An Engineering Paradigm: Noema

Ron Burback

August 5, 1997

Contents

List of Figures

List of Tables

Figure 1: An Icon Representing a Noema

A traditional engineering paradigm is very hierarchical in nature. To understand a whole, first understand the parts then combine the knowledge into an understanding of the whole.

In a noemic paradigm, the understanding of the whole comes first. The understanding of the part is a projection of the whole. The noemic paradigm reflects life.

This thesis proposes that modern software engineering built for highly distributed computing environments should be based on a noemic paradigm.

Life is an example of a Noema. A Noema is not a neural network which simulates the learning process of the brain. A Noema is not a genetic algorithm which simulates system evolution. A noemic paradigm is represented

by the body chemistry of living systems like the respiratory, circulatory, immune, and digestive systems.

This thesis will define the foundations of the noemic paradigm, give some examples, and support the theory that a Noema, though harder to build, supports change.

See Figure 1 on page 5 for an icon of a Noema. Each circle viewed in isolation is represented by a simple shaded pattern. When the three circles are combined new regions and new patterns, previously not visible, appear. They are triangular shaped and represent two distinct interaction patterns formed from pairs of circles and formed from all three circles. The role of a circle in the Noema consists of four distinct categories that of a circle itself, the interaction of the circle with the other two circles individually, and the interaction of all three circles.

In a Noema, the whole is greater than the sum of the parts.

2 Introduction

2.1 Background

Traditional western science and technology is strongly influenced by rationalism and logical empiricism that can be traced back to Plato. A good summary of this paradigm can be found in [WF85]. When faced with the problem of trying to understand a system, the rationalistic tradition indicates that three basic steps are taken:

- Characterize the whole system in terms of identifiable sub-components with well defined properties.
- Understand each sub-component by finding general rules that describe their behavior.
- Combine the sub-components into the whole system, applying the rules of the sub-components, to draw conclusions about the behavior of the whole and to establish the understanding of the whole.

The rationalist approach requires complete knowledge of sub-components and their actions and interactions. Decomposition of complex systems into simpler parts is a natural scientific paradigm in the rationalist approach.

The rationalist approach is in contrast to hermeneutics [Hei68]. Here the components of a whole system are defined as an interpretation in the context of the whole and the environment. There is no full and explicit understanding of neither the components nor the whole system. The understanding is never complete.

The whole system defines to exists a hermeneutic circle where there are no absolute facts but only interpretations of content within a context.

For example, try looking up a word in the dictionary. A word is defined in terms of other words which eventually have definitions which circle back to the original word. From Webster [Rei92] the verb, to move, is defined as to go from one place to another with a continuous motion while the verb, to go, is defined as to move on a course. Each word is defined in a circular fashion having each other's word used in each other's definition. The two words together form a noemic concept associated with motion. Of course, there are many meanings of these two words, each dependent on a context. These two words participate in many noemic concepts.

Hermeneutic circles are like fast spinning toy top. An external observer, one outside the toy top, is given the tasks of riding, or understanding, the toy top. His first attempt is to step onto the toy top and is immediately thrown off. To be successful, first the observer must gain momentum, and match the motion of the toy top, and then, step onto the toy top. One can't understand the hermeneutic circle without first understanding the whole.

Edmund Husserl called the Hermeneutic circle paradigm a Noema [Dre79]. Noema is an antiquated Greek word for an intellect.

A Noema has the following characteristics:

- The implicit beliefs within a Noema and assumptions cannot all be made explicit.
- Practical operational understanding of a Noema is more fundamental than detached theoretical understanding.
- A representation of a thing cannot be complete.
- Understanding is fundamentally in the context of the whole and cannot be reduced to activities of individual sub- components.
- A sub-component cannot avoid its interactions with the whole.
- The effects of the sub-components cannot be absolutely predicted.
- All representations of the current state are ephemeral at best.
- Every representation of a sub-component is an interpretation with respect to the whole.
- Every action of the sub-component affects the whole, even non action. The presence of the sub-component affects the whole.

The so-called hard sciences have discovered that absolute knowledge is not possible. As an example, consider Gödel's incompleteness theorem of mathematics where he proves that there is no finite axiomization of arithmetic and that there are unprovable theorems in arithmetic. Heisenberg showed that in physics there are no absolutely accurate measurements and that all measurements are only known to with in an uncertainty. Even the act of obtaining a measurement changes the system that is being measured

and thus affects the measurements. Einstein showed that velocity can only be known with respect to a relative reference frame.

The traditional hard sciences have carved a very small domain out of the universal Noema. The actions of one component affects the whole and cannot be taken in isolation.

2.2 Traditional Engineering

Traditional engineering is decompositional in nature. To understand the whole, first decompose the whole into the constituent parts. Then master the individual constituent parts, put them back together again, and the whole is now understood. The understanding of the whole is the sum of the understanding of the parts.

A good example of traditional engineering is a car. Break the car down into its parts such as the steering sub-system, the transmission, the engine, the brakes, and many more sub-systems. The mastery of each of these sub-systems, leads to the mastery of the car. The whole (in this case, the car), is completely mastered by examining the parts in isolation and then looking at their combination to form the whole.

2.3 Noema

A Noema is the opposite from traditional engineering. Any one part cannot be understood without the context of the whole. Changes in one aspect affect the whole. The role of a part is a projection into the whole. In traditional engineering, first understand the parts, then understand the whole. In a Noema, first understand the whole, then understand the role of each part.

A good example of a Noema is the human body. It contains the sub-systems of circulatory, digestion, nervous, and many others. But the role of each part is highly interdependent on the other parts. One often hears a physician say, "I need to get the total picture first before I can treat this patient." A change in one subsystem cannot be isolated from the other sub-systems. Each individual cell acts independently, yet the whole is much greater than the sum of the parts.

Distributed computer environments are a Noema. This environment contains many highly interdependent components. Together they form a system.

Computer hardware is not a Noema. Hardware is based on hierarchical layering techniques appropriate for traditional engineering. A database is

not a Noema, again for similar reasons. A life form is a good example of a Noema built over millions of years with natural selection and evolution. An information based economy is another example of a Noema.

My contention is that a Noema is the correct paradigm for the software engineering of large distributed systems.

3 Defining Characteristics of the Noema

3.1 Knowledge

3.1.1 Partial Knowledge

All specifications are partial. There is no canonical definition of things, but rather examples and partial descriptions.

Every thing has a unique name, even yet undiscovered or unknown information. At any one time, a unique name is used as a place holder for the incomplete data. A place holder represents a piece of information, even if it is incomplete.

Every component should be able to generate partial results.

3.1.2 Local Knowledge

There is no one central repository of all knowledge. Each element in the system maintains the information necessary for that element to work correctly.

3.1.3 Smart Components

Each component in the system is smart, capable of performing tasks, generating results, and learning. In many ways, an individual component is a Noema itself. A Noema may be composed of millions of similar smart components that have specialized to handle one particular task.

The Noema is built from sub systems which in turn are Noema themselves.

3.2 Change

3.2.1 Constant Change

The only thing constant in a Noema is change itself.

3.2.2 Elaboration Tolerance

Accomplishing a similar task should be easy. The system should automatically adjusts to handle similar situations. Making small incremental improvements should be easy to accommodate.

3.2.3 Constant Death and Replacement

There is constant turnover of the individual components in the Noema. As the components age, they get replaced. Each component has an expected, finite, service cycle. New, more efficient, with equivalent functioning components supersede and replace the older components.

The Noema, in total, remains the same, while individual components change frequently. There is a constant replacement of worn out parts.

3.2.4 No Down Time

The Noema is designed to always be up and running. Individual components may be down, but the total, the Noema, stays available. In this mode of continuous operations things are evolving and changing constantly. There may be many versions of a component active at anyone time. Evolution of components, new feature introduction, and constant change are common, ongoing, activities.

3.2.5 Periods of Rest

Though the Noema is always available there are periods of rest or low activity where the system is able to return to a known, stable, initial state. The system is still there, but the load on the system is minimal, giving time for the Noema to repair broken parts and accommodate the changes.

3.3 Communication

3.3.1 Global Communication

Every component in the Noema can communicate with all other components in the Noema. There is a communication device which connects the components. There may even be many separate and disjoint communication devices and more than one way to accomplish communication between components. The communication need not be perfect. Errors occurring during communication are tolerated.

3.3.2 Asynchronous Communication

Communication is asynchronous and near stateless. A component accomplishes a small task and passes the results back to the Noema for further

processing. A particular task may be serviced by one of many equivalent functioning components.

3.3.3 Languages

The components communicate using a language designed to accomplish a task. Some languages may approach the complexity of a context sensitive natural language.

3.4 Organization

3.4.1 Distributed

A Noema is built from highly distributed components. No one component has control of the total system. Each component exerts influence to accomplish tasks. The elements in the system, for the most part, willingly cooperate to reach common goals.

The control paradigm is one of exerting influence. Decision making is delegated to the components closest to the problem where the impact of the negative and positive effects are felt the most. This creates a very narrow focused and localized decision process with quick response time. To make a decision that is global to the Noema requires exerting influence on the cost-benefit equation. By increasing the cost of a particular decision, the number of components making that decision will decrease. Likewise, by increasing the benefit of a particular decision, the number of components making that decision will increase. As a whole, the Noema moves in an influenced direction.

3.4.2 Replication and Groups

A Noema has many replicated components. The components combine together to form groups with similar roles. A request is made to a group of components and any one member of the group can accomplish the task. The number of components in a group dynamically grows and shrinks with demand. A task is given to the next available component leading to dynamic load balancing of the component group.

3.4.3 Massive Interdependency

Though it is counter intuitive, a Noema consists of interdependent parts loosely coupled together to accomplish a global task. The tight interdependence should cause a complete shut down of the Noema even on the smallest of permutations. Fortunately, this is not the case, because the Noema compensates by having large replicated groups of equivalent components, each capable of accomplishing the task. The groups are highly available thus enabling the interdependence with out affecting availability.

3.4.4 Self Fixing and Self Regulating

In a Noema, as components break, they get replaced with new equivalent components. The assumption is that components are breaking all the time though not so many break at any one time that the Noema ceases to function.

As more demand is placed on particular component groups, the Noema automatically grows to meet those demands. In some cases, this growth is pro-active where the usage patterns are anticipated in advance and appropriate components are established before the demand. This is in contrast to reactive growth where the components are not established until the demand already exists.

3.4.5 No Master Control

There is no one master control component in the Noema. Rather a decision is accomplished by exerting influence and building local majority. Consensus is not necessary and seldom is a global majority reached. See [Ell77] for algorithms on distributed vote counting.

The control of the system is by indicators, governors, and trend setting. An indicator measures the health of the Noema while a governor nudges the Noema in a particular direction. Trend setting provides long term strategies.

These indicators influence the distributed decision process.

3.4.6 Conscience and Unconscious Actions

The Noema has a large number of actions that happen automatically, without explicit activation. These are assumed to exist and form a underlying infrastructure.

A second class of actions are not automatic but are driven by external events or changes in the environment. These actions may in turn trigger actions in the underlying infrastructure.

3.4.7 No System-Wide Clock

The components in the Noema do not have exactly the same clock, but rather a clock that is close enough. The clock acts like a heart beat, giving an underlying cycle time for the system. The clock may be variable in speed, increasing or decreasing, governing the rate of production of the Noema.

3.4.8 No Global State

There is no global state in the Noema. For the most part, all components in the Noema are stateless. Limited local state may be maintained between small collections of individual components.

In place of state are indicators. An indicator can measure the health of the Noema.

3.4.9 Catalyst

A catalyst is a component that is used during a task but eventually returns to the Noema unchanged.

This concept is extended in the Noema to include data conversion routines, scrubbers, extractors, permutations, and general anything-to-anything conversions.

3.4.10 Common Building Blocks

Components are made from a collection of common building blocks. Components consume and generate the collection of common building blocks. There is a common representation for all things in the Noema. This is a necessary condition of a Noema. If the Noema is to learn, the Noema must be able to build and modify tasks. Thus tasks are to be treated as products produced and consumed by components in some cases. Not all building blocks can be interpreted as a task.

3.5 Security

3.5.1 A Security (Immune) System

The Noema has an aggressive security system. This systems detects and eliminates intrusions as well as repairs damage. The four fundamental components of a security system of protect, detect, eradicate, and restore are in effect.

Let me introduce these components of security with a simple example. Consider the problem of securing a room. First lock the door. This is protection.

Now, install a room monitor that detects when the door is open or closed and if there is any motion in the room. This is detection. When the room is secured the monitor should indicate that the door is locked and closed and the room is empty. Only on explicit known occasions should the door be unlocked and the room occupied. In all other cases the monitor should detect an intrusion.

Guards watch the room monitor. If an intrusion is detected the guards come to the room and mediate the effects of the intrusion. This is eradication.

Once the intrusion is eradicated, reestablish the door lock, and the room monitor. This is restoration.

Some intrusions may come in the form of a virus which may have assumed a false identity. One of the main tasks of the security system is to identify these intrusions, catalog their signature, and remove them from the system. The security system is pro-active by pre-allocating resources to protect, detect, eradicate, and restore from a security violation event.

3.5.2 Unique Name Space

Every building block, component, task, and thing in the Noema have an unique name. Communication to an exact component is sometimes necessary. This is accomplished with the component's unique name.

3.5.3 Authentication and Authorization

The Noema forms a cell of identity. A cell of identity defines the boundary which contains components that are known and trusted. The unique name space allows us to tag every component with cell identifiers. Any component that is not correctly tagged is expelled from the cell.

The outer cell may itself contain other Noemi each with there own cell boundaries. These inner cells form a hierarchy of cells. Only a qualified selection of components can gain access to inner cells.

Even though components are authenticated within a cell, they still need authorization to accomplish particular tasks.

There may be an outer cell wall followed by an inner cell wall followed by a nucleus with each boundary providing more security.

3.6 Persistence

3.6.1 Persistent Components

Some components in the Noema are persistent while other components are ephemeral. Noemi don't normally shut down in total. The essence of a Noema is continuous operation where at any one time many individual components are active with other components being inactive. Components that are no longer functioning may be replaced.

To maintain consistency across changes some knowledge is persistent and survives the change. Ideally, these persistent components are replicated so they too can be tolerant to component failure.

3.7 Boundaries

The universe is divided into three distinct regions. They are the Noema, the noemic cell boundary, and the environment.

3.7.1 Environment

The environment lies outside of the Noema. The Noema consumes resources from the environment and returns

3.7.2 Recycling System

As the Noema continues to operate, resources that are no longer usable accumulate. There resources are filtered and removed from the Noema but only after an attempt of recycling the spent resources.

As one might expect, fresh resources are added periodically to the Noema from the environment.

3.7.3 Peripherals

Peripheral devices reside at the noemic cell boundary and interact with the environment.

A Noema has many diverse sensors, actuators, and monitors feeding information between the Noema and the environment. The Noema gathers resources from the environment and returns un- usable resources back to the environment. If an environmental change affects the Noema operations in a negative fashion, the Noema may take actions to modify the environment.

4 Noemi

4.1 The Human Body

The human body is a Noema built from cells which are also Noemi.

The cell Noema exists in the environment of the human body Noema separated by the cell membrane. The cell takes resources from the environment such as proteins, carbonates, fat, water, and oxygen. Depending on the role of the cell, the cell could produce amino acids, hormones, and other useful organic chemicals.

The cell has only local and partial knowledge and is unaware of its position in the human body Noema. Many cells have short life cycles. Each cell is capable of performing many tasks independent of the actions of the neighbor cells though they are highly specialized leading to massive interdependence. A cell can replicate.

Many cells actions are governed by hormonal signals. The common building blocks of communication are amino acids over a common language of DNA. A cell caches some resources in anticipation of future demand. Cells communicate with each other using amino acids asynchronous messages.

The human body Noema consists of many interdependent subcomponents such as the respiratory system, the digestive system, the circulatory system, and the immune system. Security is provided by the immune system. Communication is provided by the circulatory system.

The heart provides a simple global clock modified by adrenaline. The human body Noema manipulates the environment by consuming resources and producing products and modifying the environment to make the Noema more successful.

The human body Noema uses the brain for persistent storage of knowledge. Natural selection supports the constant change. The human body Noema has a collection of conscience actions and a much large collection of unconscious actions like breathing.

4.2 A Market Economy

A market economy in a Noema.

The smart components of the Noema are individual people making decisions on local and partial knowledge. The market economy Noema consumes resources and generates products where the common language is money.

There is a global clock defined by the time zones. The cell boundary is loosely defined by political and social climates.

The market economy Noema has state given by such indicators as the gross natural product, inflation rate, unemployment rate, booking, sales, cash flow, sales of large ticket items, and stock price indexes.

The market economy Noema is governed and controlled by setting the such governors as interest rate, tax schedules, and laws.

4.3 Distributed Computing System

A distributed computing system is not yet a Noema. Many of the components are present but some are still missing or not fully integrated.

The network would be the communication mechanism for the distributed computing Noema supporting message passing, protocols, and asynchronous communication. The languages of communication are the protocols built up with bytes of data.

Replication and groups of services could be made available with special name space management services available on the network.

Some information may be kept in a data warehouse for analysis.

Some information could be locally cached. Some functions could be pre-evaluated and stored in anticipation of usage.

Both code and data may have a common representation. Thus programs are to be treated as data in some cases and programs in other cases. Not all data can be interpreted as a program.

The distributed computing Noema would need a security system with authentication, authorization, and data privacy.

The next chapters define how to build a distributed computing Noema.

5 A Noemic Design Paradigm

5.1 Paradigms

When building a new component in a Noemic system there are a few supporting paradigms that will help.

- Small changes.

  Try not to think about building a large system, but cast the problems in much smaller and simple improvements over the current system.

- Evolution.

  Evolve a solution from other known successes instead of building from scratch.

- Clone from the old.

  Start with an existing Noema and clone a copy, then modify the copy to meet the current needs.

- Assume that the whole already exists.

  When implementing the Noema, assume that the whole already exists. Push off calculations to another component. Try to build only very simple components. Let the Noema environment do the integration.

- Assume that the environment already exists.

  Assume that a powerful support environment already exists and write the components accordingly. As it is discovered that elements are missing, recurs on the Noema process to build them.

  See Figure 2 on page 22.

5.2 The Noema Process

Assume that the whole already exists and an incremental improvement to the whole needs to be accomplished.

How can this be? If the whole already exists how do I get started when there is no Noema? The noemic bootstrap process can be based on traditional engineering methodology. First build simple elements forming the base level

Figure 2: Noema System Architecture

infrastructure and surrounding environment. Once this is in place, use these elements to bootstrap higher level noemic components.

Every new component must follow the rules of the Noema. This is best done with a template created by copying an existing working component.

What if this is the first component? At the minimum the noemic component template must provide:

- a list of resources necessary to perform one action
- a list of results
- a list of returnable resources to the environment
- a list of catalysts
- a list of the actions done by this component
- a mechanism of caching resources and results
- a mechanism to communicate
- a script interpreter
- authentication
- authorization
- a cell identity
- a group identity
- a component identity

Every component should be as small as possible, performing only one action. If the action requires other actions, simply do the local component action and then send it back to the Noema for further processing.

From the environment we expect a collection of services include persistent storage, name space managers, networks, transactions, programming languages, operating systems, file systems, databases, transactions, computer hardware, and locks.

A program is an interpreted script with all the complexity of current computer science combined with the current local state of the computation.

This local state includes a program counter and other temporary values that would be normally stored in memory. The local state, if any, is stored along with the script.

In general, a program is dispatched to a component that can handle the current task. The component acquires the resources, evaluates the tasks, advances the program counter to the next step, updates the local state of the program, cleans up after the tasks, and then returns the program to the Noema for further processing.

This scripting language could represent a complex work flow.

6 Theory and Examples

6.1 Comparison Criteria

In the following sections, very simple noemic algorithms are introduced and compared to their more traditional counterparts.

The comparisons are based on:

Calculation: The cost of the computation.

Latency: The time to receive the first results.

Throughput: The cost of the computation once the system is primed.

Resource: The number of elements.

One Fault Tolerance: If one component fails in the system, the entire system fails.

Fault Tolerance Threshold: The number of components that have to fail to make the total system fail.

Elaboration Tolerance: The cost of a small change in the computation.

Figure 3: Cost of Simple Computations

Consider the simplest system consisting of two components with no replication. The traditional architecture and the noemic architecture are in Figure 3 on page 26.

Let **a** be the cost of each component in the calculation. Let **c** be the cost of communication. The simple computation is done in one element using the traditional engineering paradigm. In a Noema, the calculation must be split into two elements forming a cycle. Each element only performs part of the calculation and only by looking at the whole can we describe the role of each component.

The comparison results are shown in Table 1 on page 27. The Noema performs worse when compared with the traditional computation with re-

gards to the costs of calculation, latency, through put, and resources. The Noema, however, shows elaboration tolerance.

The Noema is harder and more complex to establish than a more traditional system but once the Noema is established it can be easily changed. This will be a constant theme throughout the rest of the examples.

Measurement	Traditional	Noema
calculation	2c+2a	4c+2a
latency	2c+2a	4c+2a
throughput	2c+2a	2c+2a
resource	2c+2a	4c+2a
one fault tolerance	fails	fails
fault tolerance threshold	1	1
elaboration tolerance	2a	a

Table 1: Comparison for Simple Computations

Figure 4: Cost of Replicated Computations

Consider the simplest replication of a system consisting of two components with **n** replications. The traditional architecture and the noemic architecture are in Figure 4 on page 28.

Let a be the cost of each component in the calculation. Let **c** be the cost of communication. Let **n** be the number of replications.

The comparison results are shown in Table 2 on page 29.

Replication does not help the traditionally built application when it comes to elaboration tolerance.

Measurement	Traditional	Noema
calculation	2c+2a	4c+2a
latency	2c+2a	4c+2a
throughput	2c+2a	2c+2a
resource	2c+2na	4c+2na
one fault tolerance	success	success
fault tolerance threshold	n	$between n and 2n inclusive$
elaboration tolerance	2na	na

Table 2: Comparison for Replication Computations

6.4 Complex Replicated System

Consider a complex replication of a system consisting of **m** components with n replications. The traditional architecture and the noemic architecture are in Figure 5 on page 30.

Let **a** be the cost of each of the **m** components in the calculation. Let **c** be the cost of communication. Let **n** be the number of replications.

The comparison results are shown in Table 3 on page 29. In a computation which has many components the Noema shows elaboration tolerance.

Measurement	Traditional	Noema
calculation	2c+ma	4c+ma
latency	2c+ma	4c+ma
throughput	2c+ma	2c+ma
resource	2c+mna	4mc+mna
one fault tolerance	success	success
fault tolerance threshold	n	$n <= t <= (m-1)n$
elaboration tolerance	mna	na

Table 3: Comparison for Complex Computations

Figure 5: Cost of Complex Replicated Computations

6.5 Theorems

6.5.1 Elaboration Tolerance

For simple systems a noemic architecture has very limited value, even with replication. It is no worse but not much better than the traditional architectures. But as we increase in the complexity of the system, and as more and more components are involved, the Noema begins to show it advantages. As we know from software engineering, most of the cost of the system is in the total life cycle and not the original engineering cost. A Noema has the characteristic of elaboration tolerance where a small change is easy to accommodate.

Theorem 1 (Elaboration Tolerance) *A noemic architecture will respond to change better than a traditional replicated architecture for a system with many complex components.*

Proof 1 *Let* **m** *be the number of components. Let* **n** *be the amount of replication. Let* **a** *be the cost of the computation. The cost of the traditional replicated architecture of a small change is* **mna**. *All* **n** *replicates of the* **ma** *computation must be modified to accommodate a simple change.*

The cost of the noemic architecture is **na**. *Only* **n** *replicates of the simple computation* **a** *that changed needs to be modified.*

In a complex system with many components $m >> 1$. *Thus* $mna >> na$.

Therefore, the Noema responds to change better than a traditional replicated architecture for a system with many complex components. The Noema shows elaboration tolerance.

6.5.2 High Establishing Cost

Theorem 2 (High Establishing Cost) *A noemic architecture based system is more costly to establish than the more traditional architecture based system.*

Proof 2 *Each simple component of the Noema requires communication and infrastructure overhead not required by the more traditional components to deal with under specified components and partial knowledge.*

The development cost for these items are high.

6.5.3 Noema Computational Power

Theorem 3 (Noema Computational Power) *A noemic system has a higher computational power than a more traditional system based on established compilers.*

Proof 3 *In the theory of languages and grammars there are three distinct types numbered 1 to 3 with type 3 grammar having the lowest computational power.*

A type 3 grammar can be expressed as regular grammar and represented by non-deterministic or deterministic finite automata.

A type 2 grammar is a context-free grammar and represented by a push down automata either non-deterministic or deterministic. A type 2 grammar has more computational power that a type 3 grammar. Traditional computer languages, such as C, Pascal, Ada, and C++, all are representatives of this type.

A type 1 grammar is context-sensitive and represented by a linearly bounded turing machine. A type 1 grammar has more computational power that a type 2 grammar. A Noema, being context-sensitive, is representative of this type.

Thus a Noema system, a type 1 context-sensitive system, has a higher computational power than a more traditional system based on established compilers, type 2 context-free systems.

7 An Example

7.1 Introduction

The Internet is the heart of the World Wide Web but, as many who try to use the Internet know, it is lacking many features. This example is to show how to grow the Internet into a Noema.

The Internet gets a grade of B as a Noema. The Internet has many of the defining characteristics of a Noema include partial and local knowledge managed by semi-smart components. Change is somewhat supported with the aid of protocol version negotiation. The Internet has no down time even though many components are leaving and entering the Internet at anyone time. The Internet supports global asynchronous communication. There is no system-wide clock and no global state. The languages of the Internet are defined by the various protocols including SMTP and HTML. The Internet is highly distributed with no central master control.

What the Internet is missing is a security (immune) system, an authentication system, an authorization system, many unique namespaces, and peripheral management. The Internet needs to become stronger in the areas of self fixing and self regulating algorithms and distributed service support using replication and service groups.

The following sections first define the established characteristics of the Internet. The missing characteristics are then introduced to grow the Internet into a Noema.

7.2 Background

See Figure 6 on page 34.

The network has its foundations in the International Standards Organization(ISO) Open Systems Interconnection(OSI) seven network layers consisting of physical, data link, network, transport, session, presentation, and applications. This layering has not been precisely followed in the Internet and a more modern view would collapse the Internet into four layers consisting of physical link, network, transport, and applications [Wal97][Tan96].

7.2.1 Physical Link

See Figure 7 on page 35.

Figure 6: The Network Models

Physical Link Layer Example

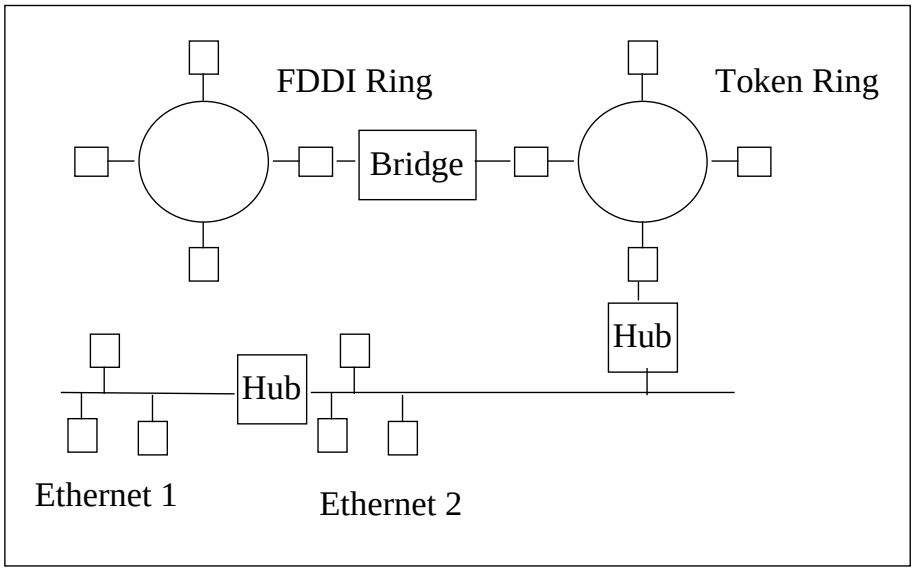

Figure 7: The Physical Link Layer

In the physical link layer, data is sent from physical network address to physical network address and is broadcast over a logical single-wire network which is hooked together with hubs and bridges with the aid of the Address Request Protocol(ARP).

Examples of technology at this level include ethernet, token ring, and Fiber Distributed Data Interface(FDDI). A hub makes many physical wires look like one logical wire. A bridge is like a hub but with knowledge about the physical topology of the network that is used to optimize performance.

Token rings and FDDI are very similar. Both use a token to manage ownership of the physical wire. Once a computer on the wire has the token, the computer is allowed to send data or pass the token to the next computer on the wire. In a token ring system the token is not released until the data has moved around the entire ring. In a FDDI system the token is released as soon as the data is on the fiber. In theory, a FDDI is more efficient than a token ring but in practice, the token in a FDDI is susceptible to being lost, leading to wasted cycles.

If a computer on an ethernet wishes to place data on the wire, it first listens to see if the wire is calm and therefore available. If the wire is available then the data is placed on the wire.

There is a race condition that might develop. Simultaneously, two computers might think that the wire is available and begin to send data. The data will collide during transmission. Each computer is must monitor the transmission, detect if there is a collision, and resend the data at a later time if necessary.

Every network interface card has a vendor-supplied physical address. This is also called a hardware address or a Media Access Control(MAC), address.

At the lowest level of the network every packet on the network is a frame packet sending information from one MAC to another MAC. The MACs that are reachable on a network are discovered with the help of the ARP protocol.

See Figure 8 on page 37.

Each hub maintains a MAC table to help it manage the local packets.

See Figure 9 on page 38.

Figure 8: The Frame or MAC Packet

Figure 9: A MAC Table

7.2.2 Network Layer

See Figure 10 on page 39. Notice that an IP network is a collection of physical networks connected by routers. This is emphasized in Figure 10 by usin

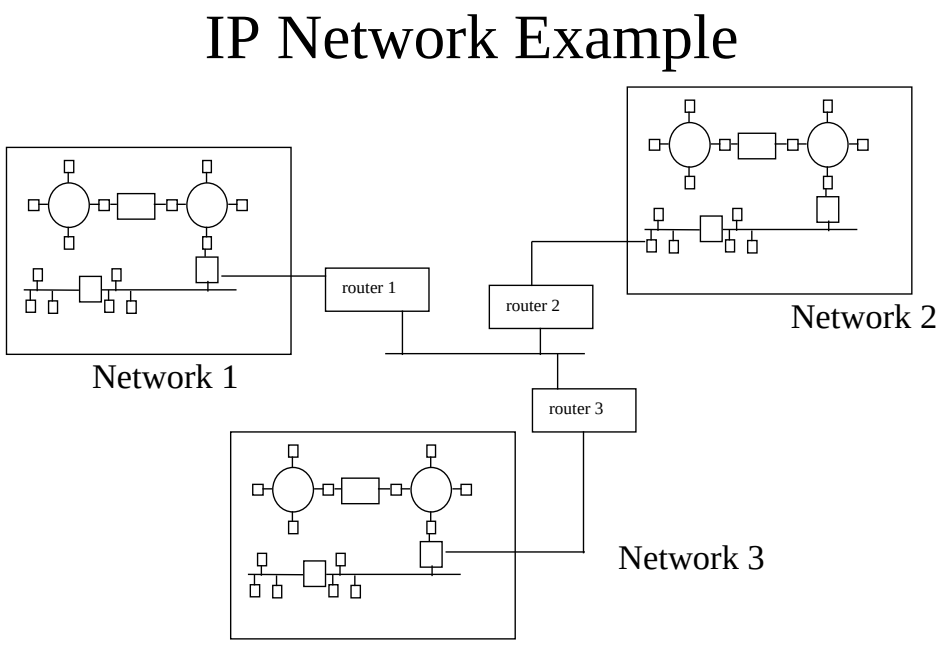

Figure 10: A IP Network

Network layer data communication is from one IP address to another IP address using datagrams connected together with routers with the aid of Domain Name Service (DNS).

See Figure 11 on page 40.

A IP network consists of many physical networks connected together by routers. Each computer on the IP network has a simple routing table that sends IP packets to the router. At each router is a much larger table representing routing information for the entire IP network. Given a destination IP address, the table returns the next hop.

10.014in,7.5in]iptable.eps

Figure 11: A IP Routing Table

See Figure 12 on page 41.

The DNS materializes a namespace for all the computers in a domain. DNS will map computer names to IP addresses. There is one domain for each IP network. A large collection of domains form the Internet.

See Figure 13 on page 42.

To prevent cycles, each IP packet has a maximum hop count. Each time the IP packet passes through a router, the hop count is decremented. If the count reaches zero, the IP packet is dropped.

If the IP packet is too big for the physical layer, the IP packet is fragmented into smaller pieces. Each piece is then delivered and the IP packet is reconstructed at the destination.

7.2.3 Transport Layer

Transport layer data communication is from application to application. The application address is defined by an IP address and port combination.

See Figure 14 on page 43.

At each computer there is a port manager. Each application is associated with a port. Data communication from one application on a computer is defined by the IP address and port combination.

When two applications want to communicate, they establish a session to another application. All TCP packets in the session are sequenced and the order of delivery is maintained. This is accomplished with the aid of the sequence number in each TCP packet.

See Figure 15 on page 44. Note that ack. num. is an abbreviation for acknowledge number and that seq. num. is an abbreviation of sequence number.

Since one computer may have different performance characteristics than the other computer in the session, the two computers enforce flow control of packet rate. This is accomplished with the aid of the maximum packet count field in each TCP packet.

Figure 12: A DNS Table

Figure 13: A IP Packet

Figure 14: The Transport Layer

Figure 15: A TCP Packet

Figure 16: A inetd Table

The inetd daemon manages the ports on each machine. Some of these ports are pre allocated and established by convention. For example, email is on port 25.

7.3 Noema based Internet

The Internet, as defined by the four layers of physical link, network, transport, and applications is not quite a Noema. There are several key missing components. Among these include the concept of a unique identity, a principal, an authentication system, an authorization system, a stronger data protection system, and a family of namespaces. All are under the watchful eye of a security system.

7.3.1 Unique Identity

Every thing in a noemic network needs a unique identity. Given the unique identity, the type of the identity can be established. This could be done with the aid of a map or by having a type field in the identifier itself.

7.3.2 Principal

A person, computer, or application service is a principal. There is a namespace of principals with associated demographical information.

Principals can be grouped together, forming a hierarchy of groups.

The principals may represent groups of services, which yield dependability and reliability.

7.3.3 Authentication

Given a principal's identifier and a password the authentication service verifies the identity. This could be based on public or private key protocols.

7.3.4 Authorization

With each principal is a list of groups of which the principal is a member. These groups grant capabilities for the principal.

On each thing in the noemic network is an access control list(ACL). The ACL is a property list of group and permission pairs. The principals in each group have stated permissions for what they can accomplish. These are granted by the authorization service.

If group membership is very dynamic, the groups take on roles. The membership of the group could be guided by a rule base.

The principals are granted access and privileges with a role-based authorization algorithm based on dynamic groups and access control lists.

7.3.5 Data Protection

Information, or data, can be protected in at least two ways.

Data Consistency When checking data consistency, a digital signature verifies that the data has not been changed.

Data Privacy By enforcing data privacy, only the intended recipient will be able to view the data. This could be accomplished with many encryption algorithms.

7.3.6 Namespaces

To aid the noemic system, there are many namespaces. The role of the namespace varies but, in general, it maps an identifier to a collection of demographical information.

Services A namespace exists that indicates where to find a desired service. Services can be grouped into service groups and then load balanced.

Principals A namespace of principals maps the principal identifier into a collection of demographical data.

Unique Identity A namespace of unique identities returns the type of the identifier.

7.3.7 Security

There are four basic premises of security. First protect from an intrusion, then detect an intrusion, then confine the scope of an intrusion, and then repair the damage caused from the intrusion and bring the noemic system back to a known secure state.

Automated software security agents monitor the system to provide the protection of information, the detection of a breach of a security mechanism,

the confinement of the breach to a small a footprint as possible, and the repair of breach bringing the security system back to a secure known state.

Protect Protection can be accomplished in many ways. The authentication, authorization, and data privacy systems play a major role in protection.

Detect No security system is perfect. An intruder, by a back door, could compromise the security protection. This event needs to be detected.

This could be accomplished by a trip wire. A trip wire notices changes to state. Many changes are expected and do not represent a security compromise event. Other changes are unexpected and could represent a security event.

Confine Once the system has been compromised, the intrusion needs to be confined. This can be accomplished by dividing the environment into a number of well defined components.

Many times an intruder is the best source of information about his intent. A security system could move the intruder to a play room. The play room is a copy of the original environment where similar, but non-critical, information is kept. The security system then logs the intruder's actions. A review of the log tells a lot about the scope of the intrusion.

In the play room, the security system gives enough information to the intruder to discover intent without compromising more of the system under protection.

Repair The security system then needs to repair the damage and return the system to a known secure environment. The intruder is removed, the back door is locked, the environment is repaired, the trip wires are reestablished, and the front door is locked.

7.3.8 Persistent Storage

A namespace of persistent objects is be available that allows for the secure sharing of information across the network. Given the name of an object and the correct privileges, access to the object is granted. This namespace would uncouple the name of an object from the location.

Figure 17: Principal to Principal Layer

Figure 18: Principal to Principal Packet

New applications can call the PTP directly. Existing applications need not be modified. The socket library would be replaced with a default PTP layer and a new authentication client needs to be made available.

7.3.11 Program Communication

Programming a Noema deals with communications, or protocols more than algorithms and data structures. The protocols negotiate a common version and can query each other for capabilities.

The design goal is to build a series of supporting algorithm-independent protocols that provide authentication, authorization, and data protection.

This is similar to the approach of SMTP for email. SMTP has remained quasi-static while the technology and algorithms to support email have changed significantly over the last several decades.

A Glossary

Asynchronous : An attributed relating to the separation by an unpredictable amount of time the relationship of a causal event and it's effect.

Authorization : The process with grants privileges to an authenticated identity.

Authentication : The process which verifies an identity.

Canonical Representation : A representation that can all encompassing. Every element of the domain can de definitively expressed in terms of a canonical representation.

Catalyst : A component in the Noema which helps other components perform their task but is neither consumed or produced by the other components actions.

Data Warehouse : A persistent repository of historical information usually used for analysis, summaries, and decision support by a family of unintegrated applications.

Elaboration Tolerance : An attributed of a system where a small change is easy to accomplish.

Life Cycle : The cradle to grave existence of a software system from initial conceptions, through development, through deployment, through version releases, and final phase out.

Paradigms : A point of view on how to understand and solve a problem.

Persistent : A property indicating stability over a long period of time.

Protocol : The communication behavior between two or more identities.

B Acronym Key

GUI Graphical User Interface

References

[Dre79] Hubert L. Dreyfus. *What Computers Can't Do: A Critique of Artificial Reason.* New York: Harper and Row, 1979.

[Ell77] C. Ellis. A robust algorithm for updating duplicate databases. In *Proc.of the Berkeley Workshop/conference on Distributed Data Management and Computer Networks, LBL 2, UCB.*, pages 146–158, May 1977.

[Hei68] Martin Heidegger. *What is Called Thinking?* New York: Harper and Row, 1968. Translated by Fred D. Wieck and J. Glenn Gray.

[Rei92] Aaron Reizes. Webster/thesaurus: A network dictionary client., 1992. Macintosh based computer software.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM 21(2)*, pages 120–126, February 1978.

[Tan96] Andrew Tanenbaum. *Computer Networks, Third Edition(ISBN 0-13-349945-6).* Prentice Hall, 1996.

[Wal97] Jean Walrand. *Communication Networks: A First Course. Second Edition.* Aksen Associates, 1997.

[WF85] Terry Winograd and Fernando Flores. *Undersanding Computers and Cognition: A New Foundation for Design.* Ablex Publishing Corporation, Norwood, New Jersey, 1985.

Index