Selected Research Articles

A Model-Driven Approach for the Formal Verification of Storm-Based Streaming Applications
F. Marconi, M. Bersani, and M. Rossi 6

A Multi-scale Modeling Approach for Systems of Systems Architectures
I. Khelif, M. Kacem, C. Eichler, and A. Kacem 17

An Efficient Visual Fiducial Localisation System
P. Lightbody, T. Krajník, and M. Hanheide 28

Computer Aided Diagnosis with Boosted Learning for Anomaly Detection in Microwave Tomography
C. Pack, S. Son, and S. Shin 39
SIGAPP FY’17 Quarterly Report

July 2017 – September 2017
Jiman Hong

Mission

To further the interests of the computing professionals engaged in the development of new computing applications and to transfer the capabilities of computing technology to new problem domains.

Officers

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>Jiman Hong</td>
<td>Soongsil University, South Korea</td>
</tr>
<tr>
<td>Vice Chair</td>
<td>Tei-Wei Kuo</td>
<td>National Taiwan University, Taiwan</td>
</tr>
<tr>
<td>Secretary</td>
<td>Maria Lencastre</td>
<td>University of Pernambuco, Brazil</td>
</tr>
<tr>
<td>Treasurer</td>
<td>John Kim</td>
<td>Utica College, USA</td>
</tr>
<tr>
<td>Webmaster</td>
<td>Hisham Haddad</td>
<td>Kennesaw State University, USA</td>
</tr>
<tr>
<td>Program Coordinator</td>
<td>Irene Frawley</td>
<td>ACM HQ, USA</td>
</tr>
</tbody>
</table>

Notice to Contributing Authors

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library and in any Digital Library related services
- to allow users to make a personal copy of the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will refer requests for republication directly to you.

Next Issue

The planned release for the next issue of ACR is December 2017.
MAKE THINGS POSSIBLE
with a degree in applied computer science from the University of Wisconsin.

Organizations need IT professionals who can excel technically and drive business results. With the UNIVERSITY OF WISCONSIN BACHELOR OF SCIENCE IN APPLIED COMPUTING, YOU CAN MAKE THINGS POSSIBLE— for your company and your career.

FUTURE-FOCUSED LEARNING
Solve today’s IT issues AND learn to develop tomorrow’s technology

TECHNOLOGY AND BUSINESS COMBINED
A computer science curriculum with a sharp focus on business skills

REAL-WORLD APPLICATION
Courses that teach applicable compound skills, plus valuable workplace experience in the capstone course
A Model-Driven Approach for the Formal Verification of Storm-Based Streaming Applications

Francesco Marconi  
Politecnico di Milano  
Milan, Italy  
francesco.marconi@polimi.it

Marcello M. Bersani  
Politecnico di Milano  
Milan, Italy  
marcellomaria.bersani@polimi.it

Matteo Rossi  
Politecnico di Milano  
Milan, Italy  
matteo.rossi@polimi.it

ABSTRACT

Data-intensive applications (DIAs) based on so-called Big Data technologies are nowadays a common solution adopted by IT companies to face their growing computational needs. The need for highly reliable applications able to handle huge amounts of data and the availability of infrastructures for distributed computing rapidly led industries to develop frameworks for streaming and big-data processing, like Apache Storm and Spark. The definition of methodologies and principles for good software design is, therefore, fundamental to support the development of DIAs. This paper presents an approach for non-functional analysis of DIAs through D-VerT, a tool for the architectural assessment of Storm applications. The verification is based on a translation of Storm topologies into the CLTLoc metric temporal logic. It allows the designer of a Storm application to check for the existence of components that cannot process their workload in a timely manner, typically due to an incorrect design of the topology.

CCS Concepts

● Theory of computation → Verification by model checking; ● Software and its engineering → Model-driven software engineering;

Keywords

Formal Verification; Apache Storm; MDE; Data-intensive Applications; Temporal Logic

1. INTRODUCTION

Data-intensive applications (DIAs) are computational systems that process, in a relative small amount of time, huge amounts of diversified information usually produced by data sources with high throughput. Some of the most popular companies nowadays—e.g., Twitter (www.twitter.com), Groupon (www.groupon.com), Spotify (www.spotify.com), etc.—make large use of DIAs to process data gathered from millions of users.

DIAs constitute a significant asset for the production of large-scale software, and have been drawing the attention of both academia and industry. The creation of frameworks that support designers over the entire life-cycle (design, development, testing, deployment, maintenance) of DIAs is of crucial importance, and constitutes a key research challenge in this area. Topics such as techniques and tools for quality assessment, architecture enhancement, agile delivery and continuous testing of DIAs are targeted by ongoing research projects like, for instance, the DICE European project [8].

The design approach envisioned by DICE is founded on model-driven principles and can be summarized as follows. The design of an application is decomposed into three distinct and consecutive phases, each one associated with a profiled UML diagram. Each phase focuses on a specific aspect of the design and represents a refinement of the previous one that has to be validated before starting the new refinement step. If design flaws are detected, designers can either change the current model, or modify the one built in the previous step, then redo the refinement. The design process starts from a conceptual model of the application, called Platform-Independent Model (PIM); this is refined, in the second step, into the so-called Platform-Specific Model (PSM), which provides the architectural schema of the application based on a specific (data-intensive) technology; finally, in the last step, the architectural model is refined to obtain a deployment model.

Nowadays, the frameworks promoting the development of DIA can be considered mature technologies. This fact is witnessed by the spread and the popularity of streaming and data-mining industrial applications in the IT market. After decades of research and industrial development, however, most of the frameworks lack tools for the analysis of the applications at design time. Nonetheless, they commonly are equipped with monitoring platforms that allow designers to manually inspect the running applications by means of statistics based on metrics measuring the processing time, the latency of the application, the throughput of the nodes and so on. We approach the assessment of DIAs by applying formal verification to the architectural models described through (metric) temporal logic. The goal of the analysis is to determine, at design time and through automated techniques, whether the behavior entailed by the architecture of the application conforms to specific properties over time. The properties that an application should satisfy typically depend on the technology adopted to implement the appli-
cacion. For instance, we employed a logic-based modeling technique for the analysis of DIA in [14] and in [13].

Most of the available data-intensive frameworks allow designers to specify the architecture of DIAs as a directed graph whose nodes are computational resources which carry out specific operations. The semantics underlying a graph, which reflects the runtime behavior of the application, is determined by the target technology (e.g., the same graph has two different interpretations in case we adopt a streaming or a batch technology). In this paper, we consider Apache Storm [1], a popular technology for stream-based applications. The architecture of a Storm application is defined by means of a topology—i.e., a directed graph—where nodes are of two kinds: computational nodes, which implement the logic of the application by elaborating information and producing an outcome; and input nodes, which bring information into the application from its environment.

Various are the resources on the web that point out criteria guiding the design of Storm topologies, such as [4]. In most of the cases, the designers can follow guidelines [16, 4] that facilitate the analysis of the application by using the statistics that are available from the monitoring platform. To the best of the authors’ knowledge, however, there are no formal techniques for the analysis of temporal properties at design-time.

This paper presents a different perspective. First, the model-driven approach fostered by the DICE workflow is conducted by means of a simple application, which is developed through an iterative refinement process. The complete verification workflow of the architectural model is carried out according to the concepts included in the DICE UML profile [10] and the validation of the topology is done through the analysis performed by D-VerT [2], the DICE tool that allows the verification of Storm topologies at design-time. We also focus on all the necessary transformations needed for translating the UML diagram of the architecture of the DIA into a formula of the CLTL metric temporal logic [7], which is solved by D-VerT to validate the Storm architecture represented by the UML model. Finally, we present the verification tool, which is the component implementing the transformations. Furthermore, we introduce an industrial use case that is provided by one of the partner in the DICE consortium and we use it to validate our verification approach. We set up an experiment to compare the result obtained with D-VerT and the behavior of the application at runtime. The real application has been abstracted by means of a topology characterized by the same non-functional properties, which has been implemented using stub components that mimic the behavior of the nodes of the real application. Then, through the monitoring platform, the topology has been analyzed and the resulting behavior has been compared with the D-VerT outcome.

The paper is structured as follows: Section 2 presents some background notions on Apache Storm and briefly recaps our approach to the modeling of Storm topologies with temporal logic introduced in [14]. Section 3 introduces the methodology for the verification of Storm topologies based on formal validation of refined UML models. Section 4 describes the structure of D-VerT and the transformations needed for enabling the verification of architectural models. Section 5 shows the application of the methodology through and example of Storm application which, at the end, undergoes verification with D-VerT. Section 6 presents another use case for the tool and addresses the validation of the verification results by monitoring the topology. Section 7 briefly discusses some related works, and Sect. 8 concludes.

2. BACKGROUND

2.1 Apache Storm

Apache Storm [1] is a stream processing system, developed and open sourced by Twitter in 2012, allowing real-time processing of large-scale streaming data on horizontally scalable systems through a parallel and distributed computation.

The computational model of Storm applications is the Storm topology, i.e., a directed graph whose nodes realize the operations performed over the data flowing through the application and whose edges indicate how such operations are combined. Data is encoded into streams that are infinite sequences of structured messages, also named tuples, which are processed by the application.

A topology node set consists of spouts and bolts (in the following also referred to as topology components). Spouts are stream sources which usually get data from external systems, such as queuing brokers (e.g., Kafka, RabbitMQ, Kestrel), or from other data sources (e.g., Twitter Streaming APIs), whereas bolts transform the incoming data streams into new output streams to be processed by the connected bolts. Connections are statically defined at design time by the subscription of the bolts to other spouts or bolts. Fig. 1 shows an example of Storm topology that will be used in Sect. 5.

Storm is capable of guaranteeing the so-called “at least once” message processing. Reliable spouts keep track of all the tuples they emit, and if one of them fails to be processed by the entire topology within a certain timeout, then the spout re-emits it into the topology. When message processing is “best-effort”, instead, (unreliable) spouts emit each tuple only once, without checking for the successful completion of the processing. Bolts usually perform operations, such as filtering, joining, functions, database interaction, which are combined through the topology to perform complex transformations. The Java interfaces defining a bolt include the execute() method that implements its functionality; it reads the input tuples, processes the data, and emits (via the emit() method) the transformed tuples on the output streams. When the spouts are reliable, bolts have to acknowledge the successful or failed processing of each tuple at the end of the
Storm is designed to be executed on distributed clusters and leverage their computational power. A deployed topology is composed of one master node and several worker nodes. Each worker node instantiates one or more worker processes to enable the execution of the functionalities implemented by spouts and bolts belonging to the same topology. Each worker process runs a JVM and spawns therein one or more executors (i.e., threads). Executors run one or more tasks which, in turn, execute either a spout or a bolt.

Running a topology requires the definition of a number of parameters, among which:

- the number of executors running in parallel each component, either spout or bolt (i.e., the parallelism associated with the execution of the component) and
- the total number of tasks over those executors.

Since each executor corresponds to a single thread, multiple tasks run serially on the same executor, though the default option is one task per executor.

Communication among workers and executors is managed through a multi-level queuing system. Each executor has its own input queue and output queue. Tuples are read from the input queue by the thread handling the spout/bolt logic; afterwards, when they are emitted, they are put into the output queue and later moved to the worker’s transfer queue by means of a send thread running within the executor. Every worker runs a Receive thread and a Send thread. The former listens for incoming tuples appended in a worker’s receive queue and, based on the received message and the nodes configuration, forwards them to the appropriate executors’ input queue; the latter, instead, gets the tuples from the executors’ output queue, puts them in a worker’s transfer queue, and forwards them either to other workers or to other executors in the same worker. Fig. 2 depicts the internal structure of a worker queueing system and shows the executor input and output queues.

![Queueing system of a worker process.](image)

A common issue of distributed applications is caused by bottleneck nodes. In Storm applications, this criticality can take place when some executors of the topology work “at capacity”. This term refers to a metric of great interest for software engineers as the designers consider the capacity of bolts to evaluate the topology performance before the deployment of the application. The capacity of a bolt \( C \) is calculated by the Storm monitoring service and it is defined as the percentage of the time that the bolt expends in processing the incoming tuples, with respect to a given time interval where the estimation is calculated. This value can be calculated for every bolt of the topology by means of Formula (1), using the following metric values:

- \( n_{\text{executed}} \): it is the total number of incoming tuples that have been processed by a bolt in the last \( T \) milliseconds.
- \( \text{latency} \): it is the average time that a tuple spent in the \text{execute}() method of the bolt in the last \( T \) milliseconds.
- \( T_{\text{capacity}} \): total time to consider for estimating the capacity of the bolt, that is by default set to 10 minutes.

The capacity of a bolt is then defined as:

\[
C = \frac{n_{\text{executed}} \cdot \text{latency}}{T_{\text{capacity}}}
\] (1)

A bolt with capacity close to 1 spends most of the time in processing the input, being characterized by a very small idleness. As remarked in various technical on-line resources about Storm performance, high capacity values should be prevented by adopting countermeasures that diminish the values when they are close to 1. In such a situation, a topology might fail the processing of the incoming data as the receive queue of the executors, running a node working “at capacity”, might grow in size and reach the maximum limit. Therefore, new tuples that cannot be appended in the queue are irremediably lost, unless the expensive reliable processing is activated. The common practice to address this issue is to increase the parallelism of the bolts (i.e., the number of executors assigned to it) or the number of tasks that are instantiated in their executors. More refined actions can also be realized and usually employ accurate information that are collected by profiling the application [16].

### 2.2 Modeling Storm topologies

Our verification approach is founded on a temporal logic model that represents the computation of Storm topologies. In particular, the model is designed to discover unwanted behaviors of running topologies. To this end, it is specifically devised to be representative of the application runtime.

The definition of the model required first the identification of malfunctions of the application that software engineers designing Storm topologies consider to be critical at runtime. We identified some of the aspects that might cause such malfunctions and we focused on one of them, which is related to the bottleneck analysis of nodes mentioned in Sect. 2.1. Therefore, we defined our model in order to allow the analysis of the existence of bolts that would saturate their capacity as they cannot process the incoming stream of tuples on time, thus causing a monotonic growth of the size of their queues. The model of a topology captures how the timing parameters of its components—such as the delays between two consecutive spout events inputting tuples in the topology and the processing time of tuples for each bolt—affect the size of the bolts’ queue.
The relevant aspects of the computation, such as the functionality that is implemented in every method `execute()`, are reflected in the modeling by means of an appropriate behavioral abstraction. For instance, emitting a tuple and storing it into a queue is modeled through an `emit` event that increments the size of the target queue. The behavior of the relevant features and parameters of spouts and bolts is extracted by reverse-engineering the Java interfaces of the Storm API.

Furthermore, some suitable assumptions are considered to generate models that can be practically managed by state-of-the-art formal verification tools in a reasonable amount of time.

- Deployment details, such as the number of worker nodes and the features of the (possibly) underlying cluster are abstracted away; topologies are assumed to run on a single worker process and each executor runs a single task, which is the default Storm configuration of the runtime.

- Each bolt has a single receive queue for all its parallel instances and no sending queue, while the workers’ queues are not represented (single-worker scenario). For generality, all queues have unbounded size and the current number of tuples in a queue is represented by means of a positive integer value.

- The contents of tuples is not modeled and, since tuples are all assumed to have the same size, the size of queues is represented by the number of tuples they contain.

- The external sources of information abstracted by the spouts are not represented, since they are outside of the perimeter of the application. So, their queues are not represented.

- For each component, the duration of each operation or the permanence in a given state has a minimum and a maximum time.

The computation of Storm topologies is captured through a set of formulae written in the CLTLoc metric temporal logic [7] augmented with counters, which are used to represent the size of bolts’ queues during the computation. CLTLoc [7] is a decidable extension of LTL where formulae are defined over a finite set of atomic propositions and a set of dense variables over $\mathbb{R}_{\geq 0}$ representing clocks. For instance, a possible atomic formula over clock $x$ is $x < 4$. Similarly to TA, a clock $x$ measures the time elapsed since the last reset, that occurs when $x = 0$. The interpretation of `clocks` is defined through a clock valuation assigning, for every time position $N$, a real value to each clock $x$ in the formula. The semantics of time adopted is `strict`, namely the value of a clock must strictly increase in two adjacent time positions, unless it is reset.

The complete description of the formal model of a topology can be found in [5]. It consists of four parts, which represent: (i) the evolution of the state of the nodes; (ii) the behavior of the queues; (iii) timing constraints; (iv) failures. [14], instead, presents some of the technical details of the adopted decision procedure, which is validated through some experimental results.

The behavior of a bolt is the most relevant part of the topology model. A bolt can alternatively be in one of the following states: `process`, `idle` or `failure`. If a bolt is idle and its queue is not empty, then it can start processing the tuples stored therein. The Storm supervisor is responsible for the activation of the threads running the method `execute()` on a selected tuple. In our model, this is represented through an instantaneous `take` action that models the action of removing tuples from the bolt’s queue and marks the beginning of the processing. The execution of method `execute()` is represented by the state `execute`, which is part of macro-state `process`, together with the actions `take`, that delimits the beginning of the processing phase, and `emit` that occurs at the end of it. In our model, we indicate the latency of a bolt with $\alpha$. Once the execution is completed, the bolt emits output tuples with an instantaneous action corresponding to the `emit` state. Bolts may fail and failures may occur at any moment; upon a bolt failure, the system goes to the `fail` state. If no failure occurs, after an `emit` a bolt goes to `idle`, where it stays until it reads new tuples.

To give a flavor of the formal model underlying our verification approach, we introduce a few examples of CLTLoc formulae. Formulae (2)-(3) capture how the number of elements in the queue of bolt $j$ ($q_j$) is updated whenever tuples are enqueued ($\text{add}_j$) or dequeued ($\text{take}_j$). They use $\mathbb{N}$-valued discrete counters to represent the amounts of tuples exchanged in the topology. For instance, $q_j$ is the size of queue of bolt $j$. Term $Xq_j$ represents the value of $q_j$ in the next position of time. Every time the component $j$ emits ($\text{emit}_j$ holds), the queues of all bolts subscribing to $j$—i.e., those which are targets of arcs outgoing from $j$ in the topology—receive $r_{\text{emit}_j}$ tuples—i.e., the variables $q_i$ representing the occupancy level of those queues are incremented by $r_{\text{emit}_j}$. When multiple components subscribed by a bolt emit tuples simultaneously, the increment on its queue is equal to the sum of all the tuples emitted (the value of $r_{\text{adj}}$ in Formulae (2)-(3)). Dually, when $\text{take}_j$ holds, the occupancy level $q_j$ is decremented by $r_{\text{process}_j}$, (number of tuples read by bolt $j$). Proposition $\text{add}_j$ is true when at least one of the components subscribed by $j$ is emitting, whereas $\text{startFail}_j$ is true in the first instant of a failure state.

$$\text{add}_j \land \neg \text{take}_j \land \neg \text{startFail}_j \Rightarrow (Xq_j = q_j + r_{\text{add}_j}) \quad (2)$$

$$\text{take}_j \Rightarrow (Xq_j = q_j + r_{\text{add}_j} - r_{\text{process}_j}) \quad (3)$$

To measure the duration of each state and to impose timing constraints between events, we use a set of dense-time CLTLoc clock variables for each component of the topology. For example, Formula (4) imposes that when `emit` occurs, the duration of the current processing phase is between $\alpha - \epsilon$ and $\alpha + \epsilon$, where $\epsilon < \alpha$ is a positive constant that captures possible (small) variations in the duration of the processing.

$$\text{process} \land \text{emit} \Rightarrow (t_{\text{phase}} \geq \alpha - \epsilon) \land (t_{\text{phase}} \leq \alpha + \epsilon) \quad (4)$$

The formal model includes a number of parameters, such as $\alpha$ introduced above, capturing the features of the topology, which can be configured at design time. In addition to $\alpha$, other parameters are, for bolts, a coefficient $\sigma$ expressing the kind of operation performed by the bolt in terms of quantity of output tuples emitted given an input tuple, and also the minimum and maximum time to failure. Spouts, instead,
are characterized by the average number of tuples emitted per time unit. Both spouts and bolts are also characterized by their level of parallelism, corresponding to the number of executors for the component.

3. ANALYSIS OF STORM TOPOLOGIES

D-VerT allows designers to validate temporal aspects of DIAs by means of a logic-based formal verification approach outlined in Section 2.2. The implementation of D-VerT currently supports the analysis of Storm topologies and Spark execution DAGs [6], but it can be easily extended to deal with other big-data technologies if and when their computational model is formalized through CLTLoc formulae.

D-VerT is part of a more complex design process which conforms to the principles of model-driven software engineering pursued by the DICE methodology. As illustrated in Fig. 3, the designer defines the application by means of domain-specific models with an iterative approach consisting of three steps: (i) application design, (ii) design evaluation and (iii) monitoring of a running deployed application. D-VerT is situated at the second level of the design workflow and enables the refinement of the architectural design of the application depending on the outcome of the formal analysis. The input of D-VerT is an annotated UML (class) diagram which specifies the architecture, i.e., the topology, of the application. In case of an incorrect timing design, the outcome of D-VerT consists of a possible execution of the topology causing an undesired accumulation of tuples in some bolts. In such a case, the designer can refine the application architecture by changing the values of some parameters of the UML diagram and then redo the evaluation until (possibly) all flaws affecting the model are removed. A different scenario, which also entails a design refinement, might occur when some parameter values that are measured on a running application differ from the values used for verification at design time. In such a situation, monitored data obtained from the deployed application can be exploited to update the model, which can then be newly verified.

Relying on UML profiles for modeling is a common practice in model-driven development as it allows for the creation of domain-specific languages by extending or restricting UML to fit a specific domain. A UML Profile is made of a set of stereotypes, tags, constraints and meta-models that allow the designer to represent artifacts in a specific domain. A stereotype is a meta-concept which is used to categorize an element in a model (for example, container) with specific notions belonging to a domain.

As shown in Fig. 4, at the starting point of the workflow the user creates an annotated UML model describing the relevant technological aspects of the architecture of a DIA. The UML model includes suitable design abstractions, capturing specific data-intensive technologies—Storm in our case—that are adopted for implementing the architecture of an application. The diagram, called DICE Technology Specific Model (DTSM), is at the PSM level (see Sect. 1) in the model-driven approach pursued by DICE. Specifically, in a DTSM diagram, a stereotype classifies an element of an application with aspects related to a specific technology. A DTSM diagram includes generic concepts that fit many data-intensive frameworks, such as ComputationNode, StorageNode or SourceNode, and specific ones, depending on the selected technology. In the case of Storm, the relevant features and aspects defining a Storm topology constitute the meta-model for designing Storm applications. Some of them, that are used in Sect. 4, are Topology, Spout, Bolt and TopologyConfigurations. For a comprehensive description of the concepts available in DTSM diagrams see [10].

DTSM diagrams for Storm include all the values of the parameters that are useful to carry out the analysis of a topology. As depicted in Fig. 4, verification of DTSM models is done through the automatic translation of the diagrams into a set of CLTLoc formulae, which are then analyzed by the Zot bounded satisfiability checker [3] using the technique presented in [14]. More precisely, Zot is fed the CLTLoc formulae capturing the application under design and the property to be checked concerning the unbounded growth of the queues of interest.

The tool produces one of two responses: (i) a trace of the modeled Storm topology—a counterexample—corresponding to an execution of the application in which one of the queues grows indefinitely—in this case, the set of formulae is satisfiable (SAT); or (ii) the notification that the set of formulae is unsatisfiable (UNSAT). In the latter case, since the language used to formalize Storm topologies is in general undecidable, we cannot conclude definitively that there is no execution of the application such that the queues grow indefinitely, but only that, within the bounds chosen for the search of counterexamples, none was found. Still, an UNSAT result increases our confidence that no major design flaws are present in the architecture of the Storm topology for what concerns its ability to process data in a timely manner.
4. TOOL DESCRIPTION

This section outlines the architecture of the D-VerT tool, the transformation enabling the verification process and the kind of analysis currently supported by the tool.

4.1 Tool Architecture

As shown in Fig. 5, D-VerT is structured as a client-server application. The client component is an Eclipse plug-in, and is part of the DICE IDE. It allows users to define the design of the DIA under development, then, after providing some additional configuration information, to launch verification tasks and to retrieve their outcomes. The server component consists in a RESTful web application written in Python. The D-VerT server exposes APIs to launch verification tasks and to obtain information about their status. To simplify the setup and deployment steps, the D-VerT server is available as a Docker2 container. The client-server architecture decouples the application design phase from the rather computationally-intensive verification phase. Based on the needs of the user, the D-VerT server can be instantiated either on the local machine or on a remote server.

4.2 Topology creation

The design of a DIA is specified through the DICE IDE, which is based on the Papyrus tool. As mentioned above, Storm topologies are described as DICE-profiled UML Class diagrams. Each computational node of the application is defined by introducing a class tagged with a stereotype specifying whether the node is a spout or a bolt. Depending on the stereotype applied, the designer defines the values for all the necessary parameters described in Sect. 2. Subscriptions (i.e., connections) of bolts to other components are modeled as associations between the corresponding classes.

4.3 Transformations

The verification process is made possible by a two-step transformation applied on the DICE-profiled UML models to obtain the corresponding set of CLTLoc formulae.

The first step of the transformation is performed in the D-VerT client by the UML2JSON module, which extracts from the DICE-profiled UML model all parameters that are relevant for the verification. These parameters are then serialized into a JSON object, which is used to invoke the server component. The extraction of the relevant features is done by suitably navigating the UML file. DIA components and their parameters are detected thanks to their being annotated with proper stereotypes from the DICE profile.

The second step takes place in the D-VerT server, which receives the request from the client, produces the corresponding formal model and feeds it to the underlying Zot [3] tool. More precisely, the JSON2MC module, based on the contents of the JSON object included in the request, generates the temporal logic model using a templating mechanism.

4.4 Analysis

In its current stage of development, D-VerT provides support for the analysis of the boundedness of bolts’ queues. Through the run configuration dialog box of the tool (see Fig. 6) the designer can specify the bolts of interest, the depth of the search over which the verification has to be performed (the “time bound”, which corresponds to the maximum length of the trace produced) and the Zot plug-in to be used. The analysis allows for the detection of possible runs of the system leading to an unbounded growth in the size of at least one of the aforementioned bolts’ queues. This corresponds to the presence in the topology of at least one bolt that is not able to manage the incoming flow of messages in a timely manner. In this case the tool provides as output to the user the trace witnessing the existence of such a run of the system—i.e., the counterexample violating the boundedness property. The trace is returned to the user both in a textual format (i.e., the bare output of Zot) and in a graphical format, in order to provide a more user-friendly visual hint about the system execution. Figure 10 shows an example of such output trace, returned by the tool for the use case of Sect. 5. It represents the evolution of the number of tuples in the queue over time. The trace is composed by a prefix and a suffix: the latter, highlighted by the gray background, captures the growing trend of the queue size, as it corresponds to a series of operations in the system that can be repeated infinitely many times. When no trace is detected, the result is UNSAT.

5. D-VERT WORKFLOW IN ACTION

In this section we illustrate the usage flow of D-VerT for the iterative refinement of a Storm topology. The use case is taken from the open source project StormCrawler.2 Suppose we want to create a web crawler application to efficiently fetch, parse and index web resources of our interest. Given the dynamic nature of the web, this kind of task can be formulated as a streaming problem, where the input consists in a continuous stream of URLs that need to be processed by the streaming application with low latency, and the output is represented by the resulting indexes.

We start by modeling the application at the PIM level. In this case, the model simply includes a source node, a computation node and a storage node, as depicted in Fig. 7. We decide to use a Kafka queue as source node, a Storm topology as computation node and ElasticSearch as storage node.

Since we are interested in analyzing the Storm topology,

1 https://hub.docker.com/r/deibpolimi/d-vert-server
2 https://github.com/DigitalPebble/storm-crawler

Figure 5: D-VerT workflow mapped onto the client-server architecture of the tool.
we focus on the computation node and consider the source node and the target storage node as “black boxes”. At the PSM level we insert more technology-related aspects, such as, in the case of Storm, the topology structure and a series of non-functional characteristics. Figure 8 shows the PSM (DICE-profiled UML diagram) of the initial design of the topology. The configuration parameters are represented as UML comments for illustrative purposes. Notice that associations between components have the opposite direction with respect to the data flow among them, since they indicate the subscription of bolts to the associated components’ streams. The diagram includes one spout in charge of fetching the input flow of URLs from Kafka and three bolts performing various steps of the web crawling process. The partitioner bolt partitions the incoming URLs, while the crawler bolt performs many operations such as resource fetching, metadata parsing and content indexing. The status bolt at the end of the chain indexes the URL, metadata and its status to a specific “status” Elasticsearch index. Each of these topology components can be executed by an arbitrary number of parallel threads, and is characterized by the (average) execution time (time needed to perform its task) and by the (average) number of tuples emitted with respect to the number of tuples received as input. These aspects are specified as parameters in the UML class diagram. The formal analysis on the initial topology design helped us to detect an unbounded increase in the queue of the crawler bolt. This outcome from the tool led us to review the topology structure, and to decide for the decomposition of the crawler bolt in a series of bolts, each of them performing a subtask of the original bolt (fetch, sitemap, parse and index). The refined version of the topology, shown in Fig. 9, aims at lightening the load on the core crawling phase by pipelining the main operations and by directly updating the status bolt with partial results computed by the new bolts.

After the refactoring the tool revealed another unwanted run of the system, this time showing a growing trend in the queue of the status bolt (Fig. 10). This bolt, subscribing to the streams of the four newly-created bolts, needs a further refinement to avoid excessive loads in its input buffer. By increasing the parallelism level of the status bolt to 4, D-VerT was not able to detect any counterexample after tens of hours of execution, and returned an UNSAT result. Since, as already remarked, an UNSAT result would not be a guarantee of absence of counterexamples, in this work we focused on the detection of potential problems (SAT results) and their validation, as described in Sect. 6. Execution times for the verification vary significantly depending on the topology configuration, ranging from the 50 seconds of the first analysis (Fig. 8) to the many hours of the third analysis.3

3Experimental analysis carried out on commodity hardware (MacBook Air running MacOSX 10.11.4. with Intel i7 1.7 GHz, 8 GB 1600 MHz DDR3 RAM; SMT solver used by Zot was z3 v.4.5.0).
6. VALIDATION

In this section, we discuss the validation of our approach with an additional use case that will be analyzed by the D-VerT. The use case consists in a Storm application (WebAnalysis topology) which has been developed by an industrial partner in the DICE project. The application, similarly to the use case presented in Sect. 5, is designed to analyze a series of web resources in order to find, extract and categorize media items and articles. As shown in the PSM model of the topology in Fig. 11, the input data is fetched from a Redis\(^4\) in-memory database by the RedisSpout component and, after a series of operations performed by the bolts of the topology, media articles are then indexed on an instance of the open-source search platform Apache Solr\(^5\) by the SolrBolt component.

Our analysis spotted a potential problem in the input buffer of the UrlCrawlDeciderBolt component, as D-VerT returned a SAT result and provided an output trace showing an unbounded increase in the queue of the bolt (Fig. 12).

In order to validate the results, since the application code is closed-source, we have designed a topology with the same structure of the topology implementing the application and which was able to simulate the temporal characteristics of the WebAnalysis topology (in terms of \(\alpha\), \(\sigma\), parallelism, etc.). We ran the application on commodity hardware and, during the execution, we collected all the relevant quality information provided by the Storm platform, such as the capacity of each bolt (introduced in Sect. 2.1), the complete

\(^4\)https://redis.io/

\(^5\)http://lucene.apache.org/solr/
latency of the topology and the number of failed tuples due to timeout. To measure the complete latency of the processed tuples, we enabled the acking mechanisms available in Storm. Through this mechanism, a tuple is marked as failed after its complete latency exceeds the default threshold of 30 seconds. In other words, if a tuple has not been processed for more than 30 seconds then the tuple will be discarded by the topology because it is not valid anymore.

The monitored values showed that capacity of UrlCrawlDeciderBolt was constantly very close to 1, meaning that the bolt was almost always busy in processing tuples. This behavior usually implies that the bolt is not able to manage the incoming flow of data in a timely manner, negatively affecting the latency of the entire topology. The presence of a problem was confirmed by the evolution of the average complete latency measure, shown in Fig. 13 and Fig. 14. Each point shown in the graphs is the value of the average complete latency measured with respect to the last 10 minutes. The plot shows a steep increase in the first minutes of execution, followed by a sharp drop after the average complete latency goes beyond 20 seconds. The same fluctuating trend is repeated later in time. This behavior appears to be due to the timeout settings. When the average value of complete latency gets close to 20 seconds, most of the tuples accumulated in the topology are failed and discarded due to timeout. After this massive disposal, the decongestion of the input buffers allows the newly emitted tuples to be completely processed by the topology in less time. However, because of the bottleneck in the topology, the average complete latency keeps increasing and then follows the same trend.

7. RELATED WORKS

Many research works in recent years investigated the usage of MDE to support the design and the formal verification of software and embedded systems. [11] presents a systematic literature review on the formal verification of static software models. Most of the works make use of UML models, often enriched with OCL constraints, and only a part of them is fully supported by a tool implementing the model transformations and the verification process. A number of other works have used a model-driven approach for the formal verification of behavioral models (see, e.g., [9, 12]), without addressing the specificities of DIAs. To the best of our knowledge, few works try to address the problem of the verification of DIAs, none of them adopting the MDE approach. They mainly focus on the verification of properties that depend exclusively on the framework by building ad-hoc models; for example, [15] verifies data locality, deadlock freedom and non-termination properties for the Hadoop parallel architecture, while [17] verifies the validity of communication data flows of Hadoop MapReduce. Our work, on the other hand, aims at allowing for the verification of properties that depend on the application design.
8. CONCLUSION
In this paper we presented the model-driven approach to the formal verification of Storm topologies supported by the D-VerT tool. It allows designers to formally check whether, given the features of the components of the topology, it is possible for the queues of some bolts to grow indefinitely, which entails that incoming tuples will not be processed in a timely manner.

Future works will focus on enlarging the set of properties that can be analyzed, on improving the efficiency of the verification technique and on performing an extensive validation on cluster infrastructures.

Acknowledgments
Work supported by Horizon 2020 project no. 644869 (DICE). We are thankful to our colleague Michele Guerriero for his precious advice and expertise in model-driven software engineering.

9. REFERENCES
ABOUT THE AUTHORS:

Francesco Marconi is a PhD student at Politecnico di Milano. He received the M.Sc. degree in Computer Science from University of Illinois at Chicago and the LaureaSpecialistica degree in Computer Science and Engineering from Politecnico di Milano in 2013. He is participating in the DICE project and his current research interests include formal methods, temporal logic and data-intensive technologies.

Marcello M. Bersani is an assistant professor at Politecnico di Milano working in the area of Formal Methods, Temporal logic and Verification. He has been working in the DICE project to develop verification methods for data-intensive applications through various formalisms and techniques.

Matteo Rossi is an assistant professor at Politecnico di Milano. His research interests are in formal methods for safety-critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design, and of their application in urban mobility scenarios. He has previously participated in several national and international projects, including MADES, DICE, and IT2Rail.
A Multi-scale Modeling Approach for Systems of Systems Architectures

Ilhem KHLIF
ReDCAD Research Laboratory
University of Sfax, Tunisia
ilhem.khilf@redcad.org

Mohamed HADJ KACEM
ReDCAD, ISIMS
University of Sfax, Tunisia
mohamed.hadjkacem@redcad.org

Cédric Eichler
LIFO, INSA
Centre Val de Loire, France
cedric.eichler@insa-cvl.fr

Ahmed HADJ KACEM
ReDCAD, FSEGS
University of Sfax, Tunisia
ahmed.hadjkacem@fsegs.rnu.tn

ABSTRACT
Modeling correct software-intensive Systems of Systems architectures is a challenging research direction that can be mastered by providing appropriate modeling abstractions. For this purpose, we provide an iterative modeling solution for a multi-scale description of software architectures. We provide a visual notation extending the graphical UML notations to represent structural as well as behavioral features of software architectures. We define a step-wise iterative process from a coarse-grain to a fine-grain description. The intermediate iterations provide a description with a given abstraction that allow the validation to be conducted significantly while remaining tractable w.r.t. complexity. The iterative process involves both system-independent structural features ensuring the model correctness, and system-specific features related to the expected behavior of the modeled domain. We implement the resulting models describing structural and behavioral properties using the Eclipse platform. To experiment our approach, we deal with an SoS dedicated to the Emergency Response and Crisis Management System (ERCMS).

CCS Concepts
• Software and its engineering \rightarrow Software design engineering;

Keywords

1. INTRODUCTION
One of the major challenges in software engineering is to guarantee the completeness and correctness of software systems, especially as they have become increasingly complex. Mastering the complexity of software-intensive systems requires a combined effort for foundational research and new engineering techniques and approaches. Software-intensive systems are complex programmable systems exhibiting properties such as adaptive behaviour and dynamically changing structure. Software architecture serves as the backbone for building successful software-intensive systems. Architecture descriptions provide the conceptual abstraction for modeling complex software systems during development. We focused our research study on model-based design of software intensive systems. This study consider the question of how can we model software architectures to facilitate their validation at different description levels. That is to say how to describe the system with enough details for understanding without ambiguity and implementing in conformance with architects requirements and users expectations. For this purpose, we propose to consider different architecture descriptions with different levels of modeling details called the scales. Our objective is to provide solutions for modeling software architectures to facilitate their validation at different description levels for Systems of Systems (SoSs).

We propose a multi-scale modeling approach for software architectures. This modeling step is proposed to provide a standard visual notation for multi-scale architectures with the most appropriate standard language. Our objective is to propose an iterative design process through providing meta-models and models based on UML notations. The UML diagrams make it possible to present the structural properties as well as the behavioral properties of a multi-scale architecture. These diagrams are submitted to vertical and horizontal transformations. The intermediate models provide a description with a given abstraction that allow the validation to be conducted significantly while remaining tractable w.r.t. complexity. The validation scope can involve intrinsic properties ensuring the model correctness w.r.t. the UML description.

Our approach is taking advantages of the power expressive of standard visual notations provided by the semi-formal language UML 2.0 which allow to easier understand software architectures. Moreover, a software environment that supports the different features of this part of the proposed approach is implemented and integrated as an Eclipse plug-in (using EMF / Ecore / GMF / OCL in Ecore). Using this platform, designers can easily create their own graphic models of multi-scale architectures.
The paper provides a modeling solution for Systems of Systems software architectures including a visual notation extending the graphical UML notations. We define an iterative design process in order to allow the validation to be conducted while remaining tractable in relation to complexity. Different properties of correctness and traceability have to be maintained between the models at the different levels of iterations. Providing Rules for conducting such a process is our objective, which we implemented in visual modeling notations.

Our approach aims at validating software architectures at different description levels through horizontal and vertical refinement steps. The validation scope involves both system-independent structural features ensuring the model correctness, and system-specific features related to the expected behavior of the modeled domain. In order to formalize the multi-scale characterization process and to use it in the design process, we have chosen to follow our iterative modeling methodology that captures diverse perspectives of the performance modeling of software-intensive Systems Of Systems (SoSs). We discuss how our approach to modeling supports the specific characteristics of an SoS, demonstrating how we may obtain correct models within a SoS. In order to show the viability and usefulness of our solution, we experiment our approach to deal with the case study dedicated to the Emergency Response and Crisis Management System (ERCMS).

The remainder of the paper is organized as follows. We describe the approach in section 2. Section 3 illustrates the case study. In section 5, we present a survey of related work. We conclude and outline some perspectives in section 6.

2. AN ITERATIVE MULTI-SCALE MODELING APPROACH FOR SOS

This work proposes a visual approach which aims to help the architect to design correct SoS architectures that respect a set of structural constraints. To ensure correctness, it is necessary to study the consistency of the system as a whole. The architecture must be left in a correct state by maintaining its conformity to its architectural style.

An architectural style defines a vocabulary of component and connector types, and some constraints on how they can be combined. So, an architecture can be considered as correct if it is an instance of its architectural style (i.e., it uses the defined component types and it preserves all the defined constraints). Our proposed approach supports the correct modeling of SoS architectures.

The proposed design approach is founded on UML notations. The diagrams are submitted to vertical and horizontal transformations for refinement; this is done to reach a fine-grain description that contains necessary details. The model transformation ensures the correctness of UML description, and the correctness of the modeled system [12]. We present the multi-scale approach by a two dimensional array describing vertical and horizontal refinements. We start by modeling the first scale, which is defined by a UML component diagram. This diagram is refined, through model transformation operations, until reaching the last scale [14]. UML provides a formal language, the Object Constraint Language (OCL), to define constraints on model elements. Our approach supports model transformation and validation of UML models with OCL constraints. The OCL language is powerful and allows the expression of constraints that cannot be described using description logic.

Figure 1 shows the general description of our approach by presenting the different used tools.

The first phase consists of the design and development of the Eclipse plugins. Thus, we chose the Eclipse environment to create plugins that allow structural modeling and behavioral modeling. Component diagrams are very useful in analysis and design, allowing us to define the components we need before developing and implementing them. They allow to express the structural aspect of models. Vertical and horizontal transformations are applied to the diagrams to obtain a fine grain description through successive refinements. Sequence diagram allow to express the behavioral aspect of models. Indeed, the use of this diagram allows us to identify all the possibilities of a complex behavior with the graphical representation of the technology of exchange of messages between the objects.

The second phase ensures validation of the model with the OCL language. Consequently, we have defined constraints on the elements of the model using OCL language. Tools are used to check and validate OCL constraints such as the OCLinEcore Editor to validate the invariants and the USE tool to validate the pre/post conditions. This part was detailed in our previous work [15].

The third phase allows XML files to be generated using XMI, which transforms the models into XML. So we can get the XML file of the templates that we created.

The multi-scale modeling approach promotes reuse of existing notations and languages: component diagrams for structuring the problem, and describing the overall software system in terms of components and interactions, sequence diagrams to describe the behavior of each component. An iterative modeling process involves both structural properties ensuring the model correctness, and specific properties related to the expected behavior of the modeled domain. We illustrate, in (Figure 2), a step-wise structural and beha-
vioral iterative process from a coarse-grain description to a fine-grain description.

We follow a top-down strategy where a model of the larger scale is built and refined with details for smaller scales until reaching all levels of details. We define a first scale architecture. In general, a scale i represents coarse grained components, such i ∈ [0, n] where n corresponds to the depth of the hierarchy defined in the SoS model. For i = 0, we obtain the first scale. Then, it is refined by adding the next scale components. The obtained architecture is refined in turn until reaching the last scale, i.e., where all system components are defined. The transition between scales is implemented following a rule-oriented refinement process.

2.1 Structural Properties

Multi-scale modeling is defined as an incremental process where we constantly modify and refine software systems descriptions. We propose a correct by design approach where properties are checked and preserved at different description scales. We model the first scale by a given coarse-grained description using a UML component diagram. This model is refined until reaching a fine-grained description representing the necessary modeling details.

We, first, elaborate an initial abstract architecture description from the user requirements. In the first iteration, application requirements are specified (a unique component C0 is identified) This is the beginning of the traceability. A new iteration is required for providing details on the application. Three component types named C1, C2, and C3 that are interconnected are added (Figure 3).

The second iteration is helpful for checking that, at the next scale, the components identification is preserved, as we keep the traceability of a component from one scale to another. This notation is used for identifying a component Cm, where m represents a cursor on the current component (m ≥ 0). It can be decomposed in the next scale. As illustrated in (Figure 3), the component C1, will be refined with two composites in the next scale identified as follows C1.1, C1.2. The component C2 will be refined with two sub-components named C2.1 and C2.2. Similarly, the component C3 is composed of the sub-components named C3.1 and C3.2.

We are especially interested in refining an enabling architectural style for component-based systems: the Publish-Subscribe style. The strength of this event-based interaction style lies in the full decoupling between producers, and consumers. This decoupling is provided by the event dispatcher. The approach may be applied in many different domains and across different architectural styles, for example Service Oriented Architecture (SOA), Client-Server, etc.

Then, in the last scale, roles are associated with components such as “Event-Dispatcher”, “Producer”, “Consumer”, “Producer-Consumer”, etc, and thus connections between them are established. A possible configuration to refine the interaction (link) between the components C1 and C2 is illustrated. If the component C1 is a “Producer” and the component C2 performs the role of an “Event-Dispatcher”, the link between C1 and C2, in the first scale, will be decomposed into a simple assembly connection, in the next scale, extending from the source C1.1 to the target C2.1. As a link is divided according to its identifiers, then a trace of the link decomposition is added. Moreover, the interaction between the two components C2 and C3 will be refined at the next scale as follows: if the component C3 is a “Consumer” and the component C2 is an “Event-Dispatcher”, the link between C3 and C2 will be decomposed into a simple assembly connection extending from the source C2.1 to the target C3.1. During the iterations, we have to check an intrinsic property ensuring the model correctness w.r.t. UML description (interface compatibility). We preserve the multi-scale architecture consistency stating whether the components of the architecture are correctly typed and well connected (each interface is connected to a compatible one). From a structural viewpoint, an architecture is complete if all its required interfaces (related to the consumers C3.1, C3.2) are connected to compatible provided ones (related to the dispatchers C2.1 and C2.2).

In addition, we preserve the model traceability from one scale to another by decomposing links, at the abstract scale, and refining them, at the next scale, to show possible connec-
2.2 Behavioral system-specific features

The aim of the multi-scale modeling is to study the required behavioral properties of the considered application. The application is initialized (at the first scale), and after successive iterations, the sets of components and interactions among them are identified in a way that supports the required behavior of the abstract application level. After identifying interactions, we consider giving structured representations of components behavior as a series of sequential steps over time. We describe the specified behavior of an application using the UML sequence diagram (Figure 4). In the first scale, the whole application is presented as a black box to illustrate the System Sequence Diagram (SSD) named “C0”. The main issue here is to secure the message transmission and how elements cooperate to ensure correct information propagation. Several events may refine an abstract event: A single message (M1) between actors from a coarse-grained description scale is translated into a set of messages (M1,1 and M1,2) at the next scale, or the content of translated messages depends on earlier received message.

The sequence diagram, represented in Figure 4, specifies behavioral features of the publish-subscribe architecture. When the Producer-Consumer component named C1 sends a message (M1 :T1) to the Event dispatcher component C2 at the first scale, the dispatcher tracks this message and, replies by sending an acknowledgement message (M2 :T1). At the next scale, those messages will be refined into a parallel sequence of messages and keep track of the type of message sent or received in the abstract scale. For example, the typed message (M1,1 :T1) sent from C1 to C2 is refined into two messages having the same type: the message (M1,1 :T1) is sent from C1,1 to C2,1 and (M1,2 :T1) is sent from C1,2 to C2,2.

We propose to check two properties to describe the behavior of a multi-scale architecture. First of all, a behavioral scale description adds information that reveals the order of interactions among the elements opportunities for concurrency time dependencies of interactions. We have to preserve the traceability property from a vertical description scale to ano-
ther. This property deals with the event deadline and shows
time sequences explicitly, making it easy to see the order
in which event must quickly occur. An event will be refi-
ned from a vertical scale to another. This iteration allows to
preserve the event structure until reaching the fine grain de-
scription. The traceability property is ensured through both
the identification and the type of exchanged messages.

Our approach is based on a multi-scale modeling that helps
to automate the construction of correct design architectures.
So, we need to specify the software architecture model that
describes the software components, their composition and
their interactions. In fact, each model is represented as a set
of scales, and each scale denotes a set of architectures. Follo-
wing our approach, the designer starts by modeling the first
scale architecture which is refined to give one or many ar-
chitectures for the next scale. Then, these architectures are
refined in turn to give the following scale architectures and
so on until reaching the last scale. The transition between
scales is ensured by applying both structural and behavio-
ral refinement rules. After constructing the architectures of
software architecture model, we apply the relation between
the two models in order to obtain model-based architectures
with different description levels.

3. APPLICATION TO THE ERCMS SOS
To experiment our approach, we deal with an SoS dedicated
to the Emergency Response and Crisis Management Sys-
tem(ERCMS) [13]. The ERCMS involves structured groups
of participants who are cooperating to achieve a common
mission (e.g. save human lives, fight against a fire, etc).

3.1 System of Systems : Motivation
At design level, an SoS is the composition of components
(constituent system), themselves systems, that are large scale
and independent, separately discovered, selected and com-
piled to form a more complex system [20].

3.1.1 SoS defining features
A set of characteristics presented by SoSs distinguish them
from complex systems. The intrinsic characteristics of an
SoS are proposed by Maier [17] : Operational Independence : The
constituent systems operate independently of each other,
each having its own mission, Managerial Independence : The
constituents of an SoS are separately integrated but man-
aged their own resources independently. Evolutionary De-
velopment : As a consequence of the independence of the
constituent systems, an SoS as a whole may evolve over
time to respond to changing characteristics of its surroun-
ding environment, constituent systems or of its own mission.
Emergent Behavior : The SoS is capable to deliver new func-
tions that are obtained from the composition of its consis-
tuents rather than from a singular constituent. Geographic
Distribution : the constituent systems are physically decou-
pied and may be widely geographically distributed in such a
way the interaction between them is limited to information
exchange.

3.1.2 Classification of SoSs
Based on these characteristics, SoSs can be classified as :
directed, collaborative, virtual, and acknowledged. Directed
SoS is a set of centrally managed systems that have been de-
veloped to fit specific purposes in the SoS, as they operate
under tight subordination [17]. Collaborative SoS in which
there is no central management and constituent systems vo-
luntarily agree to fulfill central purposes. They are distinct
from directed SoS in the inability of the central manage-
ment to run the system [17]. Virtual SoS lacks a central
management authority and an agreed purpose for the SoS
[17]. Finally, acknowledged SoS in which there are recogni-
zed objectives, a designated manager, and changes in the
systems are agreed on collaboration between the SoS and the
systems [21].

3.1.3 Case study : an acknowledged SoS
This work focuses on modeling the ERCMS SoS. This is
an acknowledged SoS as it is centrally managed and ope-
rate under loose subordination, and the constituents retain
their operational independence. This SoS presents the follo-
wing five characteristics : Operational independence : The
ERCMS can use information provided by a manager but will
operate normally its mission even if this information is
not available. Managerial independence : Independent man-
egerial areas with own action strategies for SoS (rescue and
repair ). Evolutionary development : The SoS evolves to ac-
commodate changes in this environment, for instance when
safety condition evolves. Emergent behavior : Detecting an
emergency critical situation and immediately reacting is an
emergent functionality provided by the cooperative work of
the manager, the supervisor, the coordinator and the inves-
tigators. Geographic distribution : The ERCMS constituent
systems are largely distributed in the investigation area. We
distinguish the following participant roles : a supervisor, co-
ordinators, and investigators. The supervisor manages the
whole coordinators and each coordinator supervises a sec-
tion of investigators. The coordinator leads a section of in-
vestigators and is expected to collect information received
from investigators and diffuse them to the supervisor. The
investigator acts to help, rescue and repair.
3.2 ERCMS Model

The architecture of a SoS is characterized by a hierarchical description of systems containing components. So, we define the relation between scales by a hierarchical decomposition. In other words, a fine grain description scale represents the constituent systems of the previous scale systems: The abstract scale represents coarse grained components. The refined scale represents the sub-components of the abstract scale components. The last scale describes the constituent elements and how they are connected with respect to the Publish-subscribe style. In order to construct a correct multi-scale architecture, we follow a top-down strategy. Instead of modeling the whole SoS architecture and verifying it with respect to the defined system constraints (i.e., its conformance to its architectural style), we rather ensure the correctness of each scale architecture obtained by a refinement process. Therefore, at first, we define the architectural style by identifying the different components types (constituent systems) as well as their structural constraints (connection constraints). Then, we define the first scale architecture. Then, this architecture is refined automatically by adding the next scale components. The obtained architecture is refined in turn until reaching the last scale (i.e., where all system components are defined and connections established between them). Regarding the refinement process, it is performed by applying specific rules that should respect the defined architectural style (or system constraints) ensuring in this way the correctness of the obtained scale architectures. We applied successive iterations and checked the previously described properties. We obtained then the folowing results: in the abstract scale, we define the application named “ERCMS”. In fact, participants in the ERCMS are represented (in the first scale) by their components, named supervisor, coordinator, investigator, and manager. Those participants communicate with each other via the manager.

Those relationships are represented as UML associations. In the next scale, the components are refined and specified with an associated role as shown in Figure 7. The supervisor of the mission plays the role of a component consuming events and will be defined as “Consumer”. The supervisor’s functions include monitoring and managing actions to be achieved by coordinators and investigators. In the same way, we specify the roles of each type. A manager promotes the roles “Event-dispatcher”. The manager waits for data from the coordinator who analyses the current situation of the mission, diffsus collects data to the coordinator, and sends reports to the supervisor. An investigator and a coordinator have the role of “Producer”. The propagation of events between participants is mediated through a network of dispatchers called Manager. In fact, coordinators and investigators communicate with each other symmetrically as peers, adopting a protocol that allows a bidirectional flow of communication called the “acyclic-P2P” topology. Connections are established according to this topology related the style “Publish-Subscribe”.

During the iteration process, we apply the link decomposing rule with respect to the component role: if a component plays the role of “Consumer”, the link between C2 w.r.t. “Supervisor” and C1 w.r.t. “Manager” in the first scale will be transformed into a single assembly connection in the next scale, starting from the source C21 w.r.t. “Supervisor” to the target C11 w.r.t. “Manager” having the role of “Event-Dispatcher”.

We refine the coordinator type by inserting three new sub-components. We can distinguish the robot coordinator, the plane coordinator, the firemen coordinator that manage professional actors. We check that every coordinator produces only events (Message with type Coordination noted M : Coordination) to the managers. The investigator’s functions include exploring the operational field, observing, analyzing, and reporting about the situation. Investigators also act for helping, rescuing and repairing. We refine the investigator type by inserting, first, three sub-components (Robot-Group, Fireman-Group, and Plane-Group). We check that every investigator can produce only events (Message with type Information noted M : Information) to the managers, and managers have to support only events coming from coordinators and investigators. A manager must transmit received events to the supervisor that must collect messages with type Information and Coordination. To ensure the principle of information dissemination, the produced information (M : Coordination, Information) must reach the subscribed consumer (the supervisor). To guarantee this property in the case of a network of dispatchers, we have to check that “Manager1” and “Manager2” are interconnected directly. Moreover, managers communicate together then it is necessary that the information coming from the first reaches the second. So, the communication has to be bidirectional. We illustrate this constraint through the double assembly connection between the two managers. In the acyclic peer-to-peer topology, dispatchers communicate with each other symmetrically as peers, adopting a protocol that allows a bidirectional flow of subscriptions. Based on this property, the supervisor as an event consumer receives only once the same message (M : Coordination, Information) arriving from the coordinators and the investigators to the managers.

![Figure 6: The ERCMS model](image-url)
3.3 ERCMS system-specific properties

In Figure 7, we present the UML sequence diagram to demonstrate the behavior of the different elements. The sequence diagram shows the instances participating in the interaction having two dimensions. The vertical dimension represents time. The horizontal dimension represents different objects w.r.t. the behavior of the smart home components.

We illustrate the abstract scale using the System Sequence Diagram (SSD) named “ERCMS” to show the whole system (as a black box).

The first vertical iteration allows to describe the objects Supervisor, Coordinator, Investigator, and Manager and the messages that they pass are shown in the sequence diagram. An object of class Coordinator starts the sending of a message to an object of class Manager while detecting a crisis situation. This object in turn responds by an acknowledgment message to the Coordinator. The manager sends the information to the object Investigator that will respond immediately and returns a message describing the situation after the intervention. We can distinguish different steps during the mission. We give the two most representative event.

An exploration event is concerned with the localization and the identification of the crisis situation. An Action event is needed after the identification event. Initially, all investigator groups are in the exploration phase. Investigators provide continuous feedbacks to the coordinator via the managers. The coordinator sends continuous feedbacks to the supervisor via the single access point in the network of dispatchers called “Manager1” and “Manager2”. When an investigator discovers a critical situation, its group has to be informed to face this new situation. It moves to the action phase. The investigator that discovers the critical situation keeps sending observations and analysis to the manager. The manager diffuses to the Coordinator and to the other investigators of the group. Successive iterations allow to refine messages as follows: The message presented at the abstract scale is refined (in a vertical way) into three messages between the three components of the monitoring system (at the next scale). We obtain the result that preserves the traceability property from a scale to another. This property deals with the event deadline. When detecting a received event (e.g., fire), we need to guarantee that the event is transmitted to the manager. The system has to store the relevant mission information, determine the investigators that could accomplish the mission, establish contact with at least one of them and propose the mission to accomplish. We propose to illustrate only one refinement of the message M1 :T1 into a parallel sequence of messages M1.1 :T1, M1.2 :T1, and M1.3 :T1. The manager decides whether a “rescue mission”, a “plane transport mission”, or a “remove obstacle mission” should be created for a given mission description. The details of a Mission execution depend on the type of Mission. Coordinator receives updates on the mission status from the manager. Each Mission continuously informs the coordinator when it is in execution or finished. Coordinator receives new information about the crisis from the manager. If the supervisor receives a final report (by the manager), it stops waiting for change notifications, and waits for the coordinator to close the crisis resolution session. After receiving the final report, the supervisor waits for the coordinator to close the file for the crisis resolution.

4. VALIDATION

We are interested in implementing a tool supporting our multi-scale modeling approach. In fact, a new direction for research is to perform model validation tasks. Indeed, any error in the modeling of the software architecture may damage the system functioning. Thus, the validation consists essentially in checking whether the modeling of the software architecture is error-free. To overcome these drawbacks, we tried to follow a validation process throughout the development of plug-ins under the Eclipse platform to detect any constraints not respected.

To validate our approach, we propose a tool for modeling multi-scale software architectures. In the first phase, we model the structural and behavioral aspect with UML diagrams. Thus, these models are correct by design. In the second phase, we check the architectural properties through the OCL constraints. We develop plug-ins, based on Eclipse frameworks [3], i.e., Graphical Modelling Framework (GMF) [1], Eclipse Modelling Framework (EMF) [2] and Graphical Editing Framework (GEF) [4]. Several diagrams are available in the plug-in. We model the component diagram, and the sequence diagram. The Eclipse Modeling Framework (EMF) is a set of Eclipse plug-ins which allows the developer to create the meta-model via different means such as UML. First, we create the EMF project which consists of two parts: the.ecore and the genmodel description files. The.ecore file contains the information about the defined classes (Component, Port, Interface, Connector, etc). Ecore which is essentially the class diagram subset of UML which is based on the Object Management Group’s (OMG) Meta Object Facility (MOF) specification. Second, we create the GMF project which provides a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF. We use the GMF tool to create and visualize the content of the created models. Third, we identify some OCL invariants for capturing structural constraints. We use OCL tools with Eclipse for encoding constraints, checking constraints, and obtaining feedback from the checking process. The diagram editor is a tool where diagrams
1. Structural Modeling

2. Behavioral Modeling

3. Evaluating constraints using OCL console

4. Generating XML file

![Figure 8: The implemented Multi-scale description approach (Eclipse Plugin)](image)

can be created to models. Graphical elements can be picked up from a tool palette and created in the Diagram editor pane in a drag-and-drop way. Elements of the palette are listed under Nodes and Links elements. The Property Editor can be used for changing properties of the object selected in the diagram editor pane. Property elements vary depending on the type of the chosen object. We illustrate the diagram editor of the multi-scale approach with an illustration of the model example. The model can be enriched with OCL constraints that are defined on the model (using an OCL meta-model) and can then be verified for model instances (the smart home case study) of the model. After modelling a design pattern, the plug-in generates an XML file describing it. We implement a tool supporting our approach as an Eclipse plug-in (Figure 8).

5. RELATED WORK
Considerable research studies have been presented on the description of software architectures. Our work is related to recent approaches handling formal aspects of UML and other object-oriented methods. An architecture refinement is defined [19] as an incremental process which transforms an architecture across levels of abstraction using a series of refinement maps. The maps must satisfy the correctness property, which ensures that all decisions made at a given abstract level be maintained at all specific levels and that no new decisions be introduced. A multi-scale description is introduced [5] to specify behaviours and properties of the system across multiple scales in order understanding without ambiguity the system and mastering the description details. Multi-level modeling approaches [23] have been proposed to represent the different abstraction levels. The basic idea of multi-level modeling is to explicitly represent the different abstraction levels to enable a clear encapsulation of problems in different levels to tackle them independently. Low level details are abstracted for higher levels, and therefore complexity is reduced, thus achieving a clear and coherent design. The refinement allows to provide more details between levels. Baresi et al. [6] presented a UML based approach and proposed formal verification and validation of embedded systems. The approach is implemented using the “CorrettoUML”: a formal verification tool for UML models. Brosch et al. [9] proposed a meta-model for specifying adaptability characteristics in a software product line. This model is expressed on different levels. The architectural level uses composite components to encapsulate subsystems and enable their replacement. The authors presented a tool support that allows the architects to design the architecture with UML models. Bryans et al., in [10], presented a model-based approach to assist in the integration of new or modified constituent systems into a complex system. The authors defined two levels for system composition, the high-level structural view that considers the connections within the system, and the low-level behavioral view that deals with
the behavior of contractual specifications. They treated an industrial case study for modeling Audio/Video system.

Other studies have focused on the architecture refinement concept. Oquendo et al. [22] described II-ARL, an architecture refinement language based on the rewriting logic. The core of II-ARL is a set of architecture refinement primitives that supports transformation of architecture descriptions. The authors formally modeled the stepwise refinement of software architectures. Rafe et al. [18] proposed an automated approach to refine models in a specific platform. For each abstraction level, a style should be designed as a graphical diagram and graph rules. In their approach, the model is designed by the rules of graph transformation.

Other research studies have been proposed for the specification of software systems using formal methods. Model verification activity [24] is performed to ensure the correctness of model. Formal verification means that any errors found in the design of the system should be corrected.

Kim et al. [16] present an approach to describe design patterns. This work is supported by tools using the Eclipse Modeling Framework (EMF) and the meta-model is transformed to the formal Object-Z using model transformation techniques in order to specify structural features. Behavioral features of patterns are also specified using Object-Z.

Ben Younes et al. [7] proposed a meta-model transformation between UML Activity Diagram and Event B models. A formal framework is defined to ensure the correctness of the proposed transformations, and the event B method is used for the formal verification of applications.

We can note that the previous research activities deal only with structural features during the design of the architecture. They do not take into account the respect of behavioral features to validate the architecture. Whereas, in our work, we deal with both structural and behavioral features.

In [11], Gassara et al. proposed a multi-scale modeling methodology for software System of Systems (SoS) using the formal technique of bigraphical reactive system. They implemented the transition between scales following a rule-oriented refinement process. To implement their solution, they proposed BGMTE, a tool for bigraph matching and transformation. It allows to execute the application of a reaction rule on a given bigraph to be rewritten. BGMTE is also based also on GMTE, a tool for graph matching and transformation, for executing the encoded rule on the encoded graph.

Boujbel et al. [8] presented a domain-specific language, called MuScAdel (MultiScale Autonomic Deployment Language), dedicated to autonomic software deployment of multi-scale systems. MuScAdel supports the expression of multi-scale deployment properties and component constraints. They generated the MuScAdel code which is compiled, giving the set of components properties However, they did not consider the dynamic aspect of multi-scale systems.

Other research studies have been presented on modeling software architectures for SoS. An adequate representation of SoS software architectures is crucial for the success of such systems. In [25], Mordecai and Dori developed a design approach to model interfaces and interactions among SoS constituents. They proposed a framework for SoS integration, to facilitate the transition from the constituent systems' view to a consolidated integrated model. Bryans et al., in [10], presented a model-based approach to assist in the integration of new or modified constituent systems into an SoS. The authors defined two levels for SoS composition, the high-level structural view that considers the connections within the SoS, and the low-level behavioral view that deals with the behavior of contractual specifications. They treated an industrial case study for modeling Audio/Video SoS.

We analyze that several studies have been performed on the modeling of multi-level architectures based on UML. These semi-formal approaches did not, however, include the concept of refinement. Although formal techniques and, more specifically, works based on graph transformations allow the architecture refinement, they require certain expertise in mathematics for architects. Moreover, only few studies have provided a clearly defined process that takes the compatibility between different description levels into account, a challenging condition for the multi-level description of software architectures. Model-based methods have addressed significant challenges in SoS Engineering. Semi-formal models are used in the architectural description of SoS. This representation has advantages, mainly with regard to comprehension, and can help to clarify areas of incompleteness and ambiguity in specifications. In this study, we have considered that a given modeling level can be described by both vertical and horizontal scales. This work aimed to provide solutions for modeling multiple different levels of description of the same modeling level through the scales. Thus, we applied our model-based approach for describing multi-scale architecture for SoSs, defining both the structure and the behaviour of the constituent systems of an SoS and interactions between them. The validation scope involved both intrinsic properties ensuring the model correctness w.r.t. UML description (e.g., interface compatibility), and generic properties (e.g., correct information propagation).

6. CONCLUSION

In this paper, we have presented a multi-scale modeling and specification approach for software architectures. We have proposed UML notations to represent the structure and the behavior for modeling different description scales. We have considered traceability properties during the iterative process. We have proposed a correct by design approach based on UML notations ensuring the correctness of the SoS architectures. This approach follows a refinement process for the transformation between coarse-grained and fine-grained descriptions. Considering that the first scale given by the designer is correct, this refinement is performed by adding lower scale details. It is implemented by applying correct rules (i.e., preserve the system constraints). Hence, the obtained scale architectures are in turn corrects. We have presented the application of our approach to the ERCMS SoS to validate structural and behavioral iterative features. Finally, we have presented some research studies discussing multi-level modeling for software architectures using semi-formal and formal methods. Currently, we are working on the formal verification of architectural properties to prove their correctness. Moreover, we are working on extending
the state of the art related to the area of SoS. In future work, we expect to apply the multi-scale approach to other use-cases for modeling complex systems architectures such as smart cities (virtual SoS) and validating on OM2M platform. We expect to strengthen the formal foundations of the approach, by generalizing to abstract notations (bi-graphs) and dynamic architectures.

7. REFERENCES

ABOUT THE AUTHORS:

Ilhem Khlif is a PhD student in Computer Science in the Faculty of Economics Sciences and Management of Sfax-Tunisia. She obtained a Master Degree in Computer Science and Multimedia in 2012 from the Higher Institut of Computer Science and Multimedia of Sfax. She is a member of the Laboratory on Research on Development and Control of Distributed Applications (RedCAD). In her Master Degree project, she proposed a model-based design approach for modeling and describing multi-scale software architectures. More details are available at http://www.redcad.org/members/ilhem.khlif/.

Mohamed Hadj Kacem is an Associate Professor in Computer Science at the Higher Institut of Computer Science and Multimedia of Sfax, University of Sfax, Tunisia. He is a member of the Laboratory on Research on Development and Control of Distributed Applications (ReDCAD). He obtained a PhD in computer science in 2008 from the Paul Sabatier University of Toulouse-France and the Master degree (DEA) in Computer Science from the Faculty of Economic Sciences and Management of Sfax-Tunisia in October 2002. His current research areas include Model Driven Architecture, Component-based Architectures, Architecture-based Software Evolution, Formal Design of Software Architectures, Service Oriented Architectures, and Eclipse for Modeling. More details are available at http://www.redcad.org/members/hadjkacemm/.

Cédric Eichler is an associate professor at INSA Centre Val de Loire and is currently head of the Security of Data and Systems research team at LIFO. He obtained a PhD in computer science in 2015 from the University of Toulouse III. His main research interests include formal models of dynamic systems and their transformations, self-adaptive systems, and secure and privacy-preserving evaluation of such systems.

Ahmed Hadj Kacem received his MS in Computer Science and his PhD in Computer Science from UPS, University Paul Sabatier Toulouse-III, respectively in 1991 and 1995. He joined the University of Sfax (Tunisia) as an Assistant Professor in 1997, then as a full Professor in 2007. Since August 2014, he is the Dean of the Faculty of Economics and Management of Sfax. He participated in the initiation of many graduate courses. He is a member of the ReDCAD Research Unit (http://www.redcad.org). His current research areas include formal design of multi-agent systems and adaptive software architectures for distributed systems. He is an IEEE senior member, founder & chair of the Computer Society Chapter during 2009. More details are available at http://www.redcad.org/members/ahadjkacem/.
An Efficient Visual Fiducial Localisation System

Peter Lightbody
Lincoln Centre for Autonomous Systems
University of Lincoln
United Kingdom
plightbody@lincoln.ac.uk

Tomáš Krajník
Artificial Intelligence Center
Czech Technical University
Czechia
tomas.krajnik@fel.cvut.cz

Marc Hanheide
Lincoln Centre for Autonomous Systems
University of Lincoln
United Kingdom
mhanheide@lincoln.ac.uk

ABSTRACT
With use cases that range from external localisation of single robots or robotic swarms to self-localisation in marker-augmented environments and simplifying perception by tagging objects in a robot’s surroundings, fiducial markers have a wide field of application in the robotic world. We propose a new family of circular markers which allow for both computationally efficient detection, tracking and identification and full 6D position estimation. At the core of the proposed approach lies the separation of the detection and identification steps, with the former using computationally efficient circular marker detection and the latter utilising an open-ended ‘necklace encoding’, allowing scalability to a large number of individual markers. While the proposed algorithm achieves similar accuracy to other state-of-the-art methods, its experimental evaluation in realistic conditions demonstrates that it can detect markers from larger distances while being up to two orders of magnitude faster than other state-of-the-art fiducial marker detection methods. In addition, the entire system is available as an open-source package at https://github.com/LCAS/whycon.

CCS Concepts
*Computing methodologies* → Object detection; Tracking; Vision for robotics;

Keywords
Fiducial Markers; Swarm Robotics; Necklace Code; Visual Tracking

1. INTRODUCTION
Although initially intended for Augmented Reality (AR) applications, fiducial-based visual localisation systems are broadly utilised in a number of areas throughout the field of robotics where a robust and efficient full pose vision-based estimation is required. Typical applications of such marker-based systems include swarm and bio-inspired robotics [2, 3, 11], which requires reliable localisation of a large number of robots from an external camera (see also Figure 1(e)), visual-servoing that requires highly precise robot motion [21, 26], and semantic scene understanding [7], in which the scene objects are tagged with the fiducial markers to mitigate the limitations of general vision-based object recognition.

Both within these applications and more generally, visual fiducial marker detection and tracking systems must ideally fulfil to a high standard, the following requirements:

* Robustness: Markers must be robustly detectable in adverse conditions such as when moving rapidly, when at a considerable distance or when under varying lighting conditions.
* Distinguishability: For fiducial markers to be used for applications in which a single marker is not sufficient, either because several robots need to be tracked in parallel or several features in the environment need...
to be identified simultaneously, it needs to robustly identifiable and distinguishable to the vision system. However, the number of markers required in these applications often varies considerably across application domains. The tracking method must, therefore, be able to scale accordingly to the requirements imposed by the specific application or scenario.

- **Economic Feasibility:** To ensure the system is universally accessible, it should, ideally, utilise markers which are cheap and easily producible in large quantities, such as being printable on paper. This has the added benefit of making the system not only cheap to set up, but also cheap to use in conditions which might otherwise deteriorate the markers such as use with drones or in adverse weather conditions. The system should also utilise standard, off-the-shelf sensor(s) such as RGB and grey-scale cameras. All of this, together with freely-available, open-source software, makes them customisable, and thus, minimises the cost for developers and researchers alike.

- **Precision:** Fiducial markers’ fundamental purpose is to provide a precise position of a marked object within an observed scene. This can come in the form of a 2D location in an image or a 3D position in the scene. Apart from identifying a 3D position, orientation information is often also desirable. Thus, most markers often have properties that allow the estimation of their full 6 degrees of freedom (DoF) pose.

In this paper, we propose a novel tracking system that can generate suitable markers which can be easily printed on paper, along with an integrated software component that addresses the above requirements to a very high standard. This is achieved by extending an open-source detection system for circular markers called WHYCON [12] by adding a novel encoding based on the concept of Binary Necklaces [20], which we shall refer to as WhyCode. Necklaces are a mathematical concept of combinatorics providing a generator for rotation invariant, uniquely identifiable patterns that can scale to a theoretically infinite number of individual markers, similar to the one shown in Fig. 1(d). The resulting markers are robust and efficiently detectable in the environment and also allowing for discrimination between individual markers using the Necklace coding. With our extension of the original system we now present a 6-DoF fiducial marker system. The performance of the proposed system is demonstrated through a range of experiments which compare the pre-existing WHYCON system against the new WhyCode method, in addition to the frequently used ARTAGS and APRILTAG fiducial marker detection systems.

2. RELATED WORK

In response to the need for vision-based markers within robotics, discussed above, several marker-based tracking and identification methods have been developed. These can be categorised into three groups: active markers, passive reflective markers and passive markers, depending on the type of markers that each system uses.

In vision based systems, active markers contain a source of light that can be tracked – this is often achieved using an infra-red or ultraviolet LEDs, which are easy-to-detect by cameras, but unobtrusive to humans. These systems also achieve very low false positive rates, because the active markers can emit unique identification codes, which makes them almost impossible to confuse with other objects. Some active marker systems are also capable of estimating the orientation of a marker, but this is normally achieved by combining the positions of a number of markers to calculate the orientation, rather than retrieving the orientation of a single marker. These systems, however, do offer sub-millimetre precision for a marker’s location and their multicamera setups offer extremely accurate tracking information [22].

Alternatively, passive reflective markers are a widely used approach within the field, with the most common example being the commercial motion capture system ViCon [24], which combines high-resolution and high speed cameras that utilise strong infra-red emitters. Systems like this also enable tracking with sub-millimetre precision but unlike active marker systems, their fiducials do not contain electronic components. They are instead often made from, or coated with, materials that are highly IR reflective, allowing the IR cameras to easily pick out the markers from a scene even from high distances. This has the benefit of making the markers cheaper and easier to deploy than active ones, however, passive systems are problematic to deploy in areas under direct sunlight. Although both active and passive reflective based systems offer a solid ground truth, these approaches remain very costly and are therefore not always an appropriate solution.

These issues, however, have motivated the creation of a variety of alternative low-cost tracking systems which focus on the final category of fiducial markers: passive vision-based tracking. With many of these newer methods utilising simple planar patterns, it is not only the cost that is significantly lowered, but also the difficulty of use and set up time.

One of the most well-known passive markers is the QR Code. This marker is predominantly designed to store information, such as text and URLs, and consists of a two-dimensional matrix barcode which encodes data in a pattern of black and white squares. In-built error correction codes allow the information to be correctly read, even if the marker is partly damaged, although these characteristics do restrict the range and angles from which the codes can be read. Consequently, although there is the potential to use such markers as part of a larger tracking system, their design makes them less suitable for tracking than both the methods discussed below and the proposed method.

Examples of passive markers which are more focused on the tracking element are the augmented-reality markers. These, although conceptually similar to the QR code mentioned above, are designed to encode far smaller data payloads and often use the ATag [8] and ARToolKit+ [25] software libraries.

The current ARTAGS developed from these software libraries utilises a square box fiducial marker which encodes information through the use of a large 2D black and white bar code. The real time performance of the system, coupled with its
Another augmented-reality alternative that will also be compared is the APRILTag [18] system. Using the square marker design with a 2D bar code, the APRILTag also stems from a lexicographic coding system [23] and is therefore able to be detected at both short and long range. Computational simplicity is, however, sacrificed.

The success of square markers within this field is evident, however, the use of circular markers is quickly becoming a regular occurrence in many applications. This is largely due to the need to counter the expensive operation required to correct the shifting of the centroid of a square marker under perspective transformation. Many systems, including the SyRoTek e-learning platform [14], which uses ring-shaped patterns with binary tags and [26], a planar pattern which consists of the letter 'H' surrounded by a ring, utilise circular markers due to the less expensive centroid operation. In the latter system, the pattern is first detected using adaptive thresholding and is later processed for connected component labelling. To establish whether the marker has been correctly tracked, its geometric properties are tested and the false matches are discarded. A Canny edge detector and ellipse fitting method are then applied to the positive matches.

Another system built upon the same methodology as the ARToolkit and ARTags is ArUco [10], which boasts a robust ID system with an error correction technique that can handle up to 1024 individual codes. The detection process within the ArUco system combines contour extraction and code identification with the aforementioned adaptive thresholding step and thus can determine the extrinsic parameters of the marker using the intrinsic camera parameters.

Finally, a system relatively similar to the proposed markers, that comprises a number of concentric circles broken into several angular regions and coloured either black or white, is the TRIP localisation system [6], which is able to distinguish between 39 patterns. Similarly to the ArUco system mentioned above, the TRIP localisation system also appropriates an adaptive thresholding method, with the system as a whole extracting the edges of the markers and processing the edges which correspond to the circular border of the ring patterns. The main disadvantage accompanying this system is the computational cost, as the adaptive thresholding and ellipse fitting are computationally expensive. On the other hand, the system’s ability to achieve a precision of between 1% and 3% of relative error may be said to counteract this disadvantage.

The aforementioned methods are widely considered to be the state-of-the-art methods currently existing within the field. Despite this, the real-world performance and low computational cost of the method proposed here makes it potentially superior in several application domains. The ability to expand the recognisable patterns by incorporating a scalable identification code makes the proposed method preferable in cases where one needs a computationally efficient method to detect, localise and identify a larger number of fiducial markers.

### 3. CONCEPT

The proposed system builds upon the strengths of the original WHyCON system, with the general conceptual design indicated in Fig. 2. Oncoming images are analysed following the original WHyCON approach, searching for circular patterns as detailed in Sec. 4. One the main advantages of the WHyCON system is its ability to start searching for a marker from any position in the image without any performance penalty. Thus, the use of the tracking information to predict the next position of the marker in the image results in a significant performance boost – in case when the prediction is correct, the method processes only those pixels that belong to the marker. Apart from the computational performance boost, the tracking also allows the system to employ Bayesian methods, which enhance the robustness of the system to marker misidentification by taking into account the markers’ identification history. The robustness of marker identification can be further improved by employing self-corrective codes on top of the necklace encoding. The motion prediction, that speeds up the image processing, self-corrective codes that detect misidentified markers, and a Bayesian state estimation that improves the identification robustness, are the main improvements to the system originally described in [16]. With these improvements the system presented outperforms its earlier versions [16, 13] in terms computational efficiency, orientation estimation accuracy and identification reliability.

### 4. DETECTION & LOCALISATION

The WHyCON algorithm was originally intended to localise a large number of concentric black and white circles, of known diameter, in a computationally efficient way. The article [13] shows that the method achieves the same precision as state-of-the-art black and white pattern detectors while being faster by an order of magnitude. To detect the circular pattern, the algorithm searches an image using a combination of flood-fill technique and on-demand thresholding, while gathering statistical information about the patterns on the fly. The statistics gathered allow rapidly identified false candidates to be rejected early in the processing pipeline. One of the key enabling factors in the computational simplicity of the proposed system is that the pattern search can be initiated from any position within the image. By combined this with efficient tracking, the algorithm is typically able to process only the pixels that are occupied by the pat-
terns, leading to significant performance boosts.

For the initial pattern detection phases, the image is searched for a continuous segment of black pixels, which are classified by an adaptive thresholding method that ensures a good level of robustness to adverse lighting conditions. Once a continuous segment of black pixels is found by the flood-fill method, a simple circularity test can be performed.

A pattern consisting of \( s \) pixels, with bounding box dimensions \( b_u, b_o \) and inner and outer diameters \( d_i, d_o \) is considered circular if its ‘roundness’ \( \rho \) is smaller than a pre-defined value \( \rho_{\text{max}} \), i.e.

\[
\rho_{\text{max}} > |\rho_{\text{out}}| = \left| \frac{\pi}{4s} b_u b_o \frac{d_i^2 - d_o^2}{d_o^2} - 1 \right|. \tag{1}
\]

Once the black segment passes the circularity test, a new flood-fill search is initiated to locate the inner white segment. Once the inner segment is found, the algorithm then compares the position of the inner and outer segments’ centre points to verify that the segments are concentric. From here, the algorithm then calculates the ratio of inner and outer segments’ pixels to verify that this ratio conforms to the known ratio of the black and white segments’ areas.

After passing these tests, the positions of the segments’ pixels \( u_i, v_i \) that were stored during the flood-fill search are used to calculate the pattern’s centre \( C \), major and minor axes \( \lambda_0, \lambda_1 \), and the covariance matrix \( \mathbf{C} \) as follows:

\[
\mathbf{C} = \frac{1}{s} \sum_{i=0}^{s-1} \begin{pmatrix} u_i u_i & u_i v_i \\ u_i v_i & v_i v_i \end{pmatrix} - \begin{pmatrix} uu & uv \\ vu & vv \end{pmatrix}. \tag{2}
\]

Note that \( u_i, v_i \) are integers, and the computationally most expensive part of Equation 2 is calculated using integer arithmetic. The \( u_i, v_i \) and \( \mathbf{C} \) actually represent an elliptical projection of the pattern in the image.

Then, the eigenvalues \( \lambda_0, \lambda_1 \) and eigenvectors \( e_0, e_1 \) of the covariance matrix \( \mathbf{C} \) are calculated and used to determine ellipse semiaxes \( e_0, e_1 \) as follows:

\[
e_0 = 2\lambda_0^{1/2} u_0, \quad e_1 = 2\lambda_1^{1/2} v_1. \tag{3}
\]

As the length of the ellipse semiaxes is now known, a final segment test can be performed, which verifies if the number of pixels \( s \) corresponds to the area of the ellipse:

\[
\xi > |\pi e_0 e_1 / s - 1|. \tag{4}
\]

The constant \( \xi \) represents a tolerance value much lower than \( \rho_{\text{max}} \), because the ellipse dimensions \( e_0, e_1 \) are obtained from the covariance matrix with sub-pixel precision. If the detected segments satisfy Equation 3, they are assumed to represent the pattern. The obtained eigenvalues and eigenvectors are then used to calculate the spatial position of the pattern.

To obtain the relative distance of the pattern, the pixel coordinates of the ellipse (co-)vertices are calculated and transformed into canonical camera coordinates using the intrinsic camera parameters that were obtained through standard camera calibration procedure. The transformed coordinates of the (co-)vertices are used to calculate the centre and axes of the ellipse in the canonical camera form. The vertices are used to calculate a conic \( \mathbf{Q} \) such that all the ellipse points \( u', v' \) satisfy

\[
\begin{pmatrix} u' \\ v' \end{pmatrix}^T \mathbf{Q} \begin{pmatrix} u' \\ 1 \end{pmatrix} = 0. \tag{5}
\]

Then, we calculate the eigenvalues \( \lambda_0, \lambda_1, \lambda_2 \) and eigenvectors \( q_0, q_1, q_2 \) of the conic \( \mathbf{Q} \) and use them to obtain the spatial position of the pattern by the method presented in [26]:

\[
x = \frac{d_o}{\sqrt{-\lambda_0 \lambda_2}} \begin{pmatrix} s_1 q_0 \lambda_2 - \lambda_1 \\ \lambda_0 - \lambda_2 + s_2 q_2 \lambda_0 \end{pmatrix} \tag{6}
\]

where \( d_o \) is the circular pattern diameter.

In this work, we also implement a calculation of the patterns orientation. At first, we calculate the normal \( \mathbf{n} \) by

\[
\mathbf{n} = \begin{pmatrix} s_1 q_0 \sqrt{\lambda_0 - \lambda_1} \\ \lambda_0 - \lambda_2 + s_2 q_2 \sqrt{\lambda_1 - \lambda_2} \end{pmatrix}. \tag{7}
\]

Note that the constants \( s_1 \) and \( s_2 \) are undetermined signs that have to be selected so that the \( \mathbf{n} \) points towards the camera and \( x \) is in front of it. In other words, \( s_1 \) and \( s_2 \) are chosen so that the inequalities:

\[
\begin{align*}
\mathbf{n} (0, 0, 1)^T &< 0 \\
\mathbf{x} (0, 0, 1)^T &> 0
\end{align*} \tag{8}
\]

are satisfied. While the roll and pitch of the pattern can be expressed from the normal \( \mathbf{n} \), the yaw of the original circular marker can not be determined. However, the yaw can be calculated in the subsequent step, which uses the Necklace encoding for the pattern identification.

### 4.1 Motion Prediction

As mentioned before, the flood-fill procedure, which constitutes the core of the segmentation, can be initiated from any point in the image being analysed.

If initiated near, or ideally inside of, the searched pattern, the method will process only the pixels of the pattern itself, which significantly reduces image processing time. Thus, the method’s computational efficiency relies on its ability to reuse the patterns past movement information to correctly predict its position in the image currently being processed. In the earlier versions of the system, the search simply started at the pattern’s position in the last analysed image. While this is sufficient for slowly moving targets, (e.g. in swarm robotics experiments [3]), rapidly moving robots require the system to take into account their velocity in order to maximise the tracking performance [19]. Thus, the motion prediction presented in this work uses several past detections of the pattern to estimate its velocity in image coordinates. The last estimated velocity along with the last detected position are then used to predict the position of the pattern in the currently processed image.
4.2 Automatic Parameter Tuning

Apart from tracking, the swiftness of the system depends on its ability to quickly reject false pattern candidates based on on-the-fly-calculated statistics. However, the decision to reject a segment in each statistical test (e.g. (1) or (4)) requires setting a certain tolerance range. This, in turn, rises the need to set these tolerance values depending on a particular application and hardware used. For example, rolling cameras on a quadrotor suffer from a specific ‘jitter’ or ‘trembling’ noise, caused by vibrations induced by the drone’s motors [1], which makes the segments appear deformed and non-circular. This requires $\xi$ value in Eq. 4 to be relaxed in order to work. Another tolerance value that is affected by hardware used is the expected ratio of black and white pixels in the segment – this is subject to non-linear sensitivity and chromatic aberration of the camera which also depend on the current lighting conditions. To cope with the dependence of these tolerances on the hardware used and current illumination conditions, we employed a simple scheme, which slowly adapts these tolerances to the values that the system experiences during its operation. In particular, if the detection of a given pattern is successful, the expected black-to-white pixel ratio $b_{exp}$ is updated as follows: $b_{exp} \rightarrow 0.99b_{exp} + 0.01b_{real}$, where $b_{real}$ is the value calculated by the segmentation method. The other parameters, $\rho$, $\xi$ etc. are updated analogously.

5. MARKER IDENTIFICATION

Building upon the good detection performance of the WhyCon system, and adhering to the requirement outlined in the introduction, our development of a new marker system focused on creating a marker which is compatible with the circular features of WhyCon, but also capable of providing a scalable encoding system to uniquely identify each marker. The proposed encoding chosen for the WhyCode marker was originally identified within the combinatorics field of mathematics, and currently used widely in the fields of combinatorial chemistry [4] and computational biology [5]. These sequence patterns known as Necklaces are “lexicographically the smallest element in an equivalence class of strings under string rotation” [20].

Despite it currently not being used in the field of robotics, this encoding was a highly suitable option for the proposed system due to its rotational invariant nature. By bit-rotating the detected sequence until its lowest binary value is reached, the system is able to identify a starting point regardless of the position from which the code was originally read from. This technique of altering the detected code without confusing the IDs is the core concept being identifying the markers’ yaw rotation. Taking the number of rotations required to reach the lowest binary value, we are able to identify how far rotated the marker is, from a ‘zero point’ of rotation, circumventing the issue of identifying a starting point on a circular marker. This rotation can then be transformed into 3D space to calculate the marker’s yaw rotation, making the markers position detectable in a full 6th DoF. As the ID is encoded by bit-rotating each number to their lowest binary value, both the ID calculation and subsequent yaw rotation can both be pre-calculated and stored to minimise computational costs, thus improve the performance of the system. However, for this to work reliably all codes which have rotational symmetry, must also be removed from the encoding system, as they allow for the lowest binary value to be reached from multiple start locations, which would result in ambiguity when establishing the markers’ yaw. To see an example of a marker with ambiguous yaw, see the leftmost quad-copter on Figure 1(e).

![Figure 3: An example of how the Manchester Encoding is used with the Necklace System: The inner circle of the WhyCode marker encodes a binary string which is bit-rotated to match a Necklace code. Apart from identification, the number of bit-rotations allows us to identify the marker’s rotation.](image)

To create a system which reliably identifies the markers and preserves backward compatibility with the WhyCon marker, we encoded the Necklace-based ID into the inner circle of the tags and used Manchester Encoding [9]. Thus, each individual bit of the Necklace code is encoded by two consecutive segments of opposite colour, as demonstrated in Figure 3. Although the use of Manchester Encoding halves the number of segments available on the marker, it allows us to calculate an identification confidence rating based on the expected number of pixels in each segment of the Necklace code.

In theory, the Necklace Encoding supports higher than binary bases, and it would be possible to encode the marker IDs in grey-scale values along the inner rim of the circle. However, preliminary tests showed that the edge-based Manchester Encoding is more suitable due to its robustness. This has the benefit of making the system more robust, especially when subject to various lighting condition, but does have the negative effect of only allowing binary-code sequences when encoding IDs. As a result, this restricts the encoding system and limits the number of potential IDs to:

$$N = \frac{1}{n} \sum_{d=1}^{n} \varphi(d)2^{n/d}, \quad (9)$$

where $\varphi()$ is totient function [15] and $n$ is the Necklace code length in bits. The Equation 9 is further illustrated in Table 1 which shows the number of combinations valid for the proposed marker, given that the Necklace code consists of a sequence of $n$ bits:

5.1 Identification Uncertainty Handling

Decoding the ID actually requires that one establishes correctly the falling and rising edges of the Manchester code that is encoded in the inner circle of the pattern, see 3.
errors, thus giving the system the ability to identify (\(f\) is particularly visible, state than by the current observation. However, if a marker
id low and the \(\arctan f\) equals top of the necklace code. The first logical step was to in-
lation is computationally inexpensive and as such does not
measurement is taken into account. Note that this calcu-
provide consistent ID information even if the individual ID
decoding is noisy. In order to handle noisy ID readings,
we maintain a probabilistic distribution of all possible IDs
over all visible markers. This distribution is updated using
Bayes whenever a detected marker ID is decoded. In other
words, each detected marker is associated with \(n\) numbers,
representing the probabilities \(p(id = 1 \ldots n)\) that the marker
has an ID \(1 \ldots n\). Whenever the marker’s ID is decoded, each
of these numbers is updated by a Bayesian rule

\[
p(id|o) = \frac{p(o|id)}{p(o)} p(id),
\]

where \(p(id)\) represents the prior probability of the marker
having a given \(id\) and \(p(o|id)\) corresponds to the probability
of decoding the pattern incorrectly. Since \(p(o|id)\) is deter-
mined by the quality of the marker’s decoding ring image, we
quantify the quality of the decoding ring as the sum of gra-
dients along it – this corresponds to the quality of the edges
that constitute the Manchester encoding. Thus, we assume
that the probability of the correct decoding of \(p(o == id)\)
equals \(f(s)\), where \(s\) is the sum of the gradients along the
image of the decoding ring and \(f(.)\) is a sigmoid-shaped func-
tion (modelled by an \(\arctan\)) in our case. Our experiments
have shown that \(f(s)\) captures many of the factors which af-
fact the marker’s identification correctness, such as marker
size, detection distance, motion blur, extreme viewing an-
gles and low-lighting conditions. Thus, in cases, where the
edges of the decoding ring are weak, the confidence \(f(s)\) is
low and the \(id\) of the marker is affected more by its previous
state than by the current observation. However, if a marker
is particularly visible, \(f(s)\) is high and the new, high quality
measurement is taken into account. Note that this calcu-
lation is computationally inexpensive and as such does not
affect the computational complexity of the system.

### 5.2 Hamming Code

Another possibility to improve the quality of the pattern
identification is the adoption of self-corrective encoding on
top of the necklace code. The first logical step was to in-
crease the number of code bits \(n\) and ensure that a minimum
Hamming distance \(w\) is kept between any two codes used.
In theory, this would allow for the correction of \((w-1)/2\) bit
errors, thus giving the system the ability to identify \((w-1)\)
errors in bit order. This would consequently increase the
robustness of the system because if a bit or two flips, it is
regarded as invalid and is rounded to the closest ID. Because
the proposed coding system is rotation invariant, however,
it also needs to take into account the Hamming distance be-
tween every cyclic permutation of the necklace code. This
can severely limit the number of possible IDs, see Table 2.
Moreover, more bits means smaller elements of the decoding
ring, which increases the chance of errors when decoding the
ID from the image. Thus, there is a trade-off – higher Ham-
mapping distance improves a change of detecting or correcting
an error, but it also increases a chance of introducing one.

To determine which Hamming distance to use for pattern
identification, we recorded 4 sequences, where UAVs carried
tags with Hamming distances of 1, 2, 3 and 4. These experi-
ments indicated that while using Hamming distances of 3
and 4 did not bring additional benefit in terms of identifica-
tion robustness, using Hamming distance 2 allowed for the
detection of false identifications. Taking into account the
false identifications in the confidence function \(f(s)\) from the
previous section, allows the result of the error detection to
be taken into account in the Bayesian update scheme. Thus,
the Hamming code’s ability to detect errors along with the
Bayesian update scheme of the IDs probability resulted in
less than 5% identification error on the UAV dataset de-
scribed in Section 7. Furthermore, the Hamming code’s abil-
ity could be used to learn the \(f(s)\) from the actual data on-
the-fly, improving the robustness of the identification even
further.

### 6. EXPERIMENTS

To evaluate the performance of the proposed marker, we
compared its localisation accuracy, detection range and iden-
tification reliability to state-of-the-art fiducial markers in a
series of real experiments. Each of these tests used an RGB
camera of an ASUS Xtion RGB-D sensor, as it corresponds
with the type of sensor that is widely used on robotic plat-
forms, providing a standard 640×480 image at 25 frames
per second. This sensor was fixed to a FLIR E46-17.5 Pan
Tilt Unit (PTU) which provided a ground truth for the
marker position, orientation and velocity. This PTU was
also mounted atop a mobile platform with a SICK s300 laser
scanner. As the detectable range of the markers exceeds the
range of a ASUS depth camera, the laser scanner with a
range of up to 30m provided a reliable distance measure-
ment that was also used for the ground truth in some of the
experiments. To allow for a fair comparison of the proposed
marker against the ARTags and AprilTag, each of these
markers were resized to occupy the same area of 3.817cm².

<table>
<thead>
<tr>
<th>Code length [bit]</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique IDs [-]</td>
<td>3</td>
<td>9</td>
<td>39</td>
<td>99</td>
<td>335</td>
<td>979</td>
<td>2623</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hamming distance</th>
<th>Code length [bit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>3 9 30 99 335 979 2623</td>
</tr>
<tr>
<td>2 bit</td>
<td>2 5 16 51 170 580 1759</td>
</tr>
<tr>
<td>3 bit</td>
<td>1 2 3 7 16 48 125</td>
</tr>
<tr>
<td>4 bit</td>
<td>1 1 2 5 10 24 69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hamming distance</th>
<th>Code length [bit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>3 9 30 99 335 979 2623</td>
</tr>
<tr>
<td>2 bit</td>
<td>2 5 16 51 170 580 1759</td>
</tr>
<tr>
<td>3 bit</td>
<td>1 2 3 7 16 48 125</td>
</tr>
<tr>
<td>4 bit</td>
<td>1 1 2 5 10 24 69</td>
</tr>
</tbody>
</table>
A default calibration was also used, rather than specifically calibrating the camera, to demonstrate the system’s performance in standard circumstances.

### 6.1 Detection and Identification Range

![Figure 4: Illustration of range tests](image)

The first test aimed to evaluate the effect that distance had on the performance of the system. The markers were affixed to the wall at a height equal to that of the camera. The mobile platform was then programmed to move backwards from a distance of 0.2 metres until the platform reached a distance of 7 metres from the wall. The movement occurred at a constant speed of 0.02 metres per second, which was selected in order to ensure that motion blur was not a significant factor.

**Table 3: Maximum distances at which the markers were consistently detected and identified [m]**

<table>
<thead>
<tr>
<th></th>
<th>WHYCON</th>
<th>APRILTAG</th>
<th>WHYCODE</th>
<th>ARTAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection</td>
<td>5.4</td>
<td>2.1</td>
<td>4.9</td>
<td>3.4</td>
</tr>
<tr>
<td>Identification</td>
<td>2.1</td>
<td>2.4</td>
<td>2.4</td>
<td>2.7</td>
</tr>
</tbody>
</table>

![Figure 5: Maximum distances at which the markers were consistently detected and identified](image)

As can be seen in Table 3 and Figure 5 the original WHYCON marker has proven to achieve the longest detection range of 5.4 metres. Although the WHYCODE marker was almost able to achieve a similar range, the new marker started to provide incorrect IDs once the distance had surpassed 2.4 metres. Similarly to that, the ARTAGS were undetectable at a range of 3.5 metres or more, and their correct identification was not reliable when the distance of the marker exceeded 2.7 metres. As for the APRILTAG, no incorrect IDs were reported. However, the distance at which the marker was reliably detectable was the lowest of the markers tested at only 2.1 metres.

### 6.2 Identification Range vs. Code Length

![Figure 6: Dependence of maximal identification range on the Necklace code length n. The estimate is based on a formula min(2.4, 200/n)](image)

A similar test was also conducted on the WHYCODE marker to identify how changing the number of encoding bits affects the range at which the encoding can be correctly identified. As can be seen in Figure 6 using less than 8 bits for the code does not affect the range, while increasing it has a negative impact on the identification range. This corresponds with the expectation that the limiting factor of identification range is the size of the individual elements that make up the encoding pattern.

### 6.3 Robustness to Motion Blur

![Figure 7: Illustration of motion blur tests](image)

This test, which was intended to analyse the effect of motion blur on the markers, involved keeping the markers stationary whilst rotating the PTU. This setup not only ensured the equal movement of all the markers, but also created a stable, continuous and repeatable experiment which represented one of the system’s intended applications: mobile robotic platforms with a moving on-board camera. With the markers affixed to the wall, the camera was placed exactly 1 metre from the wall and the PTU rotated from -90 degrees to +90 degrees at a constant speed. Figure 8 shows the speeds that were tested during this experiment with the resulting detection and identification ratios.

These results indicate that while both WHYCODE and WHYCON systems are less susceptible to motion blur, the APRILTAG identification scheme is more robust to motion blur compared to WHYCODE.

When attempting to decode the ID, the WHYCODE marker reported a number of incorrect results at the faster motions, which is caused by the fact that during these tests, the code did not employ any error detection or self-correction scheme. In contrast, the lexicographic error correcting [23] used by the APRILTAG meant that no incorrect IDs were ever detected during our tests.
6.4 Accuracy of Angle Estimation

Since the $x, y, z$ position estimation is identical to the original WhyCon method [13], which reports that its localisation accuracy is comparable to ARTAGS based markers, we tested only the accuracy of angle estimation. In contrast to the earlier experiments, the markers were this time placed on the robot’s PTU which, whilst facing the free-standing stationary camera, used the pan and tilt functions to vary the angle of the markers. The recorded positions and rotations of the markers were then compared to the angle taken from the PTU. This comparison was then used to calculate an error rate for the system, see Table 4.

Table 4: Average error of angle estimates [radians]

<table>
<thead>
<tr>
<th></th>
<th>WhyCon</th>
<th>APRILTag</th>
<th>WhyCode</th>
<th>ARTAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitch/roll</td>
<td>0.024</td>
<td>0.023</td>
<td>0.020</td>
<td>0.038</td>
</tr>
<tr>
<td>Yaw</td>
<td>0.034</td>
<td>0.042</td>
<td>0.044</td>
<td></td>
</tr>
</tbody>
</table>

As can be seen from the above table, all markers exhibited average errors lower than 0.05 radians demonstrating that the system’s ability to establish the marker’s orientation was successful across all four systems. It should be noted that while the original WhyCon marker is unable to provide the yaw rotation, WhyCode can estimate the yaw rotation with a high level of accuracy using the Necklace Encoding.

6.5 Robustness to Illumination Changes

The last test aimed to verify the performance of the system when subjected to various lighting conditions. To achieve this, the markers were positioned next to a large window in order to utilise natural, ambient light and avoid the flickering sometimes caused by artificial light. By taking a photo every 10 seconds during the 25 minutes before and during sunrise, the markers were able to go from complete darkness to normal daytime lighting conditions. While the ARTAGS were detected in 64% of these images, APRILTag, WhyCon and WhyCode were detected in 71%, 72%, 74% of images respectively. Since the slight differences in performance may be attributable to slight variations in light, we can state that all the markers demonstrated a similar robustness to variable illumination.

6.6 Computational Complexity

In addition to the above tests, a number of computational performance tests were conducted on each of the systems. The first of these were conducted using procedurally generated images of size $5000 \times 5000$ pixels containing over 550 randomly placed markers. This test helped to evaluate each of the systems ability to handle, not only large images, but also images which contain high number of markers and varying levels of clutter. Although WhyCon and WhyCode took more than a second to process the first frame, each subsequent frame was then processed significantly faster. The average time to process a single frame when comparing the APRILTag and the WhyCode systems can be seen in Table 5, which shows the main advantage of the WhyCode method – its computational efficiency. Table 5 also shows that the identification and yaw estimation step do not slow down the original WhyCon method, which is two orders of magnitude faster than the ARTAGS and APRILTag.

The performance boost WhyCon and WhyCode results from the on-the-fly calculation of the detected segment statistics, which is naturally achieved by the flood-fill segmentation technique and which allows tracking without any computational overhead. Although the computational efficiency of both ARTAGS and APRILTag could be improved by employing some tracking scheme, it is unlikely to achieve a two-orders of magnitude speed-up.

Table 5: Average processing time of an image with 550 markers [seconds]

<table>
<thead>
<tr>
<th>Clutter</th>
<th>WhyCon</th>
<th>APRILTag</th>
<th>WhyCode</th>
<th>ARTAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0.06</td>
<td>3</td>
<td>0.06</td>
<td>3</td>
</tr>
<tr>
<td>little</td>
<td>0.07</td>
<td>16</td>
<td>0.07</td>
<td>14</td>
</tr>
<tr>
<td>large</td>
<td>0.07</td>
<td>15</td>
<td>0.07</td>
<td>15</td>
</tr>
</tbody>
</table>
6.7 Motion Prediction Speed Up

The computational performance of the system is affected by its ability to predict the approximate positions of the markers in the currently processed image. The earlier versions of the system simply searched for the pattern at the position it was detected in the last processed image, which was sufficient for experiments, where the markers moved only slowly. However, if the system is deployed in situations, where the tracked objects move rapidly, e.g. when the system is used to track aerial robots [19], the aforementioned method failed to predict the positions properly, which resulted in system slowdown. Thus, we improved the motion prediction by taking into account the marker velocity as described in Section 4.1.

Table 6: Average processing time of sequences with rapidly-moving markers [ms]

<table>
<thead>
<tr>
<th>Tracking type</th>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
</tr>
<tr>
<td>Position only</td>
<td>21.5</td>
</tr>
<tr>
<td>Position + velocity</td>
<td>19.7</td>
</tr>
</tbody>
</table>

To evaluate the effect of the improved tracking scheme, we calculated the average time it takes to process an image of the UAV dataset described in Section 7, which contains four sequences of flying robots captured from an UAV-mounted camera. The Table 6 shows that taking into account the tracked pattern velocity reduces the computational load by ∼10-20%.

7. DATASETS

Another contribution of this work is the addition of two publicly available datasets that ensure the repeatability and verification of the results obtained above. The first of which provides numerous videos, along with position and movement statistics taken from a SCITOS robot, which can be used to test the abilities of the WhyCODE, WhyCON, APRILTAG and ARTAGS systems that are mentioned above. In addition to the benchmarking dataset, a real world example of the new system being utilised to track drones flying outside, along with their reported positions taken from the drones IMU and GPS devices. Both of these datasets are available for download from: http://lncn.eu/fidusets.

8. CONCLUSION

In this paper, we present an extension to the marker used by the WhyCON tracking system. The proposed method not only utilises a new encoding algorithm which allows identification of each marker, but also extends the system to allow the full localisation of a marker with 6 DOF. Furthermore, we introduce several improvements that strengthen the robustness, accuracy and computational efficiency of the detection and identification. By keeping the simple roundel design, the proposed marker is not only backwards compatible with the previous system, but also maintains its sub-pixel (2D) and millimetre (3D) precision, and high computational efficiency.

The results of our study show that the WhyCon system, despite the additional overhead of having to decode marker IDs, performed similarly to the original WhyCon system and outperformed the comparative systems in both accuracy and speed. By exceeding the high level of performance demonstrated by the APRILTAG and ARTAGS, and at two orders of magnitude faster, the proposed system achieves a strong level of accuracy without the high computational requirements. These achievements therefore make the proposed system particularly applicable to resource-constrained systems and scenarios, where the reliable and swift tracking of multiple robots is a necessity. Moreover, the WhyCon system can reliably detect smaller markers at longer ranges, which is also makes it a popular alternative to APRILTAG or ARTAGS.

In the future, we will explicitly model uncertainty of the marker locations, which should not only improve our system’s accuracy [17], but also its coverage by allowing to fuse input from multiple cameras.

Acknowledgements

The work is partially supported by the European Community’s funded projects 600623 ‘STRANDS’ and 732773 ‘IL-IAD’, the Research Investment Fund (RIF) of the University of Lincoln, and the Czech Science Foundation project No. 17-27006Y.

9. REFERENCES


ABOUT THE AUTHORS:

Peter Lightbody is a PhD student at the University of Lincoln and a member of Lincoln Centre for Autonomous Systems Research group in the UK. His main research include are human-robot collaboration, machine learning, adaptive planning, task scheduling and social interaction.

Tomáš Krajník is a research fellow at the Lincoln Centre of Autonomous Systems, UK. He received the Ph.D. degree in Artificial Intelligence and Biocybernetics from the Czech Technical University, Prague, Czech Republic, in 2012. His research interests include life-long autonomous navigation, spatio-temporal modelling, and aerial robots.

Marc Hanheide is a Reader (Associate Professor) in the School of Computer Science at the University of Lincoln, UK. He received his Diploma and Ph.D. degree (Dr.-Ing.) in Computer Science from Bielefeld University, Bielefeld, Germany, in 2001 and 2006, respectively. Currently he is a PI in the EU H2020 ILIAD, and in the InnovateUK “Fruit Transport” project, investigating aspects of long-term robotic behaviour and artificial intelligence in general, and human-robot interaction in particular.
Computer Aided Diagnosis with Boosted Learning for Anomaly Detection in Microwave Tomography

Chulwoo Pack  
University of Nebraska - Lincoln  
Lincoln, USA  
cpack2@unl.edu

Seong-Ho Son  
Electronics and Telecommunications Research Institute (ETRI)  
Daejeon, South Korea  
shs@etri.re.kr

Sung Shin  
South Dakota State University  
Brookings, USA  
sung.shin@sdstate.edu

ABSTRACT
Most recently developed Computer Aided Diagnosis (CAD) systems and their related research is based on medical images that are usually obtained through conventional imaging techniques such as Magnetic Resonance Imaging (MRI), x-ray mammography, and ultrasound. With the development of a new imaging technology called Microwave Tomography Imaging (MTI), it has become inevitable to develop a CAD system that can show improved performance using the new format of data. In this paper, we propose the Multilayer Perceptron (MLP) model taking MT data as an input and we boost the learning process by using Dynamic Learning Rate (DLR) and momentum. In order to eliminate the indeterminate equation problem, we optimize the number of weights to the number of the training data. To make the model capable of escaping the local minima problem, we assign the learning rate dynamically. Since training a network with data having uneven distribution for the all possible classes can cause the network to be biased to the majority class, our model assign the low learning rate if unseen data belongs to the majority class. Along with this strategy, to speed up a back-propagation, the model employs the momentum optimizer to reduce the convergence time. In experiment, we train the model with two different datasets, 15 and 30, and evaluate the performance by the following measures; precision, recall, specificity, accuracy, and Matthews Correlation Coefficient (MCC). Differences in each measure are assessed by paired t-tests at the significance level of 0.01 for validation purposes. The results show that the proposed model outperforms the conventional model for the overall measures but only precision, accuracy, and MCC are statistically significant.

CCS Concepts

• Software and its engineering → Software performance; • Applied computing → Health informatics;

Keywords
Computer Aided Diagnosis; Multilayer Perceptron; Dynamic Learning Rate; Anomaly Detection; Microwave Tomography

1. INTRODUCTION
Breast cancer is currently the most dominant cancer for women in both the developed and the developing countries. Improving the outcome and survival from breast cancer still remains as the cornerstone of breast cancer control in the modern era [1]. Detecting a tumor at an early stage plays an important role for enhancing survival rate (or reducing mortality). A CAD system has been helping the physician for detection of breast cancer at the early stage on mammograms in the United States [6]. Also, a CAD system is investigated in abounding research in order to reveal breast microcalcification and to support the practitioner [2][13]. The concept of a CAD system is not that an image processing task by a computer for a digitalized medical image replaces that of the physician or radiologist, but it could be utilized by them [6]. This should not be misunderstood as the same concept as automated computer diagnosis. Automated computer diagnosis considers a computer as a subject of final decision-making, while medical practitioners, in CAD, are using a computer’s output as a second opinion and they make the final decision. Certainly, higher performance level of a computer is possible to lead better quality on the final diagnosis, however it does not have to be equal to or higher than that of physicians [6]. Focusing on the synergistic effect obtained by combining ability of physicians and computers is the most important factor in a CAD, and the current CAD has become a practical tool in many clinics with attractive benefits [6].

A new technology for breast cancer screening, MTI, was recently introduced as an alternative scheme to diagnose breast cancer. To the best of our knowledge, instead of the mammogram, only few research studies are ongoing using the MTI as an input for a CAD system [8][19][21][5]. The MTI has many advantages over the mammogram such as cost-efficient and requiring less processing-time. To be more precise, the MTI surpasses over standard techniques, namely MRI, x-ray mammography, and ultrasound in aspects of low health risk, non-invasive, inexpensive, and minimal discomfort [16][13]. The MTI gauges dielectric properties of tissues, which are permittivity and conductivity, inside of the breast. Two parameters indicate the degree of propagation of electromagnetic radiation which will differ based on the temperature, density, water content, and geographical conditions within the breast mass.
Unlike other classifiers such as a Support Vector Machine (SVM), an Artificial Neural Network (ANN) has a few points to be cautious. Since the number of parameters depends on the combination of hyper-parameters in accordance with the network structure, it can vary a lot. If the number of parameters get either greater or less than the number of samples, then the network is hard to avoid either overfitting or underfitting problem, respectively. Mathematically, it is same as trying to solve an indeterminate equation. Moreover, while a SVM guarantees a global optimum value, an ANN has a possibility of converging to a local minimum value, which is known as local minima problem.

Our work focuses on avoiding the above-mentioned circumstances to obtain reliable classification results as Matthews Correlation Coefficient (MCC), Recall, Specificity, Accuracy, and Precision. In this paper, we propose a neural network model, which is capable of being plugged into a CAD software platform, taking the MT data as an input and we boost a learning process by using DLR and momentum. Figure 1 illustrates our model.

The remainder of this research work is organized as follows: Section 2 is a literature review that contains background of CAD system and MLP. Related works are mentioned in Section 3. Section 4 describes the proposed model. Section 5 shows experimental results and analysis. Finally, Section 6 concludes this paper and gives directions for future works.

2. LITERATURE REVIEW

2.1 Computer Aided Diagnosis System

A CAD system provides attractive benefits for an image screening interpretation. Even for the expert radiologists, interpreting screening mammogram is not an easy task because of its work load caused by the number of images and its quality [14]. To overcome this problem, using double rendering without increasing a recall rate requires much more of manpower, and accordingly it leads to increase the cost. However, when it comes to the CAD system, it a cost-effective alternative to the double rendering by acting as a second reader [14]. The way of utilizing the CAD system is that a radiologist, at the first rendering, interprets the mammogram, and then compares it to the results from the CAD at the second rendering in order to check if he or she missed or unchecked at the first rendering [14]. As described, the computer rather being a subject of making the final decision, it provides additional information to the physician in a cost-effective way, requiring less manpower and time, with even higher quality of diagnosis [14].

Since currently the CAD system is widely being used in many clinics with the mentioned advantages, developing new algorithms for the CAD system is an active research field in academia [14]. Figure 2 shows one of the example how to utilize the CAD system for early detection of breast cancer.

Figure 2: Diagram of CAD expert system for early detection of breast cancer
pending on the kinds of classifier, for instance typical Convolutional Neural Network (CNN) does not require human being to pick a particular feature set. If that’s not the case, designing feature extraction & selection phase is essential for the performance of the CAD system. Since the combination of extracted feature sets can be numerous, evaluating them can be time-consuming task and it will directly affect the quality of the classification. Finally, a classifier makes decision with the given feature sets in the classification phase. Practically, since a developer has lots of options to choose for the classifier such as an Naïve Bayes Classifier, K-Means Clustering, SVM, or ANN, it is important to choose the most feasible one based on a given problem and an expected output.

2.2 Multilayer Perceptron

A MLP is one type of an ANN such as a Single Layer Perceptron (SLP) or Self-organizing Map (SOM); one that is recently getting lots of attention as a fundamental deep neural network architecture. As its name states by itself, the MLP comprises at least three layers: one input layer, one or more hidden layer, and one output layer. Each layer has many perceptron linked with another in the next layer. All the connections get assigned a value called weight. Eventually, a well-trained network should have an optimal weights set. What the hidden layers does is basically similar to what a kernel does in the SVM that of projecting feature vectors to high dimensional space and finding a hyper-plane separating the training points properly. In the MLP, a similar effect of projecting feature vectors is achieved through the multiple layers of the perceptron. Each layer constructs a feature vector for the given data based on the assigned weight. The features from the previous layer is passed to the next layer through an non-linear activation function. This process results in changing the feature space where the data can be projected and thus the next layer can map the passed data to the new feature space.

A single perceptron is a mathematical model of a biological neuron. The perceptron transmits the given features in a similar way to the propagation of electrochemical stimulation in a neuron. Each perceptron takes weighted sum of the information from its nearby perceptron like a dendrite and pass it to an activation function. Only the filtered out information based on an activation rule is passed to the next perceptron which is called feed-forward computation. When it comes to an activation function, an non-linear function is adopted to determine which information is significant enough to be passed. Typically, a binary or bipolar sigmoid functions are the options for the activation function. Recently, various research studies on a CNN have proposed many other choices, such as a Rectified Linear Unit (ReLU), which can tackle the gradient vanishing problem.

The ultimate objective of a training process in the MLP is to find a weights set that of minimizing the result of a cost function. In a supervised learning, a cost function measures a difference between the actual output and the expected output. For every iteration, all the weights in the network are updated in a way of reducing the output of the cost function. A mechanism of deciding up to what degree the weights should be updated is called a learning algorithm and the gradient descent is a dominant technique. Due to the nature of a cost function, the next weights should be made equal to a learning rate in an opposite direction that of a derivative of a cost function at the current weights. At this point, the classification results considerably varies depending on the learning rate. A cost function with the use of a smaller learning rate would require more extra time and computation to reach a minimum value. On the other hand, if the learning rate is too large, the cost function would decrease fast on every epoch but it might bounce back from the minimum and fail to converge. The worst part of an ANN is that even though we happened to use a desirable learning rate, the network has a possibility to reach a local minimum rather it reach the global minimum, which is called the local minima problem.

3. RELATED WORK

3.1 Feature Extraction and Selection

The strategy of feature extraction & selection phase is important because the prediction quality of a classifier highly depends on the quality of feature sets. According to the work by Samaneh et al., the permittivity value, one of the measures obtained through the MTI, depends on the water contents that tissues contain. Typically, the cancerous tissues have relatively high water contents, while the healthy tissues have lower water contents. Therefore, in order to detect a suspicious region that might have the cancerous tissues, they divided the data samples into two groups based on a distribution of the permittivity value by using the K-Nearest Neighbor. First, after dividing the dataset into cancerous and healthy groups by Euclidean Distance, then some numerous features for each groups were measured as follows:

1. Mean value: This value calculates the permittivity value in a different lesion. The lesion that contains the cancerous tissue tends to have a higher value than that of the normal tissue.

2. Maximum and minimum value of permittivity in probable tumor area: These values indicate the domain of a permittivity value of the cancerous tissues.

3. Entropy: An entropy measures an amount of disorder in the permittivity data and this is calculated as the equation (1).

\[
Entropy = -\frac{1}{n} \sum_{i=1}^{n} per_i \cdot \log(per_i)
\]  

(1)

4. Energy: An energy represents the orderliness of the permittivity data. An energy is generally given by the mean squared value of a signal and is calculated as equation (2).

\[
Energy = \frac{1}{n} \sum_{i=1}^{n} (per_i)^2
\]  

(2)

3.2 Classification
The work of [2] adopted a MLP, one of the various ANN type, as a classifier. They used a network with a total of 4 layers that each of hidden layers contains 30 neurons. The output layer is implemented as a binary classifier for benign and malignant group. The training process is a supervised-learning using the Levenberg-Marquardt back propagation to minimize the cost-function. Here, the cost-function uses the sums-of-square error function as below:

\[ E = \frac{1}{2} \| e \|^2 \] (3)

Where \( e \) is the difference between the actual output and the expected value. The termination condition for the training is set to be a default as set in \textit{trainlm}, a function given in MATLAB.

4. PROPOSED MODEL

Ideally, a MLP does not require a human being to choose particular features since when a massive dataset is given, the model should be able to extract correlated features by itself during the training process. We have few datasets, however, specifying some features as an input for the MLP is compulsory. Along that fact, after separating the patient’s dataset into 2 groups of cancerous and healthy tissue by using K-Nearest Neighbor algorithm, we extract set of commonly used intensity and texture based features for each group as following [15]:

1. Mean value
2. Maximum and minimum value of both permittivity and conductivity in both healthy and tumorous tissue
3. Entropy
4. Energy
5. Skewness: The skewness measures asymmetry of the probability distribution of a real-valued random variable about its mean.

\[ Skewness = \frac{E(x - \mu)^3}{\sigma^3} \] (4)


\[ Kurtosis = \frac{E(x - \mu)^4}{\sigma^4} \] (5)

7. Ratio of Healthy/Tumorous Tissue within breast mass

As a result, we end up with 52 different features total, which is too large number as an input vector. A more detailed explanation is described in the Section 4.1, but we do not use all the features as an input vector for the network because the number of input is related with the size of the network and the size of network does matter to its performance. We thus select only 3 features listed below ranked on the top among the 52 features based on the correlation-based score, which is shown in equation [6] [7] [12].

1. Ratio of health tissue within breast mass
2. Ratio of tumor tissue within breast mass
3. Maximum value of conductivity in healthy tissue

\[ Score = \frac{\sum_{i=1}^{k} corr(c_i, f_i)}{\sqrt{k + 2 + \sum_{i=1}^{k-1} \sum_{j=1}^{k-i} corr(f_i, f_j)}} \] (6)

4.1 Classification Phase

The selected features is used as the input to the MLP that consists of 1 input layer, 2 hidden layers (each layer contains 3 and 2 perceptron respectively), and 1 output layer. How we came up with this setting is based on the following fact. First of all, it has been proved that using a three-layer perceptron can construct the complex decision regions faster with a back-propagation than two-layer perceptron [3]. Second, the network size is determined by following equation.

\[(d + 1) * p + (p + 1) * m \] (7)

where \( d \) is the number of features that goes into the input layer, \( p \) is the number of perceptron in hidden layer, and \( m \) is the number of perceptron in output layer. When the size of the network is larger than the number of samples, then it can cause a low performance and lack of reliability for the classification. For instance, training network that consists of 100 inputs, 30 perceptron in hidden layer, and 10 perceptron in an output layer with 100 samples is the same as trying to solve an indeterminate equation that has 3340 variables, which is the number of weights in the network, with 100 value. To avoid the referred problem, the proposed model has been optimized with the setting as mentioned above.

Our model works as follows. Once the optimized sized network takes over the extracted features, it feeds a signal forward and it finally yields an output through the Softmax classifier, which has a different loss function, the cross-entropy error. Suppose that we have outputs \( f_j(x_i, \theta) \) as a score for the \( j^{th} \) class of \( i^{th} \) patient, each associated with a label \( y_i \). Then a discrepancy is produced by the following formula, where \( p(x) \) indicates a true distribution:

\[ q = \frac{e^{f_{x_i}}}{\sum_{j} e^{f_{j}}} \]

\[ J(\theta; x; y_i) = - \sum_{x} p(x) \log q(x) \] (8)

Next, the weights between perceptron will be updated through the training process, which is also known as an error-correction process. Basically, the goal of the training process is to find the best set of weights, \( \theta \), that minimizes the loss. This is achieved by applying the delta rule for a neural network [10]. The next weights set is determined by adding particular amount of value, say \( \Delta v \), to the current weights set. The value \( \Delta v \) is determined by the equation (9). As a result, the training process described above can be put succinctly as the below equation.

\[ \Delta v = \mu v - \rho \frac{\partial}{\partial \theta} J(\theta; x; y_i) \]

\[ \theta = \theta + \Delta v \] (9)
where \( \mu \in [0, 1] \) is a momentum coefficient and \( \rho > 0 \) is a learning rate.

Unlike a typical SGD, the momentum term in the equation \( \mu \nu \), accelerates a gradient descent in a way that it accumulates a velocity vector in directions of persistent reduction in the objective across iterations and dampens oscillations across the slope of the ravine \( [18] \). The value of \( \nu \) is initialized at 0 and the \( \mu \) is empirically set to the values such as 0.5, 0.9, or 0.99.

In the equation \( \mu \nu \), all the factors, except the value of \( \rho \), are already determined. The value of \( \partial J/\partial \theta \) indicates the gradient of the graph in the Figure [3] at the point of the current weights set. For example, in Figure [3] suppose that the network is initialized to have a value of weights set located just to the right of the point \( A \). Then the gradient is determined to be a negative value. The sign of the gradient value tells what direction the next weights update should take. Thus, the remaining part for us to decide is determining to what magnitude the next weights update should take. This is where the learning rate comes into play. The too small learning rate makes a system slow that is obviously not good for a performance and the too large learning rate causes a neural network to be dispersed from the desired minimum. Figure[3] illustrates that a network with an improper learning rate can be a problematic.

The path from the point \( A \) to \( C \) shows that if the network has a small learning rate, the global optimum can be reached, but the processing time can be too long. On the other hand, as the path from the point \( A' \) to \( B \) shows, when the network has a large learning rate, it fails to converge to the global minimum and falls into the local minimum. So, training network with a desirable learning rate is one of the most important factor. It has been proved that using a fixed learning rate will cause a low efficiency [20]. Our network is more fragile on this concern because of lack of data and unbalance of the class. To overcome this issue, we designed a logic to assign the learning rate dynamically to each training points. The main idea is simple as shown in the algorithm [11]. During the training process, the network keeps track the class of seen data so that it knows which class is dominant or the other way around. The network inspect if a given data to be trained belongs to a dominant class. If so, a relatively large, otherwise a small learning rate is dynamically assigned. By doing so, the network can assign more flexible weights between perceptron in a way that having the network not to be overfitted with the training data belonging to the majority class by compensating the training data belonging to the minority class with a high learning rate. As a result, the network that are more sensitive to the minority class become more capable of escaping the possible local minima.

Ordinarily, there are many different strategies for updating weights between perceptron in the neural network such as a SGD, Batch gradient descent, and Mini-batch gradient descent. The batch and Mini-batch gradient descent updates \( \theta \) all at once with its average value after training either all the training samples or the specific number of samples, respectively. On the other hand, the SGD, which is used in our work, updates \( \theta \) for each training sample. It has been agreed upon in the neural network community that using the SGD is more sensitive on a noise or newly provided samples and practically it shows a better performance [11].

Result: Accelerated gradient descent process with dynamic learning rate

while Termination condition meets do

if \# of Cancer < \# of Normal then

major = \# of Cancer / \# of Total;
minor = \# of Normal / \# of Total;

\( dlr = isClass(data) eq Normal \? major : minor; \)

else

\( dlr_{major} = \# of Normal / \# of Total; \)
\( dlr_{minor} = \# of Cancer / \# of Total; \)
\( dlr = isClass(data) eq Cancer \? major : minor; \)

end

do Back-propagation with momentum;
end

Algorithm 1: A pseudo code implementing an algorithm that dynamically assigns a learning rate

The last step is determining when the training process is to be terminated. As a terminating condition to identify the convergence of the training process, we used a Mean Squared Error (MSE) among several strategies such as using an epoch or a validation set. Since no general condition does exist to confirm whether the training process has converged or not, we cross-validated numerous conditions based on the given network size and we end up with the below condition.

\[ J(\theta; x_i; y_i) - J(\theta; x_{i-1}; y_{i-1}) < 1E - 3 \]  

5. RESULTS AND ANALYSIS

5.1 Evaluation Phase

For the comparison purpose, we evaluate our model by the following measures including the confusion matrix which are used in [13]: Precision, sensitivity, specificity, accuracy, and MCC.

- Precision: The proportion of the true positive against
all the positive results.

\[
\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}
\]

- Sensitivity or Recall: Ability of test to identify positive result; correctly identify patient has cancer who has cancer.

\[
\text{Sensitivity or Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}
\]

- Specificity: Ability of test to identify negative result; correctly identify patient is healthy who has no cancer.

\[
\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FN}}
\]

- MCC: Quality of binary classifications; a value between -1 to +1. A coefficient of +1 indicates a perfect prediction, 0 no better than random prediction, and -1 indicates total failure between prediction and observation.

\[
\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}
\]

where true positive (TP) is the number of samples that correctly identified as a cancer, true negative (TN) is the number of samples that correctly identified as a normal, false positive (FP) is the number of samples that incorrectly identified as a cancer, and false negative (FN) is the number of samples that incorrectly identified as a normal.

### 5.2 Results

In this work, we used the same dataset and data shuffling scheme as used in [13] for the comparison purpose. Furthermore, we doubled the dataset in order to see if the network is capable of dealing with the enlarged data in terms of generalizability. The 30 datasets are used in which a single dataset consists of 30 samples total, which are left and right breasts of 15 patients. The each sample contains the MTI measurements which are the values of permittivity, conductivity, and coordinates of the each tissues within the breast mass from the clinic trial at Seoul National University, Korea.

The table [1] and [2] shows the confusion matrix results from the newly constructed model that are trained with a different size of dataset. The table [3] shows the results from the model using the DLR without the momentum optimizer trained with the 15 dataset and the table [4] shows the results from the conventional model. The following analysis of the given tables only considers the aspect of a performance. For the models implemented using the DLR and the momentum, the overall performance increased when training with the 15 datasets. In terms of a performance, it can be seen that the addition of the momentum has not significantly affected. In the case of datasets, augmenting a small amount of training dataset by simple suffixing is considered to have resulted in an insignificant change in the performance. There are no big differences in the performance between the proposed models, but there is a big difference compared to the conventional model. Especially, the value of precision and accuracy are

<table>
<thead>
<tr>
<th>Table 1: Confusion matrix obtained from the model using DLR accelerated by momentum trained with 30 dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition determined by Doctor</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Confusion matrix obtained from the model using DLR accelerated by momentum trained with 15 dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition determined by Doctor</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3: Confusion matrix obtained from the model using DLR trained with 15 dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition determined by Doctor</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4: Confusion matrix obtained from the conventional model trained with 15 dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition determined by Doctor</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
<tr>
<td>Test outcome</td>
</tr>
</tbody>
</table>
more than 10% and 5%, respectively. Whether this difference is statistically significant is assessed at the end of this section.

The overall measures of the optimized model outperform the conventional model especially on the precision as shown in the Figure 4. When it comes to the model trained with the momentum optimizer, it shows a higher performance than the one without. When the size of the training dataset gets bigger, however, the performance tends to get slightly lower. However, the main purpose of using the momentum optimizer here is to shorten the convergence time, which is of course also leveraged to escape the local minima, rather than the qualitative improvement of the classification task.

The figure 5 illustrates that using the momentum enable the network to be converged a bit faster compare to the SGD. Since the momentum accelerate the gradient descent process, the model can be trained a bit faster and terminated few epochs earlier than the SGD.

The optimized model also shows a better quality on the classification in terms of the MCC value. Our model produces the MCC value of 0.72 while the conventional model produces 0.6 as shown in the Figure 6. Not only the results imply that the optimized model promises more reliable outcome on the binary classification with a stronger positive relationship, but also the fact that the training of a larger dataset for the optimized model does not degrade the quality of the binary classification indicates that this model is scalable to the dataset.

In order to verify if the differences of measures between the proposed and the existing model are statistically significant, we performed the paired t-test with the significance level (α) of 0.01 as below:

\[ H_{null} = \beta_1 \geq \beta_2, \]
\[ t_{\alpha, \beta} = \frac{\bar{D} - (\beta_1 - \beta_2)}{S_D/\sqrt{n}}, \]
\[ t_{\alpha, df} = \text{rejection region} \]

Where \( \beta \) is the measure, \( \bar{D} \) and \( S_D \) are a mean and a standard deviation of differences between the two groups, \( n \) is the number of samples and \( df \) is the degree of freedom, which is 15 and 14 respectively in this case. From the above equation, we can simply calculate the t-value and the rejection region of each measure as below:

\[ t_{0.01, \text{Precision}} \approx -3.54 < -2.62 \approx t_{0.01, 14}, \]
\[ t_{0.01, \text{Sensitivity}} \approx -0.86 > -2.62 \approx t_{0.01, 14}, \]
\[ t_{0.01, \text{Specificity}} \approx -1.00 > -2.62 \approx t_{0.01, 14}, \]
\[ t_{0.01, \text{Accuracy}} \approx -3.56 < -2.62 \approx t_{0.01, 14}, \]
\[ t_{0.01, \text{MCC}} \approx -3.75 < -2.62 \approx t_{0.01, 14} \]

From the above result, we can reject the null hypothesis for the measure of precision, accuracy, and MCC. Therefore, we can verify that the purposed model shows better performance in the precision, accuracy, and MCC for detection of abnormalities in the breast MT data. Based on the experimental results, it can be summarized that optimizing the size of a neural network and assigning a learning rate dynamically by earmarking higher learning rate onto the each training data points of the minority class can produce a better performance in the precision, accuracy, and MCC.
6. CONCLUSION

A CAD software, which is heavily affected by classification performance, is still widely under the development to support practitioners as a second opinion [1] [17]. Recently, MTI has been developed with various advantages that can be applied to this CAD system [17] [13]. In this work, as a pluggable classifier to the CAD software, an optimized MLP model taking the MT data is proposed and we boost the learning process by using DLR and momentum to detect breast anomalies. The existing model suggested by [2] has an error-prone network structure. Not only assigning excessive number of perceptron in a hidden layer but also using a static learning rate with a small amount of unbalanced dataset is possible to produce unreliable result that caused by either indeterminate equation problem or overfitting problem. In the proposed design, the optimized size of the neural network ensures that the learning process does not fall into the indeterminate equation problem by making connection no more than necessary between perceptron compared to the number of sample. In addition, assigning a learning rate dynamically in each training point with the momentum optimizer makes the network not only to be more flexible but also it shorten the convergence time of the training process. Based on the experimental results, it can be summarized that optimizing size of a neural network and assigning a learning rate dynamically by earmarking higher value onto each training data of minority class with the use of the momentum optimizer not only shorten the execution time but also produce the better performance in precision, accuracy, and MCC. The findings of this research work can be applied to this CAD system [17, 13]. In this work, as future works, possibly empowers the physician to make a better decision leviates the cost burden from the patients. As future works, we aim to focus on designing a deep neural network without any feature selection phase with massive dataset.

7. ACKNOWLEDGMENTS

This work was supported by ICT R&D program of MSIP/ IITP. [11- 911-01-108, Development of Precision MT System].

8. REFERENCES


**ABOUT THE AUTHORS:**

<table>
<thead>
<tr>
<th>Mr. Chulwoo Pack</th>
<th>Dr. Seong-Ho Son</th>
<th>Dr. Sung Y. Shin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mr. Chulwoo Pack received his B.E. degree in Computer Engineering &amp; Information Technology from University of Ulsan, Korea, in 2015, and his B.S. &amp; M.S. degrees in Computer Science from South Dakota State University in 2015 and 2017 respectively. Currently, he works for a research group called Image Analysis for Archival Discover (Aida) funded by NEH and IMLS while pursuing his Ph.D. studies at University of Nebraska-Lincoln. His main research interests include image processing, computer vision, and machine learning applications.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dr. Seong-Ho Son received his B.S. degree in control and mechanical engineering from Pusan National University, Korea, in 1997, and his M.S. &amp; Ph.D. degrees in mechanical engineering from Pohang University of Science and Technology (POSTECH), Korea, in 1999 and 2009 respectively. From 1999 to 2001, he worked for the technical research center of Daewoo Motors, Incheon, Korea. As a principal researcher, he has worked for the Radio and Satellite Research Division of ETRI, Daejeon, Korea, since 2001. His research interests include microwave imaging, focused microwave thermotherapy, and RF system development.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dr. Sung Y. Shin received his M.S. and Ph.D. degrees in Computer Science from University of Wyoming in 1984 and 1991 respectively. He has served as a professor and Graduate Coordinator of Computer Science at South Dakota State University since 1991. He has authored/coauthored over 200 technical peer reviewed papers in Software Fault Tolerance, Telemedicine, Medical Image Processing and Precision Agriculture. As a visiting scientist, he worked for the Space and Life Science Division at NASA Johnson Space Center in Houston from 1999 to 2002. He chaired the ACM Special Interest Group on Applied Computing (SIGAPP) for six years until 2015 and the ACM SAC conference in 2007, 2009, 2010, 2013, 2014, and 2017. Additionally, he has served on the Steering Committee of the conference since 2011.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>