Applied Mathematics, **2012, 3, 1826-1835**
http://dx.doi.org/10.4236/am.2012.331248 Published Online November 2012 (http://www.SciRP.org/journal/am)

Going beyond Computation and Its Limits: Injecting Cognition into Computing

Rao Mikkilineni
Kawa Objects, Los Altos, USA
Email: rao@kawaobjects.com

Received August 7, 2012; revised September 7, 2012; accepted September 15, 2012

ABSTRACT

Cognition is the ability to process information, apply knowledge, and change the circumstance. Cognition is associated with intent and its accomplishment through various processes that monitor and control a system and its environment. Cognition is associated with a sense of "self" (the observer) and the systems with which it interacts (the environment or the "observed"). Cognition extensively uses time and history in executing and regulating tasks that constitute a cognitive process. Whether cognition is computation in the strict sense of adhering to Turing-Church thesis or needs additional constructs is a very relevant question for addressing the design of self-managing (autonomous) distributed computing systems. In this paper we argue that cognition requires more than mere book-keeping provided by the Turing machines and certain aspects of cognition such as self-identity, self-description, self-monitoring and self-management can be implemented using parallel extensions to current serial von-Neumann stored program control (SPC) Turing machine implementations. We argue that the new DIME (Distributed Intelligent Computing Element) computing model, recently introduced as the building block of the DIME network architecture, is an analogue of Turing's O-machine and extends it to implement a recursive managed distributed computing network, which can be viewed as an interconnected group of such specialized Oracle machines, referred to as a DIME network. The DIME network architecture provides the architectural resiliency, which is often associated with cellular organisms, through auto-failover; auto-scaling; live-migration; and end-to-end transaction security assurance in a distributed system. We argue that the self-identity and self-management processes of a DIME network inject the elements of cognition into Turing machine based computing as is demonstrated by two prototypes eliminating the complexity introduced by hypervisors, virtual machines and other layers of ad-hoc management software in today's distributed computing environments.

Keywords: Cognition; Cognitive Process; Computationalism; Turing Machine; Turing O-Machine; DIME; DIME Network Architecture

1. Introduction

"*It is a fundamental problem of science, and whether we study Gödel or Penrose, Lucas or Hofstadter, Searle or Dennett, everyone agrees that the basic question is whether human-minds are super-mechanical, though there is widespread disagreement about the answer*."[1]

Cockshott *et al.* [1] conclude their book "Computation and its limits" with the paragraph "*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*." While the last statement is not strictly correct (for example current operating systems facilitate incorporating computing resources and their management interspersed with the computations that attempt to model any physical system to be executed in a Turing machine), it still points to a fundamental limitation of current Turing machine implementations of computations using the serial von Neumann stored program control computing model. The universal Turing machine allows a sequence of connected Turing machines synchronously model a physical system as a description specified by a third party (the modeler). The context, constraints, communication abstractions and control of various aspects during the execution of the model (which specifies the relationship between the computer acting as the observer and the computed acting as the observed) cannot be also included in the same description of the model because of Gödel's theorems of incompleteness and decidability.

This paper begins where their book ends by proposing

[1]Andrew Hodges. "Alan Turing: An Introductory Biography" In: Christof Teuscher, Alan Turing: Life and Legacy of a Great Thinker, NY: Springer, 2004, p. 50.

a way to push the computation beyond its current limits circumventing the Gödel's prohibition on self-reflection in computing systems. The limitations of computers that he helped design were very much on John von Neumann's mind, who, spent a great deal of time thinking about designing reliable computers using unreliable components [2]. In the Silliman lectures, and in the Hixon symposium [3], he touches upon various shortcomings in the computing model discussing how the computers behave differently from cellular organisms. Cellular organisms are autonomic. As von Neumann pointed out "*It is 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 last for a millisecond. Such a system is so integrated that it can operate across errors. An error in it does not in general indicate a degenerate tendency. The system is sufficiently flexible and well organized that as soon as an error shows up in any part of it, the system automatically senses whether this error matters or not. If it doesn't matter, the system continues to operate without paying any attention to it. If the error seems to the system to be important, the system blocks that region out, by-passes it, and proceeds along other channels. The system then analyzes the region separately at leisure and corrects what goes on there, and if correction is impossible the system just blocks the region off and by-passes it forever. The duration of operability of the automation is determined by the time it takes until so many incurable errors have occurred, so many alterations and permanent by-passes have been made, that finally the operability is really impaired. This is completely different philosophy from the philosophy which proclaims that the end of the world is at hand as soon as the first error has occurred.*"

Autonomic computing, by definition implies two components in the system: 1) the observer (or the "self") and 2) the observed (or the environment) with which the observer interacts by monitoring and controlling various aspects that are of importance. It also implies that the observer is aware of systemic goals in terms of best practices, to measure and control its interaction with the observed. Autonomic computing systems attempt to model system wide actors and their interactions to monitor and control various domain specific goals also in terms of best practices. However, cellular organisms take a more selfish view of defining their models on how they interact with their environment. The autonomic behavior in living organisms is attributed to the "self" and "consciousness" which contribute to defining one's multiple tasks to reach specific goals within a dynamic environment and adapting the behavior accordingly.

The autonomy in cellular organisms comes from three sources:

1) Genetic knowledge that is transmitted by the survivor to its successor in the form of executable workflows and control structures that describe stable patterns to optimally deploy the resources available to assure the organism's safe keeping in interacting with its environment.

2) The ability to dynamically monitor and control organism's own behavior along with its interaction with its environment using the genetic descriptions and

3) Developing a history through memorizing the transactions and identifying new associations through analysis.

In short, the genetic computing model allows the formulation of descriptions of workflow components with not only the content of how to accomplish a task but also provide the context, constraints, control and communication to assure systemic coordination to accomplish the overall purpose of the system. That the machine learning need to mimic the learning behavior of at least the children to go beyond the mere book-keeping possible with the Turing machine limitations was not lost on Turing as he points this out explicitly [4,5].

"*In the process of trying to imitate an adult human mind we are bound to think a good deal about the process which has brought it to the state that it is in. We may notice three components*:

1) *The initial state of the mind, say at birth*;

2) *The education to which it has been subjected*;

3) *Other experience, not to be described as education, to which it has been subjected.*

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationer's. Rather little mechanism, and lots of blank sheets (Mechanism and writing are from our point of view almost synonymous). Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child."

However, the child's mind already comes with a genetic description of both execution and regulation models supporting the genetic transactions [6] of replication, repair, recombination and reconfiguration that go far beyond the capabilities of a general purpose computer implementing Turing computations. Self-management and interaction regulation capabilities are way beyond the "little mechanism" albeit with plenty of blank paper to write new programs. Even before the child is born, as soon as the sperm and the egg combine to form the complete single cell, the genes provide the complete description of how to survive not only by managing the self but also to interact with the environment and a lot of blank pages to makeup own rules based on the context and constraints. According to Sean Caroll [7], there are two factors that define the form and function of the new cellular organism that contains the genetic description. "*The development of form depends on the turning on and off of genes at different times and places in the course of development. Differences in form arise from evolutionary changes in where and when genes are used*, *especially those genes that affect the number*, *shape*, *or size of structure*." In addition a class of genetic switches regulates how the genes are used and play a great role in defining the function of the living organism.

While Alan Turing and John von Neumann both looked at the computing model analogies with neural networks, and discussed hierarchical schemes to circumvent the consequences of Gödel's theorems on the limitations of Turing machines, they could not have foreseen the current hardware breakthroughs that provide parallel computation threads in many-core processors with a hierarchy of high-bandwidth connections between the computing elements. In this paper, we describe an extension of the von Neumann stored program serial implementation of the Turing machine network using the same abstractions of self-management and regulation that provide the elegant execution of life's workflows with appropriate context, constraints, control and communication processes. It exploits the performance, parallelism and high bandwidth networks available in the new generation processors to inject real-time cognition into Turing computing machines. In Section 2, we briefly review current arguments about cognition and computing and come on the side of cognition is more than computing. We identify the basic abstractions that are instrumental in providing the self-management features that capture the behavior of the observer and the observed with optimal resource utilization in a dynamic non-deterministic environment. In Section 3, we argue that the new DIME network architecture recently introduced injects the self-management features in a Turing machine and allows building autonomic distributed systems where the computer and the computed interact with each other pushing the boundaries of Turing machines. We argue that the DIME is analogous to an O-Machine introduced by Turing in his thesis [8,9] and the DIME network architecture provides a model for a distributed recursive computing engine that allows replication, repair, recombination and reconfiguration of computing elements to implement dynamic self-managing distributed systems. In Sections 4 and 5, we discuss the impact of DNA on distributed systems design with visibility and control of the observer (the computation that is managing resources) and the observed (the computed). In Section 6, we conclude with some observations on injecting cognition into computing.

2. Cognition and Computing

An autonomous system is typically considered to be a self-determining system, as distinguished from a system whose behavior is explicitly externally engineered and controlled. The concept of autonomy (and autonomous systems) is, therefore, crucial to understanding cognitive systems. According to Maturana [10,11] a cognitive system is a system whose organization defines a domain of interactions in which it can act with relevance to the maintenance of itself, and the process of cognition is the actual (inductive) acting or behaving in this domain. If a living system enters into a cognitive interaction, its internal state is changed in a manner relevant to its maintenance, and it enters into a new interaction without loss of its identity. A cognitive system becomes an observer through recursively generating representations of its interactions, and by interacting with several representations simultaneously it generates relations with the representations of which it can then interact and repeat this process recursively, thus remaining in a domain of interactions always larger than that of the representations. In addition, it becomes self-conscious through self-observation; by making descriptions of itself (representations), and by interacting with the help of its descriptions it can describe itself describing itself, in an endless recursive process.

According to Evan Thompson [12], autonomic systems exhibit dynamic co-emergence. Emergence describes the arising of large-scale, collective patterns of behavior in complex systems as diverse as cells, brains, ecosystems, cities, and economies. Emergence is closely related to self-organization and circular causality, both of which involve the reciprocal influence of "bottom-up" and "top-down" processes. Dynamic co-emergence means that a whole not only arises from its parts, but the parts also arise from the whole. Part and whole co-merge and mutually specify each other. A whole cannot be reduced to its parts, for the parts cannot be characterized independently of the whole; conversely, the parts cannot be reduced to the whole, for the whole cannot be characterized independently of the parts.

These observations lead us to conclude that self-management is an outcome of cognitive abilities of a system with the following defining attributes of cognitive systems:

1) A self-identity that does not change when a state change occurs with interaction;

2) A domains of interaction;

3) A cognitive interaction process support that allows an observer to generate recursively representations of its interactions. The observer by interacting with several representations simultaneously, generates relations with the representations of which it can then interact and repeat this process recursively, thus remaining in a domain

of interactions always larger than that of the representations, and

4) Co-emergence

In the next section we will discuss the Turing O-machine and argue that it is more suitable to simulate the cognitive activity and such a simulation transcends the mere book-keeping capabilities of a Turing machine.

3. Turing O-Machine and the Scale Invariant Structure Processes

Extending the three mutually exclusive positions discerned by Johnson-Laird [13] which are the alternatives to the conclusion "consciousness is not 'scientifically explicable'", Copeland [9] introduces Turing's O-machine as an alternative to model brain or the brain's cognitive activity. The five alternatives that Copeland discusses are:

1) The human brain (or, variously, mind or mindbrain) is a computer, equivalent to some Turing machine;

2) The activity of a human brain can be simulated perfectly by a Turing machine but the brain is not itself a computing machine;

3) The brain's cognitive activity cannot in its entirety be simulated by a computing machine: a complete account of cognition will need "to rely on non-computable procedures";

4) The brain is what Turing called an O-machine; and

5) The cognitive activity of the brain can be simulated perfectly by an O-machine, but the brain is not itself an O-machine; such simulation cannot be effected by a Turing machine.

In this paper we argue that the DIME network architecture introduced to inject architectural resiliency in distributed computing systems [14,15] supports the fifth alternative introduced by Copeland.

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 as Cockshott *et al.* [1] point out. 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 which is an essential requirement for describing cognitive processes. Concurrent task execution and regulation require a systemic view of the context, constraints, communication and control 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 [3] 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.

Turing himself discussed the mathematical objection to his view that machines could think [16,17]. In reply to the objection, he proposed designing computers that could learn or discover new instructions, overcoming the limitations imposed by Gödel's results in the same way that human mathematicians presumably do. He also pointed out [18] that while Gödel's theorem shows that every system of logic is in a certain sense incomplete, it also "*indicates means whereby from a system L of logic a more complete system L_ may be obtained. By repeating the process we get a sequence L, L1 = L_, L2 = L_1 … each more complete than the preceding. A logic Lω may then be constructed in which the provable theorems are the totality of theorems provable with the help of the logics L, L1, L2, … Proceeding in this way we can associate a system of logic with any constructive ordinal. It may be asked whether such a sequence of logics of this kind is complete in the sense that to any problem A there corresponds an ordinal α such that A is solvable by means of the logic Lα.*" He also introduced the Oracle machine in his thesis but stopped short of injecting cognition into computing. "*An O-machine is like a Turing machine (TM) except that the machine is endowed with an additional basic operation of a type that no Turing machine can simulate.*" Turing called the new operation the "Oracle" and said that it works by "some unspecified means". When the Turing machine is in a certain internal state, it can query the Oracle for an answer to a specific question and act accordingly depending on the answer. The O-machine provides a generalization of the Turing machines to explore means to address the impact of Gödel's incompleteness theorems and problems that are not explicitly computable but are limit computable using relative reducibility and relative computability [19]. The Oracle-machine influenced many theoretical advances including the development of generalized recursion theory that extended the concept of an algorithm [19,20].

In this paper we argue that the DIME network architecture recently introduced [14] incorporates a "regulatory" function to exert external influence on a Turing machine while computation is still in progress (and has not halted yet), making it act more like an O-machine. A network of such "regulated" Turing machines acts like a managed recursive distributed computing engine with nested monitoring and control functions where each level is managed by the Oracle-like machine at a higher level. The resulting architecture allows descriptions of dynamic, scale-invariant structure processes to represent the reciprocal influences of "bottom-up" and "top-down" processes. With the introduction of the Turing O-machine-like regulation, the DIME network architecture circumvents both the halting and un-decidability problems by pushing the knowledge about the context, constraints and control of the computation up the hierarchy which regulates the sequence of hierarchical and temporal events required to implement homeostasis and self-management of the computation. At the root level, the process work-flow down the chain defines the stable computing patterns that execute the events to accomplish the system's purpose and the goals specified at each level. The specification of the system's purpose (or the intent that drives a cognitive process) at the root level (initial conditions at t = 0) is regulated by an external agent in terms of the context and constraints that define the destiny of the process flow. This architecture, resembling the self-organizing fractal structure [21,22] is suited to address some of the concerns currently afflicting distributed computing systems such as concurrency, mobility and synchronization. Further research is in progress to provide a way to implement features of π-calculus [23,24] including mobility using the DIME network architecture.

The DIME network architecture concerns itself with process work-flows that contain the descriptions to execute and regulate the tasks described to accomplish an intent. When the process is initiated by an external agent at t = 0, the whole and the parts act as an integrated system to accomplish the intent with the given descriptions of both the task executions and their regulation.

4. DIME Network Architecture and Cognitive Process Implementation

In its simplest form a DIME is comprised of a policy manager (determining the fault, configuration, accounting, performance, and security aspects often denoted by FCAPS); a computing element called MICE (Managed Intelligent Computing Element); and two communication channels. The FCAPS elements of the DIME provide setup, monitoring, analysis and reconfiguration based on workload variations, system priorities based on policies and latency constraints. They are interconnected and controlled using a signaling channel which overlays a computing channel that provides I/O connections to the MICE) [14]. The DIME computing element acts like a Turing O-machine introduced in his thesis and circumvents Gödel's halting and un-decidability issues by separating the computing and its management and pushing the management to a higher level.

In this model, the controlled computing element (the MICE) acts as a conventional Turing machine and the FCAPS managers act as the Oracles. **Figure 1** shows the functioning of a DIME and its analogy to the Turing O-machine.

There are three key modifications to the Turing machine which provide the abstractions required to provide the cognitive system attributes identified in this paper:

1) 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

2) 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 resources. The context, constraints and control op-

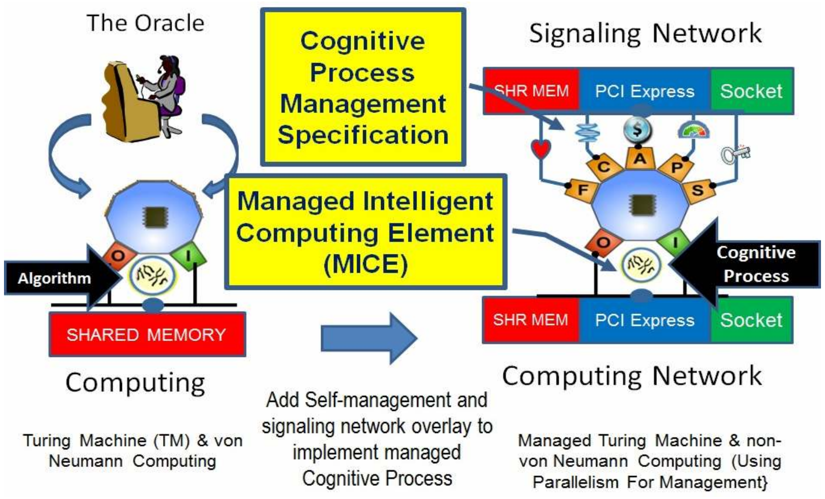

The Oracle
Cognitive Process Management Specification
Signaling Network
SHR MEM
PCI Express
Socket
C
A
P
F
S
O
I
Managed Intelligent Computing Element (MICE)
Algorithm
Cognitive Process
SHARED MEMORY
Computing
Computing Network
Add Self-management and signaling network overlay to implement managed Cognitive Process
Turing Machine (TM) & von Neumann Computing
Managed Turing Machine & non-von Neumann Computing (Using Parallelism For Management}

Figure 1. The Oracle function is implemented in a DIME using parallel fault, configuration, accounting, performance and security monitoring and control of the Turing machine implementing the algorithmic computation.

tions 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, 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.

3) 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 an 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.

The separation of computing and its management at the DIME is extended and scaled to become a two layer DIME network. The DIME network thus provides a regulatory (or signaling) network overlay over the computing network. The DIME network [14] consists of four components:

1) Nodes that encapsulate the managed intelligent computing element, MICE, with self-management of fault, configuration, accounting, performance and security (FCAPS);

2) Message-based communications (loose coupling);

3) Channels for intra- and inter-DIME communication and control; parallel and isolated channels for signaling (FCAPS management) and data (information) exchange;

4) Support for distributed recursive processes that, at some level, contain services that execute a set of tasks.

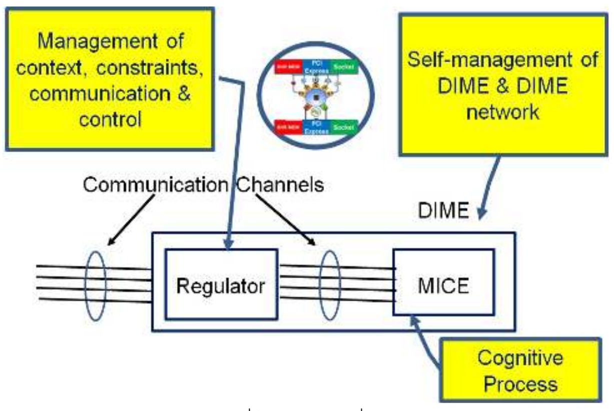
A generic structure model for the DIME network using the π-calculus recursive operation [24,25] is shown in **Figure 2**. In traditional procedural languages, recursion is implemented by suspending the current iteration while the next iteration executes, while in π-calculus the recursive iterations operate concurrently.

Let C, D, M and R represent a set of communication channels, DIME, Regulator and MICE nodes respectively.

$$d_i = \left(r_i \left|\{c_i\}\right|\{m_i\}\right)$$

where a Dime node, d_i is a set of concurrent processes $r_i \in R$, $c_i \in C$ and $m_i \in M$ and $\{c_i\}$ and $\{m_i\}$ represent a set of channels and Mice; the two sets of communication channels of **Figure 2** are together represented by the set $\{c_i\}$

$$!D = \langle r_0 \rangle \left[|D|!D \right]$$


Management of context, constraints, communication & control
Self-management of DIME & DIME network
Communication Channels
DIME
Regulator
MICE
Cognitive Process

$$D^- \left|\left(r_i \left|\{c_i\}\right|\{m_i\}\right)\right| D^+$$

Figure 2. DIME Networks modeled in π-calculus.

where "!" is the π-calculus recursion operator, "|" represents concurrency, r_0 represents the initial/root Regulator (at start-up); [⋯] represents option, and {⋯} represents a set.

Thus, from the above we know that a DIME network consists of an initial (start-up) Regulator (the root regulator, r_0 that may be connected through a set of communication channels and operate concurrently with a DIME network. We can visualize the DIME network from some node, d_i, created in the i^{th} iteration as:

$$D^- \left|\left(r_i \left|\{c_i\}\right|\{m_i\}\right)\right| D^+$$

where D^- represent the ancestors and D^+ the descendants.

A DIME can abstract a network of DIMEs thus providing an FCAPS managed DIME composition scheme. This allows us to implement both hierarchical and temporal event flows constituting the business processes. It is easy to see that the DIME's self-identity, self-management, recursive network composition scheme to implement managed network of computing elements and the dynamic control offered by the signaling channel to configure and reconfigure DIME networks provide a powerful mechanism to implement the process flows required to support cognitive process in computing systems.

Figure 3 shows various configurations that can be dynamically instantiated and reconfigured. It is easy to see that the DIME network architecture supports [14] the genetic transactions of replication, repair, recombination and reconfiguration.

Figure 4 shows that a regulated cognitive process can be implemented using the DIME network architecture. The local, group-level and global cognitive process policies are implemented using the monitoring and management capabilities offered by DIMEs. Each DIME can be interrupted and influenced by the DIME at the higher

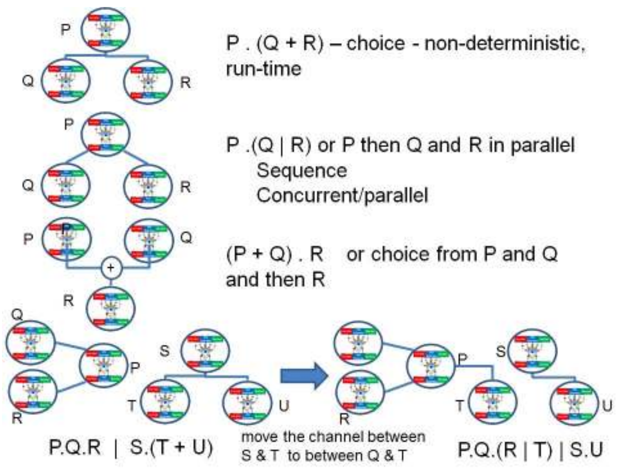

P . (Q + R) – choice - non-deterministic, run-time
P .(Q | R) or P then Q and R in parallel
Sequence
Concurrent/parallel
(P + Q) . R or choice from P and Q and then R
P.Q.R | S.(T + U)
move the channel between S & T to between Q & T
P.Q.(R | T) | S.U

Figure 3. DIME network configurations & π-calculus.

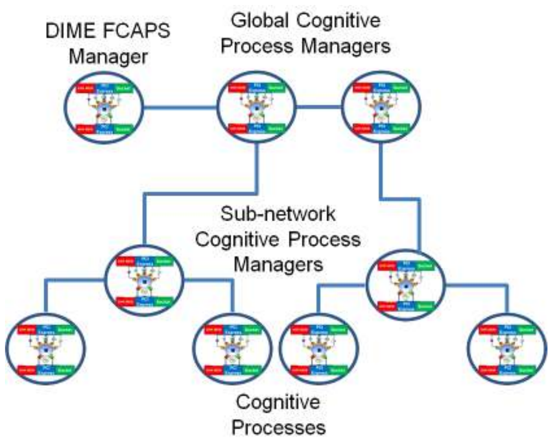

DIME FCAPS Manager
Global Cognitive Process Managers
Sub-network Cognitive Process Managers
Cognitive Processes

Figure 4. A DIME network implementing cognitive processes.

level using the Oracle-like instruction cycle. The DIME network therefore describes and implements a dynamic managed process flow exhibiting a meta-stable equilibrium where fluctuations are determined by the interactions among the various elements and their environments. More details of an application of these concepts in improving the operation and management of distributed computing in Linux and a native operating system called Parallax-OS are described in [26-29].

5. DIME Network, Entropy and Metastable Equilibrium

As mentioned above, DNA has been implemented in two instances [26-29]:

1) Using the DIME computing model to provide FCAPS management to a Linux process. This approach allows any Linux executable to be endowed with self-management and signaling capability thus allowing self-repair, auto-scaling, dynamic performance monitoring and management, and end-to-end transaction FCAPS management in a distributed system.

2) A native operating system to run in the next generation multi-core and many-core processor based computing devices to convert each core into a DIME and implement managed workflows in a DIME network spanning across multiple computing devices and geographies with network-wide policies based on business priorities, workload fluctuations and latency constraints.

In this paper, we focus on Copeland's fifth alternative mentioned above to examine how cognitive processes are simulated using DIME network architecture. The DIME network provides a way to model the computer and the computed and the system entropy depends on the overall network configuration, resources (CPU, memory, bandwidth, and storage at both the node level and at network level) and the interactions between various components. The Oracle nature of the DIME and the network-wide signaling control to influence any computing element provides a way to evolve the computing while it is in progress.

Figure 5 shows the system's entropy as a function of time depending on the system configuration. The non-determinism of a given configuration influenced by the interactions provides a set of meta-stable equilibriums. A configuration change provides a transition of one meta-stable equilibrium state to another. The network-wide coordination and collaboration of the DIME network orchestrate the global policies to implement the managed cognitive process using the MICE network. The Oracle "network effect" provides a synergy that is greater than the sum of its parts by effectively using global knowledge.

The DIME network provides the self-identity at the element level, group level and at global system level. The recursive distributed computing network created by the descriptions of managed computing elements at the node, sub-network and network level provides the required

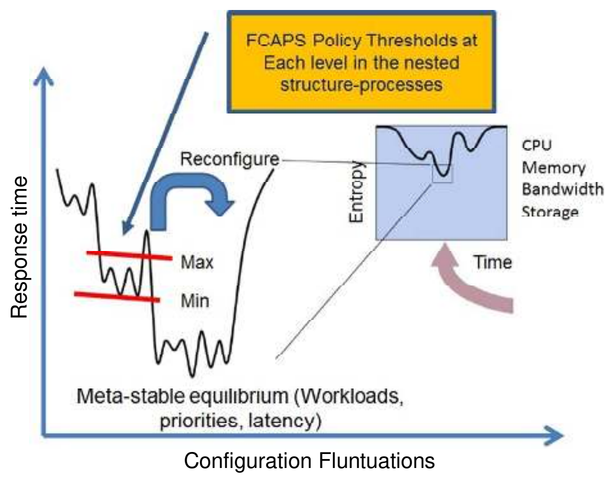

FCAPS Policy Thresholds at Each level in the nested structure-processes
Reconfigure
Entropy
CPU
Memory
Bandwidth
Storage
Time
Max
Min
Response time
Meta-stable equilibrium (Workloads, priorities, latency)
Configuration Fluntuations

Figure 5. Meta-stable equilibriums.

scale-invariant structure processes at element, group and system level cognition.

6. Conclusions

According to Andrew Wells [30] thinking is an ecological activity. The brain part manages the resources available and deploys them effectively to interact with the environment using well defined descriptions and execution mechanisms to execute and regulate the cognitive processes. He points out that the Turing machine has two principal components. "*One is a model of the individual mind*, *the other a model of the part of the environment. The mind part of the model*, *when functioning in isolations*, *is provably less powerful than the combination of the mind part and the environment part*." He also correctly emphasizes that the mind part Turing defined is constrained to the mind model dealing with the specific context of paper and pencil calculation executed by a human computer. Louise Barrett [31] making a case for the animal and human dependence on their bodies and environment—not just their brains—to behave intelligently, highlights the difference between Turing Machines implemented using von Neumann architecture and biological systems. She argues following Andrew Wells 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.

The DIME network architecture extrapolates Turing's Oracle machine with the recursive representation of the computer and the computed to create a parallel control network to manage the computing algorithms executed by the individual nodes acting as Oracle machines. This model incorporates a way to encapsulate not only the algorithm that is executed by a Turing machine but also a meta-model that provides the context, constraints, communication and control. The meta-model along with monitoring and management of the execution of the algorithm provides a way to incorporate dynamic coupling between various elements of the system, where each change in one element continually influences every other element's direction of change.

The DIME network architecture introduces parallel FCAPS management of Turing machine node with an Oracle-like intervention and a signaling network overlay to provide system-wide self-management. The introduction of a signaling network overlay over computing network with a system-wide Oracle-like intervention adds a new dimension in distributed computing by incorporating the architectural resilience of cellular organisms into computing machines. It allows specification of equilibrium patterns in computation process flows, and monitor and control exceptions system-wide. It allows contention resolution based on system-wide view and eliminates race conditions and other common issues found in current ad-hoc distributed computing practices. In systems with strong dynamic coupling between various elements of the system, where each change in one element continually influences the other element's direction of change, signaling in the computation model helps implement system-wide coordination and control based on global priorities, workload fluctuations and latency constraints.

Signaling and the separation of specification and execution of a computation provide a mechanism to introduce self-replication, self-repair, recombination and reconfiguration of computing network at run-time. These genetic transactions are essential to provide a computing environment to model, execute and regulate cognitive processes.

While we cannot answer if the brain is super-mechanical and how, we argue that injecting cognitive processes into computing is possible by extending the current von Neumann stored program control implementation of a Turing machine to execute an algorithm with Oracle-like intervention. The monitoring and control of both the algorithm execution and the resources executing the algorithm using the context, constraints, communication and control provide self-management with a systemic view to implement the system's intent. The DIME networks are restricted to a class of cognitive computing that involves the reciprocal influence of "bottom-up" and "top-down" processes. This we believe is the first step in addressing the computation and its limit by incorporating both the computer and the computed in modeling the physical world. This is analogous to how cellular organisms use their DNA (deoxyribonucleic acid) descriptions to both execute and regulate their process flows. As Mitchell Waldrop explains in his book on Complexity [32], "*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*."

7. Acknowledgements

The author is extremely grateful to Pankaj Goyal for showing that DIME network architecture can be one of the computing architectures for π-calculus and sharing his results before publication. The author wishes to express his gratitude to Giovanni Morana, Daniele Zito and Marco Di Sano for their enthusiasm and effort in implementing the self-regulating Linux, Apache, MySQL, and PHP (LAMP) cloud using the DIME network architecture thus demonstrating the elimination of the complexity of Hypervisors and Virtual Machines. The author also is grateful to Ian Seyler for implementing the native operating system converting each core in a multi-core processor into a DIME to demonstrate system-wide resiliency in workflow execution spanning across multiple many- core processors, and servers.

REFERENCES

[1] P. Cockshott, L. M. MacKenzie and G. Michaelson, "Computation and Its Limits," Oxford University Press, Oxford, 2012.

[2] J. V. Neumann, "Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components," In: C. E. Shannon and J. McCarthy, Eds., *Automatic Studies*, Princeton University Press, Princeton, 1956, pp. 43-98.

[3] W. Aspray and A. Burks, "Papers of John von Neumann on Computing and Computer Theory," MIT Press, Cambridge, 1989.

[4] A. M. Turing, "The Essential Turing," Oxford University Press, Oxford, 2004.

[5] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, Vol. 49, 1950, pp. 433-460. doi:10.1093/mind/LIX.236.433

[6] P. Stanier and G. Moore, "Embryos, Genes and Birth Defects," 2nd Edition, John Wiley & Sons, London, 2006, p. 5.

[7] S. B. Caroll, "The New Science of Evo Devo—Endless Forms Most Beautiful," W. W. Norton & Co., New York, 2005.

[8] A. M. Turing, "Systems of Logic Defined by Ordinals," *Proceedings London Mathematical Society*, Vol. 2, No. 45, 1939, pp. 161-228.

[9] B. J. Copeland, "Turing's O-Machines, Searle, Penrose and the Brain," *Analysis*, Vol. 58, 1998, pp. 128-138. doi:10.1093/analys/58.2.128

[10] H. R. Maturana, "Biological Computer Laboratory Research Report BCL 9.0," University of Illinois, Urbana, 1970.

[11] H. R. Maturana and F. J. Varela, "Autopoiesis and Cognition: The Realization of the Living (Boston Studies in the Philosophy of Science)," D. Reidel, Dordrecht, 1960.

[12] E. Thompson, "Mind in Life: Biology, Phenomenology, and the Sciences of the Mind," Harvard University Press, Cambridge, 2007.

[13] P. Johnson-Laird, "How Could Consciousness Arise from the Computations of the Brain?" In: C. Blakemore and S. Greenfield, Eds., *Mindwaves*, Basil Blackwell, Oxford, 1987.

[14] R. Mikkilineni, "Designing a New Class of Distributed Systems," Springer, New York, 2011. doi:10.1007/978-1-4614-1924-2

[15] R. Mikkilineni, A. Comparini and G. Morana, Turing O-Machine and the DIME Network Architecture: Injecting the Architectural Resiliency into Distributed Computing, in Turing-100," In: A. Voronkov, Ed., *EPIC Series*, *Easy Chair*, 2012. http://www.easychair.org/publications/?page=877986046

[16] A. M. Turing, "The Essential Turing," Oxford University Press, Oxford, 2004.

[17] G. Piccinini, "Alan Turing and the Mathematical Objection," *Minds and Machines*, Vol. 13, No. 1, 2003, pp. 23-48. doi:10.1023/A:1021348629167

[18] A. M. Turing, "Systems of Logic Defined by Ordinals," *Proceedings London Mathematical Society*, Vol. 45, 1939, pp. 161-228.

[19] S. Feferman, "Turing's Thesis," *Notices of the AMS*, Vol. 53, No. 10, 2006, p. 2.

[20] R. Soare, "Turing Oracle Machines, Online Computing, and Three Displacements in Computability Theory," *Annals of Pure and Applied Logic*, Vol. 160, No. 3, 2009, pp. 368-399. doi:10.1016/j.apal.2009.01.008

[21] A. Kurakin, "Retrieved from the Universal Principles of Self-Organization and the Unity of Nature and Knowledge," 2007. http://www.alexeikurakin.org/text/thesoft.pdf

[22] A. Kurakin, "Theoretical Biology and Medical Modeling," 2011. http://www.tbiomed.com/content/8/1/4

[23] R. Milner, "Communicating and Mobile Systems: The Pi-Calculus," Cambridge University Press, Cambridge, 1999.

[24] P. Goyal and R. Mikkilineni, "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, 25-27 June 2012.

[25] P. Goyal, "A Recursive Computing Model for DIME Network Architecture Using π-Calculus," *Private Communication*, 2012.

[26] R. Mikkilineni and I. Seyler, "A New Operating System for Scalable, Distributed, and Parallel Computing," *IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum* (*IPDPSW*), Anchorage, 16-20 May 2011, pp. 976-983.

[27] R. Mikkilineni and I. Seyler, "Implementing Distributed, Self-Managing Computing Services Infrastructure Using a Scalable, Parallel and Network-centric Computing Model," In: M. Villari, C. I. Brandic and F. Tusa, Eds., *Achieving Federated and Self-Manageable Cloud Infrastructures*: *Theory and Practice*, IGI Global, pp. 57-78.

[28] R. Mikkilineni, I. Seyler, G. Morana, D. Zito and M. Di Sano, "Service Virtualization Using a Non-Von Neumann Parallel, Distributed, and Scalable Computing Model,"

Journal of Computer Networks and Communications, 2012, in Press.

[29] G. Morana and R. Mikkilineni, "Scaling and Self-Repair of Linux Based Services Using a Novel Distributed Computing Model Exploiting Parallelism," 20*th IEEE International Workshops on Enabling Technologies*: *Infrastructure for Collaborative Enterprises* (*WETICE*), Paris, 27-29 June 2011, pp. 98-103.

[30] A. Wells, "Rethinking Cognitive Computation: Turing and the Science of Mind," Palgrave Macmillan, London, 2006.

[31] L. Barrett, "Beyond the Brain," Princeton University Press, Princeton, 2011.

[32] M. Mitchell-Waldrop, "Complexity: The Emerging Science at the Edge of Order and Chaos," Penguin Books, London, 1992, p. 218.