Digital Genome and Self-Regulating Distributed Software Applications with Associative Memory and Event-Driven History

Rao Mikkilineni *, W. Patrick Kelly , Gideon Crawley

Posted Date: 9 August 2024

doi: 10.20944/preprints202406.1622.v2

Keywords: Distributed Software Application; Digital Genome; Self-Regulation; Autopoiesis; Associative Memory; Event-Driven Interaction History

Article

Digital Genome and Self-Regulating Distributed Software Applications with Associative Memory and Event-Driven History

Rao Mikkilineni [1,*], W. Patrick Kelly [1] and Gideon Crawley [2]

[1] Golden Gate University; wkelly@ggu.edu
[2] Dominican University of California; gidstube@gmail.com
* Correspondence: rmikkilineni@ggu.edu

Abstract: Biological systems have a unique ability inherited through their genome. It allows them to build, operate, and manage a society of cells with complex organizational structures where autonomous components execute specific tasks and collaborate in groups to fulfill systemic goals with shared knowledge. The system receives information from various senses, makes sense of what is being observed, and acts using its experience, while the observations are still in progress. We use the General Theory of Information (GTI) to implement a digital genome, specifying the operational processes that design, deploy, operate, and manage a cloud-agnostic distributed application that is independent of IaaS and PaaS infrastructure, which provides the resources required to execute the software components. The digital genome specifies the functional and non-functional requirements that define the goals and best-practice policies to evolve the system using associative memory and event-driven interaction history to maintain stability and safety while achieving the system's objectives. We demonstrate a structural machine, cognizing oracles, and knowledge structures derived from GTI used for designing, deploying, operating, and managing a distributed video streaming application with autopoietic self-regulation that maintains structural stability and communication among distributed components with shared knowledge while maintaining expected behaviors dictated by functional requirements.

Keywords: distributed software application; digital genome; self-regulation; autopoiesis; associative memory; event-driven interaction history

1. Introduction

Designing, developing, deploying, operating, and managing the lifecycle of distributed software applications is a critical area of study because all our business and personal lives depend on them. Volumes have been written on the subject and the book authority organization lists 20 best distributed system books of all time [1]. A literature survey on service-oriented architecture used 65 papers published between 2005 and 2020. [2]. A systemic literature review of microservice architecture (MSA), (MSA is a more recent proposal to use fine-grained services architecture for distributed software systems) discovered in 3842 papers [3]. However, several issues with their design, deployment, operation, and management, their instability under large fluctuations in resource demand or availability, vulnerability to security breaches, and CAP theorem limitations are ever-present.

In this paper, we present an analysis of the current state-of-the-art distributed software application design, development, deployment, operation, and management. We describe a novel extension based on the foundation of the General Theory of Information (GTI) that infuses:

1. Improved resilience using the concept of autopoiesis, which refers to the ability of a system to replicate itself and maintain identity and stability while facing fluctuations caused by external influences, and
2. Enhanced cognition using cognitive behaviors that model the system's state, sense internal and external changes, analyze, predict, and take action to mitigate any risk to its functional fulfillment.

 doi:10.20944/preprints202406.1622.v2

The paper is structured as follows:

1. Discuss the limitations of both symbolic and sub-symbolic computing structures used in the current implementation of distributed software systems,
2. Discuss GTI and its application to create a knowledge representation in the form of associative memory, and event-driven transaction history of the distributed software system, and
3. Demonstrate a distributed software application with autopoietic and enhanced cognitive behaviors. An autopoietic manager configures and manages the components of the distributed software system and a cognitive network manager provides enhanced cognition to manage the connections between the software components to maintain the quality of service. A policy manager's policies are defined by best practices and experience to manage deviations from expected behaviors.

1.1. Limitations of the Current State-of-the-Art

1.1.1. CAP Theorem Limitation:

The CAP theorem [4], also known as Brewer's theorem states that it is impossible for a distributed data system to simultaneously provide more than two out of three guarantees:

1. Consistency: All users see the same data at the same time, no matter which node they connect to. For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed 'successful'.
2. Availability: Any client requesting data gets a response, even if one or more nodes are down. Another way to state this is that all working nodes in the distributed system return a valid response for any request, without exception.
3. Partition Tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

1.1.2. Complexity:

In addition, the complexity of maintaining availability and performance continues to increase with Hobson's choice between single vendor-lock-in or multi-vendor complexity [5]. There are solutions available using free and open-source software or adopting multi-vendor and heterogeneous resources offered by multiple cloud providers [6]. This can help to maintain the scalability and management flexibility of distributed applications. However, this often increases complexity, and layers of management lead to the "who manages the managers" conundrum. Moreover, the advent of many virtualized and disaggregated technologies, and the rapid increase of the Internet of Things (IoT) makes end-to-end orchestration difficult to do at scale.

1.1.3. Computation and Its Limits:

Some arguments suggest that the problems of scalability, resiliency, and complexity of distributed software applications are symptoms that point to a foundational shortcoming of the computational model associated with the stored program implementation of the Turing Machine from which all current-generation computers are derived [7-13].

As Cockshott et al., [11] p. 215 describe in their book "Computation and its Limits" the concept of the universal Turing machine has allowed us to create general-purpose computers and "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." External agents are required for the harmonious execution of the computer and the computation.

The distributed software application consists of a network structure of distributed software components that are dependent on the infrastructure that provides the resources (CPU, memory, and power/energy) which are managed by different service providers with their own infrastructure as a service (IaaS), and platform as a service (PaaS) management systems. In essence, the processes executing the computing structures (hardware and software) behave like a complex adaptive system.

 doi:10.20944/preprints202406.1622.v2

They exhibit emergent behavior when faced with local fluctuations impacting the infrastructure. For example, if a failure occurs in any one component, the execution halts and external entities must fix the problem. If the demand fluctuates, resources must be increased or decreased to maintain efficiency and performance by an external entity. These lead to a single vendor lock-in or the complexity of a third-party orchestrator that manages various component managers.

1.2. Stored Program Control Implementation of Symbolic and Sub-Symbolic Computing Structures

Current-generation computers are used for process automation, intelligent decision-making, mimicking behaviors by robots, and using transformers to generate text, images, and videos [14-16]. Figure 1, shows process automation executed by an algorithm operating on data structures. Insights obtained through machine learning algorithms or deep learning algorithms for intelligent decision-making are derived from data analytics also shown in Figure 1. In addition, deep learning algorithms (which use multi-layered neural networks to simulate the complex pattern recognition processes of the human brain) are used to perform robotic behaviors or generative AI tasks.

McCulloch and Pitts's 1943 paper [17] on how neurons might work, and Frank Rosenblatt's introduction of the perceptron [18] led to the current AI revolution with deep learning algorithms using computers.

Robotic Behavior Learning primarily involves training a robot to perform specific tasks or actions. This includes reinforcement learning, where the robot learns from trial and error, receiving rewards for successful actions and penalties for mistakes. Over time, the robot improves its performance by maximizing its rewards and minimizing penalties. This type of learning is useful in environments where explicit programming of all possible scenarios is impractical. On the other hand, transformers in GenAI focus on processing and generating text data. They use attention mechanisms to understand the context within large bodies of text. The input to a Transformer model is a sequence of tokens (words, sub-words, or characters), and the output is typically a sequence of tokens that forms a coherent and contextually relevant text. This could be a continuation of the input text, a translation into another language, or an answer to a question. In addition, when the algorithm is trained on a large dataset of images or videos, it learns to understand the underlying patterns and structures in the data and generates new, original content. The results can be surprisingly creative and realistic, opening up new possibilities for art, design, and visual storytelling.

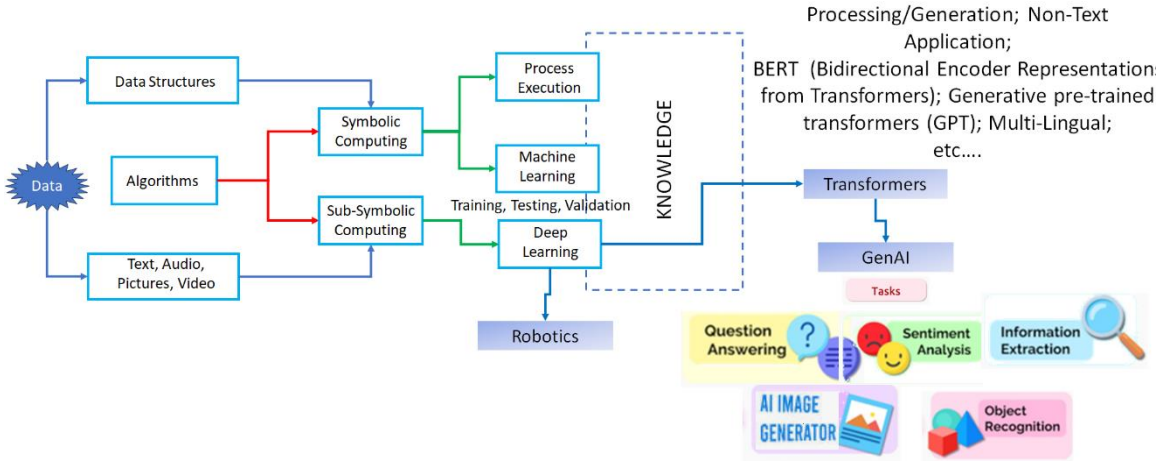

Figure 1. Current state of the art of information processing structures.

Symbolic computing uses a sequence of symbols (representing algorithms) that operate on another sequence of symbols (representing data structures describing the state of a system that depicts various entities and relationships) to change the state. Sub-symbolic computation is associated with neural networks where an algorithm mimics the neurons in biological systems (perceptron). A multi-layer network using perceptron mimics the neural networks in converting the information provided as input (text, voice, video, etc.).

Several issues with sub-symbolic computing have been identified [19-21]:

 doi:10.20944/preprints202406.1622.v2

1. Lack of Interpretability: Deep learning models, particularly neural networks, are often "black boxes" because it's difficult to understand the reasoning behind how they respond to the queries.
2. Need for Large Amounts of Data: These models typically require large data sets to train effectively.
3. Overfitting: Deep learning models can overfit the training data, meaning they may not generalize well to unseen data.
4. Vanishing and Exploding Gradient Problems: These are issues that can arise during the training process, making it difficult for the model to learn.
5. Adversarial Attacks: Deep learning models are vulnerable to adversarial attacks, where small, intentionally designed changes to the input can cause the model to make incorrect predictions.
6. Difficulty Incorporating Symbolic Knowledge: Sub-symbolic methods, such as neural networks, often struggle to incorporate symbolic knowledge, such as causal relationships and practitioners' knowledge.
7. Bias: These methods can learn and reflect biases present in the training data.
8. Lack of Coordination with Symbolic Systems: While sub-symbolic and symbolic systems can operate independently, they often need to coordinate closely together to integrate the knowledge derived from them, which can be challenging.

1.3. The General Theory of Information and Supr-Symbolic Computing:

Recent advances based on the General Theory of Information, provide a new approach that integrates symbolic and sub-symbolic computing structures with a novel super-symbolic structure and addresses various foundational shortcomings mentioned above [9,22,35,37].

Figure 2 depicts [30,32,36] how information bridges the material world of structures formed through energy and matter transformations; the mental world that observes the material world and creates mental structures that represent the knowledge received from observed information; and the digital world created using the information generated by both the mental, and material structures using the stored program control implementation of the Turing Machine.

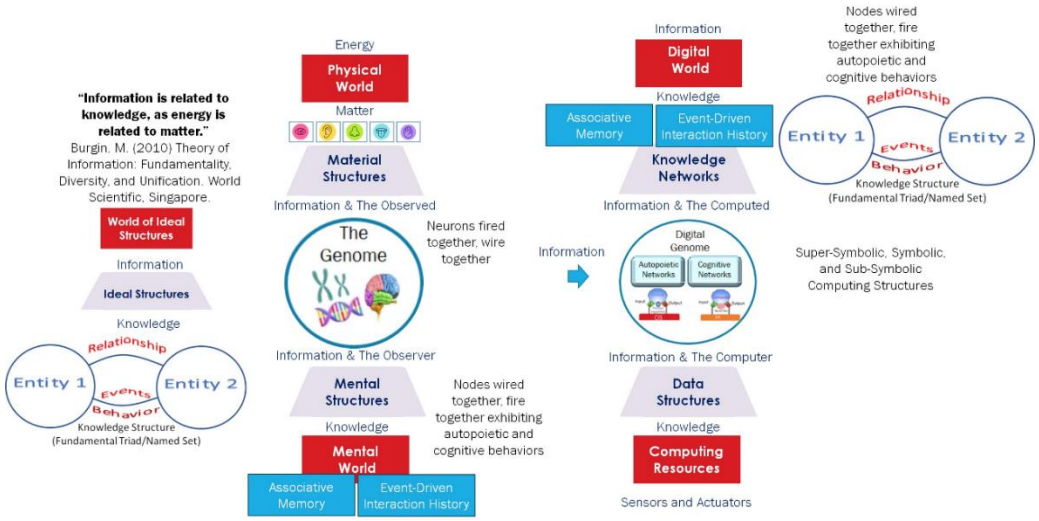

Figure 2. According to the General Theory of Information, Information is the bridge between the material, mental, and digital worlds.

Mark Burgin's General Theory of Information (GTI) bridges our understanding of the material world, which consists of matter and energy, and the mental worlds of biological systems that utilize information and knowledge. This theory is significant because it offers a model for how operational knowledge is represented and used by biological systems involved in building, operating, and managing life processes [9,21-37]. In addition, it suggests a way to represent operational knowledge and use it to build, deploy, and operate distributed software applications. The result is a new class of digital automata with autopoietic and cognitive behaviors that biological systems exhibit [23].

 doi:10.20944/preprints202406.1622.v2

Autopoietic behavior refers to the self-producing and self-maintaining nature of living systems. Cognitive behavior refers to obtaining and using knowledge.

The genome through natural selection has evolved to capture and transmit the knowledge to build, operate, and manage a structure that receives information from the material world and converts it into knowledge in the mental world using genes and neurons. The result is an associative memory and event-driven transaction history that the system uses to interact with the material world. In the symbolic computing structures, the knowledge is represented as data structures (entities and their relationships) depicting the system state, and its evolution using an algorithm. In sub-symbolic computing, the algorithm creates a neural network and knowledge is represented by the optimized parameters that result from training the neural network with data structures. The observations of Turing, Neumann, McCulloch, Pitts, and Rosenblatt have led to the creation of the digital world where information is converted into knowledge in the digital form as shown in Figure 1. The super-symbolic structure derived from GTI provides a higher-level knowledge representation in the form of fundamental triads/named sets [22, 24, 28] shown in Figure 2.

In essence, three contributions from GTI enable the developing, deploying, operating, and managing of a distributed application using heterogeneous IaaS and PaaS resources while overcoming the shortcomings discussed in this paper:

Digital Automata: Burgin's construction of a new class of digital automata to overcome the barrier posed by the Church–Turing Thesis has significant implications for AI. This allows for creating of more advanced AI systems that can perform tasks beyond the capabilities of traditional Turing machines [23-26].

Super-symbolic Computing: His contribution to super-symbolic computing with knowledge structures, cognizing oracles, and structural machines changes how we design and develop self-regulating distributed applications. These tools also allow AI systems to process and understand information in a more complex and nuanced way, similar to how humans do by interacting with the sub-symbolic and super-symbolic computing structures with common knowledge representation using super-symbolic computing which is different from neuro-symbolic computing [36-38].

Digital Genome: The schema and associated operations derived from GTI are used to model a digital genome specifying the operational knowledge of algorithms executing the software life processes. The digital genome specifies operational knowledge that defines and executes domain-specific functional requirements, non-functional requirements, and best-practice policies that maintain the system behavior, conforming to the expectations of the design. This results in a digital software system with a super-symbolic computing structure exhibiting autopoietic and cognitive behaviors that biological systems also exhibit [37-39].

Figure 3 shows the super-symbolic computing structure implementing a domain-specific digital genome using the structural machines, cognizing oracles, and knowledge structures derived from GTI to create a knowledge network with two important features:

1. The knowledge network captures the system state and its evolution caused by the event-driven interactions of various entities interacting with each other in the form of associative memory and event-driven interaction history. It is important to emphasize that the Digital Genome and super-symbolic computing structures differ from using symbolic and sub-symbolic structures together. For example, the new frameworks [39] from MIT Computer Science and Artificial Intelligence Laboratory provide important context for language models that perform coding, AI planning, and robotic tasks. However, this approach does not use associative memory and event-driven transaction history as long-term memory. The digital genome provides a schema for creating them using knowledge derived from both symbolic and sub-symbolic computing.
2. GTI provides a schema and operations [34] for representing the system state and its evolution, which are used to define and execute various processes that fulfill the functional and non-functional requirements and the best-practice policies and constraints.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 August 2024 doi:10.20944/preprints202406.1622.v2

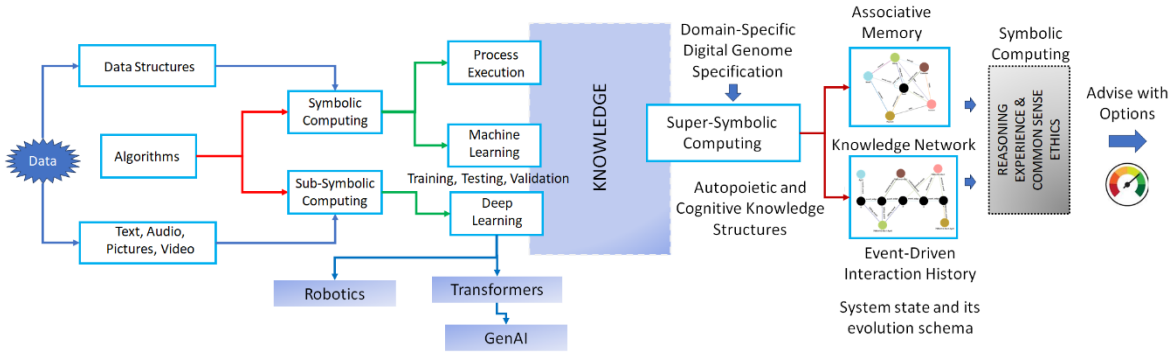

Figure 3. The digital genome implementation integrating symbolic and sub-symbolic computing.

The figure shows the theoretical GTI-based implementation model of the digital genome specifying the functional and non-functional requirements along with adaptable policies based on experience to maintain the expected behaviors based on the genome specification. In this paper, we describe an implementation of a distributed software application using the digital genome specification and demonstrate the policy-based management of functional and non-functional requirements. In section 2, we describe the distributed software application and its implementation. In section 3, we discuss the results and lessons learned. In section 4, we draw some conclusions and discuss some future directions.

Figure 4 shows the two information processing structures one based on the Turing Machine and von Neumann architecture and the other based on the structural machines and the knowledge network composed of knowledge structures and cognizing oracles.

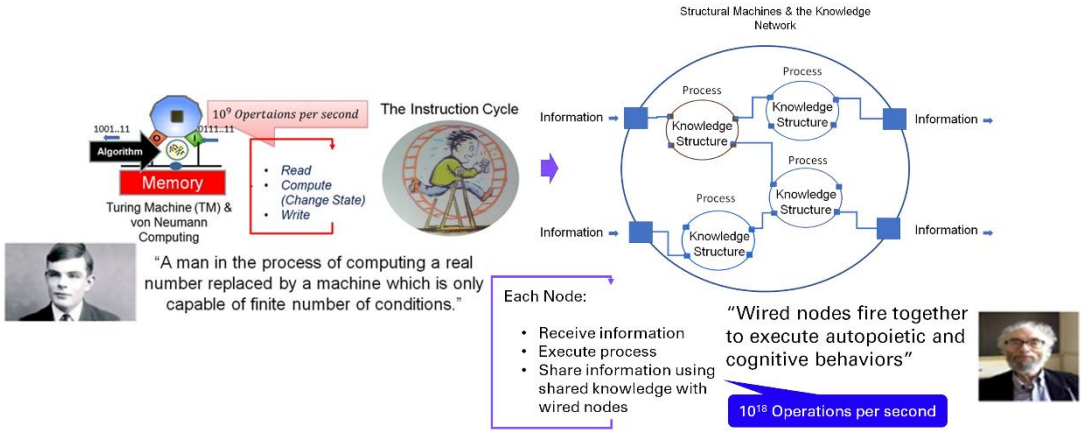

Figure 4. Two information processing structures.

The current state of the art uses the von- Neumann stored program control implementation of the Turing Machine to execute algorithms using symbolic, sub-symbolic, or a combination called neuro-symbolic computing structures. The approach presented in this paper uses structural machines with a schema and operations defining a knowledge network based on GTI. The structural machine uses various process execution methods such as symbolic and sub-symbolic computing using general-purpose computers, people, or analog devices with sensors and actuators. Figure 4 shows the two models. Each knowledge structure executes a process specified by the functional and non-functional requirements using best-practice policies and constraints to fulfill the design goals. In the next section, we describe the knowledge network and how to use it to design, deploy, operate, and manage a distributed software application.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 August 2024 doi:10.20944/preprints202406.1622.v2

2. Distributed Software Application and its Implementation

A distributed software application is designed to operate on multiple computers or devices across a network. They spread their functionality across different components with a specific role, work together, and communicate using shared knowledge to accomplish the application's overall goals. The overall functionality and operation are defined using functional and non-functional requirements, and policies and constraints are specified using best practices that ensure the application's functionality, availability, scalability, performance, and security while executing its mission. We describe a process to design, develop, deploy, operate, and manage a distributed software application using functional and non-functional requirements, policies, and constraints specified to achieve a specific goal. The goal is determined by the domain knowledge representing various entities, their relationships, and the behaviors that result from their interactions.

oth symbolic and sub-symbolic computing structures execute processes that receive input, fulfill functional requirements, and share knowledge with other wired components to fulfill system-level functional requirements. Figure 5 shows the computing models and the knowledge representation as a knowledge network used in the new approach based on GTI. As discussed in [34 p. 13] "Information processing in triadic structural (entities, relationships, and behaviors) machines is accomplished through operations on knowledge structures, which are graphs representing nodes, links, and their behaviors. Knowledge structures contain named sets and their evolution containing named entities/objects, named attributes, and their relationships. Ontology-based models of domain knowledge structures contain information about known knowns, known unknowns, and processes for dealing with unknown unknowns through verification and consensus. Inter-object and intra-object behaviors are encapsulated as named sets and their chains. Events and associated behaviors are defined as algorithmic workflows, which determine the system's state evolution.

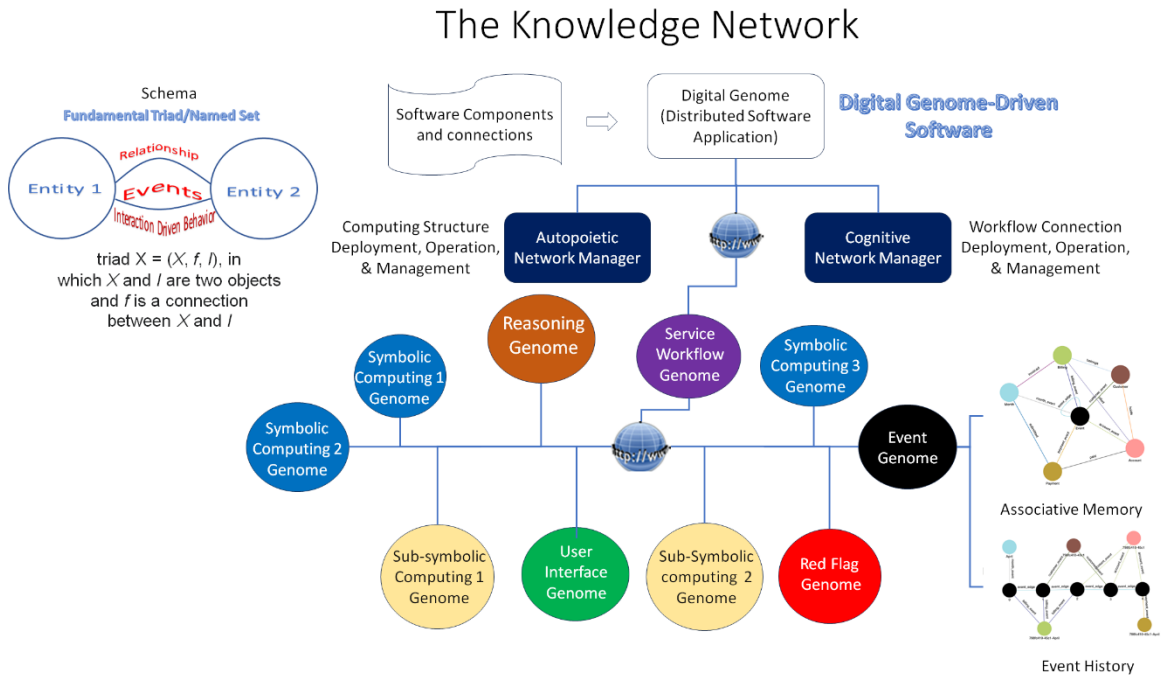

Figure 5. Digital genome-driven distributed application with associative memory and event-driven interaction history.

A named set chain of knowledge structures (knowledge network) provides a genealogy representing the system's state history. This genealogy can be treated as the deep memory and used for reasoning about the system's behavior, as well as for its modification and optimization."

 doi:10.20944/preprints202406.1622.v2

The domain knowledge for each node and the knowledge network is obtained from different sources (including the Large Language Models) and specified as functional and non-functional requirements derived from the system's desired availability, performance, security, and stability.

The digital genome specifies the functionality and operation of the system that deploys, operates, and manages the evolution of the knowledge network with the knowledge about where to get the computing resources and use them. The autopoietic network manager is designed to deploy the software components with appropriate computing resources (e.g., IaaS and PaaS in a cloud) as services. The cognitive network manager manages the communication connections between the nodes executing various processes. The service workflow manager controls the workflow among the nodes delivering the service. An event monitor captures the events in the system to create the associative memory and the event-driven interaction history. A cognitive red flag manager captures deviations from the normal workflow and alerts the autopoietic manager which takes corrective action by coordinating with the cognitive network manager. The architecture provides a self-regulating distributed software application using resources from different providers.

We describe an example implemented using this architecture to demonstrate the feasibility and the benefits of this architecture. A video-on-demand service is deployed in a cloud with auto-failover. The purpose of this demonstration is to show the feasibility of creating associative memory and event-driven transaction history that provides real-time evolution of the system as long-term memory. They can be used to perform data analytics using a transparent model to gain insights in contrast to the current state of the art as shown in Figure 3.

3. Video on Demand (VoD) Service with Associative Memory and Event-Driven interaction history

The design begins with defining the functional requirements, non-functional requirements, best-practice policies, and constraints.

Functional Requirements for User Interaction:

- User is given a service URL
- User registers for the service
- Administrator authenticates with a user ID and password
- User logs into URL with user ID and Password
- The user is presented with a menu of videos
- User Selects a video
- The user is presented with a video and controls to interact
- User uses the controls (pause, start, rewind, fast forward) and watches the video.

Functional Requirements for Video Service Delivery:

- Video Service consists of several components working together:
  - VoD service workflow manager
  - Video content manager
  - Video server
  - Video client

Non-functional Requirements, Policies, and Constraints:

- **Auto-Failover:** When a video service is interrupted by the failure of any component, the user service should not experience any service interruption.
- **Auto-Scaling:** When the end-to-end service response time falls below a threshold, necessary resource adjustments should be made to adjust the response time to the desired value.
- **Live Migration:** Any component should be easily migrated from one infrastructure to another without service interruption.

Figure 6 shows a digital genome-based architecture with various components that fulfill these requirements. Each node is a process-executing engine that receives input and executes the process using a symbolic or sub-symbolic computing structure. An example is a Docker container deployed in a cloud using local IaaS and PaaS. The roles of the digital genome, the autopoietic network manager, and the cognitive network manager are well discussed in several papers [9,21,37,40].

 doi:10.20944/preprints202406.1622.v2

Just as the genome in living organisms contains the information required to manage life processes the digital genome contains all the information about the distributed software application in the form of knowledge structures to build itself, reproduce itself, and maintain its structural stability while using its cognitive processes to fulfill the functional requirements.

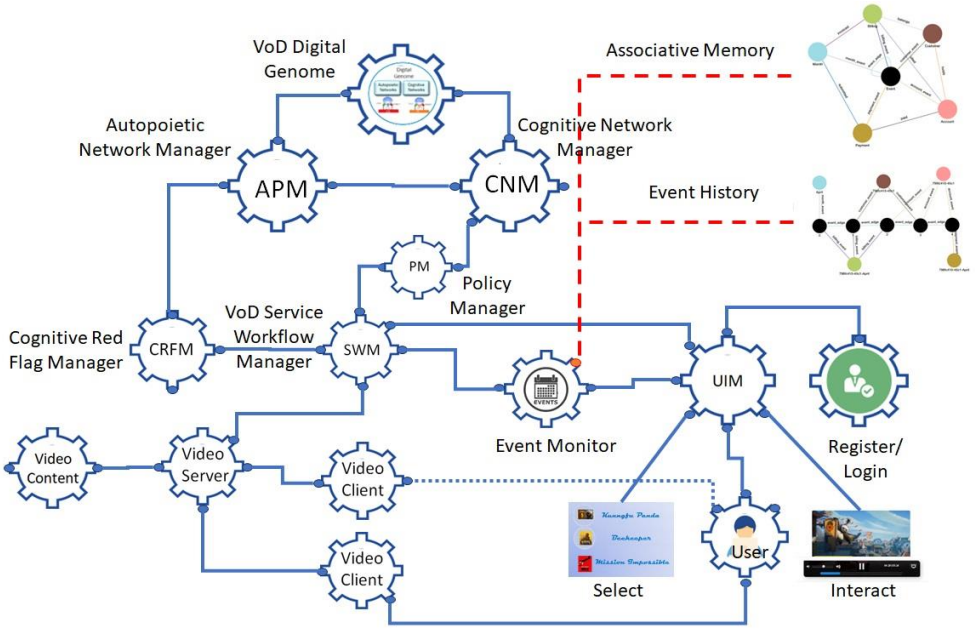

Figure 6. Schema-based Service Architecture with various components.

We summarize the functions of the schema of the knowledge network that specifies the process of processes executing the system's functional and non-functional requirements and best practice policies and constraints:

The Digital Genome Node: It is the master controller that provides the operational knowledge to deploy the knowledge network that contains various nodes executing different processes and communicating with other nodes wired together with shared knowledge. It initiates the autopoietic and cognitive network managers responsible for managing the structure and workflow fulfilling the functional and non-functional requirements.

Autopoietic Network Manager (APM) Node: It contains knowledge about where to deploy the nodes using resources from various sources such as IaaS and PaaS from cloud providers. It receives a list of Docker containers to be deployed as nodes and determines how they are wired together. At t=0, APM deploys various containers using the desired cloud resources. It passes on the URLs of these nodes and the network connections to the Cognitive Network Manager. For example, if the nodes are duplicated to fulfill non-functional requirements, it assigns which connection is the primary and which is the secondary.

Using the URLs and their wiring map, the CNM consults with the Policy Manager which specifies the requirements for fulfilling the non-functional requirements such as auto-failover, auto-scaling, and live migration, then sets up the connection list and passes it on to the Service Workflow Manager.

Service Workflow Manager (SWM): Provides the service workflow control by managing the connections between various nodes participating in the service workflow. In the VoD service, it manages the workflow of the video service subnetwork and the user interface manager subnetwork as shown in Figure 5. It also manages the deviations from the expected workflow by using the policies that define actions to correct them.

User Interface Management Subnetwork (UIM): It manages the user interface workflow providing registration, login, video selection, and other interactions.

Video Service Management Subnetwork: It provides the video service from content to video server and client management.

 doi:10.20944/preprints202406.1622.v2

Cognitive Red Flag Manager: when deviations occur from normal workflow such as one of the video clients fails, the SWM will switch it to the secondary video client as shown in Figure 6. It also communicates a red flag which is then communicated to the APM to take corrective action, in this case, restore the video client that went down and let the CNM know to make it secondary.

Event Monitor: It monitors events from video service and user interface workflows and creates an associative memory and an event-driven interaction history with a time stamp. These provide the long-term memory for other nodes to use the information in multiple ways including performing data analytics and gaining insights to take appropriate action.

We summarize the development workflow translating the functional and non-functional requirements, and best-practice policies as follows:

1. Developers design the process workflow based on the functional requirements.
2. Each process (a knowledge structure with a schema, consisting of entities, relationships, and their event-driven interactions, is defined by its inputs and actions each process executes based on the inputs and generated outputs communicated with other processes using shared knowledge.
3. All the knowledge structures are containerized and deployed as a knowledge network. For example, the user interface subnetwork contains the user registration, login, video selection, and use processes with specified inputs, behaviors, and outputs. The video service subprocess deals with video management and delivery processes. Wired knowledge structures fire together to execute autopoietic and enhanced cognitive behaviors managed by a service workflow manager under the supervision of the autopoietic and cognitive network managers.
4. The autopoietic manager manages the deployment of knowledge structures using cloud resources.
5. The cognitive network manager manages the workflow connections between the knowledge structures.
6. Autopoietic and cognitive managers along with a policy manager, who dictates best practice rules, manage the deviations from expected behavior caused by fluctuations in the availability of or demand for the resources or workflow disruptions. The best practice policies are derived from history and experience. For example, if the service response time exceeds a threshold, auto-scaling is used to reduce it. Using the return time objective (RTO) and the return position objective (RPO), the structure of the knowledge network is configured by the autopoietic and cognitive network managers to maintain the quality of service using auto-failover or live migration.

   In the next section, we discuss the results.

4. Results

Various processes shown in Figure 6 are implemented using these three technologies:

1. Python programming for creating the schema and its evolution with various instances,
2. Containers that are deployed using the Google Cloud, and
3. A graph database (TigerGraph) that represents the schema and its evolution with various instances using the events that capture the interactions as associative memory and event-driven interaction history.

The video (Digital Genome VoD Presentation – Autopoietic Machines (triadicautomata.com) (Accessed on o6/17/2024)) demonstrates the implementation of associative memory and the event-driven transaction history of a distributed software application with a digital software genome specification discussed in this paper. The video details the schema and operation of the video-on-demand service:

1. A knowledge sub-network in action, where users interact with various entities delivering the service. They can register, log in, choose a video from a menu, and interact with it.
2. A knowledge sub-network that manages and serves the video-on-demand.
3. A higher-level knowledge network with the service workflow manager, policy manager, autopoietic manager, and cognitive network manager provides structural stability and enhanced cognitive workflow management to address the impact of fluctuations in the interactions causing disruptions in the quality of service.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 August 2024 doi:10.20944/preprints202406.1622.v2

4. A graph database demonstrates the system evolution using a service schema, associative memory, and event-driven interaction history of all the users.

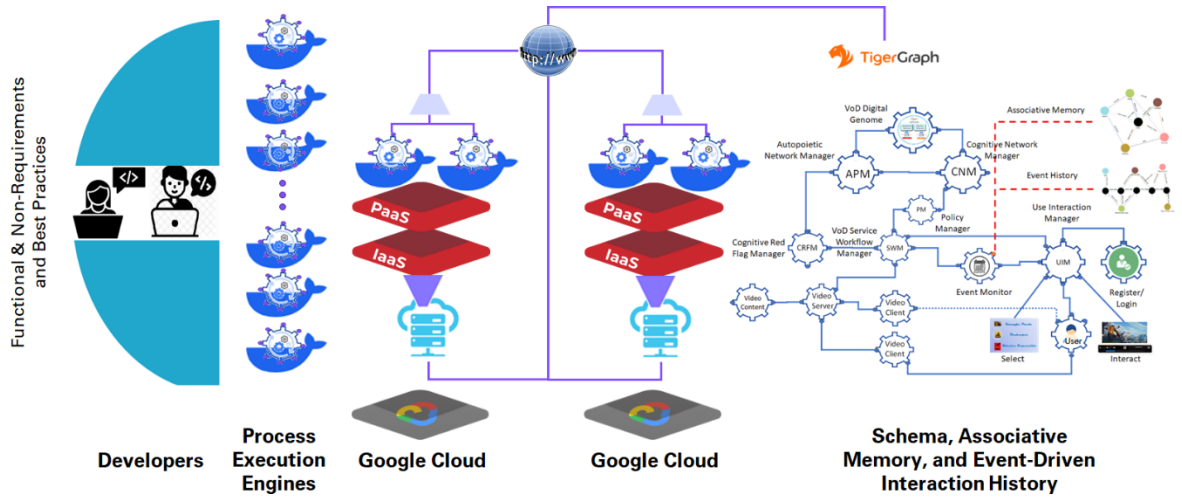

Figure 7. Deployment of the VoD service using Cloud resources. An implementation is presented in the video mentioned in this paper.

The system is structured to be resilient to the failure of one of the critical components that delivers the video to the user without interruption. The switchover occurs without the user noticing the failure.

5. Conclusions

The objective of this paper is to explain how GTI helps us to understand knowledge and its representation which belongs to the mental worlds of biological systems helps them to build operate and manage a society of cells with complex organizational structures. Self-regulation, associative memory, and event-driven interaction history are key attributes that allow them to make sense of what they are observing and act while the observation is still in progress to control the outcomes and manage risk. GTI articulated by Mark Burgin provides a unified context for existing directions in information studies. It allows us to elaborate on a comprehensive definition, explain relations between information, data, and knowledge, and demonstrate how different mathematical models of information and information processes are related [41,42]. According to GTI, information is the bridge between the material structures and their state evolution and the mental structures of biological systems that model and interact with the material structures. Information is converted into knowledge and represented by composable knowledge structures forming multi-layer knowledge networks.

This paper describes our attempt to apply the same knowledge representation to build operate and manage a society of autonomous distributed software components with complex organizational structures. The benefits of the new approach are self-regulating distributed application design, development, deployment, operation, and management which is independent of what infrastructure as a service (IaaS) or Platform as a service (PaaS) are used to execute them

Event-driven interaction history allows for real-time sharing of business moments as asynchronous events. The schema-based knowledge network implementation improves a system's scalability, agility, and adaptability through the dynamic control and feedback signals exchanged among the system's components. This architecture separates the traditional sensing, analysis, control, and actuation elements for a given system across a network. It also allows the sharing of information between systems.

Nodes and subnetworks can be easily added or deleted without impacting the rest of the system thus improving scalability, agility, and adaptability. Communication between nodes can be directed through crypto security to enhance the system's security [23]. Other examples of networks with a

 doi:10.20944/preprints202406.1622.v2

control network overlay are communication networks with signaling overlay, cellular organisms with networks of genes and neurons, and human organizational networks with hierarchical and matrix management control overlay. They all behave as a society of autonomous components with local knowledge of their role, function, and operation. They use shared knowledge about their role, function, and operation to contribute to the systemic goals. The system is designed to optimize the global process execution while considering the constraints of the local processes of the components. The digital genome specifies how to build, operate, and manage a society of software components with complex organizational structures where autonomous components execute specific tasks and collaborate in groups to fulfill systemic goals with shared knowledge. GTI-based schema of the knowledge network allows us to model a society of autonomous components executing systemic goals with autopoietic and cognitive behaviors. The VoD implementation using the schema defined in this paper illustrates this process.

5.1. Future Directions:

Our results show that schema-based distributed software applications with specific functional and non-functional requirements, and policies based on best practices improve the system's structural stability and enhance cognitive workflow management. The term digital genome we use here is a metaphor referring to the genomes of biological systems where the knowledge to build, operate, and manage a society of cells with self-regulation with autopoietic and cognitive behaviors. We have demonstrated how the digital genome is designed and implemented to replicate a VoD service.

Kelly, et al., [43] describe another implementation of a medical-knowledge-based digital assistant that uses medical knowledge derived from various sources including the large language models, and assists the early medical diagnosis process by reducing the knowledge gap between the patient and medical professionals involved in the process. This implementation suggests the integration of symbolic and sub-symbolic computing structures using the schema-based approach.

This is an emerging area where GTI provides a theoretical and practical foundation for advancing the implementation of AI, enabling the creation of more intelligent and autonomous systems [25,35,36]. They offer a new perspective on how we can infuse autopoietic and cognitive behaviors into digital machines, leading to the evolution of machine intelligence. As Naidoo [44] points out "The presentation of the overall qualitative framework, comprising a qualitative analysis of information, m data, and knowledge, will be valuable and of great assistance in delineating regulatory, ethical, and strategic trajectories. In addition, this framework provides insights (and answers) regarding (1) data privacy and protection; (2) delineations between information, data, and knowledge based on the important notion of trust; (3) a structured approach to establishing the necessary conditions for an open society and system, and the maintenance of said openness, based on the work of Karl Popper and Georg Wilhelm Friedrich Hegel; (4) an active agent approach that promotes autonomy and freedom and protects the open society; and (5) a data governance mechanism based on the work of Friedrich Hayek, which structures the current legal–ethical–financial and social society." GTI provides a framework that infuses structural stability to manage the availability, scalability, performance, security, survival and enhanced cognition and reasoning through associative memory and the history of the state evolution of the system.

5.2. RelatedWork and Contributions of this Paper

As David Baden [45] points out "According to the GTI, information exists in an abstract world of structures, somewhat analogous to Plato's ideal world of Forms. This world of ideal structures interacts with the physical and mental worlds; a conceptualization strongly similar to Popper's three-world ontology. The relation between information and the structure of reality has been noted by other theorists, including Tom Stonier and Luciano Floridi." He goes on to say that GTI is one of a number of 'gap bridging' theories that attempt to integrate ideas of information from different domains. A mathematical theory, aiming to incorporate previous approaches, including those of Shannon, Bar-Hillel, Dretske, and others, GTI provides a formalism to unify the varied ways in which information

 doi:10.20944/preprints202406.1622.v2

is understood, through a series of ontological and axiological principles which express what information is, and how it may be measured. It defines information as that which has a capacity to cause changes in a system, so that information may be seen as a form of energy. GTI may, in principle, encompass all forms of information including the physical, biological, mental, and social, and encompassing syntactic, semantic, and pragmatic information, although it has not yet been extended in any detail into social and ethical domains.

Unfortunately, there are not many applications of GTI explaining the practice of GTI in building new information technologies. The references by Burgin and Mikkilineni alluded to in this paper are the only ones that discuss this subject. Hopefully, this paper will stimulate the new generation of computer scientists and information technology professionals to take the research to the next level.

As far as associative memory and event-driven transaction history, our approach is based on a schema and operations of the schema derived from GTI and id different from the following approaches:

1. Event-Driven Associative Memory Networks for Knowledge Graph Completion by X. Wang, et al [46]. This paper explores how event-driven associative memory networks can enhance knowledge graph completion tasks. It introduces a novel approach that combines temporal information with associative memory mechanisms to improve link prediction in knowledge graphs.
2. Memory Networks by J. Weston, et al. [47]. Although not exclusively focused on associative memory, this influential paper introduces the concept of memory networks. It discusses how external memory can be used to augment neural networks, allowing them to store and retrieve information more effectively.
3. Neural Turing Machines A. Graves, et al. [48]. While not directly related to event-driven transaction history, this paper proposes a model called Neural Turing Machines (NTMs). NTMs combine neural networks with external memory, enabling them to learn algorithmic tasks and perform associative recall.

How is this work related to AGI? According to Ben Goertzel [49] "At the moment, AGI system design is as much artistic as scientific, relying heavily on the designer's scientific intuition. AGI implementation and testing are interwoven with (more or less) inspired tinkering, according to which systems are progressively improved internally as their behaviors are observed in various situations. This sort of approach is not unworkable, and many great inventions have been created via similar processes. It's unclear how necessary or useful a more advanced AGI theory will be for the creation of practical AGI systems. But it seems likely that, the further we can get toward a theory providing tools to address questions like those listed above, the more systematic and scientific the AGI design process will become, and the more capable the resulting systems."

Our work uses the current implementations of AI as nodes in our knowledge network to augment the total knowledge describing the system's state and its evolution based on event-driven interactions between nodes. In addition, GTI points to the role of knowledge representation in biological systems which allows autopoietic and cognitive behaviors. In addition, GTI provides a schema and tools to infuse these behaviors into digital automata with a knowledge representation. Associative memory and the event driven interaction history of knowledge structures representing the entities, relationships, and their behaviors provide a path to implement higher-level reasoning systems using experience, common sense, and wisdom.

What are the limitations of this approach? The limitations for quick adaption of the schema-based approach stem from the need for computer scientists and information technology professionals to overcome the learning curve involved in adapting new concepts from GTI, genomics, neuroscience, and limitations of the current state of the art. Fortunately, this requires no new tools or technologies. A new schema-based architecture that integrates existing symbolic and sub-symbolic computing structures and the knowledge acquisition from the new large language models may hasten quick adaption.

We conclude this paper with an excerpt from Mark Burgin [30, p.4]. "Knowledge processing and management make problem solving much more efficient and are crucial for big companies and institutions (Ueno, 1987; Osuga, 1989; Dalkir, 20005). To achieve this goal, it is necessary to make

 doi:10.20944/preprints202406.1622.v2

distinction between knowledge and knowledge representation to know regularities of knowledge structure, functioning and representation, and develop software (and in some cases, hardware) that is based on these regularities. Many intelligent systems search knowledge spaces, which are explicitly or implicitly predefined by the choice of knowledge representation. In effect, the knowledge representation serves as strong bias."

6. Patents

Parts of this work may be used to apply for patents with different use case implementations.

Supplementary Materials: The following supporting can be downloaded at: Digital Genome Implementation Presentations: – Autopoietic Machines (triadicautomata.com).

Author Contributions: Conceptualization, Rao Mikkilineni; methodology, Rao Mikkilineni and W. Patrick Kelly; software, W. Patrick Kelly and Gideon Crawley. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: One of the authors, Rao Mikkilineni expresses his gratitude to the Late Prof. Mark Burgin who spent countless hours explaining the General Theory of Information and helped me understand its applications to software development. Rao Mikkilineni and Patrick Kelly also acknowledge the many discussions with Justin Kromelow, CEO at Opos Solutions, and his continued support.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A: Explanation of Figures

Figure 1: Current state of the art of information processing structures

This diagram illustrates the flow and relationship between different components of data processing and artificial intelligence (AI). Here's a breakdown of each part.

- **Data Sources, Data Structures, and Algorithms**: Data can come from various sources such as text, audio, pictures, and videos. Data is organized using data structures. Algorithms are applied to process and manipulate this data
- **Computing Paradigms:** Symbolic computing involves using symbols to represent problems and logical rules to solve them. Sub-symbolic computing involves techniques like neural networks and other forms of machine learning.
- **Application in Robotics, Generative AI:** The knowledge gained from machine learning and deep learning can be applied to robotics for automation and intelligent behavior.
- **Transformers:** A type of model architecture, especially useful for processing sequential data, like language. Examples include BERT and GPT.
- **GenAI:** Generative AI models, which can generate new data similar to the data they were trained on.
- **Question Answering:** The ability of AI to provide answers to questions posed in natural language.
- **Sentiment Analysis**: Determining the sentiment expressed in text, such as positive, negative, or neutral.
- **Information Extraction:** Extracting structured information from unstructured data.
- **AI Image Generation:** Creating new images from textual descriptions or other inputs.
- **Object Recognition:** Identifying objects within images or videos.

Figure 2: According to the General Theory of Information, information is the bridge between the material, mental, and digital worlds

This diagram elaborates on the relationship between information, knowledge, and computing or information processing structures in various worlds (physical, mental, and digital).

- **World of Ideal Structures:** Information is seen as fundamental to the world of ideal structures. According to GTI, the ideal structures are represented by Named sets/Fundamental triads where entities with established relationships interact with each other and evolve their state based on the event-driven behaviors events, forming a basic knowledge structure.

 doi:10.20944/preprints202406.1622.v2

- **World of Material Structures:** In the physical world, energy relates to matter, and material structures evolve governed by the laws of conversion of energy and matter. In the mental world, Information received by the observers is processed by the neural networks in biological systems. Neurons fired together wire together to create associative memory and event-driven transaction history.
- **World of Digital Structures:** In the digital world created by humans, information is processed in digital form using symbolic and sub-symbolic computing structures. In essence, this figure illustrates the comprehensive view of how information is processed, structured, and transformed into knowledge across different realms, linking theoretical foundations to practical implementations in computing.

Figure 3: The digital genome implementation integrating symbolic and sub-symbolic computing

This figure highlights how symbolic and sub-symbolic computing can be combined to create a robust, adaptive software system using the tools derived from the general theory of information. The red and blue lines illustrate the interconnected nature of symbolic and sub-symbolic computing structures in creating the knowledge representation. Super-symbolic computing suggested by GTI uses symbolic and sub-symbolic elements to create an integrated knowledge representation as knowledge networks. This framework highlights how symbolic and sub-symbolic computing can be combined to create a robust, adaptive system capable of generating knowledge, learning from data, and providing intelligent advice and options.

Figure 4: Two information processing structures

This image illustrates the evolution from traditional computing models to more advanced structural machines and knowledge networks suggested by GTI. The Turing Machine-based computing model was derived from the observation that "A man in the process of computing a real number replaced by a machine which is only capable of a finite number of conditions.". The computing model with structural machines and knowledge networks is derived from the observation that "Nodes wired together fire together in a knowledge network to exhibit autopoietic and cognitive behaviors."

The image contrasts traditional computing (limited by sequential processing and finite state machines) with a future vision of structural machines and knowledge networks, where interconnected nodes process and share information enabling complex cognitive and self-maintaining behaviors and components of a Knowledge Network driven by a Digital Genome, which integrates both symbolic and sub-symbolic computing to manage and process information effectively.

Figure 5: Digital genome-driven distributed application with associative memory and event-driven interaction history

This figure depicts entities, relationships, and behaviors of autonomous distributed system components constituting a knowledge network. The digital Genome provides the core framework that integrates various software components to form a cohesive system with specific functional, and non-functional requirements and best-practice-based policies. Autopoietic Network Manager ensures the deployment of the computing structures and maintains structural stability. The Cognitive Network Manager manages the cognitive process workflow ensuring information flow within the knowledge network.

Knowledge network nodes are specialized units each focusing on different aspects of computation and information management:

- Reasoning Genome: Handles logical reasoning and decision-making.
- Service Workflow Genome: Manages workflows and service operations.
- Event Genome: Tracks and processes events within the network.
- User Interface Genome: Manages interactions with users.
- Red Flag Genome: Identifies and handles anomalies or critical issues.
- Symbolic Computing Genomes: Handle different aspects of symbolic computation.
- Sub-Symbolic Computing Genomes: Handle pattern recognition and machine learning tasks.

 doi:10.20944/preprints202406.1622.v2

Associative Memory stores information based on associations, allowing the network to recall and utilize relevant knowledge. Event history records all relevant events and interactions, capturing the system's state and evolution and providing a single point of truth. The knowledge is represented as a network where nodes receive input, execute the process, and share information with wired nodes capturing the system's state and evolution history based on node interactions.

Figure 6: Schema-based Service Architecture with various components

This figure depicts the architecture of a video-on-demand (VoD) system driven by a digital genome specification described in the paper and demonstrated in the video., integrating various components to manage and deliver video content efficiently while leveraging associative memory and event history for improved functionality.

It leverages the digital genome framework to ensure self-maintenance, cognitive processing, and effective information flow. The associative memory and event history components provide enhanced decision-making capabilities based on past interactions and stored knowledge. Users interact with the system through a user-friendly interface, selecting and watching video content delivered by the video server. The system continuously monitors events to maintain optimal performance and reliability.

Figure 7: Deployment of the VoD service using Cloud resources. An implementation is presented in the video mentioned in this paper

This figure demonstrates how a sophisticated VoD system can be built using cloud services and advanced data management techniques. Developers define the system's functionality, which is then executed by process engines on the Google Cloud infrastructure. The system components, including autopoietic and cognitive network managers, work together to manage workflows, handle anomalies, and ensure efficient delivery of video content. The knowledge network, supported by schema, associative memory, and event history, provides the necessary intelligence and adaptability to meet user needs and maintain optimal performance. Python programs are used to execute symbolic and sub-symbolic processes. Containers are used to deploy, operate, and manage the components in a cloud infrastructure. Graph database technology is used to manage complex data relationships, further enhancing the system's capabilities.

Appendix B: Glossary of Terms

Associative Memory: Associative memory refers to the ability to remember relationships between concepts, not just the individual concepts themselves. For instance, it involves remembering how two words are related (e.g., “man” – “woman”) or recognizing an object and its alternate name (e.g., a guitar). In psychology, it’s defined as the capacity to learn and recall the relationship between unrelated items, such as remembering someone’s name or the scent of a particular perfume. Neural associative memories map specific input representations to specific output representations, allowing recall based on associations. In the context of this paper, it refers to the schema and its evolution history of entities, relationships, and interactions that help maintain stability and safety while achieving the system’s objectives through enhanced cognition. Associative memory (psychology) - Wikipedia

Autopoiesis: A system's ability to reproduce and maintain itself. It refers to the self-producing and self-maintaining nature of living systems, applied here to distributed software systems. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1618936/

CAP Theorem: A principle that states it is impossible for a distributed data system to simultaneously provide more than two out of three guarantees: Consistency, Availability, and Partition Tolerance. https://www.ibm.com/topics/cap-theorem

Cognitive Network Manager (CNM): A component that manages the communication connections between the nodes executing various processes in a distributed system. It ensures the system's workflow and connections align with non-functional requirements. https://www.mdpi.com/2813-0324/8/1/70

Cognizing Oracles: Components derived from the General Theory of Information that are used in super-symbolic computing structures to process and understand information in a complex and

 doi:10.20944/preprints202406.1622.v2

nuanced way. They play a crucial role in enabling self-regulation and cognitive behaviors in distributed software applications. https://www.researchgate.net/figure/Oracles-as-cognizing-agents-managing-the-execution-and-evolution-of-knowledge-structures_fig5_342741582

Digital Genome: A digital specification of operational knowledge defining and executing the life processes of distributed software applications. It includes functional requirements, non-functional requirements, and best-practice policies to maintain system behavior. https://www.mdpi.com/2409-9287/8/6/107

Event-Driven Interaction History: A record of interactions and events within a system, used to maintain the system's long-term memory and aid in decision-making and stability. https://www.preprints.org/manuscript/202404.1298/v1

General Theory of Information (GTI): A theoretical framework that integrates symbolic and sub-symbolic computing structures with a novel super-symbolic structure to address foundational shortcomings in current computational models. https://www.researchgate.net/publication/318740204_The_General_Theory_of_Information_as_a_Unifying_Factor_for_Information_Studies_The_Noble_Eight-Fold_Path

Infrastructure as a Service (IaaS): A cloud computing service that provides virtualized computing resources over the internet, allowing users to run software without managing the underlying hardware. https://cloud.google.com/learn/what-is-iaas

Microservice Architecture (MSA): A design approach for distributed systems where applications are composed of small, independent services that communicate over network protocols, each serving a single purpose. https://cloud.google.com/learn/what-is-microservices-architecture

Non-functional Requirements: Requirements that specify criteria for the operation of a system, such as performance, scalability, and reliability, rather than specific behaviors or functions. https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

Platform as a Service (PaaS): A cloud computing service that provides a platform allowing customers to develop, run, and manage applications without dealing with the underlying infrastructure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas

Service Workflow Manager (SWM): A component that controls the workflow among the nodes delivering a service, ensuring that the system operates as expected and takes corrective actions when deviations occur.

Structural Machines: Machines that use knowledge structures to represent and manage system states and their evolution. They perform operations on these structures to maintain system stability and achieve desired behaviors. https://www.mdpi.com/2504-3900/47/1/26

Super-symbolic Computing: An advanced computing paradigm that integrates symbolic and sub-symbolic representations to create a more nuanced and complex understanding of information, enabling self-regulation and cognitive behaviors in digital systems. https://easychair.org/publications/preprint/PMjC

Symbolic Computing: A type of computing where symbols and operations on symbols represent and manipulate data, often used in traditional artificial intelligence systems. https://easychair.org/publications/preprint/PMjC

Turing Machine: A mathematical model of computation that defines an abstract machine which manipulates symbols on a strip of tape according to a table of rules. It is the basis for the theory of computation. https://plato.stanford.edu/entries/turing-machine/

User Interface Management Subnetwork (UIM): A component that manages the user interface workflow, including registration, login, and video selection interactions in a video-on-demand service.

Video Service Management Subnetwork: A subnetwork responsible for managing the video service from content management to video server and client interactions.

Von Neumann Architecture: A computer architecture model where a single storage structure holds both instructions and data, used in most contemporary computers. https://www.computerscience.gcse.guru/theory/von-neumann-architecture

References

1. 20 Best Distributed System Books of All Time – BookAuthority https://bookauthority.org/books/best-distributed-system-books (Accessed on 17th Jun 2024)
2. Bohloul, S. M. (2021). Service-oriented Architecture: a review of state-of-the-art literature from an organizational perspective. Journal of Industrial Integration and Management, 6(03), 353-382.
3. Söylemez, M.; Tekinerdogan, B.; Kolukısa Tarhan, A. Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review. Appl. Sci. 2022, 12, 5507. https://doi.org/10.3390/app12115507.
4. "CAP Theorem (Explained)" Youtube, uploaded by Techdose 9/12/2018, https://youtu.be/PyLMoN8kHwI?si=gtHWzvt2gelf3kly (Accessed on 17th June 2024)
5. Opara-Martins, J., Sahandi, R. & Tian, F. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. J Cloud Comp 5, 4 (2016). https://doi.org/10.1186/s13677-016-0054-z
6. Luis M. Vaquero, Felix Cuadrado, Yehia Elkhatib, Jorge Bernal-Bernabe, Satish N. Srirama, Mohamed Faten Zhani, Research challenges in nextgen service orchestration, Future Generation Computer Systems, Volume 90, 2019, Pages 20-38, ISSN 0167-739X, https://doi.org/10.1016/j.future.2018.07.039.
   a. (https://www.sciencedirect.com/science/article/pii/S0167739X18303157) (Accessed on 20th Jun 2024)
7. Burgin, M., Super-Recursive Algorithms, Springer Monographs in Computer Science, 2005, ISBN: 0-387
   a. 95569-0
8. Dodig Crnkovic, Gordana. (2012). Info-computationalism and Morphological Computing of Informational Structure. 10.1007/978-3-642-28111-2_10.
9. Burgin, M.; Mikkilineni, R. General Theory of Information Paves the Way to a Secure, Service-Oriented Internet Connecting People, Things, and Businesses. In Proceedings of the 2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI), Kanazawa, Japan, 2–8 July 2022; pp. 144–149.
10. Dodig Crnkovic, G. Significance of Models of Computation, from Turing Model to Natural Computation. Minds Mach. 2011, 21, 301–322.
11. Cockshott, P.; MacKenzie, L.M.; Michaelson, G. Computation and Its Limits; Oxford University Press: Oxford, UK, 2012; p. 215.
12. van Leeuwen, J., & Wiedermann, J. (2000). The Turing machine paradigm in contemporary computing. In B. Enquist, & W. Schmidt, Mathematics Unlimited—2001 and Beyond.LNCS. New York, NY: Springer-Verlag.
13. Wegner, P., & Eberbach, E. (2004). New Models of Computation. The Computer Journal, 47(1), 4-9. Wegner, P., & Goldin, D. (2003). Computation beyond Turing Machines: Seeking appropriate methods to model computing and human thought. Communications of the ACM, 46(4), 100.
14. Rothman, D. (2024). Transformers for Natural Language Processing and Computer Vision: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3. Packt Publishing Ltd.
15. Hu, W., Li, X., Li, C., Li, R., Jiang, T., Sun, H., ... & Li, X. (2023). A state-of-the-art survey of artificial neural networks for whole-slide image analysis: from popular convolutional neural networks to potential visual transformers. Computers in Biology and Medicine, 161, 107034.
16. Soori, M., Arezoo, B., & Dastres, R. (2023). Artificial intelligence, machine learning and deep learning in advanced robotics, a review. Cognitive Robotics, 3, 54–70.
17. McCulloch, W.S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133. doi:10.1007/BF02478259
18. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, 65 6, 386-408.
19. Karimian, G., Petelos, E., & Evers, S. M. (2022). The ethical issues of the application of artificial intelligence in healthcare: a systematic scoping review. AI and Ethics, 2(4), 539-551.
20. Groumpos, P. P. (2019). Artificial intelligence: Issues, challenges, opportunities and threats. In Creativity in Intelligent Technologies and Data Science: Third Conference, CIT&DS 2019, Volgograd, Russia, September 16–19, 2019, Proceedings, Part I 3 (pp. 19-33). Springer International Publishing.
21. R. Mikkilineni, A New Class of Autopoietic and Cognitive Machines, Information, 2022, 13, 24; https://doi.org/10.3390/info13010024
22. Burgin, M., & Mikkilineni, R. (2022). Seven Layers of Computation: Methodological Analysis and Mathematical Modeling. Filozofia i Nauka, p. 11-31
23. Burgin, M., & Mikkilineni, R. (2022, July). General Theory of Information Paves the Way to a Secure, Service-Oriented Internet Connecting People, Things, and Businesses. In 2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 144-149). IEEE.
24. M. Burgin, R. Mikkilineni, From Data Processing to Knowledge Processing: Working with Operational Schemas by Autopoietic Machines, Big Data Cogn. Comput., 2021, 5 (13); https://doi.org/10.3390/bdcc5010013

25. M. Burgin, R. Mikkilineni, On the Autopoietic and Cognitive Behavior, EasyChair Preprint no. 6261, version 2, 2021; https://easychair.org/publications/preprint/tkjk
26. M. Burgin, R. Mikkilineni, V. Phalke, Autopoietic Computing Systems and Triadic Automata: The Theory and Practice, Advances in Computer and Communications, 2020, 1 (1), pp. 16–35.
27. M. Burgin, Unified Foundations of Mathematics, Preprint Mathematics LO/0403186, 2004, 39 p.; electronic edition: http://arXiv.org. 30
28. Mark Burgin, Theory of Named Sets, Mathematics Research Developments, Nova Science, New York, 2011
29. Mark Burgin, Structural Reality, Nova Science Publishers, New York 2012.
30. Mark Burgin, Theory of Knowledge: Structures and Processes, World Scientific, New York, London–Singapore 2016.
31. Mark Burgin, Ideas of Plato in the Context of Contemporary Science and Mathematics, Athens Journal of Humanities and Arts, 2017, 4 (3), pp. 161–182.
32. Mark Burgin, Information Processing by Structural Machines, in Theoretical Information Studies: Information in the World, World Scientific, New York–London–Singapore 2020, pp. 323-371.
33. Mark Burgin, Elements of the Theory of Nested Named Sets, Theory and Applications of Mathematics & Computer Science, 2020, 10 (2), pp. 46–70.
34. M. Burgin, R. Mikkilineni, From Data Processing to Knowledge Processing: Working with Operational Schemas by Autopoietic Machines, Big Data Cogn. Comput., 2021, 5 (13); https://doi.org/10.3390/bdcc5010013
35. M. Burgin, R. Mikkilineni, V. Phalke, Autopoietic Computing Systems and Triadic Automata: The Theory and Practice, Advances in Computer and Communications, 2020, 1 (1), pp. 16–35.
36. Burgin, Mark and Mikkilineni, Rao. (2022) Information Theoretic Principles of Software Development, EasyChair Preprint no. 9222. https://easychair.org/publications/preprint/jnMd
37. Mikkilineni, R.; Kelly, W. P. Machine Intelligence with Associative Memory and Event-Driven Transaction History. Preprints 2024, 2024041298. https://doi.org/10.20944/preprints202404.1298.v1
38. https://news.mit.edu/2024/natural-language-boosts-llm-performance-coding-planning-robotics-0501 (Accessed 20 June 2024)
39. Mikkilineni, Rao. 2023. "Mark Burgin's Legacy: The General Theory of Information, the Digital Genome, and the Future of Machine Intelligence" Philosophies 8, no. 6: 107. https://doi.org/10.3390/philosophies8060107
40. Mikkilineni, Rao. 2022. "Infusing Autopoietic and Cognitive Behaviors into Digital Automata to Improve Their Sentience, Resilience, and Intelligence" Big Data and Cognitive Computing 6, no. 1: 7. https://doi.org/10.3390/bdcc6010007
41. Krzanowski, R. Information: What We Do and Do Not Know-A Review. Available online: https://www.researchgate.net/publication/370105722_Information_What_We_Do_and_Do_Not_Know-A_Review (accessed on 30 June 2023).
42. Floridi, L. Information. A Very Short Introduction; Oxford University Press: Oxford, UK, 2010. [Google Scholar]
43. Kelly, W. Patrick, Francesco Coccaro, and Rao Mikkilineni. 2023. "General Theory of Information, Digital Genome, Large Language Models, and Medical Knowledge-Driven Digital Assistant" Computer Sciences & Mathematics Forum 8, no. 1: 70. https://doi.org/10.3390/cmsf2023008070
44. Naidoo, M. (2024). The open ontology and information society. *Frontiers in Genetics*, *15*, 1290658
45. https://theoccasionalinformationist.com/2023/03/05/mark-burgin-1946-2023/ (Accessed on 28 June 2024)
46. Wang, X., Zhang, J., & Wang, Y. (2021). Event-Driven Associative Memory Networks for Knowledge Graph Completion. Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI).
47. Weston, J., Chopra, S., & Bordes, A. (2015). Memory Networks. Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS).
48. Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing Machines. arXiv preprint arXiv:1410.5401.
49. Goertzel, B. (2009). Artificial General Intelligence: Concept, State of the Art, and Future Prospects. Journal of Artificial General Intelligence, 5, 1 - 48.