WiPiEC Journal

Works in Progress in Embedded Computing Journal

March 2024 · Volume 10 · Issue 1 · ISSN 2980-7298

https://wipiec.digitalheritage.me

Contents

Works in Progress in Embedded Computing Journal (WiPiEC Journal), Volume 10, Issue 1, MARCH 2024

C Software Formal Verification

Review, challenges and future directions

Fateh Boutekkouk
ReLaCS2 laboratory, University of Oum El Bouaghi
Oum El Bouaghi, Algeria

Abstract—**This paper reviews briefly the literature on formal verification of C software. Most existing C software model checkers and automatic theorem provers deal well only with small size code C software. Furthermore, full mechanization of conventional techniques to reduce the verification process complexity as code summation and abstract interpretation is merely impossible. Another challenge is how to choose the most suitable tool(s) among a panoply of available tools. We think that Artificial Intelligence can mitigate the above problems. For instance, by applying machine learning algorithms, the verification tool can automatically infer properties to be checked and synthesize proofs.**

Keywords-C software; formal verification; Artificial Intelligence

I. INTRODUCTION

The debut of research works on mathematical reasoning about imperative programs goes mainly back to the works of Floyd [13] and Hoare on the logics of axioms (Hoare logic) [17], the works of Dijkstra on weakest preconditions calculus [8, 9] and abstract interpretation [6]. In the same context, we find other former works that tried to formalize and check imperative programs using type systems [11] and algebraic semantics [14]. Since that, many extensions and logics have been developed to reason about arrays, complex and dynamic typed data structures, unbounded loops, floating-point arithmetic, recursive functions, and concurrency. Separation logic [28], matching logic [35], dependent type theory [36], and refinement types [15] are among such extensions. Remarkable advances in SMT solvers technology have enhanced the automation level of both theorem provers and model checkers. Actual theorem provers and model checkers use SMT solvers as backend helpers and other frontend tools to reduce and simplify the verification process. Recent works indicate that the verification of low-level systems code as OS kernel has become tractable.

As it is known, the C language is still very popular programing language due to its great flexibility in terms of data representation and pointers arithmetic. C has been used to implement operating systems kernels and embedded systems. The majority of software in embedded systems is still written in C. C code can also be used to automatically generate a HDL (Hardware Description Language) code, which will be used later in embedded system hardware part synthesis. However, C is weak-typed (i.e. C's types provide no invariants about data values) and sometimes ambiguous. C standard [19] defines the C memory model as a sequence of bytes (i.e. untyped memory model) and underspecifies the semantics of the C language. Furthermore, the uncontrolled use of I/O library functions can easily create security vulnerabilities. In order to minimize the number of bugs in C code, some solutions emphasize what we call standards-based development. These standards (example MISRA) impose a set of obligations and constraints on coding. For instance, the non-use of side effect statements or pointers [29]. Despite, this solution seems useful, it constraints the programmer creativity and minimizes optimization opportunities of the C code. In addition, most of these obligations are just guidelines that lack systematization and automatization. Producing correct imperative code can be the fruit of the correct-by-construction design approach. In this approach, code can be automatically synthesized through a sequence of refinements of an abstract formal specification. Each refinement must be proved correct with respect to the previous one. Consequently, the generated code implements correctly its specification. The method B follows this approach. In contrast to functional programs, imperative programs proving is more challenging. Indeed, it is not obvious, whether the well-known Curry-Howard correspondence [16] which links a functional program to its equivalent logical proof system can be naturally applied for imperative programs. Imperative languages include some uncommon constructors for mathematical logics as pointers, global variables, and so on. This makes reasoning about imperative programs in general a non-trivial task.

On the other side, Artificial Intelligence (AI) is becoming more attractive since it can offer some powerful tools to boost the formal verification process. Our aim through this paper is first to review shortly the literature, then to define the main challenges and finally to shine the spotlight on some promising future directions in particular the synergy between software formal verification and AI. This paper is structured as follows: Section 2 is devoted to the state of the art on C software formal verification. In this context, we present a set of criteria to compare between existing approaches and tools. In section 3, we pass quickly on the main challenges and in section 4 we discuss some possible future directions before concluding.

II. LITERATURE REVIEW

The first initiative in the C language formalization returned back to the work of Sethi [31] where a denotational semantics of a subset of the ANSI C language was proposed. Despite, this work was incomplete with respect to the ANSI standard, it gave a big push to subsequent researchers to investigate more in this topic. The literature on formal verification of C software is very rich, however, we can cite some pertinent works as in [1, 3, 5, 7, 18, 21, 26, 27, 30, 34, 35], and some interesting PhD and master thesis on the same topic as in [10, 20, 22, 25, 33]. Many code C formal verification tools exist. For a fair evaluation of such tools, Competition on Software Verification (SV-COMP) has been established [4]. The first edition was in 2012 and the last one in 2023. The competition includes 23 805 verification tasks for C programs to check four properties that are reachability, memory safety, overflows, and termination. The evaluation is performed based on a scoring schema that assigns points in function of the type of the reported result (unknown, false correct, false incorrect, true correct, true incorrect) for the given property. We can however classify these works according to a set of pertinent criteria. In this context, we propose a taxonomy based on ten criteria. These criteria include the application domain, the orientation of the software, the supported C language, the C memory model used, the intermediate representation, the formalization method, the logics used in the proofs, the verification method, the reduction technique, and the type of checked properties.

- The application domain, which can be general-purpose, compilers, cryptography, OS kernels, device drivers, hypervisors, embedded systems and robotics. It is important to recognize the application domain in order to choose the more appropriate formalism and verification technique. For example, device drivers and operating systems code uses pointers as first class and the code usually contain some fragments written in assembly code. In this case, the mathematical proofs have to formalize in addition to C code, the assembly code too. The type of properties to be proved may also dependent on the domain of application. For instance, C code that implement multi-tasks OS kernels have to guarantee the mutual exclusion and isolation properties and so on.

- The orientation of the software, which can be control-oriented, data-oriented or mixt. A typical control-oriented C software is composed of control statements (e.g. if else, or switch) operating on very small-sized data. On the other hand, data-oriented C software is composed of complex operations or treatments on large-sized data. Model checking is more suitable for control-oriented software with simple properties and theorem proving for data-oriented software with complex properties.

- The supported C language which can be the full ANSI standard, a subset of the standard (i.e. the full standard excluding some constructors or features), or a specific C sublanguage such as C0 and CoreC*.

- The C memory model (i.e. the heap model) which, can be un-typed (i.e. raw arrays of bytes), typed or hybrid. The untyped model adds significant annotation burden, and render the reasoning computationally expensive. The typed model however, offers a reasonable abstraction level for verification.

- The intermediate representation of C code, which can be a restricted subset of the C language itself as CIL, LLVM-IR, an abstract model as the CFA (Control Flow Automaton), or an intermediate formal language as Simpl and Boogie. Compared to the original C code, the intermediate representation generally has fewer constructs and unambiguous syntax, which make formal verification easier.

- The formalization approach, which can be annotation-based approach, semantics-based approach, transformational approach, reverse engineering approach and the cooperation approach.

In the annotation-based approach, the original C source code is annotated by specification constructs. These logical annotations may specify functions pre-conditions and post-conditions, loop and type invariants, assertions and so on. From these annotations, verification conditions (VC) or obligations proofs are generated automatically using Hoare-style weakest precondition method and checked using an automatic or interactive theorem prover. Annotations can be burdensome for programmers especially if these annotations are expressed in an unfamiliar formal specification language. In order to overcome this issue, the C language was extended to support Design-by-contract paradigm giving the birth to ACSL (The ANSI/ISO C Specification Language). In the semantics-based approach, the semantics (i.e. operational or denotational semantics with possibly categories definition) of the C language or a substantial subset of it is explicitly defined in some formal specification language. The properties to be checked are also expressed in the same formal language. In the transformational approach, the source code is transformed either directly to another formal specification written in a certain formal language (e.g. transformation of C imperative code to a purely functional code in ML as done by the Why tool into the Coq assistant prover) or to an abstraction (i.e. predicate abstraction) of the original code in the same language (i.e. C). In the reverse engineering approach, the C source code is usually reversed automatically to a formal or a semi-formal model using UML. Then some formal checking is applied on this UML model to prove or refute the desired properties. The cooperation approach is any feasible sequential or concurrent combination of the above approaches.

- The logics and theories of the formal system and proofs. Those include Hoare, separation, rewriting and temporal logics, dependent and refinement types and category theory.

- The verification method that can be symbolic execution with its variants (static, dynamic), theorem proving with its variants (fully automatic, interactive proof assistants), model-checking with its variants to verify larger programs (with complex loops) or programs with infinite states (i.e. symbolic model checking, abstract model checking, bounded model checking), SAT/SMT solving, or any feasible combination of them. For instance, symbolic execution often calls SAT/SMT solvers. Theorem provers, even model checkers may call SAT/SMT solvers to increase the automation level. A combination of model checkers and theorem provers is also possible. For example, in the Counter Example Guided Refinement approach, first the source program is abstracted away using predicate abstraction technique. A model checker working on this abstraction may check a certain property. If the property is not true, the model checker gives a counter example. In this case, a refinement step will be triggered during which a theorem prover can be called to check whether this counterexample reflects a true error in the program or just a spurious one due to abstraction. The theorem prover can call in turn a SAT or SMT solver to prove the satisfiability of a certain condition.

- The reduction technique used to reduce the complexity of the formal verification. Among these techniques, we find abstraction and program slicing. Abstract interpretation and predicate abstraction are the two common techniques used.
- The checked properties can be simple or complex including the functional correctness, the termination, reachability properties, safety and security properties. Safety may include static safety (i.e. type safety) or dynamic safety (i.e. memory safety). Security includes mainly confidentiality, integrity, and availability properties.

III. CHALLENGES

Despite the big efforts spent in boosting software formal verification process (e.g. exploring parallel and distributed formal verification, abstraction, modular and verification reuse), one can state that software formal verification in its current form cannot meet the needs of industrial sized C software in terms of performance, accuracy and scalability. With the ever increasing in the complexity of software functionalities and non-functional requirements, most state of the art and practice tools fail to formally prove the functional correctness in addition to non-functional properties as safety and security. Most users are unfamiliar with formal techniques and often find them hard to write formal specifications or proofs and even to use tools in particular theorem provers. Furthermore, the majority of available tools do not provide explanations in the case of proof failure and in the presence of a panoply of tools; the user is not able to choose the most suitable verification approaches and tools. The choice is a tradeoff between a set of conflictual criteria such as the amount of annotation effort, the automation level, the performance, the accuracy of results but more interestingly the soundness and the completeness of the proofs system.

IV. FUTURE DIRECTIONS

In order to increase the credibility of existing formal verification tools for large C software, researchers tend to integrate some powerful promising technologies in particular AI, data mining, and quantum computing.

A. Artificial Intelligence and data mining

The idea of leveraging AI and data mining in formal verification has been attracted many researchers [2, 12, 24, 32]. In the context of software formal verification, we can apply AI with many flavors:

1. Automatic synthesis of formal proofs using machine learning algorithms.

2. Automatic inference of theorems and assertions as loop invariants and discovering pertinent properties for verification automatically.

3. Interactive aid of users to select the most suitable abstraction technique and the abstraction level.

4. Interactive assistance of users to choose the most appropriate formal approaches and tools using MCDM methods and tools integration.

5. Interactive support of users to select the most important parts in the software requiring formal verification and properties for checking since it is not feasible to formally verify the entire large software against all properties.

6. Using AI optimization meta-heuristics as genetic algorithms for example to guide the search process in model checking.

7. Integration of explication in the formal verification process and automatic repair of software vulnerabilities.

8. If the software code is supported with some informal specification expressed in natural language, NLP methods can be used to automatically or semi-automatically generate a formal specification and test cases. The latter can be used to complement the formal verification. Testing remains an efficient technique to discover compiler and hardware bugs.

9. AI can be used to automatically restructure the code software following the standards-based development approach to simplify the formal verification.

10. Benchmarking the C software formal verification processes, reuse and sharing the knowledge.

B. Quantum computing

Quantum computing emerged as a very powerful technology inspired from mechanics quantum theory. Due to the superposition and entanglement principles, researchers expect super polynomial speedup for big algorithms including formal verification algorithms. This paradigm however, still needs special algorithms to reduce noise because they do not have enough qubits to execute quantum error correction [23].

V. CONCLUSION

C software formal verification is hot research topic and a grand practical challenge. We can observe that C software formal verification has evolved over decades starting from former works focusing on the definition of a formal semantic of the C language, the use of Hoare logic and automatic theorem provers ending by the use of more expressive logics and mathematical theories such as separation logic and refined types and the usage of model checkers and SMT solvers. Unfortunately, most existing approaches and tools suffer from many obstacles prevent them from being widespread in the industry. Finally, most researchers and experts have emphasized on formal verification process rethinking by making it AI-powered to boost the performance and the accuracy and enables tools integration and scalability. As short-term perspective, we plan to apply machine learning and in particular deep learning to automatically infer loops invariants and properties to be checked in a C program with nested loops and recursive functions.

REFERENCES

[1] J. Amilon, C. Lidström, and D. Gurov, "Deductive Verification Based Abstraction for Software Model Checking," Leveraging Applications of Formal Methods, Verification and Validation. Verification Principles, 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022.

[2] M. Amrani, L. Lucio, and A. Bibal, "ML + FV = $\heartsuit$? A Survey on the Application of Machine Learning to Formal Verification," arXiv: Software Engineering, 2018.

[3] T. Ball, R. Majumdar, T. Millstein, and S.K. Rajamani, "Automatic Predicate Abstraction of C Programs," in PLDI '01: Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation, pp. 203–213, 2001.

[4] D. Beyer, "Competition on Software Verification and Witness Validation: SV-COMP 2023," in: Sankaranarayanan, S., Sharygina, N. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2023. Lecture Notes in Computer Science, vol 13994, 2023.

[5] F. Boutekkouk, "Towards Automatic Maude Specifications Generation From C Functions," Journal of Innovation Information Technology and Application (JINITA), vol. 5(1), pp. 83–96, 2023.

[6] P. Cousot, and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 238—252, 1977.

[7] P. Cuoq, F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, and B. Yakobowski, "Frama-C A Software Analysis Perspective," Formal Aspects of Computing, 2012.

[8] E. W. Dijkstra, "A constructive approach to the problem of program correctness," BIT Numerical Mathematics, vol. 8(3), pp.174-186, 1968.

[9] E.W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," Commun. ACM, vol. 18, pp. 453–457, 1975.

[10] C.M. Ellison, "A Formal Semantics of C with Applications," PhD. thesis, University of Illinois, 2012.

[11] J.-C. Filliatre, "Preuve de programmes impératifs en théorie des types," Thèse de doctorat, Université Paris-Sud, 1999.

[12] E. First and Y. Brun, "Diversity-Driven Automated Formal Verification," 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), 2022.

[13] R.W. Floyd, "Assigning meanings to programs," Proceedings of the American Mathematical Society Symposia on Applied Mathematics, vol. 19, pp. 19–31, 1967.

[14] J.A. Goguen and G. Malcolm, Algebraic Semantics of Imperative Programs (Book), MIT Press, ISBN: 9780262071727, 1996.

[15] S. Hayashi, "Logic of refinement types," in Proceedings of the Workshop on Types for Proofs and Programs, pp. 157–172, 1993.

[16] W. A Howard, "The formulae-as-types notion of construction," in Seldin, Jonathan P.; Hindley, J. Roger (eds.), To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, pp. 479–490, ISBN 978-0-12-349050-6, 1980.

[17] C.A.R. Hoare, "An axiomatic basis for computer programming," Communications of the ACM, vol. 12(10), pp. 576–580, 1969.

[18] F. Ivancic, I. Shlyakhter, A. Gupta, M.K. Ganai, V. Kahlon, C. Wang, and Z. Yang, "Model Checking C Programs Using F-SOFT," International Conference on Computer Design 31 October, San Jose, CA, USA, 2005.

[19] ISO/IEC 9899:2018, Information technology — Programming languages — C, https://www.iso.org/standard/74528.html

[20] K. Jiang, "Model Checking C Programs by Translating C to Promela," Master. thesis, Linkoping University, Sweden, 2009.

[21] E. Kamburjan and N. Wasser, "The Right Kind of Non-Determinism: Using Concurrency to Verify C Programs with Underspecified Semantics," in 15th Interaction and Concurrency Experience (ICE 2022), EPTCS 365, pp. 1–16, 2022.

[22] R.J. Krebbers, "The C standard formalized in Coq," PhD. thesis, Radboud University Nijmegen, 2015.

[23] J. Larkin and D. Justice, "Achieving the Quantum Advantage in Software," Carnegie Mellon University, Software Engineering Institute's Insights (blog), Accessed November 8, 2023, https://insights.sei.cmu.edu/blog/achieving-the-quantum-advantage-in-software/.

[24] N. Ge, M. Pantel, and X. Crégut, "Automated Failure Analysis in Model Checking based on Data Mining," 4th International Conference On Model and Data Engineering, Larnaca, Cyprus, pp.13-28, ⟨10.1007/978-3-319-11587-0_4⟩. ⟨hal-03252269⟩, 2014.

[25] M. Norrish, "C Formalised in HOL," PhD. thesis, University of Cambridge, 1998.

[26] S.H. Park, R. Pai, and T. Melham, "A Formal CHERI-C Semantics for Verification," in: Sankaranarayanan, S., Sharygina, N. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2023. Lecture Notes in Computer Science, vol. 13993. Springer, Cham, 2023.

[27] C. Pulte, D.C. Makwana, T. Sewell, K. Memarian, P. Sewell, and N. Krishnaswami, "CN: Verifying systems C code with separation-logic refinement types," Proceedings of the ACM on Programming Languages, 7(POPL), pp.1-32, 2023.

[28] J. C. Reynolds, "Separation Logic: A Logic for Shared Mutable Data Structures," in Proceedings 17th Annual IEEE Symposium on Logic in Computer Science 22-25 July, 2022.

[29] M. Richardson, "Why you should use standards-based development practices (even if you don't have to)," https://www.embedded.com/ June 8, 2020.

[30] M. Sammler, R. Lepigre, and R. Krebbers, "RefjnedC: Automating the Foundational Verifjcation of C Code with Refjned Ownership Types," in PLDI '21, Canada, 2021.

[31] R. Sethi, "A Case Study in Specifying the Semantics of a Programming Language," Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages, pp.117–130, 1980.

[32] T. Sharma, M. Kechagia, S. Georgiou, R. Tiwari, and F. Sarro, "A Survey on Machine Learning Techniques for Source Code Analysis," ArXiv, abs/2110.09610, 2021.

[33] N. Schirmer, "Verification of Sequential Imperative Programs in Isabelle/HOL," PhD. thesis, Technische Universitat Munchen, 2005.

[34] S. Sriya, L. Lavanya, M.M. Aditi, and N.S. Kumar, "Verification of C Programs using Annotations," in 2019 IEEE Tenth International Conference on Technology for Education (T4E), Goa, India, 2019.

[35] A. Stefanescu, "MatchC: A Matching Logic Reachability Verifier Using the K Framework," in Electronic Notes in Theoretical Computer Science vol. 304, pp. 183–198, 2014.

[36] H. Xi, "Dependent types in practical programming," PhD. thesis, Department Computer Science, Carnegie-Mellon University, 1998.

Works in Progress in Embedded Computing Journal (WiPiEC Journal), Volume 10, Issue 1, MARCH 2024

MATLAB Interface for Blood Pressure Determination from Oscillometric Data

Luka Mićović

University of Montenegro, Faculty of Electrical Engineering,

Podgorica, Montenegro
lukamicovic22@gmail.com

Abstract—**The accurate measurement of blood pressure is a fundamental aspect of healthcare, serving as a vital diagnostic tool for assessing an individual's cardiovascular health and overall well-being. Traditional methods of blood pressure measurement involve the use of cuff-based devices, such as sphygmomanometers, which rely on the auscultatory technique. While this method has been the gold standard for decades, recent advancements in technology have paved the way for more sophisticated and convenient approaches to blood pressure determination. Blood pressure determination involves several steps to extract meaningful information from the acquired signals. From the detected systolic and diastolic peaks, we calculated several key blood pressure-related parameters. This included determining the maximum value of the diastolic valleys and its associated time point, which provided essential information for calculating the Mean Arterial Pressure. Utilizing the computed thresholds and peak indices, we identified the cuff pressure values corresponding to the systolic and diastolic peaks. These values represented the Systolic Blood Pressure and Diastolic Blood Pressure, respectively.**

Keywords-blood pressure determination; sphygmomanometer; LabQuest Mini; MATLAB;

I. INTRODUCTION

The accurate measurement of blood pressure is a fundamental aspect of healthcare, serving as a vital diagnostic tool for assessing an individual's cardiovascular health and overall well-being. Traditional methods of blood pressure measurement involve the use of cuff-based devices, such as sphygmomanometers, which rely on the auscultatory technique. While this method has been the gold standard for decades, recent advancements in technology have paved the way for more sophisticated and convenient approaches to blood pressure determination.

One such innovative approach is the utilization of MATLAB, a powerful computational software environment widely employed in various scientific and engineering disciplines. In recent years, MATLAB has emerged as a versatile platform for processing and analyzing medical data, including the interpretation of oscillometric data for blood pressure estimation. This paper explores the development and implementation of a MATLAB interface tailored to blood pressure determination from oscillometric data, showcasing the potential for enhanced accuracy and efficiency.

The oscillometric method, which relies on the measurement of pressure fluctuations in an inflatable cuff, presents a non-invasive and user-friendly alternative to the traditional auscultatory technique. However, accurate blood pressure estimation from oscillometric data involves complex signal processing, calibration, and interpretation, often necessitating the use of specialized software tools. MATLAB's extensive computational capabilities, coupled with its user-friendly interface, make it an ideal candidate for creating an accessible and customizable solution for blood pressure determination.

This paper begins by providing a comprehensive overview of the oscillometric method, its principles, and we delve into the development of a MATLAB-based interface designed to process oscillometric data efficiently and accurately. The interface's functionality includes data acquisition, signal processing, calibration, and the generation of clinically relevant blood pressure readings.

Furthermore, this paper aims to demonstrate the potential applications and benefits of our MATLAB interface in both clinical and research settings. By facilitating the automation of blood pressure determination and improving the accuracy of results, this technology has the potential to enhance the quality of healthcare delivery and advance our understanding of cardiovascular health.

II. METHOD

The system consists of hardware and software, Fig. 1. The sensor is in fact Vernier Blood Pressure Sensor used to measure systemic arterial blood pressure (non-invasively). LabQuest Mini, manufactured by Vernier, was used for data logging. It is possible to use any newer LabQuest Mini data-collection interface.

The principal architecture of the system is shown in Fig. 1.

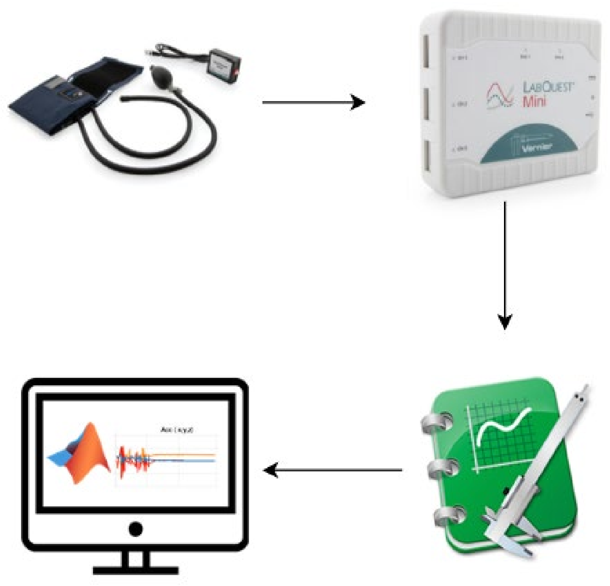

Figure 1. The principal architecture of Blood Pressure Measurement, based on Sphygmomanometer, LabQuest Mini sensor interface and MATLAB

A. Blood pressure determination from accelerometric data

Blood pressure determination from oscillometric data involves several steps to extract meaningful information from the acquired signals.

In the proposed algorithm, data acquisition and preprocessing starts with the importation of the raw data from a CSV file containing time values and cuff pressure measurements. These data were read into MATLAB.

To identify relevant features in the oscillometric cuff pressure signal, we employed peak detection algorithms. The primary objective was to locate the systolic and diastolic peaks within the signal. This involved finding points in the signal where the cuff pressure exhibited significant variations indicative of the cardiac cycle.

Initially, we detected the systolic peak, which corresponds to the maximum cuff pressure value during systole, representing the peak of the cardiac cycle. The exact time at which the systolic peak occurred was recorded.

Similarly, the end of the cardiac cycle was determined by finding the last recorded time point in the dataset. Subsequently, we calculated the indices within the dataset corresponding to the duration of a complete cardiac cycle. The cuff pressure values within this interval represented one full cardiac cycle.

To enhance the quality of the cuff pressure signal, we applied a high-pass Butterworth filter. This filtering process involved removing lower-frequency components, leaving only the relevant high-frequency variations.

Following the signal filtering step, we proceeded to detect both systolic and diastolic peaks within the positive filtered cuff pressure signal. Diastolic valleys, representing the lowest points in the cuff pressure signal during diastole, were detected by inverting the positive filtered signal and applying the same peak detection algorithm.

From the detected systolic and diastolic peaks, we calculated several key blood pressure-related parameters. This included determining the maximum value of the diastolic valleys and its associated time point, which provided essential information for calculating the Mean Arterial Pressure. Utilizing the computed thresholds and peak indices, we identified the cuff pressure values corresponding to the systolic and diastolic peaks. These values represented the Systolic Blood Pressure and Diastolic Blood Pressure, respectively.

The visual representation of the algorithm is shown in Fig. 2.

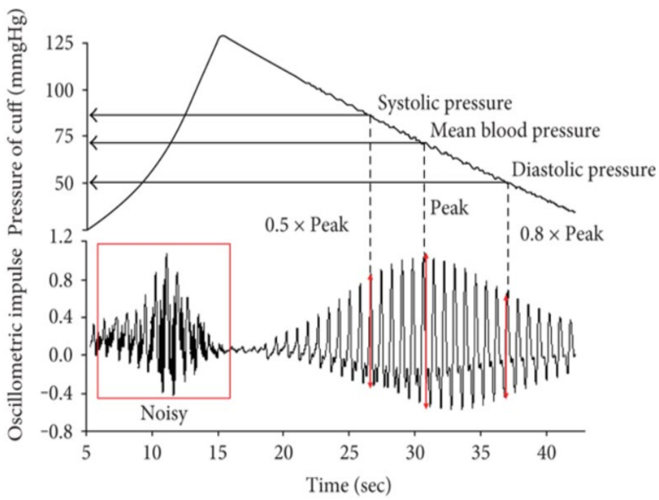

Figure 2. Cuff pressure waveform of oscillometric method [3]

III. Experiment and Results

To evaluate the effectiveness of the MATLAB-based interface for blood pressure determination from oscillometric data, we conducted experiments involving four individuals. The study aimed to assess the accuracy and reliability of the algorithm in estimating blood pressure parameters, including Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), and Mean Arterial Pressure (MAP). To validate the algorithm's performance, we will simultaneously compare the results with the ones obtained using Vernier Logger Lite software.

The algorithm successfully processed the oscillometric data and provided estimations of blood pressure parameters for each participant. To facilitate a comprehensive comparison, Table 1

below presents the algorithm's results alongside measurements obtained using the Vernier blood pressure measuring software:

Person	Blood pressure in mmHg					
	MAP-Vernier	MAP-Matlab	SBP-Vernier	SBP-Matlab	DBP-Vernier	DBP-Matlab
#1	98	97.83	124	128.74	71	76.75
#2	87	96.88	119	119.26	69	71.63
#3	79	97.76	110	111.68	66	74.77
#4	88	88.14	119	117	62	66.51

Table 1

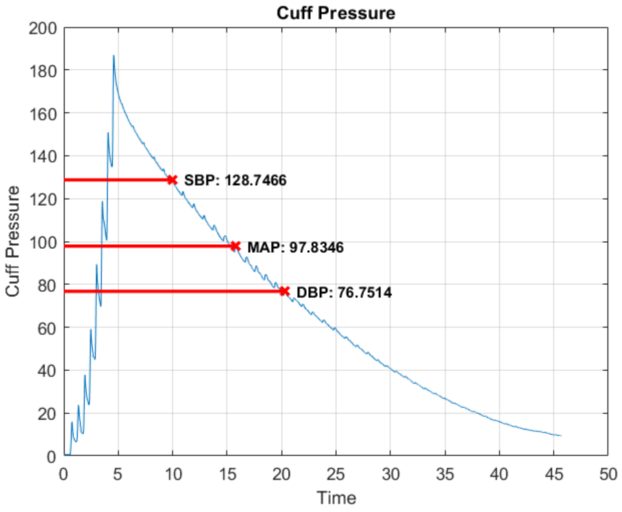

Figure 3

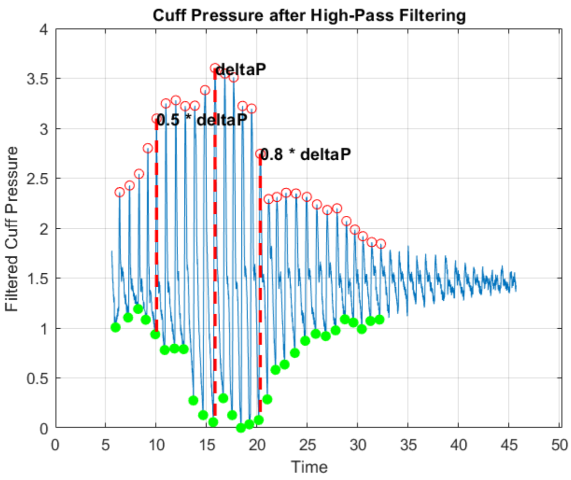

Figure 4

All the relevant information and graphs can be found in Fig. 2, which we obtained from MATLAB, and from there, we can read the relevant values.

IV. Conclusions

This paper presents an examination of the effectiveness of our MATLAB-based interface for blood pressure determination from oscillometric data. The noteworthy alignment between the algorithm's estimations and measurements obtained using the Vernier blood pressure measuring device serves as compelling evidence of its accuracy and reliability. Our interface emerges as a promising tool for blood pressure assessment in clinical and research contexts, offering the advantages of automation and efficiency while maintaining a high level of precision and consistency in blood pressure parameter estimation.

The complete MATLAB code, as well as the images used in the display, can be found at the following link:

https://github.com/UsernamekaLu/BPSAnalysis.git

Acknowledgment

I would like to express my gratitude to prof. dr Radovan Stojanović for his guidance and mentorship throughout this research project.

References

[1] Verrij, E, van Montfrans, G., Bos, J-W. (2008, December) Reintroduction of Riva-Rocci measurements to determine systolic blood pressure?

[2] https://www.vernier.com/files/manuals/bps-bta/bps-bta.pdf

[3] Kuo, C. H. et al. “Development of a Blood Pressure Measurement Instrument with Active Cuff Pressure Control Schemes.” Journal of Healthcare Engineering 2017 (2017)

Works in Progress in Embedded Computing Journal (WiPiEC Journal), Volume 10, Issue 1, MARCH 2024

Implementation of the Project Approach

Anisimov Andrei Vladimirovich
Electrotechnical University "LETI" SPGETU "LETI"
Saint Petersburg, Russia
e-mail address: avanisimov@yandex.ru

Abstract—**Automation is based on a structural approach. An overview of modern approaches used in the implementation of tasks, including modeling tasks. Modeling business processes using CASE tools. Formation of skills to solve professional problems. Methodology and technology of canonical design. Standardization and professional consortia. Design based on notation standards. Working with resources in conditions of heterogeneity of formats and interpretation of meaning in accordance with the characteristics of resources.**

Keywords-automation of design; design methods and tools; modeling of business processes; CASE-funds; pre-project survey; engineering of the design object; system analysis; standardization; professional consortia.

I. Introduction

Design automation in the framework of training provides for the development of methods of process analysis necessary for design; obtaining knowledge about processes. Acquisition of special knowledge and skills necessary to participate in the design, according to the composition, content and principles of the organization of information support used in design tasks. Using the acquired knowledge and skills to participate in specific practical development activities. The proposed material contains an overview of modern approaches used in the implementation of the tasks set, including modeling tasks. The field of application of modeling relates to knowledge management. The tools are based on the fundamental approaches of the subject area.

II. Design methods and tools

Training from the perspective of design automation tools [1] should include the development of business process analysis methods. The acquisition of knowledge about information processes is based on the acquisition of special knowledge and skills necessary to participate in the design. As a result, the use of acquired knowledge and skills ensures participation in specific practical activities. It is essential to study the technology of project management. Technical means that ensure constant monitoring of the progress of organizational activities, the concentration of resources on solving specific tasks by the main existing ones in terms of composition, content and principles of organizing information support. The design is based on the choice of methods for modeling systems, structuring and analyzing the purpose and function of systems, and conducting a system analysis of the applied field. At the same time, it is necessary to study effective algorithms using modern technologies. During the training, it is necessary to formulate and formalize design tasks, conduct modeling of business processes and data using CASE tools. As well as choosing methods for modeling systems, structuring and analyzing the goals and functions of systems, and conducting a system analysis. Solving design problems using formulated technical and economic requirements. Design and technological activities are assumed to be the main type of professional activity. At the same time, knowledge and skills should also be linked to design and research activities.

III. Training and objects of professional activity

Design automation as methods and methods of designing, debugging, production and operation of information technologies and systems is used in the fields of mechanical engineering, instrumentation, science, education, administrative management, business, management, etc., and is implemented in relation to enterprises of various profiles and all types of activities in the context of the information society economy. In relation to design and technological activity, the knowledge and skills acquired by students are oriented towards individual tasks within the framework of design and engineering activities.

The content of the discipline is focused on the formation of skills to solve the following professional tasks in accordance with the chosen main type of professional activity. The design of basic and applied design technologies involves the development of implementation tools. In particular, the development of computer-aided design tools provides for a pre-design inspection (engineering) of the design object, a system analysis of relationships. After that, technical design (reengineering), operational design, selection of source data for design, modeling of processes and systems are possible. A special role in the design is played by the calculation of ensuring safety conditions. The result is accompanied by the calculation of economic efficiency and the development, approval and release of all types of project documentation.

IV. Methodology and technology of canonical design

The concept of canonical design includes the composition of the stages and stages of canonical system design. The composition and content of the work at the pre-project stage. Survey methods have a formalized classification based on the classification of methods for collecting survey materials. This also applies to the forms of documents for the formalization of survey materials and the composition and content of work at the design stage, as well as the composition and content of work at the stages of implementation, operation and maintenance of the project. The methodology and technology of the documentation system design allows planning and organizing project activities based on project management standards. The training provides the acquisition of project planning skills using standards in project management. These include organizations for standardization: ISO, IEC, ITU-T. Standardization in the field of information technology is provided by professional consortia such as IEEE, IAB, PMI, IPMA, Regional WOS, OG, ECMA, OMG, X/Open, NMF, OSF, Joint Technical Committee (JointTechnicalCommittee). Although, of course, in modern conditions, the most important source of standards is Rosstandart.

V. The role of the project manager

The project manager carries out planning, scheduling, analysis of their implementation, evaluates results, provides information, manages various organizations, solves complex tasks and allocates resources in order to achieve a predetermined goal. The situation in which the project manager operates requires constant attention from him, since a significant part of his functions differs significantly from traditional ones. The project manager tries to complete difficult work by a certain deadline with limited resources and with the help of people whose main work may not be related to the implementation of this project. In addition, the project manager must have all the knowledge and information necessary to complete the project.

VI. Preparation of the project based on modeling

In modeling, there is a division of entities into classes and objects. The standard notation for modeling real-world objects is UML (Unified Modeling Language).as a first step in the development of an object-oriented program. It describes a single consistent language for defining, visualizing, constructing, and documenting artifacts. Developing a system model is creating a plan. The design is based on UML notation standards. The modeling language allows you to model concepts and underlies the development [2].

VII. Resource approach

Interaction with resources is based on a semantic description. The modern approach to access to resources is based on a semantic description. The task of data mining [3] is based on the annotation of knowledge specific to the selected field. The standard used in the implementation of data access is based on software metadata models. The scope of the model relates to knowledge management. The means of describing the data structure are based on the fundamental approaches of the subject area. Access to resources is based on models. To analyze resources, you need a way to describe their interaction.

The heterogeneity of formats makes it relevant to interpret the meaning of information in accordance with the resource under study. This approach is based on the organization of management, which facilitates the understanding of the description of resources. Technical capabilities for working with knowledge are necessary [4], first of all, for artificial intelligence tasks. Objects containing various types of knowledge are the intellectual space of an organization. From the point of view of the approach to modeling such objects, an approach based on the ontology of the subject area is used. This approach allows us to describe the available knowledge using the advanced mathematical apparatus of artificial intelligence.

Description of knowledge is the main task of this discipline. The methods of representing and describing knowledge in this subject area can be divided into production models and semantic networks. The production model is based on rules for presenting knowledge based on conditions. At the same time, an exemplary suggestion is made for searching the knowledge base. The field of use of such models is industrial expert systems. The semantic network is built on the basis of an oriented graph with vertices representing concepts and arcs establishing relationships. In relation to the task of accessing resources, the semantic description of data sets is based on ontology dictionaries. First of all, we note dictionaries for the annotation of information about the origin. These dictionaries include information about licenses. Dictionaries describing the actual data provide data mining. It is in these dictionaries that data types and data set specifications are presented. In addition, there are ontology dictionaries for the annotation of knowledge.

VIII. Conclusion

The construction of a training system from the perspective of design automation tools is linked to the task of creating and distributing corporate knowledge, and is a continuation of the work [5] and [6]. Model construction and formalization are described using the concept of ontology. The tasks of accessing resources used to work in corporate information systems should be solved taking into account security when organizing remote access [7]. The training is based on regulatory documents and standards in the field of information technology.

References

[1] V.N. Gridin, V.I. Anisimov, S.A. Vasiliev Fault tolerance of fault-tolerant CAD systems based on methods of diacoptics and automation of control of multitenate components, Journal of Radioelectronics ISSN 1684-1719, No11, 2023.

[2] Software engineering. Paradigms, technologies, and case tools. Lavrishcheva E.M., M. Yurait Publishing House, 2017. 280 P.

[3] Data and process analysis: studies. the manual / A. A. Barseghyan, M. S. Kupriyanov, I. I. Kholod, M. D. Tess, S. I. Elizarov. —3rd ed., reprint. and additional — St. Petersburg: BHV-Petersburg,2009. -512 p.: il.

[4] Context-oriented data collection systems focused on use as part of cyber-physical systems / A. Vodiakho [et al.] // International Conference on Soft Computing and Measurements. — 2021.-pp. 248-250.

[5] V.N. Gridin, A.V. Anisimov Compatibility of operating systems // Information technologies and mathematical modeling of systems 2020. Proceedings of the International Scientific and Technical conference. - M.: Federal State Budgetary Institution of Science Center for Information Technologies in Design of the Russian Academy of Sciences. - 2020.-p.- 12-14.

[6] A.V. Anisimov Technology of working with educational resources// Modern education: content, technology, quality. – Proceedings of the International Scientific and Technical conference. – St. Petersburg: St. Petersburg State Electrotechnical University "LETI" — 2021 -pp.39-42.

[7] A.V. Anisimov, A.A. Kuzmitsky Information security in distance learning "Telecommunications" Publishing House: Science and Technology (Moscow) - 2018 No.7 ISSN:1684-2588 pp.41-4.

Supported by:

ISSN 2980-7298