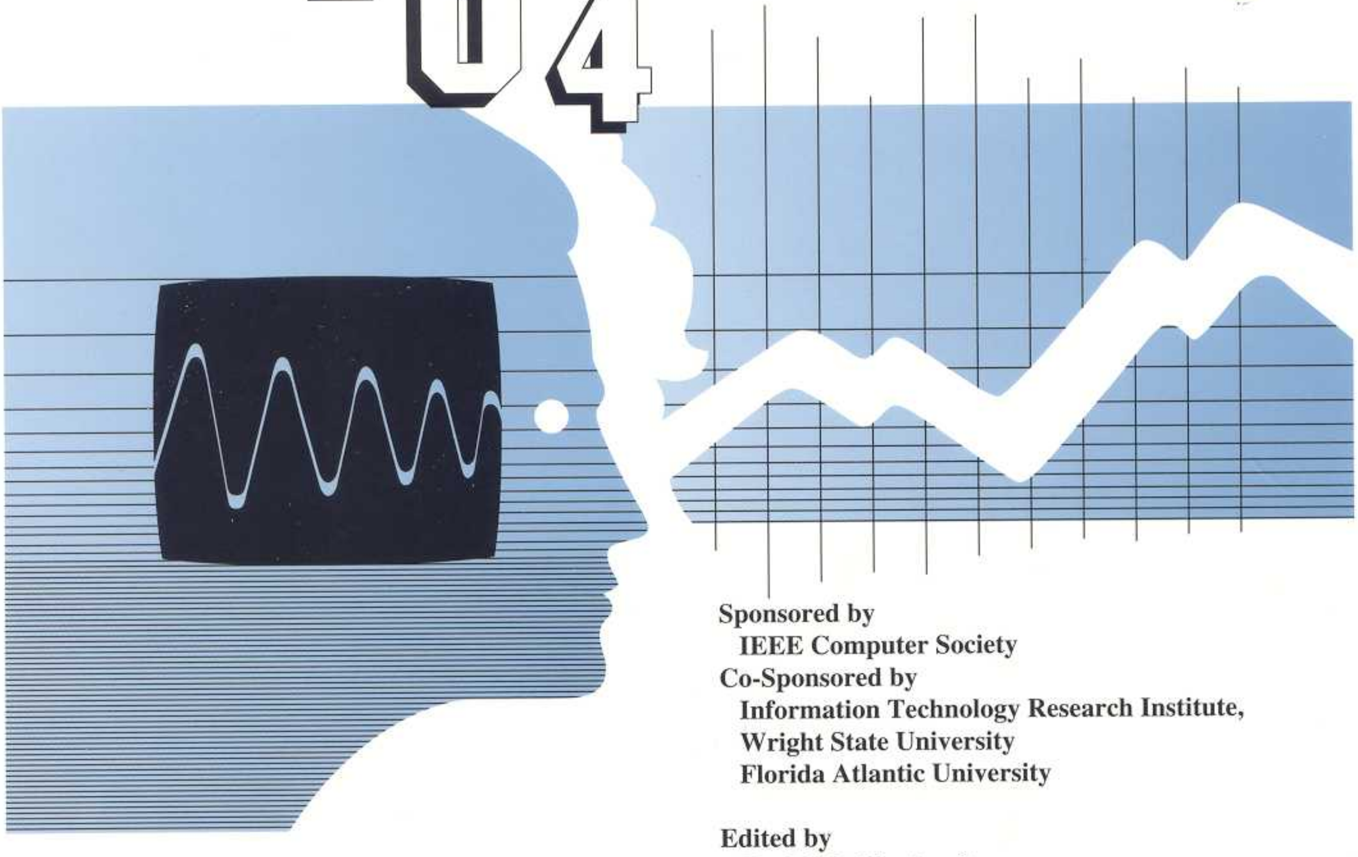
Proceedings

Sixteenth International Conference on

TOOLS WITH ARTIFICIAL INTELLIGENCE

ICTAI 04

15–17 November 2004
Boca Raton, Florida



Sponsored by
IEEE Computer Society
Co-Sponsored by
Information Technology Research Institute,
Wright State University
Florida Atlantic University

Edited by
Taghi M. Khoshgoftaar

IEEE COMPUTER SOCIETY

IEEE

Exploiting Systemic Biological Modeling for Trigger Based Adaptation in Networked Intelligent Multi-Agent Systems

Arvind K. Bansal
Department of Computer Science
Kent State University, Kent, OH 44242

Abstract

Current day networked intelligent agent based systems have limited capability of adaptability, self-repair, adaptation, and self-reconfiguration under changing external conditions. In past, evolutionary algorithms have experimented with random mutation and heuristic selection based evolution for self-adaptation. However, little research has been done to explore dynamic adaptive control to take care of sudden external stress and events at systemic response level. This paper introduces a new message based biological model of intelligent multi-agent based systems that represents agents as self-correcting dynamically modifiable genes – a reconfigurable set of dynamically regulated built-in functions, and system of agents as dynamically adaptable event-trigger controlled interacting pathways that can be altered and reconfigured in response to external stress and events. The model supports the integration of message, code, trigger, and belief states, and supports interchangeability of message, code, and trigger to provide dynamic adaptive control. The model and its implementation have been described.

Key words: Adaptability, intelligent agent, artificial intelligence biological model, distributed, genes, pathway, Systemic

Introduction

As the knowledge based systems are becoming more complex and networked, there is an increasing demand for adaptive autonomous intelligent multi-agent systems [13, 18] interacting over the Internet. An autonomous agent has its own belief system, and could be used for multiple functionalities depending upon the environment. In a networked environment an ideal agent should be able to adapt and dynamically reconfigure itself in response to the external environment to avoid system failure and to provide adequate response. The lack of such adaptation would increase the probability of cascade failure of interacting agents, hamper the overall system response, and negatively affect the capability of decision making by the humans in the loop.

Despite advances in the technology of developing intelligent agents, current day intelligent agent based systems [13, 18] do not adapt and reconfigure well to the sudden change in external conditions, and do not support complex adaptable protection against system failure which is a hall mark of biological systems.

Biological systems [2] have remarkable capability to adapt through genetic evolution, adaptive modification of system of interacting pathways, gene-repair, and gene modification through recombination and rearrangement. Evolution provides time based self-adaptation by generating new desirable random mutations. Adaptive modification provides dynamic selection and regulation of already existing set of functions at the systemic level of pathways. Adaptive control involves promoter based switching and regulation of gene functions, and invocation of latent pathways (or shutdown of pathways) in response to external events. Biological systems support controlled interchangeability between messages, triggers, and code (genetic segments), and message directed dynamic code and interaction modification through dynamic insertion, deletion, and suspension of genetic segment to handle environmental stress.

Evolutionary computing [1, 9] algorithms have experimented with gene based models to provide continuously evolving intelligent systems [10] by exploiting random point mutation of genes and limited random genome rearrangement such as cross-over of gene segments [1] combined with heuristic selection [11].

The evolutionary systems lack the capability to model genes as interacting message (or event) controllable set of domains, and do not take into account dynamically modifiable selection, regulation, invocation of latent functions, and systemic interaction of various units at multiple hierarchical control levels such as team of cooperating agents and interacting pathways. Little exploration has been done to provide dynamic adaptive control by: (1) dynamic modification of interaction between pathways, (2) dynamic modification of interaction between coordinating agents, and (3) dynamic modification of interaction between functional domains within an agent in response to external stress, external messages, and event based triggers. Only recently researchers have started looking into systemic aspect of biological systems to model complex systems [12] based upon the theory of cybernetics and control systems [3].

This paper models complex intelligent systems as a set of interacting communicating pathways. Each pathway consists of multiple interacting team of coordinating agents that are modeled as operons —

multiple genes working in tandem to compose high level functionality. Each cooperating team of agents has a common goal and shares a blackboard for the ease of communication. Genes are dynamically modifiable, and are modeled as a set of interacting domains. Each domain is a symbolic reference to a built-in function, and also acts as an identifier for domain-domain interaction. The intelligent agents are modified using dynamic domain insertion, domain fusion, domain deletion, domain duplication and domain suspension temporarily or indefinitely in response to event triggers.

The overall contributions of this paper are as follows:

1. The model supports interacting hierarchical information processing, dynamic reconfiguration and dynamic adaptive control of complex units in response to messages, sudden stress, and triggers.
2. The computing units are reorganized through insertion, deletion, duplication of built-in core functions or derived functions, suspension and regulation of agents and/or functionalities based upon message communication between different units.
3. The model supports transformation of message to code, message to trigger, code to trigger which provides a unique capability to dynamically change the code to adapt to the external changes.
4. The interaction between multiple functional domains within a gene can be modified (permanently or temporarily) either through external message or in response to an event based trigger.

The system has been implemented, and is running on a cluster of four distributed computers. The higher level system has been coded using Sicstus Prolog, and built-in functions have been coded using C and C++ library. The system has been applied to an image capture and processing system [15].

The overall paper is organized as follows. Section 2 describes the needed background. Section 3 describes a functional model of biological agents and dynamic modification of agent's functionality using the notion of dynamic affinity matrix. Section 4 describes the hierarchical construction of biological agents. System 5 describes the overall behavior of bio-agent systems. Section 6 describes the implementation architecture of the genome based bio-agent systems and check-pointing as a means to reuse the newly adapted system. Section 7 describes briefly an application. Section 8 describes the related works. The last section concludes the paper, and describes the future work.

2. Background

This section describes the needed concepts and definitions of a single cell based system, new graph based definitions to model dynamic domain-domain interaction to model an agent based system, and logical clock based communication to maintain serial property of events in a distributed multi process environment.

2.1 Biological concepts

An organism consists of multiples types of interacting differentiated cells with different functionality. Each *cell* [2] is a set of active interacting pathways. Each *pathway* models a control flow network of transformation of biochemical substrates. Each pathway consists of multiple operons and genes (or the corresponding proteins). Proteins are formed by transcription and translation of genes. Each *operon* is a set of genes involved in a common functionality, and share a common promoter — a control sequence of nucleotides used to enhance or repress the rate of transcription and translation of the a gene and thus affecting the overall processing rate of biochemical substrates. The difference between the operon and a gene is that there is a single promoter in an operon despite having multiple genes.

Each *gene* consists of multiple interacting domains using hydrogen bonding, electrostatic charge, and to some extent covalent bonds. Domain-domain interaction is present as: (1) interaction between two domains within the same gene, (2) interaction between a domain of a gene and a messenger protein, and (3) interaction between a domain of a gene and the domain of another gene of a foreign object. Interaction between the domains within the same gene causes a gene to dynamically reconfigure resulting into change of gene-function or splitting of the gene. The interaction between a domain and another domain is also used to insert (possibly temporarily) a gene segment, dynamically change the configuration of a gene, splice a gene, expose a gene's surface to be attacked by other genes, and enhance or repress the expression of gene to a protein.

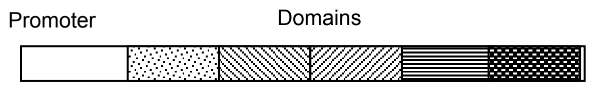


Figure 1. The organization of a gene

Pathways [2] are of many types. Some of the important pathways relevant for modeling information processing systems are: *signaling pathways* — information carriers used to regulate the activity of another gene by binding the communicated messenger protein to the promoters, *metabolic pathways* — used to transform the input biochemical compound for the required cell activity, *transcription and translation pathways* — used to generate multiple copies of proteins using the same gene template, *sensor pathways* — used to sense the environment and send signal to cells to adapt accordingly, *transporter pathways* — used to bring nutrients and information from external environment to the inside the cell, *stress-response pathways* – used to regulate or suspend the currently active pathways (or activate latent pathways) in response to sudden extreme change in the environment, *gene-repair pathways* – used to correct the faulty genes, and pathways involved in providing immunity. Many of these pathways are intricately interrelated through triggers initiated through concentration of substrates and/or foreign bodies, proximity of gene-domains to each other, and environmental conditions.

A *plasmid* [2] is an independently replicating element consisting of a bag of genes which transfer from one genome to another genome, and causes genome rearrangement. A *transposon* [2] is a mobile genetic element which moves to different specific sites within a genome carrying with it neighboring genes to cause genome rearrangement.

A bacterial organism reacts to the changes in the external environment using *stress response adaptation* and *evolution based adaptation*. Evolution based adaptation could be *point mutation* where an insertion/deletion of an individual molecule (*adenine, cytosine, guanine, thymine*) changes the gene sequence causing a change in functionality or through genome rearrangement where a subsequence of a gene gets replaced to another gene *Genome rearrangement* could be caused by transferring a domain to other gene either from external genomes such as plasmids or lateral gene transfer [2, 6], within the same genome through crossovers [2] and through transposons. Stress response is a trigger based activity, may activate a latent pathway, cause a subset of current pathways to become extinct or temporarily suspended by make some genes dormant, or reconfigure the genes so that they do not support a subset of interactions. *Heat shock* and *cold shock* proteins [8] are examples of stress response.

The biological systems use double helix structure for stability and error correction against random mutation. The mutation of nucleotides can be corrected by gene-repair genes by looking at the complementary strand.

2.2 Graph related concepts

This paper models domain interactions within a gene using *affinity* relationship. *Affinity* is an asymmetric transitive relationship. Asymmetry and transitivity of affinity relationship provides the directionality in the processing of information. A domain interacts with another domain(s) with highest affinity. The affinity between domains within a gene is modeled as a weighted directed graph (see Figure 2). The weight of an edge shows the affinity of the source node to the destination node. A *maximum affinity edge* from a node has the highest weight among all the outgoing edges from the node. Each node with at least one outgoing edge has at least one maximum affinity edge. A *maximum affinity path* consists of maximum affinity edges connecting two nodes. The information is passed along the maximum affinity edges from source node to sink node.

Example 1 Consider the weighted directed graph in Figure 2. The node V_1 is a source node, and the node V_5 is a sink node. The edges (*V1, V2*), (*V1, V3*), (*V2, V5*), (*V3, V4*), (*V4, V2*) and (*V4, V5*) have the weights *8, 20, 2, 10, 5, 9, and 4* respectively. The maximum affinity edge from the node V_1 is (V_1, V_3), the maximum affinity edge from the node V_2 is (V_2, V_5) as it is the only outgoing edge from V_2. The maximum affinity path between the nodes V_1 and V_5 is (V_1, V_3) (V_3, V_4) (V_4, V_2) (V_2, V_5) as marked by bold lines in Figure 2.

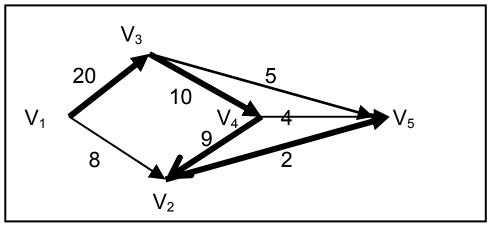


Figure 2. A maximum affinity path

The maximum affinity path changes if at least one of the maximum affinity edges are deleted, a new incident edge with a weight higher than the current maximum affinity edge is inserted, or if the weights of the maximum affinity edges are altered.

Example 2 Let us consider the graph in Figure 2. The deletion of the maximum affinity edge (V_3, V_4) will make the maximum affinity path between the nodes V_1 and V_5 as (V_1, V_3) (V_3, V_5). Similarly insertion of one or more nodes will also change the maximum affinity path.

A weighted graph is represented in the matrix form by representing the nodes as rows and columns and representing the edge (V_i, V_j) connectivity by the corresponding non zero weight in the ith row and jth column. Using matrix representation, the maximum affinity edges are identified by finding the maximum nonzero value in the corresponding row vector.

2.3 Logical clocks and communication

In a distributed processing system involving multiple processes, the order of events can not be ascertained by physical clocks due to difference in physical clocks in different processors and the scheduling delay of the processes within the same processors. Instead a distributed logical clock [14] is tagged with the events to maintain their serial order. A distributed logical clock is a vector of event counters. An event could be change of *belief state* or *transmission of a message* to another agent. This paper models an *event counter* in an agent as a *pair* of the form (*state-counter*, *message counter*) [5]. A *message counter* is incremented when a message is transmitted from an agent or a message is received from other agents. A state counter is incremented when an agent processes an incoming data or it is dynamically reconfigured. The reconfiguring operations could be modification of a domain, modification of affinity matrix, or suspension (or activation) of an agent. An event counter (S_1, M_1) precedes another event counter (S_2, M_2) if $S_1 < S_2$ and $M_1 \leq M_2$ or if $M_1 < M_2$ and $S_1 \leq S_2$.

A logical clock T_1 precedes another logical clock T_2 if there exists at least one event counter in T_1 which precedes the corresponding event-counter in T_2 and all other counters in T_1 are less than or equal to the corresponding counters in T_2.

Each agent transmits three types of counter-pairs: its own counter-pairs, counter-pairs of logical neighbors, and the counter-pairs received by the agent-team in the previous messages. The rationale is that these three counter-pairs are needed in a logical clock to ascertain if the message sent by an agent has been received by the destination agent [5]. It has been shown in [5] that if the time-stamp received by any neighbor of the agent transmitting an information *I* follows the time-stamp transmitted with *I* to a neighbor then the transmitted message has been received by the intended destination.

3. Functional Model of Bio-intelligent Agents

There are four basic units: information processing units (genes), information carrying units (messages), control units (promoters or triggers within messages), and belief systems within agents. In our computational model we do not use the notion of proteins in modeling pathways. Instead we substitute proteins by genes. The rationale for this approach is that biological systems sense the signals based upon concentration of the message units and thus need multiple copies of proteins, while our system recognizes even a single trigger in the information. This variation provides computational efficiency in our model. The basic unit of functionality in modeling a bio-intelligent agent is a gene. A gene is modeled as a pair of the form (*P, D*) where *P* is a control-unit analogous to the promoter inside a gene, and *D* is an ordered set $<D_1, D_2, ..., D_I, ... D_M>$ ($M > 0$) of set of interacting domains within a gene. The control *P* is a set of segments $<P_1, P_2, ..., P_I, ..., P_M>$ ($M > 0$) such that each control-segment P_I controls the activity of the corresponding domain D_I by shutting a domain, or turning on a suspended domain. Each P_I is a quadruple of the form (*Identifier sequence, State*, *Action*, *Counter*). The *identifier-sequence* is used to bind with the trigger-sequences from other agents or other domains within the same agent, *state* is either *on* or *off* or *suspended*, and a *counter* provides the transition back to original state after a certain time. The counter is decremented each time an event occurs. An *action* could be *splitting* the corresponding gene at a particular domain, *suspending* the domain activity, *deleting* the domain, *fusing* two adjacent domains, or *attaching* domains in an incoming message to the gene to modify the functionality of the agent.

The interaction between the domains within an agent is modeled by an *N X N* affinity matrix where *N* is the number of domains in the agent. The first domain of an agent reads the input data from the corresponding blackboard or input buffer, and the last domain writes the data to the corresponding blackboard or output buffer. The interaction is based upon an affinity value D_{ij} between the domains D_i and D_j.

3.1 Affinity matrix modifications

The affinity matrix changes dynamically in response to a domain-domain interaction. Affinity matrix is modified using (1) insertion/deletion of a domain, (2) domain duplication, (3) domain splitting, (4) fusion of two domains (5) suspension of a gene, and (6) built in timed internal modification within a domain as explained in Example 3. All these operations could be either in response to an event communicated through a message, or built in the counter in the control-segment of the corresponding domain. Except deletion which is a permanent operation, all other operations such as suspension, insertion, fusion, duplication, and splitting can be temporally constrained and could be transient. A system is *transient i*f the system gradually recovers its original state over a period of time after the condition that caused the change is no more present.

When a gene is inserted/deleted, the affinity matrix and the promoter area are dynamically modified: In case of insertion, the affinity matrix becomes ($N + 1$) X ($N + 1$`) matrix, and in case of deletion the affinity matrix becomes ($N - 1$) X ($N - 1$) matrix. Duplication of a domain is treated as insertion with an exception that there are no edges between the original node and the duplicated node. Similarly, fusion reduces the size of the matrix by 1, and creates an entry in a dynamic virtual table of composite functionality which refers to sequence of functions in the built-in function table.

Example 3 Consider an image processing agent which takes a raw image and compresses it. There are four domains in the gene: Domain 1 reads an image file, and passes the file to Domain 2. Domain 2 performs lossy compression of the incoming image by 60% every cycle. Domain 3 performs lossless compression every 10th cycle, and the Domain 4 writes the compressed image every cycle to the output buffer. All four domains are controlled by a promoter comprised of four segments one for each domain. The control-segment for Domain 3 counts the incoming messages, and enhances the affinity of the corresponding domain(s) every 10th cycle temporarily for one cycle making it maximum affinity. The control segment can also be controlled by messages transmitted by other agents or other domains within the same agent. Note that every control-segment is pre-programmed to take a predefined action. The overall gene with initial affinity matrix is given in Figure 3.

According to the affinity matrix, the input domain has maximum affinity for Domain 2, and Domain 2 has maximum affinity for the output domain. Every 10th cycle, the matrix changes for the next cycle such that Domain 1 has maximum affinity for Domain 3, and Domain 3 has maximum affinity for the output Domain 4.

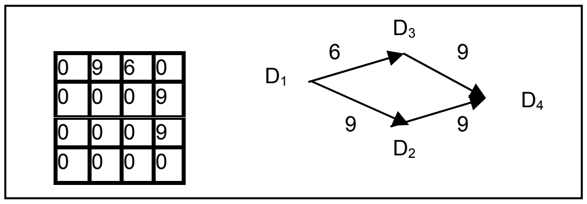


Figure 3. Domain affinity matrix in a bio-agent

4. Building Bio-intelligent Agents

Like biological system, the domain signature is a sequence of characters taken from the alphabet {*A, C, G, T*}. The advantage of a signature is to create rules for (1) specific pair-wise dynamic interaction between multiple domains, (2) to provide identification mechanism in the domains so that intruders could be avoided or destroyed, and (3) to provide indexing to look up the corresponding built-in function corresponding to a domain.

The alphabet has been modeled as 4-bit combination {*0011, 0110, 1001, 1100*}. Each 4-bit combination is two Hamming distance away to avoid ambiguity in error detection and error correction if one of the bits is corrupted. It can be seen that the pairs *0011* and *1100* are complement of each other under Boolean logic, and pairs *0110* and *1001* are complement of each other. Two characters are complements if the result of their logical-AND is 0. In order to emulate the double stranding for signature repair, each byte contains the information in the pair form (X, $\sim X$). where $X \in$ {*0011, 0110, 1001, 1100*}, and $\sim X$ is the complement of X. A corrupted bit is corrected by identifying the character X that is not an element of the alphabet, and replacing X by the complement of character $\sim X$ stored along side. Every domain is terminated by a sequence of repeats of the same character such as 'AAAAAAAAAA'. Like biological systems, repeat characters are used to avoid any misinterpretation or loss of information due to frame shift caused by character deletion. At the same time, they allow for variable length domains caused by domain fusion/splitting.

An operon is modeled as a ($P, \Gamma_1, \Gamma_2, ..., \Gamma_M$) where M denotes the number of genes, and each subscripted Γ_I represents the ordered set of domains in a gene. The control-unit is modeled as a pair of (*Global control, ordered set of segments controlling domains in the genes*). The global control shuts off or switches on all the genes, while individual segments provide individual control of the segments.

A domain in a gene is controlled by (1) interaction of message identifier and control segment identifier, or (2) internal clock controlled interaction. There is a built-in association table which describes the interaction between two domains. After a message is transmitted, all the control segments receiving the message check against the binding association table to match pair of binding patterns. After a successful match, the corresponding control segment reacts by binding to the identifier sequence of the message. Otherwise, the message is purged.

An agent-team is connected to other neighboring agent-teams within the same pathway (or other related pathways) through input-output relationships for processing the incoming data and sending the output data.

5. Overall Behavior

The pathways are distributed on a system of processors. The initial preference is to keep all the agents occurring in the same pathway on the same processor to reduce the communication overhead. Each pathway consists of multiple agent-teams cooperating towards a common goal. All the agents within an agent-team communicate to each other using the shared blackboard. In addition to information exchange, the blackboard stores the beliefs of individual agents in the agent-team, log of all the actions performed on the agents, the log about the information processed, the input and output buffer from multiple agents within the same agent-team, logical clock and the port details for the set of neighboring agent-teams, the types of actions requested by other agents, the association tables that keep the information about the domain-domain interaction, and the association table of (*domain symbol, index of the function*) to invoke functions in response to the domains present in the agents. The overall structure of an operon is given in Figure 4.

The adaptability is incorporated using the combination of sensor pathways and stress pathways. There are multiple types of sensors in an intelligent system such as temperature senor, pressure sensor, humidity sensor, traffic congestion sensor, intrusion/attack sensor (when number of messages with unidentified identity tags increases beyond a threshold), image analysis sensor such as identification of an object in the database. Once these sensors indicate an event, a trigger is activated. The trigger itself is in the form of a message, and is either broadcast, multicast, transmitted using agent to agent communication event within the same operon, or transmitted through domain to domain communication within the same agent.

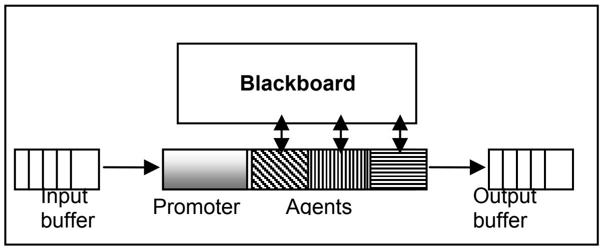


Figure 4. Basic structure of an agent-team

Each message is a 10-tuple of the form (*originating agent, originating agent-team, originating processor, destination-agent, destination agent-team, logical-time-stamp, identification key, data-type, data/action*). The *destination agent-team* is a wild-card in the case of an action which can be applied to many agent-teams.

A managing agent in each host computer decrypts the identification key and matches against the keys allocated to the originating agent-team. After matching the key, the destination is matched against the currently active agent-team in the database. If the agent-team is active, the message is written to the destination channel. In case of an action that needs to be broadcasted, first it is checked whether originating-agent team has the privileges to broadcast the action. The originating agent-teams involved in stress pathways have such capability. However, simple data processing agents such as image processing agent-teams do not have such capabilities.

Each information unit sent to other agent-teams is logically time-stamped as described in Subsection 2.3. After the information is generated and transmitted to the intended logical neighbor, the comparison of the time-stamps of incoming message and the transmitted message, can ascertain the receipt of the message by the intended agent. If the time-stamp of transmitted message precedes the time stamp of incoming message then the intended agent has received the message [5]. After a message is received, the identifier sequence in each control segment is matched with the identifier sequence in the incoming message. After a match is verified, the corresponding action is taken by the built-in function. The action invokes the corresponding function after the domain of the message and domain of a control segment matches.

6. Implementation

The system contains a master boot process, a boot up engine, a genome representation of interacting pathways, a program transformation engine, a check point engine, and pathway engines one for each pathway. The master boot starts a watchdog process and a boot-engine process. The watchdog process restarts another copy of master boot in case the master boot process fails. The boot engine starts a program transformation engine that analyzes the genetic code (shown as circle in Figure 5) to identify the input-output relationship between various agents and agent-teams, to identify separate pathways, and builds up two association files that contain mappings of the form (domain-domain interaction → action → meta program) and (domain-identifier → built-in function) respectively. The genetic code of different pathways and their relationships are also separated. The boot engine then starts a Prolog engine for each pathway.

Each pathway engine analyzes their genetic code to identify the connectivity of the agent-teams, and builds up a shared file which contains input-output connectivity between different agents. Each pathway engine starts multiple processes one each for an agent-team. These processes communicate to each other using a separate port.

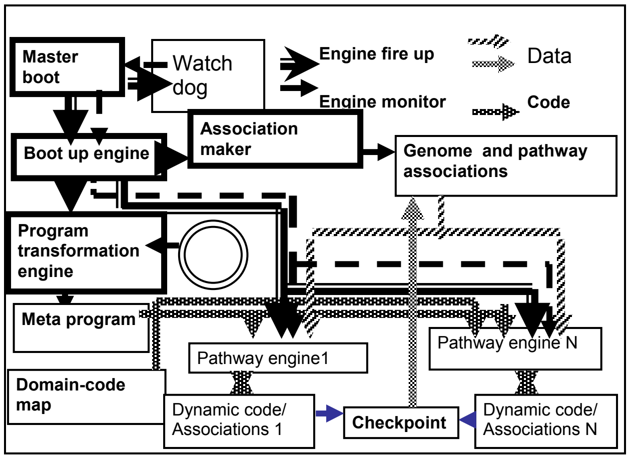


Figure 5. An implementation architecture

Each agent–team executes a meta program. The meta programs for each agent-team has additional built-in agents (1) to verify the identity of the incoming message (2) to insert (or strip) the time-stamps, identity, and destination to the outgoing messages. The meta program looks up the corresponding genetic code for the agent-team, analyzes the messages to dynamically change the affinity matrix, and executes the built-in functions associated with the domains in an agent. Each meta-program also builds an association table in its blackboard that stores mappings of the form (*domains, corresponding code reference*) to extract the built-in function for each agent. Each agent-team contains its own log file to record the changes in the domains and the corresponding affinity-matrix. As the meta-programs for agent-teams executes, the definition of genes and the corresponding dynamic matrix changes. The log file is altered every time affinity matrix corresponding to the agent-team is altered. These changes are transformed back to the genomic form periodically and are check-pointed to provide future recoverability of the changed system.

7. An Application

We have tried the system on an image capture and image processing system [15]. The image capture and processing system has six major information processing pathways: camera control, image capture, image compression, image segmentation, texture analysis to identify images, and image transmission. In addition to these pathways, there are sensor related pathways as described earlier which will temporarily slow down or shut down the processing unit in the presence of environmentally unsafe condition.

The adaptability in this system comes from five aspects: resolution, bandwidth limitation, the time and memory overhead of processing high quality images, and handling the system in extreme environmental conditions.

8. Related Works

To the best of my knowledge, no networked intelligent multi-agent based system has been modeled at systemic level as the network of interacting pathways involving the integration of message, triggers and dynamic modification of agents using domain-domain interactions. However, evolutionary computing [1, 9, 10, 11] has bio-mimicked self-adaptation through heuristic mutation, and immunity based intrusion detection researchers [4, 17] have bio-mimicked security from biological systems with considerable success. There are also successful attempts to model artificial life using intelligent multi-agents.[7], and there are many applications that use multi-agents to model biological pathways.

The notion of adaptation described in this paper is related to 'adaptive control' of systemic behavior, and is different from the notion of random mutation based self-adaptability [11] used by researchers in evolutionary computing [1, 9, 10, 11]. The systemic adaptation as described in this paper uses a finite set of dynamically selectable component functions which are dynamically rearranged using event based triggers. In contrast to random mutation of genes in evolutionary computing, the emphasis in this paper is on dynamic modification of interaction, triggered based insertion and deletion of genes, and trigger based activation or suspension of pathways. This model has a finite set of domains, and new genes are formed by the selection of a subset of domains and dynamic modification of their interaction. In contrast to classical identifier based communication in current day systems, this model supports both identifier based communication as well as functionality based communication to other agents. Functionality based communication frees the communicating agent to know the identity of the receiving agent providing extra flexibility as any agent can substitute for another agent with the same functionality, and multiple agents with the same functionality in different pathways can be instructed simultaneously

9. Conclusion

This paper describes a hierarchical systemic biological model of agent based systems. The system supports adaptability based upon sudden stress in response to the external conditions and messages that can also act as triggers. This response to the message can be time bound or event bound. The system of agents is regulated using an agent control-segment which can dynamically modify the functionality and interaction between the interacting domains in response to a trigger. Regulating the promoter regulates the execution of the corresponding built in function in an agent. The system supports both agent-to-agent communication as well as domain-domain interaction based communication where the message is broadcast, and all the agents which support a specific function are concurrently affected by the trigger.

This work provides intentional function based adaptation of the system, and dynamic activation of latent pathways in response to event triggers, periodic triggers, or triggers generated in response to messages. The agent code keeps changing in response to domain-domain interaction of messages. The gene modifications are domain splitting, domain fusion, or suspension of a domain. This modification reconfigures the functionality of an agent.

The model and the system complement evolution based systems. An interesting approach will be to integrate trigger based adaptation described in this paper, evolutionary systems, and immunity based security to completely bio-mimick fail safe adaptation of biological system. There is also a need for incorporating integrity constraints at the agent level to facilitate system stability. Currently, the genetic code is hand-coded. There is a need to develop an editor to semi-automate the process.

The model is also being extended to handle Internet based biologically inspired migratory agent based systems, and to support immunity response [4, 17] to support protection against intrusion.

Acknowledgements

This research was supported in part by Wright Patterson Base Subcontract # SKM-T1/F33615-97-D-1138 under 'Secure Knowledge Management' project. I thank Bill McQuay and Nikolaos Bourbakis to encourage a fault tolerant version of this model based on [5] that will be presented elsewhere due to space limitation. I am also thankful to Darshan Patel, Cheng Lu, Ming Ming Lu, and Qiyu Zhang for providing the image capturing and analysis system [15] needed to test this model.

References

[1] P. J. Angeline,"Multiple Interacting Programs: A Representation for Evolving Complex Behaviors," *Cybernetics and Systems*, 29 (8), 1998, pp.779-806

[2] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, J. D. Watson, "Molecular Biology of the Cell," Third Edition, *Publisher: Garland Publishing Inc*., 1994.

[3] R. Ashby, "Introduction to Cybernatics," *Publisher: Chapman and Hall Ltd*, London, 1957, http://pcp.vub.ac.be/Books/introCyb.pdf

[4] J. Balthrop, F. Esponda, S. Forrest and M. Glickman. "Coverage and Generalization in an Artificial Immune System," *Proceedings of the Genetic and Evolutionary Computation Conference. (GECCO 2002),* Morgan Kaufmann. New York, 2002, pp. 3-10.

[5] A. K. Bansal, K. Rammohanarao, A. Rao, "A Distributed Storage Scheme for Replicated Beliefs to Facilitate Recovery in Distributed System of Cooperating Agents," *Fourth International AAAI Workshop on Agent Theory, Architecture, and Languages, Lecture Notes in Springer Verlag Series*, *LNAI 1365*, 1998, pp. 77 - 92.

[6] A. K. Bansal, "An Automated Comparative Analysis of seventeen Complete Microbial Genomes," *Bioinformatics*, Vol. 15: 11, 1999, pp. 900 – 908

[7] I. Burleigh, G. Suen, and C. Jacob, "DNA in Action! A 3D Swarm-based Model of a Gene Regulatory System," In: *Proceedings of the First Australian Conference on Artificial Life*. Lecture Notes in Computer Science. Springer-Verlag: Berlin, 2003

[8] M. Fernandes, T. O'Brien, and J. Lis, "Structure and Function of Heat Shock Gene Promoters," *In the Biology of Heat Shock Proteins and Molecular Chaperones*, R. I. Moromoto and C. Tissieres (eds), Cold Harbor Spring Press, 1994, pp. 375-393

[9] D. B. Fogel "The Advantages of Evolutionary Computation," *Bio-Computing and Emergent Computation,* D. Lundh, B. Olsson, and A. Narayanan (eds.), Sköve, Sweden, World Scientific Press, Singapore, 1997, pp. 1-11.

[10] D. B. Fogel, G. B. Fogel, and K. Ohkura, "Multiple-Vector Self-Adaptation in Evolutionary Algorithms," *BioSystems*, Vol. 61:2-3, 2001, pp. 155-162

[11] M. Glickman and K. Sycara, "Evolutionary Algorithms: Exploring the Dynamics of Self-Adaptation," *Genetic Programming 1998: Proceedings of the Third Annual Conference,* Morgan Kaufmann, San Francisco, CA, July 1998, pp. 762-769.

[12] C. Joslyn, "The Semiotics of Control and Modeling Relations in Complex Systems", *Biosystems*, v. 60:1-3, 2001, pp. 131-48

[13] S. Kumar, P. R. Cohen, H. J. Levesque, "The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams." *Proceedings of the Fourth International Conference on Multi-Agent Systems*, Boston, MA, USA, July 2000, pp. 159-166

[14] L. Lamport, "Time, Clock, and the ordering of Events in a Distributed Systems," *Communications of the ACM,* 21:7, 1978, pp. 558 - 565.

[15] M. M. Lu, Q. Zhang, and C. C. Lu, "Retrieval of Multimedia objects using color segmentation and dimension reduction of features," submitted for publication.

[16] N. Medvidovic, R. N. Taylor, and D. S. Rosenblum. "An Architecture-Based Approach to Software Evolution," In *Proceedings of the International Workshop on the Principles of Software Evolution*, Kyoto, Japan, April 1998, pp. 11-15

[17] S. Stepney, J. A. Clark, C. G. Johnson, D. Partridge, and R. E. Smith, "Artificial immune systems and the grand challenge for non-classical computation," *Proceedings of the 2003 International Conference on Artificial Immune Systems*, LNCS 2787, Springer, September 2003, pp. 204-216.

[18] K. Sycara, J.A. Giampapa, B.K. Langley, and M. Paolucci, "The RETSINA MAS, a Case Study, Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications," *Alessandro Garcia, Carlos Lucena, Franco Zambonelli, Andrea Omici, Jaelson Castro, ed., Springer-Verlag, Berlin Heidelberg, Vol. LNCS 2603*, July 2003, pp. 232—25