Software Impacts 11 (2022) 100183

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

A modern Blackboard Architecture implementation with external command execution capability

Jeremy Straub

Department of Computer Science, North Dakota State University, 1320 Albrecht Blvd., Room 258, Fargo, ND 58108, USA

ARTICLE INFO

Keywords:
Blackboard Architecture
Expert system
Artificial intelligence
Decision support system
Robotic control
Rule-fact systems

ABSTRACT

A modern implementation of a simplified version of the Blackboard Architecture is presented herein. It uses a rule-fact-action network and a round-robin rule processing engine. It also includes external command actuation capabilities for action nodes, making it highly extensible. The implementation is modular so components (such as the rule processing engine) could be readily replaced, if desired, while utilizing other system functionality. The implementation features a command processor interface, allowing systems to be implemented without the need to write any procedural code. The network can be developed and run using the simplified command processor language created especially for this use.

Code metadata

Current code version	*v1.0*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-150
Permanent link to reproducible capsule	
Legal code license	*Apache License, 2.0*
Code versioning system used	*None*
Software code languages, tools and services used	*C#*
Compilation requirements, operating environments and dependencies	*Visual Studio 2019*
If available, link to developer documentation/manual	
Support email for questions	jeremy.straub@ndsu.edu

1. Introduction

A wide variety of techniques exist for autonomous decision making. Swarm intelligence techniques [1,2] such as Ant Colony Optimization [3] and Quantum Firefly Swarms [4] produce emergent effects from collections of limited scope, limited computational cost decisions which have an emergent effect, in aggregate. Neural networks [5], on the other hand, loosely mirror the human brain's learning mechanism by training to recognize patterns via the optimization of network node weights using supplied correctly classified data. Still other techniques use decision space exploration, model other real-world phenomena and combine various pre-existing techniques.

One technique that has been utilized and expanded for some time is the Blackboard Architecture. Hayes-Roth [6] introduced this technique in 1985, though it is based off of the earlier Hearsay-II system [7]. The Blackboard Architecture is conceptually similar to rule-fact expert systems, which were introduced with the Dendral (in 1965) and Mycin (in the 1970s) systems [8–10]. Unlike typical rule-fact expert systems, which provide recommendations, the Blackboard Architecture adds an actuation capability to the systems.

A wide variety of uses for and enhancements of the basic Blackboard Architecture have been proposed. Its efficacy has been demonstrated for modeling proteins [11], tutoring [12], robotic command [13,14] and vehicle control [15]. Enhancements, such as pruning for speed [16,17], parallel [18] and distributed [19] processing, and message handling and filtering [20] have been developed.

Prior work also included blackboard solving [21,22] for goal-driven Blackboard Architecture implementations, the use of boundary nodes [23] for wide-scale distributed decision making and problem solving and optimization of maintenance [24] for long-running systems that must operate without human intervention.

Despite the long duration of the technique's use and its utility for a variety of applications, implementations of Blackboard Architecture systems are not readily available (the way that, for example, neural

E-mail address: jeremy.straub@ndsu.edu.

https://doi.org/10.1016/j.simpa.2021.100183
Received 25 October 2021; Accepted 21 November 2021

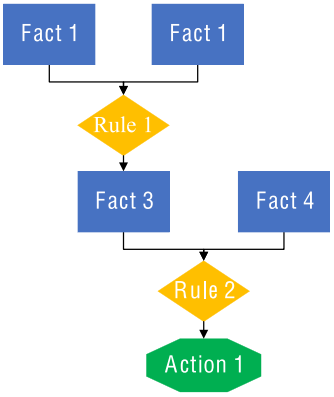

Fig. 1. Blackboard Architecture rule-fact-action network.

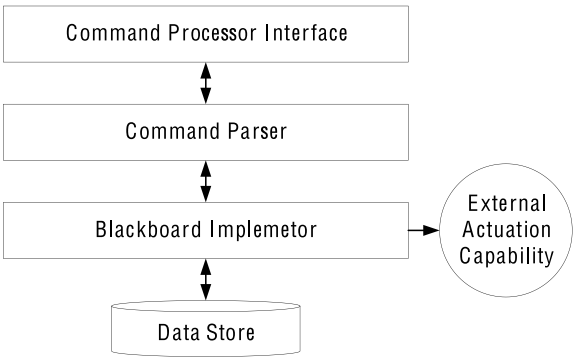

Fig. 2. Blackboard Architecture system architecture.

network implementations are). This article, thus, presents a easy-to-use Blackboard Architecture implementation, which is based on the versions developed and upgraded for [21–25]. The version presented herein includes an interface that has been specifically developed for facilitating the ease of use of the system (using command syntax that is similar to the command language developed for the gradient descent trained expert system presented in [26]). It also includes a generalized actuation capability which allows the system to make system calls (including command parameters).

This article continues with a description of the system, presented in Section 2. In Section 3, the algorithms and system operations are discussed. Sections 4 and 5 discuss the architecture of prior Blackboard Architecture implementations and provide several use case examples, respectively. Then, in Section 6, the advantages and limitations of the system are reviewed. Finally, in Section 7, the use and impact of the system is considered.

2. Software description

The software presented herein is a simplified Blackboard Architecture implementation that has all of the key aspects of a Blackboard Architecture system. While a wide variety of configurations have been proposed (including some which operate two networks concurrently or separate actuation from knowledge storage), many of these could be produced using this system (albeit potentially using multiple instances of it concurrently).

The system described herein stores rules, facts and actions in a single network (as was used for [21–25]). Facts store knowledge which, in this case, are Boolean values. Actions are defined in terms of system calls that the Blackboard Architecture system can make to external executables (making the system inherently extensible). Finally, rules establish the relationships between groups of facts and actions. Each rule is comprised of a collection of facts which are pre-conditions and facts and actions which are post conditions. Rules are able to be run when their pre-conditions are satisfied (and they are actually run when they are selected by the system's rule processing engine to execute). Fig. 1 depicts a basic rule-fact-action network to illustrate these relationships.

Note that, for the purpose of this implementation (to facilitate command brevity and processing speed), a rule can have at most four pre-condition facts and up to four post condition elements (which can either be facts or actions). The Blackboard Architecture implementation's underlying code can support unlimited pre- and post-conditions and the command processor interface could be easily augmented to support this, if it was needed for a particular application that was to operate through the command processor.

System operations begin with the creation of facts. Then actions and rules are created. After this point, the system can be run. Fact-setting and fact-querying commands also exist. Note that this order is merely suggested as a convention and commands can be run in any order. For example, the network could be run and then added to. Listing 1 presents a very simple Blackboard Architecture command set.

3. Algorithm and system operations

The majority of the functionality of a Blackboard Architecture system is provided by the rule-fact-action network that defines its operations. The system described herein, thus, is comprised of four key components, which are depicted in Fig. 2. The command processor interface provides a mechanism for users to interact with the Blackboard Architecture system without having to write any procedural code. It allows the user to enter commands to create rules, facts and actions and to run networks, change fact values and query fact values using simple commands. The interface facilitates saving, loading and executing commands scrips and saving the output results of system operations.

The command processor interface transmits the commands entered by the users to the command parser which determines whether they are properly formatted and complete. Commands that are correct are translated into system calls which are sent to the blackboard implementor module. This module is responsible for the logical operations of the system. It creates and maintains the rule-fact-action network in memory, and includes the rules processing engine and the external actuation capability. This system stores the rule-fact-network and other system operating state details and parameters in memory as a data store.

The logical operations of the simplified Blackboard Architecture are depicted in Fig. 3. As shown, the process begins with the creation of a rule-fact-action network. Notably, the rule-fact-action network can be updated as the system operates; however, some network must exist for other system operating areas to function.

Once a basic rule-fact-action network is created, network additions and updates, fact updates and system runs can be performed on an as-needed basis (though, as a convention, rules, facts and actions that are known to be needed should be created prior to system operations). In Fig. 3, these three key tasks are shown at the top of the figure. Network additions and fact value updates impact only the data store. The remainder of the figure depicts system operations.

System operations are initiated by setting one fact to a value and launching the rule processing engine. Notably, an output fact can be specified and the value for this will be returned at the end of system operations.

System operations involve running the rule processing engine iteratively. During each iteration, all rules are assessed to determine whether they have been previously run and, if not, whether their pre-conditions are satisfied. Rules which have not been previously executed and whose pre-conditions are satisfied are executed in an arbitrary

Listing 1. Blackboard Architecture Demonstration Script.

```
F0001=F:Fact 1
F0002=F:Fact 2
F0003=T:Fact 3
F0004=F:Fact 4
F0005=F:Fact 5
*
A0001:Action 1_____:notepad.exe
*
R0001:F0003+F0004+N0000+N0000>>A0001+F0005+N0000+N0000:Rule 1
*
PR:F0004=T>>F0005
```

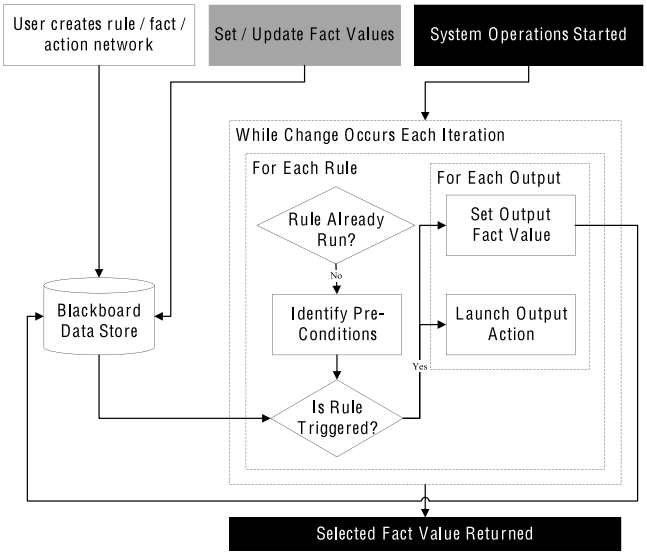

Fig. 3. Blackboard implementor system operations.

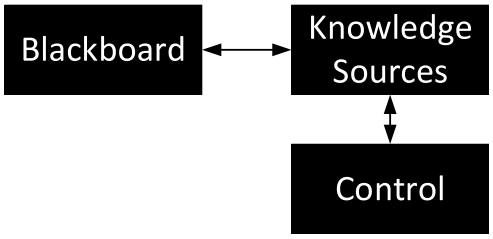

Fig. 4. Traditional Simple Blackboard Architecture.
Source: Created based on [27].

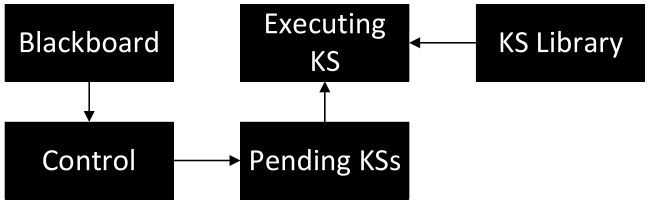

Fig. 5. Traditional Blackboard Architecture.
Source: Created based on [27].

order. An iteration concludes when all rules have been assessed and processed or determined to not have their pre-conditions met. Note that during their runs, rules can set facts to new values or trigger actions. Actions are implemented as calls to outside executables. In some operating environments, actions may trigger data collection which may trigger fact update commands (thus changing fact values).

The system continues iterations of running the rule processing engine until an iteration ends without any rules (which have not yet been run) having their pre-conditions satisfied and running. Once the rules processing engine iterations conclude, system operations also conclude, and the specified output fact's value is queried and returned.

4. Comparison to traditional blackboard architecture implementation

A variety of implementations of the Blackboard Architecture have been proposed during the over forty-year history of the technology. The key functionality of the technology is shown in Fig. 4, which depicts a relationship between a blackboard processing system, knowledge sources and a control mechanism [27]. The knowledge sources of the historical system include functionality paralleling that which implemented by the rule-fact network in the system described herein. Actions primarily implement the control functionality shown in Fig. 4.

Fig. 5 provides a bit more detail regarding the system's operations, showing that (like with the previously discussed expert systems), knowledge sources are selected (placed into a pending status) and then executed [27]. When a knowledge source executes, the results of this are based on the details of the knowledge source, as stored in the knowledge source library.

This concept is also illustrated by Fig. 6, which shows how knowledge source preconditions are used as inputs to a control mechanism, which selects knowledge sources to execute, making changes back to the shared blackboard environment [28].

Fig. 7 adds additional details to understanding the classical Blackboard architecture, showing how a monitoring module is used to track blackboard status and select knowledge sources as being ready to run [7]. A scheduler and scheduling queues are also included in this architectural depiction, filling in a key detail of how system operations are conducted (between the knowledge source selection and execution steps).

Notably, despite differences in terminology, the process of Blackboard Architecture operations is conceptually similar to rule-fact expert systems. Like with expert systems, numerous implementations have changed and enhanced components of system operations. Systems can be implemented with simple selection and queueing mechanisms or more complex predictive ones (mirroring the rule-fact expert system RETE rule selector [29]) and the selection algorithm has been shown to impact system operating speed [29] and may potentially impact the decisions that the system makes, depending on what is considered first.

The architectural implementations of prior Blackboard Architecture systems in this section can be compared to the architecture for the system described herein, which is presented in Fig. 2.

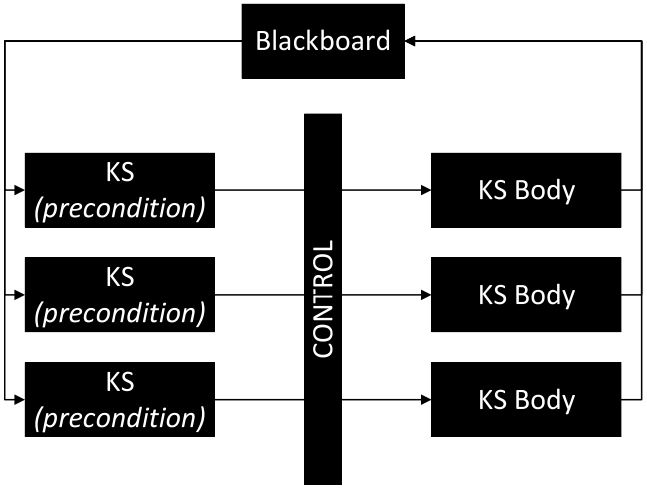

Fig. 6. Blackboard Architecture showing preconditions.
Source: Created based on [28].

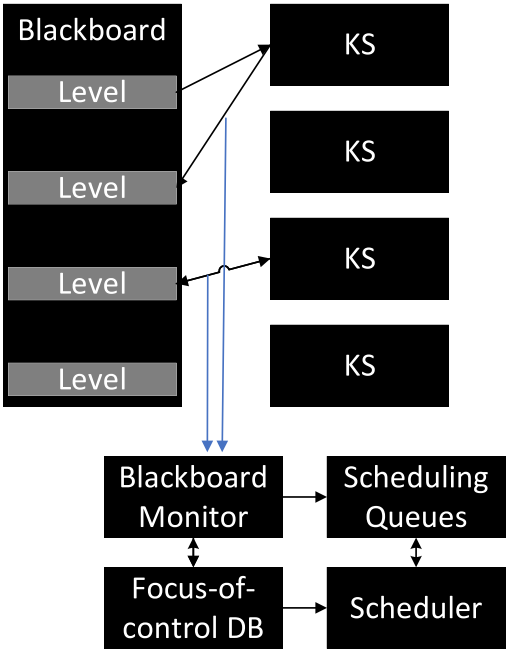

Fig. 7. Blackboard Architecture operations.
Source: Created based on [7].

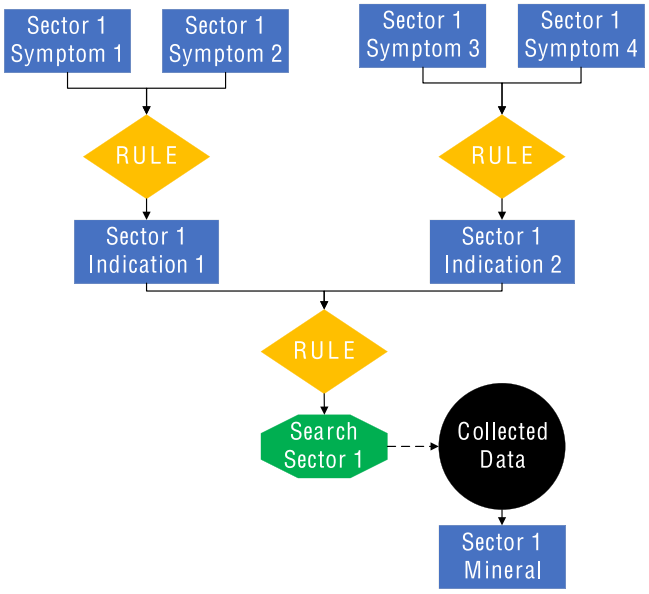

Fig. 8. Robotics application example.

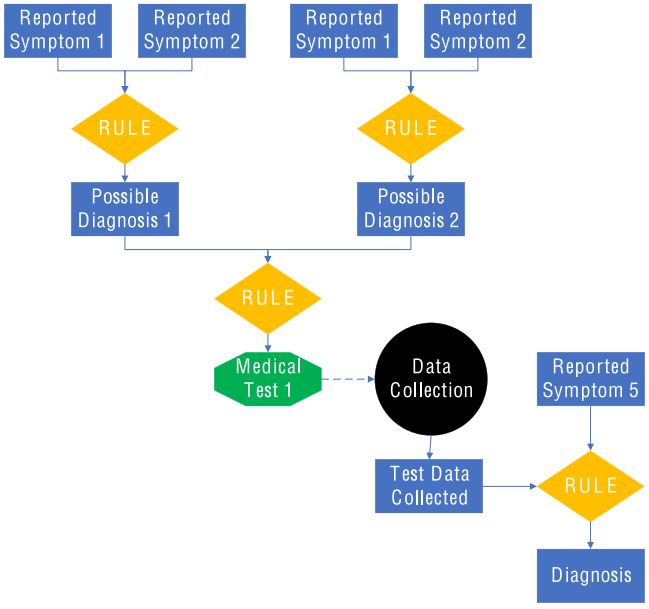

Fig. 9. Medical application example.

5. Examples of system use

To aid reader understanding of how the software described herein can be used, three examples are now presented. Fig. 8 shows how a Blackboard Architecture rule-fact-action network can be used to implement robotic control. A survey and materials gathering application is used for the example.

In this example, which is loosely based on [25], the application makes use of the action functionality to trigger a physical robotic search of an area based on analysis of prior data. Sector 1 symptoms 1 and 2's values indicate that sector 1 indication 1 can be asserted. Symptoms 3 and 4, similarly, assert indication 2. A rule triggers a search of sector 1 based on these two indications being asserted. When this search is completed, the sought mineral is detected and the sector 1 mineral fact is updated.

Another example, shown in Fig. 9, is a medical application. This application is similar to the previous robotic example, with two key differences. The symptoms are translated by the two top-most rules into possible diagnoses 1 and 2 (as opposed to complementary indications) and a rule triggers to instruct a human medical professional to perform a test (interestingly, with the computerization of many medical patient care systems, this can be easily accomplished by simply issuing an electronic order for a test, possibly after human physician review). This results in data collection, which updates the Blackboard's fact. This fact, combined with another relevant symptom, results in a diagnosis being generated.

A final example, shown in Fig. 10, highlights how the Blackboard Architecture (and the software described herein) can be utilized as part of a generic decision-making system. This type of system would be typical of those used by numerous financial institutions for account opening and loan-making decision-making. The top part of the system follows the typical rule-fact expert system pattern of drawing conclusions from inputted data.

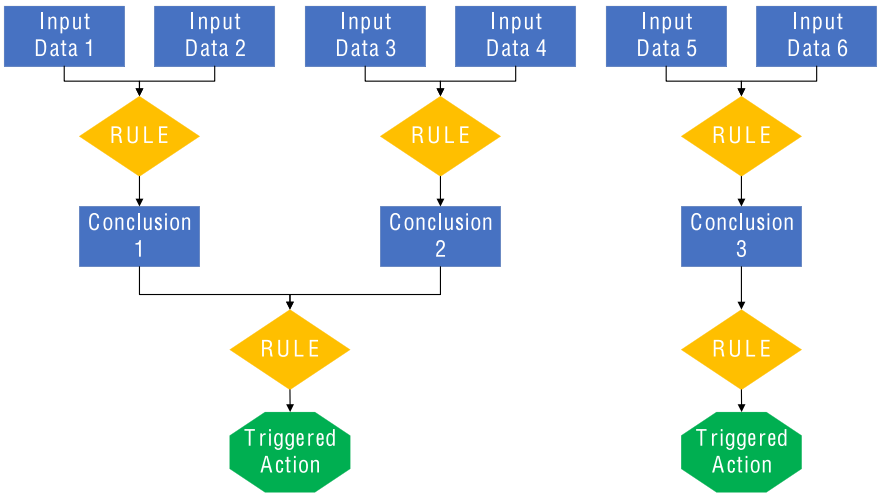

Fig. 10. Decision making system example.

The lower rules, on the other hand, show the power of the Blackboard Architecture by triggering an action (such as sending an acceptance or denial email) based on the conclusions drawn. In the left part of the network, four pieces of input data are used to draw two conclusions. Based on these two conclusions, an action is triggered. The right part of the figure shows how a single conclusion can cause another rule to trigger an action, based solely on that one conclusion.

These three examples demonstrate possible uses for the Blackboard Architecture and the software described herein. The first two (Figs. 8 and 9) show specific examples while the third (Fig. 10) shows the versatility of the system to serve any number of applications.

6. Advantages and limitations

The system presented herein has several key benefits. First, it is designed to be readily understandable and easily expandable to implementations of any size using applicable rule-fact networks. The use of a single rule-fact-action network simplifies understanding and system operations.

Second, as the system has been developed in a modular way, the command processor system can be removed and the Blackboard Architecture implementor module can be used as part of another system without the use of the command processor or command language. Alternately, the command processor and Blackboard Architecture implementor modules could be used together, simply removing the user interface, if this was desired for a given user system.

Third, the external actuation capabilities of the system allow it to be used for most applications where actuation can be performed using an executable file command, including parameters. The system could, prospectively, call functions that trigger actions in or alter the real-world environment that it is operating in.

Fourth, the system is compact, making it suitable for use on low resource level systems. The rule-fact-network requires minimal storage and the rule engine runs rapidly (taking only a fraction of a second per iteration for smaller networks).

Finally, it is hoped that by providing a simple and readily available Blackboard Architecture implementation, this may aid others in developing applications using the technology and facilitate a standardized approach to implementation, as standardization has been lacking across prior work.

In terms of limitations, the system has a few. First, it uses a basic round-robin rules processing engine. For some applications, it may be desirable to optimize the order that rules are selected to run in. The optimization of the rules processing engine, for expert systems, is a key area of system differentiation and the reason for the development of commercial systems like the later generations of RETE [30].

Second, the system lacks most of the various application- or project-specific advances that have been developed for Blackboard Architectures, such as pruning and distributed processing capabilities. Prospectively, these capabilities could be added to this simplified implementation by system implementers who desire these capabilities.

7. Use and impact

The system that the one described herein is based upon has been used, in different stages of development, for several projects. It has been refined for public release and the command processor and command parser capabilities have been added to facilitate ease of use. Additionally, all experimental code has been removed to make the system operate as expeditiously as possible.

The system that the current one is based upon has been used for several projects designed to demonstrate the efficacy of the Blackboard Architecture for different applications and to enhance the underlying Blackboard Architecture technology. In [22] and [21], a solver system was developed for the Blackboard Architecture and the impact of pruning the network on the solver's efficacy was assessed. The solver represented a different way to approach Blackboard Architecture network operations, based on searching for a desirable path through the network instead of operating in forward-only mode. This approach presumed what impact actions would have on the system's data to project what impact rules and actions would have.

In [23], the Blackboard Architecture technology was augmented with the use of boundary nodes which were shown to improve the throughput and meaningfulness of data transmissions in a distributed Blackboard Architecture system and/or reduce the cost of transmission. The boundary nodes were also shown to have significant encapsulation value as well.

In [24], the automation of system maintenance for a Blackboard Architecture system that runs without the potential for human maintenance or intervention was presented. Several techniques for facilitating long-term operations were presented and assessed.

Several projects have also used the system that this system is based upon, in different stages of development, for several applications. It was demonstrated for use in robotic control in [25,31], for spacecraft in [32] and for vehicles of multiple types in [31]. In [33], it was used to model cybersecurity attack and defense methodologies. It also served as the basis for the work in [34], though this project did not result in a fully working implementation. It has also served as the foundation for an autonomous penetration testing system [35]. This work was enhanced with a distributed Blackboard transmission steganography capability, which was presented in [36]. Work on the autonomous penetration testing system is ongoing, with three teams working on advancing this technology at present.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Thanks are given to researchers on projects that are making use of this system. Thanks are also given to Robert Fedor and Noah Ritter, whose unfruitful search for a Blackboard Architecture implementation to use demonstrated the need for this software publication. Finally, thanks are given to reviewers who provided feedback that substantially improved this manuscript.

Appendix A. Supplementary files

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2021.100183.

References

[1] C. Blum, D. Merkle, Swarm Intelligence - Introduction and Applications | Christian Blum |, Springer, Swarm Intell., 2008.
[2] M. Rath, A. Darwish, B. Pati, B.K. Pattanayak, C.R. Panigrahi, Swarm intelligence as a solution for technological problems associated with internet of things, in: Swarm Intell. Resour. Manag. Internet Things, 2020, pp. 21–45, http://dx.doi.org/10.1016/B978-0-12-818287-1.00005-X.
[3] B. Suri, S. Singhal, Analyzing test case selection &amp prioritization using ACO, ACM SIGSOFT Softw. Eng. Notes 36 (2011) 1, http://dx.doi.org/10.1145/2047414.2047431.
[4] F.B. Ozsoydan, A. Baykasoğlu, Quantum firefly swarms for multimodal dynamic optimization problems, Expert Syst. Appl. 115 (2019) 189–199, http://dx.doi.org/10.1016/j.eswa.2018.08.007.
[5] S. Park, T. Suh, Speculative backpropagation for CNN parallel training, IEEE Access. 8 (2020) 215365–215374, http://dx.doi.org/10.1109/ACCESS.2020.3040849.
[6] B. Hayes-Roth, A blackboard architecture for control, Artificial Intelligence 26 (1985) 251–321.
[7] L.D. Erman, F. Hayes-Roth, V.R. Lesser, D.R. Reddy, The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty, ACM Comput. Surv. 12 (1980) 213–253.
[8] E.A. Feigenbaum, B.G. Buchanan, J. Lederberg, On generality and problem solving: A case study using the DENDRAL program, 1970.
[9] V. Zwass, Expert system, 2016, Britannica https://www.britannica.com/technology/expert-system (Accessed 24 February 2021).
[10] R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum, J. Lederberg, DENDRAL: A case study of the first expert system for scientific hypothesis formation, Artificial Intelligence 61 (1993) 209–261, http://dx.doi.org/10.1016/0004-3702(93)90068-M.
[11] M.V. Johnson Jr., B. Hayes-Roth, Integrating diverse reasoning methods in the BBP blackboard control architecture1, in: Proc. AAAI, 1987, pp. 30–35.
[12] M.-J. Huang, H.-K. Chiang, P.-F. Wu, Y.-J. Hsieh, A multi-strategy machine learning student modeling for intelligent tutoring systems: based on blackboard approach, Libr. Hi Tech. 31 (2013) 6.
[13] G. Brzykcy, J. Martinek, A. Meissner, P. Skrzypczynski, Multi-agent blackboard architecture for a mobile robot, Intell. Robot. Syst. (2001); Proceedings. 2001 IEEE/RSJ Int. Conf., IEEE, 2001, pp. 2369–2374.
[14] Y. Yang, Y. Tian, H. Mei, Cooperative q learning based on blackboard architecture, in: Int. Conf. Comput. Intell. Secur. Work 2007, IEEE, 2007, pp. 224–227.
[15] A.M. de Campos, M.J. Monteiro de Macedo, A blackboard architecture for perception planning in autonomous vehicles, in: Ind. Electron. Control. Instrumentation, Autom. 1992. Power Electron. Motion Control. Proc. 1992 Int. Conf, IEEE, 1992, pp. 826–831.
[16] G. Goodman, R. Reddy, Alternative control structures for speech understanding systems, 1977.
[17] I.D. Craig, CASSANDRA-II: A Distributed Blackboard System, Department of Computer Science, University of Warwick, Coventry, UK, 1987, http://eprints.dcs.warwick.ac.uk/1210/1/cs-rr-090.pdf.
[18] H. Velthuijsen, B.J. Lippolt, J.C. Vonk, A parallel blackboard architecture, in: Proc. Third Int. Expert Syst. Conf, 1987, pp. 493.
[19] K.S. Ettabad, I.R. Farah, B. Solaiman, M. Ben Ahmed, Distributed blackboard architecture for multi-spectral image interpretation based on multi-agent system, in: Inf. Commun. Technol. 2006, second ed., ICTTA'06, IEEE, 2006, pp. 3070–3075.
[20] I.D. Craig, A New Interpretation of the Blackboard Architecture, Department of Computer Science, University of Warwick, Coventry, UK, 1993, http://eprints.dcs.warwick.ac.uk/1368/1/cs-rr-254.pdf.
[21] J. Straub, Evaluation of a multi- goal solver for use in a blackboard architecture, Int. J. Decis. Support Syst. Technol. 6 (2014) http://dx.doi.org/10.4018/ijdsst.2014010101.
[22] J. Straub, Comparing the effect of pruning on a best-path and naive-approach blackboard solver, Int. J. Autom. Comput. 12. 5 (2015) 503–510.
[23] J. Straub, A distributed blackboard approach based upon a boundary node concept, J. Intell. Robot. Syst. Theory Appl. 82 (2016) http://dx.doi.org/10.1007/s10846-015-0275-2.
[24] J. Straub, Automating maintenance for a one-way transmitting blackboard system used for autonomous multi-tier control, Expert Syst. (2016) http://dx.doi.org/10.1111/exsy.12162.
[25] J. Straub, H. Reza, A blackboard-style decision-making system for multi-tier craft control and its evaluation, J. Exp. Theor. Artif. Intell. 27 (2015) http://dx.doi.org/10.1080/0952813X.2015.1020569.
[26] J. Straub, Gradient descent training expert system, Softw. Impacts. (2021) 100121, http://dx.doi.org/10.1016/J.SIMPA.2021.100121.
[27] D.D. Corkill, Blackboard systems, dan corkill repos. Website, 1991.
[28] K. Ppeger, B. Hayes-Roth, An introduction to blackboard-style systems organization, stanford, CA, USA, 1997, http://frostiebek.free.fr/docs/Frames Blackoard/KSL-98-03.pdf (Accessed 25 October 2021).
[29] C.L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, Artificial Intelligence 19 (1982) 17–37, http://dx.doi.org/10.1016/0004-3702(82)90020-0.
[30] C.-A. Berlioz, The rete algorithm explained!, 2011, Sparkling Log https://www.sparklinglogic.com/rete-algorithm-demystified-part-2/ (Accessed 6 June 2021).
[31] J. Straub, H. Reza, The use of the blackboard architecture for a decision making system for the control of craft with various actuator and movement capabilities, in: ITNG 2014 - Proc. 11th Int. Conf. Inf. Technol. New Gener., 2014, http://dx.doi.org/10.1109/ITNG.2014.86.
[32] J. Straub, Swarm intelligence, a blackboard architecture and local decision making for spacecraft command, in: IEEE Aerosp. Conf. Proc, 2015, http://dx.doi.org/10.1109/AERO.2015.7119165.
[33] J. Straub, Modeling attack, defense and threat trees and the cyber kill chain, ATTCK and STRIDE frameworks as blackboard architecture networks, in: Proc. 2020 IEEE Int. Conf. Smart Cloud, Institute of Electrical and Electronics Engineers Inc., Washington, DC, USA, 2020, pp. 148–153, http://dx.doi.org/10.1109/SmartCloud49737.2020.00035.
[34] D. Rosch-Grace, J. Straub, Body area networks: a data sharing and use model based on the blackboard architecture and boundary node discovery, in: Futur. Inf. Commun. Conf, Springer, San Francisco, CA, United States.
[35] J. Straub, POSTER: Blackboard-based electronic warfare system, in: Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur, ACM, 2015, pp. 1681–1683.
[36] T. Lei, J. Straub, B. Bernard, Lightweight network steganography for distributed electronic warfare system communications, in: Proc. 2020 World Congr. Comput. Sci. Comput. Eng. Appl. Comput, Springer Nature, Las Vegas, NV, 2020.