AFRL-RI-RS-TR-2018-187

A RUNTIME CHECKER WITH EVOLUTIONARY ALGORITHM DECISION MAKING

ALABAMA A&M UNIVERSITY

AUGUST 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2018-187 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
PATRICK M. HURLEY
Work Unit Manager

/ S /
JOHN D. MATYJAS
Technical Advisor, Computing
& Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUGUST 2018	FINAL TECHNICAL REPORT	MAR 2015 – JAN 2018

4. TITLE AND SUBTITLE

A RUNTIME CHECKER WITH EVOLUTIONARY ALGORITHM DECISION MAKING

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-15-2-0106

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Yujian Fu, Xudong He and Zhijiang Dong

5d. PROJECT NUMBER
T2RS

5e. TASK NUMBER
R1

5f. WORK UNIT NUMBER
KX

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Alabama A&M University
4900 Meridian St N
Huntsville, AL 35810-1015

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-187

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
One of the major goals of the DoD information innovation is to build trusted, adaptive, attack-resilient cyber physical systems. To meet this goal, this project aims to deliver a trustworthy computational tool set to enable the development of inherently safe and secure, resilient, and reliable software. To achieve resilience, the project PIs realized an optimization approach of their runtime verification approach to handle dynamic adaptation. To ensure the computing capability is trustworthy, this project studied the overall software intensive life cycle with a complete view through a sequence of systematic steps beginning from requirements, specification, modeling, design, to implementation, testing, verification and validation. To validate the formal methods based tools, a sophisticated, hierarchical proof-of-concept platform was developed and several real world implementation and tests were conducted. With the real time scheduling technique and interaction with the physical components, a sequence of profound findings were discovered and published. The results and findings have disclosed the fact that a model-based integration approach for cyber physical system design and development plays a significant role in its overall trustworthiness and resilience.

15. SUBJECT TERMS

Trusted and Resilient System, Automated Repair, Runtime Verification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			PATRICK M. HURLEY
U	U	U	UU	55	19b. TELEPHONE NUMBER (*Include area code*)

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

PREFACE

The purpose of this report is to enable the air force research laboratory to establish a systematic approach toward the research of trustworthy computing in software intensive and resilient systems. The documentation serves as an explicit statement to fulfil the requirement of the project scheduling and measurements. This project was completed by four PIs' (Drs. Yujian Fu, Xudong He, Zhijiang Dong, Heng Yin) collaborations from four institutions: Alabama A&M University, Florida International University, Middle Tennessee State University, and Syracuse University.

ACKNOWLEDGEMENTS

This material is based on research sponsored by the Air Force Research Laboratory under the agreement number FA8750-15-2-0106. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright notation thereon.

Regarding the Air Ground Interoperation System, the authors would like to thank all those who offered assistance to this project both at AAMU and at West Point. Special thanks are due to the following: Dr. Jinlei Zheng at University of Alabama in Huntsville, Mr. Hilberto Ayala at EuroSoft Inc, Dr. Joel Fu, Dr. Chance Glenn, Dr. Mebougna Drabo, Mr. Winchien Choosilp at AAMU. Thanks for the help of *Drone Attorney* for the review of the 107.35 waiver request documentation to FAA.

1. PROJECT SUMMARY

One of the major goals of the DOD information innovation is to build trusted, adaptive, attack-resilient cyber physical systems. To meet this goal, this project aims to deliver a trustworthy computational toolset to enable the development of inherently safe and secure, resilient, and reliable software. Cyber-physical systems (CPS), as defined by Janos Sztipanovits, are systems that combine a physical system with an embedded information processing system such that the resulting system has novel capabilities that could not be achieved by either the physical or the computational entity alone. To achieve resilience, the project PIs realized an optimization approach of their runtime verification approach to handle dynamic adaptation. To ensure the computing capability is trustworthy, this project studied the overall software intensive life cycle with a complete view through a sequence of systematic steps beginning from requirements, specification, modeling, design, to implementation, testing, verification and validation. To achieve resilience, the project PIs have investigated the existing development approaches in cyber physical systems [1] from several key aspects regarding the development life cycle. To validate the formal methods based tools, a sophisticated, hierarchical proof-of-concept platform was developed and several real world implementation and tests were conducted. With the real time scheduling technique and interaction with the physical components, a sequence of profound findings were discovered and published. The results and findings have disclosed the fact that a model-based integration approach for cyber physical system design and development plays a significant role in its overall trustworthiness and resilience.

In the past decades, model-driven development of embedded software intensive systems has gained dramatic attention from researchers and software engineers as a promising approach and tool. Formal methods played a significant role to ensure trustworthiness with high confidence in the past years. Model checking [2], theorem proving [3], and static analysis [4] are the typical formal approaches to ensure the software intensive and resilient systems are trustworthy and high assurance. However, the engineering of non-software artefacts is often used for model-based computational manifestation. The model-based techniques started to find a way to integrate into the physical artefacts of the systems in recent years. During this project, we applied the model-based formal development approach to several research problems, including controller generation, and simulation engine on different hardware platforms.

Complementary to static formal verification techniques, dynamic and scalable validation runtime verification [5] techniques combine the advantages of formal methods and software testing – trusted aspects are formally specified and functionality checking is realized through monitoring the check points during program execution. In addition, we used aspect-oriented programming for program instrumentation. This monitoring approach was applied to heterogeneous robotics system to ensure the trustworthiness of the system. Several findings were captured in publications [6 – 17].

Functional and non-functional properties and uncertainty in an unknown environment are really challenging issues for researchers in cyber physical systems, especially when handling the hardware platforms. As a generic framework, evolutionary algorithms [18] perform better in large populations and/or possibilities. In this project, one of the more important tasks was to

explore all candidate possibilities to find the best adaptation solution in a timely manner, so that cyber military systems will continue to accomplish their missions without compromising security. In this study, the genetic algorithms were used to generate the robotic controller and were demonstrated as a particularly effective solution with a large searching space. In order to develop a novel approach to ensure high confidence, trusted and attack-resilient software intensive systems were developed by synthesizing the runtime verification with an evolutionary algorithm, and creating an improved version of runtime monitor with traditional evolutionary algorithms. Meaningful simulation results have been obtained, and several publications are being prepared [59 – 62].

By the end of this project performance, we believe that our model-based development approach to study the resilience and trustworthiness of a cyber-physical system (CPS) provides an effective method to ensure the computational systems are trustworthy. A system behavior model that is extracted from the CPS Java program has been model checked and runtime monitored. Based on the system behavior model and the security requirements, a logical formula has been defined for each security property or security policy. With the help of the model based runtime verification tool, Java monitoring code was constructed from the logical formula. With the integration into the genetic algorithm, a robotic controller was able to handle resilience properly. Our approach of using model-based integration runtime monitoring can be extended to more unpredictable cyberattack scenarios. Future work on the recovery from hostile attacks so that the cyber physical system can continue to fulfil its predefined missions would be necessary.

2. INTRODUCTION

Cyber physical systems are dynamic and interact with external continuous and discrete events. Cyber physical systems include a lot of highly complex computing such as dynamic analysis and verification of continuous dynamic properties. Therefore, modeling [6,13], analysis [6,13], verification of spatial properties, non-functional properties, failure and recovery of cyber physical systems to build a trustworthy computing platform have been a major challenge in the past decades. It is not enough for traditional methods to solve the computing problems, address the system behaviors, or even build physical systems. To meet the overall research goal of the DOD information innovation, this project aims to deliver an integrated trustworthy computational tool set by using the formal methods and computational model to achieve an inherently safe and secure, resilient, and reliable cyber physical software intensive systems.

A number of research methodologies and tools have been proposed and developed for the design of cyber physical systems. Even though the spectrum of potential applications of cyber physical systems is quite broad, two main areas have been targeted: domain of computational capability and problem domain of control capability for a system in general. Most methodologies are domain specific or platform dependent as the understanding of the systems is not yet complete and the complexity of dynamic behavior is difficult to capture. From this stand point, an acceptable cyber physical system design methodology must synthesize the different views of the systems and use a number of different methods consisting of systematic steps of the development life cycle to ensure all the design phases are covered. Several research aspects need to be considered in the development of a trustworthy computing platform.

First of all, as computing and networking systems in cyber physical systems interact with the physical world, predictability in the timing of these systems is critical for them to behave as expected and/or as programmed. Due to the scale, structure, and behavioral complexities of these systems, it is quite a challenge to develop an extensible and adaptable software platform to operate in a distributed and concurrent environment. The Common Object Request Broker Architecture (CORBA) of the Object Management Group (OMG) is a well-known industry standard specification for software platforms called middleware, which have been used mainly in distributed and concurrent computing environments.

In cyber physical systems, temporal predictability and communication of objects and units are another critical issues for performance and reliability. Most current communication networks are built on either peer-to-peer communication architecture or broadcast architecture with various types of devices. The speed and formulations of these communication networks need to be considered for reliability and optimal access point packet in scheduling and performance.

Secondly, as cyber physical systems include various types of engineered systems that require tight integration of computing, communication and control technologies, therefore, design and development of hardware, software, and networking contribute significantly to the overall development of life cycle and cost effectiveness. In the near future, the design and analysis of trustworthy cyber physical systems will be more challenging due to the increasing complexity, safety-critical environment, and heterogeneity of the systems. It is expected to be even more difficult to design, develop, and test and verify such computing systems due to the significant increases in overall development costs. One effort is model-based design and development [19 - 21]. The basic idea behind this approach has already been used successfully

in several research tools, such as Simulink and LabVIEW, and has proven to be useful for the development of today's real-time embedded systems. However, to handle the increased complexity and heterogeneity of cyber physical systems, it is necessary to further advance technologies for model-based design and development beyond the existing tools. Building an accurate and well-defined model that is composable and verifiable is a significant challenge. As discussed in [22], today's computing and networking abstractions and modeling approaches are not appropriate for cyber physical system modeling since they do not properly capture the dynamic behaviors of timing and concurrency of a physical system. Meanwhile, some promising research directions for better behavior capturing and for computing abstractions of the networking systems have been identified.

Challenges to cyber physical system's modeling and analysis of recent work have been presented in [23-27] using several illustrative examples. Giotto [23], a programming language for real time cyber physical systems, and Ptolemy II [24], a modeling and simulation environment for heterogeneous systems, are also notable works. Various works on the model-based design and development are noted as model-driven engineering (MDE) [25] and model-integration computing (MIC) [26] and have demonstrated meaningful results in enterprise computing environments.

Due to the complexity of cyber physical systems, the third aspect will be runtime monitoring of the real systems and simulated models. First, monitoring cyber physical systems online can take place through online measurements. The role of monitoring is to alarm the system in real time against the expected properties of correct behaviors. A primitive form of this monitoring is mostly developed for the safety-critical and mission-critical components and systems, such as airport railway scheduler, controller of traffic, conditions of patients, and SCADA system. The second one is monitoring the model simulated behaviors in specified conditions. Model-based monitoring is to observe the results when the model is executed in the specified environment model as per the predefined initial condition. The design of such model-based monitoring exists in the virtual objects that are accompanied by intellectual simulation and verification tool support. The monitoring of model-based simulation can trace the process of each design phase in a controllable manner. Many techniques and considerations are discussed along with the monitoring of adaptation to the real environment.

In our project, to realize the development of a tool for a trustworthy software intensive system, a rigorous specification of the cyber physical system was done using formal methods (Petri Nets). The expected behaviors to be monitored are defined in Linear Temporal Logic (LTL). LTL provides a compact language for specifying a task execution sequence and the relation between the executions. The logic engine generates the monitor and instruments for the implementation. An evolutionary algorithm is used to generate the logic controller to handle the unexpected situations. This tool was implemented on the developed proof-of-concepts platform – a mobile multiple collaborative robotics in an air-ground interoperation.

This project report is organized as follows: Project overview, methodology, assumption and procedure is presented in Section 3. Following section 3.1 project overview, the methodology will be covered in section 3.2, CPS formal specification and modeling will be introduced in section 3.3. Runtime monitoring and resilience handling methodology will be presented in section 3.4. Thereafter, Results and Discussion will be presented in Section 4, with details of proof-of-concept platform development. Conclusion and recommendations will be discussed in Section 5.

3. Project Methods, Assumptions, and Procedures

As this project's goal is to build upon the model-based integration of runtime verification of resilient cyber physical systems, we discuss the relation between the model and physical systems with regard to our model. Figure 1 shows the mapping schema between the model domain and the real physical domain of the cyber physical system (CPS).

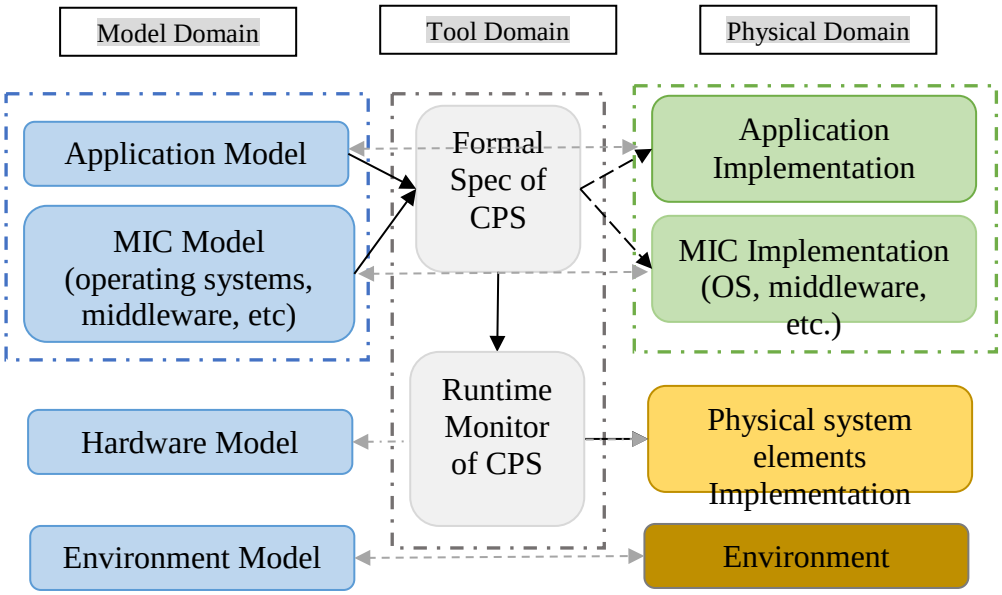

Figure 1. The Mapping Schema of the Real World Physical System and the Model

Note that the "model domain" includes the "application software" that is implemented in a computational platform, which is layered upon and interacts with a physical system (the 'hardware' of the CPS), which then interacts with the physical environment. What we envisioned and achieved here is a complete model-based integrated approach across all levels of the software intensive system development hierarchy. In short, one needs models for the environment (that is outside of the CPS), for the physical system (that is part of the CPS but is not computational), for the computational platform (that includes all hardware and software elements across different CPSs), and for the application (i.e., the software that implements all the functions of the CPSs).

3.1 Project Overview

In order to meet the project goal and present a seamless geared strategy of formal verification and computational science, which can be applied to the synthesis of genetic algorithm and runtime verification, we imagined a cyber physical system with air-ground integration which has an innate ability to correlate operational information between air layer and ground devices and to reduce the possibility of each layer being infected with possible attacks or vulnerability. Our strategy for this synthesis had to provide accurate indicators with

low falsification and failure rates, and high rigidness. To do so, we identified the following scenarios [9,18]:

A trustworthy cyber physical system must be able to adapt to the change in environment. This change will be monitored by runtime verification, which will be able to return true/false value and unknown value depending on the sensory data. The runtime monitoring can be done on both modeling domain and physical domain.

Mapping to Figure 1, this project item focuses on application modeling instead of environment modeling. In the tool domain, it will be a formal specification tool on the SAMPIPE+ [14]. A discussion on the formal specification tools is presented in section 3.2, and section 3.3 discusses runtime monitoring

In the physical domain, runtime monitor was mainly applied to the java program which was coded on two platforms: “car parking system” [15,18] and “NAO soccer game”.

As the applied tool in this approach, a typical runtime verification tool – a monitor oriented programming platform (MOP) [63]—was extended with CPS monitoring using AspectJ to instrument the temporal logic formula monitors.

Mapping to Figure 1, this project item focuses on the hardware and environment analysis using monitoring based programming framework [63]. This runtime monitor was developed based on a previous work [63] and applied to several collaborative ground robotics systems [15, 18, 19]. In our current multiple robotics platform, the runtime monitor was used in several formats: individual tasks on single robots (SS), multiple tasks on single robots (MS), and multiple tasks on multiple robots (MM). A detailed description of the runtime monitor has been given in section 3.3, and the applied robotics architecture and development is discussed in section 4.

With regard to the physical domain, most current programs are Java and AspectJ program for monitoring and robotics platform. We have selected some of the programs for this report. Our current car parking system was completed on the LEGO EV3 with special physical design and elegant environment design, which is discussed in section 4.

To find the synthesizing points in the genetic algorithm, two streams of verification were considered in the project: (1) correctness of the algorithm; and (2) the monitoring of populations and generation process. As the first task was to analyze the algorithm, which does not provide the monitoring gear, we focused on the second task in this project. We thus developed a three-phase optimization, which will be applicable to individual robotics, collaborative tasks, and communication. A discussion on the implemented optimization is presented in section 4.

In the physical domain, we envisioned an air-ground system that is able to realize collaborative work through task coordination [18]. In this domain, the heterogeneous ground robotics and unmanned aerial system had to be developed.

With regard to the physical domain, our current genetic algorithm was implemented in the collision avoidance and path generation in the car parking system [15, 18, 19]. A detailed description of the architectural overview and development process is provided in section 5.

Our strategy was to develop a framework for trustworthiness of CPSs based on an integration of runtime monitor and optimization approach, which is validated through an air-

ground system (A-G system). The A-G system safety assurance is throughout the development process on development process phases ([8], [11]).

3.2 Methods, Assumptions and Procedures

Based on the above description, figure 2 illustrates the project architecture and performance organization. It describes the architectural overview of the project in the tool domain with regard to the model-based development and physical realization domain. In this figure, we have used the abbreviation CPS to denote cyber physical systems. As one of the core components in our approach, the system behavior model describes the functional and dynamic aspects of the system in both data flow and control flow. The behavior model can either be derived from system specification and requirement, or extracted from system implementation (program) using reverse engineering techniques [28].

Many approaches have been proposed to extract semantic models from system implementation (source code or bytecode) [29, 30]. Among them, the work in [31] was chosen as the basis to construct a PrT net (a class of high level Petri nets) behavior model, which can be analyzed using existing analysis methods in PrT nets, and which can facilitate scheduling and resource allocation. Low level Petri nets are adequate for design modeling and analyzing control flows, but not suitable for the development of CPSs that are data dependent. PrT nets have an expressive, powerful modeling language, with a strong and rigorous analysis capability. They are widely used in the modeling and verification of large scale mission critical systems. Our runtime verification was able to generate the Java program from the PrT net specification while considering many characteristics of net inscription, including sorts, quantifiers in first order logic, and linear temporal logic (LTL) [32].

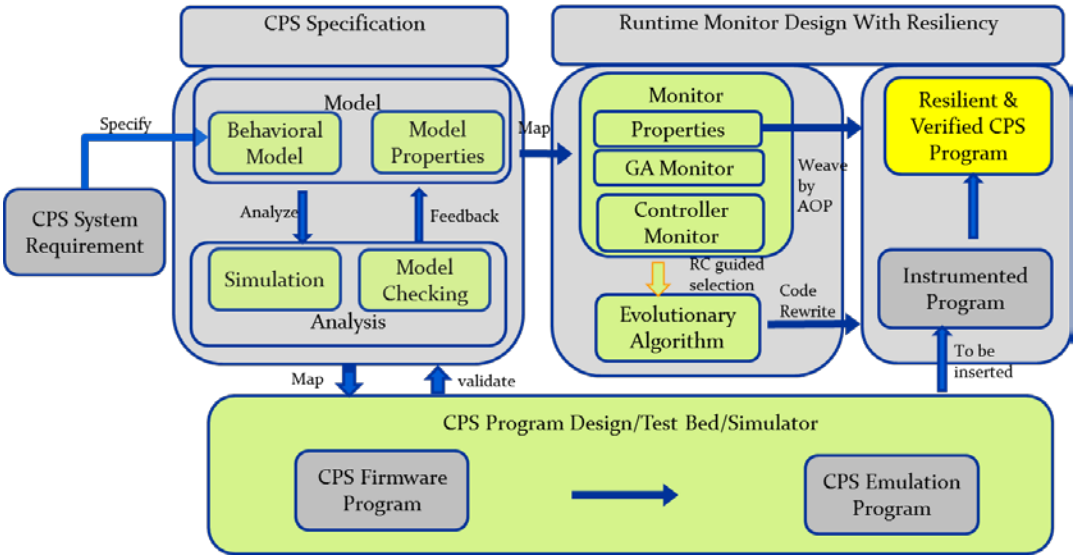

Figure 2. Project Overview Framework

The model was developed from the obtained CPS user requirement, while keeping in consideration both structural description and behavioral modeling. To analyze the behavior dynamics, two major steps were taken: model checking and simulation of the developed model.

The tool was developed by the PI team at Florida International University, which was led by Dr. Xudong He. A detailed research report has been presented in section 3.3.

To ensure trustworthy computing, a runtime monitor was developed and used to observe interesting events generated by running the CPS (Java) program. Based on the sequence of received events, the monitor can tell if the current execution trace violates the corresponding security property from which the monitor was constructed. The monitor treats the violation of the safety properties which the system experienced during execution and which can be known or unknown attacks. The result of the disclosed unexpected events are sent to the controller, which generates the proper upgraded behaviors to make the system adapt to the new environment (under attack or in an unexpected situation). This adaptation is currently implemented in the simulation of the real world implementation. The solution may require a change in the control flow or bytecode rewriting/instrumentation in the running CPS Java program. If the solution is implemented correctly, the CPS Java program will be able to continue its execution to accomplish the predefined missions without suffering any serious damage from the hostile attack. The tool was developed by the PI team led by Dr. Zhijiang Dong at MTSU. A detailed research report has been presented in subsection 3.4.

We chose the typical Mobile CPS, which is widely used in the academic settings and industry, to validate our approach – centralized controlled mobile cyber battle system. The system will be established by pursuing the air-ground collaborative cyber battle that includes both the Android controlled robotics (ground robotics and UAVs), which can be tracked by web-based system. The platform is composed of three typical architectural components: (1) a centralized controlled space to register and observe the existing mobile robotics remotely, (2) a set of Android tablets to control the robots, and (3) a set of heterogeneous robotics systems. The runtime verification (RV) on the mobile CPS proposed in this work brings a lot of significance to the RV community due to the following challenging issues: First, Android apps are run on the Dalvik virtual machine that is different from the current RV checker's platform (on JVM). Second, the robotics systems are mostly firmware platforms; the compilation and running can be done on the device. For the first issue, a few existing studies [33, 34, 35] have adopted the RV techniques on the Android systems without any modifications. In addition, many software testing tools [36] for Android systems include the on-device testing. Since AspectJ is compiled using Java bytecode, the weaving can be done on Dalvik. For the second issue, the memory restricted system pre-compilation is required. In addition, static analysis for off-the-shelf code can be applied before execution on device. This platform was developed by the PI team which was led by Dr. Yujian Fu at AAMU. The project has been presented in subsection 3.5 and the detailed research report is discussed in section 4.

3.3 Project Methods – CPS Formal Modeling and Analysis

We have developed a variety of modeling and analysis methods for CPS development in this project. These methods have been applied to several robotic systems and some well-known hybrid benchmark systems and are supported by our tool environments. In the following sections, we briefly discuss our modeling and analysis methods, their applications, and the supporting tool environment.

3.3.1 A Formal CPS Development Methodology

We have developed a framework for trustworthiness CPSs based on a model-driven approach with quality assurance throughout the development process ([8], [11]). Our

methodology consists of the following major components: modeling, model analysis, model implementation, and implementation analysis.

Modeling Methods

High level Petri nets are chosen as our modeling method due to their suitability for concurrent and distributed systems as well as their capability to deal with real-time systems. An agent oriented modeling approach is used to capture CPSs at a high abstraction level where meaningful computational components and physical processes with independent behaviors are modeled as individual agent nets. Each type of physical devices (sensors and actuators) or computation processes is modeled with an agent net that has its own meaningful and demonstrates independent reactive and/or proactive agent behavior interacting with external environments, while concrete physical devices or computation processes are with structured tokens containing unique identification in the 1st field. Specifically, we provide the following general design heuristics in building an agent net:

- Attributes of a physical device or states of a computation process are defined by places with appropriate types. Discrete values are defined using string or integer types and continuous values are defined using real type. Structured types (Cartesian product of basic types) are used to define complex attributes. Powerset is used to define multiple physical devices and computation processes;
- Actions or state transitions are modeled with transitions containing first order logic formulas defining the preconditions and post-conditions;
- The interaction between a physical device and an external environment can be modeled with a transition containing a random function emulating the possible values from the environment (open system) or with a transition picking up a possible value from an additional place denoting the external environment (closed system);
- Virtual time is modeled with tokens having an additional field denoting time stamps and a special place modeling a logical clock.

The overall agent system is obtained by integrating individual agent nets to form a system net that shows the interaction, communication, and cooperation among different agents. Synchronized activities are modeled through new joint transitions with modified constraints, and asynchronous activities are modeled through connecting a place in one agent net to a transition in another agent net. An aspect oriented approach is used to build a complex model incrementally through weaving individual Petri nets representing agents capturing physical devices and computation processes. This aspect oriented approach further supports system adaptation and evolution, and facilitates compositional analysis. System weaving is supported by our tool SAMTools (Software Architecture Modeling Tool) with extended new functionality, where a pair of nets created in PIPE+ – one as the base and the other as an aspect are combined to form a new net through connecting a pair of place and transition or merging a pair of places or transitions. Modified transition constraints and additional arcs with proper labels are added to the resulting net in PIPE+. A complex net can be built from existing agent nets successively by weaving two nets at a time and only needs a few clicks.

Model Analysis Methods

A CPS system is often a hybrid system consisting of both continuous hardware devices and discrete computation processes. In most cases, the only available technique for continuous components is simulation. However, formal verification techniques based on symbolic

reachability analysis is available for sub classes of hybrid systems such as those can be modeled using linear hybrid automata where the state transition rates are constants with restricted checking and updating actions. High level Petri nets are executable and thus support simulation of hybrid system models. Furthermore our tool PIPE+ supports reachability analysis and model checking using SPIN in addition to simulation.

Simulation is carried by firing enabled transitions. Two modes of simulation can be done in PIPE+: single steps and multiple steps. Simulation can be used to test whether a model satisfies the CPS requirements by examining the tokens in places of interest or transition firing history. In addition to simulation, we have also implemented a simple reachability analysis that checks whether a place hold a particular value during an execution, which either confirms our intention (witness) or finds a potential error (counter example). When a reachability is confirmed, the simulation time and the transition firing sequence are recorded. However simple reachability analysis may not be conclusive when a search does not end. Repeated checking can be used to eliminate possible false negative in systems with finite execution sequences.

Model checking performs exhaustive search on finite state systems and thus is not directly applicable to continuous systems. However we may be able to model check the bounds (called barrier certificates) of some continuous state variables. During the translation, each place is translated into a channel with the place's type. The translated Promela code typically contains the following parts:

(1) macro definitions for names to be used in the program introduced using #define;

(2) type definitions – enumerative constants using *mtype* and structured type using *typedef*;

(3) using channels *chan* to define places in the model. Messages (tokens) in a channel can be searched and removed using operator ? and added using operator !;

(4) two inline functions are defined for each transition – one for capturing the post-condition (*inline* fire_transition) and the other for checking the precondition and invoking transition firing (*inline* transition);

(5) a single process defined using *proctype* for non-deterministically selecting enabled transitions repeatedly;

(6) system initialization using init.

The translated Promela model after adding linear time temporal logic specifying properties is model checked using SPIN. Safety and liveness properties are expressed in the general form []*placename*(*x*) and <> *placename*(*x*) respectively, where [] and <> are the temporal operators always and sometimes in SPIN and *x* can be a variable or a constant (a specific token). More complex formulas are defined using logical connectives.

Implementation Methods

Design models help us to better understand system features including functionality, structure, and behavior as well as to detect and prevent early system development errors. To leverage the design models to increase productivity and improve code quality, we present a model driven approach to realize our high level Petri net models, which provides a systematic way of writing Java programs and establishes the traceability between the models and resulting programs. Our model driven approach consists of the general code structure and domain specific refinement. The general code structure can be systematically generated from the agent models and the overall system model. However manual domain specific refinement is needed

in identifying and defining additional features of the system, especially with regard to the physical devices.

The following translation rules are used to generate the general code structure from high level Petri net models:

(1) A class is generated for each agent net, where methods are the transitions. The body of each method is empty and requires manual refinements. The constraint of the transition is attached as comments for ensuring the correct implementation;

(2) Each user defined data type associated with a place is mapped to a class. However it is not easy to determine whether each place is mapped to an attribute or multiple places are mapped to an attribute. This decision is done during the manual refinement;

(3) A thread is created based on the class from an agent net in (1) to capture the independent active behavior of agents modeled by the agent net;

(4) A Java project is created to include the above code files.

The above code generation rules are implemented in PIPE+. A Java project is automatically generated from net by selecting JavaProject under Export button in File pulldown menu.

Implementation Analysis Methods

Implementation analysis integrates bounded symbolic model checking and runtime-time verification to ensure model level properties and additional properties are not violated in the implementation. The model level analysis and implementation level analysis are complementary. At model level, both safety and liveness properties are checked to detect potential errors in the requirements with environmental assumptions such as the hardware devices working properly. At the implementation level, safety properties are checked through bounded symbolic model checking and monitoring the actual behavior of hardware devices. Detailed runtime monitoring method is discussed in a section later.

Summary

We have demonstrated our framework using a car parking system. This result received the 3rd place in best paper award at the 29th International Conference of Software Engineering and Knowledge Engineering held in Pittsburgh, July 2017. An extended version has been published by the International Journal of Software Engineering and Knowledge Engineering.

Recently, we developed a systematic approach that enhances the above framework with the following new results: (a) a new behavior-oriented approach to incrementally model and analyze CPSs, (b) a pattern-based translation method for generating behavior programs from behavior nets, and (c) a set of behavior based runtime monitoring property patterns. Our systematic approach has been demonstrated through a multi robotic car parking system [7].

3.3.2. Modeling and Analysis Techniques for Hybrid Systems

We have developed methods for modeling and analyzing hybrid systems using high-level Petri nets as well as hybrid high-level Petri nets that contain structures to explicitly represent continuous places and transitions.

High-level Petri nets were developed for modeling and analysing discrete systems. We have developed a method of using high-level Petri nets to model various CPS behavioral features, including discrete, continuous, synchronous, hybrid, and real-time.

Modeling Discrete States and Transitions

In high-level Petri nets, system states are modeled with typed places and the tokens residing in them. One or more places of the same type are used to capture individual state variables. A system state is defined by the values of all system variables, i.e. the tokens in all places. A system state transition involves the firing of some concurrently enabled local transitions. Each transition firing changes some local distributed state according to the constraint defined by a first order logic formula associated with the transition. It is thus straightforward to model discrete states and transitions using PrT nets.

Modeling Continuous States and Transitions

High-level Petri nets can be used to model a variety of continuous system encountered in real world applications. We proposed a continuous updating pattern, where the place *x* of real type contains a single token representing the current state – a real valued quantity. Since real number is supported in high-level Petri nets, the precision of the quantity can be defined accurately. The initial specification is instantiated with an initial token. A transition *t* is used to model a timed action involving state variable *x* and the associated change rate and output result.

In many applications, we are mainly interested in the sampling of a continuous variable without defining its actual change, for example reading sensor data. In such situations, we model a continuous variable *x* by discretizing it with needed precision based on a given application. We abstract continuous variable *x*'s values with a range of possible discrete values.

Modeling Asynchronous and Synchronous Transitions

High-level Petri nets are an inherent asynchronous and distributed computation model. Enabled transitions without sharing any input places can fire concurrently and independently, thus execute asynchronously. To model synchronization, a shared place with a single token is used as an additional input place to two transitions to be synchronized.

However special care is needed to model continuous state variables involved in multiple state dependent transitions in Petri nets since these transitions can fire in any order. To accurately capture the input and output dependency between related continuous state transitions, the (input and output) transitions connected to the place need to be synchronized in a round of firings. Thus additional control structure including places and transitions is needed to realize the synchronized round of transition firings.

Modeling Hybrid States and Transitions

A hybrid system contains both discrete and continuous components, which can affect each other. A discrete component can be used to control continuous components and a continuous component can be used to trigger a discrete component when the quantity in the continuous component reaches a threshold. A continuous (input, output, or state) variable is updated continuously within each mode as time progresses while discrete mode switches are instant. A continuous (timed) action and a discrete action cannot take place at the same time. One potential problem with a hybrid model is a zeno behavior when a discrete mode switch does not happen due to an infinite timed actions in a finite time interval within a mode. In a Petri net, transitions have locality and thus concurrent transition firings are not a problem unless they are in conflict, i.e. they disable each other. Both continuous and discrete state variables are modeled with places and tokens; however a continuous state variable is modeled using a place with a single token of real type while a discrete state variable can be modeled using one or more places with 0 or more tokens of any type such as Cartesian product. Furthermore, a place modeling a continuous state variable needs to have a token representing the value of the state

variable at all times. High-level Petri nets provide a uniform treatment of discrete and continuous state variables and transitions.

Modeling Time Dependent States and Transitions

Timed Petri nets were first developed in 70s to model timed dependent communication protocols, where a time duration was associated with each transition. Other variations of timed and time Petri nets have been proposed, which either associate a duration with each place or associate a pair of bounds (lower and upper bounds) with each transition. These time and timed lower Petri nets can be effectively captured by high level Petri nets using time stamp carrying tokens and transition constraints for checking and updating token time stamps. It is straightforward to define a token type using a Cartesian product with an additional field of integer type designated as a time stamp. Each time dependent transition contains additional expressions for checking and updating token time stamps in the designated field. However, an explicit control structure is needed to model a logical clock that defines the time passage and synchronizes all time dependent transitions. A strong firing mode forces the firing of any transition reaches its upper time bound while a weak firing mode. Scheduling of timed transition firing can be complex due to various scenarios.

Modeling and Analyzing Hybrid Systems using Hybrid Hugh-Level Petri Nets

Recently, we have extended high level Petri nets with continuous places and transitions to model and analyze CPSs more conveniently and naturally. New continuous places have been introduced to capture the continuous state. Each continuous place has a component type of real and contains a single token, reflecting the current amount. New continuous transitions have been introduced to capture the change in continuous states. Each continuous transition can have built-in numerical functions for updating the relevant tokens in adjacent continuous places. The firing rate of enabled continuous transitions can be specified to the desired sampling interval. The dependencies between discrete and continuous states and transitions have been modeled through arcs between pairs of discrete/continuous places and continuous/discrete transitions.

A discrete transition can only read but not change a continuous state, but a continuous transition may update a discrete state such as a counter. Enabled discrete transitions have priority over enabled continuous transitions in firing. Firing discrete transitions are instant and do not take time while firing continuous transitions, advancing the logical clock. Explicit time constraints are not supported currently, but can be modeled in high level Petri nets, but mixing explicit timing constraints (a global logic clock) in discrete states and continuous transitions with different firing rates (local logic clocks) is complicated and requires a sophisticated scheduler to select and fire enabled transitions, and is not supported. Local timers (counters) can be used to model time-dependent states and transitions. Sophisticated transition formulas can be defined in our tool, including explicit integral representation that captures traditional implicit state space representation using differential equations. Currently, our simulator fully supports the analysis of linear hybrid systems (constant change rate). The history of continuous transition firings with regard to a continuous place plots (simulates) the function defined on this continuous state variable. The overall behavior of a CPS is defined by all the possible transition firing sequences, and can be studied by checking individual continuous state variables. We have applied these new modeling and analysis features to several well-known hybrid systems, including bouncing ball, pendulum, thermostat, water and fuel tanks, motion path planning, obstacle avoidance, and aircraft collision avoidance.

3.3.3. Alternative Analysis Techniques

In addition to exploring different model checking strategies using SPIN model checker, in which we developed different translation algorithms to automatically convert a high level Petri net model into a Promela program in SPIN, we have developed a translation technique [17] to automatically translate a high level Petri net into a first order logic formula $\varphi_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} f(s_i)$ for bounded model checking using SMT solver Z3, where $I(s_0)$ is the characteristic function of the initial state, $T(s_i, s_{i+1})$ is the characteristic function of the transition relation, and $f(s_i)$ represents the negated safety property in unrolled state s_i $(0 \leq i \leq k)$. We have implemented this translation method in PIPE+, which generates a first order logic formula from a high-level Petri net model and given safety property automatically in Z3.

We have also developed a translation technique to automatically generate a set of conditional rewriting equations in membership equation logic in Maude for term rewriting [14]. A general marking functional module fmod is defined to support all base types, which is included in all place functional modules. Each place is translated into a Maude functional module fmod. For each place p of $type(p) = s_1 \times \ldots \times s_2$, which is not a power set. We define a new Maude sort $sort_p$ preserving the above type with a unique operator $op\ OP - p\colon s_1, \ldots, s_n \rightarrow p[ctor]$ as the constructor of tokens. For each place p of $type(p) = \wp(s)$, which is a power set. We define a new Maude sort $sort_p$ preserving the above type with a unique constructor operator, and additionally define necessary set operations (union, intersection, difference) to support the power set manipulation. Each transition is translated into a conditional rewriting rule. The precondition of a transition is translated into the conditional part of rule; and the post-condition is translated into the body of rewriting rule. For an initial marking M_0, an operator $op\ init\colon \rightarrow Marking$ is defined and a sequence of equations capturing the token distribution is also defined in the following form: $eq\ init = OP - p_1; \ldots; OP - p_n$.

3.3.4. Modeling and Analysis Applications

Security patterns aim at capturing security expertise in the worked solutions to recurring security design problems. We have formally modeled and analyzed six security patterns, including account lockout, authenticated session, client data storage, encrypted storage, password authentication, and password propagation, to detect potential incompleteness, inconsistency, and ambiguity in the textual descriptions, and in turn, to prevent their incorrect implementation [13].

Android permission framework is a part of Android OS to enforce secure cross application communication. However, the android permission framework is very complex, and its descriptions are scattered on dozens of webpages. It is very difficult to understand the relationship between multiple permission levels and their potential vulnerabilities. We have developed a formal model of the Android permission framework using high level Petri nets [10]. This formal model covers more sophisticated permission relationships than previously published work, and is the first one addressing the dynamic permission granting and revoking. This formal model precisely defines the complex permission relationships that are scattered on dozens of on-line webpages and are not clearly stated or defined. This formal model serves as a starting point for us to better understand the whole Android permission framework.

3.3.5. Tool Development

We have made a lot of progress in tool development ([10], [12]). We focused on various techniques to effectively translated high level Petri nets to Promela programs in SPIN, and have carried out several experiments to push the sizes of Petri nets to be model checked. We enhanced our simulator to deal with hybrid systems. We have improved SAMTool [11] to

incrementally compose individual, high-level Petri nets to support aspect-oriented system modeling. We developed several translators to generate alternative models for external tools, including SPIN, Z3, and Maude, as well as for Java and Behavior program code template generation. Our overall tool structure is shown below:

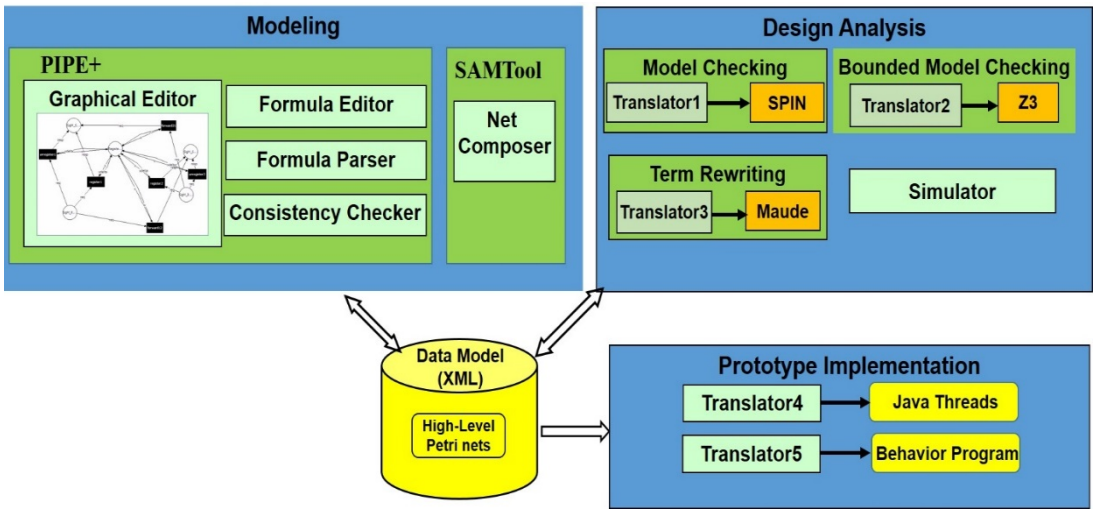

Figure 3 Overall Model base Development Integrated with SAM Tool Structure

3.4 Project Methods – Runtime Monitoring and Resilience Handling

In our research, runtime monitors were adopted to assist the development of resilient cyber-physical systems so that CPS can achieve predefined missions even in an unknown or hostile environment. The major research was conducted in three areas: integrating runtime verification within the model-driven approach to ensure confidence in the developed CPSs; adjusting controller of robotics to handle unexpected events in the environment; and monitoring online evolutionary algorithm.

Cyber physical system generates the dynamic behaviors and needs to be evaluated to some extent as good or bad, efficient or worthless, excellent or catastrophic. Such an evaluation can apply to the system in question as a whole, or to some of its components, or to a particular period of operation. We use *monitoring* to denote the act of observing and evaluating such temporal behaviors. Behaviors can be very long, spanning over a large stretch of performance and time, and densely populated with observations. They can also be very wide, recording many variables and event types. As such, they carry too much information by themselves to be easily and directly evaluated. What is distilled out of these behaviors should somehow be expressed and specified. The mathematical objects that do this job are functions that map complex and information-rich behaviors into low dimensional vectors of bits and/or numbers that indicate satisfaction of logical requirements and the values of various performance indices.

As part of the evaluation of system quality, runtime monitoring can be developed based on a gross abstraction of the behavior, which is formed by three categories – application level behavior, reaction to environment, and physical (hardware) behaviors. For example, the event of an airplane crash corresponds to a zero location on the *z* dimension and a large downward velocity at some point in time. Likewise, the misbehavior of a ground robot can be specified by the stabilization of signal input and output to a constant value. More often than not, those global catastrophic events may be related to (and preceded by) more detailed temporal

behaviors that involve intermediate steps and variables. In our case, for example, weather change during flight mission maybe shown as the density of cloud and eventually miss waypoints, which may be caused by the misreading of external targets from environment. This is the real example that was captured during our study, which is not followed by certain actions such as turning on light checking, or height checking of a UAS. In less safety-critical contexts, systems are evaluated for performance, for example, the time a client spends in a queue between requesting and being granted, or the energy consumption of a computer.

3.4.1. Monitoring Real Systems and Monitoring Simulated Models

Runtime monitoring is mainly rooted on the formal verification circles, attempting to export to the simulation-based verification domain. We need to distinguish between two major contexts where the monitoring of dynamic behaviors can take place (an elaborate discussion may be found in [28]). The first one is the monitoring of real systems where the role of monitoring is to alert in *real time* in order to trigger corrective actions, either by a supervisory layer of control or by a human operator. To reach autonomy, this work adopts the supervisory control layer which is implemented in robotics systems. It is more important to have this type of monitoring, as it is easily to see the usefulness in many domains: indicators on the control panel of a car, airplane or electronic device, monitors for physiological conditions of patients in a hospital and SCADA (Supervisory Control and Data Acquisition) systems for controlling complex large-scale systems such as airports, railways or industrial plants. It is worth to note that any information system can be viewed as performing some kind of a monitoring activity.

The other form is the one that is most popularly implemented in CPS domain. This form applies runtime monitors during model-based system design and development where all or some of the system components do not exist yet in flesh and blood and their models, as well as the model of the environment they are supposed to interact with, exist as virtual objects of mathematical and computational nature. In this work, we have presented a tool based simulation of Petri Net models for cyber physical system [16]. Simulation based verification is typical favorite for the analysis of runtime behavior of the models, provided the model has the operational semantics and/or stepwise execution semantics. The design process of such systems is typically accompanied by an extensive simulation and verification campaign where the response of the system to numerous scenarios is simulated and evaluated. Even this work is model based runtime monitoring for trustworthy CPSs, it needs to point out here, most of the work described in this section originates from the program level monitoring context, where running monitors on the real robotics is the target which has been completed.

In addition to both above two main levels of runtime monitoring process, in this work, we provide a gray-box runtime monitoring where the simulation system is executed when the real sensory data obtained from running robotics. Even in this work, our hybrid form of simulation traces constitute the input of the robotics data and monitoring process. Many techniques and considerations are shared, nevertheless, with the monitoring of real systems.

The activity of simulating or emulating a system and checking its behavior is part of the verification and validation process whose goal is to ensure that the validated CPS system behaves as expected and to avoid unpleasant surprises during or after its deployment. In some restricted contexts of simple programs or digital circuits, this process can be made exhaustive and “formal” in the sense that all possible classes of scenarios are covered. As dealing with cyber-physical systems will consider more factors external to the system development, which is due to the existence and interaction scope are not confined to the world inside a computer

for which is not captured by exact exiting models. Thus, complete formal verification is impossible, if not meaningless. To support the adaptation to change, per the environment requires, in this domain, computational science maybe able to jump in to the pool. Optimization algorithm can be able to provide (maybe) best candidates out of current selections after large amount of possibility calculations.

On top of that, it is worth noting that data inquiry during execution with the above calculation will have a large performance hit. At this moment, simulation-based lightweight verification will be the common practice, accompanied by the hope of providing a good finite coverage of the infinite space of behaviors. Our current work has considered the above situations and attempted studies and experiments.

3.4.2 Runtime Verification Integration

We have proposed a framework to develop CPSs based on a model-driven approach with quality assurance throughout the development process ([8], [11]). In the framework, runtime verification was adopted as a lightweight formal approach to detect violation of properties so as to ensure system properties at the implementation level. In our work, properties have been specified using linear temporal logic (LTL) formula. LTL properties that have been specified and analyzed at model level, as well as new properties introduced by implementation details, need to be monitored at the implementation level to improve the confidence of system implementation. However, atomic predicates of LTL formula at implementation level typically represent occurrences of events such as object creation, object initialization, method call, member data access and mutation. Such events can be specified in conjunction with conditions on function arguments, which are absent at the model level. We have proposed a set of principles to provide guidance on the mapping of atomic predicates (places) to events in runtime verification in the Petri net model.

Recently, we developed a systematic approach to enhance the above framework in behavioral programming based robotics [7]. In behavioral programming, each behavior represents an independent scenario that the system should and shouldn't follow, and these independent behaviors are interwoven at run-time, yielding integrated system behavior. Properties to be monitored in this work have been classified into two groups: properties of arbitrator's behavior, and properties based on the temporal relations among different behaviors. The former group of properties is to ensure the correctness of the arbitrator's behavior and verify the constraints of each behavior. This group of properties can be monitored for any behavioral-programming-based systems. Some properties of this group are given below:

(A1) A behavior becomes active only if it is selected by the arbitrator;

(A2) A behavior will become active at least once;

(A3) Current behavior will eventually terminate if the arbitrator notifies it to stop;

(A4) No two behaviors can be active at the same time

(A5) If both behaviors are ready to become active, the arbitrator always picks the one with higher priority.

To define the above properties in LTL, several major events have been defined for each behavior: *takecontrolT*, *takecontrolF*, *actionR*, *actionE*, and *suppress*. Event *takecontrolT* occurs whenever the method takecontrol(), in the behavior-generated code, is executed and returns true. Event *takecontrolF* is similar to *takecontrolT* except the value false is returned. Event *actionR* occurs whenever the method action() of the behavior becomes active, while

event *actionE* occurs whenever the method action() is executed. Event *suppress* occurs whenever the method suppress() of the behavior is executed. To distinguish these events for different defined behaviors, the behavior name is added in front of these events. To make the formula more concise, we use the behavior name only to represent the event *actionE* in the rest of the paper. The following JavaMop code shows the event definitions only for the behavior DetectingEntranceOne. Event definitions for other behaviors are similar.

```
event DetectingEntranceOne_takecontrolT after(DetectingEntranceOne bhv)
returning(boolean b) : execution(public boolean
DetectingEntranceOne.takecontrol()) && this(bhv) && condition(b)
{      //code to be executed when the event occurs;
}
```

Properties (A2) to (A4) can also be specified and verified at the model level. Property (A1) involves some past concept that cannot be represented in SPIN model checker. Property (A5), with regard to behavior priorities, is dealt with using an attribute of a token at the model level.

Properties (A2) and (A3) are liveness properties, and therefore cannot be verified at runtime since the monitor doesn't know when "the good thing" will happen. However, if a time constraint is associated with the occurrence of "the good thing", the above properties can be rewritten as:

<> (!*timeout* U *b_actionR*) (A2')
[](*b_suppress* → (!*timeout* U *b_action*)) (A3')

The *timeout* event is defined as below:

```
event timeout after(TimeoutEventGenerator t) :
      execution(public void TEGenerator.timeoutEvent())
      && target(t) && condition(t == teg) {
        //kill the TEGenerator if it hasn't been executed.
}
```

where TEGenerator, inherited from Java TimerTask class is used to generate time out event after a given time of period; and *teg* is an object of TEGenerator initialized when either starts monitoring of a property like property A2' or occurrence of a given event like *b_suppress* in property A3'.

The second group of properties is to validate the functionalities of each robotics by monitoring the behavior order patterns derived from system specification. Some properties of this group are given below:

(B1) An event *e1* occurs at most once before another event *e2*;

(B2) Whenever an event *e1* occurs, another event *e2* should occur later;

(B3) Whenever an event *e1* occurs, another event *e2* must occur before it;

(B4) An event *e1* should never occur before the first occurrence of another event *e2*;

(B5) An event *e1* should never occur after another event *e2*;

(B6) An event *e1* should never occur between event *e2* and event *e3*.

3.4.3 Adjusting Controllers of Robotics

We developed a framework to synthesize a controller for robot navigation automatically and monitor its execution to avoid collision with known or unknown obstacles. The framework consists of two components:

1) An evolutionary technique based on genetic algorithm was used to automatically synthesize controllers for robot navigation through simulations based on the environment specification and assumed robot sensory data;
2) A runtime monitoring method was used for added assurance to reach the target and for obstacle avoidance in a deployed controller based on actual sensory data.

The framework has been applied to a multi-car parking system. Genetic algorithm has been used to generate a path from garage entrance to exit, with a mark in between to indicate the location to turn left to park in the empty slot while avoiding collision with all known obstacles and walls. The fitness function has been designed to prefer a path that:

- follows the given road as closely as possible;
- parks at an empty slot as early as possible;
- makes less turns;
- has a shorter total travel distance.

The generated path from genetic algorithm will be mapped to a sequence of MOVE and TURN actions. Based on the sequence of actions, two approaches can be used to implement the controllers. One approach is to define the controller as one behavior to follow the whole sequence of actions. Another approach is to define the controller as multiple behaviors, each of which guides the execution of a segment of the sequence of actions.

Runtime monitor is adopted to adjust the controller of robotics to handle the gap between logic movement and physical movement, as well as environment changes such as new obstacles, which may appear after controller is generated. To ensure robotics follow the planned path as closely as possible and avoid collision, runtime monitors can perform the following tasks:

- Adjusting parameter values for each MOVE and TURN action based on the difference between its current physical location and logic location in planned path.
- Taking actions to avoid collision when it is too close to an obstacle, which may be known or unknown to the controller. If the obstacle exists when controller is generated, runtime monitors simply direct robotics to silently move inches away from the obstacle. The compensation discussed in previous task will eventually bring robotics back to original controller. If the obstacle doesn't exist when controller is generated, runtime monitor directs robotics to either walk around the obstacle and go back to planned path or move inches away from the obstacle, depending on the relative location of the robotics and the obstacle and the moving direction of the robotics.
- Monitoring major progress of the controller. Milestones can be created along the path that the controller should follow. Major progress can be specified as an LTL to describe time allowed to reach from one milestone ms1 to next milestone ms2: [](ms1 -> !TimeoutEvent U ms2). Runtime monitors can monitor the LTL formula. If it is violated, actions can be taken to get robotics back to planned path, such as using GA to generate a new path from robotics' current location to milestone ms1.

3.4.4 Monitoring Evolutionary Algorithm

Evolutionary algorithm aims to automatically synthesize robotic body plans and control software by means of evolutionary computation. It can provide artificial evolution of control systems to govern the behavior of robots. However, when evolutionary algorithm is performed to find a better controller, the potential changes in external environment, goal, or constraints on controller, may discredit the evolutionary result. We therefore developed a framework to monitor the evolutionary process so that evolutionary parameters can be adjusted dynamically in response to the external change in a timely manner.

The architecture of monitoring evolution process is shown in Figure 4. When evolutionary algorithms are launched either by the system or by other monitors to adjust its behavior to environment changes, the monitor for evolution process is also started. It is able to monitor the creation of the evolution engine; parameters of the evolution engine, such as fitness function and termination criteria; and population. More specifically, our EA monitor can

- stop evolutionary process when the outcome becomes useless due to the environment change;
- modify stop criteria, such as less time allowed for evolutionary process as a response to occurrence of some external event;
- update fitness function to find better solutions, as CPS knows more about the environment;
- modify or add operators to construct new generations to cover more search space;
- enforce rules to ensure diversity of populations and other properties.

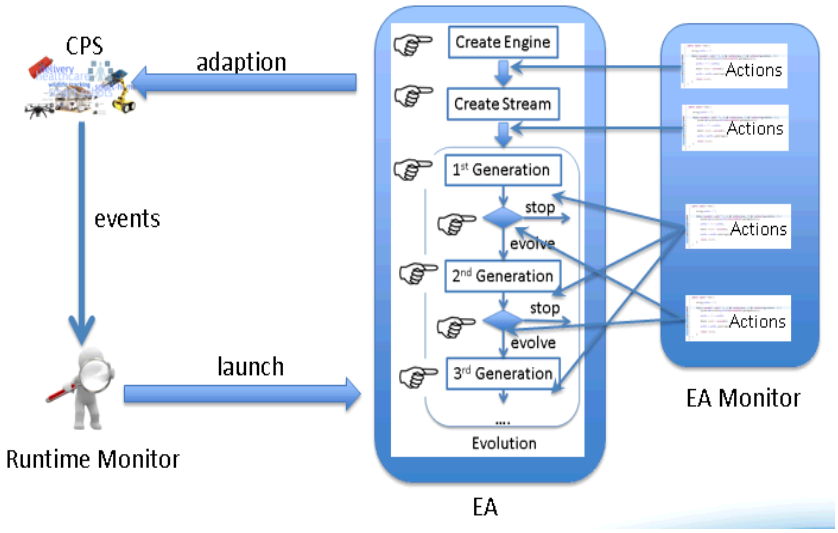

Figure 4. Architecture of Monitoring Evolution Process

The framework is implemented based on Aspect-oriented programming. As shown in Figure 5 below, the major entities in the implementation are *EAMonitor* class and *EAAspect* aspect. *EAMonitor* defines actions that can weave into EA at runtime, while *EAAspect* defines all pointcuts, and over ridable methods to be called from pointcuts.

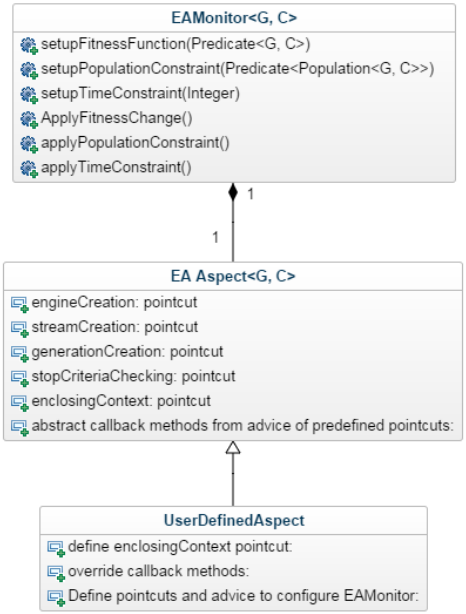

Figure 5. Implementation of Evolution Process Monitor

3.4.5 Monitoring Process on Cyber Physical System

Evolutionary algorithm can be applied to the large amount of robotics with fine motion planning. With the interaction of rigorous formalism monitoring process, the result has shown an improvement in the performance. The LTL monitors can be considered as elitism or weight of fitness function, which can either bring the stop alert to the execution or improve the next generation by this interactive alert. The exportation of these LTL formalisms and their monitoring algorithms to the cyber-physical world has to cope with the hybrid nature of such systems. The dynamics of digital systems is captured by discrete event systems such as automata, which generates discrete sequences of logical states and events. Physical systems are modelled using formalisms such as differential equations, which produce behaviors viewed as continuous signals and trajectories. Specification formalisms and monitoring algorithms should then be extended so as to express and check temporal properties of such behaviors. This topic is the focus of the present section, centered around *Linear Temporal Logic* (LTL), which is a widely adopted formalism for cyber physical system specification.

3.5 Multi-Agent Based Formal Modeling of CPS and Proof-of-Concept Testbed Design

Agent based formal modelling is well suitable for the multiple robotics system design and development. In this work, we have developed a framework of multiple agent based modelling for the task collaboration and coordination. In this framework, several operations of agents are defined and analysis. Simulation results on the multiple humanoid robotics system were presented and discussed.

To validate our design approach, we developed a typical Mobile CPS–centralized controlled mobile cyber battle system—as a testbed, with the consideration of model-driven trustworthy CPS design, which can be widely used in the academic settings and industry. In this testbed platform, we selected a set of heterogeneous robotics with typical features of

formatting and collaboration to complete certain tasks such as object recognition, navigation, and soccer game. The system was established by pursuing the air-ground collaborative cyber battle that includes both Android controlled robotics (ground robotics and UAVs) that can be tracked by web-based system. The platform is composed of three typical architectural components: (1) a centralized controlled station to register and observe the existing mobile robotics remotely, (2) a set of Android devices to control the robots, and (3) a set of heterogeneous robotics systems to implement the functionalities and complete the tasks. The runtime verification (RV) on the mobile CPS proposed in this work brings a lot of significance to the RV community due to the following challenging issues. First, Android apps are run in the Dalvik virtual machine that is different from the current RV checker's platform (on JVM). Second, the robotics systems are mostly firmware platforms, so the compilation and running can be done on the device. We will present each robotics systems first, followed by the centralized control platform in the section titled Project Results.

The developed testbeds serve three purposes with regard to the project's goal. First, the testbed provides a realistic enriched environment for CPS to validate the translation of models for the purpose of model level trustworthiness. Second, the complex testbed provides a platform for the runtime monitoring application for the implementation and the development of methodology on the resilience of CPS. This application development and design of resilience are needed to ensure the trustworthiness of the CPS application. Finally, the testbeds platform provides a large scale CPS as technical context to implement a trustworthy computing environment in the concept of an Air-Ground interoperation system.

4. RESULTS AND DISCUSSIONS

4.1 Project Results and Discussions

In sum, current results and findings of the project can be categorized into three groups. First, model-based design and model-driven engineering is a highly suitable and practical methodology for developing complex CPSs [10, 11, 57, 60, 61]. It is challenge to generate code from a formal model, but this work proposed a framework to generate Java code from Petri Net model, three basic steps of translation from a Petri Net model to Java has been highlighted (1) A class is generated for each agent net; (2) A thread is created based on each class in (1) to capture the independent active behavior of agents modeled by the agent net; (3) Agent interactions modeled through agent net weaving are translated into method calls between threads in (2) [10]. An elaborated agent oriented formal specification for the collaborative tasks of multiple robotics system was described by [60], where four operations among task agent nets were defined, and several critical properties were analyzed. These operations on agent nets provide the semantics of interactions among analysis of sophisticated agent and behavior modeling. It is possible to have certain functions (in differential equations) to be analyzed in a certain level with LTL and SMT solver [57, 61]. In [57], an array of sonar sensor based object detection was modelled by Petri Nets and a set of safety properties were validated on SMT solver Z3.

With the current results on model-based design, it is observed that (1) a gap between model and code exists and filling this gap requires dramatic efforts of researchers from both domains – modeling & analysis and software & hardware co-design; (2) trustworthy software intensive system design and development needs a rigorous design model and mathematical method on refinement to the programming to realize a smooth connection and conversion from model guideline to trusted safety implemented CPS; (3) mostly the model is case dependent due to the sophisticated and various cases of CPS structure. One of the issues is properties validated on model level will not be automatically validated at code level. Another phase of analyzing at programming level will be needed. The challenging issue still remains, such as will each case be validated on the CPS, on the simulation, will simulated results be valid on the CPS platform.

Second, runtime verification is able to be applied on some CPS at a certain level. The engagement of runtime monitor to the actual CPS will be depending on the problem domain and physical entity of CPS's. In our current study, all the cases runtime monitors were developed and applied. To handle resilience, engagement with evolutionary algorithm, runtime monitors were executed on a hybrid version of CPS, where sensory data was obtained from the running robotics and monitored on the simulated path from the sensory data. This provides an efficient way to analyze the critical properties. With all current runtime verification and development, it will be able to claim that the runtime monitor provides an efficient and effective validation of the execution of CPS.

However, as the design and system varies from system to system, physical entity to physical entity, runtime monitor is still immature to be applied to all the case of real world CPSs. To narrow the gap, we need to explore how to map simulation results to the real world systems with fewer assumptions.

Third, Air-Ground (A-G) Interoperation System can highly increase the controllability, reliability and adaptability of the ground systems. As the testbed platform, an air-ground interoperation system was developed which includes several main features. First, this A-G

system includes heterogeneous multiple robotics systems from wheeled robotics to aerial robotics. Secondly, the system is developed on the formal specification model which provides a rigid guideline for the implementation [60, 10]. Thirdly, this system is incorporated with various types of robotics platforms and interfaces of programming languages. Finally, simulated version of some collaborative robotics was developed. Thus, multiple levels of analysis can be performed in various degrees.

As the development phase, several challenge issues were faced by our team. Even though with the rigid formal specification and analysis, several years of experience of development, the software design and realization of single and multiple robotics collaboration bring a lot of challenges. With the interaction of the formal design, runtime monitoring process, we would like to claim that the A-G interoperation is a very complicated CPS.

4.2 CPS Testbed Design & Implementation

We selected the typical Mobile CPS that are widely used in the academic settings and industry to validate our approach – centralized controlled mobile cyber battle system. The system will be established by pursuing the air-ground collaborative cyber battle that includes both Android controlled robotics (ground robotics and UAVs) that can be tracked by web-based system. The platform is composed of three typical architectural components: (1) a centralized controlled space to register and observe the existing mobile robotics remotely, (2) a set of Android tablets to control the robots, and (3) a set of heterogeneous robotics systems. The runtime verification (RV) on the mobile CPS proposed in this work brings a lot of significance in the RV community due to the following challenging issues. First, Android apps are run in the Dalvik virtual machine that is different from the current RV checker's platform (on JVM). Second, the robotics systems are mostly firmware platforms, so the compilation and running can be done on the device. We will present each robotics systems first, followed by the centralized control platform.

4.2.1. Multiple Collaborative Robotics Systems

This platform design was composed of three groups of collaborative robotics systems, which includes wheeled robotics, pedal robotics and aerial robotics. The robotics form a level 3 of collaboration, based on the definition of Parker [37].

In this section, we introduce the implemented platform in two types of collaborations: communication based collaboration (CbC), and sensor based collaboration (SbC).

Definition 1. [Communication based Collaboration CbC]

For a group of (homogenous) robots Ri, and a set of tasks Tk, the collaboration schema C is formulated on the bidirectional or one directional communication channel Cc among participant robots, then we have communication based collaboration CbC:

$CbC = T_k(Cc) \bullet \Sigma_{Ri}$, where $\bullet$ denotes the collaborated operation on all participant Ri, Σ_{Ri}.

Our multiple drones formation and pedaled robotics fall in this category.

Definition 2. [Sensor based Collaboration SbC]

For a group of (homogenous) robots Ri, and a set of tasks Tk, the collaboration schema C is formulated on the sensory data Ds, then we have sensor based collaboration SbC:

SbC = $T_k(Ds) \bullet \Sigma_{Ri}$, where • denotes the collaborated operation on all participant Ri, Σ_{Ri}.

Our design of multiple robotics platform currently falls into the above two collaboration categories. Our wheeled robotics implemented as smart car parking system falls into the secondary category. A short introduction to the above multiple robotics systems will be presented in the following section.

4.2.2. Agent-based Design of Task Collaboration of Multiple Robotics Systems

An agent is a dynamic responsive entity that performs actions with response to the external events automatically. In Woodridge's (2009) definition, "an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives" (p. 21).

A Multiple Agented based System (MAS) denotes an intelligent architecture with which the system is able to make decisions and perform actions to respond to the signal from the environment. MASs have several typical characteristics, including autonomy, perception, decentralization, and self-organization. In order to ensure that MHRSs respond to their environment, the typical first-hand information about the environment is from sensory information, which forms the resource of decision making of a MAS. Therefore, a MAS must carry the capability to handle and process sensory information properly and correctly.

In order to ensure a MAS performs correct functions in response to the environmental stimuli, the behavior and control in each component play a major role towards the completion of the mission.

In this integrated MAS framework, we consider each functional component or subsystem in the MHRS as an agent. In order to complete a mission, an agent carries a capability that is defined by the tasks, actions, and operations with the necessary variables and parameters of each individual agent in a certain environment.

Each agent is described by 5-tuple: $\mathcal{A} = <\Gamma, M, \Psi, V_0, \Omega>$, where Ω denotes the set of constraints by which a mission can be completed; Γ denotes the set of tasks of the agent; M defines the set of actions that the agent can perform; Ψ specifies the operations that can be performed on the set of tasks by taking the actions from set of M. In other words, Ψ defines the relations of task set and action set while it performs an operation on the agent. Each element is described in the following.

Given a mission, a task set (Γ) defines a set of elementary logic sequences in order to complete the mission. Thus, a task set can be defined as three parts with task title (to identify the individual task), the goal of the task (to identify the end of the task and mark the completion of the task), and a sequence of steps of the task to reach the goal, a goal marks the end of steps, goal $\in$ steps, which was defined in task set.

The set of operations (Ψ) defines the interactions of agents with a certain set of tasks and actions. Performing any of these operations on one or more agents will result in a new effective agent. Each agent needs to carry fundamental operations, which include split, join, substitute, and concurrent. A detailed introduction of these operations will be given in the collaboration agent section.

Beside the sensor and communication accessories, a robotics system carries multiple actuators with complex interaction and requires a high intelligent decision making process. The parameters and logic variables handling are defined in the set of variables and parameters V_0.

In order to ensure each agent carries the desired performance during the completion of the mission, a set of constraints is specified in Ω. This set defines the regulated behaviors of tasks with the actions under a certain environment, as well as preconditions and post conditions of actions. The constraint set can be further specified as temporal logic formulae, which can be LTL in our study.

The concept of completion of the mission can be defined as a predicate on a MAS, *completion*(MAS). Ensuring the completion of a mission is a process to check, if this predicate returns true or not in the given environment R and if we are able to find the set of agents to satisfy each goal under the mission Msn.

Given any two agents $\mathcal{A}1 = <\Gamma_1, M_1, \Psi_1, V_{01}, \Omega_1>$ and $\mathcal{A}2 = <\Gamma_2, M_2, \Psi_2, V_{02}, \Omega_2>$, we will be able to describe four operators in set Ψ, we have Ψ = {joint, split, substitute, sequence, concurrency}. With the formal definition of these concepts, further analysis of behavioral based task collaboration can be done on both model level and implementation level. For the details of this integrated agent based framework, readers may refer to reference [58]. In the following, we will continue presenting the case studies as various components of proof-of-concept platform – air-ground interoperation system, which are multiple drone formation (air layer), multiple humanoid robotics collaboration (ground layer), and smart car parking system (ground layer).

4.2.3. Case Study of Multiple UAV Formation

Multiple UAV formation is one of the case studies using agent-based design, which also serves as the air layer system of the overall air-ground interoperation systems. In this case study, we focused on the level two, formation of multiple drones during mission. In the following paragraphs, the mechanical design of individual drone, the overall collaboration architecture, with task collaboration functionalities to serve for the UAV formation towards mission completion.

Mechanical Design of Individual UAV. Unmanned aerial vehicles (UAVs) are autonomous flying vehicles equipped with sensing devices and possibly weapons. They can be used to carry out tasks in dangerous situations. UAVs have many potential military and civil applications and are also of great scientific significance in academic research. However, current UAVs tend to be complex, expensive and bulky. In this multiple drone platform, we developed the coordinating control of low-cost UAVs in the formation flight, in which a group of UAVs fly in a desired graphic formation. The advantage of this study is that it will increase the efficiency of group performance, as well as the cooperation of military aircraft and group unit's concentration and simplification of control. [38]

In the following section of this report, we present the architecture of upgraded low-cost individual UAV framework, our development and implementation of the formation flight.

The processing and computation capability were updated in IRIS+^{U} with a companion computer, and includes additional accessories (case, power adapter, active cooler, and cables). This upgrade includes the Intel@ Atom^{TM} x5 Z8350 processor with 64 bit and 1.92 GHz. The computer is able to communicate with the Pixahawk through the PIN 1-4.

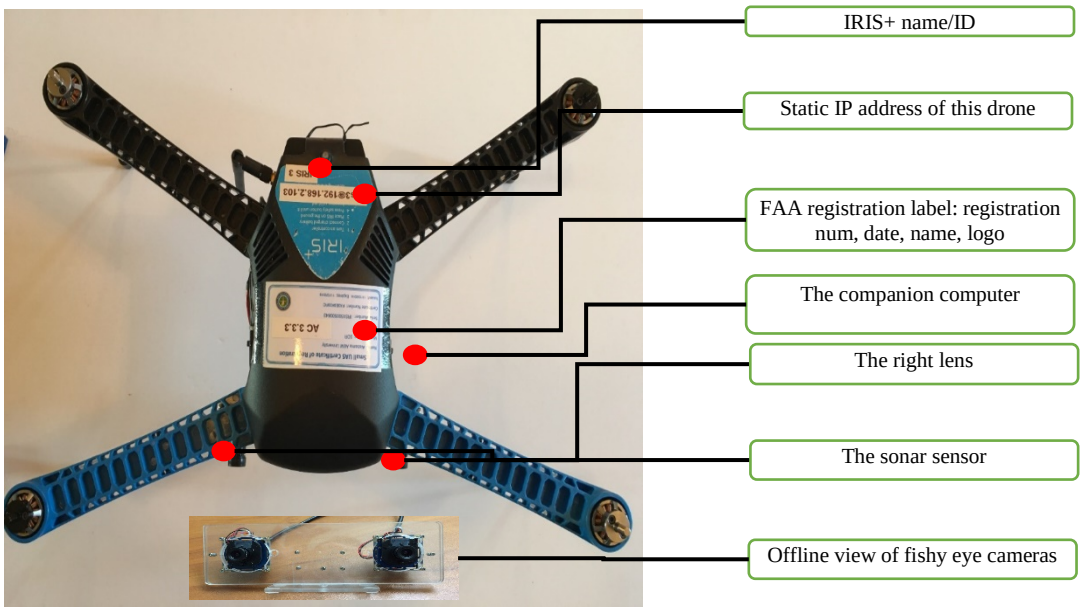

Figure 6 Overview of the IRIS+^{U} Architecture (Top view)

System Architecture. The current design of the system includes several components that support the critical flight missions during air performance time (Figure 7). From Figure 6, the main components developed in the system are shown as following: the central processor of drone, onboard computing system, sensory component, decentralized node (PC/Android), and communication component. The system architecture of IRIS+ is shown in Figure 7.

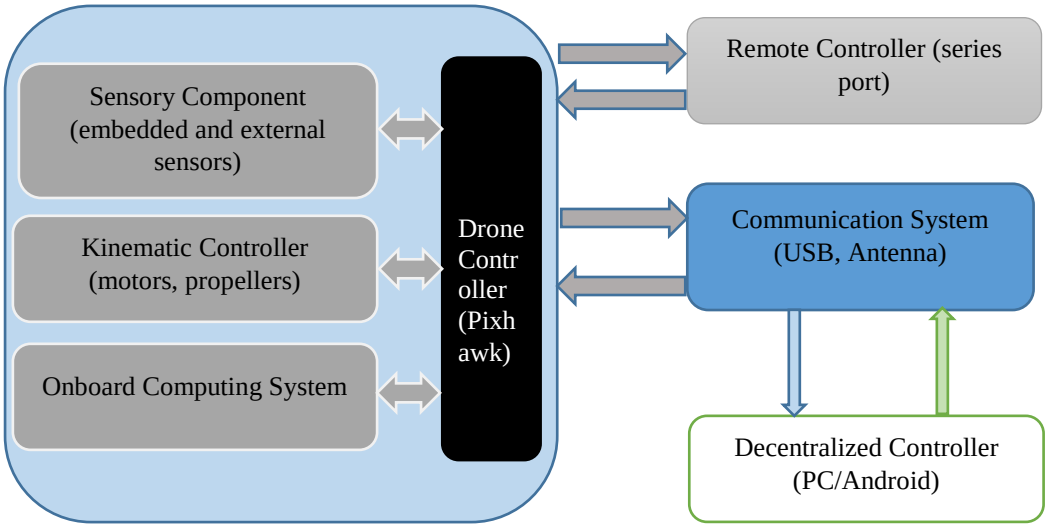

Figure 7. Overview of the System Architecture.

Object Detection Capability. The original version of IRIS+ does not carry object detection capability. In order to fulfill the purpose of this project, this IRIS+^{U} has been equipped with a sonar sensor in the front (Fig. 6), which provides accurate short distance detection. For intelligent object detection and obstacle avoidance, this IRIS+^{U} has been upgraded with 2 fishy eye lenses in the front (Fig. 6).

The baseline was well established. Two lenses were calibrated using laser calibrator. The offline results of this set can reach 0.1mm. A close view of offline lens set is shown in Fig. 6.

In order to realize remote and decentralized control of multiple drones, a communication system was set up with the large range router that is able to cover 300 square meters or above area. Each connected drone was set up with static IP that communicates with a PC with specified configuration and time frame in the loiter mode.

To fulfill the requirement of risk management of FAA regulations, a systematic methodology was developed with the consideration of hazardous situation management, hardware and software control, individual and multiple drone control, and remote control with computers, tablets, and observers. Part of the information of operational procedure of multiple UAVs is shown in Table 1.

Table 1 Operational Procedure of MRSs

1. Safety check of each drone, and make sure device is ready. Check the PC status of each drone, make sure each shown as ready for fly.
2. Place each drone on the spot, connect each drone with the battery. Each observer with RC is ready in the safe spot.
3. Once the above is done, press the start button when LED turns green.
4. After the leader communicates with all drones in the team and gets acknowledgement, the leader and the member drones will take off straight with the designated coordinates.
5. All drones will perform the actions following the program.
6. Returning starts once leader sends the signal to all team members once the leader receives the confirmation from all members, each member will call RTL and start landing.
7. The leader will land last.
8. Once all motors stop, discharge each drone.

It will be more dangerous and risky to fly multiple drones than fly one drone. The risk management and regulations have been well developed and approved by FAA under the code of Waiver # (107W-2016-01447).

Kinematics Model. This section reviews the dynamic model and control of quadcopters. A linear model was obtained using frequency-domain system identification methods. This model is valid for hovering and low velocity maneuvers [39]. A controller is used for way point navigation of a single UAV as well as multiple UAV formation flight controller design. In the current development platform, we did not consider the transient response of the relative motion between quadcopter and the surrounding air compression.

Model of leader-follower formation flight will include the position and altitude of the wind frame for both the leader and followers. In addition, a geometric analysis of the model will be considered.

For the purpose of kinematic model of UASs, we use North East Down (NED) coordination system, also known as local tangent plane(LTP). NED is a geographical coordinate system for representing state vectors that is commonly used in aviation. Three parameters in NED system

are position on northern axis, eastern axis, and vertical position, which is chosen to comply with the right-hand rule.

In the dynamic model, the separation distance is also defined along the leader quadcopter wind frame. The transformation matrix from NED frame to the leader's wind frame can be specified. Let the position vector be **P**n, the velocity vector, **V**n, and the acceleration vector, $\mathbf{a}_n$, of the NED coordinate system are adopted and are, respectively, defined as

$$Pn = [x_n, y_n, z_n],\ Vn = [u_n, v_n, w_n],\ a_n = [a_{x,n}, a_{y,n}, a_{z,n}]$$

Collision Avoidance. Two main control factors were added to enforce the collision avoidance during flight time. One is the safety zone (distance) and another is the protected height (vertical distance). The existing conflict prediction solution was adopted for the collision avoidance. Each UAV is allowed to self-optimize its own trajectory. Coordination among the UAVs when they are in each other's alert zones is by means of maneuvers which are flight modes like heading, altitude, or speed changes for each UAV. UAVs with overlapping alert zones keep each other updated about their positions. As the conflicts between UAVs depend on their relative position and velocity, the continuous models we used are relative models, describing the motion of each UAV in the system with respect to the other UAVs. For convenience, a relative model with its origin centered on host UAV was used.

Formation Control Algorithm. Formation control algorithm [40] was established by a mapping function f(.) that was defined from the leader to follower's domain in the global position system. More specifically, each drone is positioned by its own coordinates from GPS value. The leader responds to each follower with the next expected position correspondingly from the communicated information, which includes the follower's NED, position, and speed. Therefore, the next position of the drone will be defined by leader, and executed by each follower. This process will be applied to the leader too.

$$x_i' = f(x_i, \theta_i, v_i)$$

each data of x_i and x_i' will be passed bidirectional among leader and followers.

This is based on two assumptions: (1) Leader must keep the predefined geometric information (shape); (2) Follower and leader will maintain the communicated information including NED, velocity, and height, as the coordination is calculated on 2-D dimension. It should be noted that our current implementation is bidirectional, therefore, with each information or message received, the acknowledgement must be returned to sender. In the event that there is a failure to do so, then this will cause deadlock waiting, or even worse, time out, and finally the whole mission will be cancelled.

Each UAV in its formation is considered as a mass point, which is located in the center of the drone frame. We have implemented two formation model, including leader-follower formation model, and virtual structure methods [63]. The result of leader-follower formation model has demonstrated a stable and consistent flight performance during mission completion. In the leader-follower model, one of the robots is designated as the leader, while the others are designated as followers. All followers are required to follow the leader based on the bidirectional communication with commands from the leader and confirmation from followers. The whole formation shape is formed as a single rigid geometric structure. During our implementation, the structure is required to be changed with prespecified geometric 2-D shape. To model the leader-follower method, we use [64]. This method ensures asymptotic convergence of the realized trajectory to the expected node to keep formation shape.

In our implementation, the followers should listen to the message sent from leader constantly to track the leader's position and keep the shape maintained. Desired trajectories of the followers are generated based on the geometric shape that is currently maintained. Formation is formed dynamic with the environmental conditions and obstacles. Comparing to virtual structure methods, leader-follower method can realized time sensitive formation shape. Due to centralized and decentralized control, reliability of formation can be guaranteed even when some nodes (robots) loose connection with leader. This is considered a more practical method for the multiple robot formation [65].

4.2.2. Agent-based Design of Multiple BiPedaled Robotics Systems

The Bioloid GP equipped with CM-530 controller plays a key role, and is as important as a brain in a human. The CM-530 controller features a variety of options while controlling the Bioloid GP or Bioloid premium robots [41]. CM-530 is compatible with dynamixel AX series, AX-S1, and supports IR sensor, touch sensor, DMS, IR sensor array, and Gyro sensor. Wireless commutation is via ZIG-110A or BT-110A. Other than those, it comes with servo motors and includes LED motion, except other necessary accessories. It is almost comparable with CM-730, and has some more features than the new coming CM-900. Extensive projects can be developed and constructed through this highly compatible version. The CM-530 controller has six 5-pin ports that are used for input and output devices (including sensors and third party equipment). A hardware interface from the e-manual [42] is shown in Figure 8.

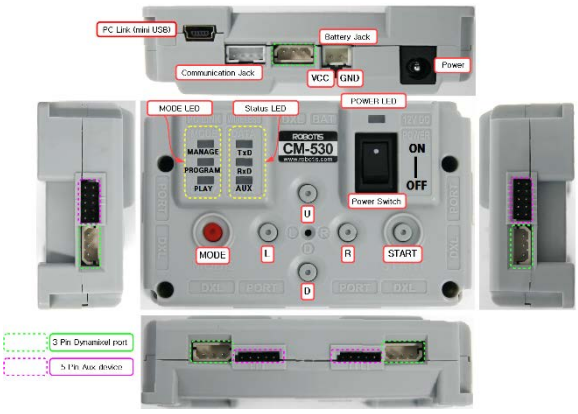

Figure 8. Top View of CM530 [42]

The controller also has a PC link that is used to load the code to the controller. The remote controller for the Bioloid humanoid robot is connected to the robot through the zig110/BT-100 wireless communication module. The wireless communication provides a versatility to the systems by giving more options to the user.

It is worth noting that BIOLOID GP robot has been built using advanced material – which reduces the weight and increases the force dramatically – in comparison to the premium and beginner version. However, nothing is perfect. The loosening of the screws and sliding during walking are the two main problems during program testing. Adding pads to the feet to increase the friction when in contact with the floor was necessary, especially when speed is increased.

A role and control flow design model of four robotics was developed and has been illustrated in figure 8. From the figure, it can be noted that a simple collaborative motion schema has been implemented in the communication based strategy. For each striker or defender, with every action performed, a message was passed to the teammate and opponents. Thus, the teammate and opponents take corresponding action per the message.

In this current implementation, the action was statically planned, as the messages for corresponding (communicating) are predefined in the controller.

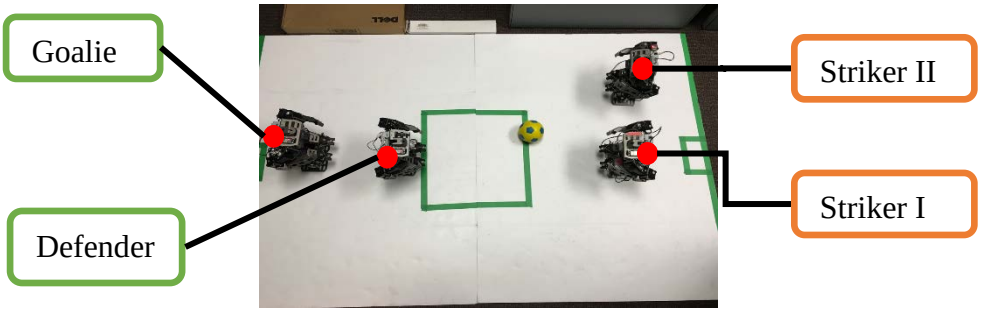

Figure 9. Current Layout of Soccer Bot Implementation

Put simply, each team has two roles—defensive and offensive—which are played by a striker and a defender. The striker robot (SR) starts the game by looking for the ball. Once the SR locates the ball, the SR kicks it after calculating the leg motion. Meanwhile, the defender robot (DR) moves repeatedly along a horizontal line in the field while facing the DR and watching the ball. Peer communication is used to trigger the defense. The current implemented layout is shown in Figure 9.

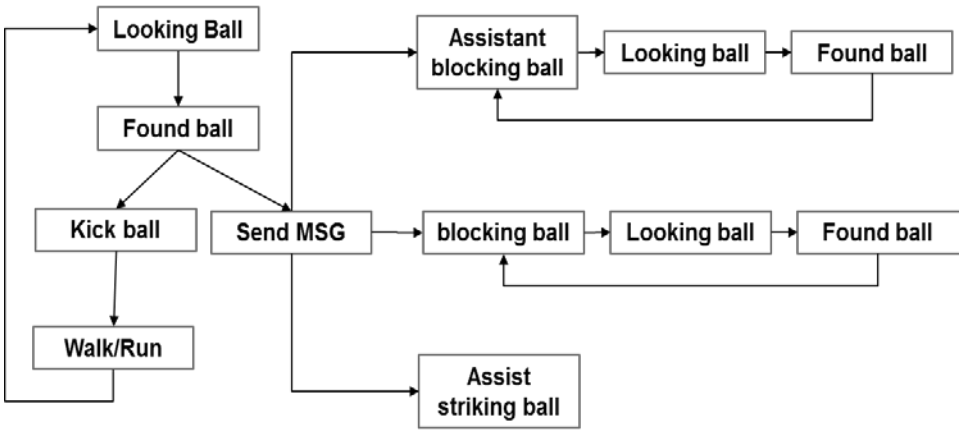

Figure 10. Flow chart of Four Bot Soccer

A standard Bioloid humanoid robot kit consists of two 8-DOF legs, namely, a 3-DOF hip, a 1-DOF knee, a 3-DOF ankle, and 1-DOF that imitates the toe joint. Each leg can be modeled as a kinematic chain with nine links connected by eight revolute joints. Each dynamixel has a local frame in the pitch, roll, and yaw (X_i, Y_i, Z_i) directions on the axis. Each has a global value relative to the base frame in the center of the robot's waist. To model and realize collaborative actions, the Bioloid robotic platform generates a steady foot gait in relation to the zero-moment point (ZMP) [49, 52] in both one-phase and two-phase supports, which stand for different function positions. For each motor at the waist, knee, and ankle, we considered two vectors, namely, position and orientation, using the world coordinate system for pitch, yaw, and roll in the body frame reference:

$Leg_{position} = [x_L, y_L, z_L]^T$ and $Leg_{orientation} = [\alpha_L, \beta_L, \gamma_L]^T$,

where T is the transition matrix in the world position system.

4.2.3. Formal Modeling Implementation of Smart Car Parking System (CPS^i)

This is a design of an intelligent car parking system (CPS^i) created using an integrated runtime framework. This work includes high level modeling in Petri Nets, model level analysis, implementation, safety properties, monitoring specifications in LTL, runtime instrumentation, and program analysis in source code (and byte code).

The automatic car parking system consists of a collective of automatic robotics system that must handle uncertainty. It is developed as a platform for modeling, analysis and runtime monitoring.

CPSi Architecture. The current CPS^i comprises of a decentralized architecture and has three main aspects: Field Robotics System (FRS), Communication Component (CC), and Control Station (CS) (see Figure 11).

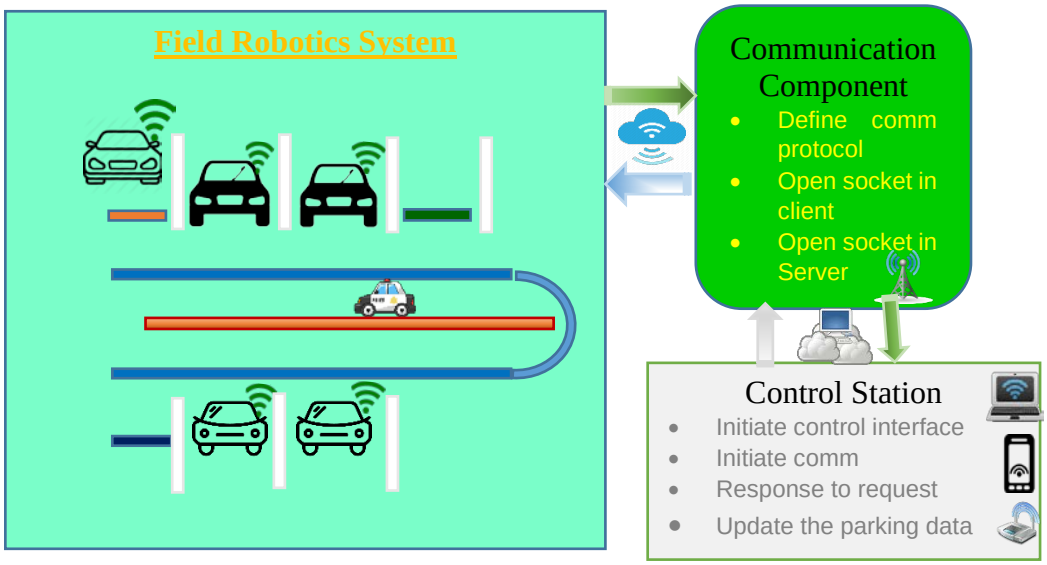

Figure 11. Overall Architecture of Automatic Car Parking System

FRC is the program running on the robot, which includes the fundamental functions of individual motion (e.g. movement, sending message to CS, communicating to peers, object identification, path generation, and others), and algorithms of planning, updating, and scheduling. In addition, some advanced features need to be considered in this component, such as uncertainty, dynamic objects detection, mapping, etc.

The functions of CC focus on the sending and receiving of information from CS and FRC. The current program was implemented in a client-server structure. In addition, a simple communication protocol was defined. WiFi was also used in the current version. CC is widely used in field robots and PC.

CS provides a loose monitoring control of field robots. The monitoring control falls in three aspects: individual motion control, parking garage information with update, and overview of all robots and motion planning. Human intervention in the control can override each individual robot's functions.

Object Detection. Triangulation is the process of deciding the location of any object from two remote points. Triangulation has a vast area of application as well as it has multiple types, including geometric triangulation, beacon triangulation, geometric circle intersections and organization of the formula for triangulation vary depending on the field of application. In terms of localization using sensors, it is the process of determining the objects location based on two sensors measurement. In this regard, the triangulated object is called landmark. In triangulation the tracked object must be in the range of two sensors. It is able to provide the coordinate and distance of objects based on geometric calculation and this geometric calculation is based on the Euclidian geometry. Besides getting the distance and coordinate of an object from a known point, the heading of a robot can also be calculated using triangulation based on the landmark detection. In our work, an array of 12 virtual sonar sensors were used to generate the object detection data and map info [56]. SMT-solver was used to verify the correctness and safety of the system [55].

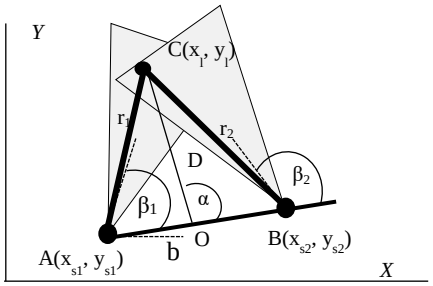

Figure 12. Triangulation using two sensors

The basic terminology about the triangulation can be derived from Fig. 11. Here the definition of triangulation and related terminologies are provided based on the 2D Cartesian coordinate.

Assume *A* and *B* are two sonar sensors and the distance between their bearings is *b*. x_{s1}, y_{s1} and x_{s2}, y_{s2} are the known position of sensor *A* and *B* in the coordinate. α_1 and α_2 are the angles that refers to the heading of sensors *A* and *B*. Let C be any object intersected by the two sonar sensor *A* and *B*. r_1 and r_2 are the distances of the object C measured from the sensor A and sensor B consecutively. Now to calculate distance of *C* from *O* and the location of $C(x_l, y_l)$, the following rules has been adopted

$$x_1 = x_{s1} + \frac{1}{b^2}\,(d_{xs}D2\ \pm |\ dys\ |\sqrt{r2^2ds^2 - dr^4}) \qquad \text{(i)}$$

$$y_1 = y_{s1} + \frac{1}{b^2}\,(dysD2\ \pm |\ dxs\ |\sqrt{r_2^2 2\ \ ds^2 - dr^4}) \qquad \text{(ii)}$$

Where variables in equation (i) and (ii) can be defined as follows

$$d_{xs} = x_{s1} - x_{s2} \qquad \text{(iii)}$$

$$d_{ys} = y_{s1} - y_{s2} \qquad \text{(iv)}$$

$$b^2 = d_{xs}^2 + d_{ys}^2 \qquad (v)$$

$$D^2 = \frac{r_1^2 - r_2^2 - b^2}{2} \qquad (vi)$$

Behavioral Programming and Implementation in CPSi System. In the decentralized architecture in the current CPSi system, one of the major contributions is the adoption of behavior programming architecture other than perception-based implementation.

Behavior Control was pioneered by Rodney Brooks at the MIT Artificial Intelligence Laboratory. Brooks adapted the insect's complex behavior, which is a series of alternations of simple behaviors, to robot control architecture to come up with what was originally known as subsumption architecture [43, 44]. Later the subsumption architecture was renamed to behavior control. A behavior is a set of conditions and the action to be taken to satisfy those conditions.

The condition could be some sensory information, either external or internal, on which it issues an action. To coordinate the action, each behavior can disable some of the other behaviors, which conflict with its own action. This is based on the priority assigned to the behaviors and this capability to override other commands is called subsumption. If we were designing a smart car parking behavior control, it could be represented as a hieratical structure of sets of structured behaviors (Table 2). In Table 2, CS denotes Color sensor, IR denotes ultrasonic sensor.

Table 2. CPS^{i} Behavior Control

BEHAVIOR	CONDITION	ACTION	PRIORITY
Collision Awareness	Object detected by IR	Stop	High
Searching Blue Line	Both CS detect blue	Move forward	Medium
Detecting Blue Line	Left CS detect blue	Make turns	Medium
Detecting entrance	Left and right CS detect a pair of color for each entrance.	Make turns	Medium
Detecting Lot	Left and right CS detect a defined pair of color for each lot.	Make turns or move forward	Low
Parking at Lot	Left and right CS detect a defined pair of color for each lot. Front IR returns infinite	Make turns	Low

The concepts of Behavior Programming [45] as implemented in leJOS NXJ [45] are very simple:

- Only one behavior can be active and in control of the robot at any time.
- Each behavior has a fixed priority.
- Each behavior can determine if it should take control.

The active behavior has higher priority than any other behavior that should take control. The Behavior API is composed of only one interface and one class. Behavior interface defines the individual behavior classes. Behavior interface is very general and defines three public methods. Each task that the robot must perform can be defined in its own class. It works quite well because, even though the individual implementations of a behavior vary widely, they are all treated alike. Once all the behaviors are created, they are given to an Arbitrator to regulate which behavior should be activated at any time. The Arbitrator class and the Behavior interface are located in the lejos.subsumption package.

According to the behavior descriptions given in Table 2, considering the hierarchies of behavior priorities given by Brooks [44], we have a structured behavior representations in Figure 13.

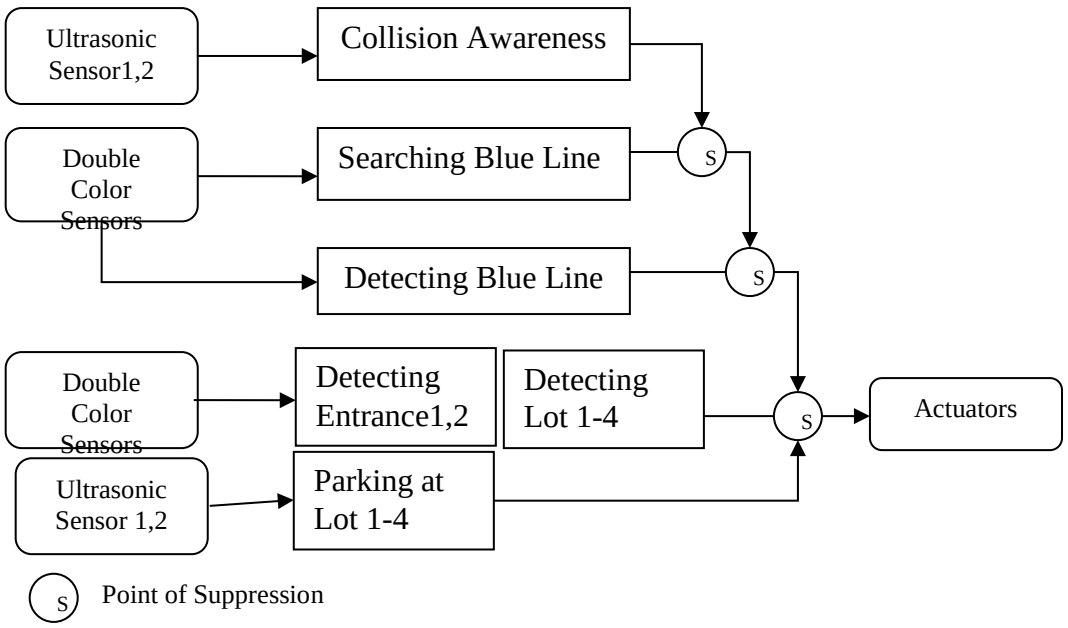

Figure 13. Hierarchy of Behavior of CPSi

A concern often associated with coding and invocation of behaviors from behavior, named as composed behavior invocations. This may cause the issue of dealing with conflicting behaviors in different modules. In that case, we must provide strategic plan to either yield the conflicted behavior from the current one, or split the current or conflict one from the current invocation. Both process enable the discovery of under-specified requirements and constraints of the system.

By using runtime monitoring, behavioral Java programs can be monitored during execution translating into a language specific to the model checker. In our current implementation, we use an abstraction point that focuses on the program execution states as insertion points. This technique has been used to validate several execution threads of smart car parking system during the runtime. Shifting from development phase to implementation phase, it is considered both model checking and runtime monitoring. We also can look ahead of the program execution and avoid any typical program problems, such as deadlock and data racing. When monitoring behaviors in CPS, the priority of behaviors with interactions of action alerts triggering, suppressing of current action, and synchronize the actuators and sensory data can be detected using fine specified conditions based on the LTL formulae and calibrated events.

The API for the Behavior interface is as follows.

- boolean take Control() – Returns a boolean value to indicate if this behavior should become active. For example, if a touch sensor indicates the robot has bumped into an object, this method should return true. This method should return quickly, and not perform a long calculation.

- void action() – The code in this method begins performing its task when the behavior becomes active. A behavior is active as long as its action() method is running, so the action() method should exit when its task is complete. Also, the action() method should exit promptly when suppress() is called. When it exits, it should leave the robot in a safe state for the next behavior.

- void suppress() – The code in the suppress() method should immediately terminate the code running in the action() method. It should also exit quickly.

4.2.4. Summary of Air-Ground Interoperation Platform

Proposed in 1996 by DOD, a high level architecture (HLA) was developed for the next generation of modeling and distributed simulations [46-48]. In this HLA, a set of low level services were composed to support interoperability among a set of elementary simulators. A considerable amount of computing power is needed to run large-scale simulations of high-fidelity entities, which is not currently available. Therefore, simulations must be able to selectively vary the resolution of entities according to the situation. On the other hand, to realize the purpose of war games and Command and Staff Trainers, the implemented platform on the physical device to reflect the HLA in an aggregate-level is still far from reality. Most researchers are still focusing on the constructive simulations and presented units due to the low resolution, and less specifically on the modeling of these units. In order to realize a real world close-the-loop simulation in a set of heterogeneous devices, dramatic efforts were made by our team, ranging from modeling, designing, and implementation to testing and verification. Our purpose was to address a number of significant technical and methodological issues related to the multi-resolution challenge from an experimental perspective, and using a combined version of existing architectures and models and to realize a run-time resilient infrastructure. To achieve this goal, we selected a simplified model of the air-ground combat architecture model. To implement the final engagement model in the implemented platform and reflect the design principles suitable for the runtime verification methodologies, a set of applications and experiments were selected and completed. In the following sections, we will illustrate the selected architecture models, our supporting scenarios for each component, and list the findings with regard to the supporting environment. An overview of our Air-Ground Interoperation Architecture is shown in Figure 14.

In this architecture, two main ideas related to HLA were illustrated: subsumption architecture and perception-based design. The engagement of these two was achieved in a smooth and integrated system named Smart Car Parking System. Due to the component of pedaled robotics system, a new control architecture MIMO [49] was adopted. In the air troop, the high level design needs to cooperate with this architecture and realize the functional layer in the execution phase, which includes object detection, collision avoidance, navigation, reconfiguration, and motion planning. In addition, it should be noted that these execution functions are spread among all component systems. Our current platform is able to realize the integration of the above architectures and seamless engagement among types of robotics. The following four aspects will be presented and discussed here: (1) Intercomponent Communication, (2) Multi-level Resolution, (3) Aggregation Mechanisms, (4) Centralized / Decentralized Control Approach.

INTERCOMPONENT COMMUNICATION

In order to reduce the cost, we chose a single quadcopter and an embedded router (for indoor navigation) to form the air group component. Each unit in the air group will be able to coordinate with the ground group to realize high level surveillance and remote navigation (guidance). This air-to-ground communication has been designed to be bidirectional wireless

communication, which is also monitored in a central station. Firstly, the mission of air group units is determined, such as position, velocity, and target. Each unit in each of the ground groups is able to perform inner communication with the units within the group. The intercomponent communication among the ground component is not supported yet. However, it is supported for the ground component in the central station in wireless communication. With this, a partial communication among ground group is achieved.

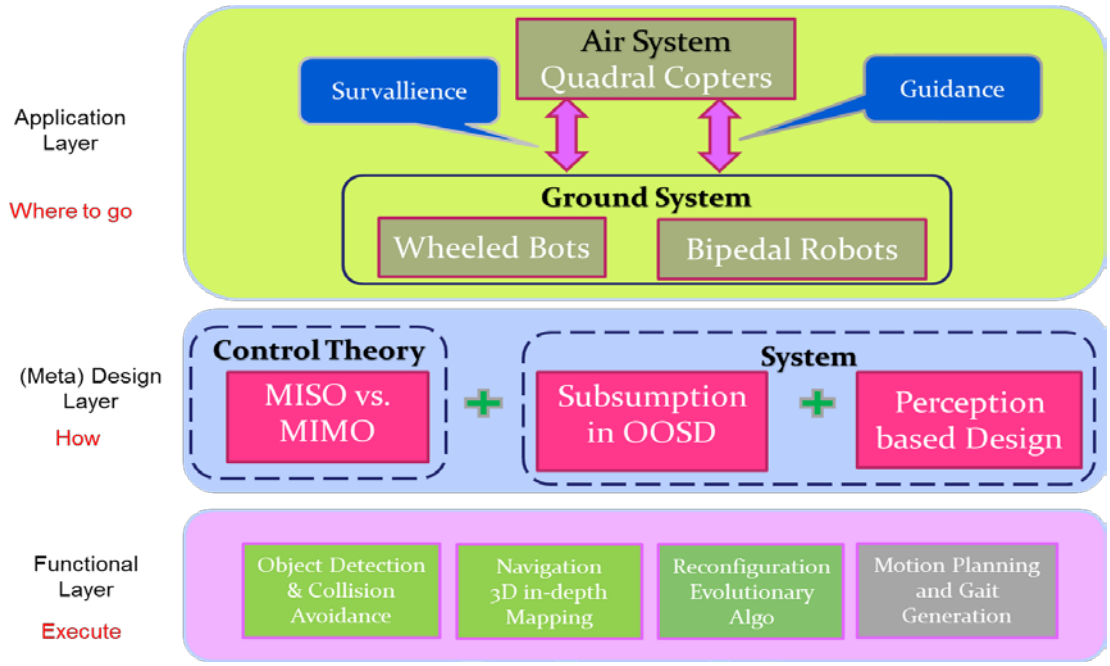

Figure 14. Architecture of Air-Ground Interoperation System

MULTIPLE LEVEL RESOLUTION

In the aggregation and disaggregation process of each group, a patrol of aircraft must be able to track each unit of the ground group through one of the above approaches (Section 5.1). Classically, several approaches can be used to manage the process of aggregation [50, 51]. In our implementation, we focused on the fixed geographical area with the basic mechanism for the patrol. An area of patrol was selected and placed around the area; this area will be specified in the system so that the unit will be informed if the motion is going to be out of range. The radius of the area can be chosen if the largest interaction is with the air/ground sensors.

AGGREGATION MECHANISMS

Each group maintains its own influence sphere and is able to identify the objects within the resolution level. This sphere can be overlapped in the real battle field and may be changed constantly, as soon as the location is updated. Location is a very sensitive data in the mechanisms of aggregation, which can affect the implementation of the aggregation-disaggregation in both centralized and decentralized approaches. The update resolution can affect the quality of patrol, and further, the quality of the each component's decision making.

Any resolution and location update affects the next step of motion, which can be amplified to the overall air-ground system coordination and interoperation.

It should be noted that identifying the conditions and constraints in a complete set is difficult if we consider the behavior of the aircrafts in the initial single craft. With the interaction with environment, unpredicted factors will cause any type of behaviors and actions in the air performance. However, this can be avoided if the air units are in a formatting group and can be tracked, which is our current design. The limitation of our implementation right now is that the dynamic adaptation was not implemented in the formatting strategy, which makes the overall mission planning a very strict template. This implementation will be difficult in any realistic aggregation event in random determination. A well-established rendezvous approach with a set of requirements will be highly recommended:

- An air unit located within the influence sphere must be responsive to the ground signal,
- A surface-to-air mission planner from each control in the ground unit (with support of the high resolution sensors),
- A systematic strategy of guidance from air units to the ground group.

CENTRALIZED/DECENTRALIZED CONTROL APPROACH

Following the idea of HLA, our current implementation in the CPS platform is able to handle the patrol of each group of units with regard to the displayed data, such as location and resolution, under the condition of effective communication. A centralized station is developed to monitor and remotely command the units if necessary. This patrol federate contains the knowledge of each unit and each component globally. The phase of each component and each unit must be known permanently along with the position. More communication is needed for the knowledge and update due to the centralized process.

Meanwhile, to increase the flexibility and capability to handle the patrol of each individual unit, in each ground component, a type of decentralized control was implemented. An optimization of the decentralized control rules to increase the resilience of the system is highly desired.

Decentralized control is able to provide flexibility in command and control, improve the performance during environmental sensing and execution. Decentralized control is implemented by passing control to each unit, which can be formulated by multiple individual intelligent device (UASs or robotics) or formulated by the same set of tasks. The communication interface is developed by the unit control. It is easy to note that the minimum unit of this control format is any individual. The maximum unit of this control will be the overall whole group (A-G system). At the top, the information flows more frequently between the center station and unit controller than center station and the individual. In contrast, at the bottom, the data flows more frequently between the unit station and the individual center station than that between the center station and unit controller.

5. CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORKS

5.1. Conclusions

In this project, we developed and validated an integrated trustworthy formal framework that is able to represent and analyse large scale and complex cyber physical system in an unknown environment. Specifically, CPS's formal design and analysis techniques using Petri Nets and SAMPIPE+ were developed. A trustworthy computational tool set with the consideration of inherently safe and secure, and resilient cyber physical software intensive systems was developed using an evolutionary algorithm. An exemplar to validate this resiliency approach with the formal model was implemented.

A model-driven approach was developed and supported by a tool set. This model-driven approach can be extended to deal with hostile and unexpected unknown environments and cyber-attack scenarios. In the model level analysis, an agent based formal model played a critical role to describe the behavioral and functional aspects of CPSs. With the layered architecture and defined operations on multiple agents, more sophisticated task cooperation and collaboration can be analyzed.

The formal model framework, integrated resilience handling methods were applied on a complex air-ground system (A-G System) which is integrated with wheel robotics, bipedal robotics, and UAVs. This A-G System performed as a testbed for the integrated formal framework.

In sum, formal model (e.g. Petri Nets) provides a precise and rigid representation of CPS at both the structural and behavioral view. Model checking and simulation tools on the formal model support trustable and robust CPS development. Meanwhile, the translation schema from the formal model to implementation provides the systematic guidance for the design and implementation/programming.

A short summary of the group's research contributions is provided in Appendix 1. At the point of this project report, 26 publications including conference, journals and workshops were submitted. Among them, 22 were accepted and 4 are under review.

5.2. Recommendations and Future Works

At present, none of the techniques explored in this project have reached maturity.

In order to advance the multiple layered, hierarchical and evolutionary platform, with high performances, we recommend a more integrated approach to the A-G Interoperation System:

(1) Formulate the parameterization scheme for A-G Interoperation System design that enhances the combined adaption and structural performance.

(2) Develop an analytical/numerical methodology that will enable predictions of the complex component-based design patterns in the formal specifications. The new methodology, based on a mathematical framework for establishing and unifying the existing solutions, will compute the multi-objective optimal problem.

(3) Develop software that integrates low and high complexity design and optimization to enhance resilience and adaptation to the hostile attacks. The software shall be appropriate for use by A-G Interoperation System developers, and will

accommodate progressive design by use of the A-G Interoperation System consistent with the model based system-level design requests and will be suitable for the formal analysis and reasoning for the security issues.

With the current research results, we have formulated following future works:

- Model and analyze variable change rates of continuous transitions and real-time constraints explicitly in hybrid systems.
- Analyze the applicability and scalability of the overall framework, including integrating formal modeling with simulation-based evolution algorithm for robot controller design and generation, and work on more case studies (currently, a multiple robotic soccer game, and multiple robotic car parking system).
- Investigate more efficient collaboration among different monitors, such as a monitor starts/suspends/stops as a result of another monitor.
- Extend our approach to deal with more complicated scenarios such as drones and more advanced unmanned ground vehicles.

APPENDIX 1. SELECTED PROJECT PUBLICATIONS

List of group contribution through the project performance:

1. He, X., "Modeling and Analyzing Cyber Physical Systems Using High Level Petri Nets", *The 2018 IEEE International Conference on Software Quality, Reliability, and Security* (QRS'18), July 16 – 20, 2018, Lisbon, Portuguese (submitted).
2. Sun, Z., Zeng, R., He, X., "A Method for Predicting Two-variable Atomicity Violations", *The 2018 IEEE International Conference on Software Quality, Reliability, and Security* (QRS'18), July 16 – 20, 2018, Lisbon, Portuguese (submitted).
3. He, X., Dong, Z., and Fu, Y., "A Systematic Approach for Developing Cyber Physical Systems," *The 30th International Conference on Software Engineering and Knowledge Engineering*, July, 2018, Redwood, CA (Accepted).
4. Alam, D, He, X., and Chu, W. "Modeling and Analyzing Hybrid Systems Using Hybrid Predicate Transition Nets," *The 30th International Conference on Software Engineering and Knowledge Engineering*, July, 2018, Redwood, CA.
5. He, X., Dong, Z., Yin, H. and Fu, Y., "A Framework for Developing Cyber Physical Systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no.9, 2017, 1361 – 1386. This paper is an extended version of [6].
6. Alam, D. and He, X., "A Method to Analyze High Level Petri Nets using SPIN Model Checker", *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no.9, 2017, 1455 –1482.. This paper is an extended version of [7].
7. He, X., "Modeling and Analyzing the Android Permission Framework using High Level Petri Nets", *Proc. of The 2017 IEEE International Conference on Software Quality, Reliability & Security* (QRS'17), Prague, Czech Republic, (July 25-29).
8. He, X. Dong, Z., Yin, H. and Fu, Y., "A Framework for Developing Cyber Physical Systems", *Proc. of the 29th International Conference on Software Engineering & Knowledge Engineering* (SEKE'17), Pittsburgh, USA (July 5-7, 2017) (Best Paper Award).
9. Alam, D. and He, X., "A Method to Analyze High Level Petri Nets using SPIN Model Checker", *Proc. of the 29th International Conference on Software Engineering & Knowledge Engineering* (SEKE'17), July 5 - July 7, 2017, Pittsburgh, USA.
10. He, Xudong and Fu, Yujian, "Modeling and Analyzing Security Patterns using High Level Petri Nets", *Proc. of the 28th International Conference on Software Engineering and Knowledge Engineering* (SEKE'16), Redwood, CA, July 1-3, 2016.
11. He, Xudong, Zeng, Reng, Liu, Su, Sun, Zhuo, and Bae, Kyungmin, "A Term Rewriting Approach to Analyze High Level Petri Nets", *Proc. of the 10th Theoretical Aspects of Software Engineering Conference* (TASE'16), Shanghai, China, July 17-19, 2016.
12. Pereyra, Jose, Mostafavi, Ali, and He, Xudong, "Multi-Agent Framework for Complex Adaptive Modeling of Interdependent Critical Infrastructure Systems", *Proc. of the 2016 Construction Research Congress* (CRC'16), San Juan, Puerto Rico, May 31 – June 2, 2016.
13. Xiao, Fangxiong, Min, Huaqing, Bo, Xu and He, Xudong, "A Formal Modeling Method for Cost-Aware Service Composition", *Proc. of the 25th International Conference on Software Engineering and Data Engineering* (SEDE'15), San Diego, CA, Oct. 12-14, 2015.
14. Liu, Su and He, Xudong "PIPE+Verifier - A Tool for Analyzing High Level Petri Nets", *Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering* (SEKE15), Pittsburgh, July 6-8, 2015.
15. Fu, Y., and Shuvo, M., "An Approach to Analyzing Adaptive Intelligent Vehicle System", 3rd International conference on Control, Decision, and Information Technologies, Saint Julian's, Malta, Apr 6-8, 2016
16. Shuvo, M., and Fu, Y., "Sonar Sensor Virtualization for Object Detection and Localization", IEEE Southeast Conference, Norfolk, VA, Mar 30, 2016.
17. Fu, Y. and Choosilp, W., "Specification based testing of Android System", International Journal of Wireless and Mobile Networks. Vol. 11, No. 2017.

18. Fu, Y., Dong, Z. and He, X., “Formal Modeling and Analysis of Collaborative Humanoid Robotics”, International Journal of Robotics Applications and Technologies (IJRAT), 2018.
19. Fu, Y. and Choosilp, W., “Model-based Development of Trustworthy Cyber Physical System”, Proceedings of the IEEE SoutheastCon, Florida West Coast Section-Tampa Bay Area, FL, April 2018.
20. Fu, Y. and Choosilp, W., “Test Driven Development of Cyber Physical System”, International Journal of Cyber-Physical Systems (IJCPS). 2018. (In Process)
21. Qian Feng, Rundong Zhou, Chengcheng Xu, Yao Cheng, Brian Testa, and Heng Yin. Scalable Graph-based Bug Search for Firmware Images, *to appear in the 23rd ACM Conference on Computer and Communications Security* (CCS'16), October 2016.
22. Qian Feng, Aravind Prakash, Minghua Wang, Curtis Carmony, and Heng Yin, “Origen: Automatic Extraction of Offset-Revealing Instructions for Cross-Version Memory Analysis”, to appear in Proceedings of the 11th ACM Asia Conference on Computer and Communications Security (ASIACCS'16), May 2016.
23. Curtis Carmony, Mu Zhang, Xunchao Hu, Abhishek Vasisht Bhaskar, and Heng Yin, "Extract Me If You Can: Abusing PDF Parsers in Malware Detectors", In Proceedings of Network and Distributed System Security Symposium (NDSS'16), February 2016.
24. Aravind Prakash and Heng Yin, "Defeating ROP Through Denial of Stack Pivot", In Proceedings of 2015 Annual Computer Security Applications Conference (ACSAC'15), December 2015.
25. Minghua Wang, Heng Yin, Abhishek Vasisht Bhaskar, Purui Su, and Dengguo Feng, "Binary Code Continent: Finer-Grained Control Flow Integrity for Stripped Binaries", In Proceedings of 2015 Annual Computer Security Applications Conference (ACSAC'15), December 2015.
26. Mu Zhang, Yue Duan, Qian Feng, and Heng Yin, "Towards Automatic Generation of Security-Centric Descriptions for Android Apps", In Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS'15), November 2015.

REFERENCES

1. Mitchell, R. and Chen, I. R., "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computer Survey.*, 46(4):55:1–55:29, Mar. 2014.
2. Clarke, E. M., Grumberg, O., and Peled, D. A. *Model Checking*. The MIT Press, January 2000.
3. Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
4. Mu Zhang, H. Y., Duan, Yue, and Zhao, Z. "Semantics-aware android malware classification using weighted contextual api dependency graphs," In *21th ACM Conference on Computer and Communications Security (CCS'14)*, Scottsdale, Arizona, USA, (November 2014).
5. Mizzi, R., *An Extensible and Configurable Run time Verification Framework*. PhD thesis, University of Malta, 2012.
6. He, X., "Modeling and Analyzing Cyber Physical Systems Using High Level Petri Nets", *The 2018 IEEE International Conference on Software Quality, Reliability, and Security* (QRS'18), July 16 – 20, 2018, Lisbon, Portuguese.
7. He, X., Dong, Z., and Fu, Y., "A Systematic Approach for Developing Cyber Physical Systems," *The 30th International Conference on Software Engineering and Knowledge Engineering*, July, 2018, Redwood, CA.
8. He, X., "Modeling and Analyzing the Android Permission Framework using High Level Petri Nets", *Proc. of The 2017 IEEE International Conference on Software Quality, Reliability & Security* (QRS'17), Prague, Czech Republic, (July 25-29).
9. He, X. Dong, Z., Yin, H. and Fu, Y., "A Framework for Developing Cyber Physical Systems", *Proc. of the 29th International Conference on Software Engineering & Knowledge Engineering* (SEKE'17), Pittsburgh, USA (July 5-7, 2017) (Best Paper Award).
10. Alam, D. and He, X., "A Method to Analyze High Level Petri Nets using SPIN Model Checker", *Proc. of the 29th International Conference on Software Engineering & Knowledge Engineering* (SEKE'17), July 5 - July 7, 2017, Pittsburgh, USA.
11. He, Xudong and Fu, Yujian, "Modeling and Analyzing Security Patterns using High Level Petri Nets", *Proc. of the 28th International Conference on Software Engineering and Knowledge Engineering* (SEKE'16), Redwood, CA, July 1-3, 2016.
12. He, Xudong, Zeng, Reng, Liu, Su, Sun, Zhuo, and Bae, Kyungmin, "A Term Rewriting Approach to Analyze High Level Petri Nets", *Proc. of the 10th Theoretical Aspects of Software Engineering Conference* (TASE'16), Shanghai, China, July 17-19, 2016.
13. Pereyra, Jose, Mostafavi, Ali, and He, Xudong, "Multi-Agent Framework for Complex Adaptive Modeling of Interdependent Critical Infrastructure Systems", *Proc. of the 2016 Construction Research Congress* (CRC'16), San Juan, Puerto Rico, May 31 – June 2, 2016.
14. Xiao, Fangxiong, Min, Huaqing, Bo, Xu and He, Xudong, "A Formal Modeling Method for Cost-Aware Service Composition", *Proc. of the 25th International Conference on Software Engineering and Data Engineering* (SEDE'15), San Diego, CA, Oct. 12-14, 2015.
15. Liu, Su and He, Xudong "PIPE+Verifier - A Tool for Analyzing High Level Petri Nets", *Proc. of the 27th International Conference on Software Engineering and Knowledge Engineering* (SEKE15), Pittsburgh, July 6-8, 2015.
16. Mezura-Montes, E., "Evolutionary computation. a unified approach," *Artificial Life*, 13(4):423–426, Oct 2007.
17. Karsai, G.; Sztipanovits, J.; Ledeczi, A.; Bapty, T., "Model-integrated development of embedded software," *Proceedings of the IEEE*, Volume: 91, Issue: 1, Jan. 2003, pp.145 – 164.

18. Huang, D., Sarjoughian, H.S. " Software and Simulation Modeling for Real-time Software-intensive System", The 8th IEEE International Symposium on Distributed Simulation and Real Time Applications, pp. 196-203, Oct., Budapest, Hungary.
19. Sztipanovits, Janos, Karsai, Gabor, Neema, Sandeep, Nine, Harmon, Porter, Joseph, Thibodeaux, Ryan, and Volgyesi, Peter, "Towards a Model-based Toolchain for the High-Confidence Design of Embedded Systems," *Work-in-Progress Workshop at the Real-Time Application Systems conference* (2008)
20. Lee, E. A., "Cyber-Physical Systems - Are Computing Foundations Adequate?" *NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap,* (2006).
21. Henzinger, Thomas A. Horowitz, Benjamin and Kirsch, Christoph Meyer, "Giotto: A Timetriggered Language for Embedded Programming", *Proceedings of the IEEE*, vol. 91. No.1, pp. 84-99.
22. J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming Heterogeneity - The Ptolemy Approach," Proceedings of the IEEE, vol. 91, no. 2, pp. 127–144, 2003.
23. OMG. Model-driven architecture. [Online]. URL: http://www.omg.org/mda/
24. Janos Sztipanovits and G. Karsai, "Model-integrated computing," Computer, vol. 30, no. 4, pp. 110–111, 1997.
25. Neuman, "Challenges in security for cyber-physical systems," *DHS: S&T Workshop on Future Directions in Cyber-physical Systems Security*, (2009).
26. Chu, W. C., Lu, C.-W., Yang, H., and He, X., "A formal approach for component retrieval and integration analysis," *Journal of Software Maintenance*, 12(6):325–342, 2000.
27. Drusinsky, D. "The Temporal Rover and the ATG Rover," In *Proceedings of SPIN00: SPIN Model Checking and Software Verification, Stanford, California, USA*, volume 1885 of *Lecture Notes in Computer Science*, pages 323–330, 2000.
28. Thati, P. and Rosu, G., "Monitoring algorithms for metric temporal logic specifications," *Electrontronic Notes Theoretical Computer Science*, 113:145–162, Jan. 2005.
29. Zhang, M. and Yin, H., "Efficient, context-aware privacy leakage confinement for android applications without firmware modding," *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 259–270, New York, NY, USA, 2014. ACM.
30. Dong, Z., Fu, Y., and He, X., "Automated runtime validation of software architecture design," *Proceedings of Second International Conference of Distributed Computing and Internet Technology (ICDCIT 2005)*, volume 3816 of *Lecture Notes in Computer Science*, pages 446–457. Springer, 2005.
31. Bauer, A. Kuster, J.-C., and Vegliach, G., "Runtime verification meets android security," In A. Goodloe and S. Person, editors, *NASA Formal Methods*, volume 7226 of *Lecture Notes in Computer Science*, pages 174–180. Springer Berlin Heidelberg, 2012.
32. Falcone, Y., Currea, S., and Jaber, M., "Runtime verification and enforcement for android applications with rv-droid," In S. Qadeer and S. Tasiran, editors, *Runtime Verification*, volume 7687 of *Lecture Notes in Computer Science*, pages 88–95. Springer Berlin Heidelberg, 2013.
33. Huang, J., Erdogan, C., Zhang, Y., Moore, B., Luo, Q., Sundaresan, A., and Rosu, G. Rosrv: "Runtime verification for robots," In *Proceedings of the 14th International Conference on Runtime Verification*. LNCS, September 2014.
34. Bahl, S. and Chaturvedi, M. M., "Article: Literature review of mobile applications testing on cloud from information security perspective," *International Journal of Computer Applications*, 79(14):15–23, October 2013.
35. Parker, Lynne E., "Adaptive Heterogeneous Multi-Robot Teams", *Proceedings of Neural Networks and Their Applications (NEURAP '98)*, March 1998, pg. 87-94.
36. Akselrod, D., Sinha, A., and Kirubarajan, T., "Hierarchical markov decision processes based distributed data fusion and collaborative sensor management for multitarget multisensor tracking applications. *In the IEEE International Conference on Systems, Man and Cybernetics*, pages 157 – 164, 2007.

37. Boivin, E., Desbiens, A., and Gagnon, E., “UAV collision avoidance using cooperative predictive control,” *The 16th Mediterranean Conference on Control and Automation*, pages 682 – 688, 2008.
38. F. Borrelli, T. Keviczky, and G. Balas. Collision-free uav formation flight using decentralized optimization and invariant sets. *In the 43rd IEEE Conference on Decision and Control*, volume 1, pages 1099 – 1104, 2004.
39. Carls Electronic Kits, “Bioloid Robot User Guide,” URL: http://www.electronickits.com/robot/BioloidUser'sGuide.pdf.
40. Robotics, “Robotics e-mannual,” URL: http://support.robotis.com/en/product/bioloid/gp_kit.htm.
41. Brooks, R. A., "A Robust Layer Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation RA-2*, 14-23, 1986.
42. Brooks, R. A., "Planning is just a way of avoiding figuring out what to do next", Technical report, MIT Artificial Intelligence Laboratory, 1987.
43. Behavior programming – LeJOS. URL: http://www.lejos.org/ev3/docs/lejos/robotics/subsumption/package-summary.html.
44. DMSO, "The DMSO High Level Architecture (HLA), 1996.
45. Adelantado, M. and Siron, P., "Air-ground combat simulation through the ONERA HLA run-time infrastructure." *Proceedings of the Symposium on Military, Government and Aerospace Simulation*, MGA2000, Washington D.C., (April 16-20, 2000), SCS, pp. 191-196.
46. Kuhl, F., Weatherly, R., Dahmann, J., “Creating Computer Simulation Systems: An Introduction to the High Level Architecture,” Prentice Hall, 1999.
47. Peca, M., Sojka, M., and Hanzalek, Z., “Spejbl – The Biped Walking Robot,” *7th IFAC International Conference on Fieldbuses and networks in industrial and embedded systems*, Toulouse: University Toulouse, (2007), pp. 63–70.
48. Reynolds, P.F., Natrajan, A. and Srinivasan, S. "Consistency Maintenance in Multiresolution Simulations." *ACM Transactions on Modeling and Computer Simulation*, Vol. 7, No 3 (July), pp. 368- 392, 1997.
49. Kester, J. E. "An approach to aggregation and disaggregation for J-MASS system models." *Proceedings of the 15th DIS Workshop*, Orlando, FL, (Sept. 1996), pp. 71-78.
50. Vukobratovic, M. Borovac, B. and Surdilovic, D., “Zero moment point—proper interpretation and new application,” *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pp. 237–244, 2001.
51. Hristu-Varsakelis, D. and Levine, W. S. Eds., *Handbook of Networked and Embedded Control Systems*, ser. Control Engineering. Boston: Birkh¨auser, 2005.
52. Henzinger, T., Sifakis, J., “The embedded systems design challenge,” In: FM: Formal Methods. *Lecture Notes in Computer Science 4085*. Springer (2006) 1–15 2.
53. Kopetz, H., Bauer, G., “The time-triggered architecture,” Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software (Oct 2001)
54. Thibodeaux, R., The specification and implementation of a model of computation. Master’s thesis, Vanderbilt University (May 2008)
55. Fu, Y., and Shuvo, M., “An Approach to Analyzing Adaptive Intelligent Vehicle System”, 3rd International conference on Control, Decision, and Information Technologies, Saint Julian’s, Malta, Apr 6-8, 2016
56. Shuvo, M., and Fu, Y., “Sonar Sensor Virtualization for Object Detection and Localization”, IEEE Southeast Conference, Norfolk, VA, Mar 30, 2016.
57. Fu, Y. and Choosilp, W., “Specification based testing of Android System”, International Journal of Wireless and Mobile Networks. Vol. 11, No. 2017.
58. Fu, Y., Dong, Z. and He, X., “Formal Modeling and Analysis of Collaborative Humanoid Robotics”, International Journal of Robotics Applications and Technologies (IJRAT), 2018 (Accepted).
59. Fu, Y. and Choosilp, W., “Model-based Development of Trustworthy Cyber Physical System”, Proceedings of the IEEE SoutheastCon, Florida West Coast Section-Tampa Bay Area, FL, April 2018. (Accepted)

60. Fu, Y. and Choosilp, W., “Test Driven Development of Cyber Physical System”, International Journal of Cyber-Physical Systems (IJCPS). (Under Review) 2018.
61. Feng Chen, Patrick Meredith, Dongyun Jin and Grigore Rosu, “Efficient Formalism-Independent Monitoring of Parametric Properties”, In Proceedings of the Automatic Software Engineering (ASE’09), IEEE/ACM, pp 383-394, 2009.
62. Oded Maler, Some thoughts on runtime verification. In proceedings of International Conference of Runtime Verification (RV’2016), LNCS, Vol. 10012, pp 3-14, 2016. Springer, Cham.
63. B. Madhevan and M. Sreekumar, Tracking algorithm using leader follower approach for multi robots. Procedia Engineering, Vol. 64, 2013, pp. 1426-1435.
64. Jing Luo, Cheng-Lin Liu, and Fei Liu, A leader-following formation control of multiple mobile robots with obstacle. In proceedings of IEEE international Conference on Information and Automation, Lijiang, China, 2015.
65. Augie Widyotriatmo, Endra Joelianto, Agung Prasdianto, Hafida Bahtiar, and Yul Y. Nazaruddin, Implementation of leader-follower formation control of a team of nonholonomic mobile robots. International Journal of Computers, Communications & Control (IJCCC) 12(6): 871. 2017.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

This section provides a list of symbols, abbreviations, and acronyms used in this report:

Abbreviation	Full Meaning
DOD	Department of Defense
CPS	Cyber Physical System
CMS	Cyber Military System
CORBA	Common Object Request Broker Architecture
GA	Genetic Algorithm
MDE	Model Driven Engineering
MIC	Model Integrated Computing
SCADA	Supervisory Control and Data Acquisition
OMG	Object Management Group
MRS	Multiple Robotics System
UAV	Unmanned Aerial System
UGV	Unmanned Ground System
MIMO	Multiple Input Multiple Output
SIMO	Single Input Multiple Output
LTL	Linear Temporal Logic
HLA	High Level Architecture
OOSD	Object Oriented Software Development
SMT Solver	Satisfiability Modulo Theory Solver
DOF	Degree of Freedom
NED System	North-East-Down Coordinate System
LTP	Local Tangent Plane
SPIN	Simple Promela Interpreter
SAMTool	Software Architecture Modeling Tool