PATTERNS 2016

The Eighth International Conferences on Pervasive Patterns and Applications

ISBN: 978-1-61208-465-7

March 20 - 24, 2016

Rome, Italy

PATTERNS 2016 Editors

Herwig Manaert, University of Antwerp, Belgium

Alexander G. Mirnig, University of Salzburg, Austria

PATTERNS 2016

Forward

The Eighth International Conferences on Pervasive Patterns and Applications (PATTERNS 2016), held between March 20-24, 2016 in Rome, Italy, continued a series of events targeting the application of advanced patterns, at-large. In addition to support for patterns and pattern processing, special categories of patterns covering ubiquity, software, security, communications, discovery and decision were considered. It is believed that patterns play an important role on cognition, automation, and service computation and orchestration areas. Antipatterns come as a normal output as needed lessons learned.

The conference had the following tracks:

- Patterns basics
- Patterns at work
- Discovery and decision patterns

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the PATTERNS 2016 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to PATTERNS 2016. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the PATTERNS 2016 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope PATTERNS 2016 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of pervasive patterns and applications. We also hope that Rome provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

PATTERNS 2016 Chairs

PATTERNS Advisory Chairs

Herwig Manaert, University of Antwerp, Belgium
Fritz Laux, Reutlingen University, Germany
Michal Zemlicka, Charles University - Prague, Czech Republic
Alfred Zimmermann, Reutlingen University, Germany
Richard Laing, Robert Gordon University, UK
Ricardo Sanz, UPM ASlab, Spain

PATTERNS Research/Industry Chairs

Teemu Kanstren, VTT, Finland
Markus Goldstein, Kyushu University, Japan
Zhenzhen Ye, IBM, Essex Junction, USA
René Reiners, Fraunhofer FIT - Sankt Augustin, Germany
Nguyen Vu Hoàng, Vertigo, France
Alexander Mirnig, University of Salzburg, Austria

PATTERNS Publicity Chairs

Bastian Roth, University of Bayreuth, Germany
Steve Strauch, IAAS at University of Stuttgart, Germany
Jaap Kabbedijk, Utrecht University, Netherlands

PATTERNS 2016

Committee

PATTERNS Advisory Committee

Herwig Manaert, University of Antwerp, Belgium
Fritz Laux, Reutlingen University, Germany
Michal Zemlicka, Charles University - Prague, Czech Republic
Alfred Zimmermann, Reutlingen University, Germany
Richard Laing, Robert Gordon University, UK
Ricardo Sanz, UPM ASlab, Spain

PATTERNS Research/Industry Chairs

Teemu Kanstren, VTT, Finland
Markus Goldstein, Kyushu University, Japan
Zhenzhen Ye, IBM, Essex Junction, USA
René Reiners, Fraunhofer FIT - Sankt Augustin, Germany
Nguyen Vu Hoàng, Vertigo, France
Alexander Mirnig, University of Salzburg, Austria

PATTERNS Publicity Chairs

Bastian Roth, University of Bayreuth, Germany
Steve Strauch, IAAS at University of Stuttgart, Germany
Jaap Kabbedijk, Utrecht University, Netherlands

PATTERNS 2016 Technical Program Committee

Ina Suryani Ab Rahim, Pensyarah University, Malaysia
Mourad Abbas, Computational Linguistics Department - CRSTDLA, Algeria
Siby Abraham, University of Mumbai, India
Adel Al-Jumaily, University of Technology, Sydney, Australia
Adel Alimi, University of Sfax, Tunisia
Junia Anacleto, Federal University of Sao Carlos, Brazil
Andreas S. Andreou, Cyprus University of Technology - Limassol, Cyprus
Annalisa Appice, Università degli Studi di Bari, Italy
Martin Atzmueller, University of Kassel, Germany
Noor Azilah binti Draman@Muda, Universiti Teknikal Malaysia, Malaysia
Senén Barro, University of Santiago de Compostela, Spain
Rémi Bastide, University Champollion / IHCS - IRIT, France
Bernhard Bauer, University of Augsburg, Germany
Noureddine Belkhatir , University of Grenoble, France

Michael Stal, Siemens, Germany
Janis Stirna, Stockholm University, Sweden
Mu-Chun Su, National Central University, Taiwan
Sam Supakkul, Sabre Inc., USA
Stella Sylaiou, Hellenic Open University, Greece
Ryszard Tadeusiewicz, AGH University of Science and Technology, Poland
Dan Tamir, Texas State University, USA
Shanyu Tang, China University of Geosciences - Wuhan City, P. R. China
Horia-Nicolai Teodorescu, "Gheorghe Asachi" Technical University of Iasi / Romanian Academy, Romania
Ghanshyam Singh Thakur, Maulana Azad National Institute of Technology, India
Daniel Thalmann, Nanyang Technological University, Singapore
Alain Léger, Orange Labs - France Telecom R&D, Cesson Sévigné, Rennes, France
Mati Tombak, University of Tartu / Tallinn Technical University, Estonia
Alessandro Torrisi, Università di Catania, Italy
George Tsihrintzis, University of Piraeus, Greece
Theodoros Tzouramanis, University of the Aegean, Greece
Domenico Ursino, University Mediterranea of Reggio Calabria, Italy
Michael Vassilakopoulos, University of Thessaly, Greece
Phan Cong Vinh, Nguyen Tat Thanh University (NTTU), Vietnam
Panayiotis Vlamos, Ionian University, Greece
Krzysztof Walczak, Poznan University of Economics, Poland
Stefan Wendler, Ilmenau University of Technology, Germany
Laurent Wendling, University Descartes (Paris 5), France
Wojciech Wiza, Poznan University of Economics, Poland
Mudasser F. Wyne, National University- San Diego, USA
Dongrong Xu, Columbia University & New York State Psychiatric Institute, USA
Reuven Yagel, The Jerusalem College of Engineering, Israel
Zhenzhen Ye, IBM, Essex Junction, USA
Jin Soung Yoo, Indiana University - Purdue University Fort Wayne, USA
Lihua You, Bournemouth University, UK
Nicolas H. Younan, Mississippi State University, USA
Hongchuan Yu, Bournemouth University, UK
Amelia Zafra Gómez, University of Cordoba, Spain
Lihong Zheng, Charles Sturt University, Australia
Alfred Zimmermann, Reutlingen University, Germany
Michal Žemlička, Charles University, Czech Republic

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Learning Multi-Class Discriminative Patterns using Episode-Trees

Eng-Jon Ong*, Nicolas Pugeault†, Andrew Gilbert* and Richard Bowden*
*Centre for Vision, Speech and Signal Processing
University of Surrey, UK
Email: e.ong,a.gilbert,r.bowden@surrey.ac.uk
† College of Engineering, Mathematics and Physical Sciences,
University of Exeter, UK
Email: n.pugeault@exeter.ac.uk

***Abstract*—In this paper, we aim to tackle the problem of recognising temporal sequences in the context of a multi-class problem. In the past, the representation of sequential patterns was used for modelling discriminative temporal patterns for different classes. Here, we have improved on this by using the more general representation of *episodes*, of which sequential patterns are a special case. We then propose a novel tree structure called a MultI-Class Episode Tree (MICE-Tree) that allows one to simultaneously model a set of different episodes in an efficient manner whilst providing labels for them. A set of MICE-Trees are then combined together into a MICE-Forest that is learnt in a Boosting framework. The result is a strong classifier that utilises episodes for performing classification of temporal sequences. We also provide experimental evidence showing that the MICE-Trees allow for a more compact and efficient model compared to sequential patterns. Additionally, we demonstrate the accuracy and robustness of the proposed method in the presence of different levels of noise and class labels.**

Keywords–Data Mining, Classification, Episodes Patterns, Decision Trees

I. INTRODUCTION

There are many problems in machine learning where the data consists of temporal sequence of events. In this work, we consider the problem where we have a collection of data streams that we want to label according to a large number of classes. Although there has been significant work concerned with mining frequently occurring patterns in data streams, few studies have focused on the different task of classifying such data streams. When solving this problem, we face several challenges: 1) a large number of classes may lead to bloated models, and large class confusion; 2) learning *discriminative* sequences rather than *frequently* occurring ones; 3) relevant sequences that contain discriminative power may be sparse in the data stream; and 4) the presence of ambiguities in the ordering of some parts of a discriminative sequence. In order to address these problems, this article presents a theoretical framework for learning of discriminative temporal sequences based on episodes [1] that are structured efficiently within a tree. We show how multiple episode-trees can be combined in a Boosted framework to yield an accurate and robust classifier.

There is a significant amount of prior research that investigate the discovery of *frequently occurring* temporal sequences, represented using sequential patterns. There are two main approaches for mining frequent sequential patterns: Apriori-based methods [2], [3] and pattern growth methods [4], [5]. An alternative to sequential patterns is the representation of so-called "episodes". Episodes are a more generic representation of temporal patterns proposed by Mannila et al.[1], [6] that allow the formalisation of ambiguity in the sequencing of some events in the pattern (whereas sequential patterns represent strict ordering). They also proposed algorithms for finding frequent episodes in data streams, in the limited case of either *serial* or *parallel* episodes. More recently, two independent groups have proposed algorithms for mining frequent episodes in the general case [7], [8], [9].

The present work addresses a different problem: our aim is the classification of data streams, and therefore the sequence of events are learnt for maximizing discrimination between the classes. Hence, a pattern may be discriminative for a single class amongst many without being frequent over the whole dataset. An early attempt at learning discriminative sequential patterns was proposed by Nowozin [10] and showed promising results for action recognition, yet the proposed approach is limited to binary discrimination. For problems containing more than two classes, this entails learning a collection of 1 versus 1 classifiers within a voting framework—this approach is clearly not scalable to problems with large number of classes. Recently, Ong et al. [11], [12] proposed a scalable approach to multi-class sequential pattern classification that makes use of boosted Sequential Pattern trees (SP-trees) for learning discriminative sequential patterns. One essential property of this approach is that the SP-trees allow for feature sharing between sequential patterns that describe different classes. To our knowledge these articles are the only attempts at learning discriminative sequential patterns.

This article extends SP-trees to the more general formalism provided by episodes. We will show that for certain specific patterns, episode trees will allow for a more compact encoding of discriminative temporal patterns. This article has four main contributions: first, we present a theoretic definition of a tree structure called MultI-Class Episode Trees (MICE-Trees) allowing for multiple classes to share sub-episodes; second, we propose an efficient algorithm for learning Boosted collections of such multi-class trees; third we demonstrate that MICE-Trees are similar to SP-Trees, but allow for a better and more compact representation of certain type of temporal sequences; and fourth, we show that the resulting classifiers can cope with a large amount of signal noise while still providing good classification accuracy.

 ISBN: 978-1-61208-465-7

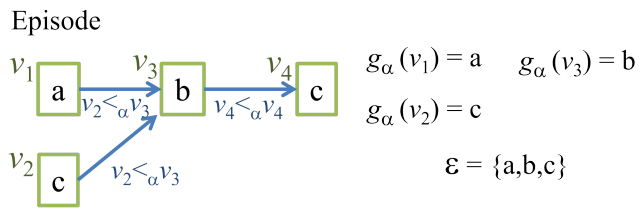


Figure 1. Illustration of a generic episode α, of the form $(a, c) \rightarrow (b) \rightarrow (c)$.

The rest of the paper is organised as follows: section 1 will provide the theoretical framework of episodes and some important results; section 2 will present the MICE-Trees and provide proofs that paths in such a tree are effectively episodes; section 3 will describe an efficient algorithm for learning such MICE-Trees from a dataset of labelled streams; section 4 will present an approach for learning a Boosted forests of MICE-Trees; section 5 will present two synthetic dataset that are used to evaluate the performance and robustness of the proposed approach; and finally we will conclude in section 6.

II. Problem Statement

Given a vocabulary of events $\mathcal{E} = \{E_1, \ldots, E_{|\mathcal{E}|}\}$, we define a *data stream* $S \in \mathcal{S}_\mathcal{E}$ (where $\mathcal{S}_\mathcal{E}$ is the set of all possible data streams for a vocabulary $\mathcal{E}$), as a sequence of pairs

$$S = \left\langle (E_1, t_1), \ldots, (E_{|S|}, t_{|S|}) \right\rangle, \tag{1}$$

where (E_i, t_i) denotes the occurence of the transient event E_i at time t_i—where the time labels t_i are such that $t_i \leq t_j \ \forall i < j$.

Given a collection of data streams $\mathcal{D} = \{(S_i, \lambda_i)\}_{i \in [1,|\mathcal{D}|}$, with associated class labels $\lambda_i \in \mathcal{L}$, the task of stream classification can be seen as the function

$$F : \mathcal{S}_\mathcal{E} \rightarrow \mathcal{L}, \tag{2}$$

that associates labels λ_i to data streams S_i. Specifically, this article proposes an approach for learning F from a set $\mathcal{D}$ of labelled streams using *discriminative episodes*.

A. Episodes

Following from Mannila et al. [6], we define an *episode* as

Definition 1. An episode is defined as $\alpha = (V_\alpha, <_\alpha, g_\alpha)$, where $V_\alpha = \{v_1, \ldots, v_{|v|}\}$ is a collection of vertices, $<_\alpha \subseteq V_\alpha \times V_\alpha$ is a strict partial order, and $g_\alpha : V_\alpha \rightarrow \mathcal{E}$ is a mapping between episode vertices and observed events.

where

Definition 2. The relation $<_\alpha$ is a *strict partial order*, iff. $\forall (a, b) \in <_\alpha$, we have:

$$a \neq b \tag{3}$$

$$(b, a) \notin <_\alpha \tag{4}$$

$$(b, c) \in <_\alpha \Rightarrow (a, c) \in <_\alpha \tag{5}$$

This is illustrated in Fig. 1. In this figure, each green box corresponds to an episode vertex v_i, and the letter in the box corresponds to the mapped event $E_j = g_\alpha(v_i)$, with an alphabet $\mathcal{E} = \{a, b, c\}$. The blue arrows joining the boxes represent the serial ordering between vertices enforced by the strict partial order $<_\alpha$.

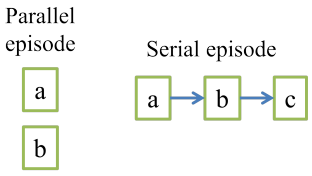


Figure 2. Illustration of a parallel episode (left) of the form (a, b) and of a serial episode (right), of the form $(a) \rightarrow (b) \rightarrow (c)$.

Moreover, we define sequential patterns as special cases, coined *serial episodes*(see Fig. 2, right):

Theorem 1. *If $<_\alpha^+$ is a strict total order, then $\alpha = (V_\alpha, <_\alpha^+, g_\alpha)$ is a serial episode (i.e., a sequential pattern).*

Proof: If $<_\alpha^+$ is a strict total order, then $\forall a, b \in V_\alpha, a \neq b$, we have

$$(a, b) \in <_\alpha^+ \ \textbf{or} \ (b, a) \in <_\alpha^+,$$

hence there exists a sequence $(\beta_1, \ldots, \beta_{|V_\alpha|})$ such that

$$\forall i, j \in [1..|V_\alpha|], i < j \Rightarrow (v_{\beta_i}, v_{\beta_j}) \in <_\alpha \tag{6}$$

and therefore $(g_\alpha(v_{\beta_i}))_{i=1}^{|V_\alpha|}$ is a sequential pattern. ■

and conversely we define *parallel episodes* (illustrated in the left panel of Fig. 2):

Definition 3. If $<_\alpha = \emptyset$, then α is a *parallel episode*.

Finally we define a relation $\alpha \sqsubseteq S$ which states that an episode α occurs within a stream $S \in \mathcal{S}$:

Definition 4. Let α be an episode and $S \in \mathcal{S}$ be a stream, we define that $\alpha \sqsubseteq S$, iff. there exists a sequence $\beta_1, \ldots, \beta_{|V_\alpha|}$ such that $\forall i, j \in [1..|V_\alpha|], i < j \Rightarrow (v_{\beta_i}, v_{\beta_j}) \in <_\alpha$, and that $\exists \{t_i\}_{i=1}^{|V_\alpha|}, \left((g_\alpha(v_{\beta_i}), t_i)_{i=1}^{|V_\alpha|} \right) \subset S$.

This is illustrated in Fig. 3, where the red arrows indicate the occurrences of the episode α's vertices v_i, mapped to events $a, b, c \in \mathcal{E}$ in the stream S. Note that the sequence of the occurrence of events a and b for the episode vertices v_1 and v_2 do not matter (they form a parallel episode).

B. MultIClass Episode Trees (MICE-Trees)

This section presents a definition of the MICE-Trees and how paths to a MICE-Tree's leaves model different episodes α_k.

First, we define a MICE-Tree as $T = (N, L)$, where $N = \{n_i\}_{i=1}^{|N|}$ is the set of all tree nodes and $L = \{l_i\}_{i=1}^{|L|}$ the set of all links, such that $L \subseteq N \times N$—such a tree is illustrated in Fig. 4. In the following section we will define the nodes and links of the MICE-Trees and their specificity for the purpose of learning and encoding episodes.

1) Tree nodes $n \in N$: MICE-Tree nodes have the double purpose of storing episode information and the most likely class label given the data. Formally:

Definition 5. A MICE-Tree node $n \in N$ is defined as the tuple $n = (V_n, g_n, \lambda_n)$, where $\lambda_n \in \mathcal{L}$ is the most likely label at this node, V_n is a set of vertices and g_n is a mapping such

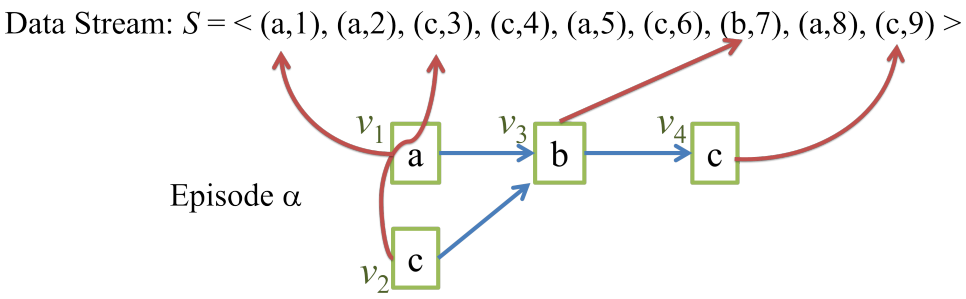


Figure 3. Illustration of how an episode α is matched in a data stream S, such as $\alpha \sqsubseteq S$.

that $\forall v_i \in V_n,\ \exists E_i \in \mathcal{E}$ such that $E_i = g_n(v_i)$ and $\{E_i\}_{i=1}^{|V_n|}$ is a set of (unordered) events.

Without loss of generality, we enforce that the set of vertices in all tree nodes be strictly disjoint:

Definition 6. For any two nodes $n, n' \in N$, if $n \neq n'$ then $V_*(n) \cap V_*(n') = \emptyset$

Moreover, it follows from the definition that:

Property 1. *A tree node $n \in N$ models a (trivial) parallel episode $\alpha = (V_*(n), \emptyset, g_*(n))$.*

For convenience we define the following accessor functions for the properties of a node $n = (V_n, g_n, \lambda_n)$, namely: $V_*(n) = V_n$, $g_*(n) = g_n$ and $\lambda_*(n) = \lambda_n$.

2) Tree links $l \in L$: Now that we have defined the MICE-Tree nodes as encoding the parallel components of episodes, we will define how the tree links encode the serial constraints in the episode, and whether these are satisfied or not by data streams. Formally, we define a tree link as follows:

Definition 7. A MICE-Tree link $l \in L$ is defined as the tuple $l = (a, b, s)$ where $a, b \in N$ are nodes in the tree and $s \in \{+, -\}$ is the type of edge (positive or negative).

For convenience, we define the root of the tree as

Definition 8. We call *root node* of a tree $T = (N, L)$ the unique node $n^0 \in N$ that satisfies the condition $\nexists l \in L, n \in N, s \in \{+, -\}$ such that $l = (n, n^0, s)$.

and its leaves:

Definition 9. We say that a node $n^* \in N$ is a *leaf node* if $\nexists l \in L, n \in N, s \in \{+, -\}$ such that $l = (n^*, n, s)$.

Without loss of generality, we will define that leaf nodes are the only nodes in the tree containing no vertices, hence we have:

Property 2. *For any node $n \in N$, we have $V_*(n) = \emptyset$ iff. n is a leaf node.*

This property is ensured by the learning mechanism described in section 3.

In this work, we restrict ourselves to using binary trees, whereby every non-leaf tree-node ($n \in N$) will have exactly two child tree-nodes $n^+, n^- \in N$, where $(n, n^+, +) \in L$ and $(n, n^-, -) \in L$. Also, as for the nodes we define for links $l = (p_l, c_l, s_l)$ the accessor function $s_*(l) = s_l$.

Property 1 showed that MICE-Tree nodes model parallel episodes; conversely, we will now show that links can be interpreted as the *serial* components of episodes. As a first step, we need to show that two nodes connected by a link form a strict partial order (see definition 2).

Definition 10. We define a function φ between two sets of vertices V_a and V_b, such that $\varphi(V_a, V_b) = \{(x, y) : x \in V_a, y \in V_b\}$.

Theorem 2. *If $V_a \cap V_b = \emptyset$ then $\varphi(V_a, V_b)$ is a strict partial order.*

Proof: Let $(a, b) \in \varphi(V_a, V_b)$, then from definition 10, we have $a \in V_a$ and $b \in V_b$, and therefore $V_a \cap V_b = \emptyset$ implies: (1) $a \neq b$; (2) $b \notin V_a$ and therefore $\forall c \in V_b, (b, c) \notin \varphi(V_a, V_b)$ ■

From definitions 6, 7, 10 and theorem 2, we derive the following property for linked nodes:

Corollary 1. *A tree node $n \in N$ models a parallel episode $\alpha = (V_*(n), \emptyset, g_*(n))$, and a pair of nodes n and n' connected by a link l models the general episode α', such as:*

$$\alpha' = (V_*(n) \cup V_*(n'), \varphi(V_*(n), V_*(n')), g_*(n) \cup g_*(n')). \quad (7)$$

3) Tree Paths $P \subset T$: In the previous section we have presented the MICE-Tree nodes and edges, we will now define paths in the tree, and show that any path in a MICE-Tree encodes an episode.

First we will define a tree path:

Definition 11. A path P is a sequence of node-edge pairs $P = ((n_1, l_1), \ldots, (n_K, l_K))$, where K is the length of the path, and all the nodes N_i in the path are connected by their edges. Formally, $l_i = (n_i, n_{i+1}, s_i)$, $\forall i \in [1, K-1]$, and $l_K = (n_K, n^*, s_K)$, where n^* is a leaf node.

and leaf nodes:

Definition 12. We say that a node $n^* = (\emptyset, \emptyset, \lambda) \in N$ is a leaf node if $\nexists l \in L, n \in N, s \in \{+, -\}$ such that $l = (n^*, n, s)$.

Then we define the function ξ that generates an episode from any MICE-Tree path:

Definition 13. Let $P = ((n_1, l_1), \ldots, (n_{|P|}, l_{|P|}))$ be a path in tree T, and let $\psi(P) \subset P$ be the indices of positive links in the path: $\psi(P) = \{i : i \in [1, |P|], s_*(l_i) = +\}$. Then we

 ISBN: 978-1-61208-465-7

define the function $\xi(P) = (V_P, <_P, g_P)$, such that

$$\begin{cases} V_P &= \bigcup_{i \in \psi(P)} V_*(n_i) \\ <_P &= \bigcup_{i \in \psi(P)} \bigcup_{j \in \{k \in S(P) : k > i\}} \varphi(V_*(n_i), V_*(n_j)) \\ g_P &= \bigcup_{i \in \psi(P)} g_*(n_i) \end{cases} . \tag{8}$$

Lemma 1. *If P is a path in a MICE-Tree g_P is a map such that $g_P : V_P \rightarrow \mathcal{E}$.*

Proof: For all nodes n in the tree, we have $g_*(n)$ a map such that $g_*(n) : V_*(n) \rightarrow \mathcal{E}$, $\forall n \in N$ (definition 5) and $V_*(n_i) \cap V_*(n_j) = \emptyset$ $\forall n_i, n_j \in N, n_i \neq n_j$ (definition 6), therefore $\bigcup_{i \in \psi(P)} V_*(n_i)$ is a map. ■

Lemma 2. *If P is a path in a MICE-Tree $<_P$ is a strict partial order.*

Proof: From definition 10, $\varphi(V_*(n_i), V_*(n_j))$ is a strict partial order for any pair of nodes $n_i, n_j \in N, i \neq j$ (i), and definition 6 enforces that the sets of vertices of all tree nodes are disjoint $V_*(n_i) \cap V_*(n_j) = \emptyset$ if $n_i \neq n_j, \forall n_i, n_j \in N$ (ii), therefore we have:

From definition 2, $<_P$ is a strict partial order iff:

(1) $(a, b) \in <_P$ implies that $a \in V_*(n_i)$ and $b \in V_*(n_j)$ with $i \neq j$, and therefore because $V_*(n_i) \cap V_*(n_j) = \emptyset$ from (ii), we have $a \neq b$

(2) According to definition 13, $(a, b) \in <_P$ implies that $a \in V_*(n_i)$ and $b \in V_*(n_j)$ with $i < j$, and therefore because $V_*(n_i) \cap V_*(n_j) = \emptyset$ we have $(b, a) \notin <_P$.

(3) let $(a, b) \in <_P$, and $(b, c) \in <_P$, from definition 7 we have $a \in V_*(n_i)$, $b \in V_*(n_j)$ and $c \in V_*(n_k)$ such that $i, j, k \in \psi(P)$, and from definition 13, $i < j < k$, and therefore $(a, c) \in <_P$. ■

Therefore, from lemmas 1 and 2 we can draw the following theorem:

Theorem 3. *If P is a path in a MICE-Tree T, then $\xi(P) = (V_P, <_P, g_P)$ is an episode.*

Proof: According to definition 1, the triplet $(V_P, <_P, g_P)$ defines an episode iff. (1) $g_P : V_P \rightarrow \mathcal{E}$, proved by lemma 1; and (2) $<_P$ is a strict partial order, proved by lemma 2. ■

An example of a MICE-Tree is then illustrated in Fig. 4. In this figure, each green square denote a non-leaf node $n \in N$, that code a parallel episode—the letters in the box indicate the events $E_i \in \{g_*(n)(v) : v \in V_*(n)\}$. The blue boxes denote leaf nodes n^* associated to a valid label $\lambda_*(n^*) \in \{c1, c2, c3\}$, and below are indicated the episodes that correspond to each leaf node's path. Finally, the orange boxes denote rejection leaf nodes where no assertion can be made on the episode's class and the blue arrows denote tree links $l \in L$.

C. Classifying Datastreams using MICE-Trees

In order to classify an input sequence S given a MICE Tree $T = (N, L)$, we define the following recursive function $C_T : N \times S_\varepsilon \rightarrow \mathcal{L}$:

$$C_T(n, S) = \begin{cases} C_T(n^+, S) & \text{if} \quad \xi(P_n) \sqsubseteq S, V_n \neq \emptyset \\ C_T(n^-, S) & \text{if} \quad \xi(P_n) \not\sqsubseteq S, V_n \neq \emptyset \\ \lambda_n & \text{otherwise} \end{cases} \tag{9}$$

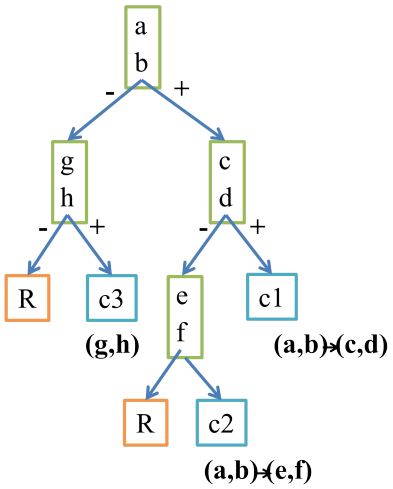


Figure 4. Example of a MICE-Tree.

where the tree node n is the triplet, $n = (V_n, g_n, \lambda_n)$, and n^+, n^- are the positive and negative child-nodes of n respectively: $(n, n^+, +), (n, n^-, -) \in L$. It is now possible to define a multi-class classification function for labelling an input sequences given a MICE-Tree with root node r:

$$h_T(S) = C_T(r, S) \tag{10}$$

This is achieved in an computationally efficient manner using Algorithm 1.

Algorithm 1 MICE-Tree Classifier: $\mathbf{P} = C_T(S)$

Input: Input datastream $\mathbf{S} =< (S_i, t_i) >_{i=1}^{|S|}$
Input: MICE-Tree $T = (N, L)$, root node r.
Output: Label of $S : \lambda \in \mathcal{L}$
Initialise current node to root node: $n^{cur} = r$
The contents of current node: $n^{cur} = (V_{cur}, g_{cur}, \lambda_{cur})$
Init start offset: $e = 1$
while n^{cur} is not leaf-node **do**
  For each $k \in [1, |V_{cur}|]$ get:
    $G_k = \{j : j \in [e, |S|], S_j = g_{cur}(v_k)\}$
  $Z = \bigcap_{k \in [1, |V_{cur}|]} [min(G_k), |S|]$
  if $Z = \emptyset$ **then**
    $n^{cur} = m$, such that $(n^{cur}, m, -1) \in L$
  else
    $n^{cur} = m$, such that $(n^{cur}, m, +1) \in L$
    $e = \min_{k \in [1, |V_{cur}|]}(\min(G_k))$
  end if
end while
Return λ_{cur}

III. Learning MICE Trees

In this section, we propose a novel method for constructing a MICE-Tree given a database of weighted and labelled data streams. Here, the construction of a MICE-Tree is achieved in a greedy and recursive manner, whereby the multi-class training dataset is recursively partitioned into smaller and smaller subsets that are distributed across different nodes of the MICE-Tree. To achieve this, we theoretically show that a node in the MICE-Tree does indeed induce a binary partition on the training dataset in Section III-A. However, it is also

 ISBN: 978-1-61208-465-7

important that there is a method for measuring the optimality of such a binary partition. To this end, a node-split criteria based on the Gini impurity measure is described in Section III-B. Finally, the MICE-Tree learning algorithm is described in Section III-C

A. MICE-Tree Node Binary Partition

For the purpose of learning, we are provided with a training data stream collection that we will denote as $\mathcal{D}$ (defined in Section II). Additionally, we are given a set of weights $W = (w_i)_{i=1}^{|\mathcal{D}|}$, where each example $(S_i, \lambda_i) \in \mathcal{D}$ is associated with the weight w_i.

In order to construct a MICE-Tree T, we firstly introduce a useful function $D : S_\varepsilon \rightarrow 2^N$ based on Eq. 9 that extracts the set of nodes a datastream example X passes through when it is classified by T (using Eq 9):

$$D_T(n, S) = n \cup \begin{cases} D_T(n^+, S) & \text{if} \quad \xi(P_n) \sqsubseteq S, V_n \neq \emptyset \\ D_T(n^-, S) & \text{if} \quad \xi(P_n) \not\sqsubseteq S, V_n \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases} \tag{11}$$

Both functions in D_T (Eq. 11 and C_T (Eq. 9) provides a deterministic mechanism for classifying an input datastream:

Remark 1. Let S be an input datastream, $T = (N, L)$ be a MICE-Tree and r its root node, then there exists only one unique path, whose nodes are $D_T(r, S)$, that will be used to classify it. That is, a sequence S will always go through the same path when C is used to classify it.

It can be observed that the properties of Eq. 11 will allow us to see that each node in a MICE-tree "captures" a subset of training datastreams from $\mathcal{D}$: Since each example training datastream can only "carve" a single unique path through a tree during classification (Rem. 1), then, only a subset of example datastreams will pass through a particular node in a tree. More specifically:

Definition 14. Let $T = (N, L)$ be a MICE-Tree, $\mathcal{D}$ be the labelled collection of datastreams. Let $n \in N$ be a node and $m \in N$ be its parent node, with P_m being the path from the root node to m. We then define the set of *captured example indices* of n as the following set: $\mathcal{I}_n = \{i \in [1, |\mathcal{D}|] : \xi(P_m) \sqsubseteq S_i, n \in D(r, S_i)\}$.

A consequence of the above definition is that a non-leaf node n will induce a binary partitioning of its captured example set $\mathcal{I}_n$ into a *positive partition*, $\mathcal{I}_n^+$, and *negative partition*, $\mathcal{I}_n^-$ where $\mathcal{I} = \mathcal{I}_n^+ \cup \mathcal{I}_n^-$ and:

$$\mathcal{I}_n^+ = \{i \in I : \xi(P_{n^+}) \sqsubseteq S_i\} \tag{12}$$

$$\mathcal{I}_n^- = \{i \in I : \xi(P_{n^+}) \not\sqsubseteq S_i\} \tag{13}$$

The partitioning induced by a node n is clearly dependent on its contents (i.e. V_n and g_n). Thus, for learning purposes, we define the following operation on a node:

Definition 15. Given a node $n = (V_n, g_n, \lambda_n)$ in a MICE-Tree, a new (episode vertex,event) pair (v', e') can be *added* into n, denoted as $n' = n + (v', e')$, where $e' \in \varepsilon$ and a v' is an episode-vertex. The new node is: $n' = (V_{n'}, g_{n'}, \lambda_{n'})$ with $V_{n'} = V_n \cup \{v'\}$, $g_{n'} = g_n \cup \{(v', e')\}$ and $\lambda_{n'} = \lambda_n$),

Importantly, adding (episode-vertex, event) pairs into any node has the effect of moving examples from its positive partition to the negative partition.

Theorem 4. *Let $n = (V_n, g_n, \lambda_n)$ be a node in a MICE-Tree, $\mathcal{I}_n$ the index set of examples captured by n and $\mathcal{I}_n^+$, $\mathcal{I}_n^-$, its induced partition respectively. Then, adding an (episode-vertex,event) pair, (v', e') to n will produce a new node, $n' = n + (v', e')$, with a potentially smaller positive partition, $\mathcal{I}_{n'}^+ \subset \mathcal{I}_n^+$ and potentially larger negative partition, $\mathcal{I}_{n'}^- \supset \mathcal{I}_n^-$.*

Proof: It is clear that the episode modelled by the path from the root node r to n is a subsequence that from r to n': $\xi(P_n) \sqsubseteq \xi(P_{n'})$. Then, by the Apriori rule, we have $\{i \in \mathcal{I}_n : \xi(P_{n'})\} \subset \{i \in \mathcal{I}_n : \xi(P_n)\}$ ■

Algorithm 2 GetBestSplit Algorithm

Input: Train Set: $\mathcal{S} = \{(\mathbf{S}_i, \lambda_i, w_i)\}_{i=1}^{|S|}$
Output: MICE-Tree Node, +ve partition, -ve partition, error
Let $\lambda_{\mathcal{S}}$ be the class with highest frequency in $\mathcal{S}$.
Initialise empty node: $n = (\emptyset, \emptyset, \lambda_{\mathcal{S}})$.
Initialise the event set: $E = \mathcal{E}$.
Initialise error: $\gamma_{best} = -1$
while $E \neq \emptyset$ **do**
  Let v' be an episode-vertex not present in any tree-node.
  Find $e_{best} \in E$ s.t. $n + (v', e_{best})$ minimises γ from Eq. 14 given weights w_i.
  if $\gamma_{best} < 0$ or $\gamma_{best} > \gamma$ **then**
    $n = n + (v', e_{best})$
    $E = E - \{e_{best}\}$ {Remove e_{best} from the set E}
  else
    break
  end if
end while
Let the path from the root node to n be P_n.
$\mathcal{S}_n^+ = \{(S, \lambda, w) \in \mathcal{S} : \xi(P_n) \sqsubseteq S\}$
$\mathcal{S}_n^- = \{(S, \lambda, w) \in \mathcal{S} : \xi(P_n) \not\sqsubseteq S\}$
Return $(n, \mathcal{S}_n^+, \mathcal{S}_n^-, \gamma_{best})$.

B. Node-Split Criteria

The final tool required for learning tree is a method of evaluating how "good" a node split is. This is achieved using an adapted Gini impurity that is popular in decision tree learning [13]. Here, we have changed the criteria to account for weighted training examples. Suppose we have found that non-leaf node n has caused a partition $\mathcal{I}^+$ and $\mathcal{I}^-$. Suppose the corresponding weights of these partitions are W'_+ and W'_- respectively. Similarly, let the corresponding labels be Y'_+ and Y'_- respectively. We define the total positive and negative partition weight coefficient as: $Z^+ = \sum_{i=1}^{|W'_+|} w'_{+,i} / \sum_{i=1}^{|W'|} w'_i$ and $Z^- = \sum_{i=1}^{|W'_-|} w'_{-,i} / \sum_{i=1}^{|W'|} w'_i$ respectively. Using both the weights and labels, it is possible to compute a normalised label histogram for each partition: F'_+ and F'_- respectively. The node-split criteria is defined as:

$$\gamma = Z^+(1 - \sum_{i=1}^{C} f_{+,i}^2) + Z^-(1 - \sum_{i=1}^{C} f_{-,i}^2) \tag{14}$$

 ISBN: 978-1-61208-465-7

Algorithm 3 MICE-Tree Learn Algorithm

Input: Training Set: $\{(\mathbf{S}_i, \lambda_i)\}_{i=1}^{|S|}$, $(w_i)_{i=1}^{|S|}$
Output: MICE-Tree $T = (N, L)$
Make Offseted Train Set: $\mathcal{S} = \{(\mathbf{S}_i, \lambda_i, w_i, o_i)\}_{i=1}^{|S|}$, $o_i = 1$
Queue element: $(ParentNode, TrainSubset, Depth, LinkType)$
Get optimal root node:
$(r, \mathcal{S}^+, \mathcal{S}^-, \epsilon) =$ GetBestSplit($\mathcal{S}$)
Initialise the queue:
$Q = \{(r, \mathcal{S}^+, 2, +), (R, \mathcal{S}^-, 2, -)\}$.
while $Q \neq \emptyset$ **do**
  Remove last item of Q: $(n^{cur}, \mathcal{S}^{cur}, D^{cur}, s^{cur})$
  if $|\mathcal{S}^{cur}| \leq \alpha$ OR $D^{cur} \geq \beta$ **then**
    Get class (λ) with highest weighted freq. in $\mathcal{S}^{cur}$
      $\lambda = -1$ if $\mathcal{S}^{cur} = \emptyset$.
    $m = (\emptyset, \emptyset, \lambda)$
    $N = N \cup \{m\}$ {Add new tree-node}
    $L = L \cup \{(n^{cur}, m, s^{cur})\}$ {Link to new tree-node}
    break
  end if
  Get optimal current node:
    $(m, \mathcal{S}^+, \mathcal{S}^-, \epsilon) =$ GetBestSplit($\mathcal{S}^{cur}$)
  $N = N \cup \{m\}$ {Add new tree-node}
  $L = L \cup \{(n^{cur}, m, s^{cur})\}$ {Link to new tree-node}
  if $\epsilon > 0$ **then**
    $Q = Q \cup \{(m, \mathcal{S}^+, D^{cur} + 1, +), (m, \mathcal{S}^-, D^{cur} + 1), -\}$.
  end if
end while
Return MICE-Tree: $T = (N, E)$

C. MICE-Tree Learning Algorithm

The algorithm for learning the MICE-Tree is given in Algo. 3, where a MICE-Tree is constructed in a greedy and recursive manner. One key mechanism for constructing MICE-Trees is given in Algo. 2. Here, episode-vertices are sequentially added in a greedy fashion to the parent nodes of leaf nodes to maximally improve the splitting criteria (Eq. 14). This process terminates when the splitting criteria cannot be improved.

Algo. 3 then starts by constructing the root node using Algo 2. This induces a partitioning of the dataset into two training subsets (i.e. +ve and -ve partitions). Two new children tree nodes (+ve and -ve) are constructed and linked to the root node. Each training subset is then passed on to its respective child node. The algorithm then recursively applies Algo 2 to configure the contents of both child nodes.This recursive process is performed via a queue-based system until one of 3 termination criteria is reached: 1) maximum tree-depth β is reached; 2) training subset is smaller than minimum size α (set here as 1); 3) The training subset is "pure" (i.e. only belongs to a single class).

IV. Boosting MICE Forests

In this section, we describe the method for learning and combining the multiple MICE-trees in order to produce a robust and accurate classifier that generalises to unseen novel sequences in the presence of noise. To this end, we propose a novel machine learning method for learning strong classifiers based on MICE-Tree within the Multi-class AdaBoost framework [14]. A strong classifier outputs a class label based on the maximum votes cast by a number (S) of selected and weighted weak classifiers:

$$H(I) = \arg\max_c \sum_{i=1}^{S} \alpha_i \mathbb{I}(h_i(I) = c) \quad (15)$$

In this paper, the weak classifiers h_i are the MICE-Tree classifiers defined in Section II-C as Eq. 10. Each weak classifier h_i is selected iteratively with respect to the following error:

$$\epsilon_i = \sum_{i=1}^{X} \mathbb{I}(h_i(X_i) \neq y_i) \quad (16)$$

Typically, in order to determine the optimal weak classifier at each Boosting iteration, the common approach is to exhaustively consider the entire set of possible weak classifiers and finally select the best weak classifier (i.e. that with the lowest ϵ_i). Such an exhaustive search will not be possible due to the large search space of possible MICE-Trees. Thus, Algo. 3 is used for choosing the appropriate MICE-Tree weak classifier instead given a set of training datastreams with associated boosted weights.

The final MICE-Tree Boosting algorithm is detailed in Algo. 4. We have chosen to iteratively learn new MICE-Tree based on the multi-class AdaBoost method. However, we are not limited to this particular form of Boosting and it would be easy to integrate the MICE-Tree learning algorithm (Algo. 3) into other Boosting methods (e.g. GentleBoost, etc...).

Algorithm 4 MICE-Tree-Boost Algorithm

Initialise example weights: $\forall w_i \in W, w_i = 1/X$
for $t = 1, ..., M$ **do**
  Select ($h_t = h^{\mathbf{T}_{best}}$) using Algo. 3
  Obtain the classification error ϵ_t for h_t (Eq. 16)
  Obtain the weight $\alpha_t = \ln \frac{1-\epsilon_{best}}{\epsilon_{best}} + \ln(C-1)$
  Update weights: $w_i = w_i \exp(-\alpha_i [h_t(X_i) \neq y_i])$
  Normalise weights: $\sum_{i=1}^{X} w_i = 1$
end for
Return the strong classifier:
$H(X) = \arg\max_c \sum_{i=1}^{M} \alpha_i \mathbb{I}(h_i(X) = c)$

V. Experimental evaluation

In this section we evaluate the proposed approach using two different sets of generated data streams, that allow us to control for properties of the encoded episodes. The first experiment, in section V-A was designed to illustrate the limitations of the more common Sequential pattern framework (as used by [10], [11]), and the advantages of the more generic episodes framework presented herein. The second experiment, in section V-B was designed to have a thorough evaluation of the MICE-Trees robustness to noise in the data streams.

A. Exp 1: Comparison with SP-trees

The first experiments evaluates the special case where classes cannot be discriminated by purely serial episodes, and therefore form a special challenge for SPs and should benefit from the greater generality of the episode model.

In order to assess this, we generated a specifically designed dataset where all serial episodes are ambiguous. We achieved

 ISBN: 978-1-61208-465-7

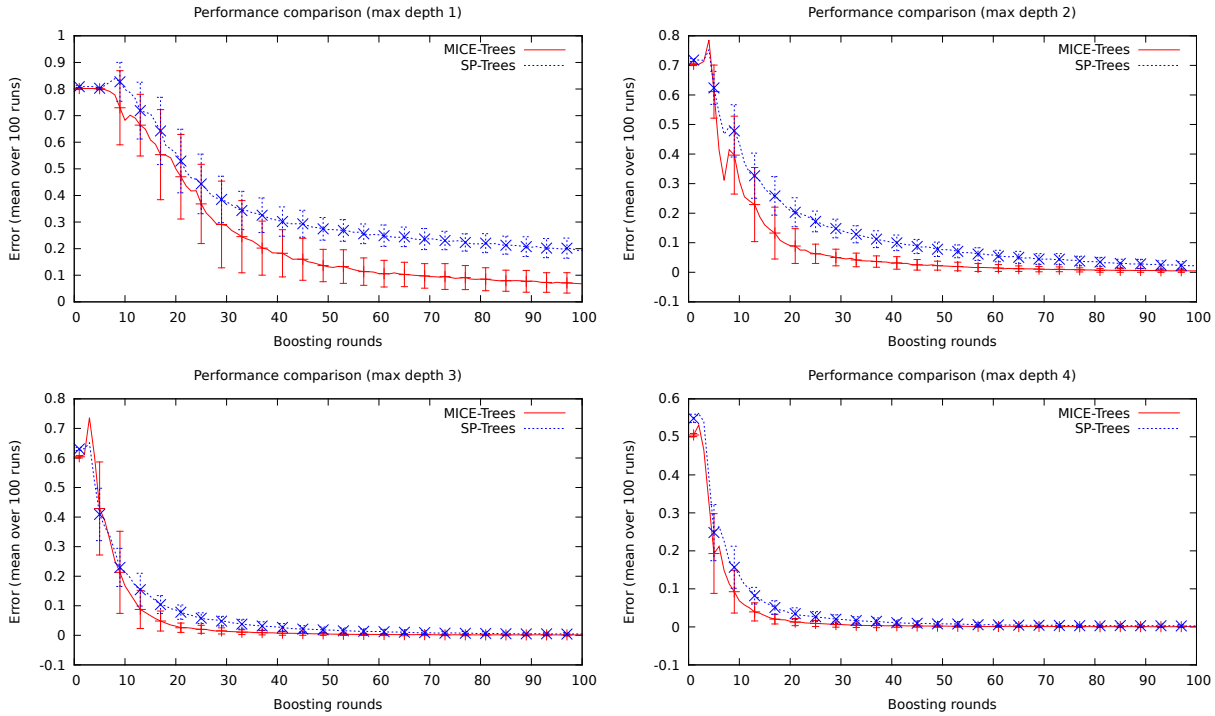


Figure 5. Performance of the classification for the dataset A (10 classes). The graphs show the mean classification error over 100 randomised tests, for rounds of Boosting from 1 to 100 and for maximal tree depths from 1 to 4. The error bars show the standard deviations for all curves.

this by generating 10 classes $A, \ldots, J$, each characterised by a unique episode of the form: $\alpha^\lambda = (E_1^\lambda, E_2^\lambda) \rightarrow (E_3^\lambda, E_4^\lambda)$, such that there are no two class episodes that share any event. Then we generated exhaustively a collection of streams that satisfy each class λ, and injected noise in the resulting streams by sampling from all possible sequence of events present in other classes $\lambda' \neq \lambda$ that do not satisfy any class episode. For example, some streams of class A perturbated with events from class B would be:

$$\begin{aligned} S_1 &= \big\langle (E_1^A,1),(E_2^A,2),(E_1^B,3),(E_2^B,4),(E_4^B,5),(E_3^A,6),(E_4^A,7)\big\rangle \\ S_2 &= \big\langle (E_2^A,1),(E_1^A,2),(E_1^B,3),(E_4^B,4),(E_3^B,5),(E_3^A,6),(E_4^A,7)\big\rangle \\ S_3 &= \big\langle (E_1^A,1),(E_2^A,2),(E_2^B,3),(E_3^B,4),(E_4^B,5),(E_4^A,6),(E_3^A,7)\big\rangle \\ &\vdots \end{aligned} \tag{17}$$

In this case, the E_i^B elements (shown in blue in Eq. 17), injected in the middle, will contain all *serial* episodes that define the episode α_B—hence only the *generic* episodes are unique to the classes we generated this way.

We then learnt a collection of 100 SP-Trees and 100 MICE-Trees from this data, using 500 training samples out of a collection of 2,880 streams and testing on the rest. The results, for different values of maximal tree depth are shown in Fig. 5. There, we can see that SP-Trees require significantly mode complex models, in terms of both number and depth of trees, to match the performance of MICE-Trees, confirming our assertion that the episode framework allow for a more efficient and compact encoding of certain classes of temporal patterns.

B. Exp 2: Robustness to noise

The aim of this second experiment was to assess the robustness of the MICE-Trees classification for a large number of classes (we used 100 classes) denoted by temporal patterns corrupted by increasing amounts of noise.

1) Episode generation: In order to generate the dataset we first defined a vocabulary $\mathcal{E}$ of 26 symbols. Then for each of the classes we generated a signature episode by the following steps: first, we generated a single sequence of events $(E_1, \ldots, E_N) \in \mathcal{E}^N$, where $N = 18$ is the number of events in the sequence; second, in order to transform this purely serial episode in a generic one, we permuted randomly a number of contiguous pairs of symbols. Hence if the original sequence created was

$$(E_1) \rightarrow (E_2) \rightarrow (E_3) \rightarrow (E_4) \rightarrow (E_5)$$

then after permutation of the leftmost (E_1, E_2) and rightmost (E_4, E_5) adjacent pairs of events, we obtain the alternative sequence

$$(E_2) \rightarrow (E_1) \rightarrow (E_1) \rightarrow (E_4) \rightarrow (E_3)$$

and therefore this class is best described by the episode

$$(E_1, E_2) \rightarrow (E_3) \rightarrow (E_4, E_5).$$

We use the same procedure to randomly generate unique episodes for each of the 100 classes.

2) Episode corruption with temporal noise: All the patterns generated at this point are perfect occurences of the class' episode: there are no spurious events in the data stream. In order to be able to control the proportion of spurious events in the training and testing data streams, we propose modelled the pattern generation as a Markov process. The process starts at the beginning of the pattern and from there has a chance α to move onto the next event in the noise-free pattern. Conversely, the process has a $1 - \alpha$ chance to generate a noisy event E^ϵ.

 ISBN: 978-1-61208-465-7

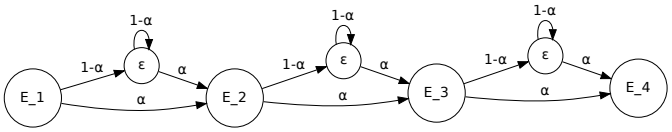


Figure 6. Illustration of the noise injection process in a noise-free sequence (E_1, E_2, E_3, E_4). The generation is processed as a Markov process where the chance to insert spurious events ε (randomly generated) is dependent on a purity parameter α.

Hence, a purity parameter of $\alpha = 1$ will generate a noise-free sequence, and a purity parameter of $\alpha = 0$ would generate an infinite sequence of randomised events. Moreover, the length of the generated pattern increases with diminishing values of α. This Markov process is illustrated in Fig. 6 for a simple noise-free sequence.

Moreover, and in order to make the injected noise more structured, spurious event E_k^ϵ are not generated using a uniform distribution, but using a noise generator that keeps a memory of the previous noisy event generated, and produces the next one using an a priori, randomly generated, Markov transition matrix encoding inhomogeneous transition probabilities $p(E_k^\epsilon | E_{k-1}^\epsilon)$—this noise generation approach ensures that the injected noise will be more likely to feature structure and repeating patterns.

The usage of the α parameter and Markov chain allows the pattern to be heavily corrupted, Figure 7 shows the corruption of a pattern with noise for decreasing levels of α.

$\alpha = 1.0$ {A W W Q W V W M B Q E U X K X F B U}
$\alpha = 0.8$ {A W W Q S W V W P M B Q E U X K X F B U S}
$\alpha = 0.5$ {A U W A C W Q E P W V H I E W I S M B Q E D U L G A X K O X F I S B K A U E H P}
$\alpha = 0.3$ {P X J A A W K I W Q V Q T W F J V B Q O F L W M M G B Q U U A H C G M K X H J E U U X K X F S B W M U}

Figure 7. A example of the pattern corruption for increasing α

Note that the pattern length increases quickly with noise; for values of α lower than 0.5 the proportion of noise elements is greater that the original pattern.

3) Results: Figure 8 shows the performance of the MICE-Trees classification for 100 classes and values of α varying between 20% and 100%. The full red line shows the classification error and the dashed blue line shows the Signal to Noise Ratio (SNR) for this value of α. This graph shows that the MICE-Trees yield excellent classification results, providing near-perfect classification for values of α above 75%. For values of α below 50%, the classification performance drops sharply, illustrating the fact that diminishing SNR values make the learning extremely challenging.

Overall, this shows that the MICE-Trees can provide excellent classification performance event in presence of large amount of spurious events and can handle a large number of classes.

VI. Conclusions

In this work, we presented a theoretic definition of a tree structure called MultI-Class Episode Trees (MICE-trees) allowing for multiple classes to share sub-episodes whilst providing the ability to classify datastreams. We then proposed an efficient algorithm for learning Boosted collections of such multi-class episode trees. The performance of the proposed model was then evaluated using two sets of experiments. The

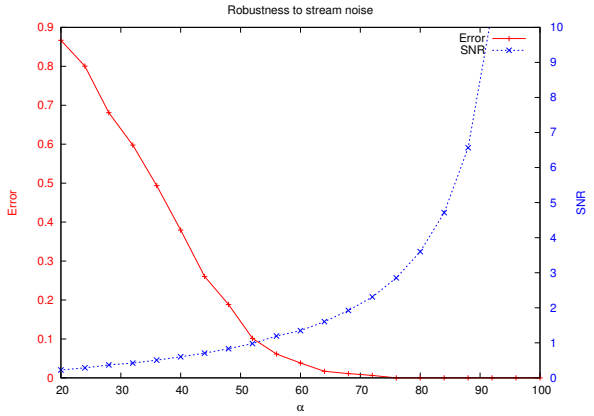


Figure 8. This figure illustrates the robustness of the MICE-Trees to noise injected in the data streams.

first demonstrated that MICE-Trees allow for a better and more compact representation of certain type of temporal sequences when compared to sequential patterns. We also show that the resulting classifiers can cope with a large amount of signal noise while still providing good classification accuracy.

References

[1] H. Mannila, H. Toivonen, and A. Verkamo, "Discovering frequent episodes in sequences," in Proc. of Int. Conf. on Knowledge Discovery and Data Mining (KDD'95), 1995, pp. 210–215.

[2] R. Agrawal and R. Srikant, "Mining sequential patterns: Generalizations and performance improvements," in Proc. of 5th International Conference on Extending Database Technology, 1996.

[3] M. Zaki, "Spade:an efficient algorithm for mining frequent sequences," Machine Learning, 2001.

[4] J. Han, J. Pei, and Y. Yin, "Mining frequent pattern without candidate generation," in Proc. of International Conference on Management of Data (SIGMOD), 2000.

[5] J. Han, J. Pei, B. Mortazavi-Asl, and H. Zhu, "Mining access patterns efficiently from web logs," in Proc of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2000.

[6] H. Mannila, H. Toivonen, and A. Verkamo, "Discovery of frequent episodes in event sequences," Data Mining and Knowledge Discovery, vol. 1, 1997, pp. 259–289.

[7] N. Tatti and B. Cule, "Mining closed strict episodes," in Proc. of the Int. Conf. on Data Mining (ICDM'2010), 2010.

[8] ——, "Mining closed strict episodes," Data Mining and Knowledge Discovery, vol. 25, 2012, pp. 34–66.

[9] A. Achar, S. Laxman, R. Viswanathan, and P. Sastry, "Discovering injective episodes with general partial orders," Data Mining and Knowledge Discovery, vol. 25, 2012, pp. 67–108.

[10] S. Nowozin, "Discriminative subsequence mining for action classification," in IEEE Int. Conf. in Computer Vision (ICCV'2007), 2007.

[11] E. Ong, H. Cooper, N. Pugeault, and R. Bowden, "Sign language recognition using sequential pattern trees," in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'2012), 2012.

[12] E. Ong, O. Koller, N. Pugeault, and R. Bowden, "Sign spotting using hierarchical sequential patterns with temporal intervals," in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'2014), 2014.

[13] L. Brieman, J. Friedman, R. Olshen, and C. Stone, Classification and regression trees. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[14] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," Statistics and Its Interface, vol. 2, 2009, pp. 349–360.

 ISBN: 978-1-61208-465-7

 ISBN: 978-1-61208-465-7

 ISBN: 978-1-61208-465-7

Car User Experience Patterns: A Pattern Collection in Progress

Tim Kaiser, Alexander G. Mirnig, Nicole Perterer, Alexander Meschtscherjakov, Manfred Tscheligi
Christian Doppler Laboratory for "Contextual Interfaces"
Center for Human-Computer Interaction, University of Salzburg
Salzburg, Austria
Email: {firstname.lastname}@sbg.ac.at

***Abstract*— Car user experience patterns are a systematic way to capture best practices and solutions to reoccurring problems in automotive interaction design. Combining empirical data, industry knowledge, and experts experience, they facilitate communication between scientists and industry stakeholders. In this paper, we present a newly generated set of eight car user experience patterns that describe answers to problems in automotive interaction design and engineering. These patterns are part of an ongoing project with the aim of providing a comprehensive, User Experience focused, pattern collection. The patterns presented in this paper mainly contain information on reducing potential distraction caused by the usage of in-vehicle information systems and on designing efficient in-car warning systems. They are the result of a novel approach combining scientific and industry know-how into very brief and domain-specific design problem solutions.**

Keywords*— *design patterns; pattern identification and extraction; pattern reuse.

I. INTRODUCTION AND RELATED WORK

The usage of patterns is well established in Human-Computer Interaction (HCI) and is advantageous for various reasons. First, patterns are a method to capture proven design solutions to reoccurring problems. Second, the use of patterns improve the design process (regarding both time and effort spent) to a considerable degree [1][2]. Moreover, scientific research in HCI also strongly relies on communicating scientific findings to the industry. By translating these findings so that they convey relevant and useful information to designers and developers, patterns can help facilitate the design process by reducing time and effort that has to be put into it.

Designing for a good User Experience (UX) has become an increasingly important topic in academia and industry [3][4][5]. User Experience can be defined as *"the users sensory, emotional and reflective response to the interaction with a system in a context"* [6]. The car industry in particular has become a fast-paced global market that can draw substantial benefits from a modular and flexible documentation of best practices.

Based on this consideration, we created a set of car User Experience patterns, of which we present the eight most recent car ones in this paper. In the following two subsections, we will give an overview on the state of the art on Contextual UX patterns in general, and our approach in particular. In Section II, we show each pattern in its entirety, and conclude with a brief summary in Section III.

A. Contextual User Experience Patterns

Recently, specific domains in HCI, such as UX research, employed patterns to collect and structure their knowledge based on empirical findings [2][7][8]. This is illustrated, e.g., by Martin et al. [9] and Crabtree [10] who use patterns for organizing and presenting ethnographic material. The first who emphasized a focus on the human perspective in the history of patterns was Alexander [11]. In 2010, Blackwell and Fincher [5] suggest to adopt the idea of patterns and UX in the form of Patterns of User Experience (PUX). Such patterns should help HCI professionals to understand what kind of experiences people have with information structures.

In the same year, Obrist et al. [2] developed 30 UX patterns for audiovisual networked applications based on a huge range of collected empirical data, which was further categorized into main UX problem areas. An extension of these UX patterns, are the so-called Contextual User Experience (CUX) patterns. Accordingly, patterns are used to describe the knowledge on how to influence the users experience in a positive way by taking context parameters during the interaction with a system into account. Within their work, the authors provide a detailed description of how to structure CUX patterns in the car context. Three years later, Krischkowsky et al. [8] presented a step-by-step guidance for HCI researchers for generating patterns form HCI study insights. In particular, they intended to support User Experience (UX) researchers in converting their gathered knowledge from empirical studies into patterns. The structural foundation for the intended patterns is the so-called Contextual User Experience (CUX) patterns format, as mentioned before.

Following in the footsteps of Obrist et al. [7], we decided to pursue a triangular approach towards driver space design and cover three major UX factors via appropriate design patterns.These factors are:

- *Mental Workload Caused by Distraction* [12]: Safety is paramount in an automotive environment, and distraction is one of the major contributing factors to accidents on the road [13][14]. Especially in UX, where functionalities and interface complexities are ever increasing, this is one of, if not the, most important factors to consider regarding driver safety.
- *Perceived Safety* [15]: The increased safety gained by designing for decreased mental workload and less dis-

 ISBN: 978-1-61208-465-7

traction needs to be communicated to the driver. The difference between objective and perceived safety can be relatively large. For many situations, it has to be evaluated if car interfaces should increase or decrease perceived safety.

- *Joy of Use* [16][17]: This is strongly tied to the previous factor, but is not the same. Cars have more and more become instruments that are not simply means of transportation but are also used for entertainment. Thus, it becomes important that car interfaces can be used not only without frustration, but also in a way that makes using them a joyful experience.

B. Pattern Generation Process

The pattern generation process for car user experience patterns has been described in detail by Mirnig et al. [18]. First, an initial knowledge transfer workshop was conducted in order to provide HCI researchers with know-how regarding patterns. The HCI researchers then generated an initial set of 16 patterns following the initial pattern structure. This lead to several issues with the initial structure, so that an iteration of the initial structure took place in a second workshop. The resulting refined pattern structure consists of 9 parts: *Name* (a description of the solution of the pattern), *Intent* (a short abstract to allow quick judgment whether the pattern can be applied in a certain context), *Topics* (problem scope and addressed automotive user experience factor), *Problem* (a short but more detailed description of the problem which should be solved by the pattern), *Scenario* (an example application context of the pattern), *Solution* (the proposed solution), *Examples* (concrete examples of best practices), *Keywords* (other topics related to the pattern), and *Sources* (origin of the pattern).

In the next step, patterns were presented to industry stakeholders in a pattern evaluation workshop. Based on their feedback, the name, intent and topics section were standardized and kept brief, so it takes less time and effort to process them. The context and forces sections were combined into the new *scenario* category.

II. PATTERN COLLECTION

We developed a list of design problems together with designers and engineers working in the automotive industry and applied the aforementioned pattern generation approach, involving the industry stakeholders at several stages in the process. The following is one part of a resulting collection of patterns, which combines scientific and industry know-how into concrete problem solutions for UX-centered driver space design problems in the automotive domain.

A. Pattern 1: Menu Depth and Number of Options

Intent: This pattern is about reducing distraction caused by navigating visual menus as a secondary task.

Topics: Workload caused by distraction, driver, haptic, input

Problem: While driving, navigation of in-vehicle user interface menus causes distraction. Given the safety implications of visual distraction, it is important to minimize visual demand of these menus.

Scenario: Drivers interact with visual menus to access information, communication and entertainment systems. Navigating menus with high visual demand severely distracts the driver and can thus lead to road deviations and crashes. Visual demand of menus is determined by a depth/breadth-trade-off. The deeper a menu, the less menu options per page there should be. A National Highway Traffic Safety Agency (NHTSA) guideline based on current research recommends that a driver should be able to complete a task in a series of 1.5 second glances with a cumulative time spent glancing away from the roadway of not more than 12 seconds [19].

Solution: Designing menus with limited depth allow drivers to complete secondary tasks in a relatively short time period. With the help of an empirically derived formula provided by Burnett et al. [20], it is possible to calculate different menu structures that comply with design guidelines:

$$T = D(0.87 + 1.24 * log(B)$$

where *T = time to complete the task, D = depth of menu* where *B = number of menu options. Table I* shows acceptable menu structures that comply with maximum task completion time according to the NHTSA guideline, as calculated using this formula.

TABLE I
MENU DEPTH AND NUMBER OF OPTIONS FOLLOWING NHTSA GUIDELINES

Menu Depth	Menu Breadth
3	12
4	5
5	3
6	2

Examples: see Figures 1 and 2.

B. Pattern 2: Display Touch Field Size

Intent: This pattern is about determining the optimal touch screen target size.

Topics: Workload caused by distraction, driver, touch screen, visual, haptic, input

Problem: Navigating in-vehicle displays while driving causes distraction, leading to road deviations and possibly to crashes. Thus, visual demand of touch screen menus has to be minimized while preserving maximum usability.

Scenario: Because they are easy to use and to understand, touch-screen interfaces are more and more used for operating in-vehicle systems. Drivers use them to control entertainment and navigation features provided by these systems as a secondary task. The key factor for navigating these displays easily is the size of the touch target like a menu button [21]. Subjective usability ratings, as well as objective measures like task completion time and error rate heavily depend on this factor.

 ISBN: 978-1-61208-465-7

Figure 1. BMW iDrive - accessing vital information requires only three navigation steps.

Figure 2. BMW iDrive - changing the radio station requires only two steps.

Solution: Touch targets need to be large enough in order to minimize task completion time and error rate. Design guidelines suggest a minimum contact surface area of 80 mm [19]. However, in a recent driving simulation study that focused on touch target size for in-vehicle information systems, the authors determined that a touch key size of at least 17.5x17.5 mm minimizes navigation error rate, lane deviations, driving speed variation and glance time while maximizing subjective usability ratings [22]. While touch screen size and overall visual complexity of the menu always have to be taken in consideration, the recommended touch key size may serve as a starting point for menu design.

Examples: See Figure 3.

Figure 3. Apple Car Play menu.

C. Pattern 3: Auditory Informations and Warnings

Intent: this pattern is about designing auditory informations and warnings that are quick to capture and easy to comprehend.

Topics: perceived safety, driver, acoustic, output

Problem: When using only visual warnings, driver distraction can occur. Still, drowsiness and inattentiveness increase the risk of traffic accidents. Thus, it is still necessary to direct the drivers attention to potential dangers by different means.

Scenario: Well-designed auditory warning systems can serve this purpose. Perceptibilty of auditory warnings depends on loudness, background noise and complexity. Also, the driver needs to know which actions have to be taken to react appropriately.

Solution: Different warning techniques are appropriate for different situations. According to Bliss and Acton [23], verbal speech notifications and auditory icons (sounds with real-world representations, e.g., the sound of a car engine) are equally efficient when it comes to response accuracy and reaction time. Auditory warnings also have to convey enough information to be accurately understood. Due to driving comfort reasons, warnings of low urgency should not be annoying and can even be quite pleasant, while high-urgency warnings are bound to be annoying [24].

Examples: **Table II** shows auditory warnings for some common situations of varying urgency. Empirical work on the perceived urgency of speech based warnings has been done [25].

D. Pattern 4: Choosing the Best Modality for Warning Displays

Intent: this pattern is about choosing the right warning display modality for different situations, combining different

 ISBN: 978-1-61208-465-7

TABLE II
RECOMMENDED WARNINGS FOR COMMON SITUATIONS OF VARYING URGENCY

Urgency	Speech Based Warnings	Auditory Icons	Appropriate Situation
Informational (low)	Signal words that convey low urgency: "Notice", "Information"	Pouring water, steam, released air	Low petrol and oil levels, low tire pressure
Warning (moderate)	Signal words that convey medium urgency: Warning, Caution	Shutting car door, Roaringmotor sound, squeaking sound	Car door opened, speed limit exceeded, hand brake on
Critical (high)	Signal words that convey high urgency: Danger	Car horn, car crash, alarm siren	Blind spot overtaking, car drifting off road, collision possible

modalities if adequate.

Topics: perceived safety, driver, multimodal, output

Scenario: In-vehicle information system (IVIS) information needs to be delivered effectively while minimizing the interference with driving. Display modality has a significant impact on the performance of in-vehicle information systems. Visual, auditory and tactile displays all have their advantages and disadvantages [26]: Visual warnings can be inspected at the drivers own pace and can be viewed multiple times. However, they cause visual distraction from the driving task and can be overlooked. Auditory warnings can be picked up without causing visual distraction, but they require the drivers full attention when they are displayed. Tactile warnings are highly noticeable, not influenced by noise and have no visual demand, but they are limited to a few types of information, such as simple alerts. In order to maximize IVIS efficiency, designers have to choose carefully between the different modalities.

Solution: When choosing between auditory and visual presentation, table III offers decision guidelines based on current empirical research for a variety of cases. Some of these cases will probably benefit if combined with another display modality.

Examples: Figure 4 shows combined auditory and visual warnings. See [27] for a live demonstration.

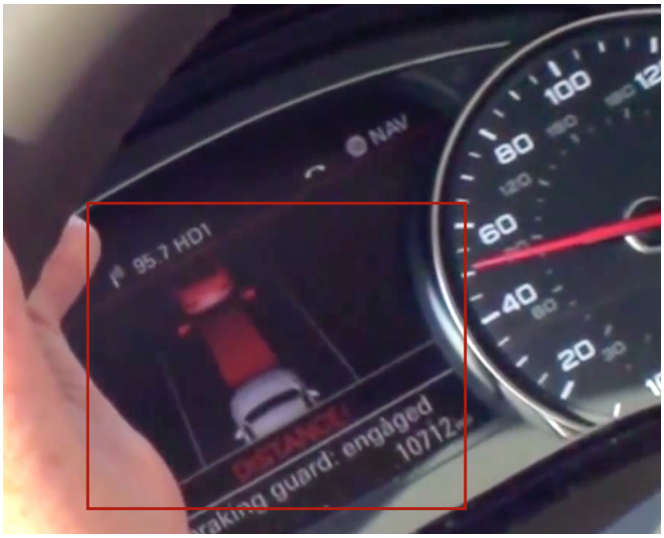


Figure 4. Audi A8 Distance Warning through a combination of auditory and visual warning displays.

E. Pattern 5: IVIS System Response Time

Intent: This Pattern addresses the role of system response time while operating in-vehicle information systems by touch interfaces or hardware keys and its influence on driver distraction and comfort.

Topics: Workload caused by distraction, joy of use, driver, keys, visual, haptic, input

Problem: While getting more and more complex, many modern in-vehicle information systems possess significant delays when using them because of the sheer amount of information that they have to process. The influence of system response time - the delay of a systems response after user input until it is ready to take new commands - has been discussed as a potential source of driver distraction and annoyance [28].

Scenario: Drivers use in-vehicle information systems for a wide variety of functions. While navigating their menus, the IVIS processes large amounts of information, which may lead to long and uncertain loading times.

Solution: Keep system response time below 250 ms. According to current design guidelines [19], control feedback should be given within 250ms after the input. A study by Utesch and Vollrath [29] showed that longer feedback delays (500 or 1000 ms) dont impair driving performance but caused significant annoyance in drivers. Keep system response times constant. It has also been shown in this study that delays that vary in their length distract the driver, while constant delays cause less off-road glances. It can be concluded that feedback delays should be kept constant so that waiting times for system response are predictable. For longer delays, use additional feedback modalities. According to guidelines of the European Commission [30], if system responses take longer than 250 ms, the system should inform the driver that it has recognized the input. If longer delays (500 ms and above) are inevitable, Utesch and Vollrath [29] recommend using acoustic or tactile feedback to indicate system readiness, as this will reduce off-road glances.

Examples:

1. Demonstration of a 2015 Audi MMI System, showing constant and short system response times [31].

2. Demonstration of a BMW 5 Series iDrive, showing long but constant delays [32].

3. Demonstration of an Apple CarPlay IVIS in the Ferrari FF showing long and variable delays. This might cause distraction and annoyance [33].

 ISBN: 978-1-61208-465-7

TABLE III
RECOMMENDED WARNINGS FOR COMMON SITUATIONS OF VARYING URGENCY

Case	Primary Modality	Reason	Combine with...
High priority messages	Auditory[34][35]	Visual warnings alone are likely to be overlooked	Tactile [36] for decreased reaction times
Complex secondary task	Auditory[37][38][39]	Further distraction due to increased glance duration	Visual[40] for reduced reaction times and less navigation errors
Driving task is highly demanding, e.g., high driving speed	Auditory[37][38][39]	Divided visual attention poses a security risk	
Displaying instructions, commands, warnings or alarms	Auditory [41]	Speech is more suitable for this information type	Tactile [42]
Auditory message cannot be kept short and precise	Visual [43]	Auditory messages that are too long cause severe distraction	
Driver performs auditory tasks	Visual [25]	Auditory perception is partially or completely blocked	Tactile [44] for reduced lane deviations and annoyance, increased pleasantness

F. Pattern 6: In-Vehicle Display Icon Size

Intent: this pattern addresses recommended IVIS icon sizes.

Topics: Joy of use, driver, icons, visual

Problem: IVIS displays transport various informations, some of which require quick and accurate recognition. However, as in-vehicle displays have to convey more and more information, available space on in-vehicle displays becomes sparse.

Scenario: Icons are a way of presenting information in a spatially condensed, yet clearly understandable way. When relying on icons, the driver needs to be able to quickly grasp and process information, which in turn requires that icons can be easily recognized.

Solution: According to Zwaga [45], icons perform better than text displays only if they are well designed. According to FHWA guidelines [46], choosing the adequate size for an icon can be determined with the following set of formulæ. See Figure 5 for an illustration of visual angle, distance and symbol height (where Symbol Height = the height of the symbol; Distance = distance from viewers eyepoint to the display; Visual Angle = angle in degrees. Height and Distance use the same unit of measure).

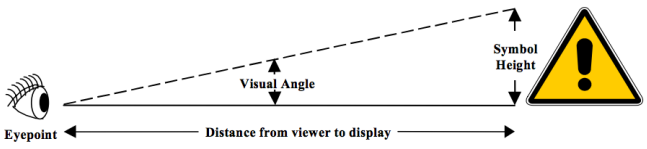


Figure 5. Relationship Between Viewing Distance, Symbol Height and Visual Angle.

1. If viewer distance and Symbol Height are known, the following formulæwill calculate the distance.

$$arctan(\frac{Symbolheight}{Distance}) \quad (1)$$

or

$$\frac{3438\frac{Height}{Distance}}{60} \quad (2)$$

2. If distance and visual angle are known.

$$Distance[tan(VisualAngle)] \quad (3)$$

3. If visual angle and symbol height are known, the following formulæwill calculate the distance.

$$\frac{Symbolheight}{tan(VisualAngle)} \quad (4)$$

Examples: See Figure 6.

Figure 6. Audi A4 2008 Dashboard Icons, taken from the users manual. [47]

G. Pattern 7: Visual Display Colour Choices

Intent: this pattern is about choosing adequate colours for visual displays.

Topics: Joy of use, driver, colors, visual

Problem: IVIS displays transport various informations, some of which require quick and accurate recognition. However, as in-vehicle displays have to convey more and more information, they still need to be processed quickly.

Scenario: IVIS displays have to display information in a clear and efficient way. One way to achieve this is picking adequate colors for displays, so that reading and recognizing symbols can be accomplished without delay.

Solution: According to NHTSA guidelines, visual display colors should comply to a number of standards.

- Avoid using red/green and blue/yellow combinations so that color blind drivers can process the display easily.
- According to a survey conducted by Lee and Park [48], senior people prefer combinations with distinc-

tive brightness contrasts between foreground and background color because of their better legibility.
- Displays that are too colorful distract the driver in various ways. Excluding black and white, a maximum of five different colours should be used.
- Use different colours for different priorities, e.g., red for critical alerts, amber for warnings, white for information.

Visual displays are easier to process if high color contrasts are used. A driving simulation study showed that inefficiently designed car displays strongly increase reaction times in driving tasks. They also increase reading errors [49]. Table IV shows color contrasts that guarantee high legibility.

TABLE IV
RECOMMENDED COLOR CONTRASTS FOR IVIS DISPLAYS

Black/yellow	Black/yellow
Black/white	Black/white
Black/orange	Black/orange
Blue/white	Blue/white
Green/white	Green/white
Red/white	Black/yellow

Examples: See Figure 7. This dashboard relies on white-on-black and orange-on-black contrasts which are highly visible. Orange is the only color besides black and white.

Figure 7. Dashboard with color contrasts that are highly visible.

H. Pattern 8: Physical Buttons Versus Touch Screen Interfaces

Intent: this pattern addresses the question whether touch screens or physical buttons should be used.

Topics: workload caused by distraction, driver, touch screen, visual, haptic, input

Problem: Current touch-screen devices provide no tactile feedback concerning control orientation, location, separation from one another. While driving, they can not be operated with eyes on the road, which in turn leads to long off-road glances. NHSTA guidelines [19] suggest that touch interfaces should not be operated while driving. On the other hand, touch screen devices provide much more flexibility,

Figure 8. BMW iDrive screen, showing blue-on-white contrasts with an orange highlight.

which is needed to operate modern, feature-rich in-vehicle information systems.

Scenario: Drivers use in-vehicle information systems for a wide variety of functions. Ways to navigate through the increasing number of functions are getting more and more complex. Touch screen interfaces are getting more and more popular, but navigating them while driving is highly distracting.

Solution:

1) While driving, limit the amount of time spent to interact with touch devices. NHTSA recommends a maximum of six touches for every 12 seconds period [50]. Physical buttons do not require such strict regulations as their functionality is limited and they are not as visually distracting. Thus, functions that must be available to the driver while the car is moving should be represented by physical buttons or clearly identifiable, big touch buttons. Recommended limitations are as follows
   - For touch devices **without** haptic feedback, limit touch screen interactions to six touches for every 12 seconds.
   - For touch devices **with** haptic feedback, limit touch screen interactions only to certain functions.
   - No restrictions apply to physical buttons while driving.
   - No restrictions apply while standing.
2) Equip touch devices with haptic feedback. According to Harrison and Hudson [51], touch screens lead to a high number of off-road searching glances and require long periods of operation time. They also found that this could be mitigated by provide touch screens with haptic feedback, which is confirmed by other studies [52]. Studies suggest that this kind of feedback greatly increases performance and reduces operation time. If haptic feedback is used, touch devices still should to be limited to the functionality provided by traditional physical buttons.
3) Also, consider alternative input methods that dont require visual attention (e.g., voice interaction).

Examples: See Figure 9.

 ISBN: 978-1-61208-465-7

Figure 9. VW Passat dashboard which combines few physical buttons with a well-readable touch display.

III. CONCLUSION

In this paper, we presented a collection of patterns, which deals with recurring questions of automotive design as reported by designers working in that area. By relying on design guidelines as well as empirical research, the collection tries to bridge the gap between government regulations, scientific findings and industry needs. These patterns were intended to be of direct practical use for automotive designers. The pattern structure and length, which we described in earlier work [18], has been fit to stakeholder demands, resulting in patterns with an increased emphasis on brevity and conciseness. The car User Experience patterns proposed in this paper constitute a small part of a constantly growing collection of design knowledge. The speed of innovations, the complexity, and the range of functions of car interfaces is increasing constantly. In addition, even if there are more and more connections between single car interfaces, innovations do not necessarily occur in parallel. Thus, an equally dynamic approach to document best practices in design is required. This pattern collection shows how a pattern approach to car UX design can meet these demands. The pattern collection will continue to grow into a substantial body of car UX design knowledge, which covers at least three of the most important UX factors for driver space design [53].

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development and AUDIO MOBIL Elektronik GmbH is gratefully acknowledged (Christian Doppler Laboratory for ”Contextual Interfaces”).

REFERENCES

[1] A. Dearden and J. Finlay, “Pattern languages in hci: A critical review,” Human-Computer Interaction, 2006, pp. 49–102, Sheffield Hallam University. [retrieved: 02, 2016] URL: http://research.cs.vt.edu/ns/cs5724papers/dearden-patterns-hci09.pdf.

[2] M. Obrist, D. Wurhofer, E. Beck, A. Karahasanovic, and M. Tscheligi, “User experience (ux) patterns for audio-visual networked applications: Inspirations for design,” in Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, ser. NordiCHI ’10. New York, NY, USA: ACM, 2010, pp. 343–352. [retrieved: 02, 2016] URL: http://doi.acm.org/10.1145/1868914.1868955

[3] J. O. Borchers, J. C. Thomas, A. Sutcliffe, J. Coplien, and R. N. Griffiths, “Patterns: What’s in it for hci?” in Extended Abstracts of the CHI 2001 Conference on Human Factors in Computing Systems. ACM, 2001.

[4] M. Hassenzahl and N. Tractinsky, “User experience-a research agenda,” Behaviour & information technology, vol. 25, no. 2. Taylor & Francis, 2006, pp. 91–97.

[5] A. F. Blackwell and S. Fincher, “PUX: Patterns of User Experience,” Interactions, vol. 17, no. 2. New York, NY, USA: ACM, 2010, pp. 27–31.

[6] M. Tscheligi, “User experience design for vehicles,” in Christian doppler laboratory: contextual interfaces. Tutorial for conference AUI. ACM, 2012.

[7] M. Obrist, D. Wurhofer, E. Beck, and M. Tscheligi, “Cux patterns approach: Towards contextual user experience patterns,” in Proceedings of the 2nd International Conferences on Pervasive Patterns and Applications, PATTERNS, vol. 10. IARIA, 2010. [retrieved: 02, 2016] URL: http://www.thinkmind.org/index.php?view=article&articleid=patterns_2010_3_20_70079

[8] A. Krischkowsky, D. Wurhofer, N. Perterer, and M. Tscheligi, “Developing patterns step-by-step: A pattern generation guidance for hci researchers,” in PATTERNS 2013, The Fifth International Conferences on Pervasive Patterns and Applications. IARIA, 2013, pp. 66–72. [retrieved: 02, 2016] URL: http://www.thinkmind.org/index.php?view=article\&articleid=patterns_2013_3_30_70053

[9] D. Martin, T. Rodden, M. Rouncefield, I. Sommerville, and S. Viller, “Finding patterns in the fieldwork,” in Proceedings of the Seventh European Conference on Computer Supported Cooperative Work. Kluwer Academic Publishers, 2001, pp. 39–58.

[10] A. Crabtree, T. Hemmings, and T. Rodden, “Pattern-based support for interactive design in domestic settings,” in Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, ser. DIS ’02. New York, NY, USA: ACM, 2002, pp. 265–276. [retrieved: 02, 2016] URL: http://doi.acm.org/10.1145/778712.778749

[11] C. Alexander, S. Ishikawa, and M. Silverstein, A pattern language: towns, buildings, construction. Oxford University Press, 1977, vol. 2.

[12] K. Young, M. Regan, and M. Hammer, “Driver distraction: A review of the literature,” Distracted driving. Australasian College of Road Safety, 2007, pp. 379–405.

[13] M. A. Recarte and L. M. Nunes, “Mental workload while driving: effects on visual search, discrimination, and decision making.” Journal of experimental psychology: Applied, vol. 9, no. 2. American Psychological Association, 2003, p. 119.

[14] J. Engström, E. Johansson, and J. Östlund, “Effects of visual and cognitive load in real and simulated motorway driving,” Transportation Research Part F: Traffic Psychology and Behaviour, vol. 8, no. 2. Elsevier, 2005, pp. 97–120.

[15] R. Buchner, D. Wurhofer, A. Weiss, and M. Tscheligi, “Robots in time: How user experience in human-robot interaction changes over time,” in Social Robotics. Springer, 2013, pp. 138–147.

[16] R. Buchner, P. M. Kluckner, A. Weiss, and M. Tscheligi, “Assisting maintainers in the semiconductor factory: Iterative co-design of a mobile interface and a situated display,” in Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia. ACM, 2013, p. 46.

[17] G. Stollnberger, A. Weiss, and M. Tscheligi, “”the harder it gets” exploring the interdependency of input modalities and task complexity in human-robot collaboration,” in RO-MAN, 2013 IEEE. IEEE, 2013, pp. 264–269.

[18] A. G. Mirnig et al., “Generating patterns combining scientific and industry knowledge: An inclusive pattern approach,” in PATTERNS 2015, The Seventh International Conferences on Pervasive Patterns and Applications. IARIA, 2014, pp. 38–45.

[19] A. Stevens, A. Quimby, A. Board, T. Kersloot, and P. Burns, NHTSA Design Guidelines for Safety of In-Vehicle Information Systems. National Highway Traffic Safety Administration, 2002.

 ISBN: 978-1-61208-465-7

[20] G. E. Burnett, G. Lawson, R. Donkor, and Y. Kuriyagawa, "Menu hierarchies for in-vehicle user-interfaces: Modelling the depth vs. breadth trade-off," Displays, vol. 34, no. 4. Elsevier, 2013, pp. 241–249.

[21] M. Pfauth and J. Priest, "Person-computer interface using touch screen devices," Proceedings of the Human Factors Society 25th Annual Meeting, no. 1, 1981, pp. 500–504.

[22] H. Kim, S. Kwon, J. Heo, H. Lee, and M. K. Chung, "The effect of touch-key size on the usability of in-vehicle information systems and driving safety during simulated driving," Applied ergonomics, vol. 45, no. 3. Elsevier, 2014, pp. 379–388.

[23] J. P. Bliss and S. A. Acton, "Alarm mistrust in automobiles: how collision alarm reliability affects driving," Applied ergonomics, vol. 34, no. 6. Elsevier, 2003, pp. 499–509.

[24] A. Guillaume, L. Pellieux, V. Chastres, and C. Drake, "Judging the urgency of nonvocal auditory warning signals: perceptual and cognitive processes." Journal of experimental psychology: Applied, vol. 9, no. 3. American Psychological Association, 2003, p. 196.

[25] J. Edworthy and E. Hellier, Complex Nonverbal Auditory Signals and Speech Warnings. Lawrence Erlbaum Associates Publishers, 2006, pp. 199–220.

[26] Y. Cao and M. Theune, "The use of modality in in-vehicle information presentation: A brief overview," in Proceedings of the 2nd International Workshop on Multimodal Interfaces for Automotive Applications (MIAA), in conjunction with IUI. ACM, 2010, pp. 1–5.

[27] "Collision warning systems at the test track," 2012, [retrieved: 02, 2016] URL: https://www.youtube.com/watch?v=rYckJqp4XTc#t=45.

[28] P. Burns and T. Lansdown, "E-distraction: the challenges for safe and usable internet services in vehicles," in Internet Forum on the Safety Impact of Driver Distraction When Using In-Vehicle Technologies, 2000. [retrieved: 02, 2016] URL: http://www-nrd.nhtsa.dot.gov/departments/Human%20Factors/driver-distraction/PDF/29.PDF

[29] F. Utesch and M. Vollrath, "Do slow computersystems impair driving safety?" in European Conference on Human Centred Design for Intelligent Transport Systems, 2nd, 2010, Berlin, Germany, 2010.

[30] T. F. HMI, "European statement of principles on human machine interface for in-vehicle information and communication systems," Tech. rep., Commission of the European Communities, Tech. Rep., 1998.

[31] "2015 Audi MMI Infotainment Review in the 2015 Audi A3 Sedan," 2014, [retrieved: 02, 2016] URL: https://www.youtube.com/watch?v=xadb1vSXw7Q.

[32] "BMW 5 Series Connected-Drive (iDrive) Full Demo - including Head-Up Display," 2013, [retrieved: 02, 2016] URL: https://www.youtube.com/watch?v=Ulcgm-GN__M.

[33] "Apple CarPlay Demo," 2014, [retrieved: 02, 2016] URL: https://www.youtube.com/watch?v=TQz64rhefqY.

[34] C. Kaufmann, R. Risser, A. Geven, and R. Sefelin, "Effects of simultaneous multi-modal warnings and traffic information on driver behaviour," in Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems. Humanist Publications, 2008, pp. 33–42.

[35] B. Seppelt and C. Wickens, "In-vehicle tasks: Effects of modality, driving relevance, and redundancy," Tech. Rep., 2003. [retrieved: 02, 2016] URL: http://www.aviation.illinois.edu/avimain/papers/research/pub_pdfs/techreports/03-16.pdf

[36] C. Ho, H. Z. Tan, and C. Spence, "Using spatial vibrotactile cues to direct visual attention in driving scenes," Transportation Research Part F: Traffic Psychology and Behaviour, vol. 8, no. 6. Elsevier, 2005, pp. 397–412.

[37] J. B. Hurwitz and D. J. Wheatley, "Using driver performance measures to estimate workload," in Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 46, no. 22. Sage Publications, 2002, pp. 1804–1808.

[38] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and D. J. Ramsey, "The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data," Tech. Rep., 2006.

[39] R. Srinivasan and P. P. Jovanis, "Effect of selected in-vehicle route guidance systems on driver reaction times," Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 39, no. 2. Sage Publications, 1997, pp. 200–215.

[40] Y.-C. Liu, "Comparative study of the effects of auditory, visual and multimodality displays on drivers' performance in advanced traveller information systems," Ergonomics, vol. 44, no. 4. Taylor & Francis, 2001, pp. 425–442.

[41] K. Stanney, S. Samman, L. Reeves, K. Hale, W. Buff, C. Bowers, B. Goldiez, D. Nicholson, and S. Lackey, "A paradigm shift in interactive computing: Deriving multimodal design principles from behavioral and neurological foundations," International Journal of Human-Computer Interaction, vol. 17, no. 2. Taylor & Francis, 2004, pp. 229–257.

[42] J. B. Van Erp and H. A. Van Veen, "Vibrotactile in-vehicle navigation system," Transportation Research Part F: Traffic Psychology and Behaviour, vol. 7, no. 4. Elsevier, 2004, pp. 247–256.

[43] Y. Cao, S. Castronovo, A. Mahr, and C. Müller, "On timing and modality choice with local danger warnings for drivers," in Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications. ACM, 2009, pp. 75–78.

[44] D. Kern, P. Marshall, E. Hornecker, Y. Rogers, and A. Schmidt, "Enhancing navigation information with tactile output embedded into the steering wheel," in Pervasive Computing. Springer, 2009, pp. 42–58.

[45] H. Zwaga and T. Boersema, "Evaluation of a set of graphic symbols," Applied Ergonomics, vol. 14, no. 1. Elsevier, 1983, pp. 43–54.

[46] J. L. Campbell, J. Richman, C. Carney, and J. Lee, "In-vehicle display icons and other information elements. volume i: Guidelines, technical report," Tech. Rep., 2004.

[47] Audi A4 2008 Owner's Manual, http://www.alkivar.com/AUDI/Audi_A4_2008_Owner_s_Manual.pdf [retrieved: 02, 2016].

[48] M. Lee and J. Park, "Senior users color cognition and color sensitivity features in visual information on web-interface," in Universal Access in Human-Computer Interaction. Aging and Assistive Environments. Springer, 2014, pp. 129–137.

[49] E. S. Wilschut, G. Rinkenauer, K. A. Brookhuis, and M. Falkenstein, "performance in a lane-change task is vulnerable to increased secondary task complexity." TNO Publications, p. 5.

[50] Authors unknown, "Visual-manual nhtsa driver distraction guidelines for in-vehicle electronic devices," Washington, DC: National Highway Traffic Safety Administration (NHTSA), Department of Transportation (DOT), 2012.

[51] C. Harrison and S. E. Hudson, "Providing dynamically changeable physical buttons on a visual display," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009, pp. 299–308.

[52] H. Richter, R. Ecker, C. Deisler, and A. Butz, "Haptouch and the 2+ 1 state model: potentials of haptic feedback on touch based in-vehicle information systems," in Proceedings of the 2nd international conference on automotive user interfaces and interactive vehicular applications. ACM, 2010, pp. 72–79.

[53] A. G. Mirnig et al., "User experience patterns from scientific and industry knowledge: An inclusive pattern approach," International Journal On Advances in Life Sciences, vol. 7, no. 3 and 4. IARIA, 2015, pp. 200–215. [retrieved: 02, 2016] URL: https://www.thinkmind.org/index.php?view=article&articleid=patterns_2015_2_30_70011.

 ISBN: 978-1-61208-465-7

Knowledge Extraction from German Automotive Software Requirements using NLP-Techniques and a Grammar-based Pattern Detection

Mathias Schraps
Software Development
Audi Electronics Venture GmbH
85080 Gaimersheim, Germany
e-mail: mathias.schraps@audi.de

Alexander Bosler
Fakultät für Informatik
Technische Universität München
85748 Garching, Germany
e-mail: alexander.bosler@tum.de

***Abstract*—In Requirements Engineering, natural language is often used to specify the system under development with textual requirements. Especially in the automotive industry it is used to specify the processing of signals and parameters, as well as the behavior of sensors or actuators. During the creation of a specification first executable software models were developed, which have to be implemented according to the corresponding requirements. Due to the asynchronous development of specifications and software models, inconsistencies and defects may occur. To overcome this issue, we developed an approach using Natural Language Processing (NLP) techniques and a formal grammar to match semantic patterns in order to extract knowledge of requirements and represent it in an ontology. This approach will be introduced based on an example of an automotive software requirement.**

Keywords-Natural Language Processing; Requirements; Ontology; Knowledge Representation; Semantic Annotated Grammar; Knowledge Extraction; Pattern Detection.

I. INTRODUCTION

The development of embedded software in the automotive domain is a challenging task. First, requirements regarding architecture, data communication and behavior of the system under development have to be elicited and documented. This involves several stakeholders like electronic engineers, software developers, architects and other domain engineers.

During the phase of Requirements Elicitation, first executable software models are created, so-called Rapid Prototyping [1]. Therein, a partial amount of these requirements are implemented so far. During the progress of the project more and more requirements will be specified and have to be covered by the model and later by the implementation. This procedure implies a high linkage between two project phases: Requirements Elicitation and Modelling. If the artifacts of these both phases were not updated permanently by the involved developers and stakeholders, defects, errors or inconsistencies may occur and could be propagated across the entire development process. The later these issues are detected and solved, the more cost-intensive is their removal [2]. Therefore, the artifacts created in early phases of a software development project have to be consistent as much as possible.

In the automotive industry, a software requirement specification consists of more than only one document. Even though these documents come from many authors with different background and interests, they share one thing in common to specify their requirements: a natural language. Unfortunately, these natural language requirements can be incomplete, ambiguous and error-prone [3]–[5], especially in early phases of development.

This paper presents a method of extracting knowledge from textual requirements formulated in German natural language using Natural Language Processing (NLP) techniques. According to the detected patterns within a single requirement, this knowledge will be transferred into a requirements ontology in order to be able to check consistency between several requirements and for reuse purposes.

The structure of this paper is as follows. Section II gives an overview about the annotation of textual requirements using NLP-techniques. This is illustrated on a sample requirement given at the end of this section. In Section III, the pattern detection and the mapping of the sample requirement into an ontology will be introduced. Section IV provides a conclusion and outlines possible future work. The sample requirement on which all illustrations in the following sections are based, is formulated as follows: *"Wenn die Klemme 50 eingeschalten ist und s_MTrig=p_MAn, dann ist der Motor zu starten (s_MStart=1)."* An English translation of this requirement would be: *"If clamp 50 is switched on and s_MTrig=p_Man, then the engine must be started (s_MStart=1)."*

II. REQUIREMENTS ANNOTATION USING NLP

The NLP annotation process is the concatenation of different NLP-tools in a pipeline-style manner, where each tool provides additional information about the processed requirement (cf. Fig 1). This process was inspired by Arora et al [6] and adds a possibility to map the knowledge within the requirement about the system under development into an ontology.

If needed, tools are able to query the underlying ontology, where the knowledge is stored. This knowledge is extracted from the requirements in later steps. At the beginning of the requirement acquisition, it is possible to initialize the underlying ontology with a priori knowledge about predefined signals, constants and parameters. Furthermore, it is possible to add an existing taxonomy, which in this case is extracted from an existing High Level

 ISBN: 978-1-61208-465-7

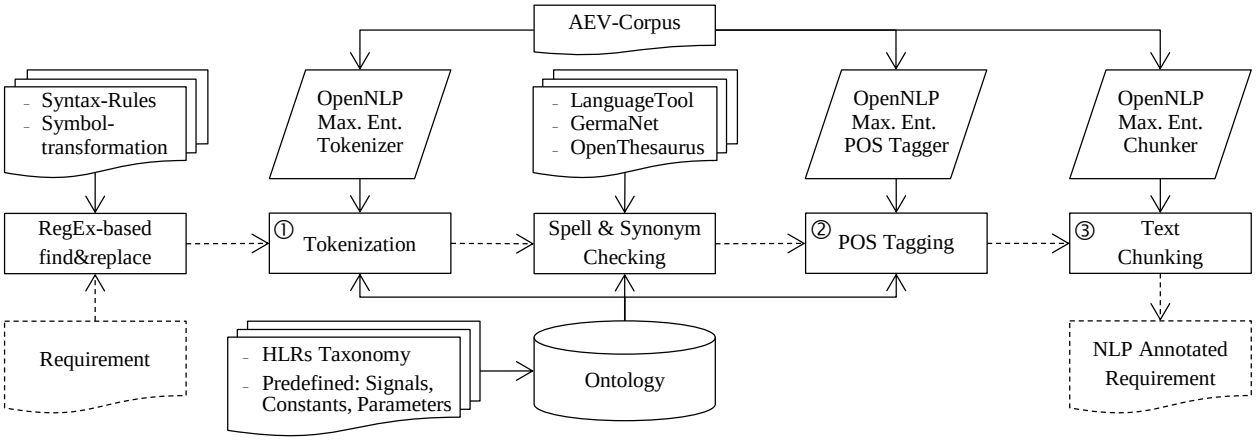


Figure 1. NLP based knowledge extraction process (part 1)

Requirements (HLRs) Specification [7]. To improve the results of the OpenNLP-tools [8], which are based on machine learning models the AEV-Corpus (internal Audi Electronics Venture Requirement based Corpus) was introduced (cf. ①, ②, ③ in Fig. 1). At the moment the AEV-Corpus consists of about 450 tokenized, POS-tagged and chunked automotive software requirements. In the following paragraphs the annotation of the requirement, which results in an "NLP Annotated Requirement" (cf. Fig. 1), is described in more detail.

The "RegEx-based search & replace" activity provides the possibility to define search and replace pairs, which are enforced at the start of the NLP-process in order to fix syntactical problems like missing or multiple whitespace characters and to standardize symbol usage.

During the "Tokenization" (cf. ① in Fig. 1), the OpenNLP Tokenizer splits the requirements according to a Maximum Entropy Model trained on the AEV-Corpus. The model achieves results similar to the default models provided by OpenNLP when tokenizing the natural language parts of a given requirement. The main advantage achieved by introducing the AEV-Corpus trained model, is the tokenization of very formal requirements: concepts like formal equations of signals and constants, C-Structs or Arrays are not part of the OpenNLP default models and thus, tokenization tends to fail. To improve the tokenization results, generated by the OpenNLP Tokenizer, the Named Entities (contained in the underlying ontology) are used to verify their correct tokenization of the currently processed requirement. The tokens, which are generated for the sample requirement (cf. Section I), are shown at ① in Fig. 2.

The "Spell & Synonym Checking" activity uses the Java version of the JLanguageTool [9] to detect misspelled tokens and provides a list of suggestions for each of them. Furthermore, a set of synonyms for each token is created using GermaNet [10] and OpenThesaurus [11]. The resulting set is used to query the underlying ontology to check if one or more of them are already included. After this, the synonyms provided and found at least once in the ontology, are added to their corresponding token.

In the "POS Tagging" activity (cf. ② in Fig. 1), each token (word, punctuation character and mathematical symbol) of the requirement is tagged with its corresponding Part Of Speech (POS) Tag. Since formal definitions are very common in automotive software requirements and common Tagsets only provide Part Of Speech Tags for natural language, we extended the STTS Tagset [12] by *$S* to tag mathematical symbols (=,<,>,≥,≤,+,-,…) and *$L* to tag listing symbols (:,->) to address this issue. The assignment of the POS-Tags to each token is done using the OpenNLP POS-Tagger based on a Maximum Entropy Model, which is trained on the AEV-Corpus. To improve the POS-Tagging results generated by the OpenNLP POS-Tagger, the Named Entities and Concepts (contained in the underlying ontology) are used to verify their correct tagging in the currently processed requirement. The POS-Tags for all tokens, generated during the "Tokenization" of the sample requirement (cf. Section I), are shown at ② in Fig. 2.

The "Text Chunking" activity (cf. ③ in Fig. 1) uses the OpenNLP Chunker to chunk the requirement according to a Maximum Entropy Model, which is trained on the AEV-Corpus. Each chunk consists of one or more tokens, tagged with the corresponding Chunk-Tag and can be considered as a "part of interest" of the processed requirement. The chunks, generated for the sample requirement (cf. Section I), are shown at ③ in Fig. 2. The meaning of the used Chunk-

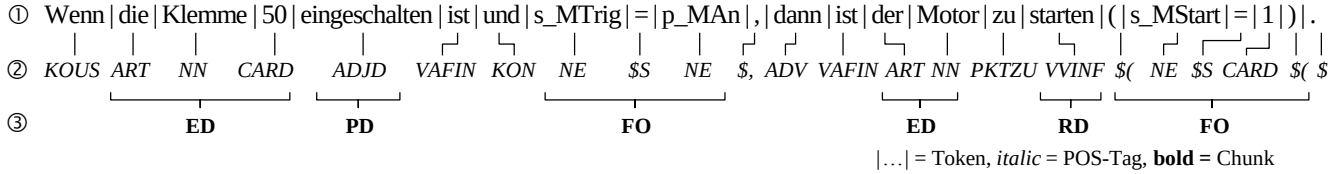


Figure 2. Sample requirement annotated by the NLP-process according to Fig. 1

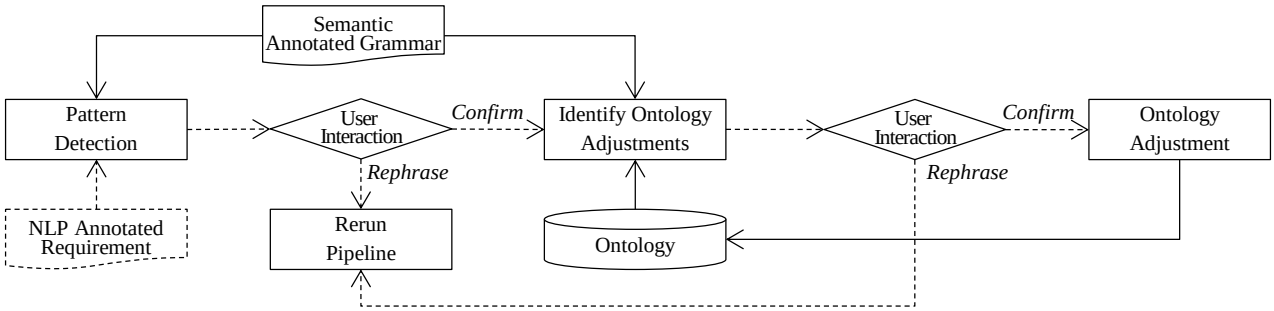


Figure 3. NLP-based knowledge extraction process (part 2)

Tags is as follows: Entity Definition (ED), State/Property Definition (PD), Action/Relation Definition (RD), Formula (FO).

III. PATTERN DETECTION AND ONTOLOGY ADJUSTMENT

Based on the NLP Annotated Requirement (cf. Fig. 2) and patterns in the form of a semantic annotated grammar (cf. Fig. 4), which is inspired by [13], the "Pattern Detection" and the "Ontology Adjustment" (which is split into "Identify Ontology Adjustments" and "Ontology Adjustment" to allow User Interaction), extracts the knowledge contained in the processed requirement and stores it into the underlying ontology (cf. Fig. 3). A more detailed view on the knowledge extraction process is given in the following paragraphs.

During the "Pattern Detection", the semantic aspect of the semantic annotated grammar is ignored since it does not provide any additional information for this activity. The first step in the Pattern Detection is the aggregation of tokens, POS-Tags and chunks to a list, which is referred to as "word". The "word" only contains elements, which are available in the NLP Annotated Requirement and also terminals of the grammar. According to this rule and the semantic annotated grammar, the following elements in the annotated sample requirement (cf. Fig. 2.) would be ignored: |ist|(VAFIN), |,|($,), |ist|(VAFIN), |zu|(PKTZU), |.|($.) and the "word" would be: "Wenn ED PD KON FO dann ED RD FO". To verify if the "word" can be expressed in the formal language defined by the semantic annotated grammar, a finite state machine based recognizer is being used. For the sample requirement the recognizer would tell us that our "word" can be expressed using the If-Then-Pattern of the semantic annotated grammar (cf. <If-Then-Pattern> in Fig. 4).

At the end of the "Pattern Detection" activity, the user is informed about the results of the "Spell & Synonym Checking" activity (cf. Section II) and whether a valid requirement pattern was found in the processed requirement or not. This allows the user to rephrase the requirement or to start the "Ontology Adjustment" for the processed requirement.

The "Identify Ontology Adjustments" activity performs two major tasks. At first, it creates a temporary knowledge representation for the processed requirement, based on the semantic annotated grammar (cf. Fig. 4, Fig. 5 and Fig. 6). Secondly, it checks whether the insertion of the temporary knowledge representation can be performed to the

```
<ED-Chunk> ::= <optional-Article> <Entity>
<optional-Article> ::= " " | ART
<Entity> ::=
  NN          => createIndividual(valueOf(NN),valueOf(NN))    |
  NN CARD     => createIndividual(valueOf(NN),
                 concat(valueOf(NN)," ",valueOf(CARD)))

<If-Then-Pattern> ::=
  Wenn <Condition> dann <Action> => createIfThenReq(
                                    individualOf(<Condition>),
                                    individualOf(<Action>))
<Condition> ::=
  FO                    => createCondition(individualOf(FO))  |
  ED PD KON <Condition> => createCondition(individualOf(ED),
                           individualOf(PD),
                           addSubCondition(<Condition>))
<Action> ::=
  ED RD      => createAction(individualOf(ED),individualOf(RD)) |
  ED RD FO => createAction(individualOf(ED),individualOf(RD),
              individualOf(FO))
```

Figure 4. Simplified Semantic Annotated Grammar for Pattern Detection and Ontology Adjustment in BNF-Style

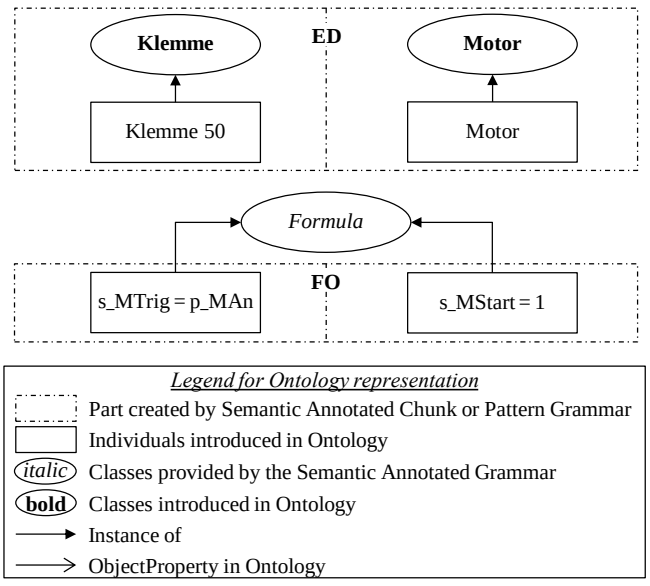


Figure 5. Ontology representation of ED and FO chunks

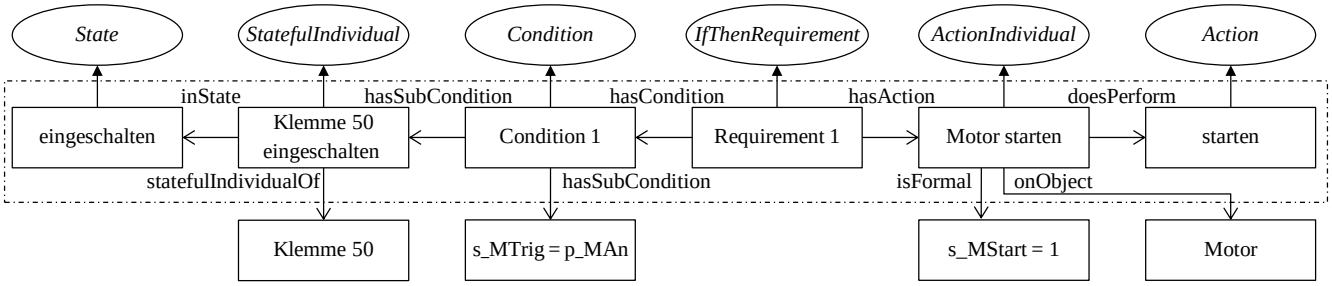


Figure 6. Knowledge representation of the sample requirement

underlying ontology without the violation of existent axioms or not. To create the temporary knowledge representation for the entire processed requirement, a knowledge representation is created for each chunk according to the semantic annotations of its Chunk-Grammar (cf. <ED-Chunk> in Fig. 4). The temporary knowledge representation, created for the ED and FO chunks of the sample requirement are different, since FO is an instance of a generic *Formula* Class and ED, as defined in the Semantic Annotated Grammar (cf. Fig. 4), creates both, the Class itself and an Individual as an instance of the Class (cf. Fig. 5). After the temporary knowledge representation for the chunks has been build, the contained individuals are connected and enhanced with new knowledge according to the semantic annotations of the Pattern-Grammar (cf. <If-Then-Pattern> in Fig. 4), determined during the "Pattern Detection" activity. The temporary knowledge representation, which is created for the sample requirement, is given in Fig. 6. Otherwise, if there is no matching pattern found during the "Pattern Detection" activity and the user confirmed the formulated requirement in the previous activity, the temporary knowledge representation of each single chunk will be linked to a temporary DefaultRequirementIndividual, which represents a lean requirement structure within the ontology. Finally, the "Identify Ontology Adjustments" activity checks whether it would be possible or not to insert the temporary knowledge representation into the underlying ontology without violating existent axioms of previous inserted requirements, which may lead to an inconsistent ontology. During this step, the underlying ontology is not updated or modified but queried to detect axiom violations.

If the "Identify Ontology Adjustments" activity determines, that it is not possible to insert the temporary knowledge representation into the ontology without violating existent axioms, the user is asked whether he/she wants to rephrase or refine the requirement or continue with the next process step according to Fig. 3 by confirming the detected issue.

"Ontology Adjustment" is the final step in the knowledge extraction process. It updates the underlying ontology according to the temporary knowledge representation of the sample requirement (cf. Fig. 6), which was created by the "Identify Ontology Adjustments" activity. If the ontology can't be updated with the temporary knowledge acquired during the previous activity without violating existent axioms (as determined by the "Identify Ontology Adjustments" activity) and the user confirmed the issue after the "Identify Ontology Adjustments" activity, every element of the temporary knowledge representation, that violates an existing axiom is removed from the remaining temporary knowledge representation and the therein remaining elements are inserted into the ontology and marked to be partial.

IV. Conclusion and Outlook

In this paper, we presented an approach to annotate automotive software requirements formulated in German natural language using NLP-techniques. The pattern detection matched predefined patterns and transforms the tagged and chunked parts of a requirement according to its semantic into a requirements ontology in order to represent the knowledge of the entire requirement.

In our next work, we will support a mapping of the developed requirements ontology to block-elements of a software model created with MATLAB Simulink. This will provide the ability to trace the semantic of requirements between the phases Requirements Elicitation and Modelling within the embedded software development process. In further stages, this approach will be evaluated by a prototypical tool with a graphical user interface to let the user write requirements and check the consistency to the corresponding software model.

References

[1] J. Schäuffele and T. Zurawka, Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen, 5th ed. Wiesbaden: Springer Fachmedien Wiesbaden, 2013.

[2] S. McConnell, Code complete: A practical handbook of software construction, 2nd ed. Redmond, Washington: Microsoft Press, 2004.

[3] E. Hull, K. Jackson, and J. Dick, Requirements Engineering, 3rd ed. London: Springer Verlag London Limited, 2011.

[4] C. Rupp and SOPHIST GROUP, Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis, 5th ed. München, Wien: Hanser, 2009.

[5] K. Pohl, Requirements Engineering: Grundlagen, Prinzipien, Techniken, 2nd ed. Heidelberg: dpunkt-Verlag, 2008.

[6] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, and R. Gnaga, "RUBRIC: A Flexible Tool for Automated Checking of Conformance to Requirement Boilerplates," Proceedings of

the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013), New York, NY: Association for Computing Machinery, 2013, pp. 599–602, doi:10.1145/2491411.2494591.

[7] M. Ringsquandl and M. Schraps, “Taxonomy Extraction from Automotive Natural Language Requirements Using Unsupervised Learning,” International Journal on Natural Language Computing (IJNLC), vol. 3, no. 4, pp. 41–51, 2014.

[8] Apache Software Foundation, “Apache OpenNLP,” [Online]. Available: https://opennlp.apache.org/. Accessed: Nov. 11, 2015.

[9] D. Naber, “LanguageTool,” [Online]. Available: https://languagetool.org/. Accessed: Nov. 11, 2015.

[10] University of Tübingen, Tübingen, Germany, “GermaNet - An Introduction,” [Online]. Available: http://www.sfs.uni-tuebingen.de/GermaNet/. Accessed: Nov. 11, 2015.

[11] D. Naber, “openthesaurus.de,” [Online]. Available: https://www.openthesaurus.de/. Accessed: Nov. 11, 2015.

[12] Universität Stuttgart, Institute for Natural Language Processing, Stuttgart, Germany, “STTS Tag Table (1995/1999),” [Online]. Available: http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html. Accessed: Nov. 16, 2015.

[13] M. Schraps and M. Peters, “Semantic Annotation of a Formal Grammar by SemanticPatterns,” 2014 IEEE 4th International Workshop on Requirements Patterns (RePa), 2014, pp. 9–16, doi:10.1109/RePa.2014.6894838.

 ISBN: 978-1-61208-465-7

Indoor Localization by Map Matching Using One Image of Information Board

Kento Tonosaki, Yoshihiro Sugaya, Tomo Miyazaki, Shinichiro Omachi
Department of Communications Engineering, Graduate School of Engineering
Tohoku University
Sendai, Japan
e-mail: {sea, sugaya, tomo, machi}@iic.ecei.tohoku.ac.jp

***Abstract*—Indoor navigation systems have not become common like an outdoor navigation system, because we cannot correctly receive the signals from Global Positioning System (GPS) satellites in indoor environments, and indoor map databases are unavailable in most locations. Existing indoor localization methods and navigation systems have problems such as management and deployment cost, and limitation of available places. In this paper, we present a novel method of indoor navigation using an information board and several functions of a smartphone. Our framework does not require large-scale preparations and expenses for owners of buildings unlike existing methods, and is available wherever an information board exists. This method comprises image analysis and map matching. The former is to analyze the picture of information board taken by a smartphone and estimate the passageway region from the image. The latter gives us real-time localization in the map by using inertial sensors in the smartphone after the input of current places by the user at two different positions.**

Keywords- indoor positioning; particle filter; map matching.

I. INTRODUCTION

There are two keys of indoor navigation; positional information and a map. GPS is now contributing outdoor localization, and in combination with an outdoor map database, such as Google Maps, it provides us a good outdoor navigation system. Many modern mobile devices such as smartphones have a built-in GPS receiver. Therefore, we can easily use applications for outdoor navigation.

On the other hand, localization using GPS is inaccurate in indoor environment. GPS positioning is performed by using the signals from some satellites, but the signals cannot arrive at the receiver in indoor environment. Indoor localization is a hot topic now, and various systems are developed; Wi-Fi, radio-frequency identification (RFID), ultrasound, camera image, inertial measurement unit (IMU), and so on. However, these methods do not have decisive superiority that can be considered as a de facto standard because they require some infrastructure and it limits available places. Another approach is pedestrian dead reckoning (PDR) making use of several inertial sensors embedded in smartphones as well as GPS receiver. Owing to no expenses except for smartphone, we can estimate relative position at a low price. The drift of gyroscope is the main cause of error in PDR estimation, then by the combination with other methods, accuracy of localization can be improved [1][2].

Moreover, the absence of indoor map database makes construction of indoor navigation system more difficult. Although "Google Maps" provides several indoor maps, there are not many available places. In some place, e.g., Tokyo Station and Narita Airport in Japan, a special application for navigation of each place is offered. However, it requires for users to install the special application on their smartphone to use the navigation.

In this paper, we propose a novel framework of indoor navigation system using an information board with a floor map (Figure 1) and several functions in smartphone. Since information boards exist in many buildings, we can easily obtain a map by taking a picture of the information board with smartphone's camera. In addition, we can estimate the current position without infrastructure by using PDR with inertial sensors.

The usage scenarios of our approach are as follows. First, a user takes a photo of the map on the information board with his/her smartphone; note that the information boards are usually installed at the entrance of malls or the side of elevators. After the picture is analyzed, the user taps the point on the map displayed on the screen corresponding to his current place. Then, the user walks a little, and taps the screen where the point corresponds to the current position again. By using the information obtained by the two taps, the reduced scale and orientation of the map could be estimated, and inertial sensors enable to estimate current position of the user.

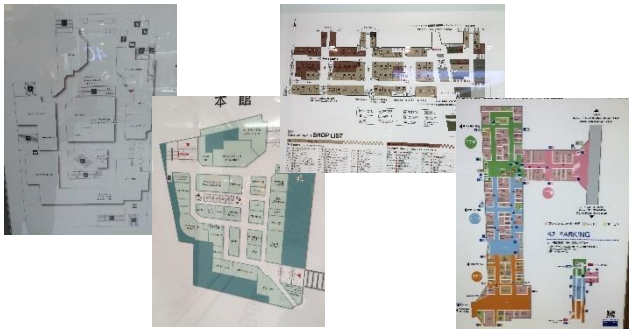


Figure 1. Examples of information board

 ISBN: 978-1-61208-465-7

II. PASSAGEWAY REGION ESTIMATION

We considered that passageway regions are necessary at least to implement a demonstration of the indoor navigation. We develop a method to estimate passageway region from an image of information board, and we show the method in this section.

The estimation is conducted using two conditions that many of information boards meet. These are based on our survey of 104 information boards throughout Japan, and about 92% of them fulfill two conditions: (1) the color of passageways is different from any shops' colors, (2) only one color is used for passageways in a map. The proposed method consists of labeling and passageway label estimation.

A. Labeling

According to the conditions, it is considered that segmentation of a map picture based on its color information is useful for the estimation of passageway regions.

At first, we create a segmentation image with mean shift [3] and an edge image with the Canny method [4] from the original image. Then, we scan the segmentation image from upper left. When we find unlabeled pixel (i,j), we also search the pixel (k,l) satisfying (1) in 4-neighborhood pixels around (i,j) except for edge pixel, and assign the same label. Although we can do labeling only using the segmentation image, which is obtained by mean shift, the labeling results are not sufficient. Therefore, we use the edge image together with segmentation image to improve the accuracy of labeling. If the area size of label is smaller than the threshold τ_l, we give them no label.

$$\begin{cases} |R(i,j) - R(k,l)| \le \tau_d \\ |G(i,j) - G(k,l)| \le \tau_d \\ |B(i,j) - B(k,l)| \le \tau_d \end{cases}. \quad (1)$$

$R(i,j)$, $G(i,j)$ and $B(i,j)$ are red, green and blue values of the pixel (i,j), respectively. The symbol τ_d is a threshold.

B. Passageway Region Estimation

After labeling, we estimate the labels of passageway region. A set of passageway labels often contains larger areas than shop ones, and according to the above-mentioned conditions, only one color different from every shop labels' colors is used for passageway labels. We estimate the passageway region by using these conditions.

At first, we decide the first passageway labels (FPL), which is the most likely passageway label, and we estimate it along the flow chart (Figure 2). We assumed that L_1 or L_2 or both are the passageway labels, where L_1 is the first largest label and L_2 is the second. We compare the label sizes because we suppose that the passageway region would be much bigger than any shop's regions. If $L_1 > \alpha L_2$, FPL is L_1. Otherwise, we compare the label colors in consideration that the passageway region in the map are divided by some lines such as arrows. If the Euclidian distance between the color

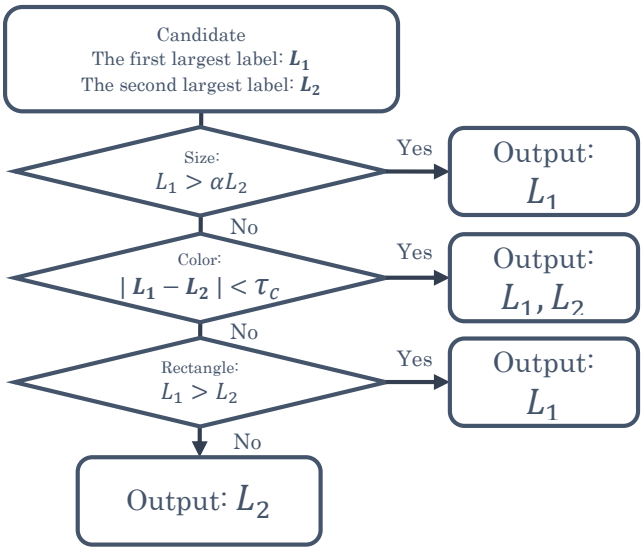


Figure 2. Flow chart of the method to decide the first passageway labels

averages of L_1 and L_2 in RGB color space is lower than threshold τ_c, FPL is the combination of L_1 and L_2. Otherwise, we compare the sizes of bounding boxes of them, and we consider that the larger one is passageway region.

After FPL is decided, we search other passageway labels using color information of FPL. We compare the average color of FPL and other labels, and when Euclidian distance between them in RGB color space is lower than threshold τ_c, they are determined as the passageway label.

III. PARTICLE FILTER AND MAP MATCHING

In this method, the scale and the orientation of a map are unknown because the map is a picture of information board. To estimate them to a certain extent, the user taps the screen to show where they are in the map at two different positions. In addition, a map of information board is inaccurate, owing to noises and distortions resulting from light reflection and the direction of taking the picture. We employ map matching with particle filter (PF), which can cope with these problems flexibly.

PF is a method to estimate a state in non-linear and non-Gaussian state place model, and is sometimes implemented to apply map filtering technique [2][5][6]. PF is the algorithm by Monte-Carlo method composed of propagation, correction and re-sampling. To obtain an observation z_t at time t, the state $s_{t|t-1}$ at time t estimated by the state s_{t-1} at time $t-1$ (particle) and its likelihood (weight) are generated using pseudorandom number, and z_t is decided with particles' distribution and weights.

In our model, particles are updated only after each step event. The k-th step event $step_k$ is represented by step time t_k, step length l_k and heading direction h_k in global coordinate system (GCS). These parameters are computed by the method of SmartPDR [7] by using accelerometer, magnetometer and gyroscope in a smartphone. In our method, each particle has a state s_k after the k-th step event, and s_k is

$$s_k = [x_k, y_k, mpp_k, \theta_k]. \quad (2)$$

Here, (x_k, y_k) is a particle's position in image coordinate system (ICS), but we possess them with not integer but float. The parameter mpp_k (meter-per-pixel) is the length in GCS per one pixel, and θ_k is the heading direction h_k when the device turns toward *x*-axis of ICS.

A. Particle Initialization

We show the method to create an initial distribution $p(s_0|\emptyset)$ of the state s_0 at the step event $k = 0$. Let the two coordinates pointed by the user be $\left(x_{tap,n}^{ICS}{}', y_{tap,n}^{ICS}{}'\right)(n = 1,2)$, which is in ICS. Because the designation of the points is performed with the user's visual, there may be an error between the designated position and the user's real position $\left(x_{tap,n}^{ICS}, y_{tap,n}^{ICS}\right)(n = 1,2)$. Assuming that the probability distribution of the error is a Gaussian distribution with standard deviation σ_{tap}, this probability distribution $P(x_{tap,n}^{ICS}, y_{tap,n}^{ICS})$ is

$$P\left(x_{tap,n}^{ICS}, y_{tap,n}^{ICS}\right) = N\left(x_{tap,n}^{ICS}{}', {\sigma_{tap}}^2\right)N\left(y_{tap,n}^{ICS}{}', {\sigma_{tap}}^2\right) \quad (3)$$

where $N(\mu, \sigma^2)$ is an error model generated using a Gaussian distribution with a mean μ and a variance σ^2.

When step events between two taps are detected k' times, the motion on x-axis and y-axis in GCS is represented as

$$\begin{bmatrix} x_{move}^{GCS} \\ y_{move}^{GCS} \end{bmatrix} = \sum_{s=1}^{k'} l_s \begin{bmatrix} \cos h_s \\ \sin h_s \end{bmatrix}. \quad (4)$$

The moving length l_{move} and the direction θ_{move} between two taps are

$$l_{move} = \sqrt{(x_{move}^{GCS})^2 + (y_{move}^{GCS})^2} \quad (5)$$

$$\theta_{move} = \mathrm{atan2}(y_{move}^{GCS}, x_{move}^{GCS}). \quad (6)$$

Where the function $\mathrm{atan2}(y, x)$ is defined as

$$\mathrm{atan2(y, x)} = 2\tan^{-1}\left(\frac{y}{\sqrt{x^2+y^2}+x}\right). \quad (7)$$

We can also calculate the length l_{tap} and the direction θ_{tap} in ICS between two taps with a similar equation.

Therefore, the initial state $\mathrm{s}_0^{(i)} = \left[x_0^{(i)}, y_0^{(i)}, mpp_0^{(i)}, \theta_0^{(i)}\right]$ of i-th particle is decided with the following equations.

$$\begin{aligned} x_0^{(i)} &= x_{tap,2}^{GCS\ (i)} \\ y_0^{(i)} &= y_{tap,2}^{GCS\ (i)} \\ mpp_0^{(i)} &= l_{move}/l_{tap}^{(i)} \\ \theta_0^{(i)} &= \theta_{move} - \theta_{tap}^{(i)} \end{aligned} \quad (8)$$

B. Particle Propagation

The particle's state s_k after the k-th step event $step_k$ is created by a posterior distribution $p(s_k|s_{k-1}, step_k)$ with the previous state s_{k-1} and $step_k$. With paying attention that $\mathrm{x}_k^{(i)}$ and $\mathrm{y}_k^{(i)}$ are coordinates in ICS, the state $\mathrm{s}_\mathrm{k}^{(i)} = s_{k|k-1}^{(i)}$ of ith particle is decided as follows.

$$\begin{aligned} mpp_k^{(i)} &= mpp_{k-1}^{(i)} + N(0, {\sigma_{mpp}}^2) \\ \theta_k^{(i)} &= \theta_{k-1}^{(i)} + N(0, {\sigma_\theta}^2) \\ x_k^{(i)} &= x_{k-1}^{(i)} + l_k \cos(h_k - \theta_k^{(i)})/mpp_k^{(i)} \\ y_k^{(i)} &= y_{k-1}^{(i)} + l_k \sin(h_k - \theta_k^{(i)})/mpp_k^{(i)} \end{aligned} \quad (9)$$

Here, $N\left(0, {\sigma_{mpp}}^2\right)$ and $N(0, {\sigma_\theta}^2)$ are error models of $mpp_k^{(i)}$ and $\theta_k^{(i)}$ based on Gaussian distributions with deviations ${\sigma_{mpp}}^2$ and ${\sigma_\theta}^2$, respectively. These models also affect the moving length and direction.

C. Correction and Re-Sampling

In this paper, we assume that users go to their destination along a passageway, namely users do not go out of the passageway region from navigating start to end. Hence, we estimate user's position with map matching using passageway region created by the proposed method. In our method, map matching is applied to compute particles' weight, and the weight $\mathrm{w}_k^{(i)}$ of the i-th particle $s_{k|k-1}^{(i)}$ is

$$\mathrm{w}_k^{(i)} = \begin{cases} 0 & if\ \left(x_k^{(i)}, y_k^{(i)}\right)\ is\ not\ passageway\ region \\ 1/M & otherwise \end{cases}, \quad (10)$$

where, M is the number of particles after the correction. The output of the current position is the average of M particles.

Re-sampling is the process that reallocates all the particles. In our method, re-sampling is based on random sampling, but each particle has the same weight before re-sampling. Therefore, a particle after re-sampling is decided from M particles randomly.

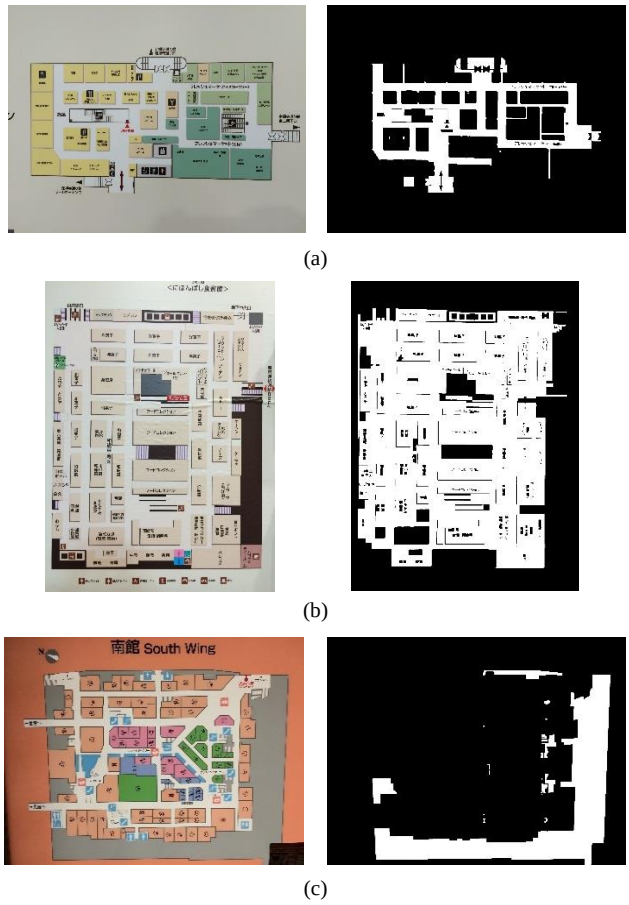
IV. EXPERIMENT

The proposed method has several thresholds and parameters, and we set them shown as Table I, which were decided by the preliminary experiments.

TABLE I. SYMBOLS AND VALUES

Symbol	Value	Symbol	Value
τ_d	3	σ_{tap}	25
τ_c	15	σ_θ	$5\pi/180$
α	2		

Symbol	Value
τ_l	1/500 of the image size
σ_{mpp}	1/60 of $mpp_{k-1}^{(i)}$

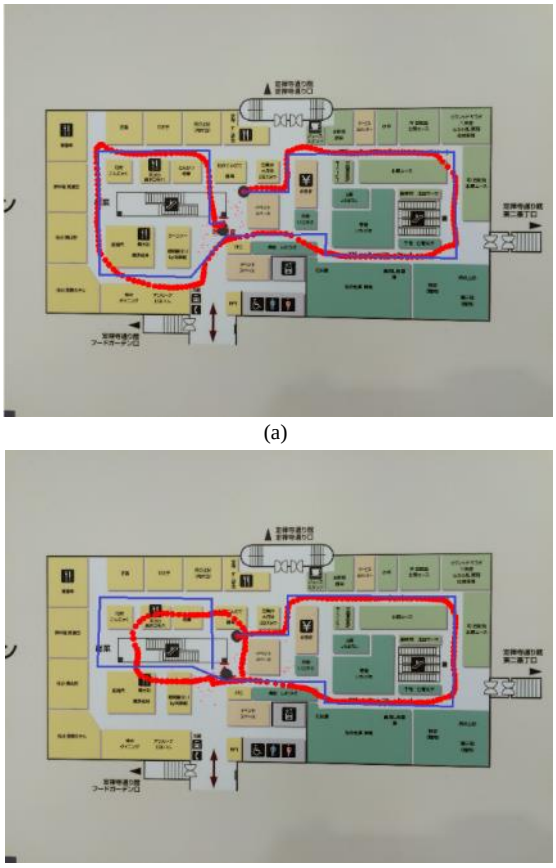


(a)

(b)

(c)

Figure 3. Results of passageway region estimation: original images (left) and estimated passageway region (right)



(a)

(b)

Figure 4. Result of the paths estimated using PDR and map matching: gray points are 2 tapped positions, red are user's estimated position and blue lines are real path.

A. *Passageway Region Estimation*

We conducted passageway region estimation using the proposed method. We applied the method to 92 pictures of information boards satisfying two conditions described in Section II except for some pictures with extremely low quality. In the method, an estimated passageway region consists of a set of labels and it has gaps at boundaries between each region of labels. To remove them, we applied erosion and dilation to each region, so that we can create passageway region more suitable for the position estimation.

Figure 3 shows examples of the results. We subjectively evaluated whether passageway region could be estimated to a certain extent, and we found that the estimation went well on 77% of information boards. We conducted the experiment using 92% of images that meet two conditions mentioned in Section II; therefore it means that the proposed method could estimate passageway region for about 70% of information boards.

The failures to obtain passageway region were mainly caused by labeling mistakes (Figure 3(b)) and failure to get FPL (Figure 3(c)). The former means that passageway and shop regions were mistakenly connected when labeling. The failure often happened when a shop label had similar color to passageway label and their edges are unclear. The latter means that a label unrelated to passageway region became FPL. The failure happened when FPL was not decided using sizes and colors.

B. *Localization*

We evaluated whether the proposed method can estimate positions by map matching with a picture of information board. The experiments were conducted on B1 floor in Jozenji-Dori building of Sendai Mitsukoshi (Fig. 3(a)). The proposed system was implemented on Sony Xperia S Tablet, and the sampling rate of each inertial sensors is 15 Hz. The users held the device on their right palm. The application for the experiment indicates the estimated path where the user walked in real time after tapping at two different position. We set the number of particles as 2000. We conducted the experiments four times.

Figure 4 shows examples of the results. A successful case is Figure 4(a), where the step events are totally 245 (11 steps between 2 taps). A failure case is Figure 4(b), where the step events are totally 218 (9 steps between 2 taps). Estimation error increased at the left area of the map in both cases. Of the four experiments, we succeed two times and failed two times.

 ISBN: 978-1-61208-465-7

We consider that the error was caused by the way to decide a current position. Now, the position is computed as the average of particle's positions before re-sampling, however, in case that particles are divided into two or more clusters, the output has a large error even if one of the clusters lies at the correct position.

V. CONCLUSION

In this paper, we proposed an indoor localization method using information board and several functions in smartphone. To estimate passageway region, we apply labeling to the picture of information board based on its color and edge, and make use of label information. Furthermore, we developed a method to estimate position on passageway region using PDR and map matching with PF. Through our experiments, we confirmed that the proposed method can accurately estimate passage region from 70% of information board, and the analysis results are useful for map matching with particle filter.

At the same time, we found out some problems. The method of labeling and passageway label estimation cannot obtain satisfactory results from some information boards. In reference to this, labeling results depend on the ambience when taking the picture. It is necessary to develop the method to analyze the picture resistant to light reflection. We were pointed out that using Lab color space might bring better results when labeling because the color space is designed to approximate human vision. Moreover, the labeling method is time consuming and requires high machine performance. Therefore, we are considering use of servers for the map picture analysis.

Furthermore, the proposed method has great limitation, and the method to decide a position from a distribution of particles has a room to be improved; we can track only on the passageway. To estimate the position on both passageway and shops, we would have to improve not only the method of map matching but also the method to analyze the picture of information board. Additionally, we have only confirmed whether the proposed method can trace the pass the user walks. It is necessary to evaluate our localization method quantitatively. They are remained as future works.

ACKNOWLEDGMENT

A part of this research was supported by JSPS KAKENHI Grant Number 15K12014. This research was also supported in part by JST, CREST.

REFERENCES

[1] V. Renaudin, O. Yalak, P. Tomé, and B. Merminod, "Indoor navigation of emergency agents,", Europian Journal of Navigation, vol. 5, no. 3, pp. 36-45, 2007.

[2] F. Zampella, A. R. J. Ruiz, and F. S. Granja, "Indoor Positioning Using Efficient Map Matching, RSS Measurements, and an Improved Motion Model, " IEEE Trans. on Vehicular Technology, vol. 64, pp. 1304-1317, no. 4, April 2015.

[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," IEEE Trans. on PAMI, vol. 24, no. 5, pp. 603-619, 2002.

[4] J. Canny, "A Computational Approach To Edge Detection," IEEE Trans. on PAMI, vol. 8, no. 6, pp. 679-714, 1986.

[5] C. Ascher, A. Kesseler, M. Wankerl, and G. F. Trommer, "Dual IMU Indoor Navigation with Particle Filter based Map-Matching on a Smartphone," Int. Conf. on IPIN, September 2010, pp. 1-5.

[6] O. Woodman and R. Harle, "Pedestrian Localisation for Indoor Environments," in Proc. 10th Int. Conf. UbiComp, 2008, pp. 114–123.

[7] W. Hang and Y. Han, "SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization," IEEE Sensor Journal, vol. 15, pp 2906-2916, no. 5, May 2015.

 ISBN: 978-1-61208-465-7

Towards Antipatterns-Based Model Checking

Hassan Loulou
University of Paris-Sud
Paris, France
email: hassan.loulou@u-psud.fr

Sebastien Saudrais
ESTACA'LAB
Laval, France
email: sebastien.saudrais@estaca.fr

Hassan Soubra
ESTACA'LAB
Saint-Quentin-en-Yvelines, France
email: hassan.soubra@estaca.fr

Cherif Larouci
ESTACA'LAB
Saint-Quentin-en-Yvelines, France
email: cherif.larouci@estaca.fr

***Abstract*—Discovering bugs in the early stages of the development life cycle is an important issue. However, software model checking realized by transforming design models into formal methods cannot test all the possible execution scenarios. Thus, we developed an approach to guide the model checker and the security engineer to the most suspicious parts of these models firstly. The objective is to build and analyze antipatterns to notify the security engineer to concentrate on specific parts of their models during the model checking. Our first contribution is dedicated to exploring ProB model checker features which help the translated model to find attack scenarios automatically. The second one is the definition and the analysis of 10 antipatterns as a step towards their automatic detection.**

Keywords–Antipatterns; Model Checking; Formal Methods.

I. INTRODUCTION

Hidden errors in software design phase lead in later software development stages into complex bugs which need a lot of time to be solved. Thus, discovering these errors at this phase is of high importance.

Few works have examined the impacts of models' functional and non-functional artifacts co-evolution on design constraints. This type of co-evolution exists in UML profiles such as SecureUML [1]. Our approach validates SecureUML models' dynamic aspects by applying model checking operations after transforming them into a formal representation called B-method [2]. Meanwhile, the validation of design models cannot explore all the possible software executions and requires certain level of experience with formal methods. For realizing a systematic validation of security policies, the model checker needs to be guided to discover design constraints by applying appropriate enhancements on the models' resulted formal representations. Furthermore, system designers would prefer intuitive solutions depending on antipatterns representations for those design structures which must be avoided. Also, a solution for highlighting suspicious parts of the models which are likely to introduce violations during the system evolution may make their work easier.

We study the existing facilities for guiding the model checker to detect security constraints' violations in SecureUML. Thereafter, we define the design artifacts which were the source of violations in form of antipatterns. These antipatterns are substructures suspected to be the reason of design constraints' violations during software evolution. They are discovered after the transformation of models into a formal representation and after launching their formal verification using ProB [3] model checker's facilities. We aim at paving the way towards an automatic detection of the root cause of security bugs by introducing design artifacts which are responsible for these bugs to the software security engineer. We applied our approach on a case study related to the verification of access control constraints.

This paper is organized as follows: In Section II, we explain the structure of SecureUML [4] profile and the limitations of its validation works. In Section III, we show how we exploited ProB model checker to find the violations automatically. Thereafter, in Section IV, we introduce an example of the extracted antipatterns. Finally, in Section V, we make a conclusion on our contribution and perspectives.

II. SECUREUML VALIDATION

In this section, we show the basic idea of SecureUML, the works related to validating it and our choice to represent SecureUML models and to exploit this representation.

A. SecureUML

Role-based access control (RBAC) has been standardized by the National Institute of Standards and Technology (NIST) [5]. It defines a role as a set of permissions to access resources. Users get their permissions by being assigned to one or more roles. SecureUML [1], is an extension of UML for specifying RBAC access control policies. In SecureUML, a permission is a relation connecting a role to actions on resources. The permission semantics are defined by the *action* elements used to classify the permissions [6]. Every *action* represents a class of security relevant operations on a protected resource. Access control can be expressed as an assignment of users to permissions by using their roles. Otherwise, authorization constraints, or more commonly the Object Constraint Language (OCL) [7] constraints, are checked on snapshots of the meta-model. OCL can specify constraints on users permissions in an expressive-contextual manner.

B. Related Works

Yu *et al* [8][9] exploited USE tool to support lightweight analysis of RBAC security policies expressed by UML and OCL to find violations. In their approach, an application design model was translated into a model containing predefined valid sequences of object diagrams. A snapshot is an object configuration that describes a system state. A sequence of object interactions, called a scenario, is then checked against the invariants defined in the snapshot model. After the analysis, if a good scenario is described as invalid, or a bad scenario is described as valid, then there is a problem in the security policy which either prevents valid scenarios or permits invalid ones. Basin *et al* [6][10] introduced SecureMOVA. This tool is

 ISBN: 978-1-61208-465-7

TABLE I. THE LACK OF AUTOMATED DISCOVERY FOR ATTACK SCENARIOS

	Supporting tool	Model expressed as	Coverage of static aspects	Animation of models	Generating attack scenarios	Funcutional evolution empacts on the security	Define attack patterns in UML	Systematic discovery of attack scenarios
Yu	USE	SecureUML diagrams	Yes	No	No	No	No	No
Basin	SecureMOVA	Component diagram *OCL + RBAC*	Yes	No	No	No	No	No
Wuliang Sun	Alloy	Translation from Class diagrams and OCL	Yes	Yes	Only security parts evolution	No	No	No
Nafees Qamar	Z	Translation from SecureUML	Yes	Yes	No	Yes	No	No

used to ask questions about a current state, i.e., a given object diagram. Such queries return the permissions authorized for a given role, or a given user. However, these two approaches do not consider the real execution sequence of operations to reach a specified state. They just give a possible object diagram representing a required constraint and conforming to the class diagram according to some specified constraints. Nevertheless, they do not ensure the reachability of this state. Another approach tried to simulate the real execution of software systems by introducing the notion of execution scenarios. It depends on transforming UML diagrams augmented with OCL constraints into Alloy [11][12][13]. However, these works do not transform the security constraint to validate them by Alloy model checker and they do not consider the impacts of the functional model evolution on the security policy. Moreover, these works suffer from the false negative alerts (Alerts that should have happened but did not because the model checker did not reach certain states). Additionally, they do not define a systematic approach for guiding the model checker. An interesting approach exploited Z formal method to animate SecureUML models [14]. However, false negatives still exist and antipatterns are not defined and thus not exploited to automate the search for attacks. We compare these works in Table I, showing the lack of a validation approach for the functional model evolution impacts on the security parts of secureUML. Also, no antipatterns are defined to guide the security engineer or the model checker into the suspicious parts of the models under test. To solve this problem, we translated both functional and security parts into B-method. Then, we proposed solutions to guide the model checker to find the co-evolution of SecureUML parts and we exploited these solutions to resume our findings in the form of antipatterns.

C. B-method for SecureUML validation

We decided to transform SecureUML models into B-method [15]. The reason is that B-method allows to simulate the co-evolution of functional and non-functional parts of models by its animator called ProB. This simulation mimics the real execution of software system. Moreover, a B-method tool called B4Msecure [16] is developed by Ledru *et al.* [17][14] to transform SecureUML models into B-method. B-method is a formal method for specifying, refining and implementing software systems. It is based on *Set Theory* and *Predicate Calculus*. Each B model is called an abstract machine. These machines are animated using ProB. Yet, the model checker needs to be guided by the user in order to find the potential

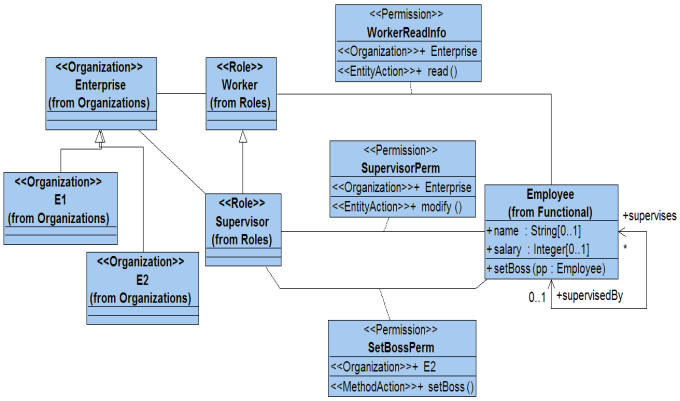


Figure 1. SecureUML case study

attacks. Thus, the tool still needs an automation in order to search for models' most suspicious artifacts and to examine the impacts of these artifacts' possible compositions.

III. OUR PROPOSITIONS FOR VALIDATING SECUREUML

In this stage, we aim at exploring ProB features which help to find an attack scenario depending on the capabilities of ProB model checker.

A. Illustrative example

We built the security model shown in Figure 1. It is supplied with a stereotype representing the notion of *organization*. In this way, a *user*, called Jack, who has the role *supervisor*, is able to access data related to *employees* in *organization* E2, but he is prevented from doing so in E1. In this model, we notice that a new function *setBoss()* is added to express the uncontrolled reflexive association presented in [10] and to examine the potential risks resulting from the solution introduced there.

B. Validating the resulting model by ProB

We explored many mechanisms to find attack scenarios. The validation process considers an initialization state as shown in Figure 2, where there are two *employees* and one *supervisor* in the same *organization* E2.

1) Finding flaws by querying the functional model: When an *employee* in the functional model gets the role *supervises* of the reflexive association, he gets simultaneously a new role in the security model and new permissions according to this new role. In our example, when the *employee* Martin becomes a *supervisor*, an interaction between the functional and the security models occurs by invoking the operation

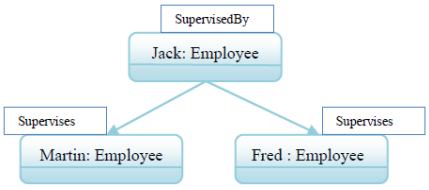


Figure 2. The initial state

 ISBN: 978-1-61208-465-7

add_userAssign. This operation makes him a *supervisor* in his *organization*. As a result, the following assignment, which adds new role to the user in his *organization*, takes place: *(user ↦ (org ↦ role))*. This relation has the following values in our illustrative example: *(Martin ↦ (E2 ↦ Supervisor))*.

We can search for this scenario in ProB by searching a state satisfying specific predicate in the checked model as follows: find a state in which *Boss(Martin) = Fred*. As a result, the model checker finds a scenario which makes an *employee* as a *supervisor*. Consequently, this new *supervisor* has new permissions and this case is suspicious. We found that as the reflexive association is not controlled by constraints, it does not have a clear interpretation. Moreover, as Basin *et al* [10] used an *actional* tool, which cannot simulate the real reachable methods in a specific system state, they could not consider the operations' real effects on the security policy and they could not find this potential attack.

2) Querying security aspects of the model: The proposition here is to make queries about all the roles which someone must not be authorized to have. Thus, we ask the model checker queries about the roles of users. If the translated model contains a scenario which can give a user, Martin, the role *supervisor*, considering that he was an *employee* with the role *worker*, then this scenario is suspicious and it can be a flaw in the security policy. The query about the system states takes the following form: *User_assign (Martin) = (E2 ↦ Supervisor)*. This query means: is there a scenario that leads the system to a state where Martin is a *supervisor*? Considering that the latter state is assumed as an invalid state. We search for this state, because the existence of such sequence of operations may make Martin able to change the salary of other users in the system, which is considered illegal in that *organization*.

The result given by the model checker is the following susceptible scenario: after giving the values of the system constants concerning the system users, the system's functional and security models are initialized. After that, the users are assigned to their permissions. Jack, who is a *supervisor*, connects to the system as a *supervisor* in the *organization* E2. This assignment of Jack to his roles occurs in the session S1. While, in another session, a user whose name is Fred and who has the role *worker* takes the session S2 and connects to the system as a *worker*. When Fred gives Martin a position *supervisor* by applying the function *secure_Employee_setBoss*, the state of the system evolves and Martin gets the role *supervisor*. As a result, with the existence of another *employee* like Bob, Martin who has got temporarily the role *supervisor* becomes able to make Bob as a *supervisor*. Subsequently, even if Martin loses his new role, Bob holds this role and can change the salary of his colleagues Martin and Fred.

3) Exploiting the attainability of an operation: An operation is enabled if its precondition and its guard's sections are true. These two sections are computed taking into account the system state. We proposed a new solution benefiting from this property. The mechanism of this scenario detection solution is explained by the following example. A system variable *currentUser* is related to an active session user. Therefore, we can add a constraint in the guard's section of an operation, where this constraint says that the operation will not be enabled if the current user is not Fred, for example. We do that taking into account that this user does not have an authorization to execute this operation. Next, we search for a scenario in which this operation could be attainable and executable. If such scenario exists, we conclude that there is a suspicious scenario which may cause a serious attack. As will be shown next, the constraint *(currentUser = Fred)* is added in the guards section of the operation *secure_Employee_SetSalary*:

```
secure_Employee_SetSalary(Instance, Employee_
    ↪ salaryValue) = SELECT currentUser = Fred
```

The disadvantage of this solution is: to reach the state we are searching for (where currentUser = Fred), we may need the same operation without the added pre-condition. Thus, the solution is to add another operation with the same name but without that obstructive constraint. In this way, this additional operation will be executed first, if needed. Then, ProB continue searching for the target suspicious state.

4) Searching flaws by asking ProB about permissions: In this way, we are interested in finding a scenario where a user is capable of reaching a permission he did not have before. For example, is there a scenario in which the user Martin can get the access to execute the operation *employee_SetSalary*? The execution of this operation was granted, at the beginning of the system execution, only to Jack who is a *supervisor*. This question can be formulated using B as follows:

```
employee_SetSalary ϵ isPermitted[currentOrgRole_
    ↪ Session] ∧ CurrentUser=Martin
```

Consequently, if the model checker reaches a state where the previous constraint holds, we conclude that the user Martin will be able to get an unauthorized permission. The resulted scenario given by the model checker is the same as the one mentioned in the previous solutions.

5) Taking the initial state into account: We consider different possible object diagrams resulted from the reflexive association and its multiplicities in the illustrative example.

a) Studying initialization object diagrams: The first object diagram, Figure 3, is composed of the following structure: an *employee* with a role *worker* who is not associated to a *supervisor* and another *employee* with a role *worker* associated to a *supervisor*. In this state, we cannot find an attack scenario. No scenario can lead Martin to change Freds salary because Jack is not the *supervisor* of Fred. Thus, he is not able to execute the *setBoss()* operation. Likewise, in the following state, Figure 4, there is no attack scenario. The only case where there is an attack scenario is, as shown in Figure 2, when there is an object diagram containing more than one *employee* managed by a common *supervisor*. This is because the existence of a *supervisor* for an *employee* is essential to change the value of the reflexive association represented as a relation called boss in the functional model. This relation issues an association between two objects in the Employees functional class diagram.

Moreover, we found another kind of attack happening when we have a hierarchy of *employees* in the same *organization*, as shown in Figure 5. It is impossible to change the salary of the *employee* E7, as he is not associated to a *supervisor*. An *employee* E1, E3 has found a scenario to change the salary of their *supervisors* E3, E5 respectively. As a result, we found out that the initial state which is constructed using the *supervisor* hierarchy two times at least is essential to have such risk.

 ISBN: 978-1-61208-465-7

Figure 3. Initialization state1.

Figure 4. Initialization state2

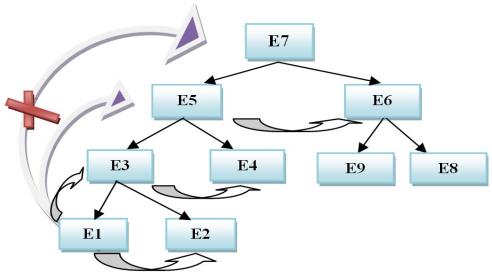


Figure 5. Initialization state3

b) An approach to construct an important object diagram (tree): A solution is needed to construct either all the possible object diagrams of the class diagram or all those states which are considered more suspicious to produce the attack scenario. The produced object diagram differs according to the structure it tries to instantiate. For example, the different states we must consider in the previous reflexive association are: (i) initializing the B machine with two *employees*, one of them has a different *manager*, (ii) two *employees*, each of them has one different *manager*, (iii) a tree structure representing the structure of the hierarchy. (iv) considering another *organization* with some users.

The initial state must be considered separately in each new diagram. However, in addition to the previous recurring structure, we try to give solutions to produce initial states for the main possible structures of SecureUML as follows:
(A) When there is an association affecting an OCL constraint, the multiplicity of the two ends of the association has to be considered in the initialization. For the *many* multiplicity, two objects at least are instantiated.
(B) Each of the entities mentioned in a constraint must be instantiated in order to construct a structure capable of tricking the OCL constraint. As shown in Figure 6, a malicious user connects with a *manager* and *staff* roles at the same time. When he is a *manager*, he delegates the user with the role *staff* (himself) in order to make him able to approve the refund, as stated by the OCL constraint associated with the permissions on the class refund. Thus, this user will be able to give the value True to the variable *IsApproaved* in the class *refund*. Hence, the constraint *Approved by 2 managers* became true, as the staff become also a *manager*. Then, this *manager* approves the refund and *refund* procedure starts. Thus, the *manager* prepares and issues the payments after approving them. This separation of duties (SoD) problem comes from the structure in

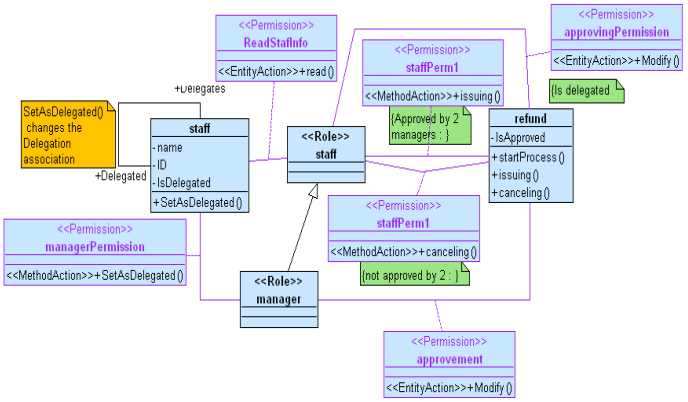


Figure 6. Example showing the importance of instantiating OCL entities

which the *manager* inherits the permissions of *staff* role. The solution to this problem is to add an OCL constraint saying that the same person can connect as a *manager* or as a staff in the same session, but not as the both.
The instantiation of this diagram could take the following way: an instance of the class refund and two users had to be instantiated as mentioned in the OCL constraint *Approved by 2 managers*. Additionally, according to the constraint *is delegated*, the delegation instance must be presented. That means, we must produce an object *delegates* and an object *delegated*. As a result, this initialization diagram is capable of starting the game of finding the attack.
(C) For a unidirectional association, the accessed object must be instantiated and linked to the instance accessing it.
(D) When there is an OCL constraint controlling permissions and depending on an attribute value, we ought to instantiate the class where this variable exists, then we assign a user who has a *modify* permission.
(E) When there is a role inheritance, we assign users to the inheriting roles to check all the prevented separation of duties conditions as happened in Figure 6. Thus, an invariant has been added to prevent this user getting two conflicting roles.

The result of Figure 1 translation contains a reflexive association: $bb \in Employee \nrightarrow Employee \wedge bb$. Before setting the permissions of users, we applied the previous approach for constructing an efficient object diagram. The approach contains the following steps: (i) give the possible values for associations' ends and specify their multiplicities, (ii) avoid constructing circular associations to avoid a state where a person is *supervisor* of himself, (iii) add properties for associations members' multiplicities, (iv) initialize the functional model, (v) produce the possible values of the relation: $user \rightarrow (\{E2\} \times \text{ROLES})$, (vi) initialize the user assignment model. To make the possible suspicious initial state (the tree shown before), the following constraint is defined on the association multiplicity:

$$(card(ran(bb))=1 \wedge card(dom(bb))=2) \bigvee (card(ran(bb)) \hookrightarrow =2 \wedge card(dom(bb))=2) \bigvee (card(ran(bb))=4$$

As a result, the way we ask our queries to the model checker must change. This is because the model checker possibly will take a shorter path while searching and this path is not normally an attack scenario. For example, if we ask ProB about a state in which Martin can change the salary of Fred, the

model checker finds rapidly a scenario satisfying this query. However, in this scenario, Martin is assigned directly to the role *supervisor* and he changes the salary of Fred, which is not an interesting scenario for us. As a solution, we propose to make the queries separated from users names and related to a general description of the attack meaning, as shown in the next section.

6) Generalizing the way we search the suspicious scenarios: To avoid the problems of making automatic initialization with specified properties, we propose to avoid asking about a user who has a specific permission. The question becomes more general as follows: is there a scenario which allows a user who have a *worker role* to have a *supervisor role*?

$$\forall(u).(u \in USERS \wedge u \in dom(user\_assign\ (ran(user\_assign)\ \{Worker\})) \wedge u \notin ran(B) \implies u \notin dom(user\_assign\ (ran(user\_assign)\ \{Supervisor\})))\wedge..$$

Where, the variable B contains the *boss* relations values calculated during the initialization of the security machine. In another way, the set *wasWorker* contains all the users assigned to the role *worker* at the initialization step of the security machine. Consequently, we add this formula as an invariant of the machine. Thus, when this invariant is violated, this means one of the users has obtained the role *supervisor*. This formula is constructed as follows:

$$\forall\ (u).(u \in USERS \wedge u \wedge wasWorker \implies ran(\{user\_assign(u)\}) \neq \{Supervisor\})$$

Where:

$$wasWorker := dom(user\_assign \rhd (ran(user\_assign) \rhd \{Worker\})) \cap dom(boss)$$

The other invariant we added to capture the attack risks concerns preserving the permissions during the execution of the model checker. It can answer the following question: is there a scenario that leads to an increase in any users permissions during the system execution? The formula is constructed to ask always about the user connected to the role which has a lower number of permission such as the role *worker* in the illustrative example. The following formula shows an example of an added invariant in the security machine. This formula searches if the number of a user's permissions with the role *worker* may increase during the execution of the system.

$$\forall\ (u,org,role).(u \in USERS \wedge u{=}currentUser \wedge u \in dom(boss) \wedge org \in ORG \wedge org{=}E2 \wedge role \in ROLES \in role{=}Worker \implies isPermitted[\{user\_assign(u)\}] - isPermitted[\{(org \mapsto role)\}] \neq \varnothing)$$

Similarly, we can add the names of other roles if they exist in the system to be contained in the last formula. By doing so, the query covers all the users in the system. As a result, searching those scenarios, which may affect the security policy, has become easier.

7) Detecting possible vulnerabilities in OCL constraints: The goal in this stage is to violate the OCL constraints which describe security aspects because the ability to violate them is considered as a flaw. To succeed in doing that, we express this constraint but with a change in their parts expressing the name of the roles. Then, these names are replaced by other

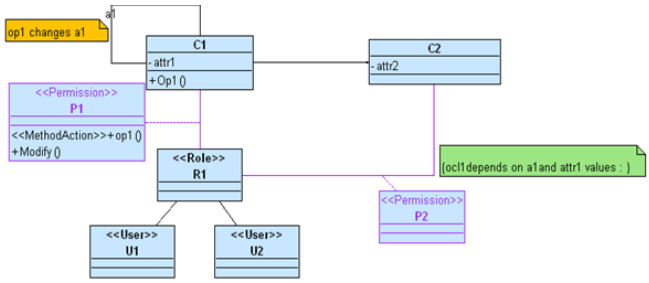


Figure 7. An example of the defined patterns

unauthorized ones. In other words, we reached this state by adding a constraint as a pre-condition for an operation like the operation *secure_setSalary()*. This constraint restricts the operation execution permission to Fred who has the role *worker*, such access must be prohibited. This constraint specifies Fred as the *employee* whose salary will change. Checking the model under this condition shows an important evolution of the functional model. This evolution violates the access permission which limits the execution of setSalary to a *supervisor* in the same *organization* of the *employee*.

Furthermore, when an employee is delegated to do the *supervisor* tasks, he keeps his previous role. That means a person who executes the operation *secure_setSalary()* can be the same one for whom this operation is executed. As a result, an *employee* is able to change his own salary. Thus, a user keeps owning his previous roles when his position changes. The structure of SecureUML model, which contains the previous artifacts, could be considered as a recurring antipattern that needs to be checked in new models under test.

IV. Extracting The Antipatterns

To reduce the impacts of the state explosion, we are going to define scenario attack patterns according to our experimentations using the previous translation and validation techniques. Thereafter, we will try to find out the suspicious recurring structures, as well as the operations affecting these structures to help the model checker to find its way to the suspicious attack states.

A. Defining SecureUML antipatterns structures

To help the model checker estimating if a translated SecureUML model may have an access attack, we constructed a table containing characteristics of previous suspicious SecureUML diagrams. For the time being, we are going to define some patterns found in the examined diagrams.

In the following, we introduce an example of the defined antipatterns structure, *suspicious recurring structure*, shown in Figure 7. In this antipattern, there is a read permission on an entity. But, after initializing the security machine, the user finds himself able to modify parameters in the entity C2.

The detailed parts of this *suspicious structure* are as follows: Users u1, u2 assigned to role r1. Class C1 has a reflexive association a1. R1 has a *modify* permission p1 on class c1. R1 has *methodAction* permission on the operation op1 which changes a1. A1 and attr1 decide the value of ocl1. R1 has permission p2 on class c2 controlled by ocl1.
This structure existed in different examples of SecureUML.

 ISBN: 978-1-61208-465-7

TABLE II. REASONS OF VIOLATIONS IN THE ANTIPATTERNS

SubStructure	Description
Sub1	There is a unidirectional association.
Sub2	The accessing class is itself accessed directly by a fullaccess permission which affects the accessed class operations and attributes.
Sub3	The role owning a direct permission on the accessing class has only Read permission on an accessed class.
Sub4	An OCL constraint depends on an association value (grant permission).
Sub5	An operation modifies an association's values.
Sub6	Methodaction permission on an operation which modifies an important association.
Sub7	An OCL constraint depends on an attribute value.
Sub8	Role inheritance without adding SoD constraints.
Sub9	More than one user are assigned to the same role, but they have some hierarchy defined by a reflexive association. Their permissions could differ according to a constraint.
Sub10	Assigning roles after changes in a hierarchy association.
Sub11	User assigned to two roles at the same time. These two roles did not come from the inheritance. They have permissions on the same entity.
Sub12	An operation changes an attributes value which participates in calculating the value of an OCL constraint.

We use it with the other supposed 9 antipatterns to analyze and predict the existence of flaws in the new models.

B. Extracting the reasons of attack in the previous structures

In order to avoid the arbitrary search of the model checker algorithms, we have extracted the most important factors of the security policy vulnerabilities depending on some characteristics, shown in Table II. These characteristics describe the most suspicious recurring structures used in SecureUML diagrams.

For each attack pattern, we search for the existence of each of the sub-structures (sub-structures 1 to 12). When this structure exists, we modify the scores of this pattern parts in a probabilistic suspicious-table. Thereafter, the suspicious-table is used for guiding the model checker. Due to the limited space, we show later how to extract the exhaustive search plan which concentrates on suspicious execution scenarios. The optimization process of the suspicious-table is done according to the incremental feedback loop process, shown in Figure 8.

V. CONCLUSION AND FUTURE WORK

According to our knowledge, bugs resulted from the co-evolution of the functional and non-functional parts of SecureUML models are not sufficiently studied. Moreover, no antipatterns have been defined to help the system designers to avoid suspicious compositions of design artifacts.

This paper introduced an approach based on the transformation of SecureUML diagrams into formal models to simulate their execution. Moreover, we exploited the possible offered validation techniques in the model checker ProB. By applying them on many different systems, we could define 10 kinds of suspicious rudimentary antipatterns. Then, we analyzed them to notify the security engineer to investigate much more validation efforts on the suspicious parts when they are introduced in their models. However, this work must be extended to detect the antipatterns automatically. Moreover, the transformation of the security models into B-method still needs user interventions to make the resulted model executable. This limitation should be avoided to make the approach presented in Figure 8 achievable.

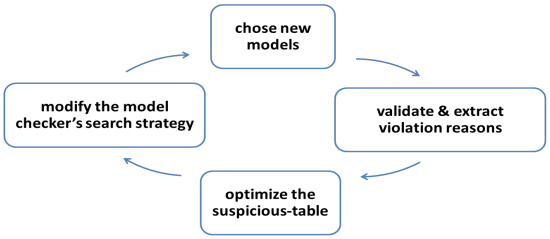


Figure 8. Feedback loop for guiding the model checker

Next, we are going to use a graph query language to efficiently discover antipatterns substructures in large models. The objective of this task is to generate a controller to guide the model checker automatically to design flaws.

REFERENCES

[1] D. Basin, M. Clavel, J. Doser, and M. Egea, "A metamodel-based approach for analyzing security-design models," in Model Driven Engineering Languages and Systems. Springer, 2007, pp. 420–435.

[2] J.-R. Abrial, M. K. Lee, D. Neilson, P. Scharbach, and I. H. Sørensen, "The b-method," in VDM'91 Formal Software Development Methods. Springer, 1991, pp. 398–405.

[3] "Prob," https://www3.hhu.de/stups/prob/index.php/Main_Page, accessed: 2016-01-21.

[4] T. Lodderstedt, D. Basin, and J. Doser, "Secureuml: A uml-based modeling language for model-driven security," in ł UML 2002The Unified Modeling Language. Springer, 2002, pp. 426–441.

[5] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, 2001, pp. 224–274.

[6] D. Basin, M. Clavel, J. Doser, and M. Egea, "Automated analysis of security-design models," Information and Software Technology, vol. 51, no. 5, 2009, pp. 815–831.

[7] "Ocl," http://www.omg.org/spec/OCL/, accessed: 2016-01-25.

[8] L. Yu, R. B. France, and I. Ray, "Scenario-based static analysis of uml class models," in Model Driven Engineering Languages and Systems. Springer, 2008, pp. 234–248.

[9] L. Yu, R. France, I. Ray, and S. Ghosh, "A rigorous approach to uncovering security policy violations in uml designs," in Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on. IEEE, 2009, pp. 126–135.

[10] D. Basin, M. Clavel, and M. Egea, "A decade of model-driven security," in Proceedings of the 16th ACM symposium on Access control models and technologies. ACM, 2011, pp. 1–10.

[11] "Alloy," https://www3.hhu.de/stups/prob/index.php/Main_Page, accessed: 2016-01-22.

[12] W. Sun, R. France, and I. Ray, "Rigorous analysis of uml access control policy models," in Policies for Distributed Systems and Networks (POLICY). IEEE, 2011, pp. 9–16.

[13] M. Toahchoodee, I. Ray, K. Anastasakis, G. Georg, and B. Bordbar, "Ensuring spatio-temporal access control for real-world applications," in Proceedings of the 14th ACM symposium on Access control models and technologies. ACM, 2009, pp. 13–22.

[14] Y. Ledru, A. Idani, J. Milhau, N. Qamar, R. Laleau, J.-L. Richier, and M.-A. Labiadh, "Taking into account functional models in the validation of is security policies," in Advanced Information Systems Engineering Workshops. Springer, 2011, pp. 592–606.

[15] J.-R. Abrial, J.-R. Abrial, and A. Hoare, The B-book: assigning programs to meanings. Cambridge University Press, 2005.

[16] "B4msecure," http://b4msecure.forge.imag.fr/, accessed: 2016-01-15.

[17] Y. Ledru, A. Idani, and J.-L. Richier, "Validation of a security policy by the test of its formal b specification: a case study," in Proceedings of the Third FME Workshop on Formal Methods in Software Engineering. IEEE Press, 2015, pp. 6–12.

 ISBN: 978-1-61208-465-7

Search++:
More Control than a Simple Search Interface without the Complexity and Confusion of Advanced Search

Alessandro Simone Agnello
Haim Levkowitz

Department of Computer Science
University of Massachusetts
Lowell, Massachusetts
01854-2874
alessandro_agnello@student.uml.edu
haim@cs.uml.edu

***Abstract*—On-line search engines are a key component of all on-line activity and have evolved through the years. Despite search engine advances, users today may need to execute multiple searches prior to reaching their desired search results. However, executing multiple searches in an effort to do so can be time consuming and may not lead to the best results. Leveraging complex search interfaces, including Boolean filters, can exacerbate the problem if the interface is too confusing and unfamiliar to the user. To help overcome these issues, we introduce Search++, a new search interface that reduces the steps for a common user to reach their desired search results, through the use of latest Web technologies and advanced visualizations.**

Keywords–Search Interface; User Decision Pattern; Priority Sorting; Web Visualization.

I. INTRODUCTION

Numerous studies have attempted to characterize the cognitive process of searching for information and model the related series of steps. A common theme amongst the models is a step or steps, where an individual will evaluate results of a search and use the information for a subsequent search to reach, or get closer to, the desired search results. These steps are referred to in many ways amongst the models: “Refinement” [1], “Formulation” [2], and “Query reformulation tactics” [3] to name a few. These models are applicable to searching for information on-line. They have been used to formulate new techniques for on-line search engine interfaces.

Search engine interfaces have evolved throughout the years starting with the first on-line search engine “Archie”, which included many Boolean filter options [4], to today's popular search engines, which include a single input box, button, and complex back-end algorithms to return results. Simplified search interfaces are preferred by common users as advanced search engines can be overwhelming to them. Despite advancements users find themselves executing multiple searches prior to honing in on the desired search results. This effort is time consuming and can lead users to feel discouraged and frustrated [2].

It has been found that a technique that includes summarized hierarchical display of data related to the search results along with the search results themselves can be effective at quickly reaching the final desired search results. However, success depends on how well the summarized data aligns with the search terms [5]. Site-specific search engines, such as Amazon, include Boolean filters with search results. However, the filters are pre-arranged based on the category (e.g., price filter may have a range from lowest to highest priced item within the result set). Furthermore, Boolean filters omit search results, which may further delay a user from honing in on the desired search results. Users have been known to enter broad search terms at first in an effort not to exclude search results that may help them further define the desired search results [6]. This suggests that a user's awareness of all available data helps her/him reach the desired search results.

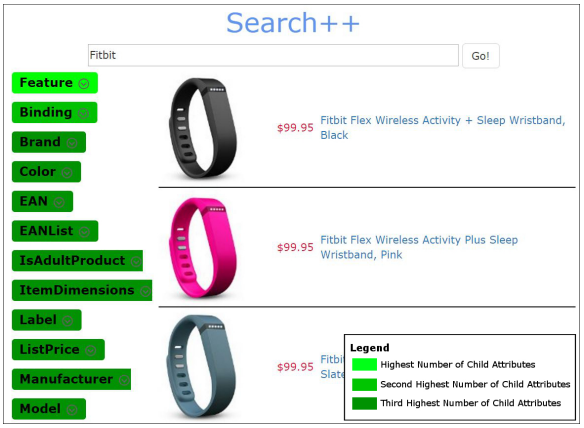


Figure 1. Subset of a result shown, when a user typed in Fitbit as a search term.

To help accommodate users, refining their search and to alleviate their struggling using advanced interfaces, we introduce a new technique, Search++. Search++ bridges the gap between the familiar but simplistic standard search interface and the complex advanced search interface. Search++ utilizes an expandable data source plugin utility that allows users to search for attributes of single or multiple source result sets. Search++ allows the user to rank these attributes, including their sub-attributes. The ranking informs Search++ what the user is more interested in (e.g., the user has a higher interest in the price of an item than in its weight). Colors are used to help depict which attributes have the largest number of sub-attributes. Search++ associates a base light green color with

 ISBN: 978-1-61208-465-7

the attribute with the highest frequency of sub-attributes. Light green was chosen to show contrast against the background and green creates a relaxing effect for people [7]. We felt that attributes with the highest frequency of sub-attributes, shows the most likely sorted set for the common user. These attributes are most common throughout the returned result set. Attributes are darkened in color proportional to their decrease of sub-attribute frequency. (E.g., Features: 20 sub-attributes, Price: 10 sub-attributes, thus Features is light green, and Price is darker). This allows the user to quickly visualize the attributes with greatest sub-attribute frequency. In Figure 1, we provide an example result set of Search++. Upon initial data retrieval or user interaction our query processing modules will be triggered. These modules retrieve, normalize, sort, and visualize user's search input and interaction (discussed in Section IV-B). The main contributions of Search++ are:

- A new search interface that provides more control than the standard stripped down interface. Without overwhelming the user with advanced search interface components.
- User defined and controlled relevance ranking.
- A framework that aggregates multiple Web Server/Client technologies.
- An interactive and visual interface that lets the user control the process and view results.

To help evaluate the capabilities of Search++, we constructed a case study using Amazon Web Services (AWS) [8]. We asked a group of ten individuals who described themselves as "non-technical" to search for six unique products they are seeking to purchase; three searches using Amazon's search interface and three using Search++. Section VI describes this in further detail.

In Section II, we discuss the past and present implementations of search interfaces. Section III define the components of Search++. Section IV describes how the various components work among each other and user interaction. Section V define the processing modules upon initial load and user interaction results. Section VI shows our case study using AWS as our data repository. Section VII recaps our findings of Search++.

II. Related Work

The goal of any search engine is to find information that matches or is relevant to some criteria [9]. In the early days of on-line computing, it was common to have complex input criteria that may not be readable/understandable to today's common user.

Complex interfaces are only usable by a few very technically savvy users who are able to negotiate the interface's complexity to their needs. They often cause confusion and frustration for the common user. Various approaches, such as WISE-Integrator [10], try to bridge this gap by giving the user more fields to fill out (e.g., price range, author name, etc.), but this assumes the user knows more about what they are looking for, which is often not the case. Common e-commerce based search engines [11] have filter-down mechanisms to help users identify some of their needs (e.g., price, brand, etc.). However, the user cannot explicitly state what is their highest relevance. These filtering mechanisms intentions are to remove entries rather than resort them. Complex, advanced search engine interfaces [12] [13] are not designed for the common user. They show powerful results, but they lack in ease of control and instant understanding/intentions of use.

Our Search++ approach is designed to support a common (i.e., not particularly technical) user, offering a little more control than just an input field, without overwhelming the user with a tremendous amount of additional fields that may or may not be coherent. We define a dynamic attribute as a parent level descriptor of a given item. A dynamic attribute instance may have child attributes. Child attributes are child level descriptors of a single dynamic attribute. It is possible that a child attribute may also be a dynamic attribute and have child attributes. Our case study (Section VI) does not show any child attributes also being a dynamic attribute. Dynamic attribute instances are obtained from the search result set and grouped in a way that shows only distinct results. Our case study (Section VI) demonstrates various forms of this scenario. We color code each attribute to show frequency of sub-attributes (discussed more in Section V-D). Additionally, the user can sort these attributes based on her/his priorities. Upon resorting attributes, Search++ will display results based upon their attribute ranking.

In Section VI, we demonstrate an application of Search++ on data exploration of e-commerce products. We explore the usage of ranking of dynamic and child attributes along with color scheme. We demonstrate the usefulness of Search++, and display the value of ranked attribute and sub-attribute results.

III. Components

The current fundamental set of Search++ components includes three base views (Dynamic Table, Input Box, and Dynamic Attributes), Session State Management, Data Source Plugins, and four Query Processing Modules (Frequency Based Priorities, Resorting Priority, Selection, and Color Code). Additional libraries provide data organization and presentation. We now describe each component in more detail.

A. Dynamic Table

Search++'s dynamic table provides a spreadsheet-like view of the data. Each table may have up to i rows and j columns. A user may select n row(s) to view the dynamic and child attributes related to the selection. Upon selecting a row, it will be highlighted. The dynamic and child attributes related to selected row(s) will be displayed. Upon deselection of the row(s), the entire list of dynamic and child attributes are displayed. This technique provides an on-demand multiview linked visualization that helps the user learn what dynamic and child attributes make up the related dynamic table entries. Figure 2 show the various states of a dynamic table.

B. Input Box

Search++'s input box is the initial visualization and first point of interaction for the user. This component is common in most search interfaces.

C. Dynamic Attributes

Search++'s dynamic attributes are created from data associated with each of the search results. A dynamic attribute may have n child attribute(s) (e.g., "Feature" being a dynamic attribute having child attributes of: "16 Megapixels", "5x

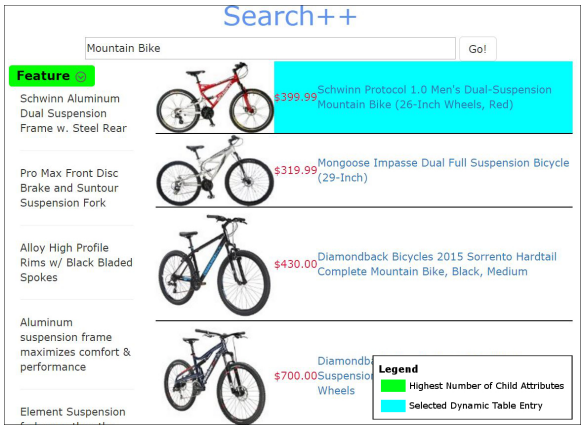


Figure 2. User has clicked on a single dynamic table entry. Only the associated dynamic and child attributes are displayed.

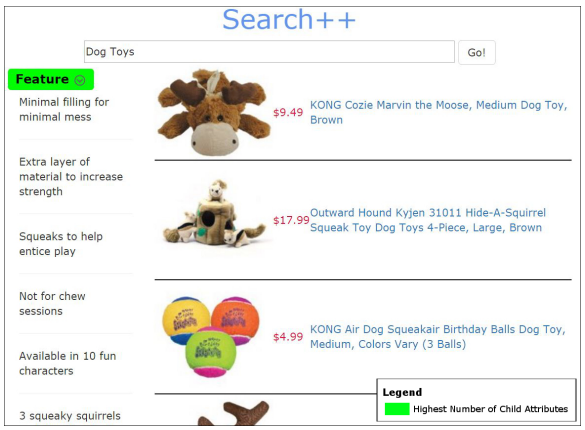


Figure 3. User has expanded dynamic attribute “Feature” and associated child attributes are now displayed.

optical zoom”, “20x digital zoom”, etc.). Figure 3 shows various child attributes.

Search++'s interface allows users to sort dynamic and child attributes according to their priority. Upon sorting, our query processing modules will be engaged (discussed in Section III-F). The dynamic attributes are color coded by their frequency of child attributes (discussed in Section V-D). This allows the user to learn how common an attribute is. Dynamic and child attributes may be hidden when the user is interacting with the dynamic table row(s). This technique allows users to view the dynamic and child attributes related only to their selection.

D. Session State Management

There are three common techniques to store session state: client, server, database [14]. Search++ utilizes server based session state. Each user has one session, which are not shared between users. Sessions allow the server to automatically garbage collect un-utilized data [15]. Session state management in Search++ allows the system to retain critical information that was normalized and processed upon the initial and subsequent execution of the search. This information is referenced upon UI interaction within the same result set. This technique eliminates the need for re-access to the data source.

E. Data Source Plugins

We define a data source plugin as a utility to retrieve and normalize external data requests. Traditional and pure plugin architectures [16] were examined. We chose traditional plugin architecture as it allowed us to retrieve data with less complexity compared than pure. Search++'s data source plugins allows for various API's to be called and normalized to a single result set. This technique allows Search++ to add and configure plugins without impacting the downstream processes. In Section VI, we demonstrate the usefulness of this technique.

F. Query Processing Modules

Search++ has four primary query processing modules (frequency based priority, resorting priority, selection, and color coding). Each module is responsible for a single task (e.g., retrieve data, organize data). This technique allows other modules to be added or removed during run-time without affecting the entire system. Each module is discussed in detail in Section V.

G. Implementation

Search++ is implemented using commonly-used libraries (Microsoft MVC5, Angular-JS, Bootstrap and jQuery-Sortable [17], [18], [19], [20]) for user interactions, design of each component, session management and algorithm runtime. The combination of these technologies is common and has been shown to provide the necessary performance capability when used in conjunction.

IV. UI, INTERACTION, AND WORKFLOW

Search++'s interface displays a single view. This view allows a user to search for criteria and rank attributes according to their priority. Search++'s query processing modules will leverage the attribute priority to obtain a list of prioritized search results. In this section, we describe the workflow that a user needs to carry out in Search++ to accomplish a goal.

A. Initial Search

Upon initial search, Search++ retrieves data to render the visualization. Upon retrieval of data from our data source plugins (Section III-E) Search++'s query processing modules will engage to normalize the data set for visualization. The result set includes dynamic attributes and tables. The dynamic attributes are presented in the default sort order in accordance to the dynamic and child attribute frequency (described in Section V-A). The dynamic table will include the sorted search results based upon the dynamic and child attribute priority (described in Section V-C).

The following is the exchange that takes place between client and server for initial data acquisition.

- Client : HTTP GET request for data
- Server : Receives request
  - Queries data within plugin architecture
  - Performs normalization and engages algorithms
- Client : Receives Data

 ISBN: 978-1-61208-465-7

B. Manipulating Data

Manipulating details within a data set to further derive insight is important. Search++ allows for manipulations within dynamic and child attributes, dynamic table, or re-searching for criteria. Upon any of these manipulations our dynamic table will update (discussed in Section IV-B2). In the following subsections we will discuss each manipulation and the reactions that occur.

1) Dynamic and Child Attributes: A user can manipulate dynamic and child attributes. A dynamic attribute can be reordered amongst other dynamic attributes. A child attribute can be reordered among other child attributes within the particular dynamic attribute. A user can reorder a dynamic or child attribute by selecting and dragging the attribute above or below other attributes. Upon reordering our query processing module is engaged and the dynamic table is updated. Figure 4 demonstrates the result of a user reordering.

2) Dynamic Table: Search++ has a simplistic view for the user to quickly understand what is available. The dynamic table entries may be reordered depending on the user's interaction. If the user changes their search criteria the dynamic table entries will be replaced. To help users understand what dynamic and child attributes are associated with a particular row in the dynamic table, Search++ allows the user to select n row(s). Upon selection of a given row, Search++ displays the dynamic and child attributes related to the selection. If a user deselects all rows previously selected, the dynamic and child attributes related to the entire result set will reappear. Figure 2 demonstrates this action. This creates a multiview linked visualization that helps the user learn what attributes comprise a particular set of entries.

3) New Search: On a new search, Search++ will reinitialize all data to the begin state. Search++'s visualization will update the result set to a similar format of all previous searches. This consistency allows users to be more effective conducting searches as the components become familiar to the user.

V. QUERY PROCESSING MODULES

Search++ has four query processing modules that work together to provide a normalized result set. We will discuss each module in further detail below.

A. Frequency Based Priority

The frequency based priority module is run when the user conducts a new search. The input to this algorithm is the output dynamic and child attributes from the originating data source. The output is a sorted occurrence list of dynamic and child attributes (i.e., attributes that are most common are earlier in the list and tail down to least). Table I shows an example input and output. This algorithm sorts dynamic attributes in descending order according to the dynamic attributes with the greatest number of distinct child attributes. This allows the user to quickly ascertain the attributes with the greatest amount of variation.

B. Resorting Priority

Resorting dynamic and child attributes allows the user to identify search results that most closely align with their ranked ordered priorities. This technique is preferable to that of complex interfaces that filter criteria as result sets are not removed but reordered.

TABLE I. THE FOLLOWING TABLE SHOWS AN EXAMPLE INPUT AND EXPECTED OUTPUT FROM THE FREQUENCY BASED PRIORITY ALGORITHM.

Input	*Output*
Fragile (5)	Feature (30)
Weight (10)	Weight (10)
Feature (30)	Warranty (8)
Warranty (8)	Fragile (5)

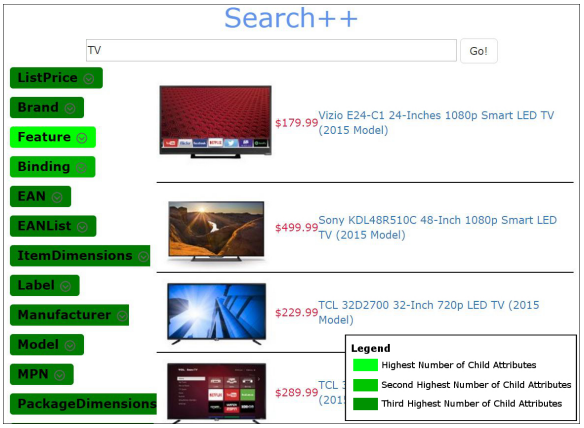


Figure 4. User has prioritized “ListPrice” and “Brand” above “Feature”.

Upon sort of dynamic or child attributes, the Web client will notify the server of change. Search++ has an optimal approach of sending this data, by sending only the unique identifiers of the new order, rather than the organized full data set. With the use of session state management (discussed in Section III-D) we are allowed to quickly manipulate the resorting and allow our selection algorithm to engage.

C. Selection

Search engines typically provide results in a table. Common users have grown accustomed to such an interface. Search++ additionally displays the results as such referred to as a dynamic table (described in Section IV-B2). The selection algorithm is designed to create entries in a table manner. The priority sort order of dynamic and child attributes define the sort order of the dynamic table entries. The selection algorithm maps the dynamic and child attributes to the data source entries to provide the dynamic table.

The selection algorithm first identifies the search results with parent and child attribute pair ranked highest in the priority. These search results are designated to be first on the list in the dynamic table. The selection algorithm will then identify search results that were not used in the first iteration and contain the parent and child attribute pair ranked second in priority. These search results are designated to be second on the list of the dynamic table. This process is repeated until all search results have been identified and designated to be included in the dynamic table. This algorithm will be triggered upon new search creation or user interaction of dynamic or child attributes.

D. Color Code

We color code dynamic attributes with the highest frequency in light green and darken the color of each subsequent

 ISBN: 978-1-61208-465-7

dynamic attribute with lower frequency than the previous. We felt that attributes with the highest frequency of sub-attributes, shows the most likely sorted set for the common user. If two or more dynamic attributes have the same number of child attributes, their color will be identical. We only color these dynamic and child attributes once, upon initial data retrieval.

Each dynamic attribute following the base attribute is shaded using the following calculation: previous dynamic attribute color + (((current dynamic attribute child count - previous dynamic attribute child count) / previous dynamic attribute child count) * base light color).

VI. CASE STUDY

To demonstrate Search++'s functionality and utility, we conducted a user case study. In order to facilitate the case study, a data source plugin for Search++ was developed to leverage AWS as a repository. We asked a group of ten individuals who described themselves as "non-technical", five male, five female, aging from 25 – 62, having highest education degree: graduate (2), undergraduate (4), and high school diploma (4) to search for six unique products they are seeking to purchase; three searches using Amazon's search interface and three using Search++. We compared the number of steps taken by participants to find the desired search result, level of confidence participants had with the search result, and overall experience using each (Amazon, Search++). In the following sections we describe the results and conclusions of this case study.

A. Results

We captured data from ten participants and sixty search attempts; thirty using Amazon's search interface and thirty using Search++. We counted the number of steps each participant took in each search attempt to find the desired search result. A step is defined as any and each manipulation that a participant performs on the view. On average we found each participant took 7.8 steps using Amazon's search interface compared to 7.3 steps using Search++ to find the desired search result. Table II shows summarized results of participant steps. Table IV shows summarized results of Amazon's search interface. Table V shows summarized results of Search++.

In general, we found participants using Search++ required fewer steps compared to Amazon's search interface to find the desired search target. Search++ required fewer steps except for filtering/re-ranking. We believe participants in this study re-ranked items more than necessary as they were experiencing the Search++ technique for the first time. In the next section we will discuss user feedback of this case study.

TABLE II. THE FOLLOWING TABLE SHOWS SUMMARIZED RESULTS OF PARTICIPANT STEPS

	Amazon	*Search++*
Average Number of Steps	7.8	7.37
Median	7	7
Mode	6	7
Standard Deviation	2.98	1.45

At the conclusion of each participant's search, we asked the participant: "How confident do you feel that your selected search result is best result that could be found? Rate 1–5 (1 being the lowest level of confidence and 5 being the highest)". Participant confidence with the selected search result was found to be higher using Search++. Table III shows the summarized results of participant confidence with the selected search result. We believe the Amazon's search interface filters, which omit results from the result set, led participants to have a low level of confidence. In contrast, Search++ re-prioritization technique does not omit results, but re-orders the result set.

TABLE III. THE FOLLOWING TABLE SHOWS USERS SUMMARIZED CONFIDENCE LEVEL

	Amazon	*Search++*
Average Confidence Level	3.53	4.37
Median	4	4
Mode	4	4
Standard Deviation	0.73	0.49

TABLE IV. THE FOLLOWING TABLE SHOWS SUMMARIZED RESULTS OF PARTICIPANT STEPS USING AMAZON'S SEARCH INTERFACE

	Back Button	*New Search Term*	*Filtering*	*Product Detail View*	*Next Page*
Average	0.6	1.37	3.33	1.6	1.33
Median	0.0	1.0	3.0	1.0	1.0
Mode	0.0	1.0	3.0	1.0	1.0
Standard Deviation	0.81	0.56	0.48	0.81	0.55
Count	18	41	100	48	40

TABLE V. THE FOLLOWING TABLE SHOWS SUMMARIZED RESULTS OF PARTICIPANT STEPS USING SEARCH++

	Back Button	*New Search Term*	*Rank Change*	*Product Detail View*	*Next Page*
Average	0.27	1.2	3.93	1.27	1.33
Median	0.0	1.0	4.0	1.0	1.0
Mode	0.0	1.0	4.0	1.0	1.0
Standard Deviation	0.45	0.41	0.64	0.45	0.55
Count	8	36	118	38	40

In addition to the confidence rating, we asked participants: "What was your experience using Search++" and "Would you like to use the Search++ technique in other areas of search on-line?" Eight out of the ten participants responded favorably to their experience using Search++.

Nine out of the ten participants would like to see Search++ in other areas of search on-line. "I like how I can just continuously re-shuffle my results without having to search over and over again."

One more observation: We collected all dynamic and child attributes that were used to present the returned Search++ fields. Table VI shows a summary of the result set. Note that the selection pool of dynamic and child attributes listed in the table provides plenty data to priority-sort search terms. This seems to offer more power to the user than the prevailing filtering approaches, which tend to remove potentially relevant data from the analysis.

B. Conclusion

The case study yielded favorable results for Search++, showing that it is easily adopted by users, can reduce the number of steps to obtain search results, and might be preferred over existing search interfaces. We believe that participants that did not respond favorably to Search++ may have factored

 ISBN: 978-1-61208-465-7

TABLE VI. THE FOLLOWING TABLE SHOWS DYNAMIC AND CHILD ATTRIBUTES RETREIVIED DURING SEARCH++ SEARCHES

	Dynamic	*Child Attributes*
Average	32.8	289.12
Mode	30	288
Min	25	233
Max	51	357
Standard Deviation	5.64	32.37
Total	820	7228

Amazon's other offerings above and beyond search results when making the comparison (e.g., "Customer Reviews", "Customers also bought"). The technique allows users to find results that they may not have known existed using other search interfaces. "I found a school supply package using Search++ including the crayons I was looking for as well as other markers and paper which I assumed I'd have to buy separate, I didn't see this on Amazon's search."

Search++ yielded only slightly better results in terms of steps to find desired search results; 7.3 steps on average compared to that of 7.8 using Amazon however, Search++ rated hire in participant confidence. However, it is worth considering that subjects had had more experience searching with Amazon's search interface than with Search++; it is reasonable to expect that with a little more experience, their performance and satisfaction using Search++ would improve.

VII. CONCLUSIONS

It has been shown that better search tools are needed for better exploration [21]. Oblinger and Oblinger [22] argue that the "Net generation" (those who learned to read after the Web) are qualitatively different in their informational behaviors and expectations; they multitask and expect their informational resources to be electronic and dynamic. The Net generation expects to be able to use Web resources to lookup, learn, and investigate tasks with fluid user interfaces. To the best of our research, even now, ten years after that observation was made, search tools still cannot meet those expectations.

In this paper, we have introduced Search++, a technique that allows users to easily execute a search against one or many sources, identify the attributes that may be most important to them, prioritize attributes according to importance, and find the search results that most closely align with their priorities. We have demonstrated that additional search options are needed. We have shown various ways to explore these integrated results within the interactive visualization paradigm, by the three UI components described in Section III (Dynamic Table, Input Box, and Dynamic Attributes). We have provided methods to efficiently organize these results without interfering with cognitive workflow.

We provided a case study that explored the value in Search++, by asking ten "non-technical" individuals to search for six unique products they are seeking to purchase; three searches using Amazon's search interface and three using Search++. We found that Search++ offered a slight improvement over Amazon's search interface. Our research was limited with our case study of ten users. We plan on advancing Search++ by introducing a collaboration feature, which will allow multiple searchers to work towards a a common goal. Search++ may be adapted to solve other decision based real world problems, by taking into considerations features inherent to each problem. (For example, consider the problem of seeking a new job, and imagine the various job-related features that might impact the search.)

REFERENCES

[1] B. Shneiderman, D. Byrd, and W. B. Croft, "Clarifying search: A user-interface framework for text searches," D-lib magazine, vol. 3, no. 1, 1997, pp. 18–20.

[2] C. C. Kuhlthau, "Inside the search process: Information seeking from the user's perspective," JASIS, vol. 42, no. 5, 1991, pp. 361–371.

[3] M. J. Bates, "Information search tactics," Journal of the American Society for information Science, vol. 30, no. 4, 1979, pp. 205–214.

[4] A. Halavais, Search engine society. John Wiley & Sons, 2013.

[5] M. W. Newman and J. A. Landay, "Sitemaps, storyboards, and specifications: a sketch of web site design practice," in Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, 2000, pp. 263–274.

[6] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger, "The perfect search engine is not enough: a study of orienteering behavior in directed search," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2004, pp. 415–422.

[7] K. Naz and H. Helen, "Color-emotion associations: Past experience and personal preference," in AIC 2004 Color and Paints, Interim Meeting of the International Color Association, Proceedings, vol. 5. Jose Luis Caivano, 2004, p. 31.

[8] E. Amazon, "Amazon elastic compute cloud (amazon ec2)," Amazon Elastic Compute Cloud (Amazon EC2), 2010.

[9] J. Y. Kim, M. Cramer, J. Teevan, and D. Lagun, "Understanding how people interact with web search results that change in real-time using implicit feedback," in Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013, pp. 2321–2326.

[10] H. He, W. Meng, C. Yu, and Z. Wu, "Wise-integrator: An automatic integrator of web search interfaces for e-commerce," in Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, 2003, pp. 357–368.

[11] Q. Peng, W. Meng, H. He, and C. Yu, "Clustering e-commerce search engines," in Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. ACM, 2004, pp. 416–417.

[12] M. Wilson, A. Russell, D. A. Smith et al., "mspace: improving information access to multimedia domains with multimodal exploratory search," Communications of the ACM, vol. 49, no. 4, 2006, pp. 47–49.

[13] J. Zhang and G. Marchionini, "Evaluation and evolution of a browse and search interface: Relation browser++," in Proceedings of the 2005 national conference on Digital government research. Digital Government Society of North America, 2005, pp. 179–188.

[14] M. Fowler, Patterns of enterprise application architecture. Addison-Wesley Longman Publishing Co., Inc., 2002.

[15] B. C. Ling, E. Kiciman, and A. Fox, "Session state: Beyond soft state." in NSDI, vol. 4, 2004, pp. 22–22.

[16] D. Birsan, "On plug-ins and extensible architectures," Queue, vol. 3, no. 2, 2005, pp. 40–46.

[17] A. Leff and J. T. Rayfield, "Web-application development using the model/view/controller design pattern," in Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International. IEEE, 2001, pp. 118–127.

[18] N. Jain, P. Mangal, and D. Mehta, "Angularjs: A modern mvc framework in javascript," Journal of Global Research in Computer Science, vol. 5, no. 12, 2015, pp. 17–23.

[19] M. Otto and J. Thornton, "Bootstrap," Twitter Bootstrap, 2013.

[20] B. Bibeault and Y. Kats, jQuery in Action. Dreamtech Press, 2008.

[21] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and T. L. Madden, "Ncbi blast: a better web interface," Nucleic acids research, vol. 36, no. suppl 2, 2008, pp. W5–W9.

[22] D. Oblinger, J. L. Oblinger, and J. K. Lippincott, Educating the net generation. Boulder, Colo.: EDUCAUSE, c2005. 1 v.(various pagings): illustrations., 2005.

 ISBN: 978-1-61208-465-7

Assessing the Suitability of Architectural Patterns for Use in Agile Software Development

Samira Seifi Jegarkandy, Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
e-mail: seifi@ce.sharif.edu, ramsin@sharif.edu

***Abstract*—The software industry is moving towards agile software development methods, as they accommodate rapidly changing requirements, and cope remarkably well with modern challenges of software development. On the other hand, it has long been recognized that software architecture has a major impact on the maintainability, scalability, and quality assurance of software systems, so much so that it is virtually impossible to produce high-quality software systems (which are inherently complex) without architectural design. Agile methodologies use lightweight architectural practices, and applying architectural patterns is a common practice in agile development. However, to this date, there has been no comprehensive study on the suitability of existing architectural patterns for agile development. We introduce a set of criteria for assessing the suitability of architectural patterns for use in agile approaches, and evaluate a set of prominent architectural patterns based on these criteria. Agile developers can use the results of this evaluation to assess the suitability of each pattern for application in their agile development projects.**

Keywords-agile software development; software architecture; software pattern; architectural pattern; criteria-based evaluation

I. INTRODUCTION

Agile software development methodologies have been gaining in popularity, among industry practitioners and researchers alike, since they emphasize rapid and flexible development [1]. On the other hand, software architecture, as a discipline, deals with modeling and managing a software system's structure, project blueprint, and communications among stakeholders, which are essential for achieving quality attributes such as usability and maintainability [2]. Architectural design has always been an important issue in agile approaches, even though these approaches have strived to keep their architectural design tasks as lean as possible [3]. It has been suggested that agile methods' support for architectural activities should be enhanced even further [4].

In software engineering, an architectural pattern is a structured description of a reusable coarse-grained architectural *solution* to a commonly occurring *problem* within a given *context* [5]. Architectural patterns mainly target the non-functional requirements of the product, and provide an overall structure for the target system in order to address these requirements [2]. Architectural patterns are widely used today in all development approaches, including agile development, for structuring software systems. Although several approaches have been proposed for applying architectural patterns in agile development (such as [6]-[9]), there is currently no comprehensive review of these patterns to assist developers in selecting agile-friendly patterns. This paper focuses on evaluating the suitability of architectural patterns for application in agile development. To this aim, we propose a criteria-based evaluation approach. The evaluation criteria used in our approach have been elicited from the Agile Manifesto and Principles [10] and the CEFAM evaluation framework [11]. We have used these criteria to evaluate existing architectural patterns; due to space limitations, however, only the patterns that are most prominent and relevant will be focused upon in this paper.

The evaluation results obtained in our research (reported herein) can be leveraged to identify a set of agile-friendly architectural patterns. Thus, our proposed set of criteria provides a valuable framework for assessing the suitability of architectural patterns for use in agile development, and also for warning against the use of architectural patterns that are not particularly suitable for agile development. Another potential benefit of our proposed criteria is the applicability of the evaluation results for improving architectural practices in agile approaches; this has indeed been our ultimate intention in this research: we intend to combine architectural patterns and agile methodologies in order to address architectural issues in agile development without any adverse effect on agility. Even if an agile method puts sufficient emphasis on software architecture, misusing the method can cause architectural problems; the individual criteria and their respective evaluation results can potentially alleviate this problem by helping to determine the appropriate time and situation for applying each pattern in an agile context, thus enabling the developer to address architectural design issues.

The rest of this paper is organized as follows: Section II reviews the architectural patterns used in agile development; Section III introduces the proposed evaluation criteria, based on which the evaluation results are presented in Section IV; and Section V presents the concluding remarks and discusses possible directions for furthering this research.

II. ARCHITECTURAL PATTERNS AND AGILE DEVELOPMENT

In this section, we will provide an overview of architectural patterns and their use in agile development.

 ISBN: 978-1-61208-465-7

A. Review of Architectural Patterns

Patterns have been defined for many different areas of software development. However, our focus here is strictly on patterns for software architecture, which are commonly used for shaping the high-level structure of a software system. The terms architectural *style* and architectural *pattern* are widely used for describing these reusable structures [2]. In this subsection, brief descriptions will be provided for major architectural patterns; it should be noted that in cases where there are several variants for a pattern, the variant that is more widely used has been included, even if it is an older (earlier) variant. In later sections, we will assess these patterns as to their applicability in agile development.

We have categorized the architectural patterns according to their application areas in order to better manage the complexity of the spectrum of patterns under review. Even though some of the patterns are usually categorized as design patterns, we have included them herein as they can also be used for solving architectural problems, e.g., the Decorator pattern [12] can be used for creating dynamically configured chains of subsystems, and can thereby offer a solution at the architectural level. Some patterns address several application areas; in such cases, one of these areas has been designated as the main application area. All such patterns have been categorized based on their main application areas**,** e.g., we have assigned the Architecture with Component-as-a-Service (CaaS) pattern [13] to the Mobile Software Development category, even though it belongs to the Distributed Systems Development category as well. The patterns have been briefly described in Tables I and II; these tables also show the main category to which each pattern belongs.

There are six pattern categories, as explained below:

- *Patterns for Mobile Software Development:* Various architectural patterns yielding different levels of qualities, especially as to performance and energy consumption, which are used for developing mobile systems and applications (shown in Table I).
- *Patterns for Cloud Systems Development:* Patterns for using cloud-platform services (shown in Table I).
- *Security Patterns:* Patterns that provide a high level of security (shown in Table II).
- *Patterns for Distributed Systems Development:* Patterns that define how distributed components collaborate with each other (shown in Table II).
- *Patterns for Agent-Oriented Systems Development:* These include patterns for developing systems in agent-oriented contexts (shown in Table II).
- *General-Context Patterns:* General patterns that do not belong to a specific application area or development context (shown in Table II).

TABLE I. ARCHITECTURAL PATTERNS FOR MOBILE SOFTWARE DEVELOPMENT AND CLOUD SYSTEMS DEVELOPMENT

Pattern Category	Name	Brief Description
Mobile Software Development	**Architectural Pattern for Mobile Groupware Platforms [14]**	Used for developing groupware platforms, providing separate functionality for three basic concerns: Collaboration, Communication, and Coordination. It divides systems into three separate layers: the collaboration layer consists of mobile groupware applications; the communication layer handles messages interchanged among mobile units; and the coordination layer provides the services required by applications to coordinate their operations on shared resources.
	Balanced MVC Architecture [15]	Used for supporting service-based mobile applications. This pattern is an extended Model View Controller (MVC) architecture where client and server systems embody the MVC pattern.
	External Customizer [16]	Focuses on adapting web content to mobile clients by creating a component that converts data from arbitrary mobile web information systems to a suitable format for potential clients.
	Internal Customizer [16]	Removes the need for external customization by providing the client with a response directly suitable for its manipulation. This pattern uses customization mechanisms in the design of mobile web information systems.
	Web Channel Broker [16]	Extends the Broker pattern with the capability to interact with the whole web while presenting only a subset of it.
	Application with External User Interface (UI) Elements [17]	Represents interactive applications with physically separated UI components. This pattern is an adaptation and extension of the MVC approach.
	Standalone Mobile Application [13]	Runs the entire expected functionality on a mobile device without referring to any external services or servers [13].
	Mobile Application with Full Offloading [13]	Offloads the whole application and its associated database to a server. The mobile device just transmits the required dataset to the server, and is not involved in any computations.
	Mobile Application with Partial Offloading [13]	Offloads parts of the application and the dataset to an external server.
	Architecture with Software-as-a-Service (SaaS) [13]	In this pattern, the client incorporates a simple web browser or dedicated client program, while all the functionality required is fulfilled by external services.
	Architecture with Component-as-a-Service (CaaS) [13]	Provides cloud services as finer-grained units of common and reusable functionality.
	CaaS-Based Architecture with Offloading [13]	Divides the required functionality into three parts: one part is fulfilled by the CaaS architecture, one part is offloaded to a dedicated server, and the remaining part is implemented in the client application.
	Extended MVC [18]	Extends the common MVC pattern with additional components and adaptations specifically intended for mobile application development.
Cloud Systems Development	**Cloud Policy Management Point [19]**	Controls security functions, including authentication, authorization, cryptography, and control of virtual machine images.
	Eventually-Consistent User Interface [20]	Ensures the eventual consistency of the user interface in cases where it is not possible (or desirable) to ensure the consistency of the cloud data stores that are used by the user interface.
	Loose Coupling [21]	Reduces dependencies among distributed applications and their components by using Brokers.
	Service Level Agreements Compliance Checking [22]	Provides a three-layered architecture for distinguishing probe concerns from monitoring data collection concerns according to the SLAs.

 ISBN: 978-1-61208-465-7

TABLE II. ARCHITECTURAL PATTERNS RELATED TO SECURITY, DISTRIBUTED SYSTEMS DEVELOPMENT, AGENT-ORIENTED SYSTEMS DEVELOPMENT, AND GENERAL CONTEXT

Pattern Category	Name	Brief Description
Security	**Secure MVC [23]**	Shows how several fundamental security patterns must be applied to MVC components in order to provide secure access/modification of the information residing in the Model component.
Distributed Systems Development	**Component-Based Architectural Style [24]**	Decomposes the application into reusable components (functional or logical), which are location-transparent and expose well-defined communication interfaces.
	Service-Oriented Architectural Style [24]	In this style, some of the components provide services to other components.
	Event-Based Integration Style [25]	Removes the need for identities on the connector interfaces in order to reduce the coupling among components. This style is also known as the Implicit Invocation or Event System style.
	Client/Server Architectural Style [24]	Provides distributed systems consisting of separate clients and servers and a connecting network. There are variants such as Client-Queue-Client and Peer-to-Peer (P2P).
	Distributed Publish/Subscribe [26]	Decouples the publishers of events from those interested in them.
	Enterprise Service BUS [26]	Integrates a variety of distributed services and related components. Within this architectural pattern, various components connect to a service bus via their service interfaces.
	Broker [5]	Achieves better decoupling of clients and servers through providing indirect, location-transparent access to distributed services by handling message calls to the appropriate objects.
	A-3 Style [27]	Defines a structure for coordinating distributed components. This style adopts the concept of "group" as an abstraction for organizing an application into semi-independent slices, providing a single and coherent view of these aggregates, and coordinating them.
AO Systems Development	**Layered Agent [28]**	Provides a structure for supporting the behavior of agents. This pattern decomposes agents into layers. All agents do not have the same layers.
	AO-Broker [28]	This pattern is a special kind of Broker specifically customized and extended for use in agent-oriented systems [28].
	Presentation-Abstraction-Control (PAC) [5]	Provides a structure for interactive systems. This pattern defines the system as a set of cooperating agents, each of which is responsible for a specific aspect of the system's functionality.
General Context	**Model-View-Controller (MVC) [5]**	Divides an interactive system into three interconnected, highly specialized, and loosely coupled components: Model, Views and Controllers. The model component encapsulates core data and functionality, view components display information to the user, and controllers handle user input.
	Model-View-Controller-Context (MVCC) [29]	An extension of the MVC pattern that also incorporates a context component, which is solely responsible for handling context-awareness concerns.
	Zone [30]	Provides flexibility in changing the logical and physical architecture of the processing unit, and the resources the processing units need to accomplish their tasks.
	Microkernel [5]	Provides mechanisms for extending the software system with additional and/or customer-specific functionality, thus making systems adaptable and extensible. In this pattern, the most important core services of the system are encapsulated in a microkernel component.
	Reflection [5]	Supports extension of applications and their adaptation to evolving technology and changing functional requirements. This pattern splits the system into two levels: a base level defines the application logic, and a meta level makes the software self-aware by providing information on its essential features.
	Façade [12]	Provides a unified interface to a complex subsystem. This pattern shields the components of a subsystem from direct access by their clients.
	Blackboard [5]	Useful for combining patchy knowledge to arrive at solutions, even if they are sub-optimal or not guaranteed. This pattern tackles problems that do not have any deterministic solution strategies.
	Component-Based Framework [31]	Used for developing component-based systems. This pattern provides a mixture of fixed and flexible elements that maximize the scalability and extensibility of systems.
	Configured Handler Method [32]	Performs event handling by using metadata, thus avoiding proliferation of empty methods or reduction in class cohesion. Also known as Metadata-based API and Metadata-based Invoker,
	Layers [5]	Divides the system into distinct layers where each layer is at a particular level of abstraction and handles a specific concern of the system.
	Pipes and Filters [5]	Used for processing date streams. This pattern divides the tasks of a system into several sequential processing steps (filters) that form a pipeline. Data is passed between adjacent filters through pipes.
	Adapter [12]	Used for translating calls between two different interfaces. This pattern converts the interface of a class into the interface that clients expect.
	Decorator [12]	Additional responsibilities can be dynamically attached to an object by using this pattern.
	Command [12]	Encapsulates a request as an object, thereby letting users parameterize clients with different requests, queue or log requests, and support undoable operations [12].
	Command Processor [5]	Complements the Command pattern [12] by addressing the management and scheduling of Command objects.
	View Handler [5]	Manages all the UI views that are provided by the system.

B. Using Architectural Patterns in Agile Development

Architectural concerns have always been addressed in agile development methodologies; the system "metaphor" used in XP is a prominent example [1]. However, there is a growing interest in further extending agile methods with architectural approaches [3][4]. One way to improve architectural design in agile software development is to use architectural patterns. These patterns should make architectural tasks more agile or add architectural tasks to

agile activities [7]. Research is ongoing on combining agile development with architectural patterns in order to improve agile processes; MASAM [8] and Mobile-D [9] are two such methods that use architectural patterns in agile approaches in order to capture architectural knowledge. References [33]-[35] provide examples of the use of architectural patterns in agile software development; these patterns have reportedly improved the quality of the systems produced, and have provided a holistic view of the system, without which agile projects could face serious impediments.

III. PROPOSED EVALUATION CRITERIA FOR ASSESSING SUITABILITY OF ARCHITECTURAL PATTERNS FOR AGILE DEVELOPMENT

As mentioned before, the suitability assessment proposed herein is based on suitability criteria. To this aim, we propose a special set of qualitative criteria based on the agile traits outlined in the Agile Manifesto and Principles [10], and the CEFAM Framework for evaluating agile methodologies [11]. This approach is based on the rather obvious observation that a pattern that violates these characteristics and damages agility cannot be used in agile development.

Our proposed set of evaluation criteria will be introduced throughout the rest of this section. The criteria, listed in Table III, are divided into two categories according to the type of evaluation results obtained through applying them:

- *Simple form:* The evaluation results for these criteria are of the "Yes/No" type, denoting the satisfaction or non-satisfaction of the criterion. "Need for formalism" is the only criterion in this category.
- *Scale form (multilevel):* The result of applying a Scale-form criterion is selected from among a number of predefined discrete levels. The levels are numbered in descending order of satisfaction; in other words, level 1 signifies the highest degree of satisfaction of the criterion. To provide a more precise evaluation, two of the criteria ("Reusability" and "Complexity Management) have been further divided into finer-grained sub-criteria.

In order to show the validity of the proposed criteria for their ultimate purpose (i.e., assessment of agile-friendliness), Table III also depicts the Agile Principles [10] that underlie (are addressed by) each and every criterion. The proposed criteria have also been assessed based on the validity metacriteria of [36]; this assessment shows that the proposed criteria are valid in that they are: 1) *General* enough to be used for evaluating all architectural patterns as to their suitability for application in agile development; 2) *Precise* enough to help discern and highlight the similarities and differences among architectural patterns as to their agile-friendliness; and 3) *Comprehensive* enough to cover all significant features of architectural patterns as pertaining to their suitability for application in agile development.

IV. SUITABILITY OF ARCHITECTURAL PATTERNS FOR USE IN AGILE DEVELOPMENT: EVALUATION RESULTS

In this section, we provide the results of assessing the reviewed architectural patterns based on the proposed criteria. The evaluation results, as assessed by the authors, are presented in Table IV.

Assessing the overall suitability of an architectural pattern for use in agile development can be a matter of opinion, as many patterns are strong in some of the criteria, but weak in others. Deciding the overall suitability of a pattern is therefore subjective, and depends on the priority of the criteria in the mind of the assessor. The last column of Table IV shows the overall suitability of each pattern as judged by the authors: "✓" denotes "Overall Suitable", and "×" signifies "Overall Unsuitable". In our opinion, the criteria and sub-criteria pertaining to "Reusability", "Decomposability", and "Complexity Management" are more important than others in assessing the overall agile-friendliness of the patterns. We have therefore given more weight to these criteria when giving our final verdicts.

To better understand the nature of the assessments made in this section, Table V illustrates how the proposed criteria have been used for assessing the MVC architectural pattern. Overall, based on the evaluation results, the MVC architectural pattern has been deemed as a suitable pattern for use in agile software development.

The selection of the appropriate pattern depends on the results of applying the whole set of criteria, not just one criterion; this means that the final evaluation result might be the same for all the assessors. Yet, even if the assessors do not concur on the final result, the evaluations are still valuable in that they help identify the strengths and weaknesses of the patterns under review; this knowledge can be used for comparing alternative patterns and improving the use of the patterns in agile methods. As an example, consider comparing MVC to Layers. Comparing the evaluation results shows that MVC fares better than Layers in most of the criteria; therefore, if these criteria are deemed crucial in a project, MVC would be preferable to Layers in that particular project situation.

V. CONCLUSION AND FUTURE WORK

The software industry is becoming increasingly keen on using agile methodologies to achieve rapid and flexible development. On the other hand, software architecture has evolved into a vast, essential discipline in software engineering; architectural design has become indispensable, especially when reusable, distributed, and maintainable software systems are required. Agile methodologies are in need of improvement as to their support for architectural design, and architectural patterns seem to be a promising means for addressing this need. This has been our ultimate goal in this research: to use architectural patterns for enhancing architectural design in agile methodologies.

As the first step towards this goal, we have evaluated existing architectural patterns as to their suitability for use in agile development. A set of qualitative criteria have been defined for evaluating existing methodologies. The results of criteria-based evaluation reveal that not every architectural pattern is suitable for use in an agile context; therefore, if an application requires the use of an architectural pattern that has been deemed as agile-unfriendly, using an agile approach for its development would be considered risky (at best).

 ISBN: 978-1-61208-465-7

TABLE III. PROPOSED EVALUATION CRITERIA FOR ASSESSING THE SUITABILITY OF ARCHITECTURAL PATTERNS FOR AGILE DEVELOPMENT

Criterion		Description	Possible values	Underlying agile principles*
Reusability	**Encapsulation and abstraction**	How does the pattern promote abstraction and encapsulation? Abstract modules are more reusable by nature, and encapsulation enhances reusability by setting up barriers among modules and reducing coupling [5].	1: At the level of components/classes; 2: Only at the level of subsystems/layers; 3: Addressed implicitly; 4: Not addressed, but not adversely affecting reusability; 5: Reusability reduced due to violation of encapsulation or lack of abstraction.	CR; CS; FD; FWS; GD; S
Reusability	**Separation of concerns and high cohesion**	How does the pattern support separation of concerns and promote high cohesion in modules? Separation of concerns and high cohesion go hand in hand, and enhance reusability by encouraging non-complex, specialized modules.	1: At the level of components/classes; 2: Only at the level of subsystems/layers; 3: Addressed implicitly; 4: Not addressed, but not adversely affecting reusability; 5: Reusability reduced due to clustering of functionality and execution of non-related work at the level of layers/components.	CP; CR; FD; GD; R; TP
Decomposability		How is the structure decomposed by the pattern so that each individual piece is small enough to be developed in an agile manner?	1: Explicitly addressed for the entire system; 2: Explicitly addressed for part of the system; 3: Addressed implicitly; 4: Not addressed, but not adversely affecting decomposability; 5: Decomposability is adversely affected by the pattern.	CP; CR; CS; CW; FD; FWS; R
Complexity Management	**Coupling and change propagation**	How does the pattern promote low coupling and prevent the propagation of change?	1: At the level of components/classes; 2: Only at the level of subsystems/layers; 3: Addressed implicitly; 4: Not addressed; 5: Coupling and change propagation is adversely affected by the pattern.	CP; CR; CS; FD; GD
Complexity Management	**Modularity**	How does the pattern provide a meaningful decomposition of the software system into subsystems and components? How does the pattern indicate how to physically package the entities that form the logical structure of the system?	1: At the level of components; 2: Only at the level of subsystems/layers; 3: Addressed implicitly; 4: Not addressed.	CR; R; S; TP
Hiding of implementation details		Does the pattern hide implementation details?	1: Only provides the overall system architecture; 2: Shows class structure; 3: Shows the classes and interfaces needed to create the elements; 4: Implicitly implies implementation details; 5: Explicitly states implementation details.	FTFC; SOT; TP
Removal of extra/duplicated work		Does the pattern pay special attention to removing extra/duplicated parts, thereby enhancing the simplicity of the design and avoiding unnecessary development work?	1: Addressed; 2: Addressed, but needs extra effort when applying the pattern; 3: Addressed implicitly; 4: Not addressed, but not adversely affecting simplicity; 5: Extra/duplicated work is introduced by the pattern itself.	CS; FD; FWS; GD; S
Application costs		Are the time, cost, and effort required for applying the pattern justifiable?	Application costs are: 1: Lower than the "before" state; 2: Reasonable; 3: Acceptable, because the pattern solves important problems; 4: High, because the problems solved are not important.	CS
Explicit definition		Does the pattern define the architectural solution (structure of the system and the relationships among its constituent elements) in a detailed and explicit fashion?	1: Explicit definitions of structure and relationships are provided; 2: Explicit definition of structure and implicit definition of relationships are provided; 3: Implicit definition of structure and explicit definition of relationships are provided; 4: Implicit definitions of structure and relationships are provided; 5: Not addressed.	CW; R
Need for modeling		Does applying the pattern require modeling (analysis/design)?	1: The modeling required can be supported by all agile methodologies (e.g., in a "metaphor" document); 2: The modeling required can be supported by some (but not all) agile methodologies; 3: The modeling required cannot be supported by agile methodologies, as it can have an adverse impact on agility.	CS; FTFC; FWS
Need for Formalism		Does applying the pattern require formalism? If yes, the pattern is not recommended for use in agile development.	"Y": Yes; "N": NO.	CR; FTFC; FWS

Legend:

* CP: Consistent Pace; CR: Changing Requirements; CS: Customer Satisfaction; CW: Collaborative Work; FD: Frequent Delivery; FTFC: Face-to-Face Conversation; FWS: Focus on Working Software; GD: Good Design; R: Reflection; S: Simplicity; SOT: Self-Organizing Teams; TP: Trust in People.

 ISBN: 978-1-61208-465-7

TABLE IV. EVALUATION RESULTS

Pattern		Encapsulation and Abstractness	Separation of Concerns, and High Cohesion	Decomposability	Coupling, and Change Propagation	Modularity	Hiding of implementation Details	Removal of Extra/Duplicated Work	Application Costs	Explicit Definition	Need for Modeling	Need for Formalism	Overall Suitability
Mobile Software	Architectural Pattern for Mobile Groupware Platforms [14]	2	2	3	2	2	2	2	2	1	3	N	×
	Balanced MVC Architecture [15]	1	1	2	2	1	1	1	2	1	1	N	✓
	External Customizer [16]	1	2	3	1	2	2	2	3	2	2	N	✓
	Internal Customizer [16]	5	4	4	4	3	1	4	3	2	2	N	×
	Web Channel Broker [16]	1	1	3	2	2	1	3	2	1	2	N	✓
	Application with External User Interface Elements[17]	2	1	2	2	1	1	2	2	1	2	N	✓
	Standalone Mobile Applications [13]	4	4	4	4	4	1	4	2	3	1	N	×
	Mobile Application with Full Offloading [13]	2	2	4	4	2	1	1	2	3	1	N	×
	Mobile Application with Partial Offloading [13]	2	2	2	2	2	1	2	2	3	1	N	✓
	SaaS [13]	1	1	2	2	2	1	1	1	3	1	N	✓
	CaaS [13]	2	2	2	2	2	1	1	1	3	1	N	✓
	CaaS-Based Architecture with Offloading [13]	2	2	2	3	2	1	1	3	3	3	N	×
	Extended MVC [18]	1	1	2	2	1	2	2	2	1	1	N	✓
Cloud Systems	Cloud Policy Management Point [19]	1	1	1	2	2	2	3	3	2	3	N	✓
	Eventually-Consistent User Interface [20]	3	3	4	4	4	4	3	2	4	1	N	×
	Loose Coupling [21]	1	1	2	1	2	1	3	3	1	1	N	✓
	SLA Compliance Checking [22]	2	2	3	2	3	1	2	2	3	2	N	×
Security	Secure MVC [23]	1	1	2	2	1	2	2	3	1	2	N	✓
Distributed Systems	Component-Based Architectural Style [24]	1	1	1	1	1	1	2	2	1	1	N	✓
	Service-Oriented Architectural Style [24]	1	1	1	1	2	1	1	1	1	1	N	✓
	Event-Based Integration [25]	1	2	1	1	1	1	2	3	1	2	N	✓
	Client/Server Architectural Style [24]	2	2	2	2	2	1	2	2	1	1	N	✓
	Distributed Publish/Subscribe [26]	2	2	2	2	2	1	2	2	1	1	N	✓
	Enterprise Service Bus [26]	2	1	1	1	2	1	2	2	1	1	N	✓
	Broker [5]	2	1	2	1	2	1	1	1	1	1	N	✓
	A-3 style [25]	2	2	2	2	1	2	3	2	1	2	N	×
AO Systems	Layered Agent [28]	1	1	1	2	1	1	2	2	1	1	N	✓
	AO-Broker [28]	1	1	2	1	1	1	1	1	1	1	N	✓
	PAC [5]	1	1	1	1	1	1	2	2	1	2	N	✓
General Context	MVC [5]	1	1	2	2	1	1	2	2	1	1	N	✓
	MVCC [29]	1	1	2	2	1	1	2	2	1	1	N	✓
	Zone [30]	1	2	2	2	2	1	2	2	1	3	N	✓
	Microkernel [5]	1	2	1	1	1	1	1	2	1	1	N	✓
	Reflection [5]	3	2	3	2	3	1	3	3	4	3	N	×
	Façade [12]	2	1	3	1	2	1	2	1	1	1	N	✓
	Blackboard [5]	3	2	5	3	2	1	3	4	2	2	N	×
	Component-Based Framework [31]	1	1	2	1	2	1	1	2	1	2	N	✓
	Configured Handler Method [32]	2	2	2	2	3	2	3	2	1	2	N	×
	Layers [5]	2	2	3	2	2	1	2	1	3	1	N	✓
	Pipes and Filters [5]	2	1	1	1	1	1	2	2	1	2	N	✓
	Adapter [12]	2	2	3	2	3	2	2	2	1	1	N	✓
	Decorator [12]	2	2	2	2	2	2	1	2	1	1	N	✓
	Command [12]	1	1	2	1	1	3	2	3	1	1	N	✓
	Command Processor [5]	1	1	2	2	1	3	2	3	1	1	N	✓
	View Handler [5]	1	1	2	2	1	2	2	2	1	2	N	✓

We have based our proposed evaluation approach on the Agile Manifesto and Agile Principles in order to ensure that agility requirements are properly and comprehensively addressed by the proposed criteria. The proposed criteria have also been validated through the application of assessment metacriteria.

Future research can focus on using the patterns that have been deemed as agile-friendly to improve specific agile software development methodologies. Architectural patterns can also be empirically evaluated by application to real-world development projects, the results of which can be used for enriching the results of criteria-based evaluation.

 ISBN: 978-1-61208-465-7

TABLE V. DETAILED EXPLANATIONS FOR THE EVALUATION RESULTS OF THE MVC PATTERN

Criterion	Description	Value
Encapsulation and abstraction	The Model, View and Controller components defined in MVC are encapsulated and abstract.	1
Separation of concerns, and high cohesion	MVC separates the business logic from the presentation logic, so it supports separation of concerns. Constituent components are highly specialized and cohesive.	1
Decomposability	Model, View, and Controller components can be developed in different releases; but MVC is silent as to further decomposition of these components, especially the Model component.	2
Coupling and change propagation	MVC decouples the Model from Views and Controllers, so changes in the UI do not propagate to the system's core functionality (implemented in the Model). However, Views and Controllers are tightly coupled.	2
Modularity	MVC provides modularity by decomposing the system into Model, View and Controller components.	1
Hiding of implementation details	MVC is silent as to implementation, and just defines the overall architecture of the system.	1
Removal of extra/duplicated work	The system structure defined by MVC implicitly removes duplications and extra parts, and thereby precludes extra/duplicated development work.	2
Application costs	The large number of updates and runtime components increases the cost; however, this is controllable, and the cost of applying the pattern can be considered as reasonable.	2
Explicit definition	MVC provides explicit and detailed definitions for the system's main components and the relationships among them.	1
Need for modeling	MVC can be modeled in a "metaphor".	1
Need for Formalism	No Formalism is required.	N

Another strand of research can focus on defining detailed quantitative criteria for assessing the suitability of architectural patterns for use in agile development. This would enable developers to obtain a more rigorous assessment of architectural patterns.

REFERENCES

[1] R. Ramsin and R. F. Paige, "Process-centered review of object oriented software development methodologies," ACM Computing Surveys, vol. 40, no. 1, February 2008, pp. 1–89, doi:10.1145/1322432.1322435.

[2] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 2nd ed. Addison-Wesley, 2003.

[3] S. Ramakrishnan, "On integrating architecture design into engineering agile software systems," Proc. Informing Science and IT Education Conference, June 2010, pp. 9–25.

[4] H.P. Breivold, D. Sundmark, P. Wallin, and S. Larsson, "What does research say about agile and architecture?" Proc. 15th International Conference on Software Engineering Advances, August 2010, pp. 32–37, doi:10.1109/ICSEA.2010.12.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns, vol. 1. Wiley, 1996.

[6] N. Harrison and P. Avgeriou, "Pattern-based architecture reviews," IEEE Software, vol. 28, no. 6, November 2011, pp. 66–71, doi:10.1109/MS.2010.156 .

[7] C. G. Álvarez, Overcoming the Limitations of Agile Software Development and Software Architecture. Master's Thesis, Blekinge Institute of Technology, September 2013.

[8] Y. Jeong, J. Lee, and G. Shin, "Development process of mobile application SW based on agile methodology," Proc. 10th International Conference on Advanced Communication Technology, February 2008, pp. 362–366, doi:10.1109/ICACT.2008.4493779.

[9] P. Abrahamsson, et al., "Mobile-D: An agile approach for mobile application development," Proc. 19th Conference on Object-Oriented Programming Systems, Languages, and Applications, October 2004, pp. 174–175, doi:10.1145/1028664.1028736.

[10] K. Beck, et al., "Manifesto for agile software development," Available online at http://www.agilemanifesto.org [retrieved: January, 2016].

[11] M. Taromirad and R. Ramsin, "CEFAM: Comprehensive evaluation framework for agile methodologies," Proc. 32nd Software Engineering Workshop, October 2008, pp. 195–204, doi:10.1109/SEW.2008.19.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[13] J. Kim, "Architectural patterns for service-based mobile applications," Proc. International Conference on Service-Oriented Computing and Applications, December 2010, pp. 1–4, doi:10.1109/SOCA.2010.5707181.

[14] A. Neyem, S. F. Ochoa, J. A. Pino, and D. Franco, "An architectural pattern for mobile groupware platforms," Proc. On the Move to Meaningful Internet Systems Workshops, November 2009, pp. 401–410, doi:10.1007/978-3-642-05290-3_52.

[15] H. J. La and S. D. Kim, "Balanced MVC architecture for developing service-based mobile applications," Proc. 7th International Conference on E-Business Engineering, November 2010, pp. 292–299, doi:10.1109/ICEBE.2010.70.

[16] W. A. Risi and G. Rossi, "An architectural pattern catalogue for mobile web information systems," International Journal of Mobile Communications, vol. 2, no. 3, September 2004, pp. 235–247, doi:10.1504/IJMC.2004.005162.

[17] A. Lorenz, "Architectural patterns for applications with external user interface elements," Pervasive and Mobile Computing, vol. 9, no. 2, April 2013, pp. 269–280, doi:10.1016/j.pmcj.2012.09.006.

[18] F. E. Shahbudin and F. F. Chua, "Design patterns for developing high efficiency mobile application," Journal of Information Technology & Software Engineering, vol. 3, no. 3, 2013, pp. 1–9, doi:10.4172/2165-7866.1000122.

[19] E. B. Fernandez, R. Monge, and K. Hashizume, "Two patterns for cloud computing: Secure virtual machine image repository and cloud policy management point," Proc. 20th Conference on Pattern Languages of Programs, October 2013, pp. 1–11.

[20] C. Fehling, et al., "Capturing cloud computing knowledge and experience in patterns," Proc. 5th International Conference on Cloud Computing, June 2012, pp. 726–733, doi:10.1109/CLOUD.2012.124.

[21] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer, 2014.

[22] A. Chazalet, "Service level agreements compliance checking in the cloud computing: Architectural pattern, prototype, and validation," Proc. 5th International Conference on Software Engineering Advances, August 2010, pp. 184–189, doi:10.1109/ICSEA.2010.35.

[23] N. Delessy-Gassant and E. B. Fernandez, "The secure MVC pattern," Proc. 1st International Symposium on Software Architecture and Patterns, July 2012, pp. 1–6.

[24] J. D. Meier, et al., Microsoft Application Architecture Guide, 2nd ed. Microsoft Corporation, 2009. Available online at https://msdn.microsoft.com/en-us/ee658086 [retrieved: January, 2016].

 ISBN: 978-1-61208-465-7

[25] R. T. Fielding, Architectural Styles and the Design of Network-Based Software Architectures. PhD Thesis, University of California at Irvine, 2000.

[26] E. Fernandez and N. Yoshioka, “Two patterns for distributed systems: Enterprise service bus (ESB) and distributed publish/subscribe,” Proc. 18th Conference on Pattern Languages of Programs, October 2011, pp. 1–10, doi:10.1145/2578903.2579146.

[27] L. Baresi and S. Guinea, “A-3: An architectural style for coordinating distributed components,” Proc. 9th Conference on Software Architecture, June 2011, pp. 161–170, doi:10.1109/WICSA.2011.29.

[28] E. A. Kendall, P. V. M. Krishna, C. V. Pathak, and C. B. Suresh, “Patterns of intelligent and mobile agents,” Proc. 2nd International Conference on Autonomous Agents, May 1998, pp. 92–99, doi:10.1145/280765.280781.

[29] H. Shams and K. Zamanifar, “MVCC: An architectural pattern for developing context-aware frameworks,” Proc. 11th International Conference on Mobile Systems and Pervasive Computing, July 2014, pp. 344–351, doi:10.1016/j.procs.2014.07.035.

[30] K. J. Rothenhaus, J. B. Michael, and M. Shing, “Architectural patterns and auto-fusion process for automated multisensor fusion in SOA system-of-systems,” IEEE Systems Journal, vol. 3, no. 3, September 2009, pp. 304–316, doi:10.1109/JSYST.2009.2022572.

[31] D. Parsons, A. Rashid, A. Telea, and A. Speck, “An architectural pattern for designing component-based application frameworks,” Software: Practice and Experience, vol. 36, no. 2, February 2006, pp. 157–190, doi:10.1002/spe.694.

[32] E. Guerra, C. Fernandes, and F. F. Silveira, “Architectural patterns for metadata-based frameworks usage,” Proc. 17th Conference on Pattern Languages of Programs, October 2010, pp. 1–25, doi:10.1145/2493288.2493292.

[33] S. Prakash, A. Kumar, and R. B. Mishra, “MVC architecture driven design and agile implementation of a web-based software system,” International Journal of Software Engineering & Applications, vol. 4, no. 6, November 2013, pp. 13–26, doi:10.5121/ijsea.2013.46021.

[34] R. Moser, A. Sillitti, P. Abrahamsson, and G. Succi, “Does refactoring improve reusability?” Proc. International Conference on Software Reuse, June 2006, pp. 287–297, doi:10.1007/11763864_21.

[35] R. Mordinyi, “Towards an Architectural Framework for Agile Software Development,” Proc. 17th International Conference and Workshops on Engineering of Computer Based Systems, March 2010, pp. 276–280, doi:10.1109/ECBS.2010.38.

[36] G. M. Karam and R. S. Casselman, “A cataloging framework for software development methods,” IEEE Computer, vol. 26, no. 2, February 1993, pp. 34–44, doi:10.1109/2.191987.

 ISBN: 978-1-61208-465-7

Fast Fingerprint Recognition Using Circular String Pattern Matching Techniques

Oluwole Ajala, Moudhi Aljamea, Mai Alzamel, Costas S. Iliopoulos
Yoann Strigini

Department of Informatics
King's College London
London, UK
e-mail:{oluwole.ajala, mudhi.aljamea, mai.alzamel, costas.iliopoulos, yoann.strigini}@kcl.ac.uk

***Abstract*—The performance of Automated Fingerprint Identification System (AFIS) heavily relies on how efficiently minutiae are extracted. Most, if not all, AFIS compare minutiae information (such as ridge endings and bifurcation position) in form of sets of coordinates for verification or identification. Surprisingly, research on alternative minutiae extraction schemes is scarce. This paper, proposes the implementation of the novel approach to fingerprint recognition based on the extraction of minutiae in form of circular strings, which are suitable for approximate circular string matching. In addition to that, the proposed solution is able to detect the exact location and rotation of the input fingerprint regardless of its location on the scan surface.**

***Keywords*—Biometrics; Fingerprints; Matching; Verification; Orientation Field.**

I. Introduction

Previously, a plethora of schemes for identification such as knowledge based schemes like passwords, Personal Identification Number (PIN) and token based schemes like passports, driving license were used for identification purposes. However, with the emergence of the internet,the need for automatic person identification has become imperative. Specially with the increase of the dynamic nature of personal activities, particularly business and industry. Consequently, biometric means of attaining this has become predominant. [1], [2].

Biometrics has to do with the metrics or statistical analysis of biological data which can be human traits or characteristics. Biometric identifiers are peculiar and unique to individuals; personal identification based on biometric data offer the most accurate means of identification, hence, among all other forms of biometrics such as eye, face, voice and speech [3], the fingerprint identification remains the most popular till date. Fingerprints have provided an impeccable means of user authentication and personal identification for a long time, possibly dating back to the 19th century, when the records of fingerprint details of criminals in Argentina were released [4]. It has long since been adopted not just for law enforcement purposes (forensics and police) but also for commercial purposes like financial transactions and most recently, it is used as an authentication method in mobile devices and computers. With regards to application, two kinds of fingerprint recognition systems exist (identification and verification). In the identification system, the query fingerprint is inputted and then matched against a computed list of stored fingerprints for resemblance. In this case, the output will be understandably short or non-existent as no two fingerprints are alike. The verification system however, involves an input of query fingerprints with claimed identities, to be matched against already stored IDs (name and fingerprint) within a database to corroborate consistency. The system then outputs a result which can be either an affirmative or a negative message. The bulk of research that has focused on fingerprint authentication, has however, neglected the rotational issues that arise with fingerprints resulting to incorrect orientation identification. This is because it is assumed and often times wrongly, that the direction of the fingerprint will align with the stored fingerprint image. This singular issue poses tension in fingerprint matching, which only a negligible number in the literature [5] have considered. As computers and mobile devices adopt fingerprint recognition as a way to authenticate user, this apparent tension gains more popularity, becoming an integral research area which must be addressed.

A. Our Contribution

Responding to this rotational issue, this paper proposes a novel pattern matching technique that caters for orientation differences in fingerprints. Despite a plethora of fingerprint matching algorithms, there is still room for improvement [6]. Our proposed solution will employ A Novel Pattern Matching Approach for Fingerprint-based Authentication proposed by [7], by implementing a pre-matching stage called the orientation identification stage and then match the fingerprint image with stored images using an efficient, error tolerant, pattern matching algorithm. The fingerprint is intercepted with a series of scan circles and the minutiae information is derived. This information will then be translated into a string. This fingerprint string information is now matched against a database of stored images using approximate string matching techniques. With this approach, identification of fingerprints can be done in linear time, with respect to the total length of all strings to be searched [7].

 ISBN: 978-1-61208-465-7

B. Road Map

The organization of the rest of this paper is as follows. In Section II, we present some background related to fingerprints. Section III presents a very brief literature review. We present our approach in Section IV. The experiment and the result analysis will be presented in section V. Finally, we briefly conclude and state the future work in VI.

II. Background

Fingerprints are made up of minutiae, which are basically ridges and furrows in parallelism with each other. These minutiae form a complicated pattern that when impressed on a fingerprint scanner, leaves a print. These prints are matched to stored images on a database for either verification, authentication or both purposes. The fundamental fingerprint (FP) patterns that exist are whorl, loop and arch [8]. However, a commonly used classification is the Henrys classification [9] [10] consisting of eight classes: Plain Arch, Tented Arch, Left Slant Loop, Right Slant Loop, Plain Whorl, Double Loop Whorl, Central Pocket Loop Whorl and Accidental Whorl (see Figure 1).

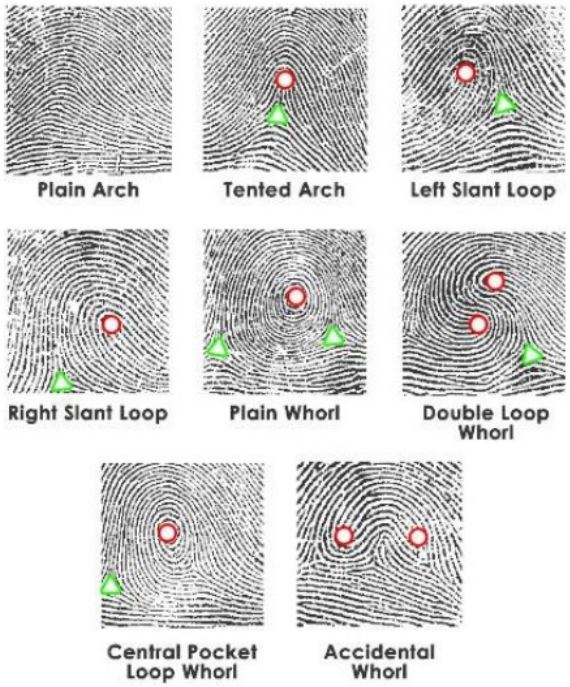


Figure. 1. Classification of fingerprint patterns

Each fingerprint is permanent and of course unique. This distinctiveness is derived by features such as ridges-ridge endings, ridge bifurcation, valleys and furrows referred to as minutiae which form a unique pattern. Recent studies have shown that the probability of two persons sharing same fingerprint is less than one in a billion [11], hence its uniqueness. In early cultures, fingerprints have been relied on to identify individuals using the so-called ink-technique [12]. This ink-technique required that a persons fingers were first coated in printing ink to get an impression on paper cards. This copy was then scanned to get a digital image. The ink-technique, though an off-line obtainment method, is still applicable today especially in forensic studies as fingerprints often have to be gathered from crime grounds. However, it is impractical for biometric studies [13]. The alternative approach is of course to scan and match fingerprints in real time.

III. Related works

Fingerprint recognition has been a core study since prehistoric times, leading to the proposal of several algorithms to developing an almost precise recognition system. Literature on fingerprint recognition has attempted to cover a wide span on the minutiae of fingerprints [14], [15], [16], [17]. Additionally, the memory and processor intensive computation issues has been discussed and addressed in some previous works. However, most of these recognition approaches hinge on the assumptions that the fingerprint impression was got from a vertically placed finger to produce a linear pattern. The minutiae based matching remain the most popular approach. This is because minutiae are believed to be the most discriminating and reliable features [18].

These previous works that have been grounded on the fingerprint minutiae recognition ignored to deliberate on the image distortions that can occur when obtaining a print with different rotation (see Figure 2). As a result, researchers have also used other features for fingerprint matching. For example, the algorithm in [19] works on a sequence of points in the angle-curvature domain, after transforming the fingerprint image into these points. A filter-based algorithm using a bank of Gabor filters to capture both local and global details in a fingerprint as a compact fixed-length finger code is presented in [20]. In the literature, the combination of different kinds of features have also been studied [21], [22]. There exist various other works in the literature proposing different techniques for fingerprint detection based on different feature sets of fingerprints [16], [23], [24]. Due to brevity we do not discuss these works in this paper. Interested readers are referred to a very recent review by Unar et al. [2] and references therein.

Note that, in addition to a large body of scientific literature, a number of commercial and proprietary systems are also in existence. In the related industry, such systems are popularly termed as Automatic Fingerprint Identification System (AFIS). A problem with the Automated Fingerprint Identification system (AFIS) has to do with the sensors used to capture the image. It is impractical to assume that the fingerprints to be compared are obtained from a central sensor. This will inevitably lead to a conflict in pattern matching when a different sensor is used [25].

Also, with the commercially available AFISs, there poses the challenge of increasing the matching speed without compromising the accuracy in the application context of identification, more so, when the database is large [6]. For this reason,

 ISBN: 978-1-61208-465-7

the quest for yet a better fingerprint recognition algorithm is nascent [6].

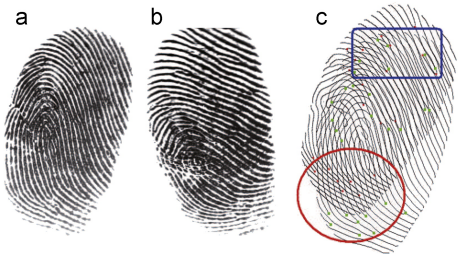


Figure. 2. An example of large distortion from FVC2004 DB1 [26]

IV. Our Approach

As opposed to gathering information about ridge endings and bifurcations from each fingerprint, the proposed algorithm extracts minutiae information in form of circular strings. Thereafter, the Approximate Circular String Matching via Filtering (ACSMF) algorithm [27] is applied to the circular strings, to find all occurrences of the rotations of a pattern of length m in a text of length n [28], where n is the concatenation of all string representations of the fingerprints in the database, and m is the string representation of the fingerprint to identify. It follows that complexity of this approach is $O(n)$. The solution proposed in [7] is divided into two main stages:

- Stage 1 Orientation Identification
- Stage 2 Verification and Matching

A. Algorithmic Overview

```
1   ALGORITHM novel-minutiae-extraction (char[][] img, int r, int cx, int cy)
2   // INPUT: "img", a 2d char array representing a fingerprint scan
3   // INPUT: "r", the radius of the current circular scan
4   // INPUT: (cx, cy) the coordinates of the center of the current circular scan
5   // OUTPUT: "pattern", a circular binary string
6   FOR i FROM cx - r TO cx + r
7       FOR j FROM cy - r TO cy + r
8           IF r^2 = (i – x)^2 + (j – y)^2 THEN
9               // pixel img[i][j] is at the intersection of the FP with the circle
10              IF x < cx THEN
11                  // Left half of scan circle
12                  IF pixel at img[i][j] < 125 THEN
13                      append 0 to the left of pattern
14                  ELSE
15                      append 1 to the left of pattern
16              ELSE
17                  // Right half of scan circle
18                  IF pixel at img[i][j] < 125 THEN
19                      append 0 to the right of pattern
20                  ELSE
21                      append 1 to the right of pattern
22  RETURN pattern
```

Figure. 3. Minutiae Extraction Algorithm.

B. Details of Stage 1: Orientation Identification

In this stage, we employ a novel approach based on circular templates as follows. Let us use f_i to denote the image of the input fingerprint. Let us assume that we know the appropriate center point, p of f_i. We then can convert f_i to a representation consisting of multiple circular bit streams by extracting circular segments of the image as shown in the algorithmic overview /Figure 3. This is achieved by constructing k concentric circles C_j of radius $r_j, 1 \leq j \leq k$, with center at point p. For each circle, we obtain minutiae features of the image by storing 1 wherever the edge of a circle intersects with a ridge and a 0 if it intersects with a furrow. So, in this way, for f_i, we get k concentric circles, which can be transformed into k circular binary strings see Figure 4. Clearly, this procedure can be easily applied on a fingerprint data stored on the database. In what follows, we will use $Y_j, 1 \leq j \leq k$ to denote the k circular strings obtained after applying the above procedure on a fingerprint data stored in the database. In what follows, we may slightly abuse the notation and say the Y_j corresponds to the circle of radius r_j.

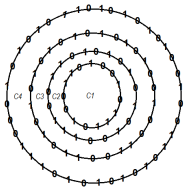


Figure. 4. Intersection of a circle with the fingerprint

Now, to identify the location and orientation of the input fingerprint we generalize the above approach to extract the minutiae feature and apply the approximate circular string matching algorithm of [28] as described in Figure 5. What we do is as follows. For the input fingerprint, we cannot assume a particular center point to draw the concentric circle which is actually the main reason for difficulty in the process. So, instead, we take reference points at regular intervals across rows and columns of the entire frame of the image (i.e., the input scanning area) and at each point p_ℓ, concentric circles $C_{j\ell}$ of radius r_j are constructed. Like before, k is the number of circles at each reference point p_ℓ. So, from the above procedure, for each point p_ℓ we get k circular strings $X_{j\ell}, 1 \leq j \leq k$.

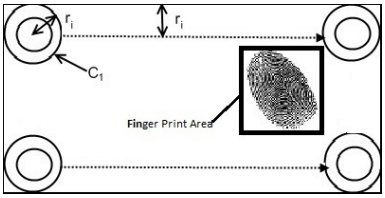


Figure. 5. Identifying the orientation and surface area of the fingerprint impression

At this point the problem comes down to identifying the best match across the set of same radius circles. To do this we make use of the Approximate Circular String Matching via Filtering (ACSMF) algorithm, presented in [28], which is accurate and extremely fast in practice. To do this we take a particular $X_{j\ell}$, construct $X_{j\ell}.X_{j\ell}$ (to ensure that all conjugates of $X_{j\ell}$ are considered) and apply algorithm ACSMF on $X_{j\ell}.X_{j\ell}$ and

 ISBN: 978-1-61208-465-7

Y_j. In other words, we try to match the circular string Y_j (corresponding to the circle of radius r_j) to all circular strings $X_{j\ell}$ (corresponding to the circle of radius r_j) generated at each point p_ℓ. Thus we can identify the best matched circular string, i.e., the best matched circles and thereby locate and identify the fingerprint impression with the correct orientation. Once the orientation has been identified, we can apply standard techniques to reorient the image to match with the image from the database in the next stage.

C. Details of Stage 2: Verification and Matching

As with most other fingerprint recognition systems, a database with fingerprint information is kept. It is against this, that the queried fingerprint will be matched. Once stage 1 (the orientation identification stage) is complete, we can then simply re-orient the fingerprint impression to suit the stored format in the database, then the matching algorithm runs on an assumed dual image of the same orientation and magnitude. This is called the verification and can be effectively carried out thus. Each image, now viewed as a two dimensional matrix, consisting of zero and one values can be converted to a binary string (one dimension). At this point, we are left with just pattern matching between two strings of equivalent length. However, note that the possibility of errors must be considered here. Hence, we simply compute the edit distance between the two binary strings and if the distance is within the tolerance level, we consider the fingerprint to be recognized. Otherwise, the authentication fails. In fact, the used matching algorithm (Levenshtein algorithm) computes the best alignment by using an edit distance which simply states the number of differences that must be changed to attain a perfect match, without considering error possibilities,

V. The Experiment

The proposed approach has been developed in ANSI C/C++ using the external library OpenCV (freely available for academic use, under the BSD licence, at http://opencv.org) for standard image processing. Different inputs have been tested by running the `fp_auth` several times against the Fingerprint Special Database of the National Institute of Standards (NIST) [29]. All external sources are open/free for academic purposes under (BSD licence). The experiment has been tested with black and white Tiff images. These images have been preprocessed to be thinned fingerprints using C++ implementation of the Guo-Hall image thinning algorithm [30]. The results in (Table I) shows the experiment results over enhanced images with different parameters.

The data entries in the table are explained as follows: Mated image refers to the input image whether it is related to the compared image or not, No. of mismatch allowed is the tolerance threshold under which the input fingerprint is to be considered as candidate match corresponding to the set of circular strings, , Max radius, is the radius for the maximum circle by pixels that can be scanned per image, Radius distance, is the number of interval in pixels between each circle centre point.Elapsed time to get scans is the time in seconds to get the total circular scans per image. No. matches is the number of candidate matches after applying the ACSMF algorithm. Finally,Finally, Rotation in pixels is the rotation to be applied on the input fingerprint image in pixels.

TABLE I. Experiment Results 1

Mated Image	No. Mismatch allowed	Max Radius	Radius distance	Elapsed time of get scans	No. Matches	Rotation in pixels
Y	10	60	2	0.7197	0	0
Y	30	50	2	0.7308	1	10
Y	30	60	10	0.077261	6	<10
Y	30	60	5	0.292098	5	<10
Y	50	60	2	0.6865	>30	10
Y	60	60	10	0.074452	>30	<10
N	80	60	2	0.7823	2	137
N	20	60	10	0.054203	0	0

In particular, the table shows that the time to get scans for each image is less than a second. Essentially, it displays that increasing the number of allowed mismatch, will result in increasing the number of matched candidates returned by ACSMF. For instance, when mated images are scanned and compared when the number of allowed mismatch is very low, almost equal to 10, it results to a negative return. . In contrast, when the number of mismatch allowed is very high for example 80, the number of returned matches is 2 even though the input image is different to the stored image. However, the correct match is shown when the number of mismatch is equal to 30. In general, the results show the effect of choosing the number of mismatch allowed which should not be very high to avoid false positive returns nor very low to prevent the false negative rate either. Finally, the results indicates the direct proportion between the circles centre point in each image and the scanning speed. Finally, the results indicate the direct proportion between the circles centre point in each image and the scanning speed.

The main advantage of this approach is regardless of the fingerprint rotation degree, the accuracy of the result will not be affected, whereas most of the other finger print detection algorithms accuracy results are affected by the rotation degree. Moreover, according to the Fingerprint Matching and Non-Matching Analysis for Different Tolerance Rotation Degrees study in [31] where they evaluated three biometric systems Neurotechnology Verifinger 6.0 Extended, Innovatrics IDKit SDK and Griaule Fingerprint SDK 2007 and the influence of the fingerprint rotation degrees on false match rate (FMR), their results showed that the FMR values increase as rotation degrees increase too. Additionally, it was stated that one of the factors that affect the performance of the matching algorithm is the fingerprint rotation. However, this is not the case in our approach.

 ISBN: 978-1-61208-465-7

A. Accuracy and Speed

We have two parameters that determine the accuracy of our approach. In Stage 1, the accuracy depends on the number of concentric circles, k. The larger the value of k, the higher the accuracy of pinpointing the location with the correct orientation. However, as k increases the computational requirement and time also increases. In Stage 2, we have another parameter d which is the tolerance level, i.e., the (edit) distance allowed between the two strings. At this point a brief discussion on the response time of our algorithm is in order. Note that, the bulk of the computational processing in our approach is required in Stage 1, where we apply algorithm ACSMF to identify the best matched circles. As has been shown in [28], on average, ACSMF works in linear time to the size of the input and is extremely fast. The size of the circles and hence the corresponding circular strings are very small and can be assumed to be constant for all practical purposes. As a result the running time of Stage 1 would be extremely fast. Again, since the size of the fingerprint image is very small, any efficient approximate string matching algorithm in Stage 2 would give us a very quick result. As an example, for an image of 300x300 pixels, extracting 45 circular strings having radius of 60 pixels takes on average 0.2200 seconds. Furthermore, the per-column loop nested inside the per-row loop allows for a 10x faster access this is because of the way C accesses memory.

B. Two Modes of Fingerprint Recognition System

As has been mentioned before, in terms of applications, there are two kinds of fingerprint recognition systems. So far, we have only considered the mode where the input is a query fingerprint with an identity (ID) and the system verifies whether the ID is consistent with the fingerprint (i.e., verification mode). Here, the output is an answer of *Yes* or *No* and we need only match against one fingerprint from the database (i.e., the finger print coupled with the ID). To handle the other mode (identification mode), we need to match the query fingerprint against a list of fingerprints in the database. This can be done using an extension of algorithm ACSMF, namely Approximate Circular Dictionary Matching via Filtering algorithm (ACDMF) [32]. We omit the details here due to space constraints. Both ACSMF and ACDMF implementations are available at [27].

VI. Conclusion and Future Work

This paper proposes yet a new pattern matching based approach for fast and accurate recognition of fingerprints. A notable challenge in fingerprint matching is that the rotation of the fingerprint is assumed to be in sync with the stored image; in this paper we have tackled this issue. The novel element of this paper is the process of using a series of circles to transform minutiae information into string information consisting of 0s and 1s, and then using the approximate circular string matching algorithm to identify the orientation. This technique has improved the performance and accuracy of the fingerprint verification system. Although our matching algorithm produces nearly accurate results at high speed, implementing the suffix tree technique to this approach will improve the accuracy and speed for big volume data.

References

[1] S. Sebastian, "Literature survey on automated person identification techniques," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 5, pp. 232–237, May 2013.

[2] J. Unara, W. C. Senga, and A. Abbasi, "A review of biometric technology along with trends and prospects," *Pattern Recognition*, vol. 47, no. 8, pp. 2673–2688, August 2014.

[3] P. Szor, *The art of computer virus research and defense*. Addison-Wesley Professional, 2005.

[4] National Criminal Justice Reference Service, *The Fingerprint Sourcebook*, A. McRoberts, Ed. CreateSpace Independent Publishing Platform, 2014.

[5] A. Agarwal, A. K. Sharma, and S. Khandelwal, "Study of rotation oriented fingerprint authentication," *International Journal of Emerging Engineering Research and Technology*, vol. 2, no. 7, pp. 211–214, 2014.

[6] P. Gutierrez, M. Lastra, F. Herrera, and J. Benitez, "A high performance fingerprint matching system for large databases based on gpu," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 62–71, 2014.

[7] M. Aljamea, T. Athar, C. S. Iliopoulos, S. P. Pissis, and M. S. Rahman, "A novel pattern matching approach for fingerprint-based authentication," in *PATTERNS 2015: The Seventh International Conferences on Pervasive Patterns and Applications*. IARIA, 2015, pp. 45–49.

[8] K. H. Q. Zhang and H. Yan, "Fingerprint classification based on extraction and analysis of singularities and pseudoridges," in *Proceedings of the Pan-Sydney area workshop on Visual information processing-Volume 11*, ser. VIP 2001. Sydney, Australia: VIP, 2001, pp. 83–87.

[9] E. R. Henry, *Classification and Uses of Finger Prints*. Routledge, 1900.

[10] H. C. Lee, R. Ramotowski, and R. E. Gaensslen, Eds., *Advances in Fingerprint Technology, Second Edition*. CRC Press, 2002.

[11] S. Sebastian, "Literature survey on automated person identification techniques," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 5, pp. 232–237, 2013.

[12] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer-Verlag, 2009.

[13] Griaule Biometrics. (2014, Nov.) Online and offline acquisition. [retrieved: 01.2016]. [Online]. Available: http://www.griaulebiometrics.com/en-us/book/

[14] X. Tan and B. Bhanu, "Fingerprint matching by genetic algorithms," *Pattern Recognition*, vol. 39, no. 3, pp. 465–477, 2006.

[15] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An identity-authentication system using fingerprints," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1365–1388, 1997.

[16] Z. M. Kovacs-Vajna, "A fingerprint verification system based on triangular matching and dynamic time warping," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1266–1276, 2000.

[17] X. Tan and B. Bhanu, "Robust fingerprint identification," in *International Conference on Image Processing 2002*, vol. 1. IEEE, 2002, pp. I–277.

[18] C. Kai, Y. Xin, C. Xinjian, Z. Yali, L. Jimin, and T. Jie, "A novel ant colony optimization algorithm for large-distorted fingerprint matching," *Pattern Recognition*, vol. 45, no. 1, pp. 151–161, 2012.

[19] A. A. Saleh and R. R. Adhami, "Curvature-based matching approach for automatic fingerprint identification," in *System Theory, 2001. Proceedings of the 33rd Southeastern Symposium on*. IEEE, 2001, pp. 171–175.

[20] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *Image Processing, IEEE Transactions on*, vol. 9, no. 5, pp. 846–859, 2000.

[21] A. Jain, A. Ross, and S. Prabhakar, "Fingerprint matching using minutiae and texture features," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3. IEEE, 2001, pp. 282–285.

[22] A. V. Ceguerra and I. Koprinska, "Integrating local and global features in automatic fingerprint verification," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3. IEEE, 2002, pp. 347–350.

 ISBN: 978-1-61208-465-7

[23] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , vol. 21, no. 4, pp. 348–359, 1999.
[24] M. R. Girgis, A. A. Sewisy, and R. F. Mansour, "A robust method for partial deformed fingerprints verification using genetic algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2008–2016, 2009.
[25] A. Ross and A. Jain, "Biometric sensor interoperability: A case study in fingerprints," in *Biometric Authentication*. Springer, 2004, pp. 134–145.
[26] C. Xinjian, T. Jie, Y. Xin, and Z. Yangyang, "An algorithm for distorted fingerprint matching based on local triangle feature set," *Information Forensics and Security, IEEE Transactions on*, vol. 1, no. 2, pp. 169–177, 2006.
[27] S. P. Pissis. (2015) Acsmf implementation. [retrieved: 01.2016]. [Online]. Available: https://github.com/solonas13/asmf/
[28] C. Barton, C. S. Iliopoulos, and S. P. Pissis, "Fast algorithms for approximate circular string matching," *Algorithms for Molecular Biology*, vol. 9, no. 1, pp. 1–10, 2014.
[29] NIST. (2015) Biometric special databases and software. [retrieved: 01.2016]. [Online]. Available: http://www.nist.gov/itl/iad/ig/special_dbases.cfm
[30] OpenCV-code. (2015) Implementation of guo-hall thinning algorithm. [retrieved: 01.2016]. [Online]. Available: http://opencv-code.com/quick-tips/implementation-of-guo-hall-thinning-algorithm/
[31] A. Perez-Diaz and I. Arronte-Lopez, "Fingerprint matching and non-matching analysis for different tolerance rotation degrees in commercial matching algorithms," *Journal of applied research and technology*, vol. 8, no. 2, pp. 186–199, 2010.
[32] C. Barton, C. S. Iliopoulos, S. P. Pissis, and F. Vayani, "Accurate and efficient methods to improve multiple circular sequence alignment," submitted.

 ISBN: 978-1-61208-465-7