AN APPROACH TO SUPPORT INCREMENTAL SOFTWARE CONSTRUCTION AND VERIFICATION IN COMPONENT-BASED SYSTEM DEVELOPMENT

FARANAK NEJATI

FSKTM 2019 56

AN APPROACH TO SUPPORT INCREMENTAL SOFTWARE CONSTRUCTION AND VERIFICATION IN COMPONENT-BASED SYSTEM DEVELOPMENT

By

FARANAK NEJATI

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

October 2019

DEDICATION

I would like to dedicate this thesis to my loved ones including my dependable father and my kind brother Mr. Shahram who are recently passed away. Also, to my warm-hearted mother for her endless love and encouragement. I love you and I appreciate your sacrifice, devotion, and everything that you have done to me.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

AN APPROACH TO SUPPORT INCREMENTAL SOFTWARE CONSTRUCTION AND VERIFICATION IN COMPONENT-BASED SYSTEM DEVELOPMENT

By

FARANAK NEJATI

October 2019

Chairman	: **Professor Abdul Azim Abdul Ghani, PhD**
Faculty	: **Computer Science and Information Technology**

Component-based System Development (CBSD) is a promising way of thinking or philosophy to reduce the cost and time of software system development. Moreover, CBSD is able to tame the complexity of today's software systems development while the quality is guaranteed. However, supporting correctness and building trust in CBSD are of great importance to detect errors as early as they appear.

It is commonly acknowledged that formal specification and verification methods are reliable methods that are able to offer fundamental aid to reveal errors and increase confidence in designing software systems. One of the approaches to verify systems is model checking. It is a brute-force verification method which is able to automatically and systematically analyze the state space and formal properties of a given system to discover hidden faults. However, it is limited by *State Space Explosion (SSE)*. The amount of *State Space (SS)* of a given system tends to increase dramatically and quickly exceed the memory capacity even for a small system.

Many techniques and approaches have been proposed to deal with SSE. They commonly try to circumvent SSE after the entire system is constructed. Among them, there is the incremental model checking which verifies systems before the construction is completed.

The incremental verification style has been introduced in model checking of CBSD and it is considered suitable for the implicit style of model checking. However, the explicit style is not supported before in model checking of CBSD. In this thesis, a verification approach is proposed to support the incremental verification of CBSD that is considered suitable for explicit model checking. The proposed approach is provided through three main steps in preparing a component model, constructing systems incrementally, and integrating verification into the incremental construction.

In the first step, a new component model called PUTRACOM is proposed. Components in PUTRACOM support encapsulation in the sense that computation is completely private. It has been achieved by adding Observer/Observable Unit (OOU) in the components. This feature leads to minimized coupling between the components in the systems and facilitate incremental construction. To compose components new exogenous connectors are introduced. The substantial feature of the exogenous connectors is encapsulating all controls in the system. Combining the new connectors with OOU provides a way to prevent having multiple ports and let the computation part of components be truly encapsulated.

In the second step, an approach to construct systems incrementally is proposed. The technique emphasizes on iteratively constructing and enhancing an approach version of a system by adding new increments. To achieve this, the provision of new conditions and rules is essential to maintain the system behavior during construction. A set of definitions, rules and conditions are introduced to define the system's behavior, explore the entire system to find them, and proof their preservation in each level of construction. The applicability of the approach is also elaborated by an example.

In the third step, an approach to integrate a verification process into the levels of constructions is proposed. The approach is able to avoid re-verifying lower level of constructions and lesses the state space size and verification effort via deleting the encapsulated part of each component. The approach is specified through steps, definitions and rules. Its applicability is verified by performing the rules on implementing on a real world case study.

The utilized real world case study is CoCoME which implemented in Colored Petri Net (CPN) Tools. It demonstrates how PUTRACOM provides a way to construct encapsulated components, control interactions between them by new connectors, incrementally construct and verify systems, and reduce verification efforts. The results indicate that the proposed technique can reduce the amount of state space to be checked in the component-based development. Consequently, the reduction of the state space leads to reduce the amount of execution time during the verification process. Moreover, the counterexamples can be found as early as it appears.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

PENDEKATAN UNTUK MENYOKONG PENOKOKAN PEMBINAAN DAN PENGESAHAN PERISIAN DALAM PEMBANGUNAN PERISIAN BERASASKAN KOMPONEN

Oleh

FARANAK NEJATI

Oktober 2019

Pengerusi	: **Profesor Abdul Azim Abdul Ghani, PhD**
Fakulti	: **Sains Komputer dan Teknologi Maklumat**

Pembangunan sistem berasaskan komponen (PSBK) merupakan cara pemikiran atau falsafah yang dapat memberi harapan untuk mengurangkan kos dan masa pembangunan sistem perisian. Di samping itu, PSBK berupaya mengurangkan kerumitan pembangunan sistem perisian masakini sementara itu kualiti adalah terjamin. Walau bagaimanapun, menyokong kebenaran dan meningkatkan keyakinan dalam PSBK sangat penting untuk mengesan ralat seawal mungkin pada masa ianya muncul.

Diketahui secara umum bahawa spesifikasi formal dan kaedah pengesahan adalah kaedah yang boleh dipercayai yang dapat menawarkan bantuan asas untuk mendedahkan ralat dan meningkatkan keyakinan dalam mereka bentuk sistem perisian. Salah satu pendekatan untuk mengesah sistem adalah penyemakan model. Ianya dikenali sebagai kaedah pengesahan daya kasar yang secara automatik dan sistematik dapat menganalisis ruang keadaan dan sifat formal sesuatu sistem untuk mengesan kesalahan yang tersembunyi. Walau bagaimanapun, ia terhad oleh masalah yang dipanggil Ledakan Ruang Keadaan (LRK). Amaun ruang keadaan sesuatu sistem cenderung untuk meningkat secara dramatik dan cepat melebihi kapasiti memori walaupun untuk sistem yang kecil.

Banyak teknik dan pendekatan telah dicadang untuk menangani LRK. Mereka biasanya cuba memintasi LRK selepas keseluruhan sistem dibina. Di antara mereka adalah penyemakan model secara penokokan yang mengesah sistem sebelum pembinaan dilengkapkan. Stail pengesahan penokokan telah diperkenalkan dalam penyemakan model PSBK dan ianya dianggap sesuai untuk penyemakan model stail implisit. Walau bagaimanapun, stail eksplisit tidak disokong sebelum ini dalam penyemakan model PSBK. Dalam tesis ini, satu pendekatan pengesahan di cadangkan untuk menyokong penokokan pengesahan PSBK yang dianggap sesuai untuk penyemakan model eksplisit. Pendekatan cadangan disediakan melalui tiga langkah utama iaitu menyediakan model komponen, pendekatan membina sistem secara penokokan, dan pendekatan untuk menintegrasikan verifikasi ke dalam pembinaan penokokan.

Dalam langkah pertama, satu model komponen baharu dipanggil PUTRACOM dicadangkan. Komponen dalam PUTRACOM menyokong pengkapsulan dengan erti bahawa komputasi adalah bersifat peribadi sepenuhnya. Ianya telah dicapai dengan menambah Observer/Observable unit (OOU) dalam komponen. Ciri ini mengurangkan kebergantungan di antara komponen dalam sistem dan memudahkan pembinaan penokokan. Untuk menggubah komponen, penyambung eksogen baharu diperkenalkan. Ciri utama penyambung eksogen tersebut adalah mengkapsulkan semua kawalan dalam sistem. Menggabung penyambung baharu tersebut dengan OOU menyediakan cara untuk mengelak daripada mempunyai port berbilang dan membolehkan bahagian komputasi komponen betul-betul dikapsulkan.

Dalam langkah kedua, satu pendekatan untuk membina sistem secara penokokan telah dicadangkan. Pendekatan ini menekankan pembinaan secara iteratif dan mempertingkatkan versi sistem yang tidak lengkap dengan menambah tokokan baharu. Untuk mencapai yang tersebut, penyediaan syarat dan peraturan baharu adalah penting untuk mengekal tingkah laku sistem semasa pembinaan. Satu set takrifan, peraturan, dan syarat diperkenalkan untuk mentakrif tingkah laku sistem, menjelajah seluruh sistem untuk mencari mereka, dan membukti pengekalan mereka dalam setiap peringkat pembinaan. Kebolehgunaan pendekatan ini di huraikan dengan contoh.

Dalam langkah ketiga, satu pendekatan untuk mengintegrasi proses pengesahan ke dalam peringkat pembinaan dicadangkan. Pendekatan ini dapat mengelak pengesahan berulang di peringkat bawah pembinaan dan mengurangkan saiz ruang keadaan dan usaha pengesahan dengan cara menghapus bahagian dikapsulkan bagi setiap komponen. Pendekatan ini dinyatakan melalui langkah, takrifan, dan peraturan. Kebolehgunaannya disahkan dengan melaksanakan peraturan ke atas implementasi kajian kes dunia sebenar.

Kajian kes dunia sebenar yang digunakan ialah CoCoME yang diimplemen dalam Colored Petri Net (CPN) Tools. Ia mendemonstrasi bagaimana PUTRACOM menyediakan cara untuk membina komponen dikapsulkan, kawal interaksi di antara mereka melalui penyambung baharu, membina dan mengesah sistem secara menokok, dan mengurangkan usaha pengesahan. Keputusan menunjukkan pendekatan cadangan boleh mengurangkan amaun ruang keadaan yang disemak dalam pembangunan berasaskan komponen. Akibatnya, pengurangan ruang keadaan membawa kepada pengurangan amaun masa pelaksanaan semasa proses pengesahan. Tambahan, counterexample dapat dijumpai seawal ianya muncul.

ACKNOWLEDGEMENTS

Firstly, I am greatly thankful to God for the well-being and strength that is necessitated for the accomplishment of this journey. Then, I would like to offer my most earnest and deepest thanks to my supervisor Prof. Abdul Azim Abd Ghani that all his continuous support, inspiration, immense knowledge and guidance during this Ph.D journey is invaluable. He rather than just indicating the direction actually gave me motivation, sound advises, true guides and lots of unique ideas. I am honored to take my PhD under his supervision as I would never imagine to find a better supervisor than him. Thanks for believing in me.

I am thankful to my lovely and kind co-supervisor Prof. Azmi Jaafar for constructive critics and supports that I received from him. I would like to express my very sincere gratitude to my co-supervisor Dr. Ng Keng Yap for his enthusiasm, wide knowledge and friendliness during this journey. For his patient and precious points that he provides me in our long discussion sessions. His hard questions always motivate me to think and widen the research in many aspects.

I must offer my very deep gratitude to my generous and kind parents, brothers and family to providing such unfailing support throughout my life. Last but not the least, I would like to thank Sina who has been a constant source of motivation, kindness and support.

Finally, I also thank Malaysia and Universiti Putra Malasia for it hospitality, geniality and friendly memories.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy.

The members of the Supervisory Committee were as follows:

Abdul Azim Abdul Ghani, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairperson)

Azmi Jaafar, PhD
Assosiate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Ng Keng Yap, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ROBIAH BINTI YUNUS, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

Declaration by graduate student

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any other institutions;
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature:____________________ Date:____________________

Name and Matric No: Faranak Nejati, GS40996

Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) are adhered to.

Signature: ______________________________
Name of Chairman of Supervisory Committee
Professor Abdul Azim Abdul Ghani, PhD

Signature: ______________________________
Name of Member of Supervisory Committee
Associate Professor Azmi Jaafar, PhD

Signature: ______________________________
Name of Member of Supervisory Committee
Dr. Ng Keng Yap, PhD

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SSE	State Space Explosion
SS	State Space
BIP	Binary Behavioral Constraints
BBC	Binary Behavioral Constraints
BDD	Binary Decision Diagram
OOU	Observer/Observable Unit
CU	Computational Unit
RTS	Reactive Transition Systems
CSP	Communication Sequential Processes
NDF	Nested Depth First
I/O	Input/OutputInput/Output
GA	Genetic Algorithm
EA	Evolutionary Algorithm
CTL	Computational Tree Logic
LTL	Linear Temporal Logic
COP	Constructive Orbit Problem
DFS	Depth First Search
SCC	Strongly Connected Components
OBDD	Ordering Binary Decision Diagram
LTS	Label Transition Systems
NP	Non-Deterministic Polynomial
RQ	Research Question
CoCoME	Common Component Modeling Example
CPN	Colored Petri Net
Obs	Observable
INC	Increment
ARR	Alternating Refinement Relation
TC	Trace Containment
TCC	Trace Containment Checking
SML	Systematic Meta-model Language
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

This thesis intents to propose an approach to support formal verification and model checking in Component-based System Development (CBSD). This chapter, presents the motivation behind our work, problem statement, research questions, objectives, contributions, scope and thesis structure.

1.1 Motivation

Component-based system development is a promising way of thinking or philosophy to reduce the expense and time of software system development. Moreover, CBSD is able to tame the complexity of today's software systems development meanwhile the quality is guaranteed. The most important principle in CBSD is to construct massive and complex systems by composing smaller and simpler units of software (components).

Over the last five decades, the astonishing progresses made in systems and technology have increased the scale and complexity of systems. It is fairly evident that it also leads to a growth in the probability of errors and obstacles sneaking into system development and construction even in CBSD. Thus, to support correctness and build trust in CBSD, the modeling and verification of components and their composition is fundamental.

Errors in systems are mostly serious and may lead to destructive results. A single error can lead to the crashing of entire a system, similar to an error that defected the Arian-5 rocket. There was a small application in Arian-5 which was trying to assign a number of 64-bit floating point into a variable with 16-bit space (Lions et al., 1996). This small mistake led to the catastrophic result. There are many critical systems similar to Arian-5 that, if contain errors, could result in a disastrous outcome such as nuclear power stations, Avionic software, Aircraft light control, and Trac control. Another example of such failures is the Therac-25 (Leveson et al., 1995) which was a radiation therapy machine. Six cancer patients died as a result from the machine exposing them to an overdose of radiation. The defect was caused by an overflow from a one-bit counter.

Therefore, it is critical to utilize approaches that can build trust for accuracy and the correctness of the system before it is developed. It bodes well if design errors being detected as early as development process, *”the sooner, the better”* (Baier and Katoen, 2008). Besides that, the cost of finding an error during the early stages of the design is 50 times lower than finding it during maintenance (Baier and Katoen, 2008).

Formal methods have great potential to verify and ensure system correctness as early as possible.It offers more precise and effective verification techniques based on rigorous mathematics. ”Federal Aviation Authority (FAA) and National Aeronautics and Space Administration (NASA)” reported the following result about formal methods (Fahroo et al., 2013):

> ”Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied math is a necessary part of the education of all other engineers.”

One of the well-known approaches to formal methods is model checking (Clarke et al., 1999b). This model is a brute-force verification method that is able to automatically and systematically analyse the state space and formal properties of a given system to demonstrate if its properties are completely satisfied or not. This approach has been proposed independently by Clarke et al. (1999b) and Queille and Sifakis (1982). The complete checks of model checking significantly expands the rank of confidence in the system.

Compared to other approaches like simulation, model checking is further accurate, automatic and easy to use. IBM demonstrated in one case study involving a memory-bus adapter design, that 24% of all errors were found with model checking, while only 40% of those defects would not possibly be found by simulation (Abarbanel-Vinov et al., 2001; Vakili, 2016). Additionally, when a property is violated, model checking is able to produce a counterexample for eliminating and locating the error.

Regardless of these points of interest, model checking is limited by a rather critical problem, state space explosion (SSE) (Clarke et al., 2012). SSE inherently is restricted the number of state space for a given which can check via a model checker. Especially for concurrent systems containing interleaving processes, the impact of SSE is higher. However, there are many promising advantages offered by model checking that are encouraging research communities to alleviate its drawbacks. Overcoming this obstacle emerged the major direction in model checking research and as a result, a massive collection of methods have been presented to remedy the problem (Rafe et al., 2013; Groote et al., 2015; Comert and Ovatman, 2015).

The presented methods are mostly based on scaling down the state space by abstraction (Holzmann et al., 2013), symbolic model checking (Thierry-Mieg, 2015; Sharma and Singh, 2014), and bounded model checking (Alipour and Groce, 2016; Phan et al., 2015). There are many pieces of research that deal with the SSE problem by divide-and-conquer approaches (He et al., 2016; Müller et al., 2016; Elkader et al., 2016). Moreover, some studies utilize evolutionary algorithms to explore a system's state space in a shorter time with providing an approximate solution to avoid the SSE problem (Yousefian and Rahmani, 2014; Duarte et al., 2010).

Despite the fact that proposed methods in literature have enhanced model checking in CBSD by reducing the SSE, the utilized ways for constructing systems somehow may effectively impact the verification effort and the size of state space. Nevertheless, there is scarcity of researches have focused on the way of constructing a system to reduce the SSE problem. Most of the proposed model checking methods are applied after the system construction has completed. Despite being able to provide counterexamples which is the hallmark feature of model checking after the whole system is built, revealing the obstacles as early as it appears in the system even before the whole system is constructed is more desirable.

Incremental construction and verification is one of the promising methods in model checking for producing counterexamples as early as it appears. Moreover, the idea of incremental construction may provide a way to mitigate verification efforts and its drawbacks.

Incremental construction can be presented as a multi-waterfall cycle of software development which is an alternative to traditional sequence waterfall. Mills (1976) defined the importance of the incremental development rather than sequential waterfall as the following:

> ”There are dangers, too, particularly in the conduct of these [waterfall] stages in sequence, and not in the iteration—i.e., that development is done in an open-loop, rather than a closed-loop with user feedback between iterations. The danger in the sequence [waterfall approach] is that the project moves from being grand to be grandiose, and exceeds our human intellectual capabilities for management and control.”

Incremental development provides a way to manage and control systems by dealing with a smaller part of the system in each iteration and mitigates the risks that may occur in the ”big bang” approach. It also reduces defect density during system design (Mohagheghi, 2004). It develops systems gradually and the verification process could be integrated with the construction from the initial stage of development. Obviously, the lower levels of construction in the incremental construction have a smaller amount of state space rather than the entire system's state space. In other words, model checking may need to check less state space. Furthermore, the counterexamples could be revealed as early as the system's construction is completed.

Incremental development has various aspects with different terminologies such as versioned development, time boxing, and etc., (Larman and Basili, 2003). However, this study uses the following definition proposed by Gilb (1977, 1981):

> ”A complex system will be most successful if it is implemented in small steps and if each step has a clear measure of successful achievement as well as a “retreat” possibility to a previous successful step upon failure. You have the opportunity of receiving some feedback from the real world before throwing in all resources intended for a system, and you can correct possible design errors.”

This definition emphasizes on iteratively constructing and enhancing an incomplete version of a given system by adding new increments along with verifying, learning, experiencing, and finding failures in each iteration. Some requirements and changes may gradually be applied to the up-front specification of the system to tackle design errors. However, the important question is how a system can be incrementally constructed.

CBSD can be considered as a paradigm for constructing systems incrementally (Mohagheghi, 2004; Bensalem et al., 2016; Johnson et al., 2013). In CBSD, based on a component model, systems can be constructed by preexisting reusable components. The components could put together step by step via a composition mechanism provided by the component models. Composition mechanism indicates the way of deploying components into the system. In incremental construction, increments could be considered as a set of components that are composed together by a composition mechanism iteratively until the entire system is constructed.

In this study, a new component model that would be able to support incremental construction and verification of concurrent discrete-event systems is presented.

1.2 Problem Statement

CBSD needs to be verified precisely by formal methods in order to find the obstacles and produce counterexamples as early as possible. Model checking is one of the most well-known formal methods to achieve that and build the trust of systems. Model checking, in the first step, constructs a finite representation of a given system. The representation is called *state-space* and identifies and enumerates full feasible states of the system. In the next step, it formulates a set of required properties and checks all the state space in order to find a violation of those properties.

The brute-force verification of model checking significantly expands the level of confidence in the system. However, it suffers from *state space explosion (SSE)*. SSE occurs because the amount of state space might increase dramatically by the number of its processes. As a consequence, it exceeds the memory capacity and the model checking process will be halted. This greatly limits the amount of the system's state space that can be copied by model checking. SSE problem occurs specially when there are interleaving processes running in concurrent systems.

In order to tackle SSE a body of techniques and methods have been presented in literature which can be categorized into five main approaches. These includes heuristics and probabilistic, *e.g.* (Yousefian and Rahmani, 2014; Duarte et al., 2010), Scaling down the state space, *e.g.* (Thierry-Mieg, 2015; Phan et al., 2015; Holzmann et al., 2013) , compositional verification, *e.g.* (He et al., 2016; Sun et al., 2014; Bensalem et al., 2008), memory handling, *e.g.* (Wu et al., 2015; Lengauer and Mössenböck, 2014), and bottom-up verification, *e.g.* (Patel et al., 2015; Johnson et al., 2013; Bensalem et al., 2016).

Although the aforementioned techniques have enhanced the model checking of CBSD in many aspects, very rarely do studies focus on the following two factors during verification; first, the way of system construction to reduce the SSE impacts on model checking and secondly, systems verification before the entire system is not completed.

Among the model checking approaches, bottom-up verification is a promising approach that is focused on the way of constructing systems and verify the systems even if the entire system has not been constructed yet. This approach has several benefits like verifying without constructing the entire system, integrating the verification process with system construction, and finding obstacles as early as it appears. These benefits could be considered as the promising direction of the bottom-up approach.

Two bottom-up verification methods exist in model checking which is on-the-fly model checking and incremental model checking. The on-the-fly method utilizes a depth-first search and creates a path of state space while verifying it without storing in memory. Despite substantially reducing the memory requirement, this method is very time-consuming. The amount of time will increases exponentially to regenerate already verified states.

The other bottom-up approach called incremental model checking is based on the idea of constructing systems gradually along with verification. The verification process starts from a base and iteratively checks properties of the unfinished system until the system is completed. Incremental model checking fully satisfies the promising direction of the bottom-up approach of model checking. However, there are three major factors have limited the construction and verification of systems incrementally.

Firstly, the incremental development of a system needs to be supported in the component-based models. Unfortunately, not all state-of-the-art component-based models supporting incremental construction. Moreover, the composition and interaction styles in the current state-of-the-art component-based models are mostly a port-to-port connection or method-call based. It confers complex patterns, since the number of interactions may increase dramatically due to the number of method calls, ports, and connectors.

Second, during incremental construction, the behavior of the system must remain unchanged. However, by adding a set of new increments in each level of construction, the behavior of the system may be affected. Therefore, the preservation of the system's behavior when new increments are added is vital.

Thirdly, integrating the verification process into the level of incremental construction while avoiding the re-verifying the already-verified state space from lower levels of construction is important in identifying the errors as early as the construction is completed. It is also saves the efforts in the model checking process. Moreover, it contributes in reducing the total state space needed to be verified by model checking. It is possible to establish such a method and avoid the re-verifying as presented in (Bensalem et al. (2016)). However, their method is symbolic and there is no formal method in terms of avoiding the re-verifying the already-verified explicit state space.

In Chapter 2, a wider investigation on these limitations shall be presented which provides further justification on these concerns.

The goal of this research is to present an approach to support incremental construction and verification in CBSD. To achieve this goal, a new component-based model with a new way of incremental construction and verification is provided.

1.3 Research Questions

In the following, the main challenges of the research have been identified by several research questions. These questions can be considered as a guideline for all steps towards achieving the goal of this thesis.

1. Not all current state-of-the-art component-based models support incremental construction. What are the extensions required for CBSD to support incremental construction?

2. What are the necessary rules to construct systems incrementally in CBSD?

3. How could the way of constructing system help in reducing the total state space of the system?

4. How can model checking be integrated into the system design incrementally to verify the system and detect the errors as soon as they appear?

5. How to avoid using the already-verified state space of the system to reduce the verification effort?

1.4 Research Objectives

The main objective of the research reported in this thesis is focused on proposing an approach to support incremental construction and verification in concurrent component-based systems.

The sub objectives of this thesis are as the following:

- To propose a concurrent component model to support incremental construction and verification.
- To propose an approach of construction for constructing systems incrementally along with preserving the system behavior and enforced synchronization in each level of construction.
- To propose an approach to integrate verification in each level of incremental construction, avoid reverifying the already-verified state space of the systems, reduce the total size of the state space of the system and consequently reduce the verification effort;
- To evaluate the proposed approach by demonstrating the applicability of them on a complex real-world case study.

1.5 Summary of Research Contributions

In this study, multiple techniques and models to support model checking and reduce the verification effort in CBSD is provided. A summary of contributions is listed below.

1. In chapter 4, a component model called *PUTRACOM* has been proposed. The sub contributions of this component model are as follows:
    - **Encapsulating computation by using the concept Observer / Observable Unit (OOU).**

Components support encapsulation in the sense that computations are private. All computations occur within the component itself and no other components have intervention which leads to minimizing coupling in the system. This encapsulation is achieved by equipping components with the concept Observer / Observable Unit (OOU). OOU is responsible for notifying, therefore the computation part of a component does not need to be involved in any message passing or invocation. Moreover, OOU provides a way to prevent having multiple ports which may lead to complex interaction patterns like what is presented in the current state-of-the-art component models.

- **Encapsulating control by a new exogenous connector.**
  As C. E. Hewit said ”one actor in an actor model is no actor” (Hewitt et al., 2012), one component cannot react individually in a component-based system. It must be composed in the system and interact with other components and their computing environment. To do so, this work proposes a new *exogenous connector* to compose and coordinate interactions among components. The substantial feature of our exogenous connectors is encapsulating controls in the system. Exogenous connectors set and coordinate control and data. It may lets components be more encapsulated and decoupled. The novel part of the new exogenous connectors is that they always observe the OOU of every component subscribed to them. They can observe event data from components and coordinate its entire system without evolving components in sending messages and data or controlling.

- **Multiple types of synchronization are supported.**
  It encompasses modeling systems with a mechanism for parallel composition, structuring interactions involving strong synchronization (handshaking), weak synchronization (broadcasting), sequencing, conditional synchronizing, and iteration.

- **Developing formal notation for capturing essential component and connectors behaviors.**
  It provides a description of components, their control, and composition. Components that are considered as a sequential individual process are presented by *Reactive Transition System (RTS)*

proposed by Jin (2004). The behavior of the components is controlled and restricted by connectors that are based on composition operations expressed in *Communication Sequential Processes (CSP)* by Hoare and Antony (1978).

2. In chapter 5, an approach to construct concurrent component-based systems incrementally is proposed. The sub contributions in this approach are listed below:

   - **Defining incremental construction.**
     The definitions permits to specification of increments by Reactive Transition Systems (RTSs) and capture visible and non-visible behavior of increments. Increments will be added to the system by connectors in which are specified based on Communication Sequential Processes (CSP).

   - **Rules to assist behavior preservation.**
     This includes a set of rules to trace increments and reveal hidden and observable parts of the increments. This is suitable for omitting hidden part of increments, check the system's behavior preservation, and reduce the state space generation in the verification phase.

   - **An approach to preserve the system's behavior and enforced synchronization in each iteration.**
     This includes a set of rules for adding increments in each iteration of construction that is suitable for preserving system's behavior.

3. In chapter 6, an approach to integrate verification process into construction is proposed. It has the following sub contributions.

   - **A way for generating state space of the increments.** This includes a set of steps to generate state space which serves in the verification phase.
   - **A way to verify the local and global properties of the increments.** These rules allow to verify the hidden part of the increments as local properties and then verify the observable part of the increments as the global properties. This is suitable for verifying local properties only once and avoid reverifying them in the next level of construction and consequently reduce verification efforts.

4. In chapter 7, the applicability of the approach by using a case study is illustrated.

1.6 Research Scope

This thesis focuses on proposing an approach to support verification in component-based system development. The proposed approach is an incremental construction and verification in concurrent discrete-event CBSD. Supporting verification in terms of reducing the number of state space to avoid of SSE problem during verification of explicit model checking. This research focuses on explicit model checking (explained in chapter 2) because the existing incremental verification is implicit (symbolic).

It is also focuses on concurrent systems because the SSE problem occurs when there are interleaving processes running in the systems. In sequential systems, state space does not growing up exponentially.

Due to the fact that incremental construction cannot be applied in all current component models, a component model called PUTRACOM is provided. It is a component model with specific characteristics to construct and verify systems incrementally. Enforced synchronization and deadlock-freeness are the main properties that are checked in each level of construction when new increments are added.

1.7 Thesis Structure

This thesis is organized in accordance with the standard template of thesis and dissertations at University Putra Malaysia. It is organized in a manner to provide detailed information on how the research is carried out. As the final report of this research, this thesis consists of eight chapters as presented below.

Chapter 1. presents the introduction to the background of this research. It describes the rationale for conducting this research and outlines the researcher's motivation, research objectives, problem statement, and research questions in this work. The research contributions and research scope are also explained briefly in this chapter.

Chapter 2. reviews the literature on different aspects of formal verification and model checking methods, exogenous-based component-based design, specification formalism, incremental construction, incremental verification. It presents a discussion of past works relevant to this research. In this chapter, resource materials such as journals, conference proceedings, seminar, thesis, books, and on-line resources are used as the main references.

Chapter 3. presents the research methodology designed that has been utilized in conducting this study. Additionally, it is also present the main concepts related to this thesis including temporal logics and model checking, X-MAN component model, Reactive Transition Systems (RTS) and Communication Sequential Processes (CSP).

Chapter 4. specifies formally the definition of atomic components and their characteristics in our proposed component model PUTRACOM, a formal specification of composition operators, interactions, and composite components. An example to illustrate more about the model and its applicability is also provided.

Chapter 5. specifies formally the definition of incremental construction based on PUTRACOM, the approach to trace increments, and preservation of the system's behavior. Moreover, it shows the applicability of the proposed component model by using a case study of a real system.

Chapter 6. defines the way of generating state space of increments. The definition of state space reckons on the incremental PUTRACOM presented in previous chapters. The way of verifying systems and avoiding the reverification of already-verified increments are also described in this chapter.

Chapter 7. presents the illustration of Colored Petri Net (CPN) Tools and the implementation of the PUTRACOM and other techniques in CPN Tools.

Then, a case study called CoCoME showing the applicability of the PUTRA-COM and incremental construction and verification is provided. The results of the implementation based on the number of state space in each level of construction and the time consumed is presented.

Chapter 8. presents the conclusion of the research and its limitations and indicates potential areas for future research.

REFERENCES

Abarbanel-Vinov, Y., Aizenbud-Reshef, N., Beer, I., Eisner, C., Geist, D., Heyman, T., Reuveni, I., Rippel, E., Shitsevalov, I. and Wolfsthal, Y. (2001). On the Effective Deployment of Functional Formal Verification. *Formal Methods in System Design* 19 (1): 35–44.

Alipour, M. A. and Groce, A. (2016). Bounded Model Checking and Feature Omission Diversity. *arXiv preprint arXiv:1610.08020* .

Alliance, O. (2018), Open Services Gateway Initiative (OSGi).

Alur, R., Brayton, R. K., Henzinger, T. A., Qadeer, S. and Rajamani, S. K. (1997). Partial-order Reduction in Symbolic State Space Exploration. In *International Conference on Computer Aided Verification*, 340–351. Springer.

Alur, R., Henzinger, T. A., Kupferman, O. and Vardi, M. Y. (1998). Alternating Refinement Relations. In *International Conference on Concurrency Theory*, 163–178. Springer.

Appel, A. W. (2004). *Modern compiler implementation in C*. Cambridge university press.

Armando, A., Carbone, R. and Compagna, L. (2014). SATMC: A SAT-based Model Checker for Security-critical Systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 31–45. Springer.

Armando, A., Mantovani, J. and Platania, L. (2006). Bounded model checking of software using SMT solvers instead of SAT solvers. In *International SPIN Workshop on Model Checking of Software*, 146–162. Springer.

Armando, A., Mantovani, J. and Platania, L. (2009). Bounded model checking of software using SMT solvers instead of SAT solvers. *International Journal on Software Tools for Technology Transfer* 11 (1): 69–83.

Arnold, A. (1994). *Finite Transition Systems: Semantics of Communicating Systems*. Hertfordshire, UK: Prentice Hall International Ltd.

Arnold, A. and Plaice, J. (1994). *Finite Transition Systems: Semantics of Communicating Systems*. Prentice Hall International (UK) Ltd.

Babai, L. and Luks, E. M. (1983). Canonical Labeling of Graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, 171–183. ACM.

Back, R.-J. (2005). Incremental Software Construction with Refinement Diagrams. *Engineering Theories of Software Intensive Systems, NATO Science Series II: Mathematics, Physics and Chemistry* 3–46.

Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

Barrett, C. W., Sebastiani, R., Seshia, S. A. and Tinelli, C. (2009). Satisfiability Modulo Theories. *Handbook of satisfiability* 185: 825–885.

Bastian, M., Heymann, S., Jacomy, M. et al. (2009). Gephi: an Open Source Software for Exploring and Manipulating Networks. *Third international AAAI conference on weblogs and social media* 8: 361–362.

Basu, A., Bozga, M. and Sifakis, J. (2006). Modeling heterogeneous real-time components in BIP. In *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM)*, 3–12. IEEE.

Ben-Ari, M. (2008). Principles of the Spin Model Checker. *"http://spinroot.com/spin/whatispin.html* .

Ben-Hafaiedh, I., Graf, S. and Quinton, S. (2010). Reasoning about Safety and Progress using Contracts. In *International Conference on Formal Engineering Methods*, 436–451. Springer.

Bensalem, S., Bozga, M., Legay, A., Nguyen, T.-H., Sifakis, J. and Yan, R. (2016). Component-based Verification Using Incremental Design and Invariants. *Software & Systems Modeling* 15 (2): 427–451.

Bensalem, S., Bozga, M., Nguyen, T.-H. and Sifakis, J. (2010)a. Compositional Verification for Component-based Systems and Application. *IET software* 4 (3): 181–193.

Bensalem, S., Bozga, M., Sifakis, J. and Nguyen, T.-H. (2008). Compositional Verification for Component-based Systems and Application. In *International Symposium on Automated Technology for Verification and Analysis*, 64–79. Springer.

Bensalem, S., Griesmayer, A., Legay, A., Nguyen, T.-H., Sifakis, J. and Yan, R. (2011). D-finder 2: Towards efficient correctness of incremental design. In *Nasa Formal Methods Symposium*, 453–458. Springer.

Bensalem, S., Legay, A., Nguyen, T.-H., Sifakis, J. and Yan, R. (2010)b. Incremental invariant generation for compositional design. In *Theoretical Aspects of Software Engineering (TASE), 4th IEEE International Symposium*, 157–167. IEEE.

Berezin, S., Campos, S. and Clarke, E. M. (1998), In Compositionality: The Significant Difference, In *Compositionality: The Significant Difference*, 81–102, Springer, 81–102.

Biere, A., Cimatti, A., Clarke, E. and Zhu, Y. (1999)a. Symbolic model checking without BDDs. In *International conference on tools and algorithms for the construction and analysis of systems*, 193–207. Springer.

Biere, A., Cimatti, A., Clarke, E. M., Strichman, O. and Zhu, Y. (2003). Bounded Model Checking. *Advances in computers* 58: 117–148.

Biere, A., Clarke, E., Raimi, R. and Zhu, Y. (1999)b. Verifying Safety Properties of a PowerPC- Microprocessor Using Symbolic Model Checking without BDDs. In *International Conference on Computer Aided Verification*, 60–71. Springer.

Bollig, B. (2014). On the width of Ordered Binary Decision Diagrams. In *International Conference on Combinatorial Optimization and Applications*, 444–458. Springer.

Bollig, B. and Wegener, I. (1996). Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers* 45 (9): 993–1002.

Brookes, S. D., Hoare, C. A. and Roscoe, A. W. (1984). A theory of communicating sequential processes. *Journal of the ACM (JACM)* 31 (3): 560–599.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 100 (8): 677–691.

Bryant, R. E. (1992). Symbolic Boolean Manipulation with Ordered Binary-decision Diagrams. *ACM Computing Surveys (CSUR)* 24 (3): 293–318.

Burch, J., Clarke, E. M. and Long, D. (1991). Symbolic Model Checking with Partitioned Transition Relations. *Computer Science Department* 435.

Burch, J. R., Clarke, E. M., McMillan, K. L. and Dill, D. L. (1990). Sequential Circuit Verification using Model Checking. In *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*, 46–51. IEEE.

Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L. and Hwang, L.-J. (1992). Symbolic Model Checking: 1020 States and Beyond. *Information and computation* 98 (2): 142–170.

Bures, T., Carlson, J., Crnkovic, I., Sentilles, S. and Vulgarakis, A. (2010). *ProCom—the Progress Component Model Reference Manual*. Malardalen University, Vasteras, Sweden.

Chen, T., Chilton, C., Jonsson, B. and Kwiatkowska, M. (2012). A Compositional Specification Theory for Component Behaviours. In *European Symposium on Programming*, 148–168. Springer.

Chen, Y.-F., Clarke, E. M., Farzan, A., Tsai, M.-H., Tsay, Y.-K. and Wang, B.-Y. (2010). Automated assume-guarantee reasoning through implicit learning. In *International Conference on Computer Aided Verification*, 511–526. Springer.

Chilton, C., Jonsson, B. and Kwiatkowska, M. (2012). Assume-guarantee Reasoning for Safe Component Behaviours. In *International Workshop on Formal Aspects of Component Software*, 92–109. Springer.

Chilton, C. J. (2013). *An algebraic theory of componentised interaction*. PhD thesis, University of Oxford, UK.

Christensen, S. and Mortensen, K. H. (1996). *Design/cpn ask-ctl manual*. University of Aarhus.

Cimatti, A., Clarke, E., Giunchiglia, F. and Roveri, M. (2000). NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer* 2 (4): 410–425.

Cimatti, A., Clarke, E., Giunchiglia, F. and Roveri, M. (2017). NuSMV: a new symbolic model checker. *"http://nusmv.fbk.eu"* .

Cimatti, A. and Tonetta, S. (2015). Contracts-refinement Proof System for Component-based Embedded Systems. *Science of Computer Programming* 97: 333–348.

Clarke, E., Biere, A., Raimi, R. and Zhu, Y. (2001). Bounded Model Checking using Satisfiability Solving. *Formal methods in system design* 19 (1): 7–34.

Clarke, E. M., E. E. A. and Sistla, A. P. (1986). Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8 (2): 244–263.

Clarke, E. M. (2008). *The Birth of Model Checking*, 1–26. Berlin, Heidelberg: Springer Berlin Heidelberg.

Clarke, E. M., Emerson, E. A., Jha, S. and Sistla, A. P. (1998). Symmetry Reductions in Model Checking. In *International Conference on Computer Aided Verification*, 147–158. Springer.

Clarke, E. M., Grumberg, O. and Long, D. E. (1994). Model Checking and Abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)* 16 (5): 1512–1542.

Clarke, E. M., Grumberg, O., Minea, M. and Peled, D. (1999)a. State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer* 2 (3): 279–287.

Clarke, E. M., Grumberg, O. and Peled, D. (1999)b. *Model Checking*. MIT Press.

Clarke, E. M., Klieber, W., Nováček, M. and Zuliani, P. (2011). Model Checking and the State Explosion Problem. In *LASER Summer School on Software Engineering*, 1–30. Springer.

Clarke, E. M., Klieber, W., Nováček, M. and Zuliani, P. (2012), In Tools for Practical Software Verification, In *Tools for Practical Software Verification*, 1–30, Springer, 1–30.

Cobleigh, J. M., Avrunin, G. S. and Clarke, L. A. (2006). Breaking up is Hard to do: an Investigation of Decomposition for Assume-guarantee Reasoning. In *Proceedings of the 2006 international symposium on Software testing and analysis*, 97–108. ACM.

Cobleigh, J. M., Avrunin, G. S. and Clarke, L. A. (2008). Breaking up is Hard to do: An Evaluation of Automated Assume-guarantee Reasoning. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 17 (2): 7.

Comert, F. and Ovatman, T. (2015). Attacking State Space Explosion Mroblem in Model Checking Embedded TV Software. *IEEE Transactions on Consumer Electronics* 61 (4): 572–579.

Cordeiro, L., Fischer, B. and Marques-Silva, J. (2012). SMT-based bounded model checking for embedded ANSI-C software. *IEEE Transactions on Software Engineering* 38 (4): 957–974.

Cormen, T. H. (2009). *Introduction to Algorithms*. MIT Press.

Cousot, P. (1996). Abstract Interpretation. *ACM Computing Surveys (CSUR)* 28 (2): 324–328.

Crnkovic, I., Sentilles, S., Vulgarakis, A. and Chaudron, M. R. (2011). A Classification Framework for Software Component Models. *IEEE Transactions on Software Engineering* 37 (5): 593–615.

De Alfaro, L. and Henzinger, T. A. (2001). Interface Automata. In *ACM SIGSOFT Software Engineering Notes*, 109–120. ACM.

De-Meuter, W. and Roman, G. (2011). *Coordination Models and Languages*. 1st edn., , vol. 6721. Springer-Verlag Berlin Heidelberg.

Dillinger, P. C. and Manolios, P. (2004). Bloom Filters in Probabilistic Verification. In *International Conference on Formal Methods in Computer-Aided Design*, 367–381. Springer.

Duarte, L. M., Foss, L., Wagner, F. R. and Heimfarth, T. (2010), In Distributed, parallel and biologically inspired systems, In *Distributed, parallel and biologically inspired systems*, 221–232, Springer, 221–232.

Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L. and Wong, K. (2011), About Computing Science Research Methodology.

Elkader, K. A., Grumberg, O., Păsăreanu, C. S. and Shoham, S. (2015). Automated Circular Assume-Guarantee Reasoning. In *International Symposium on Formal Methods*, 23–39. Springer.

Elkader, K. A., Grumberg, O., Păsăreanu, C. S. and Shoham, S. (2016). Automated Circular Assume-Guarantee Reasoning with N-way Decomposition and Alphabet Refinement. In *International Conference on Computer Aided Verification*, 329–351. Springer.

Fahroo, F., Wang, L. Y., Yin, G. et al. (2013). *Recent Advances in Research on Unmanned Aerial Vehicles*. , vol. 444. Springer.

Flanagan, C. and Godefroid, P. (2005). Dynamic Partial-order Reduction for Model Checking Software, 110–121. ACM.

Gamukama, E. A. and Popov, O. (2008). The Level of Scientific Methods Use in Computing Research Programs. In *31st International Convention on Information and Communication Technology*, 176–183.

Gastin, P., Moro, P. and Zeitoun, M. (2004). Minimization of Counterexamples in SPIN. In *International SPIN Workshop on Model Checking of Software*, 92–108. Springer.

Geldenhuys, J. and Valmari, A. (2004). Tarjan's Algorithm makes on-the-fly LTL Verification more Efficient. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 205–219. Springer.

Gibson-Robinson, T., Armstrong, P., Boulgakov, A. and Roscoe, A. (2016). FDR: a parallel refinement checker for CSP. *International Journal on Software Tools for Technology Transfer* 18 (2): 149–167.

Gilb, T. (1977). *Software metrics*. Winthrop Publishers.

Gilb, T. (1981). Evolutionary Development. *ACM SIGSOFT Software Engineering Notes* 6 (2): 17–17.

Gilles, K. (1974). The Semantics of a Simple Language for Parallel Programming. *In Information Processing* 74: 471–475.

Girault, C. and Valk, R. (2013). *Petri Nets for Systems Engineering: a Guide to Modeling, Verification, and Applications*. Springer Science & Business Media.

Godefroid, P. (1990). Using Partial Orders to Improve Automatic Verification Methods. In *International Conference on Computer Aided Verification*, 176–185. Springer.

Godefroid, P. and Khurshid, S. (2002). Exploring Very Large State Spaces using Genetic Algorithms. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 266–280. Springer.

Groote, J. F., Kouters, T. W. and Osaiweran, A. (2015). Specification Guidelines to Avoid the State Space Explosion Problem. *Software Testing, Verification and Reliability* 25 (1): 4–33.

Hassani, H. (2017). Research Methods in Computer Science: the Challenges and Issues. *arXiv preprint arXiv:1703.04080* .

He, F., Mao, S. and Wang, B.-Y. (2016). Learning-Based Assume-Guarantee Regression Verification. In *International Conference on Computer Aided Verification*, 310–328. Springer.

He, N., Kroening, D., Wahl, T., Lau, K.-K., Taweel, F., Tran, C., Rümmer, P. and Sharma, S. (2012). Component-based Design and Verification in X-MAN. *Proc. Embedded Real Time Software and Systems* .

Herold, S., Klus, H., Welsch, Y., Deiters, C., Rausch, A., Reussner, R., Krogmann, K., Koziolek, H., Mirandola, R., Hummel, B. et al. (2008), In The Common Component Modeling Example, In *The Common Component Modeling Example*, 16–53, Springer, 16–53.

Hewitt, C., Meijer and Szyperski (2012), The Actor Model.

Hoang-Minh, D., Le-Khanh, T. and Hung, P. N. (2013). An Assume-guarantee Model Checker for Component-based Systems. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), IEEE RIVF International Conference*, 22–26. IEEE.

Hoare, C. and Antony, R. (1978). Communicating Sequential Processes. *the Origin of Concurrent Programming* .

Holzmann, G. J. (1988). An Improved Protocol Reachability Analysis Technique. *Software: Practice and Experience* 18 (2): 137–161.

Holzmann, G. J. (1997). The model checker SPIN. *IEEE Transactions on software engineering* 23 (5): 279–295.

Holzmann, G. J. (1998). An Analysis of Bitstate Hashing. *Formal methods in system design* 13 (3): 289–307.

Holzmann, G. J., Godefroid, P. and Pirottin, D. (2013). Coverage Preserving Reduction Strategies for Reachability Analysis. In *Proc. 12th IFIP WG*, 349–363.

Howar, F., Kahsai, T., Gurfinkel, A. and Tinelli, C. (2015). Trusting Outsourced Components in Flight Critical Systems. *AIAA Infotech, Aerospace* 1868.

Iosif, R. and Sisto, R. (2000). Using Garbage Collection in Model Checking. In *International SPIN Workshop on Model Checking of Software*, 20–33. Springer.

Isazadeh, A. and Karimpour, J. (2009). A new Formalism for Mathematical Description and Verification of Component-based Systems. *The Journal of Supercomputing* 49 (3): 334–353.

Ismail, H. I., Bessa, I. V., Cordeiro, L. C., de Lima Filho, E. B. and Chaves Filho, J. E. (2015). DSVerifier: a Bounded Model Checking Tool for Digital Systems. *Model Checking Software* 126–131.

Jensen, K. (2013). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.* , vol. 1. Springer Science & Business Media.

Jensen, K. and Kristensen, L. M. (2009)a. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer Science & Business Media.

Jensen, K. and Kristensen, L. M. (2009)b. *CPN ML Programming*, 43–77. Springer.

Jin, Y. (2004). *Compositional Verification of Component-based Heterogeneous Systems*. PhD thesis, Adelaide University, Australia.

Johnson, K., Calinescu, R. and Kikuchi, S. (2013). An Incremental Verification Framework for Component-based Software Systems. In *Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering*, 33–42. ACM.

Jongmans, S.-S. T., Santini, F. and Arbab, F. (2015). Partially Distributed Coordination with Reo and Constraint Automata. *Service Oriented Computing and Applications* 9 (3-4): 311–339.

Josephs, M. B. (1992). Receptive Process Theory. *Acta Informatica* 29 (1): 17–31.

Kahn, G. and MacQueen, D. (1976). Coroutines and Networks of Parallel Processes. *Hal-Inria* .

Kanters, O., Verhoef, C. and Schut, M. (2010). QoS Analysis by Simulation in Reo. *Citeseer* .

Kautz, H. A., Selman, B. et al. (1992). Planning as Satisfiability. In *ECAI*, 359–363. Citeseer.

Kissmann, P. and Hoffmann, J. (2014). BDD Ordering Heuristics for Classical Planning. *Journal of Artificial Intelligence Research* 51: 779–804.

Kwiatkowska, M., Norman, G. and Parker, D. (2011)a. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)* (eds. G. Gopalakrishnan and S. Qadeer), 585–591. Springer.

Kwiatkowska, M., Norman, G. and Parker, D. (2011)b. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)* (eds. G. Gopalakrishnan and S. Qadeer), 585–591. Springer.

Lamborn, P. and Hansen, E. A. (2008). Layered Duplicate Detection in External-memory Model Checking. In *International SPIN Workshop on Model Checking of Software*, 160–175. Springer.

Larman, C. and Basili, V. R. (2003). Iterative and Incremental Developments: a Brief History. *Computer* 36 (6): 47–56.

Latorre-Biel, J.-I. and Jiménez-Macías, E. (2018), In Simulation Modelling Practice and Theory, In *Simulation Modelling Practice and Theory*, IntechOpen.

Lau, K.-K., Ng, K.-Y., Rana, T. and Tran, C. M. (2012). Incremental Construction of Component-based Systems. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering*, 41–50. ACM.

Lau, K.-K., Nordin, A. and Ng, K.-Y. (2011). Extracting Elements of Component-based Systems from Natural Language Requirements. In *Software Engineering and Advanced Applications (SEAA), 37th EUROMICRO Conference*, 39–46. IEEE.

Lau, K.-K., Ornaghi, M. and Wang, Z. (2006). A Software Component Model and its Preliminary Formalisation. In *Formal Methods for Components and Objects*, 1–21. Springer.

Lau, K.-K. and Rana, T. (2010). A Taxonomy of Software Composition Mechanisms. In *Software Engineering and Advanced Applications (SEAA), 36th EUROMICRO Conference*, 102–110. IEEE.

Lau, K.-K. and Tran, C. M. (2012). X-MAN: An MDE Tool for Component-based System Development. In *Software Engineering and Advanced Applications (SEAA), 38th EUROMICRO Conference*, 158–165. IEEE.

Lau, K.-K. and Tran, C. M. (2019), X-MAN Tool Set.

Lau, K.-K. and Wang, Z. (2005). A Taxonomy of Software Component Models. In *Software Engineering and Advanced Applications. 31st EUROMICRO Conference*, 88–95. IEEE.

Lengauer, P. and Mössenböck, H. (2014). The Taming of the Shrew: Increasing Performance by Automatic Parameter Tuning for Java Garbage Collectors. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, 111–122. ACM.

Lerda, F. and Visser, W. (2001). Addressing Dynamic Issues of Program Model Checking. In *International SPIN Workshop on Model Checking of Software*, 80–102. Springer.

Leveson, N. et al. (1995). Medical devices: The therac-25. *Appendix of: Safeware: System Safety and Computers* .

Lions, J.-L. et al. (1996). Flight 501 failure .

Liu, G. and Jiang, C. (2016). Petri Net Based Model Checking for the Collaborativeness of Multiple Processes Systems. In *Networking, Sensing, and Control (ICNSC), 2016 IEEE 13th International Conference on*, 1–6. IEEE.

Martens, M., Minnameier, C. and Majster-Cederbaum, M. (2006). Deciding liveness in component-based systems is np-hard. *Manuskripte/Reihe Informatik* 6.

McHaney, R. (2009). *Understanding computer simulation*. Bookboon.

McMillan, K. L. and Checking, S. M. (1993). *an Approach to the State Explosion Problem*. PhD thesis, Carnegie Mellon University, CMU-CS-92-131.

Meulen, M. G., Stappers, F. P. and Willemse, T. A. (2009). Breadth-bounded model checking. *Technische Universiteit Eindhoven* .

Might, M., Chambers, B. and Shivers, O. (2007). Model Checking via ΓCFA. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 59–73. Springer.

Mills, H. D. (1976). Software Development. *IEEE Transactions on Software Engineering* (4): 265–273.

Milner, R. (1989). *Communication and Concurrency.* , vol. 84. Prentice Hall New York.

Minnameier, C. (2006). *Deadlock-detection in component-based systems is np-hard*. Universität Mannheim/Institut für Informatik.

Mohagheghi, P. (2004). Impacts of Software Reuse and Incremental Development on the Quality of Large Systems. *Fakultet for informasjonsteknologi, matematikk og elektroteknikk* .

Müller, A., Mitsch, S., Retschitzegger, W., Schwinger, W. and Platzer, A. (2016). A Component-based Approach to Hybrid Systems Safety Verification. In *International Conference on Integrated Formal Methods*, 441–456. Springer.

Nam, W. and Alur, R. (2006). Learning-based symbolic assume-guarantee reasoning with automatic decomposition. In *International Symposium on Automated Technology for Verification and Analysis*, 170–185. Springer.

Nordin, A. (2013). *Constructing Component-based Systems Directly from Requirements using Incremental Composition*. PhD thesis, University of Manchester.

Owen, D., Menzies, T., Heimdahl, M. and Gao, J. (2003). On the advantages of approximate vs. complete verification: Bigger models, faster, less memory, usually accurate. In *Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard*, 75–81. IEEE.

Parizek, P. and Plasil, F. (2010). Assume-guarantee Verification of Software Components in Sofa 2 Framework. *IET software* 4 (3): 210–211.

Patel, R., Patel, K. and Patel, D. (2015). On-the-Fly Symmetry Reduction of Explicitly Represented Probabilistic Models. In *International Conference on Distributed Computing and Internet Technology*, 203–206. Springer.

Peled, D. (1994). Combining Partial Order Reductions with on-the-fly Model Checking. In *International Conference on Computer Aided Verification*, 377–390. Springer.

Pham, N. H., Nguyen, V. H. and KATAYAMA, T. (2010). A Minimized Assumption Generation Method for Component-based Software Verification. *IEICE Transactions on Information and Systems* 93 (8): 2172–2181.

Phan, Q.-S., Malacaria, P. and Păsăreanu, C. S. (2015). Concurrent Bounded Model Checking. *ACM SIGSOFT Software Engineering Notes* 40 (1): 1–5.

Pnueli, A. (1985). In Transition from Global to Modular Temporal Reasoning about Programs. *Logics and models of concurrent systems* 123–144.

Prasad, P., Assi, A., Harb, A. and Prasad, V. (2006). Binary Decision Diagrams: An Improved Variable Ordering using Graph Representation of Boolean Functions. *International Journal of Computer Science* 1 (1): 1–7.

Queille, J.-P. and Sifakis, J. (1982). Specification and verification of concurrent systems in CESAR. In *International Symposium on Programming*, 337–351. Springer.

Rafe, V., Rahmani, M. and Rashidi, K. (2013). A Survey on Coping with the State Space Explosion Problem in Model Checking. *International Research Journal of Applied and Basic Sciences* .

Rausch, Andreas. Reussner, R., Mirandola, R. and Plasil, F. (2008). The Common Component Modeling Example. *Lecture notes in Computer Science* 5153.

Robinson, A. J. and Voronkov, A. (2001). *Handbook of Automated Reasoning*. , vol. 1. Elsevier.

Roscoe, B. (1998). The Theory and Practice of Concurrency. *University of Oxford, Departmentof Computer Science, UK* .

Schäfer, T., Knapp, A. and Merz, S. (2001). Model Checking UML State Machines and Collaborations. *Electronic Notes in Theoretical Computer Science* 55 (3): 357–369.

Schwoon, S. and Esparza, J. (2005). A Note on on-the-fly Verification Algorithms. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 174–190. Springer.

Sebastiani, R. (2007). Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation* 3: 141–224.

Sharma, P. K. and Singh, N. K. (2014). Improved BDD Compression by Combination of Variable Ordering Techniques. In *Communications and Signal Processing (ICCSP), International Conferences*, 617–621. IEEE.

Si, Y., Sun, J., Liu, Y., Dong, J. S., Pang, J., Zhang, S. J. and Yang, X. (2014). Model checking with fairness assumptions using PAT. *Frontiers of Computer Science* 8 (1): 1–16.

Somenzi, F. (2015). CUDD: CU Decision Diagram Package Tool, Release 3.0. *"http://vlsi. colorado. edu/˜fabio/CUDD/"* .

Spijkerman, W. (2008). Marking Pocket States for Bounded On-the-fly Model Checking. *semanticscholar* .

Stern, U. and Dill, D. L. (1995). Improved Probabilistic Verification by Hash Compaction. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, 206–224. Springer.

Stern, U. and Dill, D. L. (1996). A New Scheme for Memory-efficient Probabilistic Verification. *Formal Description Techniques IX* 333–348.

Sun, C., Xi, N., Li, J., Yao, Q. and Ma, J. (2014). Verifying Secure Interface Composition for Component-Based System Designs. In *2014 21st Asia-Pacific Software Engineering Conference*, 359–366. IEEE.

Tarjan, R. (1972). Depth-first Search and Linear Graph Algorithms. *SIAM journal on computing* 1 (2): 146–160.

Tavolato, P. and Vogt, F. (2012). Integrating Formal Methods into Computer Science Curricula at a University of Applied Sciences. In *TLA+ workshop at the 18th international symposium on Formal Methods, Paris, Frankreich*.

Thierry-Mieg, Y. (2015). Symbolic Model-checking using ITS-tools. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 231–237. Springer.

Vakili, A. (2016). *Temporal Logic Model Checking as Automated Theorem Proving*. PhD thesis, University of Waterloo.

Valmari, A. (1989). Stubborn Sets for Reduced State Space Generation. In *International Conference on Application and Theory of Petri Nets*, 491–515. Springer.

Valmari, A. (1990). A Stubborn Attack on State Explosion. In *International Conference on Computer Aided Verification*, 156–165. Springer.

Van Harmelen, F., Lifschitz, V. and Porter, B. (2008). *Handbook of Knowledge Representation*. , vol. 1. Elsevier.

Wetherbee, J., Nardone, M., Rathod, C. and Kodali, R. (2018). *Beginning EJB in Java EE 8: Building Applications with Enterprise JavaBeans*. Apress.

Woźna-Szcześniak, B. (2016). SAT-based Bounded Model Checking for Weighted Deontic Interpreted Systems. *Fundamenta Informaticae* 143 (1-2): 173–205.

Wu, L., Huang, H., Su, K., Cai, S. and Zhang, X. (2015). An I/O Efficient Model Checking Algorithm for Large-Scale Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23 (5): 905–915.

Yousefian, Rosa and, V. and Rahmani, M. (2014). A Heuristic Solution for Model Checking Graph Transformation Systems. *Applied Soft Computing* 24: 169–180.