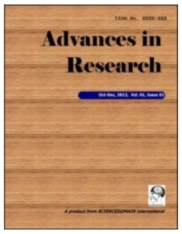


Advances in Research
2(4): 207-224, 2014, Article no. AIR.2014.003

SCIENCEDOMAIN *international*
www.sciencedomain.org

Infusing Intent and Its Management into Turing Machine: A Path to Cognitive Distributed Computing

Rao Mikkilineni[1*]

[1]C3 DNA, 2520 Mission College Boulevard, Suite 110, Santa Clara, California 95054, United States.

Author's contribution

This whole work was carried out by the author RM.

Original Research Article

Received 30th December 2013
Accepted 23rd February 2014
Published 13th March 2014

ABSTRACT

The holy grail of Artificial Intelligence (AI) has been the reproduction of the cognitive processes in Silicon based computing machines to mimic the human or animal cognitive behavior. Computationalism attempts to explain cognition in terms of different internal representations and symbol-manipulating processes operating over these representations. On the other hand, Connectionism explains cognition in terms of a finite automaton which computes associative function specified by a set of input-output pairs that in turn, introduce interaction into the computing model. However, both theories have critics who believe that they fall short in explaining the cognitive processes observed either in humans or animals. Computationalism tied to the Turing computing model which is restricted to single, sequential processes does not support concurrency, mobility and synchronization observed in cognitive processes. The connectionist systems are also dedicated to single tasks and do not support complex environments which require behaviors that are coordinated and integrated. Above all, cognition is associated with intent of a system and its accomplishment efficiently through various processes that monitor and control itself and its environment. Any computing model incorporating cognition must accommodate dynamic coupling between various elements of the system, where each change in one element continually influences every other element's direction of change. We discuss the newly introduced DIME (distributed intelligent managed element) computing model which is shown to be one of the implementing architectures for π–calculus and argue that its non-von Neumann parallel implementation of a managed

*Corresponding author: E-mail: rao@c3dna.com;

Turing machine with a signaling network overlay addresses some of the limitations of both Computationalism and Connectionism. The DIME network architecture provides a mechanism for injecting sensors and actuators into a Turing Machine and allows implementing autonomic distributed computing where the computers and the programs they execute are orchestrated to achieve the overall intent while optimizing the computing resources available.

Keywords: Computationalism; connectionism; cognitive process; turing machine; AI; DIME network architecture; genome; connectome; turing o-machine

1. INTRODUCTION

As McLaughlin [1] points out "The Holy Grail of the field of artificial intelligence is to build something with mental abilities out of something other than living tissue." This is a tall order given the mental abilities are supported by about hundred billion neurons each having tens of thousands of synapses interconnecting them. Each neuron in turn is designed over a long period of time evolving from a simple single-celled organism to the complex multi-celled organism with a gene describing the form and the function of each cell and how to orchestrate trillions of them constituting one individual. However, it is remarkable that both good old fashioned artificial intelligence (GOFAI) and connectionist models have made significant contributions with simplistic models to mimic a small set of mental abilities. GOFAI infers a logic-like internal representation system where inner states are symbols and the interactions are syntactic and formal. It is believed that, with enough explicitly coded knowledge, heuristics and axioms, a machine could develop intelligence. The Connectionist model, on the other hand, assumes that the information is stored in the connections among a network of nodes, and not the nodes themselves, making the knowledge implicit in the structure and not explicit in the individual nodes (in the case of brains, the individual units being neurons). Thus the information, or knowledge, if it has a physical existence, must exist within the networks of constantly firing neurons.

More recently, both models have been criticized as inadequate [2,3] to model the complex cognitive processes supported by the neural networks in the brain and suggest that a new model is required to address the embodied, embedded, enacted and extended mental processes involved in the living systems and their environment. In this paper we examine both the Computationalism and Connectionism approaches to AI and propose a new approach that goes a step further. We discuss the newly introduced DIME (distributed intelligent managed element) network architecture [4,5] modeled after a network of Turing O-machines with an overlay of a control network over the conventional Turing machine network and its expressive power to not only describe and execute a process but also include a control architecture that allows dynamic regulation and orchestration of various executing processes. We argue that this architecture provides a way to include the concept of self identity and model interactions between the self and the environment to provide autonomic process execution with available resources. In addition, the recursive network composition scheme which the model provides to create managed network of networks allows highly scalable dynamic process-flow implementation with global optimization of resources. The DIME network architecture provides both functional and non-functional behavior modeling and execution in an integrated fashion and enhances current autonomic computing models.
In Section II, we examine both Computationalism and Connectionism and their critique along with some thoughts on cognition and computing. In Section III, we present the DIME network

architecture and discuss its expressive power as one implementation architecture for π-calculus. In Section IV, we present ways to exploit the DIME network architecture to implement cognition in computers. In Section V, we conclude with some observations and remarks on future direction.

2. CURRENT MODELS OF ARTIFICIAL INTELLIGENCE

2.1 Computationalism

According to the computational theory of mind [2], all cognitive functions can be computed by algorithms – effective procedures. According to the Church–Turing thesis, every function that can be computed by an algorithm can be computed by a Turing machine. If the Church–Turing thesis is correct, then the computational theory of mind is committed to the thesis that every cognitive function is Turing-computable. This then, assures that all cognitive processes can be realized in silicon using the Universal Turing Machine (UTM) implementation with von Neumann serial stored program control computing model.

As Louis Barrett [6] points out, the theory gravitates towards those cognitive tasks like natural language, formal reasoning, planning, mathematics, and playing chess, in which the processing of abstract symbols in a logical fashion. It does not take into account an active organism's synergistic interactions of the mind, body and the environment where the notion of dynamic coupling (each change in one element of a system continuously influences every other element's change) is not taken into account. This observation is consistent with the conclusion [7] "the key property of general-purpose computer is that they are general purpose. We can use them to deterministically model any physical system, of which they are not themselves a part, to an arbitrary degree of accuracy. Their logical limits arise when we try to get them to model a part of the world that includes themselves". These observations show a need for a computing model which includes both functional and non-functional (related to the resource availability to execute the function) behavior.

2.2 Connectionism

A connectionist network is composed of interconnected units (or nodes). Individual units do all the information processing: there is thus no executive or CPU. There is, moreover, no program stored in memory; the program is implicit in the pattern of connectivity exhibited by the units. Many units process information simultaneously, and so the network as a whole engages in parallel distributed processing (PDP).

The Connectionism [2] suggests that the brain is a neural network that computes by non-symbolic means, principally by the summation of multiple inputs and the adjustment of connection strengths. While they can model temporal sequences, the standard connectionist models are not sufficiently powerful because they do not include reliable structure in the environment. In addition, "connectionist modelers tend to think in terms of single tasks and the most common forms of network are not good at handling multiple tasks which interact". The solution he suggests is "to recognize that the connectionist networks are finite automata and to use Turing's analysis to understand the role they should play in a theory of the mind".
According to Mark Rowlands [3] the Cartesian approaches based on rules and representations fall short to model the cognitive processes and perhaps other mental processes because they do not incorporate what organisms achieve by what they do in and to the world that is outside their brains –whether their bodies or the wider environment and

not just something occurring exclusively inside their brains. The Cartesian approach depends on information-bearing structures known as mental representations solely located in the brain of cognizing organisms about the world and their operations. He argues that the Cartesian approach is limited and does not address the external structures, which contain the information relevant to the accomplishing the task at hand. The cognitive processes often transform the information they contain by using such structures in appropriate ways. The cognitive processes also exploit detection of the information required for further evolution by using the external structures rather than storing or constructing it. He points out that control of external structures is an important part of the cognitive process, which presumes intent.

The long and short of the discussion is that we must understand cognitive processes and their execution in an organism before we can replicate it in Silicon. Current understanding based on Computationalism and Connectionism seems to fall short. Therefore, we must look elsewhere for further progress. Recent advances in biology and neuroscience address the processes of managing and safekeeping of life and developing intelligence and the relationships among them [8,9,10].

DNA encodes the information required to evolve a single cell carrying it into a fully functional organism with form and function designed to manage its life. According to Antonio Damasio [11], managing and safe keeping life is the fundamental premise of biological value and this biological value has influenced the evolution of brain structures. “Life regulation, a dynamic process known as homeostasis for short, begins in unicellular living creatures, such as bacterial cell or a simple amoeba, which do not have a brain but are capable of adaptive behavior. It progresses in individuals whose behavior is managed by simple brains, as in the case with worms, and it continues its march in individuals whose brains generate both behavior and mind (insects and fish being examples)”. Homeostasis is the property of a system that regulates its internal environment and tends to maintain a stable, constant condition of properties like temperature or chemical parameters that are essential to its survival. System-wide homeostasis goals are accomplished through a representation of current state, desired state, a comparison process and control mechanisms.

He goes on to say that “consciousness came into being because of biological value, as a contributor to more effective value management. But consciousness did not invent biological value or the process of valuation. Eventually, in human minds, consciousness revealed biological value and allowed the development of new ways and means of managing it”. The governance of life’s processes is present even in single-celled organisms that lack a brain and it has evolved to the conscious awareness, which is the hallmark of highly evolved human behavior. “Deprived of conscious knowledge, deprived of access to the byzantine devices of deliberation available in our brains, the single cell seems to have an attitude: it wants to live out its prescribed genetic allowance. Strange as it may seem, the want, and all that is necessary to implement it, precedes the explicit knowledge and deliberation regarding life conditions, since the cell clearly has neither. The nucleus and the cytoplasm interact and carry out complex computations aimed at keeping the cell alive. They deal with the moment-to-moment problems posed by the living conditions and adapt the cell to the situation in a survivable manner. Depending on the environmental conditions, they rearrange the position and distribution of molecules in their interior, and they change the shape of sub-components, such as microtubules, in an astounding display of precision. They respond under duress and under nice treatment too. Obviously, the cell components carrying out those adaptive adjustments were put into place and instructed by the cell’s genetic material.”

This vivid insight brings to light the cellular computing model that:

1. Spells out the computational workflow components as a stable sequence of patterns that accomplishes a specific purpose,
2. Implements a parallel management workflow with another sequence of patterns that assures the successful execution of the system's purpose (the computing network to assure biological value with management and safekeeping),
3. Uses a signaling mechanism that controls the execution of the workflow for gene expression (the regulatory network) and
4. Assures real-time monitoring and control (homeostasis) to execute genetic transactions of replication, repair, recombination and reconfiguration [12].

The specialized cell called the neuron plays a key role in defining organism's intelligence. With over a billion neurons, the brain is organized to perceive, think and carry out mental feats. According to Seung [10] the function of a neuron is defined chiefly by its (input and output) connections with other neurons and these connections explain memory and other mental phenomena, in addition to perception. He suggests that a map of neural connections, which he calls a connectome, defines uniquely each individual's memories, beliefs and behavior. Just as genome defines the evolution of the structure and function of the organism's body, the connectome define the organism's dynamical behavior executing cognitive processes.

2.3 Non-functional Requirements, Autonomic Computing Sensors, Actuators, and Turing Machines

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture. These requirements include availability, reliability, performance, security, scalability and efficiency at run-time. Autonomic computing is the ability of a system designed to adapt to change in accordance with business policies and objectives. In an autonomic environment, components work together, communicating with each other and with high-level management tools. They can manage or control themselves and each other. Components can manage themselves to some extent, but from an overall system standpoint, some decisions need to be made by higher level components that can make the appropriate trade-offs based on policies that are in place. The autonomic computing models suggest that self-management properties can be implemented if sensors and controllers are introduced into an element and the knowledge is provided for managing it [13]. Fig. 1 shows the MAPE-K (Monitor, Analyze, Plan, and Execute using Knowledge) model of autonomic computing. The monitoring and execution at run-time using the sensor and the actuator provide the ability to respond to environmental changes while the system is active. The autonomic computing model supports recursive (fractal) scale-invariant management processes that scale to manage a system of systems as long as sensors and actuators are available. This model has successfully been exploited to model and control many systems where functional requirements address business processes, robotics and other complex systems. However, addressing non-functional requirements which deal with the system resources such as computing resources, network resources and storage resources at run time is another matter. Different architectures and many implementations have contributed to the progress of the autonomic computing discipline [14].

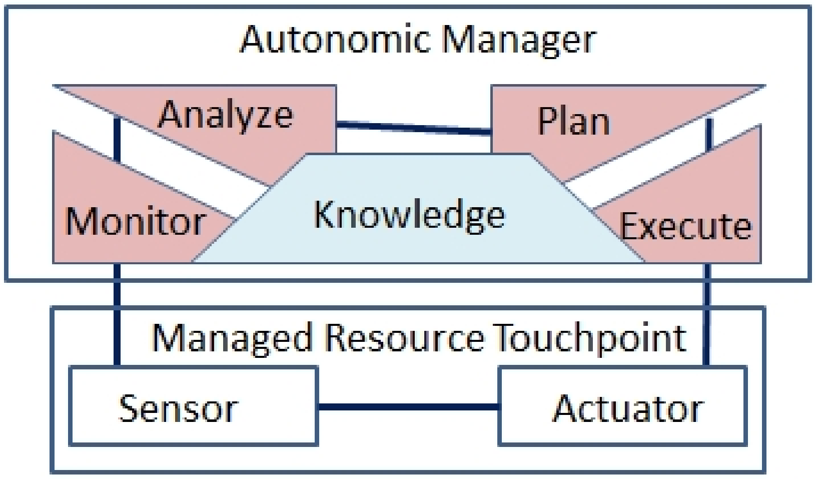


Fig. 1. The autonomic computing model

Object, component, service, and agent orientation paradigms have emerged as major paradigms for implementing autonomous distributed systems. Nonetheless, it is argued [15] that "none of these paradigms is able to adequately describe all kinds of distributed systems and inherent conceptual limitations exist." While each paradigm addresses some aspects of distributed computing, none of them are capable of supporting concurrency, distribution, and non-functional aspects such as availability, reliability, performance, scalability and end-to-end transaction security. These authors propose an extension by combining agent and service component architecture to create an active component which provides a hierarchical composition scheme to implement managed workflows. Another effort that defines a computing model that supports self-resiliency at both the node and network level is the FRACTAL component model [16]. However, all these ad hoc approaches to distributed computing depend on their implementations on an SPC (stored program control) implementation of the Turing machine which is constrained by its serial implementation of computations. In this paper we present a solution by introducing the sensors and actuators in the Turing machine using the Turing O-machine discussed in Turing's thesis [17] which allows autonomic management of non-functional requirements along with functional requirements.

2.4 Cognition and Computing

The cognitive processes are executed in the mind/brain system using a complex web of cellular structures that have been refined over a long period of time through evolutionary principles. Current neuroscience findings attribute the uniqueness of an individual to both the genome (which consists of a sequence of nucleotides in the individual cellular DNA) and the connectome [10] (which is a network of special cells called neurons.)

The genes are shorter segments in the genome which give the form and function of cellular structures which cooperate and collaborate through complex managed processes that provide a unique self with a mission that optimizes the resources available, interacts with its environment and evolves to preserve and propagate itself using the genetic transactions of replication, repair, recombination and reconfiguration [12]. The genes provide the definition of both the tasks and their management. As Mitchell Waldrop explains in his book on Complexity [18] “the DNA residing in a cell’s nucleus was not just a blue-print for the cell – a catalog of how to make this protein or that protein. DNA was actually the foreman in charge of construction. In effect, was a kind of molecular-scale computer that directed how the cell was to build itself and repair itself and interact with the outside world.” The genes equipped with the descriptions of tasks and their regulation, proceed to manufacture the body as soon as the full description of the genome is put together by the joining of the egg and the sperm. According to Dawkins [19] “The manufacture of a body is a cooperative venture of such intricacy that it is almost impossible to disentangle the contribution of one gene from that of another. A given gene will have many different effects on quite different parts of the body. A given part of the body will be influenced by many genes, and the effect of any one gene depends on interaction with many others. Some genes act as master genes controlling the operation of a cluster of other genes.”

In a very short period of time compared to the gestation period, the basic body plan of the embryo becomes established. During this time an apparently homogeneous group of cells, the inner cell mass of the blastocyst, becomes transformed into various parts consisting of specialized cells all arranged in correct positions relative to one another. Cells in different regions become switched on to different pathways and the evolution is orchestrated based on the local resources, functional requirements and interactions with other cells. The main point is that the description of the functioning of individual cells and the orchestration of groups of cells are given in the genetic DNA. Not only the function to be performed in each cell but also its context, constraints, communication, connections and control mechanisms are specified. In addition, a hierarchical composition scheme provides same abstractions involving a group of cells or groups of groups constituting the whole system. Armed with the full description of the genetic processes along with their regulation, the form and functions evolve from the embryo to create a multitude of cells with a multitude of specialized functions such as a liver cell, a heart cell etc. One specialized cell, the neuron plays a very special role in creating the essence of cognition and consciousness.

The managing and safekeeping of life efficiently are evident at the lowest level of biological architecture that provides the resiliency that von Neumann was discussing in his Hixon lecture [20]. ‘‘The basic principle of dealing with malfunctions in nature is to make their effect as unimportant as possible and to apply correctives, if they are necessary at all, at leisure. In our dealings with artificial automata, on the other hand, we require an immediate diagnosis. Therefore, we are trying to arrange the automata in such a manner that errors will become as conspicuous as possible and intervention and correction follow immediately’’. Comparing the computing machines and living organisms, he points out that the computing machines are not as fault tolerant as the living organisms. He goes on to say ‘‘It’s very likely that on the basis of philosophy that every error has to be caught, explained, and corrected, a system of the complexity of the living organism would not run for a millisecond.’’

The connection between consciousness and computing models is succinctly summarized by Samad and Cofer [21]. While there is no accepted precise definition of the term consciousness, “it is generally held that it is a key to human (and possibly other animal) behavior and to the subjective sense of being human. Consequently, any attempt to design automation systems with humanlike autonomous characteristics requires designing in some elements of consciousness. In particular, the property of being aware of one’s multiple tasks and goals within a dynamic environment and of adapting behavior accordingly.” They point to two theoretical limitations of formal systems that may inhibit the implementation of computational consciousness and hence limit our ability to design human-like autonomous systems. “First, we know that all digital computing machines are “Turing-equivalent”-They differ in processing speeds, implementation technology, input/output media, etc., but they are all (given unlimited memory and computing time) capable of exactly the same calculations. More importantly, there are some problems that no digital computer can solve. The best known example is the halting problem; we know that it is impossible to realize a computer program that will take as input another, arbitrary, computer program and determine whether or not the program is guaranteed to always terminate.

Second, by Gödel’s proof, we know that in any mathematical system of at least a minimal power there are truths that cannot be proven. The fact that we humans can demonstrate the incompleteness of a mathematical system has led to the claims that Gödel’s proof does not apply to humans”.

As pointed out in the introduction, Louis Barrett highlights [6] the difference between Turing machines implemented using von Neumann architecture and biological systems. “Although the computer analogy built on von Neumann architecture has been useful in a number of ways, and there is also no doubt that work in classic artificial intelligence (or, as it is often known, Good Old Fashioned AI: GOFAI) has had its successes, these have been somewhat limited, at least from our perspective here as students of cognitive evolution.” She argues that the Turing machines based on algorithmic symbolic manipulation using von Neumann architecture, gravitate toward those aspects of cognition, like natural language, formal reasoning, planning, mathematics and playing chess, in which the processing of abstract symbols in a logical fashion and leaves out other aspects of cognition that deal with producing adoptive behavior in a changeable environment. Unlike the approach where perception, cognition and action are clearly separated, she suggests that the dynamic coupling between various elements of the system, where each change in one element continually influences every other element’s direction of change has to be accounted for in any computational model that includes system’s sensory and motor functions along with analysis. To be fair, such couplings in the observed can be modeled and managed using a Turing machine network and the Turing network itself can be managed and controlled by another serial Turing network. What is not possible is the tight integration of the models of the observer and the observed with a description of the “self” using parallelism and signaling that are the norm and not an exception in biology.

3. THE DIME NETWORK ARCHITECTURE

The cognitive process execution and management needs to support, use and accommodate, among others, the following:

- Self-identity
- Recursive composition scheme and scale-invariant self-management process structures
- Sequential and parallel process flows,
- Dynamic provisioning of appropriate resources to assure process flow execution,
- Multiple dynamically assignable communication channels,
- Fault tolerance to process, communications or infrastructure failure,
- Process and infrastructure level security, and

- Run-time monitoring and control of resources to meet changing process priorities, latency constraints and workload-fluctuations caused by interactions with the environment.

In short, the cognitive process requires active management of dynamic coupling between various elements of the system, where each change in one element influences some other element's direction of change while its computation is still in progress and must be accounted for in any computational model. In addition, the resulting changes in computational resource requirements caused by the non-deterministic influences of various changes in the overall system have also to be dynamically managed.

The Turing machine is an abstract model that uses an instruction cycle,{read -> compute (change state) -> write} to "replace a man in the process of computing a real number (using a paper and pencil) by a machine which is only capable of finite number of conditions." In modern terms, a program provides a description of the Turing machine and the stored program control implementation in some hardware allows its execution. A universal Turing machine is also a Turing machine but with the ability to simulate a sequence of synchronous Turing machines each executing its own description. This allows a sequence of programs to model and execute a description of the physical world. However, the Turing's system is limited to single, sequential processes and is not amenable for expressing dynamic concurrent processes where changes in one process can influence changes in other processes while the computation is still in progress in those processes. Concurrent task execution and regulation require a systemic view of the context, resource constraints, communication, connections and control abstractions where the identities, autonomic behaviors and associations of individual components also must be part of the description. However, an important implication of Gödel's incompleteness theorem [22] is that it is not possible to have a finite description with the description itself as the proper part. In other words, it is not possible to read yourself or process yourself as a process. The DIME (distributed intelligent computing element) computing model [4, 5, 23] introduces three key modifications to the Turing machine to enable both the process execution and its management.

First, The {read -> compute -> write} instruction cycle of the Turing machine is modified to {interact with external agent -> read -> compute -> interact with external agent -> write} instruction cycle which allows the external agent to influence the further evolution of computation while the computation is still in progress.

Second, the external agent consists of a set of parallel managers monitoring and controlling the evolution of the computation based on the context, constraints and available computing resources (CPU and memory utilization, network bandwidth, external storage capacity and storage throughput). As discussed in [4,5,24,25], the context, constraints and control options "are specified as a meta-model of the algorithm under computation. The context refers to local resource utilization and the computational state of progress obtained through the interaction with the Turing machine. Each DIME contains two parts; the service regulator (SR) that specifies the algorithm context, constraints, connections, communication abstractions and control commands which are used to monitor and control the algorithm execution at run-time; and the algorithm executable module (SP) that can be loaded and run in the DIME.

Third, in addition to read/write communication of the Turing machine, the managers communicate with external agents using a parallel signaling channel. This allows the external agents to influence the computation in progress based on the context and constraints just as a Turing Oracle is expected to do. The external agent itself could be another DIME in which case, changes in one computing element could influence the evolution of another computing element at run time without halting its Turing machine executing the algorithm". Fig. 2 shows the DIME computing model [5]. DIME can be described as a managed Turing machine using a parallel management of the computing resources based on the knowledge about the process being executed, its context, constraints, communication, connections, and control abstractions. The fault management channel provides a heartbeat for each DIME which is used by the DIME self-management processes to implement auto-failover, auto-recovery and other repair processes based on the context and constraints. The accounting, performance and security channels are used to implement auto-scaling, auto-protection and other self-management processes.

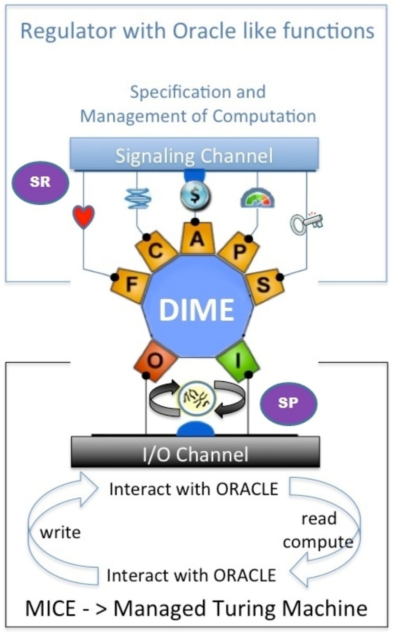
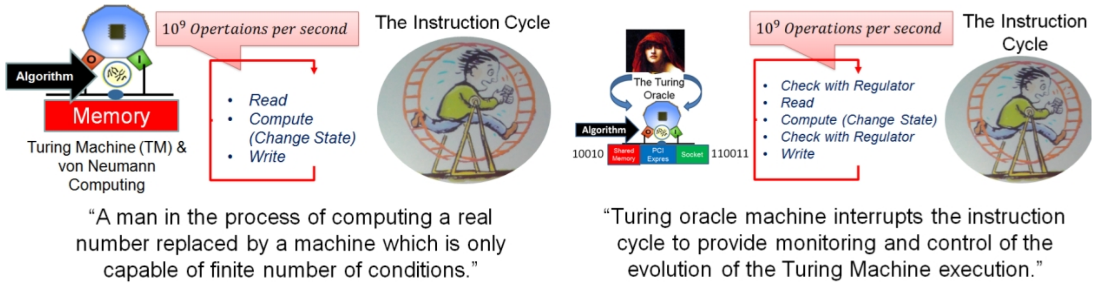

Regulator with Oracle like functions
Specification and Management of Computation
Signaling Channel
SR
C
A
P
F
S
DIME
O
I
SP
I/O Channel
Interact with ORACLE
write
read
compute
Interact with ORACLE
MICE - > Managed Turing Machine

Fig. 2. The Distributed Intelligent Managed Element (A Turing machine with modified instruction cycle and a parallel signaling network overlay modeling after the Turing O-machine

The configuration channel is used to implement DIME self-configuration and also cognitive processes required to manage the process being loaded and executed by the Managed Intelligent Computing Element (the Turing machine equivalent). The signaling channel overlaying the data channel enables a group of DIMEs to be interconnected to form a managed DIME computing network or just DIME network. The DIME network thus enables a network of DIMEs by integrating computing and communication abstractions in a single computing unit. It enables the creation of a network of MICEs executing a distributed computation and a network of managers that provide regulation of the distributed computing events. This allows the composition of hierarchical recursive (each node can itself be a sub-network) managed process implementation. Fig. 3 shows a comparison between the Turing instruction cycle and DIME instruction cycle.

A workflow is implemented as a set of tasks, arranged or organized in a directed acyclic graph (DAG) and executed by a managed network of DIMEs. These tasks, depending on the computation process requirements are programmed and executed as loadable modules in each DIME. The distributed software components along with associated profiles defining their use and management constraints are executed by DIMEs. The profiles are used as blueprints to setup, execute and control the down-stream DAG at each node based on global and local policies that depend on system-wide priorities, process resource utilization fluctuations and communication latency constraints.

In the next section we will discuss implementing cognitive processes using a DIME network. We show how the DIME network architecture provides tools to model various abstractions required not only in both Computationalism and Connectionism but also address other aspects of cognition such as embedded, embodied, enacted and extended cognitive processes.



"A man in the process of computing a real number replaced by a machine which is only capable of finite number of conditions."

"Turing oracle machine interrupts the instruction cycle to provide monitoring and control of the evolution of the Turing Machine execution."

Fig. 3. The instruction cycles in the Turing machine and the DIME computing model

4. INFUSING AUTONOMIC PROCESSES IN DISTRIBUTED COMPUTING: THE DIME NETWORK ARCHITECTURE (DNA)

DIME is like a stem cell whose function can be defined along with its context, constraints, communication, and control abstractions. The DIME at run time provides self-management and supports evolution of computation while computation is still in progress. It is also shown that the DIME computing model supports the four genetic transactions of replication, repair, recombination and reconfiguration [12,2].

The DIME network architecture Fig. 4, discussed in detail by the author [25] elsewhere, provides the following functions to implement cognitive processes using distributed computing machines to implement an intent involving the reciprocal influence of "bottom-up" and "top-down" processes:

1. Managing the "Life" of a Cognitive Process – Policy based resource assurance for system-wide process execution: Each DIME provides self-monitoring of resources and takes action based on local policies and an ability to load and execute a process using local operating system at run-time. A DIME network provides self-management of group-wide policies and system-wide policies also at run-time. This allows the decoupling of cognitive process management from the underlying infrastructure resources management using the replication, repair, recombination and reconfiguration properties of the DIME network architecture.

2. Instantiating a distributed process flow with scale-invariant composition and management abstractions: Each DIME has a unique identity upon instantiation. A regulator allows provisioning, provides heart-beat, security, performance and account management for each process it executes. The self-identity extends to network of DIME networks executing an intent.
3. A signaling scheme (addressing, alerting, mediation and supervision) to facilitate interaction with other DIMEs. The signaling allows same regulation features as the DIME for a group of DIMEs (sub-network and network level).
4. Supporting [3] embodied, embedded, enacted and extended process flows using a DIME network with a system-wide awareness: The DIME network provides a natural framework for implementing reliable managed cognitive processes that are embodied (partly constituted by distributed and specialized structures and processes). Individual processes at the leaf level may be embedded (designed to function only in tandem with system's environment). By interacting with the environment through embedded processes, the DIME network supports cognitive processes that are enacted. The distributed nature and interaction with environment provides the execution of extended cognitive processes using the recursive composition capability of the DIME network.
5. Dynamic process Management: The run-time monitoring of both resource utilization and process execution along with signaling capabilities allows policy based intervention in each DIME while computation is still in progress using its Turing O-machine like behavior of each DIME.

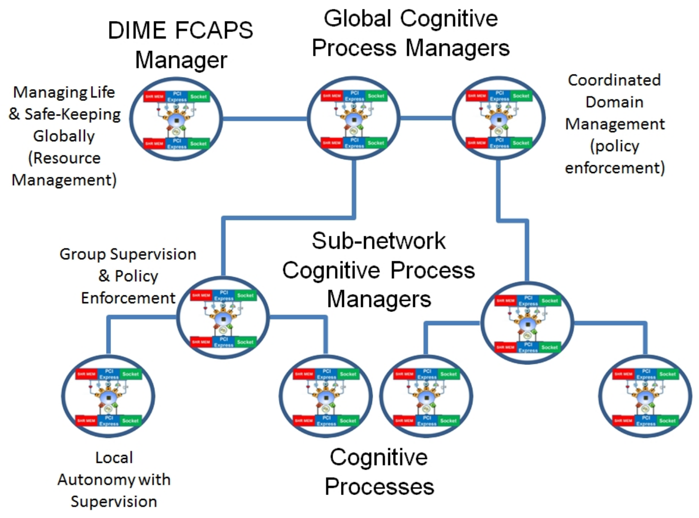

DIME FCAPS Manager
Global Cognitive Process Managers
Managing Life & Safe-Keeping Globally (Resource Management)
Coordinated Domain Management (policy enforcement)
Group Supervision & Policy Enforcement
Sub-network Cognitive Process Managers
Local Autonomy with Supervision
Cognitive Processes

Fig. 4. A distributed cognitive process using DIME network architecture [25]

We argue that the DIME network architecture provides modeling capabilities with high degree of scaling, resilience and efficiency to represent various aspects of both Computationalism and Connectionism. It provides the right functionality for individual DIMEs by creating a managed DIME network with appropriate local, sub-network and network – wide policies to ensure that the intent is executed. The ability to change the evolution of computation in progress in any Turing machine using the new instruction cycle provides self-management properties using the descriptions of process execution and its management. The Turing O-machine like control circumvents the limitations on Turing machines imposed by Gödel's theorems. The self-identity at the element level, sub-network level and network level provides the expressive power of ownership [25] for the cognitive process with appropriate resource monitoring and policy-based control of safety and survival while executing the intent.

The signaling network overlay over the Turing machine network provides a self-managed recursive distributed computing engine which is highly scalable, resilient and efficient to address many of the problems amenable to Connectionism. Each DIME with many connections to other DIMEs in a network provides some of the modeling capabilities of neural networks. For example the connection strengths can determine their affinity. In addition, the scale-invariant FCAPS management at the element, sub-network and the network level allows the expressive power of supervision of distributed, embodied, embedded, enacted, and extended cognitive processes [25].

Details of DIME network architecture implementation demonstrating two use cases, one in cloud computing and another in a multi-core server are published else-where [24]. In one implementation, a Linux process is converted into a DIME to demonstrate auto-scaling, self-repair, load-balancing, live-migration and security management without a Hypervisor or interfaces to infrastructure management systems at run-time. In another implementation a core in a multi-core server is converted to a DIME using a new operating system to execute a program. A network of DIMEs is used to demonstrate FCAPS management of a workflow. More recently, Goyal [5, 26] have shown that the DIME network architecture is one of the implementation architectures for π-calculus to implement mobility, synchronization and concurrency.

Currently, the work is being extended to implement ontology based workflows using the service regulator to define, execute and control tasks constituting the workflow [27]. Ontologies are computational formalizations of the knowledge about a given domain, allowing computers to manage the information at a semantic level. In the semantic web (which is a means to build a World-Wide Web where the semantic content is accessible for computers, not just for the human users) ontologies are the mechanism for providing a vocabulary that will describe data held in a common data model. The vocabulary and the semantics provided by the ontology all facilitate machine processing. As Aranguren [28] points out, the knowledge representations using ontologies are used for different functions, such as web agents [29] and web services [30], GRID technology [31], e-commerce [32], data mining and text mining [33,34] and computer security [35] among others.

In order to inject intelligence into machines, a cognitive process must cope with a set of predefined rules along with complex decision making procedures, which calls for a solid knowledge base in which entities are captured from the real-world but distilled into machine readable models. The models must capture the descriptions of the entities, their behaviors including the context, constraints and relationships between the entities, communications and abstractions for their control. In this section we discuss how the DIME net-work architecture describes the intent, the regulatory behavior and the execution behavior of a cognitive process using their ontological descriptions.
The approach we take in our use of ontologies is to focus on the intent of the cognitive process and its execution. Taking the cue from living organisms, we select two main objectives driving the design:

1. First, imparting a sense of self with ownership of the intent to a cognitive process.

2. The second is to describe both the execution of the temporal and hierarchical process flows required to accomplish the intent but also their regulation using management process flows.

The self-management consists of:

1. Fault management: A heartbeat is imparted to each DIME on instantiation that is used for fault management.
2. Configuration management: The ability to execute commands to configure the context, constraints, communication, connections and control abstractions specified for each process loaded and executed in the DIME provides dynamic control of the evolution of computational task contributing to the intent of the cognitive process
3. Account management: Each DIME's contributions to the overall cognitive process are ac-counted for with appropriate ownership and other attributes.
4. Performance management: The resource utilization is monitored to assure the task at hand which is contributing to overall cognitive process based on global priorities, workload variations and latency constraints. This provides a vehicle to allocate appropriate resources to appropriate tasks in overall cognitive process execution.
5. Security management: The Turing oracle nature of DIME allows monitoring and enforcing security in Turing Machine's instruction cycle. This allows task security to be monitored and matched with global cognitive process security (based on ownership and identity).

The DIME computing model provides a very convenient container to model a self-managed computing engine that executes a managed task. It allows the description that contains the execution specification along with its intent showing the context, communication, constraints, connections and control of the task being executed. The Turing oracle nature of the computing element provides dynamic control of the computational flow during execution by modifying the instruction cycle.

5. CONCLUSION

The DIME computing model inserts sensors and actuators into the Turing machine following the clue from the Turing O-machine. It infuses the intent of the function in each computing element using a meta-model of the intent with context, constraints, communications, connections, and control and implementing them using a managed Turing machine with a non-von Neumann parallel regulation. The FCAPS manager and the modified instruction cycle of the Turing machine provide the best practices to manage life of computation (which depends on the resources available) and safe-keeping [11]. The signaling overlay and parallel distributed execution model with support for replication, repair, recombination and reconfiguration provide additional expressive power to evolve changes in one process element executing a function based on changes in other elements while computation is in progress. The recursive composition and scale invariant management structure processes provide the ability to implement embodied, embedded, enacted and extended cognitive processes with intent and ownership to assure successful execution with optimal resource management. Both these capabilities are possible to be implemented today because of the new computing power and high bandwidth made possible by multi-core chip design and high bandwidth networks.

It is important to note that the DIME network architecture is agnostic to operating systems or underlying hardware and a DIME is a virtual managed service execution container we call

Cognitive Container that can be implemented in computers in different forms as is described in [4]. High degree of scaling spanning across multiple devices and geographies and the ontology based approach to define and execute managed temporal and hierarchical sequences of events are suited to model both computational and connection models of reasoning using the recursive network composition scheme to design and deploy DIME networks on distributed servers. The signaling network overlay allows run-time end-to-end service visibility and control while the Turing O-machine like behavior allows distributed service transaction management independent of the infrastructure management systems. Thus the DIME network architecture provides the architectural resiliency of cellular organisms by providing safety and survival of the service and its components through replication, repair, recombination and reconfiguration of the cognitive containers and providing history of the service transaction for sectionalizing, isolating, diagnosing and fixing the problems in the distributed infrastructure at leisure.

While proofs of concept have been implemented to demonstrate feasibility [4,24], we understand that this is a bold new approach that goes against current assertions in computer science [36]. Thus the DIME network architecture is in its infancy and requires both theoreticians and implementers to validate or disprove these assertions before it can be useful in mission critical environments. As pointed out in [5], our prototypes demonstrate that the DIME network architecture enables the self-management of a response to the ephemeral nature of distributed computing caused by the non-deterministic impact of environmental interaction with the system. It has the potential to replace current ad-hoc approaches to distributed cognitive computing systems. Most recently, the DIME network architecture has been used to demonstrate a distributed web services delivery using a Linux, Apache, MySQL, PHP suite [37] deployed in a distributed computing environment consisting of physical and virtual servers shown in Fig. 5. The policy based end-to-end service management allows self-repair, auto-scaling, live-migration and end-to-end transaction security using the mobility of the containers executing the applications. The authors discuss container mobility across physical servers and virtual servers distributed in different datacenter and cloud infrastructure using the end to end service visibility and control using the DIME network architecture.

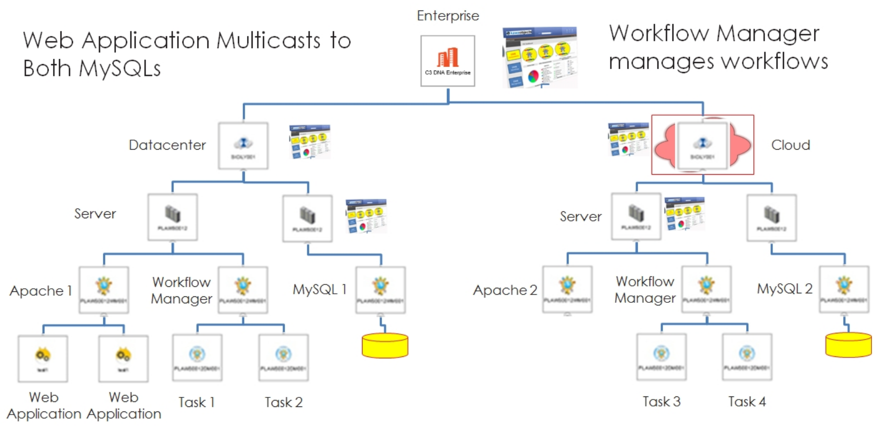


Fig. 5. A DIME Network deploying multiple replicated and load-balanced services withauto-scaling, self-repair, Live Migration with multi-casting of file/device writes with transaction consistency and end-to-end service transaction security assurance at run-time

Author was made aware of certain similarity to other work by Simeonov [38,39] which also point out the usefulness of the knowledge of a node in the network about its neighbors and environment in serving the overall intent of the networked community of nodes. DIME computing model uses the intent to define the function, structure and relationships to deliver the intent of the overall system.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. McLaughlin B. Computationalism, Connectionism, and the philosophy of mind. In: Floridi L, (Ed.). The Blackwell guide to philosophy of computing and information. Malden: Blackwell; 2004.
2. Wells A. Rethinking Cognitive computation: Turing and the science of mind. Palgrave Macmillan: London; 2006.
3. Rowlands M. The new science of the mind. The MIT Press: Cambridge; 2010.
4. Mikkilineni R. Designing a new class of distributed systems. Springer: New York; 2011. ISBN: 1461419239.
5. Mikkilineni R, Comparini A, Morana G. The Turing O-machine and the DIME network architecture: Injecting the architectural resiliency into distributed computing. In Turing-100. The Alan Turing Centenary, (Ed.) Andrei Voronkov, Easy Chair Proceedings in Computing; 2012;10.
   Available: http://www.easychair.org/publications/?page=877986046
6. Barrett L. Beyond the brain. Princeton University Press: Princeton; 2011.
7. Cockshott P, MacKenzie LM, Michaelson G. Computation and its limits. Oxford University Press, Oxford; 2012.
8. Carroll SB. The new science of EvoDevo - Endless Forms Most Beautiful. W. W. Norton & Co: New York; 2005.
9. Damasio A. The feeling of what happens: Body and emotion in the making of consciousness. New York, NY: Harcourt & Company; 1999.
10. Seung S. Connectome: How the brain makes us who we are. Houghton Mifflin Harcourt: Boston; 2012.
11. Damasio A. Self comes to mind: Constructing the conscious brain. New York: Pantheon Books; 2010.
12. Stanier P, Moore G. Embryos, Genes and Birth De-fects, (2nd Edition), Edited by Patrizia Ferretti, Andrew Copp, Cheryll Tickle, and Gudrun Moore, London, John Wiley & Sons. 2006;5.
13. Huebscher MC, McCann JA. A survey of autonomic computing—degrees, models, and applications. ACM Computing Surveys. 2008;40(3):7.
14. Denko MK, Yang LT, Zhang Y. Autonomic computing and networking. New York, NY, Springer; 2009.
15. Braubach L, Pokahr A. Addressing challenges of distributed systems using active components. In Intelligent Distributed Computing V, New York, NY: Springer; 2011.
16. Bruneton E, Coupaye T, Leclercq M, Quéma V. The FRACTAL component model and its support in Java. Softw Pract Exper. 2006;36:1257-1284.
17. Turing AM. Edited by Copeland JB. The essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence and Artificial Life: Plus the Secrets of Enigma", New York: The Oxford University Press; 2004.

18. Waldrop, Mitchell M. Complexity: The emerging science at the edge of order and chaos. Penguin Books, London; 1992;218.
19. Dawkins R. The selfish gene, Oxford University Press, Oxford; 1989.
20. Von Neumann J. Papers of John von Neumann on computing and computing theory. Hixon Symposium, September 20, 1948, Pasadena, CA, The MIT Press, Massachusetts. 1987;474.
21. Samad T, Cofer T. Autonomy and automation: Trends, technologies, In Gani R, Jørgensen SB, (Ed.). Tools in European symposium on computer aided process engineering Amsterdam. Netherlands: Elsevier Science BV. 2001;11.
22. Gödel K. Monatsheftefür Mathematic und Physik Leipzig. 1931;38:173-198.
23. Mikkilineni R, Morana G, Zito D, Di Sano M. Service virtualization using a non-von neumann parallel, distributed and scalable computing model. Journal of Computer Networks and Communications; 2012.
24. Mikkilineni R, Morana G, Seyler I. Implementing distributed, self-managing computing services infrastructure using a scalable, parallel and network-centric computing model. In Villari M, Brandic CI, Tusa F. Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice IGI Global. 2012;57-78.
25. Mikkilineni R. Going beyond computation and Its limits: Injecting cognition into computing. Applied Mathematics. 2012;3(11):1826-1835. doi: 10.4236/am.2012.331248.
26. Goyal P, Mikkilineni R. Implementing managed loosely-coupled distributed business processes: A new approach using dime networks. Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 21st IEEE International Conference, Toulouse. 2012;25-27.
27. Wang L, Mikkilineni R. Private Communication; 2013.
28. Aranguren ME. Ontology design patterns for the formalisation of biological ontologies. M. Phil Thesis, University of Manchester; 2005.
29. Hendler J. Agents and the semantic web. IEEE Intelligent Systems Journal. 2001;16:30–37.
30. Gibbins N, Harris S, Shadbolt N. Agent-based semantic web services. Journal of Web Semantics. 2004;1(1):141–154.
31. Stevens RD, Robinson AJ, Goble CA. MyGrid: personalised bioinformatics on the information grid. Bioinformatics. 2003;19:302–304.
32. Kwon OB, I know what you need to buy: context-aware multimedia-based recommendation system. Expert Systems with Applications. 2003;25:387–400.
33. Li Y, Zhong N. Web mining model and its applications for information gathering. Knowledge-Based Systems; 2004.
34. Kim JD, Ohta T, Tateisi Y, Tsujii J. GENIA corpus - a semantically annotated corpus for bio-textmining. Bioinformatics. 2003;19:180–182.
35. Lin SC, Tseng SS. Constructing detection knowledge for DDoS intrusion tolerance. Expert Systems with applications. 2004;27:379–390.
36. Eberbach E, Mikkilineni R, Morana G. Computing models for distributed autonomic clouds and grids in the context of the DIME Network Architecture, Proc. of 21st IEEE Intern. Conf. on Collaboration Technologies and Infrastructures WETICE Track on Convergence of Distributed Clouds. Grids and Their Management, Toulous, France. 2012;25-27.
37. Morana G, Mikkilineni R. An implementation of end-to-end service visibility and control using DIME network architecture to provide self-repair, auto-scaling, live migration and transaction security across distributed computing infrastructures. Submitted to WETICE, Parma, Italy; 2014.

38. Simeonov PL. The wandering logic intelligence, a hyperactive approach to network evolution and its application to adaptive mobile multimedia communications. Ph.D. Thesis; 2002.
39. Ehresmann AC, Simeonov PL. WLIMES: Towards a theoretical framework for wandering logic intelligence memory evolutive systems. In: Integral Biomathics: Tracing the Road to Reality, Simeonov PL, Smith LS, Ehresmann AC, (Eds.). Springer-Verlag; 2012.