



COMPUTATION TOOLS 2017

The Eighth International Conference on Computational Logics, Algebras,
Programming, Tools, and Benchmarking

ISBN: 978-1-61208-535-7

February 19 - 23, 2017

Athens, Greece

COMPUTATION TOOLS 2017 Editors

Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-
Universität Münster / North-German Supercomputing Alliance, Germany

COMPUTATION TOOLS 2017

Forward

The Eighth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (COMPUTATION TOOLS 2017), held between February 19-23, 2017 in Athens, Greece, continues an event under the umbrella of ComputationWorld 2017 dealing with logics, algebras, advanced computation techniques, specialized programming languages, and tools for distributed computation. Mainly, the event targets those aspects supporting context-oriented systems, adaptive systems, service computing, patterns and content-oriented features, temporal and ubiquitous aspects, and many facets of computational benchmarking.

We take here the opportunity to warmly thank all the members of the COMPUTATION TOOLS 2017 technical program committee, as well as all the reviewers. We also kindly thank all the authors that dedicated much of their time and effort to contribute to COMPUTATION TOOLS 2017. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also gratefully thank the members of the COMPUTATION TOOLS 2017 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that COMPUTATION TOOLS 2017 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of computational logics, algebras, programming, tools, and benchmarking. We also hope that Athens, Greece provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

COMPUTATION TOOLS 2017 Committee

COMPUTATION TOOLS 2017 Steering Committee

Ricardo Rocha, University of Porto, Portugal

Cristian Stanciu, University Politehnica of Bucharest, Romania

Ekaterina Komendantskaya, Heriot-Watt University, UK

COMPUTATIONAL TOOLS 2017 Industry/Research Advisory Committee

Miroslav Velev, Aries Design Automation, USA

Cornel Klein, Siemens AG, Germany

COMPUTATION TOOLS 2017

Committee

COMPUTATION TOOLS Steering Committee

Ricardo Rocha, University of Porto, Portugal

Cristian Stanciu, University Politehnica of Bucharest, Romania

Ekaterina Komendantskaya, Heriot-Watt University, UK

COMPUTATIONAL TOOLS 2017 Industry/Research Advisory Committee

Miroslav Velez, Aries Design Automation, USA

Cornel Klein, Siemens AG, Germany

COMPUTATION TOOLS 2017 Technical Program Committee

Lorenzo Bettini, DISIA - Università di Firenze, Italy

Ateet Bhalla, Independent Consultant, India

Narhimene Boustia, University Saad Dahlab, Blida 1, Algeria

Azahara Camacho, Universidad Complutense de Madrid, Spain

Emanuele Covino, Università degli Studi di Bari Aldo Moro, Italy

Marc Denecker, KU Leuven, Belgium

António Dourado, University of Coimbra, Portugal

Tommaso Flaminio, DiSTA - University of Insubria, Italy

George A. Gravvanis, Democritus University of Thrace, Greece

Fikret Gurgen, Bogazici University - Istanbul, Turkey

Hani Hamdan, Université de Paris-Saclay, France

Cornel Klein, Siemens AG, Germany

Ekaterina Komendantskaya, Heriot-Watt University, UK

Annie Liu, Stony Brook University, USA

Glenn Luecke, Iowa State University, USA

Roderick Melnik, Wilfrid Laurier University, Canada

Ralph Müller-Pfefferkorn, Technische Universität Dresden, Germany

Adam Naumowicz, University of Bialystok, Poland /

Cecilia Esti Nugraheni, Parahyangan Catholic University, Indonesia

Javier Panadero, Open University of Catalonia, Spain

Mikhail Peretyatkin, Institute of mathematics and mathematical modeling, Almaty, Kazakhstan

Alberto Policriti, Università di Udine, Italy

Enrico Pontelli, New Mexico State University, USA

Ricardo Rocha, University of Porto, Portugal

Patrick Siarry, Université Paris-Est Créteil, France

Cristian Stanciu, University Politehnica of Bucharest, Romania

Martin Sulzmann, Karlsruhe University of Applied Sciences, Germany

James Tan, SIM University, Singapore
Miroslav Velev, Aries Design Automation, USA
Marek B. Zaremba, Université du Québec, Canada

Copyright Information

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

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

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

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

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

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

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

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

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

Table of Contents

Composite Event-Driven Programming <i>Fredy Cuenca</i>	1
A Logic-based Service for Verifying Use Case Models <i>Fernando Bautista and Carlos Cares</i>	5
An Answer Set Solution for Information Security Management <i>Carlos Cares and Mauricio Dieguez</i>	11

Composite Event-Driven Programming

New Concepts for New Types of Interaction

Fredy Cuenca

School of Mathematical Sciences and Information Technology
Yachay Tech
San Miguel de Urcuquí, Ecuador
Email: fcuenca@yachaytech.edu.ec

Abstract—Implementing multi-touch and multi-modal systems requires splitting the code across several event handlers, which complicates programmers work. The present paper finds the root of this problem in the event-driven paradigm; more concretely, in the fact that event-driven languages lack abstractions for representing event sequences. It then suggests to augment event-driven languages so that programmers can have the possibility to define event sequences –herein called composite events– that can then be bound to event handlers. The main features of the composite event-driven language developed by the authors are outlined, as well as its benefits and problems. The paper suggests that, since its design, the event-driven paradigm was tailored for mouse-based interactions, and it may be important to question its suitability for implementing multi-touch and multi-modal interactions.

Keywords—Multi-modal Systems; Multi-touch Systems; Interactive Systems; Event Languages; Composite Events.

I. INTRODUCTION

Many researchers agree that implementing (multi-)touch and multi-modal systems results in programs that are difficult-to-read and difficult-to-maintain [1][2][3]. In the domain of (multi-)touch systems, even simple gestures, such as the *pinch-to-zoom* gesture, require the system to handle a stream of *touch-down*, *touch-move*, and *touch-up* events, from within the intention of the user to enlarge a particular region of the touchscreen has to be unveiled. Similarly, multi-modal commands, like speech-and-pointing commands, will be perceived by multi-modal systems as a series of speech events and pointing events. Thus, such multi-modal systems have the difficult task of having to continuously identify which speech events and pointing events are part of the same command.

The implementation of those interactions that are reflected, in the system, as sequences of interrelated events, forces programmers to litter their code with a multitude of flags and global variables that have to be updated across different event handlers in order to keep track of the event sequences. The resulting difficult-to-read, difficult-to-maintain “callback-soup” [1][2] is not a consequence of bad programming habits or poor comprehension of event-driven principles. Rather, it is accidental complexity: complexity caused by the languages and tools chosen for programming [4]. This type of complexity can only be reduced by selecting or developing better programming languages and tools [4]. The present work intends to shed some light on how to develop better languages and tools.

We believe that the appearance of the “callback-soup” is largely due to the fact that event-driven languages, which

are widely used to implement interactive systems [5][6], only offer abstractions, called events, for representing simple user actions, such as a touch-down or a speech input; but these languages do not offer abstractions for representing sequences of user actions.

This paper proposes a programming model that enables programmers to compose events. In the proposed model, there is an abstraction called composite event, this being a programmer-defined event sequence. Composite events are defined by connecting primitive events, such as touch events or speech inputs, through a set of operators, where each operator represents a temporal, spatial, or semantic relation among their operands. Composite events can then be bound to event handlers, callback functions that implement the system’s runtime behavior.

For instance, two basic interactions with a touchscreen photo viewer can be described by binding the composite event *touch-flicking-left* to the event handler `ShowNextPhoto()`, and the speak-and-touch *remove-this* event to the event handler `RemovePhoto()`. The two aforementioned composite events would be defined by the programmer as a combination of touch events (former case) or as a mix of touch events and speech events (latter case).

In the proposed programming model, at runtime, the event handlers are to be launched every time their associated composite events are automatically detected by a composite event-driven tool. This model will save programmers from having to implement a supervisory mechanism for tracking event sequences; such a mechanism would be incorporated in the composite event-driven tool to be exploited by programmers. By delegating the detection of event sequences to the composite event-driven tool, programmers can clean their source codes of the flags and global variables that were necessary for this task when using event-driven languages.

The remainder of this paper proceeds as follows: In Section 2, the proposed approach is compared against others that also intend to ease the creation of multi-modal/multi-touch interaction. Section 3 outlines the main features, gains, and limitations of a composite event-driven tool that was implemented as part of a PhD project. Finally, Section 4 argues in favor of augmenting event-driven languages with composite events.

II. RELATED WORK

The benefits of using composite events for rapid prototyping of multi-modal systems have already been highlighted

[3][7]. Additionally, this paper reports similar gains when prototyping (multi-)touch gestures and, most importantly, proposes composite events as a unified solution to the “callback soup” problem that infects both multi-modal and multi-touch interactive systems.

Other (mostly visual) languages have also been proposed with the aims of easing the “callback soup” problem. One important contrast is that while our composite event-driven model allows describing interactions in terms of events and event binding, other existing languages require concepts (e.g., Petri nets and block diagrams) and programming practices (e.g., depicting visual models) that may be unfamiliar to programmers of interactive systems. As the rankings of programming popularity published by IEEE [6] and TIOBE [5] attest, programmers are more accustomed to textual, event-driven languages. Given that familiarity with a language has a strong, positive influence on programming language adoption (even stronger than intrinsic properties of the language, such as performance, reliability, and simple semantics) [8], the proposed programming language retains the textual and event-driven nature of mainstream programming languages. Other more concrete differences of our approach and existing languages can be found below.

A. Multi-modal interaction description languages

In Squidy [9], multi-modal interactions are represented as block diagrams that programmers can use to channel and transform the data coming from different input modalities to the application. One issue of this model is that each modality has its own independent channel. Data from different modalities must be collected in the application as the human-machine interaction occurs. With our approach, a composite event can be defined by combining events from the same or different modalities. The data carried by these events is stored in parameters that arrive to the application all at once –no need for queuing events in the application.

Similar to our model, SMUIML [10] allows composing events so that each composite event can be bound to one event handler. At runtime, these handlers are launched once their associated composite events have occurred. We generalize this approach by allowing programmers to attach event handlers to very specific stages of a composite event—in our model, event binding has a time component. This allows launching several event handlers at different moments of the lifecycle of a composite event, which makes it possible to provide the end user with partial feedback.

ICO [11] is a formal language for modeling both multi-modal and multi-touch interactions. It has an underlying mathematical apparatus that allows predicting properties of the interaction in static time, without having to run the ICO model. One of its drawbacks is that Petri nets were not tailored for modeling interaction, thus ICO models do not map close to the problem domain. ICO users have to tweak their interaction models to fit them into a Petri net. Our approach, instead, makes use of a domain-specific notation that has designated symbols for representing time constraints among events and special keywords for specifying modalities.

B. (Multi-) touch gesture definition languages

Besides the already reported gains experienced when prototyping multi-modal systems [3][7], composite events also

bring about advantages over existing, salient gesture definition languages, such as GDL [12], Proton [1] and GestIT [2]. To a greater or lesser extent, all these languages have proven to ease the description of touch gestures: programmers can define gestures in a declarative fashion without having to write fine-grained code for tracking the gesture state.

GDL [12] is intended for simple description of touch gestures that can be used across multiple hardware platforms. A touch gesture is defined as a set of rules that must be met by the raw touch data along with the value(s) to be returned when the gesture is detected. GDL allows defining multi-stroke gestures (e.g., a cross) as long as the strokes can be issued sequentially. Our language, in contrast, allows defining gestures involving both sequential and parallel strokes (e.g., simultaneous vertical flicks). Furthermore, unlike the proposed language, GDL does not allow specifying temporal constraints.

On the other hand, Proton allows users to represent gesture interaction as tablatures. A user study proved that tablatures are easier-to-comprehend than event-callback code [13]. The expressiveness of Proton was shown by implementing multi-user touch-based applications like the classic Pong game, a tennis-like game between two opponents [13]. One issue with Proton is its lack of time variables, which makes it difficult to calculate the duration of a gesture, for instance. In contrast, our approach allows defining and maintaining different types of variables (including time variables) throughout the composite event lifecycle.

As to the Petri nets-based language GestIT, partial feedback is only possible by decomposing gestures definitions into smaller, sub-gestures definitions. This is because GestIT only launches event handlers at the end of a (sub-)gesture. Our approach does not force programmers to make such decompositions because multiple event handlers can be bound to different stages of one single gesture. Furthermore, Proton and GestIT do not include timeout events, which make it unnecessarily complex to define recursive gestures. For instance, the single, double, and triple tap require three separate definitions with Proton and GestIT. In our approach, timeout events exist and can be connected with any input event (e.g., touch events or speech inputs) to be part of a composite event. By using timeout events, our approach makes it possible to describe the three aforementioned gestures with one single definition: a sequence of N taps that end after a period of “silence”.

C. Languages for reactive systems

It should be made clear that the goal of the proposed approach differs from that of languages such as P [14], Esterel [15], or Lustre [16].

P is oriented more towards the development of distributed systems. Therefore, the type of interaction to be modeled with P is among the components of the intended system. Such a component-component interaction is uncoordinated and consists of messages sent from different sources. In contrast, the proposed language is designed for human-machine interaction, a type of interaction where the inputs are coordinately issued by one single agent, the end user. Due to this fundamental difference, P focuses on forcing asynchronous events to be responded within a reasonable timeframe. For this, P includes notations for explicit declaration of event deferrals. Our language, instead, focuses on describing relations between user actions and system responses, which can be done concisely with the proposed composite event binding notations.

Similar contrasts can be found against Esterel or Lustre, which are intended to develop real-time, embedded systems. Thus, these are much closer to P than to the proposed language.

III. A FIRST COMPOSITE EVENT-DRIVEN TOOL

In the context of a PhD research, a composite event-driven language along with its supporting tool was developed and evaluated for rapid prototyping of multi-modal systems [3][17][7].

A. Automatic detection of composite events

A composite event-driven tool must be in charge of tracking every programmer-defined composite event and launching its associated event handler(s) in a timely manner. In our particular implementation, every composite event is internally represented as a finite state machine [3]. The human-machine interaction is described with a textual notation, as a mapping of composite events to event handlers, and, under the hood, the proposed tool generates a set of finite state machines by means of specialized algorithms [3]. As the constituent events of a composite event occur, in the specified order, its reciprocal finite state machine switches to different states. Finally, the end-node of this machine is reached when its reciprocal composite event has occurred. Programmers can attach event handlers to every node or link of a finite state machine when writing event binding code [17]. Given that a composite event can be reused in the definition of other, more complex composite events, our finite state machines are hierarchical.

B. Experimental results

This language was compared against C#, a mainstream event-driven language, by means of a within-subjects experiment. A user study involving twelve participants (experienced developers) was conducted to compare programming efficiency. After modifying an interaction model with both the composite event-driven language and the baseline language, it was revealed that the former leads to higher completion rates, lower completion times, and less code testing [7]. Another study with non-developers is being conducted to measure whether the proposed language is simple enough to be understood and used as a discussion tool within multidisciplinary teams (e.g., in a robotics project). We have not yet conducted experiments about tool performance (e.g., recognition rate or recognition speed of composite events). For now, our focus is on evaluating the feasibility and efficacy of composite event-driven programming rather than the efficiency of our particular implementation.

C. Expressiveness

The expressiveness of the proposed language has already been evaluated by implementing a variety of interactions involving mouse gestures, keystrokes, and speech inputs [3][17]. Later, as part of a PhD research, we implemented a proof-of-concept application that recognizes hand gestures, body movements, and a variety of touch gestures (e.g., single-stroke, multi-stroke, free-form, and multi-touch). More recently, composite events have also been applied in the field of robotics [18]. The size of the developed applications is small: we always used less than 30 composite events in our applications. We still do not have indications, neither in favor nor against, of whether the easiness-to-maintain will increase linearly with the size of the applications or not.

D. Limitations

The current version of the proposed composite event-driven language does not include general-purpose constructs (e.g., for's and if's). It is a declarative language that includes notations for describing human-machine interaction as mappings of user actions to system responses or, more concretely, as mappings of composite events to event handlers. But the fine-grained code required to implement the event handlers and the graphical user interface has to be written with a general-purpose language, as part of a canned application that has to be imported into our composite event-driven tool. The separation of interaction code from application code brought about many problems, such as the inability to create autonomous executable files (e.g., the imported application is developed with a general-purpose language whose syntax is unknown to the developed composite event-driven tool), the excessive amount of function calls required to exchange data between the external application and the composite event-driven tool (e.g., application variables have to be maintained by calling functions; these variables cannot be set directly from the composite event-driven language), and the difficulty to debug a program that is broken into two separate, independent pieces (e.g., composite event variables are traced within the proposed tool whereas external application variables are traced with external tools), among others.

IV. DISCUSSION

The proposed shift up from events to composite events would be the reflection of a fundamental change in human-computer interaction: in the past, systems were mainly commanded by simple user actions such as clicks on widgets; but modern multi-modal/multi-touch systems are intended to be commanded by several coordinated user actions, such as pointing-and-speech. These sets of coordinated actions would be abstracted as composite events in the proposed model.

It is true that existing event-driven languages have enough expressiveness to develop multi-modal and multi-touch interactive systems, but the complexity of the resulting code can be reduced by using composite events: the interaction state that has to be updated manually with event-driven languages is maintained automatically by composite event-driven tools. This benefit was noticed by twelve participants of a comparative user study: All of them agreed that the task of modifying interaction descriptions was simpler when using composite events than when using C#, a mainstream event language [7].

The event-driven paradigm was inspired in academic research (e.g., University of Alberta UIMS and Sassafra) carried out in the 80's [19], when mouse-based interactions were predominant. By 2000, the mouse-based interactions introduced by the Apple Macintosh, in 1984, had been widely adopted by almost all applications [19]. Mainstream event-driven languages such as Visual Basic and Java were released within that period; more concretely, in 1991 and 1995, respectively. Therefore, it should not be a surprise to realize that the event-driven paradigm with all its underlying concepts was likely tailored to deal with a type of interaction that is much simpler than multi-touch and multi-modal interaction, which now claim silently their own paradigm and tooling.

In a seminal paper, published in 2000, when discussing event-driven languages, Myers et al. [19] foretold that, in order to deal with the then-emerging modalities, such as touch and

speech, a new paradigm may be needed. To the best of our knowledge, no one has yet given a clue about how to start building the new programming paradigm. Based on 4+ years of research, this paper is suggesting one direction: to extend the fundamental concept of event to composite events.

We expect that the first, positive results obtained after implementing our programming model can encourage other researchers and practitioners to create more full-fledged composite event-driven tools, which, aside from including code editors, runtime environment, and debugging tools, like our tool, must also include interface builders and a language enriched with general-purpose constructs. With such a set of tools, programmers will no longer need to separate application code from interaction code and, thus, the aforementioned limitations might disappear.

ACKNOWLEDGEMENTS

We would like to acknowledge the effort of our former colleagues of Hasselt Universiteit, namely, Jan Van der Bergh, Kris Luyten, and Karin Coninx, for helping us implement a first version of the vision exposed in this paper.

REFERENCES

- [1] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala, "Proton: multi-touch gestures as regular expressions," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12), 2012.
- [2] L. Spano, A. Cisternino, F. Paterno, and G. Fenu, "Gestit: a declarative and compositional framework for multiplatform gesture definition," in Proceedings of the EICS'13. ACM, 2013.
- [3] F. Cuenca, J. Van den Bergh, K. Luyten, and K. Coninx, "A domain-specific textual language for rapid prototyping of multimodal interactive systems," in Proceedings of EICS'14. ACM, 2014.
- [4] C. Scholliers, L. Hoste, B. Signer, and W. De Meuter, "Midas: a declarative multi-touch interaction framework," in Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction. ACM, 2011.
- [5] "TIOBE Index," <http://www.tiobe.com/tiobe-index/>, 2016, [Online; accessed 21-December-2016].
- [6] "IEEE Spectrum," <http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages/>, 2016, [Online; accessed 21-December-2016].
- [7] F. Cuenca, J. V. d. Bergh, K. Luyten, and K. Coninx, "A user study for comparing the programming efficiency of modifying executable multimodal interaction descriptions: a domain-specific language versus equivalent event-callback code," in Proceedings of the PLATEAU'15. ACM, 2015.
- [8] L. A. Meyerovich and A. S. Rabkin, "Empirical analysis of programming language adoption," ACM SIGPLAN Notices, vol. 48, no. 10, 2013.
- [9] W. König, R. Rädle, and H. Reiterer, "Interactive design of multimodal user interfaces," Journal on Multimodal User Interfaces, vol. 3, no. 3, 2010.
- [10] B. Dumas, D. Lalanne, and R. Ingold, "Description Languages for Multimodal Interaction: A Set of Guidelines and its Illustration with SMUIML," Journal of multimodal user interfaces, vol. 3, no. 3, 2010.
- [11] D. Navarre, P. Palanque, J.-F. Ladry, and E. Barboni, "ICOs: A Model-Based User Interface Description Technique dedicated to Interactive Systems Addressing Usability, Reliability and Scalability," ACM Transactions on Computer-Human Interaction, vol. 16, no. 4, 2009.
- [12] S. H. Khandkar and F. Maurer, "A domain specific language to define gestures for multi-touch applications," in Proceedings of the 10th Workshop on Domain-Specific Modeling. ACM, 2010.
- [13] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala, "Proton++: a customizable declarative multitouch framework," in Proceedings of the 25th annual ACM symposium on User interface software and technology. ACM, 2012.
- [14] A. Desai, V. Gupta, E. Jackson, S. Qadeer, S. Rajamani, and D. Zufferey, "P: safe asynchronous event-driven programming," ACM SIGPLAN Notices, vol. 48, no. 6, 2013.
- [15] G. Berry and G. Gonthier, "The estereel synchronous programming language: Design, semantics, implementation," Science of computer programming, vol. 19, no. 2, 1992.
- [16] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language lustre," Proceedings of the IEEE, vol. 79, no. 9, 1991.
- [17] F. Cuenca, J. Van den Bergh, K. Luyten, and K. Coninx, "Hasselt uims: a tool for describing multimodal interactions with composite events," in Proceedings of the EICS'15. ACM, 2015.
- [18] J. Van den Bergh, F. Cuenca Lucero, K. Coninx, and K. Luyten, "Toward specifying human-robot collaboration with composite events," 2016.
- [19] B. Myers, S. E. Hudson, and R. Pausch, "Past, present, and future of user interface software tools," ACM Transactions on Computer-Human Interaction (TOCHI), vol. 7, no. 1, 2000.

A Logic-based Service for Verifying Use Case Models

Fernando Bautista, Carlos Cares

Computer Science and Informatics Department, University of La Frontera (UFRO)
Temuco, Chile

Email: fernandobautis@gmail.com, carlos.cares@ceisufro.cl

Abstract—Use cases are a modeling means to specify the required use of software systems. As part of UML (Unified Modeling Language), it has become the *de facto* standard for functional specifications, mainly in object-oriented design and development. In order to check these models, we propose a theoretical solution by adapting a general quality of models framework (SEQUAL), and, following our approach, a rule-based solution that includes both expert-based and definition-based rules. In order to promote a distributed set of quality assessment services, a Web service has been developed. It works on XMI (XML Metadata Interchange) files which are parsed and verified by Prolog clauses.

Keywords—Rule-based quality; UseCase verification; Logic-based services; XMI; Prolog.

I. INTRODUCTION

In Software Engineering, use cases are a means to specify the required uses of software systems. Typically, they are used to represent what the system is supposed to do. Use cases are part of UML (Unified Modeling Language) specifications and they have become so wide spread that they are now considered the *de facto* standard for requirements specification of object-oriented software systems [1].

Quality assurance of use cases is a common topic in Software Engineering. For example, some heuristics including UML use cases, have been proposed for model revision [2]. Moreover, preparing good use cases for connection with other static and dynamic models remains important as they are a key representation for verification and validation [3].

Other studies have tried to assist in the semi-automatic verification / validation of use cases. Kotb and Katayama [4] present a novel approach to check the verification of the use case diagrams. Shinkawa [5] proposes a formal verification process model for UML use case, and Gruner [6] details a meta model of possible relations between use cases, which may, in the future, be implemented in Prolog. However, these proposals assume that the set of steps in order to implement it (known as basic course or basic flow) is always part of the use case specification. However, this assumption is not broadly true, and, moreover, it is shown that the way of this narrative presents ambiguity [7]. From a formal perspective, an interesting summary is found in [8], where different lapses are identified for the analyzed approaches.

The objective is to show a tool for supporting a quality assessment process of use case diagrams, even, when some of these use cases have no proper narrative inside of them. The theoretical base is given by SEQUAL (SEmiotic QUALity) [9]. It is a highly spread quality framework for models and it addresses different kinds of qualities including syntactic, semantic, pragmatic and social qualities which make it very complex to include all these quality perspectives from the scratch. Under this assumption, a rule-based system is a

modular and scalable solution in order to initially implement some types of verifications and then another group of them under an incremental development.

In this paper, we present a first Prolog prototype, as proof of concept, of a tool that can aid the quality assessment of a use case diagram. Moreover, we have implemented it as a Web service in order to illustrate that logic-based solutions can also be part of key quality assessment process in cloud-based software development environment.

In order to check use case models and their application, in Section 2 we present a set of verification syntactic and semantic rules and how they have been derived from SEQUAL conceptual framework. In Section 3, we show how the derived rules have been implemented in Prolog clauses into our first version of the Use Case Checker (UC2). In the Conclusion section, we summarize the contributions and we outline the future work on two tracks: technological improvement of UC2 and broadening the scope of the quality assessment approach.

II. WHERE DO VERIFICATION RULES COME FROM

In order to derive verification rules, we have used the SEQUAL conceptual framework [9]. Although it was proposed more than 20 years ago, it has had great influence on verifying the quality of multiple kinds of models [10]. However, it has never been used to verify use case models. The SEQUAL framework is based on semiotic, hence it includes syntactic, semantic and pragmatic qualities where interpretations and subjectivity of modelers and users are also considered. We have specialized the SEQUAL framework as it is illustrated in Figure 1. Therefore, from a theoretical point of view, this specialization is our quality of models theory's contribution, and, our rule-based solution, may also be considered a proof of concept to it.

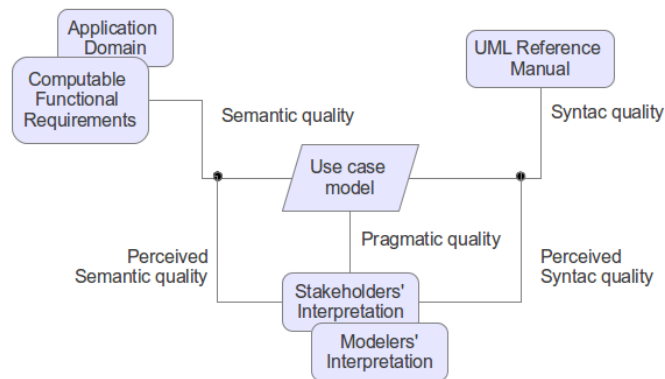


Figure 1. SEQUAL specialization for use case models

Following our SEQUAL specialization, we have generated rules to principally verify syntactic quality, but also we have included semantic quality rules in order to illustrate its implementation.

A use case model may contain several use case diagrams, and the quality application developed here is applied only to one use case diagram. Therefore, the process of verifying a use case model implies verifying all its containing diagrams.

The following rules were generated from the definition of OMGs (Object Management Group), applicable to UML use cases (specified in the UML superstructure version 2.4.1). These rules mainly verify the syntactic quality of a use case diagram, as defined in the quality model described above. Later, some of these rules will be verified automatically by the software prototype developed.

The generated rules are listed and described by adding an identification number. However, this numbering is arbitrary and does not represent any kind of hierarchy. In each case, we have illustrated the right case in opposition to a wrong case.

A. Rule 1 - There must be clear system boundaries

A use case diagram represents the interaction of an actor with the system. The UML Reference Manual says that the subject is the system under consideration to which the use cases apply [1]. Thus, it does not make any sense to model a use case diagram without a subject or system boundary, which must be represented either by a package or a classifier. Given that the existence of a system boundary may be verified in an explicit diagram structure, we classify this rule as part of the syntactic verification of a use case model. It is illustrated in Figure 2.

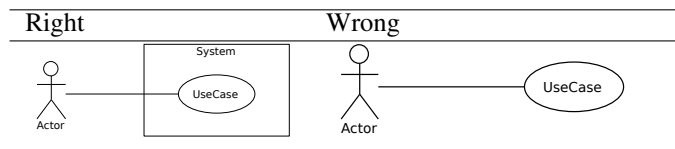


Figure 2. Example of Rule 1

B. Rule 2 - Actors must not be isolated

The UML Reference Manual says that an actor specifies a role played by a user or any other system that interacts with the subject [1]. Modeling an isolated actor does not make any sense. Although the diagram shows a system boundary and use cases in it, there is not possible to inference that the actor interacts with all or some of the present use cases. Given that the existence of an isolated actor may be verified in an explicit diagram structure, we classify this rule as part of the syntactic verification of a use case model. It is illustrated in Figure 3.

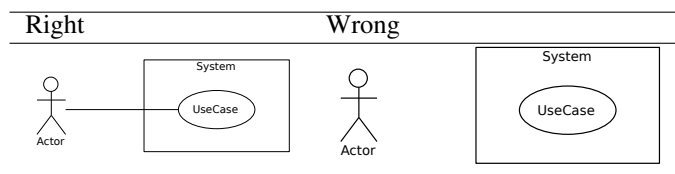


Figure 3. Example of Rule 2

C. Rule 3 - Use cases must not be isolated/inaccessible

The UML Reference Manual says that a use case represents a behavior of the system in which an actor or another system interact with it. Therefore, an isolated use case can never be executed. Here we refer not only to the fact of being isolated from actors interactions, but also of other dependencies coming from its interaction with other use cases. Given that the existence of an isolated actor may be verified in an explicit diagram structure, we classify this rule as part of the syntactic verification of a use case model. It is illustrated in Figure 4.

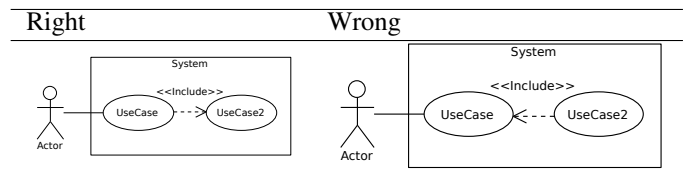


Figure 4. Example of Rule 3

D. Rule 4 - Actors must not be inside the system

The UML Reference Manual says that an actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject [1]. Therefore, the actor should not be inside the system or subsystem being modeling. This kind of diagram does not have nested representation of involved system or subsystems. Therefore, there is no case in which it can be possible. Given that an actor being inside the system boundary may be verified in an explicit diagram structure, we classify this rule as part of the syntactic verification of a use case model. It is illustrated in Figure 5.

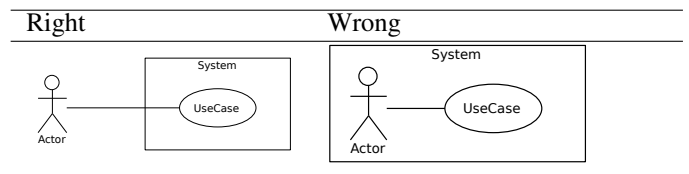


Figure 5. Example of Rule 4

E. Rule 5 - Use cases should be within system boundaries

The UML Reference Manual says that a use case is the specification of a set of actions performed by a system, which yields an observable result [1]. We, therefore, created a rule that a use case should not be outside the system boundary because it is necessary to specify which system is the owner of that behaviour. Given that a use case is part of a system, i.e., that it belongs to an existing system, then it may be verified by parsing the diagram structure, then we classify this rule as part of the syntactic verification of a use case model. It is illustrated in Figure 6.

F. Rule 6 - Use cases must start with a verb

The UML Reference Manual says that a use case is the specification of a set of actions performed by a system [1]. Therefore, its description requires, at least, a verb, i.e., the verb that specifies that action; additionally, as a quality description,

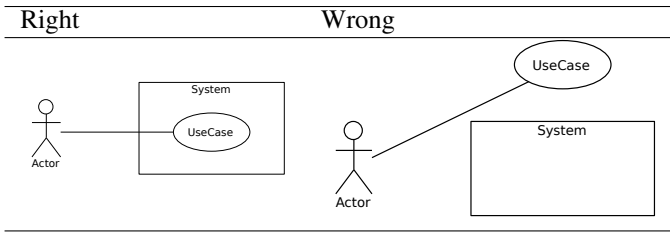


Figure 6. Example of Rule 5

we would ask for that verb to appear in the first place of the sentence that describes the use case behaviour.

In Spanish language, in which we worked, the verb should properly be used in the imperative form, but given that this mode is not commonly used, we recommend the use of verbs in its infinitive form. This last recommendation is irrelevant in English where these two forms are identical.

Given that verifying the verb form and its position in a sentence, i.e., that it is necessary to look for external references beyond UML structure, we classify this rule as part of the semantic verification of a use case model. It is illustrated in Figure 7.

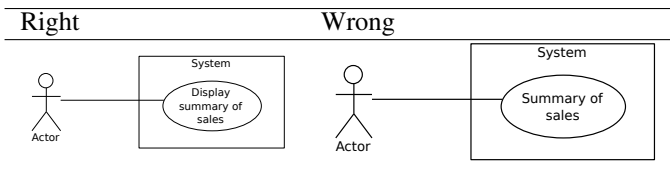


Figure 7. Example of Rule 6

G. Rule 7 - Use cases must represent an observable behavior of the system

The UML Reference Manual says that a use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system [1]. We therefore created a rule that a use case should represent actions of the system, which are observable to the actor, written from the perspective of the system.

In order to make this possible we have described a set of typical verbs of a system behaviour and other that can be warnings in the redaction of the use case descriptions. For example, a common mistake in use cases is the use of non-observable verbs like “To save” or “To register”. Also a common mistake is the use of a human actions like entering data or including high level behaviours like selecting or managing. In order to verify this feature, we have used additional list of non-observable actions (verbs) of classical system behaviours and another of classical human behaviours under a system interaction in order to give warning about them. Given that these lists are external to the own nature of UML structures, we classify this rule as part of the semantic verification of a use case model. It is illustrated in Figure 8.

H. Rule 8 - Actors names should be singular

UML superstructure says: A single physical instance may play the role of several different actors [1]. Class modeling

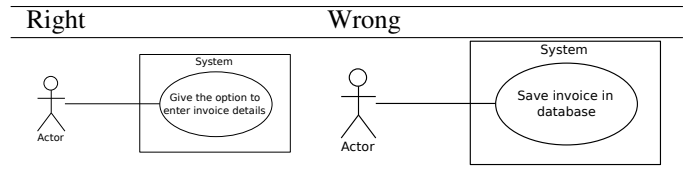


Figure 8. Example of Rule 7

assumes that actors are classes, which are required to follow the standard class nomination, including its expression in the singular case. Due to this verification goes beyond the UML structure, i.e. it exceeds syntax, we have classified this rule as part of the semantic verification of a use case model. It is illustrated in Figure 9.

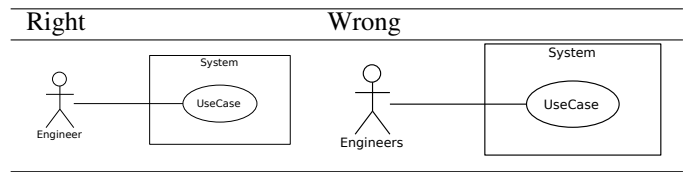


Figure 9. Example of Rule 8

I. Rule 9 - Computable verbs

The verb represents the behavior of the system, therefore the action represented by the verb must be unambiguous and capable of being implemented into a computer system. Therefore, using also a reference list of verbs we can warn about the use of a verb that is not part of “computable verbs”. Given that the existence of non computable verbs may not be verified in an explicit diagram structure, we classify this rule as part of the semantic verification of a use case model. It is illustrated in Figure 10.

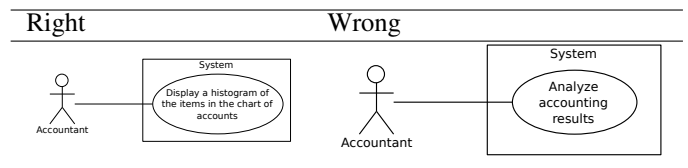


Figure 10. Example of Rule 9

III. PROLOG CLAUSES FOR USE CASES VERIFICATION

The system consists of software for analyzing a use case diagram (XMI file), applying the rules generated and subsequently delivering the result of the application in JSON (JavaScript Object Notation) format. This software will be implemented in Web technologies as a service. In Figure 11, the inputs and outputs of the system are represented.

A. Involved technologies

The XMI (XML Metadata Interchange) standard was defined for the exchange of UML diagrams. XMI is an XML-based integration framework for the exchange of models, and any kind of XML data. Thus XMI is used in the integration of tools, repositories, and model-related applications in general.

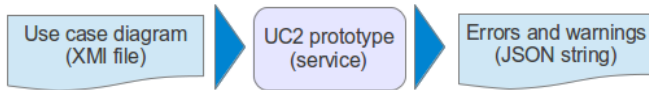
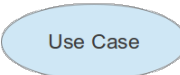
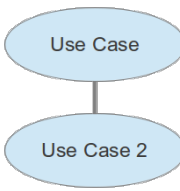
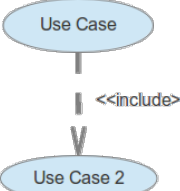


Figure 11. System view of Use Case Checker (UC2) Prototype

The framework defines rules for generating XML schemas from a metamodel based on the Metaobject Facility (MOF). Although XMI is most frequently used as an interchange format for UML, it can be used with any MOF-compliant language [11].

Table I shows the XMI representation of some elements of a use case diagram.

TABLE I. XMI REPRESENTATION OF POSSIBLE USE CASE EXPRESSIONS

Element	XMI Representation
Use Case 	<pre><packagedElement name="{NAME}" ↪ xmi:id="{ID}" xmi:type="uml:UseCase"> </packagedElement></pre>
Association 	<pre><packagedElement xmi:id="{ID}" ↪ xmi:type="uml:Association"> <ownedEnd aggregation="none" ↪ association="{ID ELEMENTO ↪ ORIGEN}" xmi:id="{ID}" ↪ xmi:type="uml:Property"> </ownedEnd> <ownedEnd aggregation="none" ↪ association="{ID ELEMENTO ↪ DESTINO}" xmi:id="{ID}" ↪ xmi:type="uml:Property"> </ownedEnd> </packagedElement></pre>
Include 	<pre><!-- An include is nested in a ↪ packageElement Use Case, this ↪ use case correspond to the UC ↪ arrow source. --> <include addition="{ID TARGET UC}" ↪ xmi:id="{ID}" ↪ xmi:type="uml:Include"> </include></pre>

SWI-Prolog is a portable implementation of the Prolog programming language. SWI-Prolog aims to be a robust, scalable implementation supporting a wide range of applications, providing interfaces to other languages and providing support for parsing XML and RDF (Resource Description Framework) documents. The system is particularly suited for server applications due to its support for multithreading and HTTP server libraries [12].

To develop UC2 we have used SWI-Prolog v 7.2.3, which provides some improvements over version 6.x.x in the creation and reading of JSON structures.

This component is responsible for implementing the pre-

viously generated rules. For this proof of concept, only some of these rules are shown.

For the implementation of the rules that verify the quality of a use case diagram, we defined a set of logical Prolog rules which allow us: 1) to identify the elements of a use case diagram within a file XMI; and 2) to see if these elements comply with the quality rules generated.

Figure 12 shows the code that identifies part of the elements within an XMI file.

```
%useCase(XML, ID, NAME)
useCase(XML, ID, NAME) :-
  xpath(XML,
    ↪ //packagedElement(@'xmi:type'=
    ↪ 'uml:UseCase'), A),
  xpath(A, /self(@'xmi:id'), ID),
  xpath(A, /self(@'name'), NAME).

%association(XML, ArrowSource,
  ↪ ArrowTarget)
association(XML, Source, Target) :-
  xpath(XML,
    ↪ //packagedElement(@'xmi:type'=
    ↪ 'uml:Association'), A),
  xpath(A, ownedEnd(1), O),
  xpath(O, /self(@'type'), Source),
  xpath(A, ownedEnd(2), U),
  xpath(U, /self(@'type'), Target).

%extend(XML, ArrowSource, ArrowTarget)
extend(XML, Source, Target) :-
  xpath(XML, //packagedElement, A),
  xpath(A, /self(@'xmi:id'), Source),
  xpath(A, extend, E),
  xpath(E,
    ↪ /self(@'extendedCase'), Target).
```

Figure 12. Prolog clauses for parsing XMI input files

When the above-defined rules are implemented, they must pass as a text parameter in the XMI file (coded as a variable named "XML").

When the Prolog Rules have been identified through the elements of a use case diagram, the hierarchical relationship between them must be known. To do this, we defined the Prolog Rule shown in Figure 13.

```
%in(XML, ChildElement, ParentElement)
in(XML, Child, Parent) :-
  xpath(XML, //packagedElement, A),
  xpath(A, /self(@'xmi:id'), Parent),
  xpath(A, packagedElement, E),
  xpath(E, /self(@'xmi:id'), Child)
```

Figure 13. Prolog clause for getting container-content relationships

B. Service Functions

In this section, we describe some of the previous rules in order to illustrate the high cohesion and low coupling reached by this Prolog implementation which has been implemented

as a REST service.

Unrelated Actor. This rule is the implementation in SWI-Prolog of Quality Rule 2 “Actors must not be isolated”. Figure 14 shows the diagram in the input file and its corresponding JSON output.

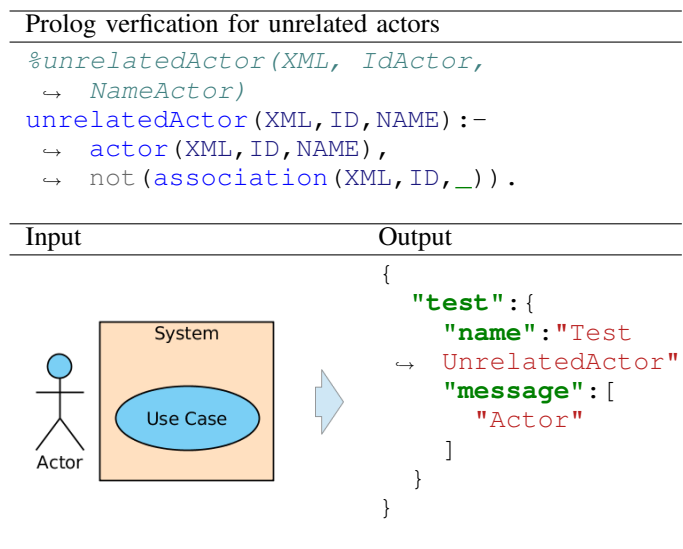


Figure 14. Implementation and sample for Rule 2: “Actors must not be isolated”

Isolated Use Case. This rule is the implementation in SWI-Prolog of Quality Rule 3 “Isolated/Inaccessible use case”. Figure 15 shows the diagram in the input file and its corresponding JSON output.

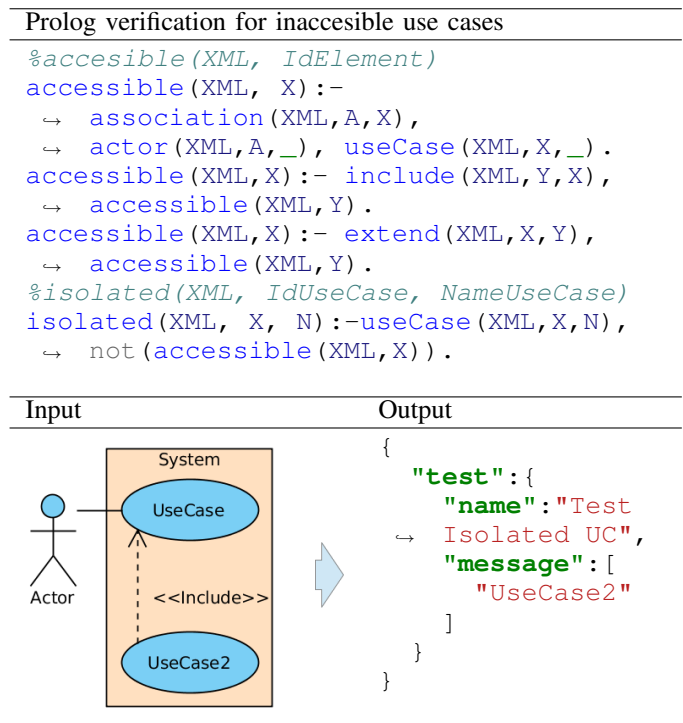


Figure 15. Implementation and sample for Rule 3: “Use cases must not be isolated/inaccessible”

Actor Inside the System. This rule is the implementation in SWI-Prolog of Quality Rule 4 “Actors must not be inside the system”. Figure 16 shows the diagram in the input file and its corresponding JSON output.

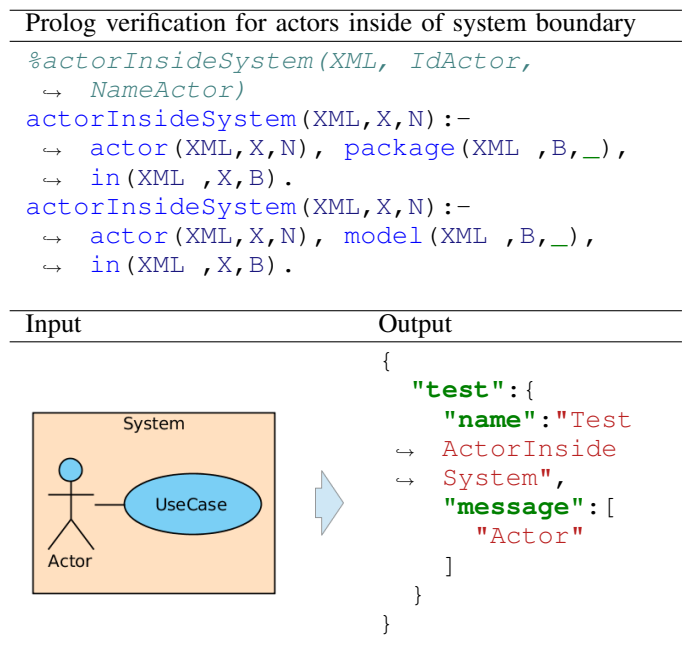


Figure 16. Implementation and sample for Rule 4: “Actors must not be inside the system”

Use Case Outside of System Boundary. This rule is the implementation in SWI-Prolog of Quality Rule 5 “Use cases should be within system boundaries”. Figure 17 shows the diagram in the input file and its corresponding JSON output.

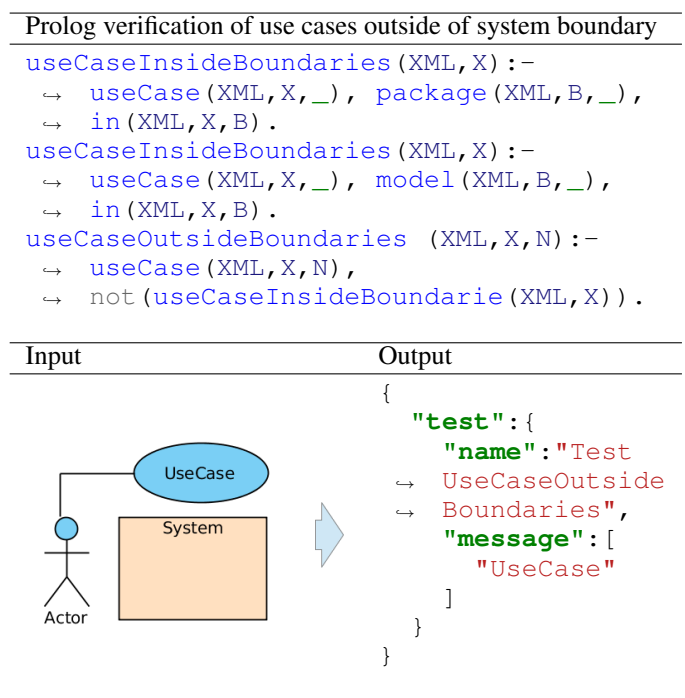


Figure 17. Implementation and sample for Rule 5 “Use cases should be within system boundaries”

Finally, as a sample of a Web site that puts together the above rules, we show the corresponding Prolog calls in Figure 18.

```
load_xml(Target, XML, []),
  findall(F, isolated(XML,_,F),
  ↪ Isolated),
  findall(F,
  ↪ useCaseOutsideBoundarie(XML,_,F),
  ↪ UseCaseOutsideBoundarie),
  findall(F, actorInsideSystem(XML,_,F),
  ↪ ActorInsideSystem),
  findall(F, unrelatedActor(XML,_,F),
  ↪ unrelatedActor),
  ↪ reply_json(json([
  ↪ test=json([name='Test Isolated',
  ↪ message=Isolated]),

  ↪ test=json([name='Test
  ↪ UseCaseOutsideBoundarie',
  ↪ message=UseCaseOutsideBoundarie]),

  ↪ test=json([name='Test
  ↪ ActorInsideSystem',
  ↪ message=ActorInsideSystem]),

  ↪ test=json([name='Test UnrelatedActor',
  ↪ message=UnrelatedActor])
  ]))
```

Figure 18. Prolog clauses for return the result of all the rules as a JSON response.

Additional expressions like extends or the representation in XMI of include are not explicitly represented in this document. The full UC2 source code is available at [13].

IV. CONCLUSION

Use cases are a common choice to specify software requirements. In spite of existing theoretical approach to use case verification, there are no known implementations that put together a theoretical background about quality of models and a derived implementation. In this paper, we propose a general quality framework and we show an initial implementation for verifying use case model, represented on demonstrations that cover the part of our system realized to the moment. This rule-based approach besides the chosen architecture, shows that a declarative approach is not only effective but also a solution presenting two main characteristics: low coupling and high cohesion which allows an easy maintenance and high scalability. Moreover, the implemented rules allow to report both errors and warnings, which can be used in a software process to improve quality of use cases due to, even being a partial approach as is, these results are measurable and repeatable ones, which makes it a valuable under the perspective of software engineering at any label, as part of a computer-aided step in a traditional development approach, or as a part into a model driven development approach.

Future work is related to limitations of the current approach. Firstly, the integration of SWI-Prolog as part of a Web service impose a limitation because it does not provide a good integration to classical Web services as Apache. Thus, we need

to review the technology behind the current solution approach. However, the most relevant challenge is to add pragmatics rules and to work with perceived quality. Classical features of a set of requirements, i.e., completeness and consistency, can not be validated without considering the context to which the system has been conceived to work, i.e., the set of stakeholders' expectations. However, this approach seems to be a way to reach it.

ACKNOWLEDGMENT

The authors would like to thank DIUFRO project DI13-0068 from the Vice-rectory of Research and Development and the Master Program of Informatics Engineering both from University of La Frontera, by supporting different aspects of this work.

REFERENCES

- [1] O. UML, "2.4. 1 superstructure specification," document formal/2011-08-06. Technical report, OMG, Tech. Rep., 2011.
- [2] B. Berenbach, "The evaluation of large, complex UML analysis and design models," in *Proceedings. 26th International Conference on Software Engineering*. Institute of Electrical & Electronics Engineers (IEEE), 2004.
- [3] G. Kösters, H.-W. Six, and M. Winter, "Coupling use cases and class models as a means for validation and verification of requirements specifications," *Requirements Engineering*, vol. 6, no. 1, pp. 3–17, Feb 2001.
- [4] Y. Kotb and T. Katayama, "A novel technique to verify the uml use case diagrams," in *IASTED Conf. on Software Engineering*, 2006, pp. 300–305.
- [5] Y. Shinkawa, "Model checking for UML use cases," in *Software Engineering Research, Management and Applications*. Springer Science Business Media, 2008, pp. 233–246.
- [6] S. Gruner, "From use cases to test cases via meta model-based reasoning," *Innovations Syst Softw Eng*, vol. 4, no. 3, pp. 223–231, Aug 2008.
- [7] S. Tena, D. Díez, P. Díaz, and I. Aedo, "Standardizing the narrative of use cases: A controlled vocabulary of web user tasks," *Information and Software Technology*, vol. 55, no. 9, pp. 1580–1589, 2013.
- [8] M. Oliveira Jr, L. Ribeiro, É. Cota, L. M. Duarte, I. Nunes, and F. Reis, "Use case analysis based on formal methods: An empirical study," in *International Workshop on Algebraic Development Techniques*. Springer, 2015, pp. 110–130.
- [9] J. Krogstie, O. I. Lindland, and G. Sindre, "Towards a deeper understanding of quality in requirements engineering," in *Advanced Information Systems Engineering*. Springer Science + Business Media, 1995, pp. 82–95.
- [10] J. Krogstie, G. Sindre, and O. I. Lindland, "20 years of quality of models," in *Seminal Contributions to Information Systems Engineering*. Springer, 2013, pp. 103–107.
- [11] M. Weiss, "Xml metadata interchange," in *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, pp. 3597–3597.
- [12] J. Wielemaker, S. Ss, and I. Ii, "Swi-prolog 7.2.3-reference manual," 2015.
- [13] F. Bautista and C. Cares. Uc2: A prolog checker for use cases. [Http://dci.ufro.cl/fileadmin/Software/UC2.zip](http://dci.ufro.cl/fileadmin/Software/UC2.zip), [retrieved: January, 2017].

An Answer Set Solution for Information Security Management

Carlos Cares, Mauricio Diéguez

Computer Science and Informatics Department,
University of La Frontera (UFRO)
Temuco, Chile

Email: {carlos.cares,mauricio.dieguez}@ceisufro.cl

Abstract—Information Security Management is focused on processes and it is currently guided by control-based standards such as ISO27002. Controls may be: management objectives, available resources or desired behaviours that contribute to information security. Under this process perspective, to reach some security level means to accomplish a specific set of controls. There are qualitative approaches and maturity models that help managers to select what controls to implement next, whilst quantitative approaches have just recently emerged under simplified formulations. The purpose of this paper is to show an answer set solution to the problem of selecting what controls to implement next, based on a given budget, security profit, and temporal dependencies between controls. The solution is illustrated by using Clingo.

Keywords—Information security; Controls selection; Answer set programming; Clingo.

I. INTRODUCTION

A standard for information security consists of a set of rules that aim to regulate a company's operation, with a special emphasis on information management and information assurance. In general, the accomplishment of some information security standard means to achieve a set of objectives, get resources or implement actions defined by the standards [1]. All these elements are known as information security controls [2] and may be grouped by dimensions.

In particular, one of the most widely-known security standards is ISO/IEC 27001:2013 [3]. This standard proposes 114 controls classified in 14 main dimensions, described in ISO/IEC 27002:2013 [4]. The degree of compliance with these controls determines the organization's security level and whether it can apply for certification.

Therefore, the map of implemented/non-implemented controls becomes a management tool to progress in information security. In Table I, some examples of controls and their corresponding dimensions from ISO27002 are shown.

The general problem of managing information security has been addressed through different approaches and different disciplines [5]–[7]. Various frameworks have been proposed for measuring the level of standard compliance [8]–[11]; however, these approaches do not suggest a plan for the implementation of controls, obtained quantitatively from the current level of compliance to some desired security level.

Investigations in this area have led to the incorporation of quantitative methods for managing security controls, some of them based on multicriteria analysis, such as, [12]–[14]. Other investigations [15]–[21], combined the System Grey Theory [22] with other quantitative techniques of analysis.

TABLE I. EXAMPLES OF INFORMATION SECURITY CONTROLS AND DIMENSIONS FROM ISO27002.

Domain: Information security policies	Policies for information security
	Review of the policies for information security
Domain: Human resource security	Terms and conditions of employment
	Information security awareness, education and training
Domain: Cryptography	Policy on the use of cryptographic controls
	Key Management
Domain: Physical and environmental security	Physical security perimeter
	Equipment maintenance
Domain: Operations security	Documented operating procedures
	Information backup
Domain: Compliance	Protection of records
	Technical compliance review

Another investigation proposed a simulation-based approach [23]–[25]. In this approach, simulated attacks are run over a model of the organization. Each attack occurs under different scenarios of implemented controls. According to the results of the simulations, the optimal set of controls is determined. The difficulty of this method is that choosing different sets of controls to simulate an attack is a human task within a combinatorial framework.

It seems clear that optimizing controls implementations is an open problem where quantitative approaches are just emerging. In [26], the conceptual framework for a quantitative optimization approach and several types of constraints are described. Mainly, we remark the temporal dependency between controls, the existence of a given budget, and the objective function focused on maximizing security by minimizing vulnerabilities.

To solve this quantitative optimization problem, we propose an answer set solution approach. Answer set programming is a research product on knowledge representation, logic programming and constraint satisfaction to cope mainly with np-hard problems [27]. Nowadays, there are some mature tools allowing the specification and solution of general models [28].

In this paper, we propose a specific solution approach for selecting controls by using the answer set tool named Clingo [29]. Clingo is a framework to solve combinatorial problems with Answer Set Programming (ASP), a simple and powerful modeling language [30].

In order to show the solution, in section II, we explain

the basis of the model by means of a small, but illustrative example. In section III, we show the results of applying the model using data from a public governmental office. The conclusion section highlights the simplicity of the proposed model but also the necessity to compare this approach to traditional models from operational research.

II. ANSWER SET MODELLING FOR SELECTING SECURITY CONTROLS

The basic principle to be applicable in an answer set solution is that there are different possible solutions, i.e., there is a space of solutions to be evaluated. An answer set program may be composed of four sections: (i) the first section expressing the basic configuration of the problem, (ii) the second section for generating the different answer sets, (iii) the third section for the derived or non basic definitions plus the terms to evaluate solutions, and (iv) the fourth section containing the problem constraints. These forms are briefly reviewed.

The basic configuration is composed of true facts, which are represented as literals (classic propositions) or predicates on specific literals (objects). The possible forms are as follows:

$$\begin{aligned}
 & l_0. \\
 & P_1(l_1). \\
 & P_2(l_2, l_3).
 \end{aligned} \tag{1}$$

To represent the different answer sets, rules are needed. These rules are known as the disjunctive form because several alternatives, represented as a set, may be generated starting from proved facts. These rules have the following form:

$$I\{A_0, A_1, \dots, A_k\} \ u \leftarrow A_1, \dots, A_n. \tag{2}$$

Under this form, the alternatives in the set stay bound by 1 and u , and these values represent the minimum and maximum number of elements in the set, provided, of course, that it is possible to derive them from the true conjunctions A_1 to A_n .

The third section should represent the definitions in the universe of discourse. In this case, the rules follow a simplified version of the previous form, this time without alternative sets, i.e., having only predicates on variables or literals in the left part. Thus, classical definitions may have the following form:

$$\begin{aligned}
 & P_0(V_0) \leftarrow A_1, A_2, \dots \\
 & P_1(V_1, V_2) \leftarrow B_1, B_2, \dots
 \end{aligned} \tag{3}$$

The fourth section specifies the constraints. These are specified similar to the previous rules, but having only the right part, as follows:

$$\leftarrow A_0, A_1, \dots, A_n. \tag{4}$$

In order to illustrate the given solution, we consider an example having ten controls, a set of temporal dependencies, and each control having its corresponding implementation cost and also a corresponding security profit. In Fig. 1, we show the basic configuration of the example. First, we have the identification of the control (C1 to C10), its cost and profit (third value). This security profit is abstract and may be considered from a single increment in the percentage of a

standard accomplishment, to the reduction of vulnerabilities belonging to key information assets. The dependencies are represented by the curly brackets. For example, the control C5 may be implemented if and only if controls C1 and C2 have been already implemented.

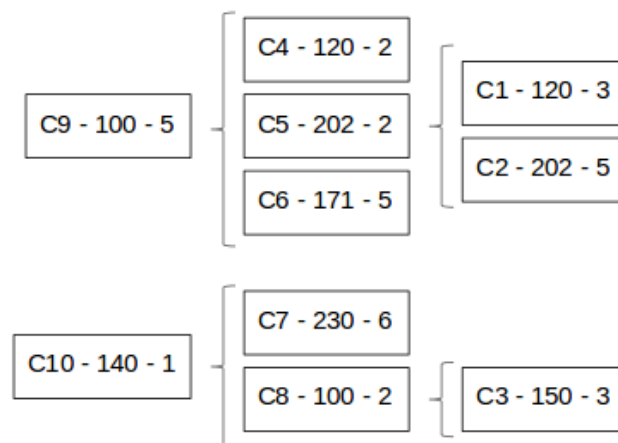


Figure 1. Candidate controls to implement, their cost, profit and dependencies.

Firstly, we notice that there are several possible answers that match the answer set conditions to be applied, under a bounded budget. For example, for a given budget of 500, we may have an implementation plan including the controls C3, C8 and C10, having a total cost of 390 (150+100+140) and a total profit of 6 (3+2+1). But also we may have an implementation plan including the controls C1, C2 and C6 having a total cost of 492 and a total profit of 11.

In order to code the solution, we have used Clingo 4.5.4 [29]. As described above, we divide the explanation in four parts.

To configure the basic initial state, we have used the predicates: *control*, for setting the variables that represent controls, *cost*, for specifying the implementation cost of each control, and *require*, for representing the dependencies between controls. In Fig. 2, we show the Clingo sentences for this configuration.

```

control (c1 ; c2 ; c3 ; c4 ; c5 ; c6 ; c7 ; c8 ; c9 ; c10 ).
cost (c1 , 200 ).
cost (c2 , 301 ).
cost (c3 , 150 ).
% ...
profit (c1 , 3 ).
profit (c2 , 5 ).
profit (c3 , 3 ).
%...
require (c9 , ( c4 ; c5 ; c6 ) ).
require (c5 , ( c1 ; c2 ) ).
require (c10 , ( c7 ; c8 ) ).
require (c8 , c3 ).
    
```

Figure 2. Partial configuration of current state.

To represent the answer sets, we have used two rules that we show in Fig. 3. The first rule represents that it is possible to plan a specific control (variable Y) provided that this control does not depend on other controls. The bounds 0 and 1 specify that this control may or may not be part of the solution. The second rule states that a control that depend on others can also be part of the solution, provided that all their required controls (*totalRequired*) have been planned (*totalIncluded*).

```
%Generate
0 { planned(Y) } 1
  :- terminal(Y).
0 { planned(X) } 1
  :- totalRequired(X,D) ,
     totalIncluded(X,W) , D=W.
```

Figure 3. Rules generation.

The necessary definitions are in Fig. 4. Here, we define the formulas for total cost, total profit, total number of required controls for each control, and total number of planned controls among the required ones. Finally, we show the definition of terminal controls that are defined as those controls that do not require the implementation of previous controls in order to plan them.

```
%Define
totalRequired(X,D):-
  control(X),
  D = #count {Z: require(X,Z),
              control(Z)}.
totalIncluded(X,D):-
  control(X),
  D = #count {Z: require(X,Z),
              planned(Z)}.
totalProfit(Y):-
  Y = #sum {D,X: planned(X),
            profit(X,D)}.
totalCost(N) :-
  N = #sum {D,X: planned(X),
            cost(X,D)}.
terminal(X):-
  control(X),
  not require(X,_).
```

Figure 4. Definitions.

Finally, the fourth section is shown in Fig. 5. This contains the definition of the available budget and the constraint that the total cost needs to always be less than the budget. Moreover, Clingo allows to add optimization expressions by using the macro clauses *maximize* or *minimize*. In this case, we have used *maximize* to search for the best result on information security profit (*totalProfit*). In Fig. 6, the result is illustrated, as displayed by Clingo. For the given example, the solution included the controls C1, C3, C4, C6 and C7, having a total cost of 871 and a total profit of 19.

Finally, we present Table II, to illustrate the combinatorial power of this problem, under the given constraints.

```
%Test
budget(900).
:- totalCost(N),
   budget(T),
   N>T.

%Optimization
#maximize
{ I: totalProfit(I) }.
```

Figure 5. Constraints and Optimization.

```
clingo version 4.5.4
Reading from sms.lp
Solving...
Answer: 1
totalProfit(0) totalCost(0)
Optimization: 595
Answer: 2
planned(c3) totalProfit(3) totalCost(150)
Optimization: 592
Answer: 3
planned(c3) planned(c7) totalProfit(9) totalCost(380)
Optimization: 586
Answer: 4
planned(c1) planned(c3) planned(c7) totalProfit(12) totalCost(580)
Optimization: 583
Answer: 5
planned(c1) planned(c2) planned(c3) planned(c7) totalProfit(17) totalCost(881)
Optimization: 578
Answer: 6
planned(c1) planned(c3) planned(c4) planned(c6) planned(c7) totalProfit(19) totalCost(871)
Optimization: 576
OPTIMUM FOUND

Models      : 6
  Optimum   : yes
Optimization: 576
Calls       : 1
Time        : 0.116s (Solving: 0.03s 1st Model: 0.00s Unsat: 0.02s)
CPU Time    : 0.120s
```

Figure 6. Candidate controls to implement, their cost, profit and dependencies.

TABLE II. NUMBER OF POSSIBLE ANSWER SETS BY BUDGET.

Budget	Answer sets
2000	147
1500	144
1000	103
900	87
700	57

In this table, we summarize, a what-if analysis for the current example showing the total of possible answer sets given by different budgets. It is possible to get them running Clingo with the option -n 0.

III. EXAMPLE

To illustrate the operation of the proposal in a real situation, we have applied the proposed model to a situation adapted from a real audit of a public organization of the Chilean State. The proposal must recommend the optimal set of controls to

be implemented, considering a limited budget, the costs of implementing the controls and the benefits obtained by the progress in complying with the controls of the standard.

In this case, the organization wished to evaluate its compliance with three standards of information security to which it subscribed. As a public entity, the organization must comply with the information security regulations established by the Government of Chile: (i) Supreme Decree 83 (DS83) [31], a security standard for public offices; and (ii) the methodological guide for information security (GUI) [32], in the framework of the Chilean government's improvement program, which describes the technical requirements associated with the diagnosis, planning and implementation of an information security system. In addition, the organization decided to evaluate its compliance with the international ISO Standard 27001, in its 2005 version.

For the purposes of the example, we will only present the dimension referring to the security of facilities, since it was the main focus of evaluation after the earthquake of the year 2010 in Chile. In this dimension, we analyzed 30 controls from the three above standards.

The benefits associated to each control were established considering the standard to which they belong to. We represent greater benefits on those controls explicitly mentioned into the Chilean norms of information security for public institutions. Therefore, those controls, that belong to more than one standard, will report a greater benefit to the organization. Considering this, a higher score was given to the controls that met the rules of the government of Chile and those that satisfied more than one norm.

The implementation costs of each control were estimated based on the operating conditions of the organization. In addition, a budget constraint was assigned to the problem.

In this way, the model delivers the set of controls that, considering costs and budget, provides the higher benefit to the organization. The implementation of the situation yielded 18 possible responses to the problem, which met the constraints of the problem. Table III, summarizes the progress of the optimization process. The table shows the number of implemented controls (second column), their corresponding cost (third column) and the previewed benefit (fourth column).

It should be noted that the 18 delivered answers do not correspond to the total universe of possible solutions. The implementation only shows those sets of controls that present a better profit than the previous answers. Therefore, the last line represents the final recommendation which includes a set of 22 controls that reports a benefit of 55 at a cost of \$ 19.950.000 (expressed in Chilean money).

The model, i.e. predicates, rules, and constraints of this case example can be downloaded from [33]

IV. CONCLUSIONS

A contemporary approach to manage information security on organizations is following process-based standards as the family of norms ISO/IEC27000. This process recommends the implementation of a set of security controls (e.g. ISO/IEC27002). From this perspective, in order to accomplish the standard, an information security assessment produces, as relevant outcomes, a set of controls already implemented and another set of controls to implement. Making a decision

TABLE III. FEASIBLE ANSWERS IN THE CHILEAN PUBLIC CASE EXAMPLE.

Answer	Number of Controls	Total Cost	Total Profit
Answer 1	8	\$ 11.360.000	21
Answer 2	9	\$ 13.360.000	24
Answer 3	10	\$ 16.360.000	27
Answer 4	11	\$ 17.160.000	30
Answer 5	12	\$ 19.960.000	34
Answer 6	14	\$ 17.800.000	37
Answer 7	15	\$ 18.600.000	38
Answer 8	15	\$ 18.500.000	41
Answer 9	16	\$ 19.300.000	42
Answer 10	17	\$ 19.700.000	43
Answer 11	16	\$ 18.760.000	45
Answer 12	17	\$ 19.160.000	46
Answer 13	18	\$ 19.960.000	47
Answer 14	16	\$ 19.720.000	49
Answer 15	21	\$ 19.850.000	52
Answer 16	22	\$ 19.750.000	53
Answer 17	21	\$ 19.250.000	54
Answer 18	22	\$ 19.950.000	55

about the next information security controls to implement is a np-hard problem. This has been demonstrated in [34] for the general problem of process-based compliance norms. Under this approach, the unique isomorphism to apply is to consider as separate tasks the implementation of security controls, which is what we have modeled in our answer set programming approach.

Although other quantitative solutions have been proposed, here we have presented a solution having three kinds of constraints: temporal dependencies between controls, a limited budget, and different information security profits given by the different controls to implement. Under the consideration of this set of different variables, as far as we know, it is the most complex quantitative solution shown in an academic setting.

We have shown an answer set programming solution simple and illustrative. Firstly, we have shown that a quantitative solution is not hard to implement, and, moreover, secondly, it can be easily extended to support additional controls (facts) and constraints, due to the modular nature of rules in answer set programming.

However, it is known that answer set solutions are based on general optimization settings. For this reason, it logically follows that specific operational research solutions may present a better performance. Therefore, in terms of future work, we will compare this answer set programming solution to classical optimization algorithms on operation research platforms, but we would like to add modelling time as a variable to observe.

ACKNOWLEDGMENT

The authors would like to thank DIUFRO project DI13-0068 from the Vice-rectory of Research and Development from University of La Frontera, by supporting different aspects of this work.

REFERENCES

- [1] T. Pereira and H. Santos, "Challenges in information security protection," *Proceedings 13th European Conference on Cyber Warfare and Security*, pp. 160–166, 2014.
- [2] H. Yau, "Information security controls," *Advances in Robotics & Automation*, vol. 3, no. 2, 2014, doi: 10.4172/2168-9695.1000e118.
- [3] ISO/IEC27001. Information security management. [Online]. Available: <http://www.iso.org/iso/home/standards/management-standards/iso27001.htm>, [retrieved: January, 2017]

- [4] ISO/IEC27002. Information technology – security techniques – code of practice for information security controls. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=54533, [retrieved: January, 2017]
- [5] M. Siponen and H. Oinas-Kukkonen, “A review of information security issues and respective research contributions,” *ACM SIGMIS Database*, vol. 38, no. 1, pp. 60–80, 2007.
- [6] K. Padayachee, “Taxonomy of compliant information security behavior,” *Computers & Security*, vol. 31, no. 5, pp. 673–680, 2012, doi:10.1016/j.cose.2012.04.004.
- [7] E. Kolkowska and G. Dhillon, “Organizational power and information security rule compliance,” *Computers & Security*, vol. 33, pp. 3–11, 2013, doi:10.1016/j.cose.2012.07.001.
- [8] T. Butler and D. McGovern, “A conceptual model and is framework for the design and adoption of environmental compliance management systems,” *Information Systems Frontiers*, vol. 14, no. 2, pp. 221–235, 2009, doi:10.1007/s10796-009-9197-5.
- [9] R. Bonazzi, L. Hussami, and Y. Pigneur, “Compliance management is becoming a major issue in is design,” *Information Systems: People, Organizations, Institutions, and Technologies*, pp. 391–398, 2009, doi:10.1007/978-3-7908-2148-2_45.
- [10] H. Susanto, M. Almunawar, and Y. Tuan, “Information security challenge and breaches: Novelty approach on measuring iso 27001 readiness level,” *International Journal of Engineering and Technology*, vol. 2, no. 1, pp. 67–75, 2012, doi:10.1016/j.im.2008.12.007.
- [11] M. Montanari, E. Chan, K. Larson, W. Yoo, and R. Campbell, “Distributed security policy conformance,” *Computers & Security*, vol. 33, pp. 28–40, 2013, doi:10.1016/j.cose.2012.11.007.
- [12] Y. Yang, H. Shieh, J. Leu, and G. Tzeng, “A vikor-based multiple criteria decision method for improving information security risk,” *International journal of information technology & decision making*, vol. 8, no. 2, pp. 267–287, 2009, doi:10.1142/s0219622009003375.
- [13] J. Lv, Y. Zhou, and Y. Wang, “A multi-criteria evaluation method of information security controls,” *Fourth International Joint Conference on Computational Sciences and Optimization*, pp. 190–194, 2011, doi:10.1109/cso.2011.43.
- [14] Y. Yang, H. Shieh, and G. Tzeng, “A vikor technique based on dematel and anp for information security risk control assessment,” *Information Sciences*, vol. 232, pp. 482–500, 2013, doi:10.1016/j.ins.2011.09.012.
- [15] L. Chen, L. Li, Y. Hu, and K. Lian, “Information security solution decision-making based on entropy weight and gray situation decision,” *Fifth International Conference on Information Assurance and Security*, vol. 2, pp. 7–10, 2009, doi:10.1109/ias.2009.9.
- [16] X. Cuihua and L. Jiajun, “An information system security evaluation model based on ahp and grap,” *International Conference on Web Information Systems and Mining*, pp. 493–496, 2009, doi:10.1109/wism.2009.105.
- [17] C. Yameng, S. Yulong, M. Jianfeng, C. Xining, and L. Yahui, “Ahp-grap based security evaluation method for mils system within cc framework,” *Seventh International Conference on Computational Intelligence and Security*, 2011, doi:10.1109/cis.2011.145.
- [18] J. Breier and L. Hudec, “New approach in information system security evaluation,” *IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1–6, 2012, doi:10.1109/estel.2012.6400145.
- [19] —, “On selecting critical security controls,” *International Conference on Availability, Reliability and Security*, pp. 582–588, 2013, doi:10.1109/ares.2013.77.
- [20] —, “On identifying proper security mechanisms,” *Information and Communication Technology*, pp. 285–294, 2013, doi:10.1007/978-3-642-36818-9_29.
- [21] J. Breier, “Security evaluation model based on the score of security mechanisms,” *Information Sciences and Technologies Bulletin of the ACM*, vol. 6, no. 1, pp. 19–27, 2014.
- [22] J. Deng, “Introduction to grey system theory,” *The Journal of grey system*, vol. 1, no. 1, pp. 1–24, 1989.
- [23] E. Kiesling, C. Strauss, and C. Stummer, “A multi-objective decision support framework for simulation-based security control selection,” *Seventh International Conference on Availability, Reliability and Security*, pp. 454–462, 2012, doi:10.1109/ares.2012.70.
- [24] E. Kiesling, A. Ekelhart, B. Grill, C. Straub, and C. Stummer, “Simulation-based optimization of it security controls: Initial experiences with meta-heuristic solution procedures,” in *14th EUME Workshop*, 2013.
- [25] E. Kiesling, C. Strauss, A. Ekelhart, B. Grill, and C. Stummer, “Simulation-based optimization of information security controls: An adversary-centric approach,” *Winter Simulations Conference (WSC)*, 2013, doi:10.1109/wsc.2013.6721583.
- [26] M. Diéguez, S. Sepúlveda, and C. Cares, “On optimizing the path to information security compliance,” *Eighth International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 182–185, 2012.
- [27] G. Brewka, T. Eiter, and M. Truszczyski, “Answer set programming at a glance,” *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.
- [28] M. Gebser and et al., “Potassco: The potsdam answer set solving collection,” *Ai Communications*, vol. 24, no. 2, pp. 107–124, 2011.
- [29] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, “Clingo= asp+ control: Preliminary report,” *arXiv preprint arXiv:1405.3694*, 2014.
- [30] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele, “A user’s guide to gringo, clasp, clingo, and iclingo,” 2008.
- [31] E. de Chile. Decreto 83: Norma técnica para los órganos de la administración del estado sobre seguridad y confidencialidad de los documentos electrónicos. [Http://www.leychile.cl/Navegar?idNorma=234598](http://www.leychile.cl/Navegar?idNorma=234598), [retrieved: January, 2017].
- [32] G. de Chile. Programa de mejoramiento de la gestión sistema de seguridad de la información: Versión 2011. [Http://www.dipres.gob.cl/594/w3-propertyvalue-16887.html](http://www.dipres.gob.cl/594/w3-propertyvalue-16887.html), [retrieved: January, 2017].
- [33] C. Cares and M. Diéguez. Oscufro: Asp configuration for optimal security controls. [Http://dci.ufro.cl/fileadmin/Software/OptimalSecurityControls-OSCUFRO.zip](http://dci.ufro.cl/fileadmin/Software/OptimalSecurityControls-OSCUFRO.zip), [retrieved: January, 2017].
- [34] S. C. Tosatto, G. Governatori, and P. Kelsen, “Business process regulatory compliance is hard,” *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 958–970, 2015.