature of the JavaScript, many of the optimizations which could be applied to a normal compiled language, can not necessarily be applied when using JavaScript.

The pidigits benchmark stands out as the benchmark where JavaScript performs poor compared to both Java and C. The number PI is calculated using a step-by-step algorithm in this benchmark. No cause for this highly deviating measurement has been found. A likely cause is a sub optimal implementation, yet this idea has not been further explored.

	OS	Linux & Windows			Windows	Android		iOS	Android & iOS
Technique	Component\Browser	Firefox 10	Chrome 17	Opera 11.6	IE 9	Android browser 4	Firefox for Android 10	Safari 5	Opera Mini
HTML 5	Offline	✓	✓	✓	x	✓	✓	✓	x
	Canvas	✓	✓	✓	✓	✓	✓	✓	✓
	Canvas text API	✓	✓	✓	✓	✓	✓	✓	x
WebGL		✓	✓	x	x	x	✓	x[1]	x
WebSockets		✓	✓	✓[2]	x	x	✓	✓[2]	x
Layout engine		Gecko	WebKit	Presto	Trident	WebKit	Gecko	WebKit	Presto

Table 1. Web application technology support for different browsers and devices. Browsers sorted by popularity per platform. 1) only available for use in iAds [20]. 2) Browsers offer partial WebSocket support because of an old protocol or feature is disabled by default.

Platform	Desktop			Laptop			Smartphone		
Number of cubes	GLUT	WebGL	%	GLUT	WebGL	%	Android app	WebGL	%
1	670	60	8.96%	1282	61	4.76%	59	16	27.12%
10	558	60	10.75%	1102	61	5.54%	59	13	22.03%
100	537	60	11.17%	1046	61	5.83%	59	11	18.64%
1,000	495	37	7.47%	590	58	9.83%	43	4	9.30%
10,000	122	5	4.10%	122	7	5.74%	3	0	0.00%
100,000	13	0	0.00%	14	0	0.00%	0	0	–

Table 3. Measured number of frames per second on several platforms with a native application and a web application. Values are rounded down. The percentage column indicates the percentage of the frames per second of WebGL compared to the native implementation.

The regex-dna benchmark is the only benchmark which is executed faster in JavaScript than in the other languages. This benchmark performs several regular expressions on the output of another benchmark. Without code inspection the higher performance of JavaScript can be attributed to the fact that JavaScript has a specialized data type for regular expressions. With high probability the V8 engine contains at least some optimizations for performing these regular expressions.

8 Conclusion

In this paper, we have investigated to which extent web application techniques are capable of data visualization on mobile devices. We have explored feature support for different browsers, platforms and devices. A WebGL versus native OpenGL implementation is shown for several devices. Finally a performance comparison of web application coding as opposed to native coding was done.

WebGL visualization is available for all popular mobile devices, except iOS based devices. Simpler data visualization on HTML5 canvas elements is however possible on all of these devices. Many users need to install additional software to run WebGL applications, while canvas elements are directly available for most mobile browser users.

While WebGL is a good way to create fast 2-dimensional and 3-dimensional graphics inside the browser, it is still significantly slower than native OpenGL applications. This holds true for both desktop computers as our Android Smartphone.

JavaScript has come a long way with performance. New JavaScript engines have achieved an incredible amount of performance increase in the last couple of years. However, algorithms implemented with JavaScript and executed by the modern V8 JavaScript engine are still significantly slower than algorithms written in other programming languages.

Generally, applications written in JavaScript are around ten times slower than native coded applications. This is quite a significant performance loss when using web applications. As a consequence developers either need to make concessions on the program functionality or spend additional effort to compensate the performance hit.

Web applications could save development effort if an application has to run on several platforms. In addition, web applications are easy to deploy, automatically store documents and files in a central database and allow easy remote processing. However, we think that neither HTML5 techniques nor the market is ready for it. Many of the latest browsers support the needed HTML5 techniques. The currently most used browsers however do not yet support WebGL, a key technique for mobile data visualization. Other techniques that might be important, such as canvas and WebSockets, also are not yet widespread. Furthermore, the performance of both WebGL and JavaScript is significantly lower than that of native applications. This is a problem, since resources are already an issue on mobile devices. Mobile devices don't have the processing power that desktop and laptop computers have. The most pressing concern is that the leading mobile operating systems either do not support WebGL, or do not do so with the pre-installed browser.

Data visualization using web application on mobile devices simplifies development when multiple platforms must be supported. The current performance and support however is just too low to become a feasible option for complex visualizations.

9 Future work

The performed research suggests a number of valuable follow-up efforts.

First, much of the support table input can be refined. The mobile support statistics can greatly benefit from browser version usage data. Also the conclusions are currently based on browser traffic data which may not be a very well representation of actually installed applications on mobile devices. With such additional information stronger conclusions can be drawn.

Next, the general processing benchmark assumes that Objective-C performance is similar to standard C. A better approach would simply use Objective-C for a benchmark. Also, since WebGL only runs on

Language	JavaScript V8	Java 7 - server JRE		C - Gnu CC	
Benchmark	CPU timing (s)	CPU timing (s)	%	CPU timing (s)	%
binary-trees	36.07	14.04	257%	12.14	297%
fannkuch-redux	80.45	71.26	114%	45.26	118%
fasta	18.97	5.00	379%	4.92	386%
k-nucleotide	272.01	33.43	814%	38.44	708%
n-body	79.67	24.22	329%	20.45	390%
pidigits	268.51	5.54	4847%	2.74	9800%
regex-dna	4.18	23.44	18%	5.82	72%
reverse-complement	16.56	1.88	881%	1.07	1548%
spectral-norm	33.16	16.13	206%	10.22	324%
Average	90.0	21.66	871%	15.7	1522%

Table 4. JavaScript compared to native code performance on range of benchmarks. Percentages are running times of native code compared to JavaScript timings rounded off.

Firefox for Android, the JavaScript performance on that browser is of more relevance than the currently compared Chrome's V8 JavaScript engine.

Finally, a web application performance may be measured using other criteria such as startup time. Furthermore other elements can be researched such as bandwidth utilization over WebSockets compared to native sockets.

ACKNOWLEDGEMENTS

The authors wish to thank prof. dr. Jos B.T.M. Roerdink as well as Katsantonis Fotis and Christian Manteuffel for reviewing this paper.

REFERENCES

[1] Browser statistics (n.d.). Retrieved March 12, 2012, from http://www.w3schools.com/browsers/browsers_stats.asp.

[2] Cascading style sheets (css) snapshot 2010 (2011, may 12). Retrieved April 10, 2012, from http://www.w3.org/TR/CSS/.

[3] Chromium os (n.d.). Retrieved March 12, 2012, from http://www.chromium.org/chromium-os.

[4] The computer language benchmarks game (2012, march 4). Retrieved March 19, 2012, from http://shootout.alioth.debian.org/.

[5] Ecmascript language specification (2011, may 5). Retrieved April 10, 2012, from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33835.

[6] Html5 (2012, march 29). Retrieved April 10, 2012, from http://www.w3.org/TR/html5/.

[7] Mobile/tablet browser market share (2012, march 16)). Retrieved March 16, 2012, from http://netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1&qptimeframe=M&qpsp=157&qpnp=1.

[8] Molgrabber 3d (n.d.). Retrieved April 11, 2012, from http://web.chemdoodle.com/demos/molgrabber-3d.

[9] Opengl es - the standard for embedded accelerated 3d graphics (n.d.). Retrieved April 10, 2012, from http://www.khronos.org/opengles/.

[10] Scalable vector graphics (svg) 1.1 (second edition) (2011, august 16). Retrieved April 10, 2012, from http://www.w3.org/TR/SVG/.

[11] Webgl specification (2011, february 10). Retrieved April 10, 2012, from https://www.khronos.org/registry/webgl/specs/1.0/.

[12] The websocket api (2012, april 2). Retrieved April 10, 2012, from http://dev.w3.org/html5/websockets/.

[13] When can i use... (2012, march 14)). Retrieved March 16, 2012, from http://caniuse.com/.

[14] A. Charland and B. Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[15] Y. Granot, A. Ivorra, and B. Rubinsky. A new concept for medical imaging centered on cellular phone technology. *Plos one*, 3(4):e2075, 2008.

[16] J. Hao and K. Zhang. A mobile interface for hierarchical information visualization and navigation. In *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, pages 1–7. IEEE, 2007.

[17] G. I. A. M. I. Fette. The websocket protocol draft-ietf-hybi-thewebsocketprotocol-17 (2011, september 30). Retrieved March 13, 2012, from http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17.

[18] L. Lippert and M. H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum*, 14(3):431–443, 1995.

[19] E. Luke and C. D. Hansen. Semotus visum: A flexible remote visualization framework. In *IEEE Visualization*, pages 61–68, 2002.

[20] M. Staff. Apple accepts webgl into iad, not public ios 5 (2011, june 17). Retrieved March 12, 2012, from http://www.ipodnn.com/articles/11/06/17/multiple.parties.call.technology.security.risk/.

[21] A. Wessels, M. Purvis, J. Jackson, and S. Rahman. Remote data visualization through websockets. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, pages 1050–1051. IEEE, 2011.

[22] M. A. Westenberg and J. B. T. M. Roerdink. Frequency domain volume rendering by the wavelet X-ray transform. *IEEE Trans. Image Processing*, 9(7):1249–1261, 2000.

[23] H. Zhou, H. Qu, Y. Wu, and M.-Y. Chan. Volume visualization on mobile devices. *National Taiwan University Press: Taipe*, 2006.

Handling Variability in Open Source Software

Edy Suharto, Andres Tello Guerrero

Abstract—The reuse of software helps to decrease the effort and costs of software development, and increase the quality of software products. On the other hand, variability is one way to make software reuse possible. For this reason variability management has been the target of software engineering studies for about two decades. During all these years of study researchers have established the base of variability-related concepts, they have categorized variability in different types, and they have proposed diverse variability realization techniques. However, there is not clear knowledge about variability management in open source software (OSS) projects. Therefore, this research tries to identify how OSS projects manage variability. We focus our study in three OSS projects that shows an intense variability: Drupal, WordPress and Oxwall. This study identifies how these projects handle variability with regard to predefined variability realization techniques, and tries to determine if there are any ad-hoc ways to handle variability. In this study, we present which types of variability we found in the studied OSS projects, and which are the realization techniques used in OSS projects to deal with variability.

Index Terms—Variability, Variability Management, Open Source Software

1 Introduction

Variability is a field of research that has been exploited by academy and industry for many years. The feature-oriented domain analysis method and the synthesis approach are the first contributions to variability management research and practice [2]. In addition, in the software product line (SPL) domain, researchers have proposed diverse methods to define, represent, exploit, implement and evolve variability in a product, which is called variability management (VM) [2]. However, far too little attention has been paid to VM in OSS. It is known that OSS give us the possibility to customize products in order to meet specific needs. In addition, OSS products can run on different platforms or devices. Moreover, OSS components can be assembled and reused to create another software products. Furthermore, there are open source ERP systems and Content Management Systems (CMS) which are pre-planned to allow extending its functionality. Therefore, the presence of variability in OSS practices is apparent, but the question that arises now is *how OSS projects handle variability.*

In order to study VM in OSS, we conduct a research which is focused on analyzing three variability-intensive open source systems: Drupal, WordPress and Oxwall. The study identifies whether these projects implement the predefined variability realization techniques or they have their own ways to deal with variability. Moreover, it is important to determine which types of variability are the most common in OSS practices.

The rest of this paper is organized as follows. Section 2, which presents the current state of knowledge regarding to variability and VM, is the theoretical base for this work. Section 3 presents the methodology we use to conduct this study. Section 4 presents how Drupal, WordPress and Oxwall OSS projects deal with variability. In Section 5 we discuss the findings encountered in Section 4, and the paper is concluded in Section 6.

2 Variability and Variability Management

After 20 years of research, there has been important contributions in the field of variability and VM. In this section we present the contributions that are related with the purpose of our study. We present some terminology necessary to understand what is variability and VM, the types of variability, the scope of variability, and the variability realization techniques proposed in previous studies.

- *Edy Suharto, University of Groningen*
  E-mail: e.suharto@student.rug.nl; edys@undip.ac.id
- *Andres Tello Guerrero, University of Groningen*
  E-mail: m.a.tello.guerrero@student.rug.nl; andres.tello@ucuenca.edu.ec

2.1 Variability-related concepts

Within the context of variability and VM research there are a number of terms that results difficult to understand. Therefore, it is necessary to define some terminology which will help to a better understanding of this topic. The concepts presented in this section are a compilation of previous studies, but they will be explained in a intuitive manner. Figure 1 shows the features of a mobile phone, and we relate this diagram with the variability concepts to simplify its understanding.

A **feature** represents a unit of software functionality [9] that is specified by a set of functional and quality requirements [1], e.g. the features of the mobile phone are the input mechanism, send/receive calls and extras (see Fig. 1). Previous studies [7] proposed the following categorization of features:

- **Mandatory Features.** They can be implemented directly in the application because they are not environment-dependent [9]. Mandatory features represent functionality that must be included in all products, e.g. the input mechanism and send/receive calls in the mobile phone (see Fig. 1).
- **Optional Features.** These are features that, when enabled, add some value to the core features of a product, e.g. the extras in the mobile phone (see Fig. 1).
- **Variable Features.** also known as **Variant Features.** These are features that require customization [9]. In our example, the input mechanism and the extras are variable features (see Fig. 1).

The options available for a variable feature are called **Variants** [9], e.g in the mobile phone's feature diagram the input mechanism is a variable feature with two variants: keyboard and touch screen. However, only one of these variants can occur at one time (see Fig. 1). A **Collection of Variants** is the whole set of variants available for one variant feature [15]. **Variability** is the ability of an artifact to be configured, customized, extended, or changed for use in a specific context [2]. In other words, it is the ability to change or customize a system [14]. A **Variation Point** is a particular place in a software system at which the variation will occur [6]. The idea behind this is that a variant feature derives in a collection of variants and a number of variation points in the software system, and these variation points are used to tie in a particular variant to the rest of the system [15]. **Variability Management** is the action of define, represent, exploit, implement and evolve variability throughout the lifecycle of a software product [2]. VM consists of two main tasks: manage the collection of variants, and bind the system to one variant [14].

The concepts presented so far are the base to understand VM. In addition, we present the terminology proposed by Kim et *al.* [8] necessary to address the types and scope of variability.

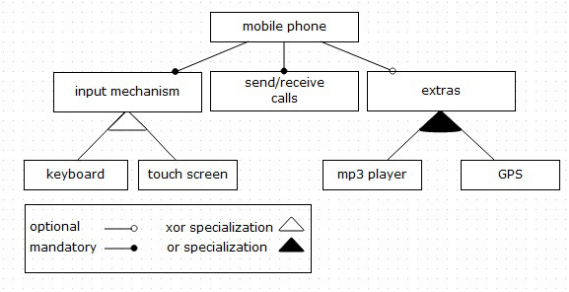


Fig. 1. Feature diagram of a mobile phone.

A **Domain** is a business sector that wants to develop an application or a reusable component [8]. A **Family** is a set of organizations in a domain interested in developing or using reusable components. Each organization belonging to a family is called a Member of the family, and it could be a producer or consumer of reusable software [8]. In addition, Kim et *al.* [8] proposed REQ as a software requirement, and REQ_i as a software requirement of $Member_i$. Moreover, a $CommonREQ$ is a software requirement that is common among some members of the family and has to be included in a software product. R_i is an individual function of $CommonREQ$, which is identifiable and distinguishable by domain experts. Thus, $CommonREQ$ are a set of atomic functions $(R_1.....R_n)$ in the view of domain experts. On the other hand, a $VariableREQ$ is a subset of $CommonREQ$ and each atomic function $VR_i \in VariableREQ$ presents a minor and detailed difference among some family members.

2.2 Types of variability

Kim et *al.* [8] proposed a categorization of variability types, which serves as the theoretical base to identify types of variability in the studied OSS projects.

Attribute variability. It implies the presence of variation points on the set of attributes used by an atomic function $R_i \in CommonREQ$. It could be a variation on the number of attributes, a variation on the datatype of the attributes, or a variation on the value of persistent attribute used by the function R_i between family members.

Logic variability It implies the presence of variation points on the algorithm or the logic of the function $R_i \in CommonREQ$. The variation could be on the algorithm or procedural flow, on the post condition, on the exception handling, or on side effects (e.g. database update) of the function R_i between family members.

Workflow variability It implies variation points on the sequence of method invocations within the function $R_i \in CommonREQ$. The variation could be on order and type of method invocations, or on the number of methods invoked by the function R_i.

Persistency variability It implies variation points on the physical schema or representation of the persistent attributes on a secondary storage. The variation could be on the methods to connect a component to the representation on secondary storage, or in the physical database schema.

Interface variability It implies variation points on the method signatures of the interface used by an atomic function $R_i \in CommonREQ$. Interface variability occurs because each family member could have its own form of interface for a function (e.g. the method name, order and type of parameters, return type). In addition, a Member can adopt a convention for method names and the interface of components to be used do not meet these standards. Moreover, a Member can use legacy systems which need to invoke functionality of a new component, in this case the signature of the methods are already fixed and the component has to be customized.

Combined variability Normally one type of variability causes another type. Kim et *al.* [8] found the that the more common combinations of variability are attribute combined with logic, interface combined with logic, and attribute combined with persistency.

2.3 Scope of variability

Kim et *al.* [8] proposed three kinds of scopes of variation points: binary scope, selection scope and open scope.

Binary scope In a binary scope there are only two variants for a variation point and they are already known. Thus, a variation point with binary scope can adopt one of those variants.

Selection scope In a selection scope there are three or more variants for a variation point and they are already known. A variation point with selection scope can adopt only one of those variants. The reason why Kim et *al.* [8] separate binary and selection scope is because the language constructions to implement the variants are different in each case. While, the construction *if-then-else* is used in a binary scope, the *switch-case* construct is employed in a selection scope.

Open scope In a open scope there are any number of known variants for a variation point, but also a number of unknown variants which could be used later to allow evolving and customizing components.

2.4 Variability realization techniques

A variability realization technique is a way in which one can implement a variation point [15]. Before to present the variability realization techniques proposed by Svahnberg et *al.* [15], we present some terms that are necessary to understand each of the variability realization techniques.

Introduction time refers to at which phase of the lifecycle the variant features can be introduced. After introducing the variant features, it is necessary to add new variants for each variant feature. *Open for adding variants* refers to at which phase of the lifecycle the new variants can be added. The *Collection of variants* in a realization technique can be implicit if they are not represented in the software system, or explicit if they are present in the source code of the software system. *Binding times* refers at which phase of the lifecycle the system can be tied to a specific variant. *Functionality for binding* can be internal or external, it depends on whether that functionality is in the source code of the system or not.

Architecture Reorganization In this technique the components are presented as subsystems and the variants are introduced in the system by means of configuration management tools. Due to there is no first-class representation of the architecture in the system this technique is implicit and external. It allows to add new variants only during the architectural design and the binding time is when a particular product architecture is derived from the product line architecture [15].

Variant Architecture Component This technique proposes support several architectural components representing the same conceptual entity, and the selection of which component to use any given moment is then delegated to the configuration management tools. The introduction time of variant features is during the architecture design. This technique has an implicit collection of variants, the binding time is when the specific product architecture is derived from the product line architecture, and the functionality for binding is external [15].

Optional Architecture Component This technique is intended to provide support for a component that may or may not be present in the system. The solution proposed depends on where it should be implemented. If the solution will be implemented in the calling side, then another variability realization technique introduced in a later phase of the development process, will implement it. Conversely, if the solution will be implemented in the called side, it is necessary to create a null component which has an interface but replies dummy values. This implies that there are predefined dummy values that other components know they have to ignore. The binding time is external to the system [15].

Binary Replacement - Linker Directives This technique represents the variants as standalone library files, and instructs the linker which library link to the systems. The introduction time in this technique is during the architecture design. This technique is open for new variants and bound to a specific variant during the linking phase. The collection of variants may be implicit or explicit, and the binding functionality may be external or internal. It depends on when the linking is done. If the linking is done at runtime the collection of variants is explicit, and therefore the functionality for binding is internal. But if the linking is done at compilation and linking phase prior to delivery, the collection of variants is implicit and the functionality for binding is external [15].

Binary Replacement - Physical This technique is intended to allow introducing new variants after delivery. The software systems are not always perfect and they need variations or corrections after delivery. To introduce variations after delivery it is necessary to change the binaries of the system. Therefore, the solution is to replace the entire binary file with a new copy. However, the system has to be organized as a number of small binary files in order to restrain the impact of replacing a file. The new file can replace the entire function of the old one, or introduce new functionality, e.g. a new variant feature using other variability realization technique. This technique is always open for adding new variants, the collection of variants is implicit and functionality for binding is external [15].

Infrastructure-Centered Architecture This technique tries to make the connections between components a first class entity. This technique proposes the architecture becomes an explicit entity in the system, enabling reorganize the architecture, or replace a component in the architecture. Thus, the components are not connected each other, but they are connected to the infrastructure instead. Hence, the infrastructure has to provide mechanisms to match the component's interfaces and thereby allow components to interact each other. The collection of variants may be implicit or explicit and the functionality for binding is internal, provided by the infrastructure. This technique is open for new variants at architecture design, linking, or runtime [15].

Variant Component Specializations . Sometimes is necessary to modify the interface of a component in order to this fits in the architecture. Hence, this technique intends to adjust the component implementation to the architecture of the system. The solution is to implement the component interface as separated classes, and allow to choose which to use by means of a configuration management tool. This technique has a implicit collection of variants and the functionality for binding is external [15].

Optional Component Specializations In this technique, customizing certain component implementation includes adding or removing parts of its behaviour. Optional behaviour is separated into separate class and a null class that acts as placeholder is created. Then configuration management tool decides which one of them is included in system. Software entities involved here is smaller than that of Optional Architecture Component technique [15].

Runtime Variant Component Specializations This technique requires component implementation that it adapts to the environment in which it runs. Several design patterns are applicable to collect alternating behaviour into separate classes explicitly and mechanism are introduced to select among classes during run-time. Design patterns implement variability using language construct such as inheritance and polymorphism [15].

Variant Component Implementations Motivated by the fact that architecture component typically represents some domain or subdomain, this technique implements several component implementations adhering to the same interface and make them tangible entities in system architecture. Part of flexibility of this technique stems from collection is explicitly represented in the system while the binding is internally done [15].

Condition on Constant This techniques includes several variants of which only one will be used. To implement variations using this technique we can use preprocessor directives (e.g. #define, #ifdef, etc. in C++) or traditional if-statements of a program language. The introduction time is at implementation phase, when also this technique is open for new variants. The binding time is at compilation. The collection of variants and the functionality for binding may be implicit and external, or explicit and internal. It depends on the implementation mechanisms. The first case is when we use preprocessor directives and the latter is when we use if-statements [15].

Condition on Variable It supports several ways to perform an operation, of which only one will be used at any given moment, but allow the choice to be rebound during execution. The solution to implement this technique is replace the constant used in Condition on Constant for a variable, and provides functionality to change this variable. The introduction time of variable features and adding new variants are during the implementation phase. The collection of variants is explicit and the functionality for binding is internal [15].

Code Fragment Superimposition This technique tries to introduce variants in the system but without modifying the source code. The solution to achieve this is to develop coarse-grained software, and then superimpose the specific concerns of a product when the source code is completed anyway. Using this technique the additional behavior is wrapped around existing behavior. This technique is open for new variants at compilation, and the system may be bound to a specific variant at compilation or runtime. The collection of variants in this case is implicit, and thereby the functionality for binding is external [15].

3 METHODOLOGY

To conduct this study it is necessary to follow a certain order or methodology. Therefore, we formulate some questions which established the steps to follow.

3.1 Selection of OSS

The question that we want to solve is how OSS projects deal with variability management. However, it is necessary to determine which projects we will examine. As we cited earlier, variability is the ability to change or customize a system [14]. Thus, in order to facilitate the study, the chosen OSS projects should be highly customizable. Therefore, we decide to analyze three variability-intensive systems such as Drupal, WordPress and Oxwall.

3.2 Analysis of OSS

After presenting and analyzing the theoretical base about variability and VM introduced in section 2, it is necessary to define what we will explore in the chosen OSS projects. Thus, we consider important to identify which types of variability are present in each of the chosen OSS projects. In addition, we try to identify which is/are the variability scope(s) used by each OSS project. Moreover, we try to identify the realization techniques that those projects use with regard to the techniques presented earlier.

We count on two resources to identify the variability realization techniques used by the studied OSS projects: the source code and the technical documentation. Therefore, first, reading the general documentation we tried to identify the components of each system. Second, we tried to identify which of those components allow customization or extend functionality to the core system. And finally, after having analyzed the conceptual background presented in section 2, reading the technical documentation, and looking into the source code, we tried to identify variant features, variants and variation points within each system. With these concepts identified, it was possible to determine which of the types, scopes, and variability realization techniques are present in each OSS system. However, we did not analyzed all the modules and core functions of each system, but we explored variability-related concepts in the modules that allows high customization.

4 Variability in Open Source Software

In this section we present the mechanisms that Drupal, WordPress and Oxwall use to manage variability.

4.1 Drupal

Drupal is an open source content management system, which has different distributions available. However, for this study we analyzed the last stable one: version 7.12. Drupal is designed using five layers: data, modules, blocks, user permissions and templates. However, in this study we analyze how drupal manage variability in the Module System and the Database Abstraction Layer.

Module System. Drupal enables to add functionality by means of installing new modules, and allowing them to interact with the Drupal core. Drupal's module system is based on the concept of *hooks*; which in computer science refers to a range of techniques used to alter or augment the behavior of an operating system, of applications, or of other software components by intercepting function calls or messages or events passed between software components [16].

In Drupal, a hook is a PHP function that is named foo_bar(), where 'foo' is the name of the module with filename foo.module, and 'bar' is the name of the hook. Each hook has a defined set of parameters and a specified result type.

To extend Drupal, a module need simply implement a hook. When Drupal wishes to allow intervention from modules, it determines which modules implement a hook and calls that hook in all enabled modules that implement it" [3]. Let us clarify this with an example. Consider the file "help.api.php" that has the core functions of the drupal module *help*. In this file, the core function hook_help is implemented; this function provides online user help (see Figure 2). Therefore, each module that wants to provide documentation to users only have to implement the function hook_help(). For our example, we take the module blog which implements the function blog_help() in the file "blog.module" (see Figure 3). Hence, when the user tries to see the documentation of that module, Drupal will look into its .module file and execute the function blog_help() instead of the core function hook_help().

```
function hook_help($path, $arg) { // this function introduces a variation point
  switch ($path) {
    // Main module help for the block module
    case 'admin/help#block':    //variant 1
      return '<p>' . t('Blocks are boxes of content rendered into an area, or region, of
a web page. The default theme Bartik, for example, implements the regions "Sidebar
first", "Sidebar second", "Featured", "Content", "Header", "Footer", etc., and a block
may appear in any one of these areas. The <a href="@blocks">blocks administration
page</a> provides a drag-and-drop interface for assigning a block to a region, and for
controlling the order of blocks within regions.', array('@blocks' =>
url('admin/structure/block')) . '</p>';
                                        .........
  }
}
```

Fig. 2. Code snippet of the function hook_help().

```
function blog_help($path, $arg) { //implementation of the variation point hook_help
  switch ($path) {
    case 'admin/help#blog':  // variant 1
      $output = '<h3>' . t('About') . '</h3>';
      $output .= '<p>' . t("The Blog module allows registered users to maintain an
online journal, or <em>blog</em>. Blogs are made up of individual <em>blog entries</em>.
By default, the blog entries are displayed by creation time in descending order, with
comments enabled, and are promoted to the site's front page. For more information, see
the online handbook entry for <a href='@blog'>Blog module</a>.", array('@blog' =>
'http://drupal.org/handbook/modules/blog/')) . '</p>';
                                      ......
  }
}
```

Fig. 3. Code of the function blog_help().

Database Abstraction Layer. Drupal provides the database abstraction layer to allow the use of different database servers using the same core functionality. To achieve this, the Drupal database layer requires a driver for each database type. A driver consists of a set of files located in includes/database/*driver*, where *driver* is a string representing the unique key for that driver. Each driver consists of several classes derived from parent classes in the core database system. These driver-specific classes may override whatever behavior is needed to properly support that database type [4]. The binding to a specific variant is made via configuration tool at Drupal's installation.

4.2 WordPress

WordPress is another Content Management System, and it is specially used to create websites and blogs. In this study we present the mechanisms that WordPress uses to handle variability.

WordPress enables to add functionality using what they call plugins. A WordPress Plugin is a program, or a set of one or more functions, written in the PHP scripting language, that adds a specific set of features or services to the WordPress weblog, which can be seamlessly integrated with the weblog using access points and methods provided by the WordPress Plugin API [18].

WordPress, as the case as Drupal, uses "hooks" to allow plugins interact with its core functions. However, the hooks in WordPress work slightly different. WordPress has two types of hooks:

- **Actions** are the hooks that the WordPress core launches at specific points during execution, or when specific events occur. A plugin can specify that one or more of its PHP functions are executed at these points, using the Action API [17].
- **Filters** are the hooks that WordPress launches to modify text of various types before adding it to the database or sending it to the browser screen. Your plugin can specify that one or more of its PHP functions is executed to modify specific types of text at these times, using the Filter API [17].

WordPress uses some specific functions to introduce variability in its core functionality. In this study, we describe two functions which are strongly related with variability concepts: *do_action()* and *add_action()*. The first one enables to create an action hook which WordPress will launch at some point, and the latter one, allows to specify the action to be executed when the hook is launched. WordPress has a variety of predefined "action hooks"; however, to explain how WordPress works, we will use the *wp_head*. This action hook is normally located in the last line of the $< head >$ section of a theme's header file. The *wp_head* action hook allows to any developer to add functionality when the $< head >$ of a web page is loaded.

```
11  ?><!DOCTYPE html>
12  <html <?php language_attributes(); ?>>
13  <head>
14  <meta charset="<?php bloginfo( 'charset' ); ?>" />
15  <title><?php
16      /*
17       * Print the <title> tag based on what is being viewed.
18       */
19      global $page, $paged;
                                .......
51      wp_head(); //action hook. (variation point)
52  ?>
53  </head>
```

Fig. 4. Header section of the WordPress' default Theme

Let us illustrate this with an example again. Consider the header section of the WordPress' default Theme, which has the *wp_head* action hook (see Figure 4). Thus, if for example a developer wants to override the link colors of the theme, he has only to create a plugin with a function to do so, and then says WordPress that use it by means of the *wp_head* action hook (see Figure 5). The *add_action* function says to WordPress executes *my_own_function* when you find the *wp_head* action hook. On the other hand, to allow other developers add functionality to the plugin just created, new action hooks can be introduced by means of the *do_action()* function. *do_action('my_action_hook')*.

```
function my_own_function() {
   //some code here
}
add_action('wp_head','my_own_function')
```

Fig. 5. Use of the *add_action()* function

Now we want to explain the variability concepts behind the analyzed functions. The *do_action()* function enables to introduce *variation points* into the WordPress' classes, functions or methods, and the *add_action()* function enables to tie in a particular variant to the rest of the system.

4.3 Oxwall

Oxwall is another PHP/Mysql community software platform, but the difference for this Content Management System is that it also allows to create custom social Networks. Oxwall provides website developers with the ability to change the way it works using plugins. In this case, plugins are also the artifact to handle variability. These plugins are defined as complete units of functionality which can be used to meet different purposes. Oxwall manages core and plugins compatibility in order to make any updates are quite easy for developers [11].

Oxwall has standardized a design language that plugins and themes should use. It is a set of controls, CSS class, decorators and interface paradigm to ensure compatibility among plugins and themes. The plugins implement functionality that requires user interfaces. User interfaces use standard tools which themes implement [12].

Plugins in Oxwall are located under *ow_plugins* directory. Plugins need to be registered in an xml file mentioning plugin key and developer key (see Figure 6), installed and activated before development. Since Oxwall employs MVC pattern, it needs such an organization of files into controllers and views folders. It specifies some rules in order to keep the application runs stable although it interacts with many standardized plugins. For instance, the rule to create a controller class is that its name should start with a prefix containing plugin key in upper case followed by underscore and the word 'CTRL'. The name of its associated file should be its class name minus the prefix, then its capital letters must be changed into lower case separated by underscore. All controller classes should inherit *OW_ActionController* class or its child classes [13]. Oxwall also provides some classes for further specific purposes such as *OW_ActionController* for controller, *OW_entity* and *OW_BaseDao* for database purposes, and *ADMIN_CTRL_Abstract* for managing data in admin panel page. The architecture of Oxwall

```
<?xml version="1.0" encoding="utf-8"?>
<plugin>
        <name>Blogs</name>
        <key>blogs</key>
        <description>User blogs with archives, tags, comments and rates
        </description>
        ...
        <developerKey>e547ebcf734341ec11911209d93a1054</developerKey>
        ...
</plugin>
```

Fig. 6. Plugin registration in file plugin.xml

is built on simple and compact core completed with additional plugins that bring functionalities. Even standard platform features such as user admin and admin panel were developed as system plugins that cannot be uninstalled. As mentioned previously, MVC in Oxwall is realized using model, view and controller. Controller processes user input then communicates with model and provides data for view rendering. Model interacts with database and implements the business logic. Model is categorized as two instances: service and data access object (DAO). Service deals with business logic while DAO handles database manipulation. View receives processed and ready for markup data and contains presentation logic [10].

Beside MVC, Oxwall employs Managers which are represented by global objects to provide main functionality for plugins developers. There is a unique static class for getting access to all managers called OW. OW provides all necessary functions such as *getAutoLoader()* and *getPluginManager()*. The first function deals with class and package registration for keeping developer from manual class includes while the latter one handles plugins management including plugins initialization based on naming rules [10]. Figure 7 depicts the use of OW class to handle a blog plugin.

4.4 Findings

After analysing the three OSS systems, we present the variability-related concepts that we found in each of them.

Table 1 summarizes the findings with regard to variability-related concepts. In this table we show the variation points, the variants, the types, scopes and variability realization techniques found in each of the studied systems.

```
final class OW
{   ...
    /* Returns autoloader object. */
    public static function getAutoloader()
    {
        return OW_Autoload::getInstance();
    }
    /* Returns system plugin manager object. */
    public static function getPluginManager()
    {
        return OW_PluginManager::getInstance();
    }
    ...
}

final class OW_PluginManager
{   ...
    /* Returns active plugin object.*/
    public function getPlugin( $key )
    {
        ...
        return $this->activePlugins[mb_strtolower(trim($key))];
    }
    ...
}

// getting root directory for blogs plugin
OW::getPluginManager()->getPlugin('blogs')->getRootDir();
```

Fig. 7. Use of the OW class

Table 1. Findings Summary

OSS Project	Drupal	Wordpress	Oxwall
Variation Points	hook_foo()	action and filter hooks, functions created using do_action()	-
Variants	implemented hooks, each module	functions invoked using add_action(), each plugin	each plugin
Variability Type	Persistency	Workflow	-
Variability Scope	Selection, Open	Open	Open
Realization Techniques	Condition on Variable, Variant Architecture Component	Code Fragment Superimposition	-

Drupal introduces variation points by means of each hook function (e.g. hook_help, hook_form, hook_menu, etc.) implemented in the core system. The variation points in Drupal are each of the hook functions implemented by the third party modules (e.g. blog_help, blog_form, etc.). One of the types of variability founded in Drupal is the persistency variability in the database abstraction layer. The scopes of variability in Drupal are selection and open. We can identify the selection scope in the implementation of the function hook_help(); it has more than two options(variants), and the language construction used in the implementation is switch-case. The open scope is clearly identifiable because Drupal is designed in a way that allows integrating as many modules as it is necessary. The variability realization techniques used by Drupal are condition on variable in the module system, and variant architecture component in the database abstraction layer.

The variation points in WordPress are each of the action and filter hooks implemented in the core functionality (e.g. wp_head, wp_footer, publish_post, etc), and each of the custom hooks introduced by means of the function do_action(). The variants are each of the function calls made using the function add_action(), and each plugin that is added to WordPress. The type of variability identified in WordPress is the workflow variability. Every time WordPress launches a hook and a customized function is invoked, the execution workflow will change. The scope of variability is open because we can add as many plugins as it is necessary. The realization technique identified in WordPress is code fragment superimposition, which proposes to develop the software to

function generically, and then superimpose the product-specific concerns when the source code is completed. The additional behavior is wrapped around existing behavior [15].

While the variability-related concepts were clearly identified analyzing the technical documentation of Drupal and WordPress, Oxwall does not provide enough documentation to make the same analysis. However, the implicit concepts are variants and scope of variability. The variants are each of the plugins added to the core functionality, and the scope of variability is open because Oxwall allows to add as many plugins as is necessary.

5 Discussion

In section 4 we presented a general overview about the mechanisms that Drupal, WordPress, and Oxwall OSS projects use in order to handle variability. In their architectural view, the software is divided into parts based on types of functionality. The basic functionality which remains invariable are handled by the core modules. In the other hand, the advanced functionality which tends to change are managed in additional modules (in Drupal) or plugins (in Wordpress and Oxwall). This separation of concerns keeps the application size lean but still allows the application itself to grow as new requirements on features appear. In the perspective of developers as the user, the software can be developed in various versions depending on their need. Thus, it is obvious that these OSS projects have designed their software so flexible and adaptable that in the future variability could be well managed.

The use of plugins is the common technique among the projects. In addition, all of them use those plugins to extend its functionality. However, the mechanism that each OSS project uses to integrate those plugins with the core functions differs a bit among each other. For example, both Drupal and WordPress use "hooks". However, while Drupal uses "hooks" to call a plugin's function instead of a core function, WordPress uses "hooks" to call a plugin's function in the flow of execution of the current one. Moreover, WordPress enables developers to create new variation points into the software execution. On the other hand, Oxwall provides very limited technical information to explore in detail how it integrates the plugins with the core functions, but the fact of adding new functionality involves variability management. Such a unique "hooks" mechanism in Oxwall is implemented using an xml file which contains plugins information.

In general, Drupal, Wordpress and Oxwall provide a sort of abstract classes containing abstract functions which then are extended by the user in the implementation stage. Those classes define some interfaces to connect the core classes and plugins. Specific plugins are created by inheriting these classes and modifying the behaviour of a particular function. The interface itself is bidirectional. It must contain not only some functions that are called by the core classes which the plugin implements, but it must also contain functions which the plugin can call at the core classes [5].

6 Conclusion

Software reuse allows to reduce the development effort and increase the quality of a software product. Variability is one way to achieve the software reuse. In this paper we presented the more important concepts needed for a better understanding of variability and variability management such as feature, variant, variability, variation point, variability management. In addition, we presented the types of variability, the scopes of variability and the variability realization techniques proposed in previous studies in this field.

The purpose of the current study was to determine *how open source software handles variability*. Thereby, after examining the technical documentation provided by three variability-intensive OSS projects: Drupal, WordPress, and Oxwall, we discovered that in their development practices they implicitly involve variability concepts such as variants, and variation points. Furthermore, in the modules that we analyzed in each system, we discovered they use existing variability realization techniques such as Condition on Variable, Variant Architecture Component, and Code Fragment Superimposition. However, exploring variability in all the modules that allows customization might reveal other variability realization techniques employed by the OSS projects.

This study also revealed the commonalities of the systems being studied. Having that the three projects that were examined share the same domain of application; they are Content Management Systems. In addition, they use PHP as scripting programming language and Mysql as database management system. These similar characteristics led us to converge in the scope of this study. However, handling variability in other domains such as desktop applications and non-PHP-Mysql platforms might not provide the same results. Therefore, it challenges further work to explore variability management in more OSS platforms.

Finally, an important limitation needs to be considered. We based our findings in concepts that arose into the SPLs context. The fact that the information about VM in OSS is very scarce led us to use the technical information of each system and make a comparison with the information of previous studies. Nonetheless, this limitation made it possible to reveal that variability-related concepts proposed within the SPLs domain are also used in OSS practices. Therefore, our research could motivate practitioners who wants to integrate OSS systems into their product lines.

Acknowledgements

The authors wish to thank Matthias Galster who has reviewed this paper as an expert in Software Engineering field, and has proposed some improvements for this research. In addition, the authors also want to thank our colleagues Sardar Yumatov and Gerjan Konterman who have reviewed this paper as well. Last but not least, the authors give special thank to Rein Smedinga, Michael Biehl, Femke Kramer and Janneke Geertsema who have inspired and given useful guidance for doing this work.

References

[1] J. Bosch. Design and use of software architectures - adopting and evolving a product line approach. 2000.

[2] L. Chen and M. A. Babar. Variability management in software product lines: An investigation of contemporary industrial challenges. 14th International Software Product Line Conference, Springer Verlag, Jeju Island, South Korea, 2010.

[3] Drupal^{TM}. Drupal api - hooks. http://api.drupal.org/api/drupal/includes%21module.inc/group/hooks/7.

[4] Drupal^{TM}. Database api - general concepts. http://drupal.org/node/310070, 2011.

[5] F. Faase. The plug-in pattern. http://www.iwriteiam.nl/PlugInPattern.html.

[6] L. Geyer and M. Becker. On the influence of variabilities on the application engineering process of a product family. Computer Science, 2379 Proceedings of the Second Software Product Line Conference, 2002.

[7] M. L. Griss, J. Favaro, and M. d'Alessandro. Integrating feature modelling with the rseb. Fifth International Conference on Software Reuse. IEEE Comput. Soc, Los Alamitos, CA, USA, 1998.

[8] S. D. Kim, J. S. Her, and S. H. Chang. A theoretical foundation of variability in component-based development. 2005.

[9] A. Lozano. An overview of techniques for detecting software variability concepts in source code. June 2011.

[10] S. Madumarov. Oxwall: The developer's view. http://oxart.net/blogs/oxwall-the-developers-view, December 2011.

[11] Oxwall.org. Oxwall software. http://www.oxwall.org/.

[12] Oxwall.org. Oxwall design overview. http://docs.oxwall.org/design:overview, January 2012.

[13] Oxwall.org. Plugin development crash course. http://docs.oxwall.org/dev:begin:crash-course, January 2012.

[14] M. Svahnberg. Variability in evolving software product lines. 2000.

[15] M. Svahnberg, J. V. Gurp, and J. Bosch. A taxonomy of variability realization techniques, software. 2005.

[16] Wikipedia.org. Hooking. http://en.wikipedia.org/wiki/Hooking, January 2012.

[17] Wordpress.org. Plugin api. http://codex.wordpress.org/Plugin_API.

[18] Wordpress.org. Writing a plugin. http://codex.wordpress.org/Writing_a_Plugin.

A systematic mapping study on sustainable software engineering: A research preview

Christian Manteuffel, Spyros Ioakeimidis

Abstract— Since computers have become pervasive in our daily life, software influences the way we live potentially impacting sustainability in various ways. Sustainable software engineering aims to create reliable, long-lasting software that meets the needs of users while reducing negative impacts on economy, society, and environment .
Currently, the research community aims to define a research agenda to explore sustainability in software engineering. An overview of existing work in this area supports the creation of a research agenda by defining the field of interest and by identifying the community of researchers.
In this paper we present a preview on a systematic mapping study on sustainability in software engineering. The goal is to provide an overview of the existing research by classifying published literature. The paper discusses the motivation, the research questions, the definition of the search strategy as well as selection criteria and data extraction. In the end we show the results of a preliminary evaluation of the search string.

Index Terms—Mapping study, Sustainable software engineering, Software development life cycle, Sustainability.

1 Introduction

Researchers have recently started to explore how sustainability issues can be addressed in the field of software engineering. This includes the perceived direct and indirect impact of software on society, economy and environment. Computers and software influences our everyday decisions and the way we live, potentially impacting sustainability in various ways. For instance, a shopping advisor for a mobile phone could help to identify local sustainable-grown vegetables in supermarkets or a car sharing app would help to reduce traffic in metropolitan areas, both positive influencing the society and the environment.

Moving towards a more sustainable humanity mandates technological improvement, national and international regulations, urban planning but also cultural adjustment and a critical questioning of the status quo. Hence, achieving sustainability is a multidisciplinary problem that, among others affects the software engineering discipline. Software systems are omnipresent in all areas of life, both business and private. Consequentially, they can impact the society and the environment in ways, the software engineer has not thought about during development. Examples of such effects is the impact of different architectural styles on the overall energy consumption or the impact of the print layout on the amount of printed paper. Those inconspicuous design decisions can have a severe effect on sustainability. Understanding the influence of design decisions on sustainability is difficult due to the veiled and multi-facetted nature of impacts, which is illustrated by the following example.

Overall, the rising attention of the climate change and the awareness of the fragility of the human environment have lead to an increased research effort on sustainability — also in software engineering. According to [14], the number of publications on sustainable software engineering (SSE) has increased since 2005.

We propose a study that aims to give an overview of the status quo in SSE research and thus helps to identify challenges and topics for future research. We choose to conduct a systematic mapping study because it is particularly designed to "map out" research that has been undertaken [7]. The study protocol is based upon the template described in [10]. It defines the context of the study, the research questions that should be answered, the methodology, possible threats to validity and the overall schedule.

- *Christian Manteuffel is with University of Groningen, E-mail: cm@notagain.de.*
- *Spyros Ioakeimidis is with University of Groningen, E-mail: spyrosikmd@gmail.com.*

This report presents a preview of our systematic mapping study. It discusses the research questions, the methodology as well as the analysis. At first the concept of sustainability in general and sustainability in software engineering are introduced. Section 2 presents related work, the research questions as well as expected results. Section 3 discusses the methodology of our study. This includes the design of the search query and the definition of the selection criteria. Section 4 describes the data extraction and classification procedure. Section 5 assesses threats to validity. Section 6 presents a preliminary evaluation of the search string. The paper concludes with an outlook of the next steps.

1.1 Sustainability

Sustainability "is the capacity to endure and can be seen as the challenge to facilitate cultural, economical and technological development without environmental damage [3]". According to the Brundtland Commission of the United Nations, sustainable development ensures that the needs of the present are met "without compromising the ability of future generations to meet their own needs [1]". The long-term goal of sustainable development can be summarized as keeping the earth habitable for future generations. According to [3], sustainable development is based on three interdependent and mutually reenforcing pillars, which are economic development, social development and environmental protection.

Environmental sustainability ensures that the environment is able to replenish itself at a faster rate than it is destroyed by human actions. For instance, the use of recycled material for IT Hardware production helps to conserve natural resources. The second pillar is concerned about creating a sustainable society which includes social justice or reducing poverty. In general all actions that promote social equity and ethical consumerism. The economic pillar ensures that our economic growth maintains a healthy balance with our ecosystem, it integrates environmental and social concerns into business [3].

1.2 Sustainable Software Engineering

According to [5], "sustainable software engineering aims to create reliable, long-lasting software that meets the needs of users while reducing negative impacts on economy, society, and environment".

Those impacts are multi-facetted and not directly visible. Therefore, Goehring defines three levels of how software can affect sustainability [9]. The first-order effects, also called "effect of ICT supply", are direct effects that result from production and use of ICT. Second-level, or "effects of ICT usage" are effects that result indirectly from using the software, i.e. dematerialization effects due to process optimizations. The third-order effects, or "systemic effects of ICT", are indirect effects that result from a long-term use of ICT. For instance,

if ICT changed common life-style behavior that leads to a more sustainable living. However, this also includes rebound effects, which outweigh intentionally or formerly archived improvements [13].

The definition by Amsel et al. focusses on how software itself impacts sustainability. However, it omits one important factor in software engineering — the process. Hence, Nauman et. al distinguish between a sustainable process and a sustainable product. "Sustainable Software Engineering is the art of developing green and sustainable software with a green and sustainable software engineering process [13]". The software development process should facilitate an assessment of the effects on sustainability.

As pointed out by [13], sustainable software engineering includes both using a sustainable software engineering process and building a sustainable software product. We scope the two dimensions as follows:

Process Sustainability in the complete software development process lifecycle, from requirement definition, design, maintenance and evolution until replacement by a new system.

This aspect is concerned with optimizing the development process itself, e.g. reducing the amount of traveling, improving the energy of the developer's workstations but also using agile methods, which usually improve the working environment and thus the human and social aspect of sustainability. Furthermore, software systems should be endurable in order to last as long as possible. The maintenance process should be designed to ensure a constant quality without increasing effort and workload.

Product Sustainability of the software as a product, with respect to its impact during runtime and use of the software.

This means that the software itself should be sustainable, using less resources for during runtime. But also the direct and indirect effects that result from using the software need to be considered. For example, applications that are designed to support a more sustainable life-style or software that optimizes a business process and thus results in dematerialization of resources.

Nauman et al. propose a reference model that includes a cradle-to-grave product life cycle model for software products, called GREENSOFT model [13]. This model includes sustainability metrics and criteria, software engineering extensions for sustainability sound software design and development, as well as appropriate guidance.

1.3 Need for a mapping study

According to [17], "as a research area matures there is often a sharp increase in the number of reports and results made available, and it becomes important to summarize and provide overview."

Researchers have recently start to explore how to address sustainability issues in the field of software engineering. Examples are the First International Workshop on Requirements Engineering for Sustainable Systems at the REFSQ 2012 [16] or the First International Workshop on Green and Sustainable Software at the ICSE 2012. In order to define a research agenda, it would be helpful to get an overview of work that has been already done in this area. This helps to build upon existing topics but also to identify topics for collaborative research.

Our research uses the systematic mapping study approach [17] to provide an overview of the existing sustainable software engineering research by classifying the research according to the field of sustainability, type of research and addressed aspect. These classifiers are used to create mappings that allow to identify areas of interest but also to show aspects of sustainability that have not been well addressed [7].

2 Background

This section illustrates the background of the mapping study by presenting previous research, the research questions and their motivation as well as expected results.

2.1 Previous research

The first systematic literature review (SLR) on sustainability in software engineering by Penzenstadler [14] aims to create a body of knowledge as a starting point for future research. Out of 500 published papers, 96 have been considered relevant. She concludes that based on the results of the SLR, not enough research has been published to build a body of knowledge and instead she proposes an extended body of knowledge, which includes research on sustainability from related domains.

Mahaux et al. published an experience report on a software project in which sustainability was treated as a first class quality [11]. As part of this report, they performed a search on the DBLP Computer Science database for articles related to sustainability and requirements engineering. Only 11 articles have been identified as being relevant.

The need for research on SSE is emphasized by Amsel et al. who conclude that software engineering should focus more on environmental responsibility and add sustainability to the list of important quality attributes [5].

2.2 Research Questions

The following enumeration lists the research questions that the mapping study aims to answer along with their intended purpose and background.

RQ1 What are the most published types of research in the field of sustainable software engineering and how has the distribution of the type of publication changed in recent years?

RQ2 Which of the three pillars of sustainability is addressed most frequently — economic, social or environmental?

RQ3 Which aspect of sustainable software engineering is addressed most frequently — software process or software product aspect?

RQ4 If the research addresses the process aspect, which phases of the software development life cycle are addressed — requirements engineering, design, implementation, or verification?

We ask RQ1 because we want to provide an overview of the research. According to [12] such a classification helps in guiding the reader through the SE literature, and in making the researcher reflect on directions for improvements. If a lot of opinion papers have been published it maybe useful to try to empirical validate some of the statements in order to provide a stronger research foundation. The classification of research papers will be based upon the criteria defined by [12].

We ask RQ2 because sustainability is more than reducing environmental impacts but also impacts on society and economy. We want to check, if research focusses on one particular field or if the fields are equally addressed.

We ask RQ3 in order to provide an overview based on the classification proposed by [14]. Along with RQ1 - RQ2, this question sketches the research that has been done. In particular, RQ3 shows whether research is focussing on sustainability in the software process, sustainability in the software product or both.

We ask RQ4 because sustainability can be addressed in different phases of the software development lifecycle (SDLC), which would raise the interest of different research communities. For instance, sustainability in requirements would be of interest for the requirements engineering community, while achieving energy efficiency could be interesting from a developers point of view. We think that the sustainability aspect of the software development lifecycle is too coarse grained and should be further refined in terms of the SDLC. Therefore we use the SDLC defined in [18].

2.3 Expected results

The expected results of the study is an overview of the research in the field of sustainable software engineering and an identification of gaps in the existing research. Since sustainable software engineering is a young topic, we expect that research consists mostly of opinion, philosophical, and solution proposals and less empirical work has been

done. However, we do not expect a major trend towards case studies and empirical validation but expect to identify a minor trend. Furthermore, we expect that most of the research has been done in the field of environmental sustainability. Concerning RQ3, we are uncertain about the expected results and we imagine that both the process and product aspects are addressed equally. For RQ4 we expect to see the majority of research in the field of requirements engineering and implementation.

3 METHODOLOGY

We conduct a systematic mapping study, which includes the design of a study protocol, the search on selected scientific databases, the identification of relevant papers, data extraction and classification of selected papers, as well as reporting (Figure 1). Our systematic mapping study follows the approach described in [17].

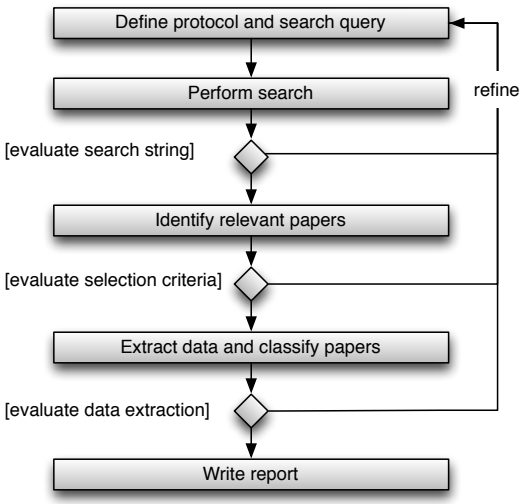


Fig. 1. Mapping study process

3.1 Search Strategy

To identify relevant literature, we perform an automated search on four major electronic databases relevant for software engineering and computer science. An automated search is more efficient but its performance depends on the quality of the search string and the technical capabilities of the search engine.

We follow the subjective definition approach described in [20], in which the search string is constructed based on observations from relevant literature and the authors' knowledge of the field. The sensitivity and precision of the search string is evaluated against an a priori defined set of relevant literature. The search will be performed at least on the title, abstract and keywords. If possible a full-text search should be performed to avoid exclusion of papers that do not include our search terms in the mentioned fields, but are still relevant.

Table 1 lists the databases along with the reason for their selection. These databases have been selected because they are perceived to be relevant to software engineering [6] but also because they allow us to retrieve results from subject areas other than computer science. This is particularly important since relevant sustainability literature may be published in a different field, e.g. Art and Humanities. The search query is derived from the defined research questions [17]. The presented search query does not consider any syntax restrictions of the selected search engines, instead an ideal search engine is assumed. In the actual search process, the query needs to be adapted to the specifics of each search engine. The challenge is to define a query that is extensive enough to identify all relevant papers, but at the same time narrows the search results to a manageable number.

Table 1. Selected Data sources

Library	Reason for selection
Scopus	Multidisciplinary database for scientific literature.
IEEE Xplore	Full-text access to all IEEE transactions, journals, magazines and conference proceedings. Relevant in Computer Science.
Web of Science	Multidisciplinary citation index for Science, Social Sciences and Arts & Humanities.
ACM Digital Library	Relevant in Computer Science.

Based on the research questions, the search query should retrieve literature that is related to the keywords *sustainable* and *software engineering*. However, it is possible that a paper does not explicitly mention these keywords but refers to a similar concept e.g. "ecological application development". Therefore, we extend our search query with synonyms and related concepts.

Sustainable endurable, maintainable, viable, green, ecological, economic, environmental, social

This list is based on synonyms for "sustainable" from the website Thesaurus.com, related concepts for sustainability based on [3] as well as the three pillars of sustainability. We excluded the following synonyms because we do not think that they are significant enough: livable, arguable, tolerable, inhabitable, sufferable.

Software Engineering Software requirements, Software design, Software construction, Software testing, Software maintenance, Software configuration management, Software engineering management, Software engineering process, Software engineering tools and methods, Software quality

This list is deduced from the Software Engineering Knowledge Areas described in the Software Engineering Body of Knowledge [2].

The search query should look for word-stems in order to find papers that contain sustainable but also sustainability. In a first naive approach the two keyword groups have been combined with a logical conjunction in order to ensure that at least one keyword of each group is represented in the paper. This search string has been piloted on Scopus, which resulted in over 35.000 results. We analyzed the results and figured out that the keywords quality, tools and the three pillars resulted in many false positives. After removing these keywords, the search showed 2698 papers. However, the results still contained many false-positives. Hence, we followed a bottom-up approach by including or excluding keywords from the two groups , which resulted in the following search string.

("sustainab" OR "endurab*" OR "maintainab*" OR "viab*" OR "green" OR "ecologic*")*

AND

("software engineering" OR "requirements engineering")

3.2 Selection Criteria

The inclusion and exclusion criteria are used to narrow the number of results by excluding those, which are not relevant to answer the research questions [17]. The selection criteria are used after retrieving the search results from the databases.

The criteria are applied on the title, the abstract, and the keywords of the primary studies. The following lists depict the inclusion and

exclusion criteria. The list of criteria is expected to grow as soon as the first results have been obtained.

Inclusion Criteria

- Studies on software engineering concerning sustainability.

Exclusion Criteria

- Studies which discuss sustainability but are not related to the field of software engineering, or studies in the field of software engineering which do not approach sustainability.
- Studies published before 2005 and after 2011. According to [14], no relevant research on sustainability in software engineering has been published before 2005. We decided to include only complete years into our study in order to simplify the analysis and the interpretation of the results. We can expect that a significant number of papers will be published in the remaining months of 2012, hence our results would not be complete.
- Duplicated papers that have been published in multiple venues like journals or proceedings. Only the most recent one is selected [8].
- Papers already included from another database [8].
- Papers written in any language other than English. The research is conducted in an international team, which common language is English.

3.3 Selection Procedure

The selection procedure describes the process of creating a set of relevant literature for classification. At first a list of papers is obtained from the selected databases by using the search query defined in section 3.1. As proposed by [17], in the subsequent step the list of papers is screened by applying the selection criteria. The result of the selection procedure is a number of papers that are the input for data extraction and classification.

The selection procedure is conducted by two assessors, in this case two graduate students. The search results are stored in a spreadsheet. This file is divided into two files and each assessor applies the selection criteria individually. In the next step, the files are exchanged for review. In order to ensure quality and correctness of the selection procedure, an expert reviewer evaluates the results by taking samples. An expert reviewer is also called in order to solve disagreements among the assessors.

4 Data Extraction / Classification

The data extraction strategy defines the necessary information to answer the research questions and how the information is processed. According to [10], the data extraction form should be defined and piloted during the design of the study protocol in order to reduce the possibility of bias.

Besides general citation information, five variables are extracted from each paper. The first variable is the type of publication based on the criteria defined by Montesi and Lago [12]. We use this classification because it is particularly designed for software engineering research. The second variable structures the papers based on the addressed field of sustainability. We limit the answers to the tree pillars listed in [3], although human or cultural sustainability are often considered as fourth pillar. The third variable captures whether the paper addresses the software process, the software product or both. We distinguish between process and product based on the demarcation described in section 1.2. The fourth variable is only extracted in case the paper addresses the development process aspect. It classifies the studies according to the phases of the software development life cycle. The fifth variable categorize the paper according to the effect of software on sustainability as defined by [9]. Table 2 illustrates the data extraction form and the mapping of variables to research questions.

Table 2. Data Extraction Form

Variable	Data	RQ
Title	Free text	
Author	Free text	
Venue	Free text	
Publication Year	Free text	RQ1
Type of research	Publishable papers: general $\oplus$ Extended versions of conference papers $\oplus$ Empirical research reports $\oplus$ Experience papers $\oplus$ Theoretical papers $\oplus$ Tutorial articles $\oplus$ Surveys $\oplus$ Short papers $\oplus$ Papers oriented towards practice $\oplus$ Opinion papers	RQ1
Sustainability Field	Economy $\vee$ Society $\vee$ Environment	RQ2
Aspect of Sustainable SE	Process $\vee$ Product	RQ3
Activities of SDLC	Requirements Engineering $\vee$ Design $\vee$ Implementation $\vee$ Verification $\vee$ None	RQ4
Effects of Software as Product	1^{st} order $\vee$ 2^{nd} order $\vee$ 3^{rd} order impacts $\vee$ None	RQ3

$\vee$ = logical or, $\oplus$ = exclusive or

The data extraction and classification is performed after the selection procedure. Therefore the spreadsheet is extended according to the form in Table 2. We use the same procedure as for paper selection. The spreadsheet is divided and filled out by each assessor. Afterwards they are exchanged for evaluation by an expert reviewer.

The analysis of the results focuses on presenting the frequencies of publications for each category. This shows which categories have been emphasized in past research and thus helps to identify gaps and possibilities for future research [17]. Graphical visualization is used to present the paper distributions by classification type, as this is considered an effective type of reporting mechanism [10].

5 Threats to validity

In order to guarantee valid results, we assessed the following threats to validity.

5.1 Internal

The construction of the search string is a risk to validity, since its accuracy and extensiveness determines if we are able to discover all relevant research. In order to ensure a good quality of the search string, we perform several pilots and use a quasi gold-standard to asses precision and sensitivity [20]. In the end we cannot ensure that our search string covered all research, but that our search and selection process is transparent and reproducible. Since the search string needs to be adapted to each database, this could result in different performances of the query. This risk is also addressed by piloting and evaluating the search string on each database. A further problem is that the study is conducted by student researchers with little experience in the field of empirical research and sustainable software engineering. This problem is addressed by consulting an expert reviewer that guides the research and checks the results. The a priori design of the study protocol eliminates major failures in advance.

5.2 External

Our research maybe objective to domain bias, which means that relevant literature is published in another field than computer science. Indeed during pilots, we discovered that relevant studies have been published in the different fields, e.g. Arts and Humanities. We address this problem by including multidisciplinary databases into our search strategy.

A publishing bias is not a major threat to validity for our study, since our results do not rely on evidence in published papers. Hence, an unpublished negative or minor result of a research paper does not have an impact on the results of this mapping study.

6 Evaluation

In order to preliminary asses the performance of the search string, a pilot on the Scopus database has been performed in April 2012. Table 3 shows the adapted search string. ID1 shows the plain search string without any refinements, which yielded 1055 results. By using the year of publication as exclusion criteria (ID2), the results could be reduced to 603 papers. By limiting the results to english (ID3), 593 papers were found. The results were evaluated against a previously defined set of ten papers that were considered to be relevant. This set has been manual created by looking at the references of articles. Scopus was only capable of finding four of the ten papers ([5], [13] [4] [15]). All four papers are listed in the results of the search string.

Table 3. Number of results yielded by the adapted search query.

ID	Predicate	Results
1	`TITLE-ABS-KEY(("sustainab*" OR "green" OR "viab*" OR "endurab*" OR "maintab*" OR "ecologic*") AND ("software engineering" OR "requirements engineering"))`	1055
2	`AND PUBYEAR > 2004 AND PUBYEAR < 2012`	603
3	`AND (LIMIT-TO(LANGUAGE, "English"))`	593

7 Conclusion

This paper describes the methodology and possible results of a systematic mapping study on sustainability in software engineering. It provides a preview of the research and thus helps to discuss and assess possible threats in advance towards a trustworthy, rigorous, and credible mapping study.

It can be seen that the definition of a thorough and extensive search string is not trivial and that it is an essential step of a systematic mapping study. A weak search string, would not be able to provide an overview of the field and thus the results of the study would not be reliable. The initially proposed search string was not accurate enough as it produced an unmanageable number of results. The bottom-up approach produced a simpler search string, but it imposes the risk of missing relevant literature. Therefore, a preliminary evaluation has been performed, which showed that the search string was able to find all papers of the control set. However, the control set is not complete and far from being a quasi gold standard. In a next step this quasi gold standard needs to be established by interviewing experts in the field of sustainable software engineering. This helps to further optimize the search string with respect to sensitivity and precision.

Since the whole process of a systematic mapping study is strictly iterative, the presented study protocol is not finished. Further pilots are necessary, which will uncover additional selection criteria and maybe help to refine the research questions.

It is planned to make the study protocol as well as the complete results of the classification available online and publish the results in a separate report.

References

[1] World summit outcome, resolution a/60/1. Technical report, United Nations General Assembly, September 2005.

[2] A. Abran, J. W. Moore, P. Bourque, and R. Dupuis, editors. *Guide to the Software Engineering Body of Knowledge*. Number 0-7695-2330-7. IEEE Computer Society, 2004.

[3] W. M. Adams. The Future of Sustainability Re-thinking Environment and Development in the Twenty-first Century . Technical report, IUCN, June 2006.

[4] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang, and C. Liu. Measuring the sustainability performance of software projects. In *2010 IEEE 7th International Conference on e-Business Engineering (ICEBE*, pages 369–373, November 2010.

[5] N. Amsel, Z. Ibrahim, A. Malik, and B. Tomlinson. Toward sustainable software engineering (NIER track). In *ICSE '11: Proceeding of the 33rd International Conference on Software Engineering*, pages 976–979. ACM Request Permissions, May 2011.

[6] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4), Apr. 2007.

[7] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham. Using mapping studies in software engineering. In J. Buckley, J. Rooksby, and R. Bednarik, editors, *Proceedings of the 20th Annual Meeting of the Pschology of Programming Interest Group*, volume 2008, pages 195–204, 2008.

[8] P. da Mota Silveira Neto, I. Carmo Machado, J. McGregor, E. de Almeida, and S. de Lemos Meira. A systematic mapping study of software product lines testing. *Information and Software Technology*, 2010.

[9] W. Goehring. The memorandum "sustainable information society". In P.Minier and A. Susini, editors, *Proceedings of the 18th International Conference "Informatics for Environmental Protection"*, pages 278–286., 2004.

[10] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *Version*, 2(EBSE 2007-001):2007–01, 2007.

[11] M. Mahaux, P. Heymans, and G. Saval. Discovering sustainability requirements: An experience report. In D. Berry and X. Franch, editors, *Requirements Engineering: Foundation for Software Quality*, volume 6606 of *Lecture Notes in Computer Science*, pages 19–33. Springer Berlin / Heidelberg, 2011.

[12] M. Montesi and P. Lago. Software engineering article types: An analysis of the literature. *Journal of Systems and Software*, 81(10):1694–1714, 2008.

[13] S. Naumann, M. Dick, E. Kern, and T. Johann. The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 1(4):294–304, Dec. 2011.

[14] B. Penzenstadler, V. Bauer, C. Calero, and X. Franch. Sustainability in Software Engineering: A Systematic Literature Review for Building up a Knowledge Base. In *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, pages 1–95, Feb. 2012.

[15] B. Penzenstadler and A. Fleischmann. Teach sustainability in software engineering. In *Proc. 24th IEEE Conference on Software Engineering Education and Training, IEEE Computer Society*, pages 454–458, 2011.

[16] B. Penzenstadler, B. Tomlinson, and D. Richardson. 1st international workshop on requirements engineering for sustainable systems. In J. Dörr, S. Fricker, and B. Paech, editors, *Proc. Requirements Engineering: Foundation for Software Quality - 18th International Working Conference (REFSQ 2012)*, volume 2 - Workshops, Doctoral Symposium, Empirical Tracks of *ICB Research Report 46*, 2012.

[17] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE 08)*. BCS, 2008.

[18] I. Sommerville. *Software engineering*. Number 0321313798 in International computer science series. Addison-Wesley, 8th edition, 2007.

[19] R. Wieringa, N. Maiden, and N. Mead. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. In *Requirements Engineering (RE 2006)*, 2006.

[20] H. Zhang, M. A. Babar, and P. Tell. Identifying relevant studies in software engineering. *Information and Software Technology*, 53(6), 2011.

e-Government - Implementing digital services

Jory Kits, Thom Koenders

Abstract— Software as Service(SAS) for the varying needs of Local e-Governments(SAS-LeG) is a joint project of the University of Groningen, Cordys [1] and local municipalities of the north of the Netherlands. Citizens demand more and more digital services from its local municipalities. This increasing demand makes it difficult for municipalities to guarantee a sufficient level of quality. SAS-LeG is focused on providing a solution to this problem, effective e-Government. SAS-LeG proposes to use the SAS principle in order to implement the law once and offer it as a customizable service to several municipalities.
To achieve this goal, SAS-LeG is developing a flexible and dynamic software solution. There are multiple approaches similar to the SAS-LeG solution, some of which are focused on run-time while others are focused towards design-time templates. The goal is to introduce variability to these solutions in order to enhance reusability, customizability and flexibility. However, since variability often goes hand in hand with complicated mathematics, understandability is introduced and added to the initial goals.
In this paper, we analyzed service flow modeling techniques which enhance variability. Subsequently, recommendations were given to the SAS-LeG project solution.

Index Terms—Software as Service, Variability, Procedural (imperative) and declarative mechanisms, Effective e-Government.

1 Introduction

Service flow technology is being applied in quite diverse areas. All of these areas have their own needs when it comes to service flow technologies. Therefore, it is important that the service flow technology is able to provide the required functionality to all the different areas. This calls for a dynamic and flexible service flow technology which enables easy and efficient usage.

Service flow systems have traditionally been used to facilitate production processes which are predefined and repetitive. Recently, in new domains, these sort of service flow systems have put forth the need for flexibility and dynamic specifications. However, this flexibility cannot be offered at the expense of control which is important when it comes to business processes.

The traditional approach to service flow systems is too static and rigid and therefore needs to be made more dynamic and flexible. This can be done by extending the traditional approach with extensions providing a more flexible character. There are multiple ways to accomplish adding flexibility. Approaches to this are documented in papers, from which we studied three. From these papers, the approaches are extracted. These approaches are analyzed and based on the results recommendations are made towards the SAS-LeG project.

1.1 SAS-LeG

The SAS-LeG project [3] proposes to use a Software as Service (SaS, also known as Software as a Service) approach for Local e-Governments (LeG). Currently, there are 418 municipalities in the Netherlands. All these municipalities have their own e-Government system. In the area of e-Government municipalities need to adapt to a new digital reality in order to keep up with the needs of citizen. The local government needs to implement national laws in their local context. Each time a law is implemented or changed all municipalities take individual initiatives to manage the required modifications in their systems. The SAS-LeG project proposes to use the Software As Service principle in order to implement national laws just once and offer them as a customizable service to municipalities. The goal is to provide municipalities with a framework to adapt their local systems while still implementing the national laws correctly.

Until today, a new law is introduced that is centrally decided. Every municipality has to implements the law. That means, understanding the law, adapting the information systems and making sure that it works correctly. The law needs to be executed by the local municipalities, so all the municipalities study the law and have to work out how the law should be implemented. This clearly shows that a lot of effort is done for designing the software system. The aim of SAS-LeG project is to develop software once and add a process model to the implementation to identify and design significant variety within the model. Such a process model is formal and can be executed by a machine.

The process models that are currently mainly used, typically have a prefixed way of working. The processes are completely defined and the designers know all the possible execution paths. Clearly, there is restricted variability in these models.

Currently, one of the process modeling standards is BPMN 2.0, [3], [2], which has no variability. The SAS-LeG project investigates whether variability can be added to such a standard.

1.2 Service flow

A service flow is a process specification language for a web service, which is equivalent to process models. Service flows enable to capture a web service within one scheme and enact its execution paths. Traditional service flows have adopted many concepts from classical service flow management systems. Typically, these languages are rather constrained and sealed and this does not fit with the autonomous nature of service flows, [9]. These service flows have restricted variability and its intended use is only directed to predefined execution paths of a flow. This makes it hard to impossible to customize or extend the service flows. In this paper possible techniques are shown and analyzed to enhance the opportunities of variability of service flows.

To clarify the use of service flows an example is visualized of the WMO (Wet maatschappelijke ondersteuning, Social Support Act, 2006) [6] The WMO law is providing citizens with support ranging from wheelchairs, help at home, home improvement and homeless sheltering. This process is found at one of the Dutch municipalities of the Northern region of the Netherlands. To illustrate this law a simplified version of this law is shown. The service flow is designed from the law and the corresponding service flow is depicted at Figure 1.

1.3 Procedural modeling and declarative modeling

To specify a service flow in detail two main modeling principles are distinguished, namely procedural (also known as imperative) modeling and declarative modeling.

The procedural model style is based on the execution of the main activity within a process. The focus lies on how the main activity

- *Jory Kits, E-mail: kitsjory@gmail.com.*
- *Thom Koenders, E-mail: thomkoenders@gmail.com.*

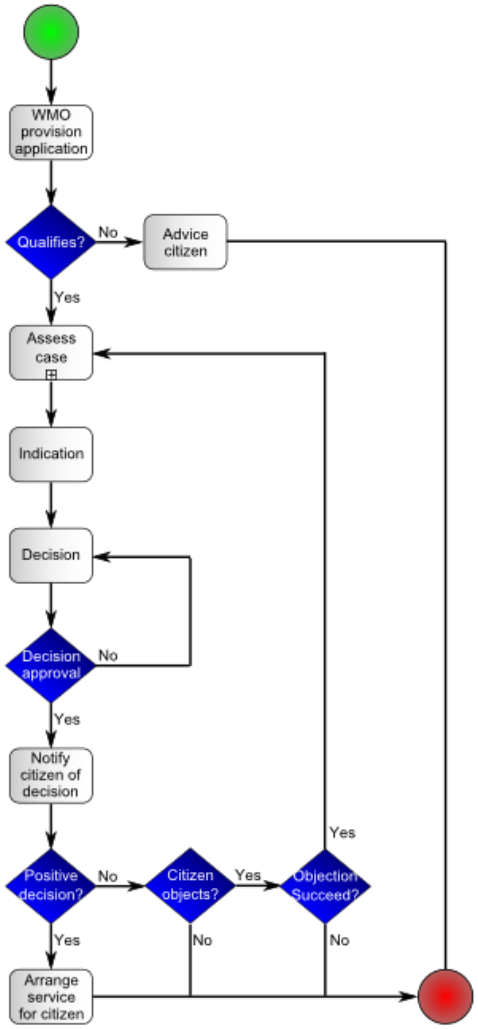


Fig. 1. WMO Service flow, [6].

is performed. If variations from the main activity occur within a process the procedural model style is difficult and inefficient to implement since constraints and assumptions should be added. The source of such restrictions lies in the fact that procedural approaches require all variations from the main process to be predefined, limiting variations to only those added explicitly. Procedural model style is considered to be coincided with design-time aspects.

While a procedural model style is oriented on how the main activity within a process is performed the declarative style is focused on what activities are performed, [6]. Thus, declarative style is more targeted on offering dynamic modeling, i.e., multiple activities can be executed multiple times without extra assumptions and constraints. The disadvantage of declarative frameworks are related to the semantic gap between the traditional and well understood way of procedural process modeling and the intuitive way of declarative modeling. Declarative model style is considered to be coincided with run-time aspects.

To elaborate on the difference between a procedural and a declarative style an example is used. Suppose that there are two activities A and B. Both can be executed multiple times but they cannot both be executed in one sequence, i.e., after the first occurrence of A it is not allowed to use B anymore and vice versa. The following execution sequences are possible based on this description: [ ], [A], [B], [A,A], [B,B], [A,A,A], [B,B,B], etc. In a procedural language it is difficult to specify the above process without implicitly introducing additional assumptions and constraints. In a procedural language one typically needs to make a choice with respect to whether no activities are to be executed, only activity A is to be executed or only activity B is to be executed. Moreover, the number of times A or B needs to be executed also has to be decided. This means that one or more decision activities need to be executed before the actual intended activity can be executed, [9].

1.4 Temporal logic

In logic, linear temporal logic or linear-time temporal logic (LTL) [5], [7] is a type of logic with modalities referring to time.

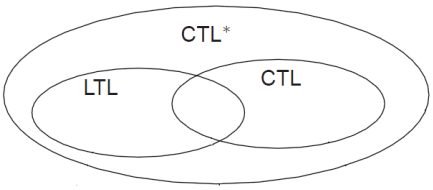


Fig. 2. Temporal logic.

CTL* (also known as CTL+) [4] is a superset of computational tree logic (CTL) and LTL. It freely combines path quantifiers and temporal operators. Like CTL, CTL* is a branching time logic. CTL* includes the TLT class, meaning that all properties of TLT can also be expressed in CTL*. In figure 2 the class inclusion is shown.

Temporal logic allows for a more declarative style of modeling. It includes temporal operators such as next-time ($\circ A$), eventually ($\diamond A$), always ($\Box A$) and until ($A \bigsqcup B$). Designing service flows with underlying temporal logic offers the possibility to specify, enact and monitor service flows. The language has the potential to be extensible; constructs can be added to any part of the service flow without changing the engine or semantical basis. However, a language with these operators is generally, especially for non-experts, difficult to understand.

2 Techniques using temporal logic

Declarative Service Flow Language (DecSerFlow) [9] and Process Variability - Declarative'n'Imperative (PVDI) [6] both use temporal logic to specify their own model language. The main advantage of these language is the understandability for non-experts (and experts) since the underlying complicated temporal logic is graphically visualized in order to keep the overview of the models preserved.

DecSerFlow is a declarative language model mapped onto LTL. A PVDI process is defined as a directed graph and can serve as a frame for a modal logic of processes, including CTL*.

By using temporal logic, constraints for processes are introduced, which allow to capture the basic meaning of a process in such a way that it controls changes within the process while keeping its essence intact as long as none of these constraints are violated.

3 Declarative Service Flow Language modeling

Developing a model in DecSerFlow starts with creating activities. The notion of an activity is like in any other service flow-like language, i.e., an activity is atomic and corresponds to a logical unit of work.

Relations between activities are referred to constraints in DecSerFlow. Each of the constraints represents a policy or a business rule. At any point in time during the execution of a service, each constraint evaluates to true or false. This value can change during the execution. If a constraint has the value true, the referring policy is fulfilled. If a constraint has the value false, the policy is violated. The execution of a service is correct at some point in time if all constraints evaluate to true. Similarly, a service has completed correctly if at the end of the execution all constraints evaluate to true. The goal of the execution of any DecSerFlow model is not to keep the values of all constraints true at all times during the execution. A constraint which has the value false during the execution is not considered to be an error.

To create an activity or constraints in DecSerFlow templates are used. Each template consists of a formula written in LTL and his corresponding graphical representation (These templates are derived of [9]). These templates define various types of dependencies between activities at an abstract level. Once defined, a template can be reused to add or respecify an activity or a constraint. It is possible to change, remove and add templates which makes the language flexible and extensible.

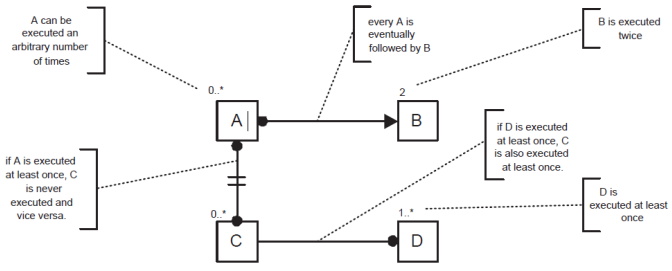


Fig. 3. DecSerFlow model, [9].

Figure 3 shows a DecSerFlow model consisting of four activities: A, B, C, and D. Each activity is tagged with a constraint describing the number of times the activity can be or should be executed. Each arc has a different meaning.

The arc between A and B is an example of a relation formula. It means that A is eventually followed by B, i.e., each time an A occurs somewhere in the execution sequence it has to be followed by a B eventually.

The connection between C and D denotes another relation formula. The relation formula means that if D is executed at least once, C is also executed at least once in the execution sequence.

The arc between A and C denotes a negation formula. If A is executed at least once, C is never executed and vice versa. Therefore, A and C exclude each other in the execution sequence.

Activity A and C can be executed an arbitrary number of times, i.e., both are able to be executed nil to infinite number of times. Activity B should be executed twice. B cannot be executed other than two times. Activity D should atleast be executed once and has no maximum number of executions, i.e., one to infinite.

For more information about constraints and cardinality of activities considering the DecSerFlow model formulas we refer to [9].

3.1 Addition to SAS-LeG

We consider the DecSerFlow technique to be a suitable template model for the SAS-LeG project. DecSerFlow is highly related to SAS-LeG since DecSerFlow proposes a solution to variability considering service flows. The variability is needed in the SAS-LeG project since municipalities need the opportunity to customize or extend their software solution. In addition, DecSerFlow can be used to specify, enact, and monitor service flows. This can be done by the editor that is created. Currently, the designers are experimenting with different ways in which they can build useful automatons for enactment [9].

4 Process Variability - Declarative'n'Imperative modeling

PVDI is based on both declarative and procedural (imperative) model language. Variability in PVDI process models are enhanced through the use of the PVDI framework which is based upon temporal logic techniques. Using temporal logic, constraints can be used for processes, which allows to capture the basic meaning of a process in such a way that changes can be controlled within the process while keeping its essence, its intended use, intact as long as none of these constraints are violated. Subsequently, it is possible to allow anyone to design a variant from such a template process without compromising its intended use.

In order to facilitate template design a number of graphic elements for the design process are used. These graphical-elements refer to activities or (sub-)processes. Constraints as embedded within templates are then generated from these graphical elements. For every element contained in the template, one or more CTL* formulas are generated. Of course, this process differs greatly per element and even per situation. Some of the simpler elements directly resemble simple CTL* formulas, whereas other more complex structures resemble a set of CTL* formulas.

4.1 Flow Constraints

Flow Constraints state that all elements from one set are followed by at least one element from another set in either a single or all paths. With a path being a series of consecutive transitions.

Fig. 4. Flow constrains, [6].

The graphical elements depicted at figure 4 describe flow relations over two dimensions; path and distance, [6]. The rows relate to the temporal dimensions; F (Finally) and X (neXt), which require the linked elements to either follow each other eventually or immediately. The first two columns relate to the paths; E (there Exists a path) and A (for All paths), which require the linked elements to follow each other in either a path or all paths respectively. The third column represents a negation of two of these flows. These flows are related to temporal logic formulas.

4.2 Parallel Constraints

Parallel Constraints enforce that two sets of states do not appear in the same path. Meaning that any series of consecutive transitions taken from any state in either set may never lead to any element from the other set.

4.3 Frozen Groups

Frozen Groups are sub-processes which cannot be modified. Such a restriction is achieved by generating a set of temporal logic formulas which constrain all elements inside of a frozen group.

4.4 Semi-frozen Groups

Semi-frozen Groups are Frozen Groups with less strict constraints, allowing for removal or replacement, addition, or moving of activities. The advantage of this group representation is that for example any activity inside a frozen group can be made optional. Therefore, it can be removed during the customization process.

4.5 Conclusion

Due to the approach taken with PVDI, both declarative and imperative techniques are being considered. In cases of imperative techniques a frozen groups is considered which ensures that the block is not customizable. On the other hand, semi-frozen blocks are introduced which keeps the process structure intact while allowing some modification which enhances declarative modeling.

5 Specification and validation of process constraints for flexible workflows

This method presents a foundation set of constraints for flexible service flow specifications. These constraints are intended to provide an appropriate balance between flexibility and control. The constraint specification framework is based on the concept of "pockets of flexibility". This allows ad hoc changes and/or building of service flows (i.e. workflows) for highly flexible processes. This approach provides the ability to execute a partially specified model. The full specification of the model is made at runtime. Because of this the model may differ at every execution.

In order to keep the work flow stable and prevent errors, verification is essential. The verification is performed by ensuring that the model conforms to specified constraints, which is not difficult to perform. A more interesting and challenging problem is ensuring that the constraint set itself does not carry conflicts and redundancy. Therefore this method also provides a discussion on both static and dynamic verification aspects. Also a prototype service flow engine, implementing these concepts, named Chameleon will be briefly presented.

5.1 Specification framework

The service flow should be dynamic and flexible however at the same time it is important that it stays reliable. Therefore, the right balance needs to be achieved in order to be able to provide both. This is achieved by having the following components in the service flow.

- Core process.
- Pockets of flexibility.

These components will be explained in order to make clear what is meant.

Core process A core process consists of one or more pockets. These pockets can be considered sub processes within the core process. Pockets consist of fragments and constraints providing the desired behaviour. All of the pockets combined form the process.

Pockets of flexibility The pockets of flexibility is the part of the framework that provides the flexibility. The pockets of flexibility represent the components of the service flow that are not predefined but are rather determined at runtime. What exactly happens in a pocket of flexibility is determined when this part of the service flow is actually used and is based on the required functionality. Pockets of flexibility consist of the following.

- Fragments.
- Build constraints.

These components will be explained in order to make clear what is meant.

Fragments The fragments provide the actual functionality of the service flow. Each fragment is responsible for performing a certain part of the service flow. All the fragments combined provide the functionality of the the service flow.

Build constraints Constraints are the building blocks that are used to ensure the service flow is performed in the correct manner. Constraints define how service flows are to be performed. This is done by controlling the fragments in a way that they are performed in the correct way. The constraints are further discussed in the upcoming subsection.

Figure 5 shows an example of a flexible service flow representing a call center response to a user request in a typical CRM environment. This service flow is defined as a directed graph. In this directed graph two kind of nodes can be found. The activity nodes are denoted with rectangle and represent an activity that is performed at that point. The coordinator nodes are denoted with ellipses and represent constraints.

This image is used to give a simple overview, therefore only basic constraints are used that already exist in currently used languages. The most interesting part of this image is the flexibility pocket which is denoted with a dotted line. In this flexibility pocket reside five activities. Depending on the situation and further constraints on the process, either one or multiple of these activities are performed. This is a clear example about the functioning of the flexibility pockets.

A precise explanation of this model can be found at [8] on pages 3 and 4.

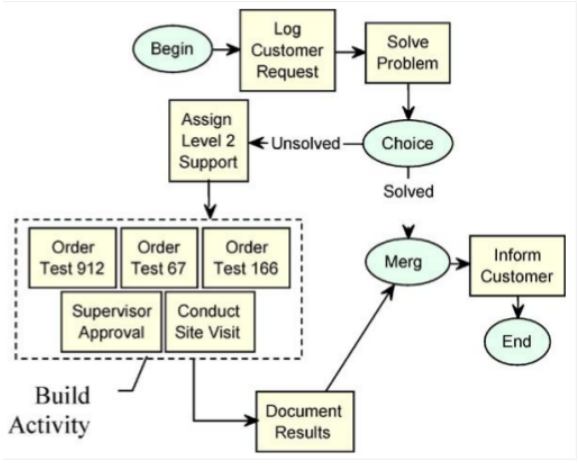


Fig. 5. Flexible service flow.

5.2 Constraints

There are two main classes of constraints identified, the structural class and the containment class. The constraints under the structural class impose restriction on how fragments can be composed in the templates. Three structural class constraints will be discussed:

- Serial constraint.
- Order constraint.
- Fork constraint.

Constraints belonging to the containment class define when fragments can or cannot be contained in the resulting templates. The two containment class constraints that will be discussed are

- Inclusion constraint.
- Exclusion constraint.

An elaboration on all of these constructs will follow in the upcoming paragraphs. It is also possible in some cases to use combinations of constraints.

Serial The serial constraint dictates that corresponding fragments are all performed. The idea is that order is not important as long as all of the corresponding fragments are performed. This is denoted by the operator S. Situation: There are two possible paths for the serial constraint of S{A, B}, AB or BA.

Order The order constraint is similar to the serial constraint in the way that both dictate that all of the fragments are to be processed. The main difference however is the fact that it needs to be done in a certain order. This is denoted by the operator O. Situation: O{A, B} results in the service flow AB.

Fork The fork constraint enables fragmentations to be executed in a fork. A typical application of the fork constraint is for example do no more than three of A; B; C or D. Execution in fork does not necessarily mean fragments are executed in parallel. Situtation: F{A, B} Results either in A or B or both, the outcome is user-driven.

Inclusion The inclusion constraint identifies a dependency between two sets, which means the presence, or absence, in one set has an impact on the way another set is executed. dictates that a certain activity will take place if other activities have been performed as well. This means g. e. when activity A and B are performed in one set, activity C in another set will be performed as well. If however activity A and B are not both performed, no rule is enforced on the execution of C and therefore C may or may not be executed. Situation: I({A}. {B, C}) results in, If an A from one set is performed, B and C from another set have to be performed. If however A is not performed, no rule is enforced on the execution of B and C. therefore B and C may or may not have been executed.

Exclusion The exclusion constraint is similar to the inclusion constraint, both constraints identify a dependency between two sets of fragments. Exclusion dictates that if the fragments in one set are performed, fragments in another set can not be performed anymore. For instance, this means if fragment A from one set is performed B of another set can not be performed anymore. If fragments A is not performed, no rule is enforced on the set which contains B and therefore B may or may not be performed. Situation: E({A}, {B}) results in, if an A from one set is performed B from another set may not be performed anymore. If however A from that set is not performed, no rule is enforced on the other set and therefore B may or may not be performed.

Extending the constraint set The five constraints that have been discussed here provide a basic set which can be used to build a substantial number of templates. However, it is always possible for organisations to have particular requirements that these constraints may not entirely satisfy. Therefore it is possible to extend the constraint set by adding additional constraints, focussed on solving the particular requirements. Extensions will allow further diversity in building templates, however introducing additional constraints will also have an impact on the verification.

Validation An important aspect of the given framework is the validation of the set of constraints. These constraints are the basis for controlling dynamic building. Therefore it is vital that the constraints themselves do not create restrictions that can not be met. This means possible structures under a given constraint should be acceptable. This means, for instance, that constraints should not be incompatible with each other or with the activities that are executed.

5.3 Verification

An important part of the framework is the verification of the constraints. The verification is done on two separate levels.

- Constraint validation
- Template verication

How these verifications are performed is discussed per level.

5.3.1 Constraint validation

Since the introduced constraints are the basis for controlling a dynamic service flow, it is vital that these constraints themselves do not impose any restrictions that can not be met. There are multiple aspects that need to be verified:

- Transitivity in constraints
- Redundant constraints
- Conflicting constraints
- Minimal specication

These verification aspects will now be discussed.

Transitivity in constraints Three of the discussed constraint types specifically inclusion, exclusion and order, have a transitivity property. This means, for example, when there is an order constraint over A and B, and there is an order constraint over B and C, that there is also an order constraint over A, B and C. Because of this transitivity property, the service flow can be made more efficient, therefore the service flow is being verified in order to discover transitivity in constraints.

Redundant constraints Redundant constraints result in inefficiency in the service flow due to a non-minimal specification. An example of this is the order constraints over A, B, C and the order constraint over A, B. A and B are performed two times in the same manner with the difference being that the first order constraint also holds C. Therefore the second order constraint holding A and B is redundant and can therefore be removed from the service flow, making it more efficient.

Conflicting constraints Constraints can conflict with each other resulting in a service flow that can not be executed. Therefore it is important that there are no conflicts between the constraints. An example of conflicting constraints is the order constraint over A, B and the order constraint over B, A. These constraints are obviously conflicting since they both claim the fragments should be executed in another order. Since it will not be possible to determine which order should be followed in this kind of situation, the service flow will not be able to perform correctly. Therefore, it is important conflicting constraints are not present in the service flow.

Minimal specication During the verification, it is important to keep in mind that there are multiple different aspects that need to be taken into account in order to make sure the service flow functions properly. Especially in larger service flows, it is far from uncommon to encounter multiple aspects that could go wrong. In order to keep sufficient overview over the service flow, it is important to keep the specification as minimal as possible. This is done by finding the minimal representation for each of the constraint types. This will result in an equivalent but minimal redundancy and conflict free specification.

5.3.2 Template verication

There are three types of verification.

- Syntactic verication
- Structural verication
- Template verication

Typically verification in service flow specification only provide the first two verification methods. The template verification takes verification verification further due to the verification mentioned in the constraint verification.

Syntactic verication During the syntactic verification, the model is verified with respect to the grammar of the language. This means the service flow should be built according to the rules of the language.

Structural verication There are two possible errors under the given language, deadlocks and lack of synchronization. However, because of the restrictions that are imposed on the language of the template, these errors do not arise.

Template verication The introduced constraints given earlier provide a means by which requirements for a highly flexible processes can be captured. Therefore it is important that the model built conforms to the given constraints.

5.4 Addition to SAS-LeG

The approach that was discussed here has a lot to offer to the SAS-LeG project. The strong framework that this approach offers is a good feature and therefore a good recommendation towards the SAS-LeG project. A strong framework is easier to manage and therefore less error prone. Therefore this strong framework could be a valuable addition to the SAS-LeG project and is therefore a recommendation.

Apart from that the verification is also a very important part of this approach that could mean a lot to the SAS-LeG project. This approach offers a very thorough and efficient way of verifying. This is done by moving through multiple levels of verification. These multiple levels of verification each provide a certain way of verifying, together offering a complete verification. Because of this thorough verification, it is easier to make sure the service flow functions as required.

6 Conclusion

The SAS-LeG project is focused on developing a software solution for the increasing demands of citizen for digital services, e.g., gathering information and applying for a passport, from its local municipalities. The three analyzed process modeling methods were considered in order to determine recommendations for the SAS-LeG project. To determine recommendations criteria are used in the form of understandability for non-experts, reusability, customizability and flexibility for both design-time and run-time.

The PVDI and DecSerFlow modeling methods have relative high understandability for non-experts because of the graphical modeling opportunities which is grounded by temporal logic. In addition, both model specification languages enhance variability when designing a model.

The DecSerFlow project developed an editor which supports DecSerFlow models. The editor allows users to specify service flows. Moreover, the user can add user-defined constraint templates by simply selecting a graphical representation and providing parameterized semantics in terms of temporal logic.

PVDI has procedural and declarative modeling options and by introducing constraint blocks it is able to handle and implement precise processes. These constraints enhance understandability for non-experts, since the overview is conveyed and potential options can be predefined, though, they are not mandatory.

The strong framework that Specification and validation of process constraints for flexible service flows offers is a good foundation for service flows. Strong frameworks are easier to manage and thus less error prone. Therefore, this strong framework could be a valuable addition to the SAS-LeG project.

ACKNOWLEDGEMENTS

The authors wish to thank their loved ones for their unconditional support throughout the process. It is highly appreciated and we could not have done it without you!

REFERENCES

[1] Cordys. http://www.cordys.com.

[2] Object Management Group. Business Process Model and Notation. http://www.bpmn.org/.

[3] SaS-LeG. Software As Service for the varying needs of Local egovernments. http://www.sas-leg.net/.

[4] E. A. Emerson, E.A., and J. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Proceedings of the fourteenth annual ACM symposium on Theory of computing.*, pages 169–180, 1982.

[5] D. Giannakopoulou and K. Havelund. Automata-Based Verification of Temporal Properties on Running Programs. *In Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE01) IEEE Computer Society Press, Providence*.

[6] H. Groefsema, P. Bulanov, and M. Aiello. Declarative Enhancement Framework for Business Processes. In *9th Int. Conference on Service-Oriented Computing (ICSOC-2011) http://www.cs.rug.nl/ aiellom/publications/icsoc11.pdf*, pages 495–504, 2011.

[7] K. Havelund and G. Rosu. Monitoring Programs Using Rewriting. *In Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE01) IEEE Computer Society Press, Providence.*, pages 135–143, 2001.

[8] S. W. Sadiq, M. E. Orlowska, and W. Sadiq. Specification and validation of process constraints for flexible workflows. *School of Information Technology and Electrical Engineering, The University of Queensland, Qld 4072, Australia Corporate Research, SAP Australia Pty Ltd, Brisbane, Australia*, 2003.

[9] W. van der Aalst and M. Pesic. DecSerFlow: Towards a Truly Declarative Service Flow Language. *Department of Information Systems, Eindhoven University of Technology*.

Transparent Device Interoperation: A Local Cloud Protocol

Avdo Hanjalic and Fotis Katsantonis

Abstract— Nowadays people own many devices that allow them to perform the same and/or related tasks, e.g. PC's, notebooks and smartphones. Still performing a task is typically done by using one device at a time; Device interoperation and collaboration are not addressed properly in everyday life. To allow multiple users to (concurrently) use hardware over a Local Area Network (LAN), a Transparent Device Interoperation (TDI) protocol is needed. We explain why interoperability, security and efficiency are Key Drivers for any TDI protocol architecture. We propose a TDI protocol architecture by incorporating currently available techniques, used in various research areas for service discovery, authentication and hardware sharing. Finally we determine how our protocol architecture satisfies the Key Drivers, by applying the Force Resolution Map [10] evaluation model.

Index Terms—Transparent Device Interoperation, Interoperability, Local Cloud, Device Collaboration, Hardware Sharing

1 Introduction

Over the past years many new devices became commonly available to people. The penetration of portable smart-phones and notebooks has increased drastically. Not only portable devices are evolving, televisions and washing machines among many others are changing into smart devices.

Each device has a certain purpose, capabilities, typical way of interaction, but also specific technical limitations. Ideally it would be possible to perform the same task on many different devices, combining their capabilities to achieve the desired goal in a more efficient way. Sadly the practice still forces the user to select one device to complete the task. With the rise of (commercial) cloud services results of work can be easily synchronized, however this is typically possible *after* a task is completed. Using capabilities of different devices concurrently, for the sake of performing the same task, is an unfamiliar approach in every day life. The main reason for this is the absence of an universal (device) protocol that enables interoperation of all our devices.

In this paper we explain what such a protocol should look like. In order to do so, we first attempt to categorize the issue. No computing science field exists to cover the whole issue, however parts of the problem are covered by Pervasive Computing, Cloud Computing and High Performance Computing. For this reason we review techniques coming from the various research fields and attempt to combine them in order to achieve an universal protocol for Transparent Device Interoperation (TDI).

The rest of this paper is organized as follows: In the next section we analyze and decompose the problem. We start Section 2 by sketching some scenarios to emphasize subproblems, followed by an extraction of Key Drivers. Subsequently we briefly review related techniques, protocols and applications in Section 3. To counter the problems posed in the Analysis, in Section 4 we propose a protocol that will gracefully incorporate the techniques discussed in the related work. The design is then reviewed in Section 5 as follows: First we assign weights to each key driver, then we use a Force Resolution Table [10] to express the effectiveness of our architecture in terms of each Key Driver. Finally, we briefly discuss our proposal and its evaluation to expose weak points and therewith possible future work in Section 6.

- *Avdo Hanjalic is MSc. Computing Science student at the University of Groningen, E-mail: a.hanjalic@student.rug.nl.*
- *Fotis Katsantonis is MSc. Computing Science student at the University of Groningen, E-mail: f.katsantonis@student.rug.nl.*

2 Analysis

To give an example of the problems just described we look at a university or workplace environment. In such an environment intranets providing various services are available. Examples of such services are printing, browsing a map of the building/floor and locating the office of someone. Users often use mobile devices for accessing such services. Their concerns in such scenarios include: efficiency, security and battery saving. Moreover, privacy could be of interest in public environments. Another benefit of such a pervasive environment is reduction of costs: Mobile devices could be provided by the university, featuring minimal hardware (screen, input and networking). The tasks can be executed by the university's infrastructure and the results shown on the mobile device.

Another example environment for using a TDI protocol would be in a (smart) house. If the user wants to view an image from his mobile device on the television, he would have to: Connect to the computer, synchronize the device with the computer, connect with the television and send the file to be displayed. With a TDI protocol, once the mobile client authenticates with the local server he will be able to show the file directly from his mobile phone on the television. For this only authentication of the mobile phone and selection of the render device is necessary, which is the television in this case.

2.1 Problem Decomposition

The first problem that must be overcome is interoperation of devices and therewith the transparency of used protocols. The syntactic heterogeneity of interoperation protocols, along with diversity in their layers, currently prevents applications to use any available equivalent device on the network to accomplish a task. For the interoperation to be transparent, capabilities of devices from different manufacturers must be recognized as the same. For example, an user should not be concerned with the communication protocol of a printer, he simply wants to print paper.

Once an environment of ubiquitously connectible devices is achieved, a major requirement for shared environments has to be addressed: Humans require the possibility to use the devices in a secure and preferably non-traceable way. For instance, a user does not want his holiday pictures to leak to his colleagues because he is displaying them on a shared screen.

For people to delegate tasks to other devices, increased efficiency of task execution is the primary motivation. Traditionally efficiency was measured as computational performance, i.e. in the amount of computations per second. However, due to increasing dependence of humans on mobile device functionality in everyday live, power consumption has become a very relevant property of portable devices today. Delegation of computations to other machines will also provide the opportunity to run applications otherwise impossible to run (due to hardware limitations). Furthermore the protocol must increase perceived speed of the used device and/or prolong battery life. If setting up a TDI connection takes a long time, or the user interface starts lagging due to usage of the protocol, the protocol will render itself useless.

2.2 Capability Requirements

Nowadays consumer peripheral devices are typically provided with an Universal Serial Bus (USB) connector. Moreover, most integrated peripheral devices, e.g. webcams and touchpads in notebooks, are internally connected through USB. For this reason it is unavoidable to integrate USB device sharing in any TDI protocol.

Since decades computer output is provided visually to users. The user experience of an device/application heavily depends on the Graphical User Interface (GUI). GUIs have improved over time and lately most GUIs require graphical hardware acceleration, which is provided by the Graphics Processing Unit (GPU). Not only contemporary personal computers and notebooks, but even devices with relatively weak hardware, e.g. mobile phones, are equipped with GPU's that have enough processing power to fluently render a complex GUI. As the GPU is responsible for rendering the images that the user has to interpret, it is a component of major importance. Moreover, GPU computations are a heavy burden on the energy consumption of a device. Delegation of GPU intensive operations from portable devices to dedicated machines can improve user experience, while battery life can be extended. Hence this must be supported by any TDI protocol.

2.3 Key Drivers

The Key Drivers are assessable quality attributes, indicating where the focus of a system's design lies. The Key Drivers are derived from the analysis, which describes the main constraints of a TDI protocol. Therefore these Key Drivers can be used for verification of any TDI protocol proposal. A listing and explanation of them is given below:

Interoperability of devices of different vendors is the primary requirement for people to utilize their capabilities concurrently. If this is not achieved, collaboration of devices will be impossible.

Security of communication is necessary for people to use devices shared to many. If this is not achieved, the protocol is doomed to fail adoption in public environments. For public environments user privacy is also of some concern.

Efficiency must be taken seriously, as device collaboration must reduce the amount of resources (e.g. time, energy) needed to perform a task. If this is not achieved, the protocol would beat its purpose and render itself useless.

Capability extension of different devices is a result of both Interoperability and Efficiency. Interoperability enlarges the set of capabilities by 'lending' these from other devices. Efficiency magnifies the potential of capabilities, which increases the set of possible computer applications.

Energy saving is another possible result of combining Interoperability and Efficiency. If delegation of tasks to other devices (Interoperability) is performed efficiently, the energy consumption of the client device can be reduced significantly.

3 Related work

Recently approaches have been developed to address parts of the various issues. The main ideas of related work, form which the building blocks of our proposed system emerge, are discussed next. With the context of this paper in mind, these ideas are refined and molded together in the next section.

3.1 Interoperability

To counter the problem of device interoperability, generating proxy modules published as Universal Plug and Play (UPnP)[1] standard devices for controlling non-UPnP devices is proposed in [4]. They combine the UPnP, Intelligent Grouping and Resource Sharing (IGRS)[2] and Device Profile for Web Services (DPWS)[3] protocols. All mentioned protocols use the Simple Object Access Protocol (SOAP)[4] for interaction. UPnP and IGRS use the Simple Service Discovery Protocol (SSDP) [3] for discovery and General Event Notification Architecture (GENA) for eventing, while DWPS uses a set of standard web service protocols.

To improve device interoperability, El Kaed et *al.* propose grouping other protocols (IGRS and DWPS) under UPnP using a proxy. The protocol finds corresponding services and attributes between devices supporting different protocols and groups them under a generic UPnP proxy. Using this proxy, discovery of new devices and their services can be done only using UPnP which eliminates the complexity of multiple protocols.

Many technologies for the purpose of sharing devices are proposed in the pervasive computing area, however these techniques typically impede performance and/or lack transparency as their implementation depends on the underlying Operating System. USB/IP provides a solution to this problem, as described by Hirofuchi et *al.* in [5]: They propose to use a Virtual Host Controller Interface driver, implemented as a peripheral bus driver.

3.2 Security

To address the problem of security of users, an authentication and key establishment protocol is proposed in [9]. The protocol is implemented in the application level. It provides mutual authentication between the two parties (user and service), while preserving the anonymity of the user. The privacy (anonymity) is a "sub-concern" of security. This is accomplished using two cryptographic primitives, blind signature [7] and hash chain [1].

3.3 Efficiency

The approach used for USB peripheral sharing cannot be applied to GPU's as these are highly complex and evolve constantly, while documentation of concrete inner functioning is barely or not available at all.

An approach to GPU virtualization is described by Dowty et *al.* in [2], which is applied in the context of machine virtualization. Though the context of Virtual Machines is different from the context in this paper, the described approaches (front-end, back-end and hybrid virtualization) and related techniques can be applied to a TDI context.

A technique for GPU sharing in cloud environments is proposed by Ravi et *al.* in [8]. Here they focus on how GPU processes can be consolidated to achieve maximum utilization of GPU hardware. The described techniques are Space sharing and Time sharing of GPU resources, which become relevant as soon as more than one task is performed simultaneously by the GPU.

4 Protocol Architecture

In this section we provide our high-level architecture proposal of a TDI protocol. After careful consideration of the requirements defined in Section 2 we have derived the sequence of steps required for TDI be performed. An overview of the steps is given below, which are explained in more detail in the following subsections.

1. **Authentication:** Local clouds can be private property of a person, but also of a company. Hence authentication with the cloud must be performed first to prevent unprivileged users to access information about the system. Authenticated users receive an authentication token, which grants them access to devices shared to their privilege level. This step relates to security.

2. **Device Discovery & Selection:** Once an authorization token is obtained from the previous step, discovery of services (shared devices) must be performed by the users device to get a list of

[1] Universal Plug and Play protocol, `http://www.upnp.org/`

[2] Intelligent Grouping and Resource Sharing, `http://www.igrs.org/`

[3] Devices Profile for Web Services, `http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01`

[4] Simple Object Access Protocol, `http://www.w3.org/TR/soap/`

accessible capabilities. The device to be used has to be chosen by the user or application, driven by current need. This step relates most to interoperability.

3. **Application Execution:** Once all previous steps are accomplished, the remote device(s) are considered local by the client. Applications are able to invoke other devices' capabilities, without modifications to the application source code. For this purpose hardware requests/replies have to be routed during program execution. For execution of applications this step is the most important one. The implementation of this step has strong influence on efficiency.

4.1 Authentication

Before an user can see the list of available devices and services he has to authenticate first. The authentication protocol consists of two protocols, the user authorization covered by steps 1 and 2 and the user operational protocol covered by steps 3 and 4.

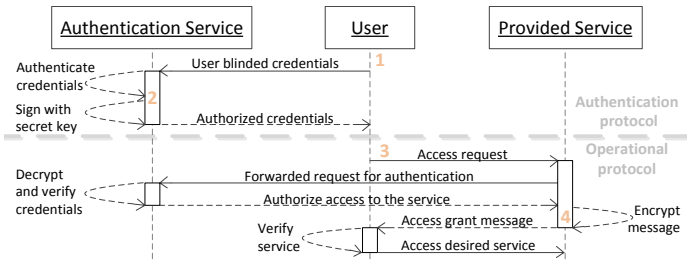


Fig. 1: Security protocol sequence diagram

Figure 1 shows the overall authentication process which we will now explain more elaborately:

1. **Authentication request:** First the user generates two fresh nonces[5]. He signs his own identity with one of the nonces and computes the anchor value C^0 of the credential chain with the signature. The signature in C^0 has a non-repudiation property due to the fact that only the user can generate it and the fresh nonce guarantees its freshness. Then a credential chain of length n is computed by a hash operation. After that the user blinds the credential chain tail C^n using the blind signature technique. The signature, along with a user self-signed certificate and a service type identifier is sent to the authentication server.

2. **Credential verification:** User credentials are first checked. If the credentials are valid, the server signs C^n with the private key for the services that the user is entitled to and sends it back to the user. During a single protocol cycle, the user is allowed to submit multiple service requests, thus reducing overhead.

3. **Service request:** Using the credentials acquired from step 2, the user can request services. The user sends an access request to the service. The service in turn delegates this call to the authentication server. The authentication server verifies the authenticity of the user's credentials and also ensures that the user has access to the requested service. A positive or negative response is sent to the service. This response includes decrypted information of the user it verified.

4. **Access grant:** Continuing from step 3, the authorization message is then sent back to the service, which generates two session keys. The service encrypts the obtained secret information and its own identity with one of the session keys just generated. It sends this encrypted information along with a shared encryption key. The user receives the access grant message, decrypts it using the shared encryption key and then authenticates the service. This is possible by comparing the user's secret information of the message received from the service, to his own original secret information.

It is important to note, that during this authentication process, the authentication server does not know who the user is, except for the requested service type. This is possible due to the properties of the blind signature scheme.

This authentication protocol eases the life of the network administrator, as he can group users in user groups, assigning access rights to the groups instead of each individual user.

4.2 Device Discovery & Selection

For the discovery of devices the UPnP protocol is used in the way described in Section 3.1. The process is split into 6 steps:

1. **Ontology generation:** An ontology is generated automatically, derived from the device description.

2. **Ontology alignment:** Correspondences are found between the ontologies in a semi-automatic manner. This is based on a heuristic method.

3. **Alignment validation:** Due to the heuristic method used in step 2, expert[6] validation is required.

4. **Pattern detection:** This is applied to assist the expert in step 3. Using pattern detection rules, automatic classification of fully compatible actions is possible which will ease the work of the expert up to some extent.

5. **Alignment adaptation:** Having the ontologies that need adapting from steps 3 and 4, the actual adaptation is performed in this step by referring to the specifications: Default values, data and code adaptation.

6. **Proxy generation:** Automatic generation of the proxies using the validated ontology alignments. The alignments essentially are rules for going from a non-UPnP device description to a UPnP standard device description, using a proxy. This is based on Model Driven Engineering (MDE), which is a software development method whose main idea is to abstract a domain with a high level model and then transform it into a lower level model, until the model can be made executable.

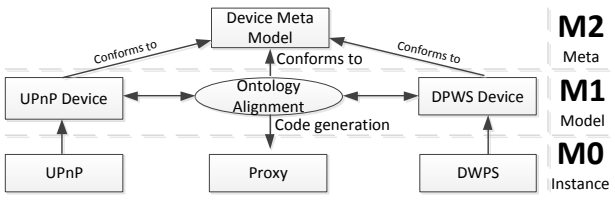


Fig. 2: Proxy generation in detail

Figure 2 shows a more detailed diagram of the approach used. M0, M1 and M2 represent the different MDE layers, M2 is abstract and as we move towards M0 it becomes concrete. M0 represents the heterogeneous plug and play device descriptions. UPnP, DPWS and IGRS use different description semantics. The automatic generation of ontologies lifts the device description from M0 to M1 layer, which corresponds to step 1. Each ontology represents a device using unified concepts in conformance to the meta model based on UPnP in the M2

[5] Nonce is a security term, describing a "key" that is unique. This often is a combination of time stamp and something else, that is unique.

[6] A human being that has knowledge of ontologies and can verify their alignment.

layer. This meta model describes devices this way: every device has one or more service, every service has one or more action and each action has variables.

To resolve the heterogeneity in M1 layer, the ontologies are automatically aligned, which is step 2. For example if a UPnP and a DWPS printer are aligned, ”setClock()” and ”setTimer()” methods will automatically be matched.

After the automatic alignment, expert intervention may also be required as described earlier. Along with the expert, pattern recognition techniques are applied to aid the expert in the aligning process. These are steps 3 and 4 as described earlier. Then the alignment adaptation is performed according to the results of steps 2-4, which is step 5.

Last, the automatic code generation techniques allow the ontologies to go from an independent technology representation in the M1 layer to an executable proxy in the M0 layer. The proxy transfers the received invocations to non-UPnP devices. This is step 6.

This approach is not based on a central ontology, but establishes correspondences between existing ontologies. Human interaction is needed for validation of the established correspondences which is a disadvantage. However tackling the issue with a centralized ontology will inevitably lead to disaster, since it is too optimistic to try to group all devices under one model. The establishment of correspondences provides more capability extension. Therefore the solution provided is implementable and has been proven to work with a UPnP and a DWPS printer as explained in [4].

Once the user retrieves the list of available devices and services, he has to pick a device to perform/delegate the task. An example list of discovered devices can be seen in Figure 3. The server device, to which tasks will be delegated, can be selected in multiple ways: The user could select the device manually, for example if he wants to play a movie on a specific screen. Also applications might want to be aware of devices shared through the local cloud, in order to set up on-demand connections, based on the application context. This is an issue common to pervasive computing applications and is out of the scope of this paper. The third possibility is to use the categories described in step 2 and connect to the device that provides best service, in terms of efficiency[7], for this category.

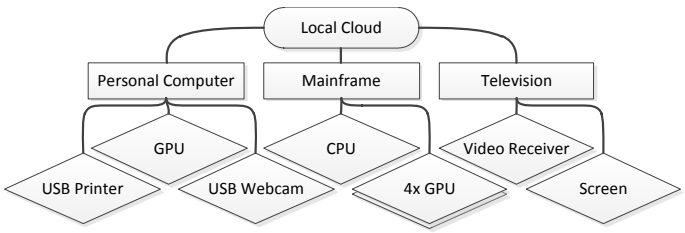


Fig. 3: Shared devices in Local Cloud

The available bandwidth/latency of the communication link (e.g. WiFi, Gbit LAN, IP over USB) with the device is measurable and should meet the least required values, defined in the UPnP descriptor of the remote device. If for example a graphics card is to be added to the local device, the requirement of a Gigabit network connection is to be expected. On the other hand, for a remote web-cam a WiFi connection should be sufficient.

4.3 Application Execution

Once the remote device is selected, the client has to add it as a virtual device in its list of attached hardware, as depicted in Figure 4. This way applications can access remote devices as if they are local. When a communication link is obtained, encapsulation of hardware commands has to be performed, to be routed over Ethernet. Moreover, requests coming from different clients have to be multiplexed, in order to guarantee fairness and increase performance. For this purpose we

[7]Efficiency can be measured in several contradicting ways, therefore the exact unit has to be defined by the user or context.

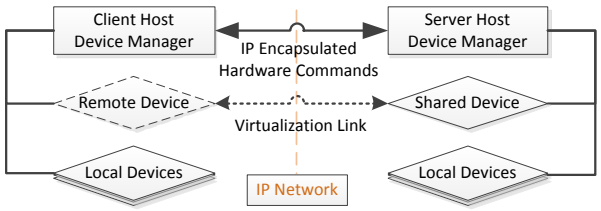


Fig. 4: Interconnection of remove device

distinguish approaches for USB peripheral devices and the GPU, due to fundamental difference in their hardware operation protocols.

4.3.1 USB Peripheral Sharing

A USB Virtual Host Controller Interface (VHCI) driver was proposed in [5], to be implemented as a peripheral bus driver. The USB Host Controller Driver (USB HCD) communicates on the lowest possible level with the hardware and provides an API to higher level drivers like USB Per Device Drivers (USB PDD). As the communication with the physical devices implements the USB protocol, it does not depend on the Operating System. However, every Operating System requires PDDs for communication with the peripheral.

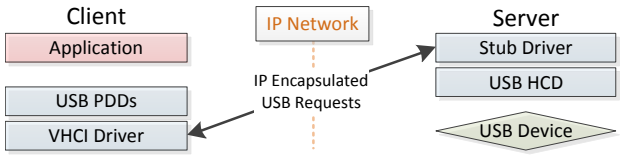


Fig. 5: USB/IP Sharing, based on [5]

.

This idea is depicted more elaborately in Figure 5. On the client a VHCI driver is installed, which appears as the usual USB HCD to the system. On the server a stub driver is installed that passes the received requests to the USB HCD having ’physical’ access to the hardware. Assuming the drivers are installed as part of the protocol and a connection is obtained in the previous steps, the operational procedure looks as follows:

1. Client application invokes USB peripheral, through PDD.
2. PDD instructs the VHCI to command the peripheral. VHCI encapsulates the USB commands in IP packets and transmits them to the server.
3. Server passes the commands to the physical hardware and routes its replies back to the client. If multiple clients request for the same device, their commands are multiplexed where possible.
4. The VHCI receives the replies and passes them to PDD, as if they were generated by locally attached hardware.
5. Client application receives requested data from USB peripheral.

Due to heterogeneity in the way USB devices are used, they must be distinguished by operation mode. USB supports four modes of operation, which are implemented by a separate data transfer type.

Control is used to operate functions of peripheral devices, e.g. requesting a disk for data. *Bulk* (asynchronous) mode is used by devices to return data on demand, e.g. data replies from a disk. *Isochronous* mode is needed for devices that deliver data periodically, e.g. a webcam. *Interrupt* mode is used for Human Interface Devices (HID), e.g. to deliver a keystroke when input is applied to the keyboard.

For Isochronous and Interrupt modes timing is critical, however this issue appears solvable without much effort. The issue is mostly implementation related, which is out of scope of this paper. For more details we refer to [5].

4.3.2 GPU Sharing

Two distinct approaches were proposed in [2, 8] to obtain GPU sharing. The first method focuses on sharing the GPU among many virtual machines and achieve maximum performance. The second technique is developed for cloud environments, where the GPU is used for general purpose computing, rather than rendering images. For this environment maximum GPU utilization is more relevant than the performance of an individual process. We combine the techniques of mediated pass-through with workload consolidation for our solution. In order to understand the GPU sharing method explained in this section, some knowledge of the GPU is necessary.

Basics

A GPU (Graphics Processing Unit) is the device responsible for rendering images, which are usually shown on a screen to the user. The device is highly capable of multiprocessing, and unlike a CPU (Central Processing Unit), follows a Single Input Multiple Data (SIMD) approach. The GPU contains many cores, nowadays 2048 'stream cores' are provided by the AMD Radeon 7970. Moreover, it is supported by a large amount of Random Access Memory (RAM). Loading resources to the GPU's RAM is a costly operation, therefore most resources are loaded before process execution, even though they might not be used.

Consolidated pass-through

Initially the same approach is followed for GPU sharing as for USB/IP sharing. As the instruction set of a GPU is more complex than of an USB peripheral, more sophisticated multiplexing is necessary. For this purpose a Resource Manager is introduced, which replaces the 'stub driver' from USB/IP. Furthermore, the resource manager is supported by a Workload Consolidator.

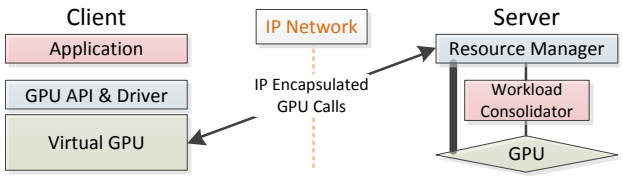


Fig. 6: Consolidated GPU pass-through, based on [2]

The topology of the components is depicted in Figure 6. The operational procedure is derived from [2], however the mediated pass-through is extended with workload consolidation from [8].

1. Client application invokes GPU through API & Virtual GPU.
2. Virtual GPU forwards calls to Server, indicating call type.
3. Server Resource Manager analyzes call type.
   - If resource allocation is needed, call is forwarded to GPU.
   - If execution is needed, multiplexing is performed by Workload Consolidator and calls are forwarded eventually.
4. Resource Manager routes GPU replies back to client.
5. Client application receives requested data from Virtual GPU.

The Resource Manger [2] keeps track of GPU memory usage and passes resource allocations directly to the GPU if possible. However, execution time is the scarcest resource.

The Workload Consolidator [8] investigates the currently active processes and attempts to mold them together. It analyzes the minimum amount of cores needed for each process and attempts to fit them together so that the processes are executed simultaneously. This is also known as space sharing. If molding is not possible due to requirements of running processes, the GPU will be shared by scheduling the processes, also known as time sharing. This approach is less resource effective but necessary if space sharing is not possible.

5 Evaluation

In this Section the protocol described in Section 4 will be evaluated. First the impact of each step on the Key Drivers (Section 2.3) is discussed, then we apply the FRM evaluation method on our protocol architecture. Finally, the Key Driver weights and FRM values are compared to assess how well our architecture covers the issue.

5.1 Key Driver Weights

The proposed protocol exists of 3 steps, though the steps do not affect the Key Drivers in equal proportions. Before the systems total impact on the Key Drivers can be calculated, the impact of each step on every Key Driver is discussed and rated with a factor between 0 and 1. For each step the factors sum up to 1 in total.

Step 1: Authentication

Only authorized users must be able to access the system. Security measures should be taken to prevent attacks, thus we assign it 0.6. Interoperability is the second most important Key Driver, as devices need to connect before they can authenticate. The authentication server needs to support different device and connection types to increase interoperability. Therefore we assign it 0.3. Efficiency is not very important for this step, therefore we assigned 0.1. It still needs to be taken into account, else it could prove a bottle-neck for the protocol.

Step 2: Device Discovery & Selection

The ability to discover and communicate with all devices available is crucial for this step. Therefore interoperability is the most important Key Driver and we assign it 0.7. Security is still of concern for this step, as devices should only be connectible with proper privileges. As security is main concern of the previous step, we assign it a value of 0.2. Efficiency is the least important key driver for this step. Though it is not crucial, it must be taken into account.

Step 3: Application Execution

One of the main goals is to enhance the user experience, which is affected mostly by this step. Hence efficiency is of major importance. However, interoperability is important for being able to use the peripherals. Therefore both efficiency and interoperability are assigned a value of 0.4. Again security is assigned 0.2, for the same reason as in step 2.

Overview of Weights

Below we summarize the discussed values for a better overview. The total indicates how the Key Drivers should be prioritized. Altogether, interoperability is the most important key driver, followed by security and last efficiency. All of them need to be taken into account and the focus differs for each step.

Step	Interoperability	Security	Efficiency
1. Authentication	0.3	0.6	0.1
2. Discovery & Selection	0.7	0.2	0.1
3. Execution	0.4	0.2	0.4
Total	**1.4**	**1.0**	**0.6**

5.2 Architecture assessment

To assess how the used sub-protocols effect our protocol architecture, we evaluate them one by one, using the FRM evaluation model. The effect of the sub-protocols is summarized in an overview table at the end of this section. There we explain what this means for the whole protocol architecture.

Authentication protocol

This protocol does not affect interoperability (0), which is out of scope. It provides features such as mutual authentication, user context privacy, non-linkability and data confidentiality. Therefore security and privacy are greatly increased (+2). The protocol needs two rounds to accomplish mutual authentication and session key establishment,

which is the least amount of rounds possible for any authenticated key establishment protocol [9]. The computation overhead is minimized since part of it can be done off line, which saves bandwidth. Even though the proposed protocol is highly efficient, it still has a negative impact on efficiency as it does cause overhead. For this reason we assign (-1) for efficiency. The performance is further compared in [9].

Device discovery protocol

This protocol greatly increases interoperability (+2), by proxying various protocols to UPnP. Security and privacy are not addressed by this protocol and therefore graded (0). The generation of a proxy for a DWPS printer on a PC took 187 seconds according to the tests described in [4]. This reduction can be even greater in the case the proxy for a complex device (e.g. a media center) does not exist and needs to be generated. However, the proxy generation is performed only once for each new device; when the proxy is generated, it is saved and is accessible for the future. For this reason we grade efficiency a (-0.5).

USB/IP protocol

This protocol allows all USB devices to be shared, therefore it greatly increases interoperability (+2). Efficiency is increased (+1), compared with UPnP, because the protocol is implemented in the lowest layer of the Operating System. Results of the USB/IP project show performance in the range of 54-79% compared to local performance [5]. Middle-ware based approaches, e.g. NFS for file sharing, can outperform[6] USB/IP, though dedication to a specific service will come at cost of reduced transparency. Security is not explicitly addressed by this protocol and therefore graded (0).

GPU sharing protocols

The combination of GPU sharing protocols [2, 8] has positive effect on interoperability as graphical output is used constantly. However, variety of supported devices is limited to GPUs, therefore we assign (+1). Security is not of direct concern for this protocol, therefore (0) is assigned to it. Clearly efficiency is the primary motivation to use this protocol. Two techniques are combined in order to maximize resource allocation and workload consolidation. For this reason we assign (+2) to efficiency.

Overview by Force Resolution Table

In the table below we show an overview of how the Key Drivers are affected by each of the sub-protocols. By summing up effects of each sub-protocol we find the total effect of our design.

Protocol	Interoperability	Security	Efficiency
Security protocol	0	+2	-1
UPnP proxy	+2	0	-0.5
USB/IP	+2	0	+1
GPU Consolidation	+1	0	+2
Total	**+5**	**+2**	**+1.5**

5.3 Architecture Verification

Now we have defined the weights for the Key Drivers and evaluated the effect of the used sub-protocols used for our architecture, we verify whether the design satisfies the Key Drivers properly. Below a table is shown, where the totals of previous steps are normalized to allow meaningful comparison.

Evaluation	Interoperability	Security	Efficiency
KD Weight	1	0.7	0.4
Protocol Effect	1	0.4	0.3

We see that our protocol architecture addresses the Key Drivers in proper order. However, the results also show that our design impedes security and efficiency, in favor of interoperability. This could indicate that the evaluation was not performed properly, however it could also mean that our focus was biased towards interoperability. As interoperability is by far the most important Key Driver for any TDI protocol, the latter seems likely.

6 Discussion

In this paper we have explained why a TDI protocol is necessary and analyzed to what criteria any TDI protocol architecture should be assessed. Key Drivers that such protocol should strive for have been defined, together with necessary steps and related Key Driver weights.

We have combined a set of protocols into a proposed TDI protocol architecture of our own. The proposed architecture was evaluated the using the FRM evaluation method. The results of evaluation are mostly in accordance with the weight of the Key Drivers that we have defined, however our architecture appears biased towards interoperability. As this was the main motivation for our effort, we think it is not necessarily a bad thing. However, it does indicate that we have a reason to dedicate future work to the subject.

The proposed protocol is expected to reduce the problems arising when multiple devices are used for the same task, however the design is far from complete. Some issues of the current protocol have been mentioned, for example the need for separate peripheral drivers for each Operating System. However, this paper scratches only the surface of the problem. Future research on the topic should elaborate on the proposed combination of techniques. Also other methods, that relate to the Key Drivers, should be investigated and added to the architecture where possible. Though many apparent issues could be resolved with little effort, a thorough investigation must be performed to indicate which problems are not addressed or even thought of yet. If this research is left out, the effort is very likely to result in just another of the many incomplete protocols, inducing even more fragmentation in the field.

Linux kernel based Operating Systems offer promising opportunity for testing initial implementations as they are open source and can be executed on many devices including network routers and Android based smartphones. However, research and development of a TDI protocol should be coordinated with producers of hardware and developers of the widely used Operating Systems.

References

[1] Y. chun Hu, M. Jakobsson, and A. Perrig. Efficient constructions for one-way hash chains. In *In Applied Cryptography and Network Security (ACNS*, pages 423–441, 2005.

[2] M. Dowty and J. Sugerman. Gpu virtualization on vmware's hosted i/o architecture. *SIGOPS Oper. Syst. Rev.*, 43(3):73–82, July 2009.

[3] M. Dynamics. Ssdp simple service discovery protocol. *Discovery*, 2007.

[4] C. El Kaed, Y. Denneulin, and F.-G. Ottogalli. Dynamic service adaptation for plug and play device interoperability. In *Proceedings of the 7th International Conference on Network and Services Management*, CNSM '11, pages 46–55, Laxenburg, Austria, Austria, 2011. International Federation for Information Processing.

[5] T. Hirofuchi, E. Kawai, K. Fujikawa, and H. Sunahara. Usb/ip: A transparent device sharing technology over ip network. *IPSJ Digital Courier*, 1:394–406, 2005.

[6] . I. Ligon, W. B. and R. B. Ross. Implementation and performance of a parallel file system for high performance distributed applications. In *Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing*, HPDC '96, pages 471–. IEEE Computer Society, 1996.

[7] D. Pointcheval and J. Stern. Provably secure blind signature schemes. pages 252–265. Springer-Verlag, 1996.

[8] V. T. Ravi, M. Becchi, G. Agrawal, and S. Chakradhar. Supporting gpu sharing in cloud environments with a transparent runtime consolidation framework. In *Proceedings of the 20th international symposium on High performance distributed computing*, HPDC '11, pages 217–228, New York, NY, USA, 2011. ACM.

[9] K. Ren, S. Member, W. Lou, K. Kim, R. Deng, and S. Member. A novel privacy preserving authentication and access control scheme for pervasive computing environments. *IEEE Transactions on Vehicular Technology*, 55:1373–1384, 2006.

[10] J. Souza, S. Matwin, and N. Japkowicz. Evaluating data mining models: A pattern language, 2002.

The Semantic Web

Christian Hulleman and Gerjan Konterman

Abstract—The World Wide Web consists of more than 600 million web sites, all with several web pages of information. It is becoming difficult for computers to effectively search for information as requests are getting more complex. That is because web pages are designed to be read by humans and not for machines. In order for web pages to be read more effectively by computers, and thus allowing them to be able to process complex requests, the web has to evolve.

The solution to these problems is the semantic web, also known as Web 3.0. This next version of the web is considered to be the extension of the current web and is able to present information in both human readable and machine processable form. The goal of the semantic web is to give meaning to data that is accessible via the web. By using uniform semantic representations all the data in the web can be linked together and this makes it easy for machines to process it and create relationships between it.

At the moment the semantic web is still in active development with no unified adopted standards. Also, there is still no final architecture available. Several techniques and implementations consist to create a semantic web, but there are a lot of uncertainties about it. We want to provide some insights and solutions on this matter in order to speed up the adoption of the semantic web.

In this paper we will shortly explain what the semantic web exactly is and why it is needed. We will also explain the latest architecture and we will give some recommended ways how to make a web page semantic so that machines can read it.

Index Terms—Semantic Web, Web 3.0.

1 Introduction

The World Wide Web was created almost twenty years ago by Tim-Berners Lee. It was created with the idea to be able to present information on any operating system regardless of hardware. The web evolved tremendously, and now consists of more then 600 million pages with various information available. [3]

Currently information is shown using HyperText Markup Language (HTML) as a mark-up language. While this way of presenting data is useful for humans to read and easy to render for a web browser, it is hard for a machine to understand the actual meaning. When it is just plain text it is very difficult for a computer to extract the semantics out of it. In order to solve this, all the data should be labelled in a fashion that it tells the computer, what kind of data it contains. HTML is unable to provide ways to add semantics to data. For instance, in a web shop articles are displayed with various types of information. Information about the article is the name, price and the manufacturer of the item. Each of the items in the web shop is shown with this information. HTML only has the capability to position the text together in order for the user to see that these three types of data belong together. However a machine is unable to see if these three types of data belong together or represent a whole. Furthermore, it is unknown for the machine what kind of articles are listed on the web shop. This problem can be solved by using meta tags, but is a limited method to solve the problem rigorously. The semantic web solves these problems so that it is clear for a computer which kind of information is shown. In the web shop example this could be that the articles shown are parts for a car. Another advantages that happens when giving *semantics* to text, is that the data can be linked together. By linking data together, we can easily let computers search on various web sites, with the same articles, and find the cheapest article available. The question is however, how can we achieve this? The focus of our research is on current web sites. We want to provide a solution how a current web site can become more semantic. [4]

In this paper we will start off with explaining the current situation of the development of a semantic web architecture. In chapter 3 we will talk about the certain techniques that are important for the semantic web. Chapter 4 will discuss how to add semantics to a current web site. Current implementations of the semantics web will be discussed in chapter 5. Finally a conclusion is given in chapter 6 about the semantic web.

- *Christian Hulleman, student MSc. Computing Science University of Groningen, e-mail: c.hulleman.1@student.rug.nl*
- *Gerjan Konterman, student MSc. Computing Science University of Groningen, e-mail: g.konterman@student.rug.nl*

2 Semantic Web Architecture

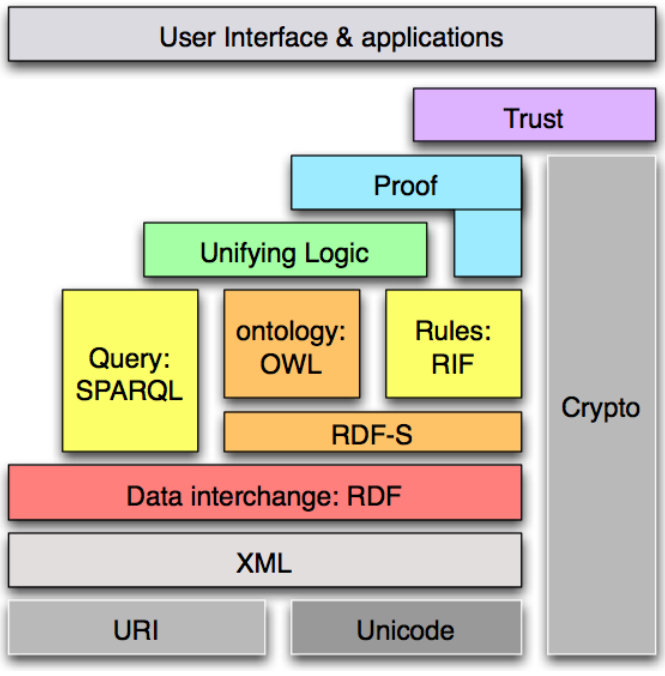


Fig. 1. Semantic Web Stack

The World Wide Web Organization (W3C) oversees the development of an architecture for the semantic web. Which is also considered as the reference architecture for the semantic web. Tim Berners-Lee has demonstrated various iterations of this semantic web architecture. The latest version was shown on the Twenty-first National Conference

on Artificial Intelligence (AAAI-06) in 2006. Figure 1 gives a graphical representation of this architecture. [2]

Although it is dubbed as an architecture, it actually is more a stack than an architecture. This is because it combines different techniques and languages, to give an overall picture what is important for the semantic web. The following enumeration gives an overview of the different techniques and languages presented in the semantic web architecture, starting from the bottom layer. [7] [8]

- *URI and UNICODE*: URI are used to identify the different semantic web resources. UNICODE is used as a standard for consisted encoding of characters in all languages.
- *XML*, is used as a syntax for the mark-up language of the semantic web resources.
- *RDF*, the Resource Description Framework(RDF) is probably the most important aspect of the semantic web stack. RDF is used to give semantics to specific data. RDF is based on an ontology that is described in OWL.
- *RDF-S*, the RDF-Schema gives a set of classes, properties and utility properties to be used for RDF.
- *SPARQL*, in order to query the RDF resources a query language is needed. The SPARQL query language is specifically written for the RDF framework. SPARQL allows users only to retrieve information, in order to update, add, or delete RDF records, SPARQL/Update is needed. The update module extends the query language to allow these query forms.
- *OWL*, is a declarative knowledge representation language for the semantic web. This language is used as a guideline for the RDF framework.
- *RIF*, which stands for Rule Interchange Format, is a layer to exchange the various rules between each other. This allows support for various different rule languages.
- *Cryto*, is another important layer in the semantic web stack. Crypto ensures that statements are coming from reliable sources.
- *Unifying Logic, Proof and Trust*, the most important layer is eventually Trust. This can be achieved by retrieving information from reliable resources and having a sound set of layers to retrieve this information from.
- *User Interface and Applications*, which consists of the user interfaces and applications using the semantic web stack.

In the following subsections we will go into detail about certain parts, such as how to represent semantics in a textual way.

3 Represent Semantics

We want to represents semantics so how can this be achieved? The answer for this is OWL and RDF. Here we first explain The Web Ontology Language (OWL) and after that we will give an implementation of it called RDF, which is the recommended way by W3C, and part of the Semantic Web Stack.

3.1 The Web Ontology Language

The Web Ontology Language, also known as OWL, is a declarative knowledge representation language for the semantic web. The most recent version of the OWL language is OWL 2, which is an extension and revision of OWL. We will discuss OWL 2 in this section. The W3C describes OWL 2 as following:

> *The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. [11]*

In other words, OWL 2 is a declarative language to express ontologies in. OWL 2 does not provide a syntax to describe how a document should be structured. it should not be confused with a schema language. The OWL ontology is used to store knowledge using the following three types:

- Entities, which are the basic elements of the ontology. Entities are either classes, properties or individuals.
- Expressions, which describe complex notions in the domain.
- Axioms, which are statements about the domain being described.

The abstract nation of OWL 2 can be used to create various concrete syntaxes. The primary syntax that has to be supported by tools implementing the OWL 2 ontology, is RDF/XML. This allows other applications or tools to use this data. As seen in Figure 2 there are also other syntaxes which are supported such as, OWL/XML, Functional Syntax, Manchester Syntax, and Turtle. As for the semantics of ontology, there is either the choice to do Direct Semantics or RDF-Based Semantics. The difference being that the latter one directly assigns meaning to RDF graphs. Direct Semantics however, gives direct meaning towards ontology structures. For instance with the SROIQ description logic, which is useful for its computing properties. In order to use OWL ontologies, one must use a ontology editor to create an ontology. Reasoners can then be used to query ontologies to determine if statement are either true or false regarding the ontology. For the semantic web, OWL 2 creates an useful semantic web language to represent data about various objects and there relationships.

3.2 Resource Description Framework

Now we have discussed OWL the next step is to give a more concrete solution for describing semantics. As said, RDF/XML is the primary syntax for OWL where RDF is the model and XML the syntax. The W3C gives the following definition about RDF:

> *RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a "triple"). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications. [22]*

RDF is the first and fundamental technology of the semantic web. It defines the underlying information model - based on triples - that underpins all semantic web mechanisms. The RDF model is very simple and yet powerful. It describes Resources through a series of Statements. Each statement is a Property and a Value that is about a Resource.

Listing 1. XML representation

```
<rdf:Description rdf:about="http://road.com/incidents/2010/06/15/00000001111222">
    <x:inLocation rdf:nodeID="Intersection"/>
</rdf:Description>
<rdf:Description rdf:nodeID="Intersection">
    <x:hasRoad rdf:resource="http://road.com/road/garden/st/01"/>
    <x:hasRoad rdf:resource="http://road.com/road/central/av/01"/>
</rdf:Description>
<rdf:Description rdf:about="http://road.com/road/garden/st/01">
    <x:inSuburb rdf:resource="http://road.com/locality/Eveleigh/01"/>
</rdf:Description>
```

Figure 3 gives an example of a RDF model together with the XML representation. Here you see a basic representation of a traffic accident and how it is related to its context. In the XML you can see that you can represent the accident very nicely: the accident happens at a intersection, the intersection consists of 2 roads, one road is called Garden

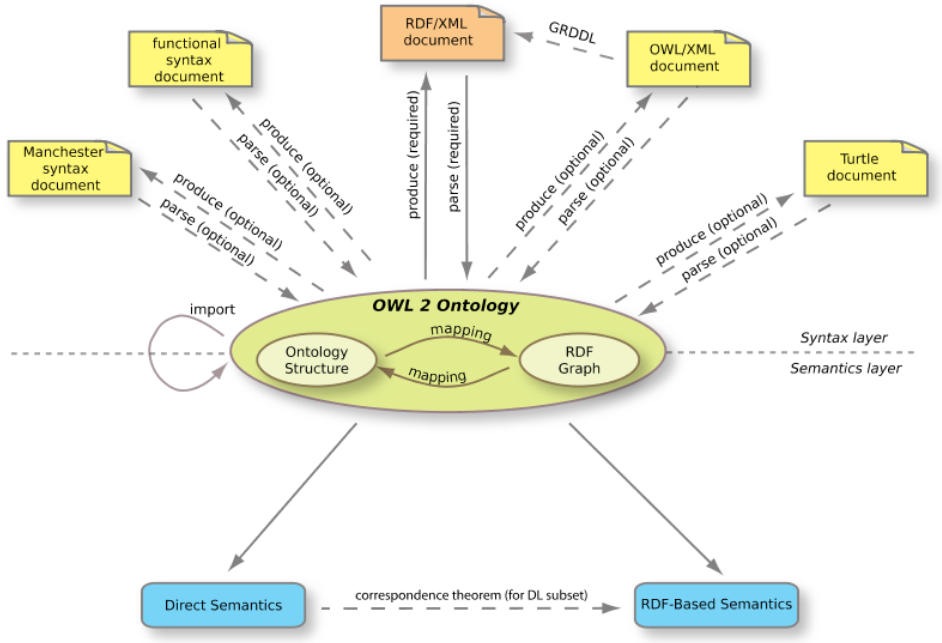


Fig. 2. Structure of OWL 2

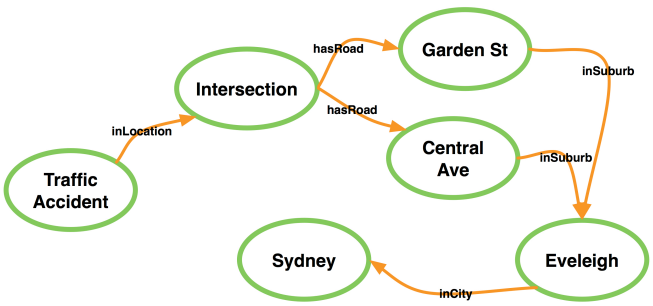


Fig. 3. Example of a RDF model

Street and this street is in Eveleigh. Imagine what a computer can do with all this information (ambulance paths, news, traffic light etc.)

This example is by far not everything you can do with RDF, as said before you can create Classes, SubClasses, Properties, Instances, Types, Domains/Ranges and more. For example, we can define ”length” as a Property for all Road classes. It is a powerful way to share and connect semantics/knowledge, it is possible to connect resources located at various locations by means of a relation defined at yet another location.

Also notice that RDF uses URI identifiers for Resources, Properties, and in many cases Objects, this way you can talk about the same entities. For example, if some other application was to refer to ’http://road.com/road/garden/st/01’ then both applications are referring to the same unique entity (Garden St in Eveleigh, Sydney).

You can query RDF with the RDF Query Language SPARQL which we discuss in the next section. It is also possible, since it is still XML, to use XPath/XSLT/XQuery for processing and searching.

For further technical information, see the RDF Primer. [22]

3.3 SPARQL

The *SPARQL Protocol and RDF Query Language* (SPARQL) is a query language for RDF. Currently the first version of SPARQL has been approved by the W3C. There is also a new version coming up, but that is still in draft phase as the moment. So we will be focusing on the version that is considered completed. The language itself is quite similar to the popular query language SQL. As already explained in the previous sections, is that the RDF framework consists of triples. These triples consist of a subject, predicate, and an object, each of these might be replaced by variable in a query. In SPARQL these three are called a basic graph pattern. Using this pattern, the query language can match a specific sub graph in the RDF data. We will be using the following information as an input for our query. [13]

1. As an identifier we will use: `<http://example.org/book/book1>`
2. As a predicate for our book we will use a title: `<http://purl.org/dc/elements/1.1/title>`
3. And the data (object) of our title is ”SPARQL Tutorial”

We might be curious for the titles in our dataset. The following query allows us to extract all the books:

```
SELECT ? title
WHERE
{
  < http:// example. org/ book/ book1>< http:// purl. org/ dc/ elements/1.1/
        title> ? title .
}
```

In return we get the title that is in de data, ”SPARQL Tutorial”. Besides the SELECT query, the language also has support for other query forms such as:

1. CONSTRUCT, which returns the data in graph forms. This allows us to easily serialize the data to RDF/XML form.
2. ASK, which can be used to query if a given pattern exists in the dataset. The return value is then a boolean value.
3. DESCRIBE, which returns a description on a specific identifier.

While SPARQL offers more functionalities than described in this section, we can conclude that SPARQL is a simple and easy to use query language. The language offers a wide range of features which can be used to retrieve various data from RDF data.

4 Implementing Semantics

Now we have presented how to represent semantics by using RDF we want to apply this to current web sites. Here we will discuss this is possible with an implementation called RDFa.

4.1 RDFa

How can a current web site become more semantic? It would be desirable that a web site does not need to have two versions of its web pages (a *current* web page and a *semantic* web page), because it requires a lot more work and it is bad for maintainability. Luckily RDF supports a compact syntax called RDFa, which stands for Resource Description Framework in attributes. RDFa provides a set of XHTML attributes to augment visual data with machine-readable data. This way you embed RDF data into your web page so you do not need to repeat the data twice, it is human readable and a computer can understand it. Here you can see how Figure 3 can be shown in RDFa.

Listing 2. RDFa Example

```
<div about="http://road.com/incidents/2010/06/15/000000011111222" typeof="x:
    TrafficAccident">
  A traffic accident at the
  <div typeof="x:Intersection">
    corner of
    <a rel="x:hasRoad" href="http://road.com/road/garden/st/01">Garden
      St</a> and
    <a rel="x:hasRoad" href="http://road.com/road/central/av/01">Central
      Ave</a>
  </div>
  in
  <a rel="x:inSuburb" href="http://road.com/locality/Eveleigh/01">Eveleigh</a>,
  <span property="x:inCity">Sydney</span>.
</div>
```

4.2 RDFa Elaborated Example

A more elaborated and concrete example for web sites is the following: a simple web page which displays some basic information over a person John D. The HTML code for this can be:

Listing 3. Basic HTML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>John's Home Page</title>
  </head>
  <body>
    <h1>John's Home Page</h1>
    <p>My name is John D and I like Star Wars</p>
    <p>My favorite book is the inspiring Weaving the Web by Tim
      Berners-Lee</p>
  </body>
</html>
```

This is not really exciting. It is just basic XHTML with the needed standard tags. Here you can see that it uses a h1 tag and paragraph tags to present the data. The problem with this is that this piece of code is very hard to understand for a computer. It is difficult for machines to extract the meaning from this data.

With RDFa you can state explicitly what is shown on the web page. By doing this you can 'help' a computer to understand what is inside a web page. Example:

Listing 4. Basic HTML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    version="XHTML+RDFa 1.0" xml:lang="en">
  <head>
    <title>John's Home Page</title>
    <base href="http://example.org/john-d/" />
    <meta property="dc:creator" content="Jonathan Doe" />
    <link rel="foaf:primaryTopic" href="http://example.org/john-d/#me" />
  </head>
  <body about="http://example.org/john-d/#me">
    <h1>John's Home Page</h1>
    <p>My name is <span property="foaf:nick">John D</span> and I like
      <a href="http://www.starwars.org/" rel="foaf:interest"
        xml:lang="de">Star Wars</a>.
    </p>
    <p>
      My <span rel="foaf:interest" resource="urn:ISBN:0752820907">favorite
      book is the inspiring <span about="urn:ISBN:0752820907"><cite
      property="dc:title">Weaving the Web</cite> by
      <span property="dc:creator">Tim Berners-Lee</span></span>
    </span>
    </p>
  </body>
</html>
```

You clearly see that there is much more data. To be more specific, there are more tags and attributes used to add semantics to a web page. For example the name John D is now wrapped within a span element and with an attribute you specify that inside the element there will be a name of a person. Now a computer knows exactly what kind of data is inside. Also the interests of this person are specified with extra tags and attributes. You probably see that this way you are creating links between entities/things. Especially these links are very interesting because then it is possible to create or see relationships, even across multiple web sites and data sources.

4.3 Alternatives

We think that RDFa is the best way to add semantics to a web page because it uses RDF which is part of the Semantic Web Stack. It is also developed and recommended by the W3C. However, there are some alternatives which we will discuss here shortly to show that there are other ways to add semantics.

4.3.1 Microformat

A microformat is a web-based approach to semantic markup which seeks to re-use existing HTML/XHTML tags to convey metadata and other attributes in web pages and other contexts that support (X)HTML. Example:

Listing 5. Microdata Example

```
The birds roosted at
  <span class="geo">
    <span class="latitude">52.48</span>,
    <span class="longitude">-1.89</span>
  </span>
```

4.3.2 Microdata

Microdata is a WHATWG HTML specification used to nest semantics within existing content on web pages. Microdata use a supporting vocabulary to describe an item and name-value pairs to assign values to its properties. Microdata is an attempt to provide a simpler way of annotating HTML elements with machine readable tags than the similar approaches of using RDFa and Microformats. Example:

Listing 6. Microdata Example

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
    Hello, my name is
  <span itemprop="name">John Doe</span>,
    I am a
  <span itemprop="title">graduate research assistant</span>
    at the
  <span itemprop="affiliation">University of Dreams</span>.
    My friends call me
  <span itemprop="nickname">Johnny</span>.
    You can visit my homepage at
  <a href="http://www.JohnnyD.com" itemprop="url">www.JohnnyD.com</a>.
  <section itemprop="address" itemscope itemtype="http://data-vocabulary.org/
      Address">
    I live at
    <span itemprop="street-address">1234 Peach Drive</span>
    <span itemprop="locality">Warner Robins</span>
    ,
    <span itemprop="region">Georgia</span>.
  </section>
</section>
```

5 Current Implementations

As mentioned earlier, currently there are some initiatives that are more or less semantic applications. We will discuss a few of them in this section to give an impression of the Semantic Web.

5.1 Wolfram Alpha

Wolfram Alpha is an answer-engine developed by Wolfram Research. It is an online service that answers factual queries directly by computing the answer from structured data, rather than providing a list of documents or web pages that might contain the answer as a search engine might. It should be noted that Wolfram Alpha is not really a semantic web search engine because it has its own very large database

that they maintain themselves. It also uses clever algorithms and natural language processing to provide the correct answer. The reason we include Wolfram Alpha is because we think a semantic web search engine can give similar results because of the fact everything is linked together.

5.2 DBpedia

DBpedia is a project aiming to extract structured content from the information created as part of the Wikipedia project. This structured information is then made available on the World Wide Web. DBpedia allows users to query relationships and properties associated with Wikipedia resources, including links to other related datasets. DBpedia has been described by Tim Berners-Lee as one of the more famous parts of the Linked Data project.

5.3 Visual Data Web

Visual Data Web is a web site that presents developments, related publications, and current activities. The goal is to generate new ideas, methods, and tools that help making the web easier accessible, more visible, and thus more attractive. They will do this with techniques from the semantic web like RDF and Linked Data. The tools on the web site are really nice and we encourage to look at these because it shows how powerful a complete semantic web can be. For the demos they mostly use DBpedia as their data source.

5.3.1 RelFinder

The RelFinder helps to get an overview how things are related with each other. It extracts and visualizes relationships between objects in RDF data and makes these relationships interactively explorable.

Fig. 4. RelFinder

5.3.2 SemLens

SemLens provides a visual interface that combines scatter plots and semantic lenses. The scatter plots offer a global visualization that can be locally explored with the semantic lenses

5.3.3 gFacet

gFacet supports faceted exploration of RDF data by representing facets as nodes in a graph visualization. The facets can be interactively added and removed in order to produce individual search interfaces.

5.3.4 tFacet

tFacet applies known interaction concepts to allow hierarchical faceted exploration of RDF data. The aim is to facilitate ordinary users to formulate semantically unambiguous queries so as to support the fast and precise access to information.

5.4 Freebase

Freebase is a large collaborative knowledge base consisting of metadata composed mainly by its community members. It is an online collection of structured data harvested from many sources, including individual 'wiki' contributions. Freebase aims to create a global resource which allows people (and machines) to access common information more effectively. It was developed by the American software company Metaweb and has been running publicly since March 2007. Metaweb was acquired by Google in a private sale announced July 16, 2010.

5.5 Nextbio

NextBio is a privately owned software company that provides a platform for drug companies and life science researchers to search, discover, and share knowledge across public and proprietary data. The NextBio Platform is an ontology-based semantic framework that connects highly heterogeneous data and textual information. The semantic framework is based on gene, tissue, disease and compound ontologies. This framework contains information from different organisms, platforms, data types and research areas that is integrated into and correlated within a single searchable environment using proprietary algorithms. It provides a unified interface for researchers to formulate and test new hypotheses across vast collections of experimental data.

6 Conclusion

To create a fully semantic web is hard but very challenging. When everybody sees the strength of adding semantics to the web and will do this to their web sites, there will be a considerable movement to the current web. This paper gives advice how the current web can become more semantic. It describes how semantics can be represented in a textual representation. To do this you can make use of Resource Description Framework (RDF). This framework is invented and supported by the W3C so it is a good way to represent semantics.

The main goal of this paper is to give a solution for current web sites so that they can become more semantic. We have presented a relatively easy way to do this, called RDFa. With RDFa it is possible to add semantics to a current web page by embedding RDF inside HTML.

We can conclude that it is not very hard to add semantics to current web sites, but the problem still remains that most of the regular web sites are not doing it. We think this is because a lot of people have not heard about the Semantic Web and they think it does not result in direct benefits for a web site. At the moment a visually attractive web site is much more important than the way the data is presented to machines. Another problem is that there is no unified standard for the semantic web. The W3C is working on it, but there are still uncertainties about implementations and also big companies like Google and Microsoft are not fully embracing it. With this paper we have targeted a specific area in this field. We hope that we make a more clearer picture how you can add semantics and this way we can speed up the evolution of the web.

References

[1] A. v. d. M. Aurona Gerber and A. Barnard. A functional semantic web architecture, 2006.

[2] T. Berners-Lee. Semantic web - xml2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/, 2000.

[3] T. Berners-Lee. Tim berners-lee on the semantic web. http://www.ted.com/talks/lang/en/tim_berners_lee_on_the_next_web.html, 2009.

[4] T. Berners-Lee. Tim berners-lee: The year open data went worldwide. http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html, 2010.

[5] G. Bouma. Advanced web technology - semantic web, 2011.

[6] B. C. Grau. A possible simplification of the semantic web architecture, 2003.

[7] M. Haytham Al-Feel and H. Suoror. Toward an agreement on semantic web architecture, 2009.

[8] P. P.-S. Ian Horrocks, Bijan Parsia and J. Hendler. Semantic web architecture: Stack or two towers, 2005.

[9] R. Iannella. Semantic web architectures, 2010.

[10] H. B. Michael Kifer, Jos de Bruijn and D. Fensel. A realistic architecture for the semantic web, 2005.

[11] B. P. P. F. P.-S. S. R. Pascal Hitzler, Markus Krtzsch. Owl 2 web ontology language primer. http://www.w3.org/TR/owl-primer/, 2009.

[12] S. L. Philipp Heim and T. Stegemann. Interactive relationship discovery in rdf data. http://www.visualdataweb.org/relfinder.php, 2012.

[13] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/, 2008.

[14] L. Slaghuis. Semantic web - hoe werkt het nou echt. http://www.frankwatching.com/archive/2009/02/18/semantic-web-hoe-werkt-het-nou-echt/, 2009.

[15] Various. Internetworldstats. http://www.internetworldstats.com/stats.htm.

[16] Various. Wikipedia - resource description framework. http://en.wikipedia.org/wiki/Resource_Description_Framework.

[17] Various. Wikipedia en - microdata. http://en.wikipedia.org/wiki/Microdata_(HTML5).

[18] Various. Wikipedia en - microformats. http://en.wikipedia.org/wiki/Microformats.

[19] Various. Wikipedia en - semantic web. http://en.wikipedia.org/wiki/Semantic_Web.

[20] Various. Wikipedia nl - web 3.0. http://nl.wikipedia.org/wiki/Web_3.0.

[21] Various. Worldwidewebsize. http://www.worldwidewebsize.com/.

[22] Various. Rdf primer. http://www.w3.org/TR/rdf-primer/, 2011.

[23] Various. Semantic web architecture. http://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html, 2011.

[24] W3C. Resource description framework (rdf). http://www.w3.org/RDF/, 2011.

[25] W3C. W3c semantic web activity. http://www.w3.org/2001/sw/, 2011.

university of groningen / faculty of mathematics and natural sciences / computing science