

ENG

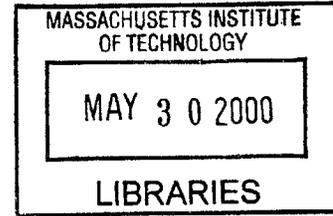
Candidate System Architecture for Intelligent Integrated Infrastructure

By

Charles O'Bryan Fleet

B.S. Civil Engineering
Virginia Military Institute, 1999

ENG



SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING
AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE, 2000

© 2000 Charles Fleet. All Rights Reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper
and electronic copies of this thesis document in whole or in part

Signature of Author:
Department of Civil and Environmental Engineering
May 5, 2000

Certified by:
Professor Jerome J. Connor
Thesis Supervisor

Accepted by:
Daniele Veneziano
Chairman Departmental Committee on Graduate Studies

Candidate System Architecture for Intelligent Integrated Infrastructure

By

Charles O'Bryan Fleet

B.S. Civil Engineering
Virginia Military Institute, 1999

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE, 2000

Abstract

During the explorative and concept stages of Intelligent Integrated Infrastructure, it is essential to establish an optimal system architecture outlining task separation and component structure. The architecture must be comprehensive and complete while also abstracting itself from specific implementation and current technology because of the large scale of the system and the length of development time. By using Object Oriented Organizational principles, a proper architecture can be defined allowing proper functionality through componentization, abstraction, and scalability.

Thesis Supervisor: Prof. Jerome J. Connor

Title: Professor

Acknowledgments

Parents and Family

My Close Friends

Professor Connor

Professors Madnick and Kocer

All of M.Eng.

Skyauk for formatting assistance

Table of Contents

Chapter 1 Introduction	7
1.1 FROM SMART STRUCTURES TO INTELLIGENT INFRASTRUCTURE	7
1.1.1 First Generation (Passive/Reactive)	7
1.1.2 Second Generation (Active)	8
1.1.3 Intelligent Integrated Infrastructure	8
1.2 OBJECT ORIENTED THINKING	8
1.3 NOT IN SCOPE.....	10
1.3.1 Current technology	10
1.3.2 Development Procedure	11
1.3.3 Personnel Structure.....	11
Chapter 2 – Issues for Design	12
2.1 INTRODUCTION	12
2.2 METHOD OF CREATING DESIGN	12
2.3 REQUIRED DESIGN CHARACTERISTICS	13
2.3.1 Object Oriented Organization.....	13
2.3.2 Abstraction.....	14
2.3.3 Scalability.....	14
2.3.4 Flexibility.....	15
2.3.5 Quality.....	15
2.3.6 Robustness/Stability	15
2.3.7 Reliability	16
2.3.8 Safety and Catastrophe handling.....	16
2.3.9 Risks.....	16
2.3.10 Security.....	17
2.3.11 Inference Ability.....	18
2.3.12 System Renovation	19
2.3.13 Cost.....	22
Chapter 3 – Proposed Task Division Architecture.....	23
3.1 COLLECTION	24
3.2 INITIAL PROCESSING.....	24
3.2.1 Baseline	24
3.2.2 Reliability	25
3.2.3 Emergency	25
3.2.4 Specific Monitoring.....	25
3.3 DATA STORAGE	25
3.3.1 Location.....	26
3.3.2 Duration of Storage.....	26
3.4 DATA ORGANIZATION.....	26
3.5 DATA PROCESSING	27
3.6 EXECUTION ON FLAGS	28

3.7 DEFINITION OF FLAGS	28
3.8 TOTAL SYSTEM DESIGN AND REQUIREMENTS.....	29
3.9 SYSTEM CONTROLS.....	29
3.9.1 System Failure Control	29
3.9.2 System Optimization	29
3.9.3 Reliability	30
Chapter 4 – Semicentralized Object Network.....	32
4.1 ARCHITECTURE CAN BE APPLIED AT ANY LEVEL	32
4.2 PROCESSING AT POINT OF CAPTURE.....	32
4.2.1 Discussion of Bottleneck calculations	32
4.2.2 Lower Bandwidth required	33
4.2.3 Flag processing at N level	34
4.2.4 Both Server and Client can be fat	35
4.2.5 Sharing tasks through Load Balancing across N tier	36
4.3 EACH COMPONENT IS SELF MANAGING.....	36
4.4 DATA HIDING.....	36
4.5 NETWORK CONNECTION.....	36
Chapter 5 – Conclusion	38
5.1 TRANSPARENCY OF STRUCTURAL AND INFORMATION OBJECTS	38
5.2 REMOVAL OF CURRENT TECHNOLOGY.....	38
5.3 USE OF TECHNOLOGY IN THE FUTURE	38
5.3.1 Wireless	38
5.3.2 Faster Computing Speed.....	39
5.3.3 Different Storage of Data Procedures.....	39
5.4 OBTAINABLE BENEFITS	39
5.4.1 Controlled Modeling	39
5.4.2 Total systems optimization and safety	40
5.5 DIRECT FUTURE OF RESEARCH AND DEVELOPMENT	40
Chapter 6 References	41

List of Figures

Figure 1.1 – Abstraction of Collection Properties	9
Figure 2.1 – Abstraction Level vs. Comprehension Level	14
Figure 2.2 – Information system life cycle evolution.....	20
Figure 2.3- Data Model Progression	21
Figure 3.1 – Total System Architecture	23
Figure 3.2 – Data Processing	27
Figure 4.1 – Pure Centralized Computing	33
Figure 4.2 – Decentralized Computing with N+1 Flag passing	34
Figure 4.3 – Modified Duel Computation Ability.....	35

List of Tables

Table 1.1 – Comparison of Abstracted Properties.....	10
--	----

Chapter 1 Introduction

Intelligent Integrated Infrastructure (I³) is the collective operation of infrastructure components in one system allowing for data storage and exchange, modeling simulation capabilities, and decision support systems. The immediate advantages are comprehensive modeling ability, operational optimization across systems, and better enabled execution of actions. This paper has three main sections: A discussion of required issues that must be considered when designing and architecture, the proposed architecture of task separation, and general points to consider when implementing the proposed architecture.

1.1 FROM SMART STRUCTURES TO INTELLIGENT INFRASTRUCTURE

Before considering Intelligent Integrated Infrastructure, a short discussion of smart structures is essential. Even within smart structures though, there is a distinction between first and second-generation smart structures. First generation smart structures characterized by passive or minimally reactive systems operating functionally separately from their enclosed infrastructure. Second generation smart structures are active systems integrated individually into and part of their enclosing infrastructure.

1.1.1 First Generation (Passive/Reactive)

First generation smart structures provide information for users and basic level administrators of the structure. Information provided for users could be automated and personalized directory for a visitor to a building, or the amount of credit left on to their account as they drive through a toll. Information provided to the basic level administrator could be the total electricity, phone, and water usage over any recent time interval. This information, although it can be current and continually updated, is not active by nature. It is primarily statistical data which is collected and either stored or used at the time of capture. It is generally one directional from point of capture to presentation to the user. It then can not be used to impact the operation and maintenance of the system except through a specific user or administrative intervention.

These applications do optimize the static nature of infrastructure allowing many added features to the building, but they do not actually change the infrastructure. The features added are exactly that: added to the existing infrastructure as separate systems, not integrated and part of the actual system. Consequently, there is no way to handle structure modeling, predictions, or optimization except through the collection of statistical data.

1.1.2 Second Generation (Active)

The transportation industry and specifically smart highways have developed the closest implementation of second generation smart structures through their development of active highway control systems. They have been able to integrate into highway and road systems the optimization of traffic flow, accident handling, toll taking, automatic cars (full cruise control at highway speed), weather conditions, and many other aspects of vehicle travel.

In second generation smart structures a single system is designed from the beginning with integrated technology meant to optimized, monitor and model the structure. The system though acts individually, and each particular aspect of technology is independently integrated into the system.

1.1.3 Intelligent Integrated Infrastructure

The final step of intelligent integrated infrastructure is to implement it across types of infrastructure using one common architecture enabling cross system information flow. Coastal, utilities, urban, and transportation are some of the types of infrastructure that can be immediately impacted, allowing optimization of maintenance procedures, resource allocation, and future planning. With the different types of infrastructure IT enabled, a virtual city can be realized, allowing a modeling of the city as interactive and intertwined systems.

1.2 OBJECT ORIENTED THINKING

Within an object oriented organizational strategy everything considered can be parsed into a specific object. Each object has two properties: the information it holds and the actions it

can perform. The objects created can be either completely tangible with a physical representation, or can be completely intangible without and physical representation. Consider the two following two examples:

Structural Beam

A beam has attributes such as length, unit weight, max shear stress, etc. It also has different actions associated with it such as to carry load, to bend, to enter plastic failure, etc.

Database

A database can be considered (removed from the actual method of storing the data) as a collection of data with specific properties and operations associated with it. The database can have a total number of entries, size per entry, maximum acceptable entries, etc. The actions that can be associated with the database could be to enter a new entry, to delete an entry, to return a value given a set of parameters within a query (e.g. SQL), etc.

In each case, each object is self contained with its own properties. And each object has a template which has properties to be specified if it is implemented. But in both the structural beam and the Database it is evident that there are definitive properties and methods associated with any object.

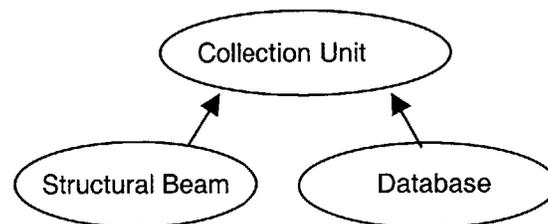


Figure 1.1 – Abstraction of Collection Properties

By defining objects, you also begin to introduce abstraction. It could be considered that Structural Beam and Database are merely types of a higher object: Collection Unit. As seen in Figure 1.1, they would both be derived from a collection unit because a structural beam collects force and a database collects data. But Table 1.1 shows how the properties of the two lower objects can be abstracted to the higher level of the Collection Unit type.

Table 1.1 – Comparison of Abstracted Properties

Structural Beam		Collection Unit		Database
Length	→	Size	←	Number of Entries
Unit weight	→	Unitized Size	←	Size per entry
Add Load	→	Add to Collection	←	Add entry

This abstraction as clearly seen, causes discussions of organization to become vague and sometimes meaningless. Abstraction is required though for comprehensive coverage and consideration of all types of possible implementations. Throughout the paper, a certain level of abstraction in the definition of the architecture is maintained while trying to remain structured and based on physical and organizational concepts. †

1.3 NOT IN SCOPE

1.3.1 Current technology

If the same technological progress that has been made in [the computer] industry over the past thirty years had been made in the automotive industry, you could now purchase a Cadillac for less than a quarter. Your car would be a thousand times more reliable than a 1950 Cadillac and would have an EPA rating of over a million miles per gallon.¹

With this well known quote about the computer industry clearly shows how quickly technology changes. Intelligent Integrated infrastructure is still in the very earliest stages of

† See further discussion on Abstraction in Sec 2.3.2 on page 14

¹ Wetherbe, pg. 6

exploration and is decades from full integration into industry standard construction. Because it is the goal of this paper to discuss an optimal architecture for I^3 , a discussion containing the current technology would be counterproductive and cause the discussion to be dated. The removal of current technology does abstract the discussion and does make it harder to define, but can be avoided by concentrating on the principles of the architecture and the connectivity.

1.3.2 Development Procedure

This paper intends not to outline how I^3 can be realized, but more what the architecture of the I^3 will be. Development of systems and projects can take place through many methods while keeping the goal constant. The time frame until implementation is dependent on many factors like financial feasibility, industrial support, and the state of technology. Therefore the time required to develop the system architecture will not be considered.

1.3.3 Personnel Structure

No matter how automated we would like to design a I^3 , it will still require a staff of personnel involved in the operation and maintenance of the system. Because this is completely dependent on the actual requirements of each system developed and the automation level of each system it can not be considered at this level of development.

Chapter 2 – Issues for Design

2.1 INTRODUCTION

When designing any system, regardless if it is structural, organizational, or informational, certain properties must be considered. These properties are independent of both the type of system and the use of the system. These include:

- Organization
- Scalability
- Robustness/Stability
- Quality and Reliability
- Safety and Catastrophe handling including Risk Management
- Security
- System Renovation ability
- Cost

There are some properties though that are more specific to Information Systems and data handling that do not have clear equivalents in a physical system:

- Level of Automation
- Level of Abstraction

Because this is a candidate Architecture for a specific purpose, I^3 , and we intend to have a Object Oriented Organization then we must define our issues for design more specifically before actually discussing the design itself.

2.2 METHOD OF CREATING DESIGN

There are three macro stages involved in the realization of I^3 . In the first stage exploratory/candidate architectures are researched. This is the current stage of I^3 . The purpose of this stage is to consider different architectures that can be implemented for the entire system and for defining the controls for those systems. This stage is highly theoretical and does not involve any real testing or any integration of current technology. From the controls that are developed, a smaller set of candidate architectures can then be explored in the next stage.

The primary activity of the second stage is to narrow down the field of possible candidate architectures ultimately to one and to further refine the control criteria for the system. Actual field-testing and some integration of current technology must occur in order to determine the optimal architecture. Once this final architecture is developed full scale implementation can begin.

The final stage is the full scale integration into industrial standards and practices. This requires a consideration of the parameters for each implementation and integration of the current technology.

2.3 REQUIRED DESIGN CHARACTERISTICS

2.3.1 Object Oriented Organization

As mentioned in Section 1.2 the role of Object Oriented Organization (O^3) is essential to the System Architecture. Some of the issues associated with object oriented organization are data hiding, information passing, and abstraction.

The key to O^3 is the way that information flows between objects. Because each object has its own hidden data and functionality, the only way to access and manipulate the data is through pre-defined methods. For the purposes of the I^3 architecture, the following definition is required:

Flag Passing: the action of passing information between objects causing transmission of data, notification of an event, or call of method within an object.

The advantages of an O^3 are the optimization of the system characteristics like abstraction, scalability, flexibility, robustness and stability, reliability, security, and system renovation. The advantages though are predicated on deliberate controlled system architecture.

2.3.2 Abstraction

When designing a system it is important to maintain a proper level of abstraction. Abstraction is always required when operating in an object oriented environment because of inheritance. If you are given for example a sedan, a road tractor, and a dump truck these items can be abstracted to the higher level of vehicle. These three objects share the characteristics of max speed, fuel capacity, etc. and therefore can be defined within the abstracted vehicle level.

A certain balance of abstraction must be maintained though. If you do not abstract your analysis enough there will be too many components with too many relationships between them. If you abstract too much there is a shorter presentation but the ability of the designers to physically comprehend the system is hindered. This is shown in Figure 2.1.²

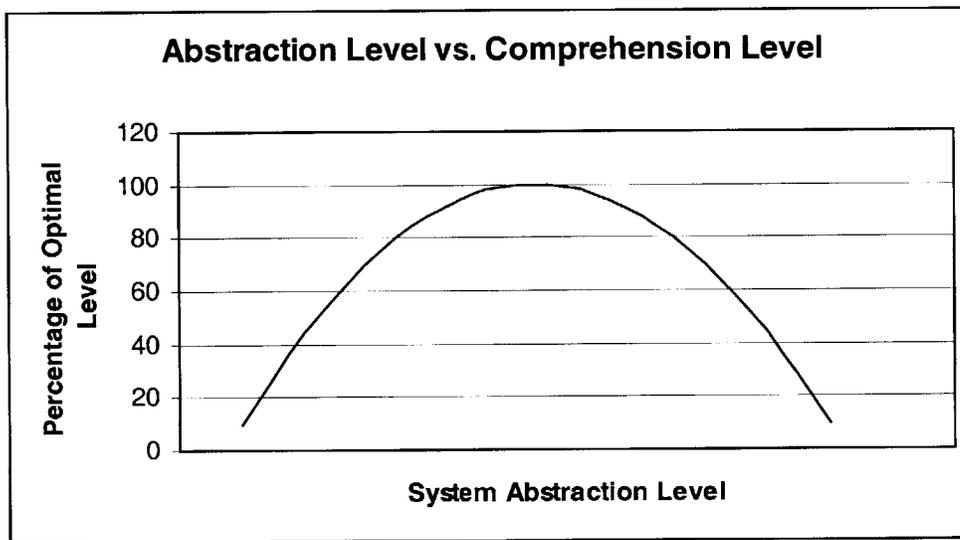


Figure 2.1 – Abstraction Level vs. Comprehension Level

2.3.3 Scalability

Scalability is the ability to maintain desired functionality and organization under any size of the implementation. In I³, scalability is essential because after a system is implemented

there might be a demand to increase the size. Without inherent scalability, a total system renovation would be required instead of a system expansion. Scalability can be obtained by ensuring that the operation of the system design is not dependent on the number of objects in that system.

2.3.4 Flexibility

Merriam Webster defines flexibility as an object characterized by a ready capability to adapt to new, different, or changing requirements. Specific examples of flexibility within I³ are:

- Ease of upgrade to the new technology
- Ease of integration with other products
- Ease of enhancements
- Transportability between multiple technical environments.

Flexibility is different than system renovation in that the operation and organization of the systems do not need to be changed, only the type of implementation.

2.3.5 Quality

There are two different definitions of Quality when describing a system. The first is the quality of the system architecture and design and the second is the quality of the system implemented. The quality of the architecture and design is defined and bounded from the initial stages of development. It is independent of the implementation of the system. The quality of components though, is dependent on implementation of the system. If the specific implementation prescribes lower quality components or construction because of specific conditions that does not mean the system itself is of low quality. Quality of the system architecture is the only aspect of quality considered for system architecture.

2.3.6 Robustness/Stability

The robustness and stability of the architecture is closely related to the quality of the system. By ensuring the system is robust, you prevent failure of systems or data loss because of improper design like improper exception handling or improper data and method structures.

² Olle et al., pg. 76

This type of failure is generally a software type because of its sudden nature. Accepting that there is no corruption of the software (programmed operations) then any failure that occurs must be caused by problems from the design stage. Robustness and stability do not include the failure of systems or data loss due to operating conditions.

2.3.7 Reliability

Reliability is the ability to maintain its quality or level of service over a time scale. Perfect reliability of the system means that there is no failure in operation of the system. The two failures that could occur are one associated with hardware due to wear and use of the system and one associated with software failure which indicates mistakes made in the design stage.

A proper method of ensuring reliability is the use of preventive maintenance. But integrating checks on reliability into the system is not enough to ensure acceptable reliability. The use of redundant systems within the architecture will help ensure acceptable reliability.³

2.3.8 Safety and Catastrophe handling

The system must be design so that no internal faults will cause partial or full system failure thus causing personal or physical injury. There are two types of catastrophes to consider: human error and natural disaster. Human error can either be from the design stages or through operation of the system. The two approaches to avoiding a catastrophe due to human error are:

- During design stages check against improper design that could at some point cause a sudden software or functional failure
- Incorporate into the design operations that will during implementation and operation protect the stability of the system from human error.

2.3.9 Risks

Any system is subjected to risks, or something that could create a hazard to stability, functionality, or safety of the system. Some of the most common physical causes are

³ Klaas and Peppen, pg. 85

mechanical stress, fatigue failure, crack propagation, fire hazards, explosion, and toxicity.⁴ When planning for the handling of risks and the possibility of their occurrence, it is important to recognize the two type: Hazop and Zonal.

Hazop – This a single system fault which is isolated and influences only one system.

Zonal analysis – This is an interactive fault which is caused usually by a Hazop failure but which spreads into another system.

Hazop failures are obviously most desired since they are contained to an isolated area or system. If a system is build redundant enough then even if a failure occurs then the failure will not spread into other systems. But if the isolated failure causes can not be contained by the immediate system then it is likely the failure will become zonal.

2.3.10 Security

Security within the system that the information should both be protected from outside tampering and the information should be used only for the purpose specified by the system itself. These two distinctions can be defined as functional security (specific purposes) and physical security (tampering).⁵

During a discussion of security, the term use of “who” is tampering with the data is quite general. The “Who” could be a personal tampering with hardware, or physically altering the software and trying to break into the functionality of the system. Or the “Who” could be an actual method or object from somewhere else in the system, which is either malfunctioning or has been changed by a person intentionally or unintentionally.

2.3.10.1 Functional

Within object oriented system the logical security is already integrated. Each object has its data protected and is can only be changed by recognized actions determined by itself. Unfortunately though, functional security does not occur automatically within the system. Functional security is the definition of the actual methods that can modify the data and the

⁴ Cox and Tait, pg. 47

definition of who can use those methods. This set of rules must be intentionally integrated into the system in both the software and the hardware. Some objects should be connected to others physically while some should remain separated and often security is the determining factor in this decision.

2.3.10.2 Physical

Even in the earliest stages of system, architecture physical security must be considered. The hardware that is installed must not be vulnerable to tampering from anything outside the system. By the same turn, because of innovations in physical security and circumvents provided by new technology, the attention to physical security should not limit the functionality of the architecture.

2.3.11 Inference Ability

The line between Artificial intelligence (AI) and the inference ability of a computer is not so easily drawn. Within the system architecture, it is essential to expect an elevated level of automated inference, but the system can never be expected to run completely autonomously.

The level of reasoning created could be of either of cooperative systems, which closely assist humans in decision making, or autonomous systems, which can function without human interaction. In determining the level of reasoning created, the risks of the system and the required reliability must be calculated.⁶

It must be remember though that any inference ability of a computer is still based on the following analysis methods that help the decision from the premises:

- Algorithmic (algebraic, differential equations)
- Programming structures (linear programming, matrices)
- Stochastic (probability theory)
- Boolean logic (algebra of sets, Bayesian)
- Involving uncertainty (possibility theory)
- Rule Based (logic programming)
- Through decomposition (Lamda-, Beta-, Epsilon reduction)

⁵ Steenis, pg. 109

⁶ Chorafas, pg. 70

- Taxonomical (classification, menus, trees, frames)⁷

These methods all have given domains determined on facts, rules, heuristic and situational patterns programmed in and collected from operation. Because automated systems always are based on mathematical procedures, the role of human decisions and event handling can never be ignored or removed from the architecture.

2.3.12 System Renovation

There are three main reasons for the need for renovation within any system:

- Implementation of an improper system design or requirements.
- A widening of Scope not foreseen at the time of implementation
- The evolution and progression of technology through its organization and implementation

It is impossible to think that as we design a system it will be the perfect architecture. A design process by nature is cyclical and never really has a definite point of completion. Figure 2.2 shows this procedure.

⁷ Chorafas, pg. 80

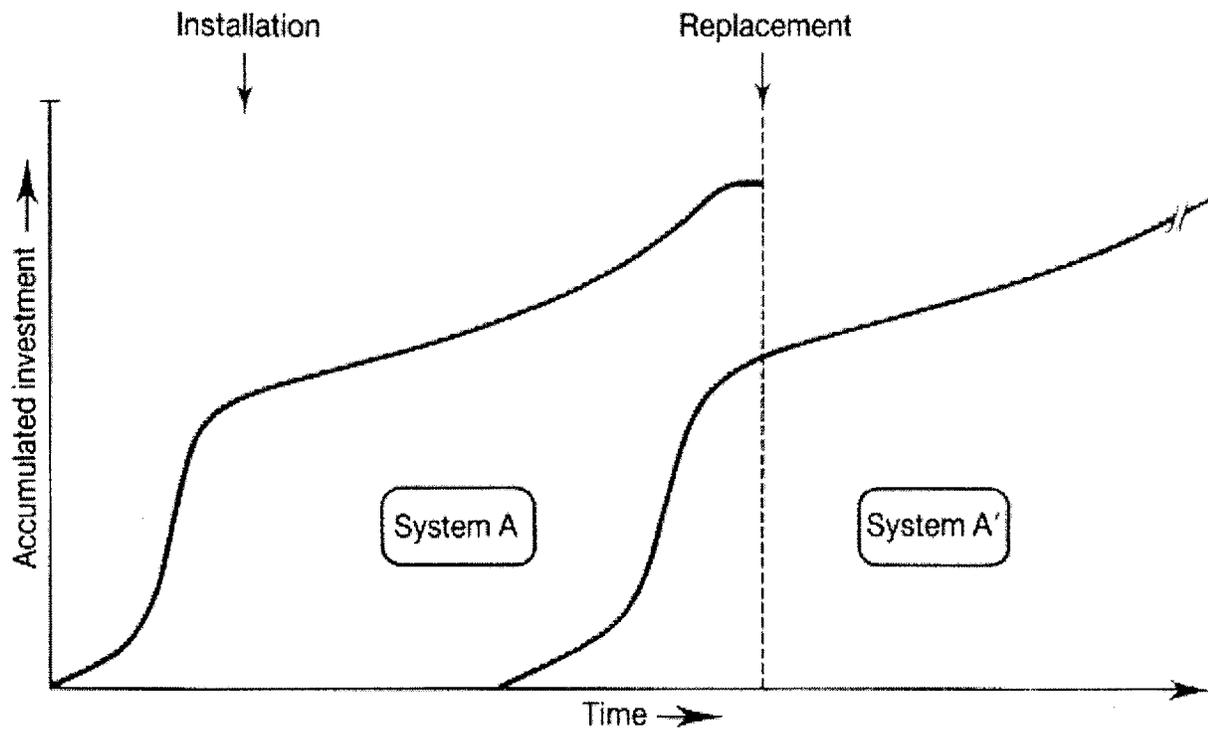


Figure 2.2 – Information system life cycle evolution

However, because a design process is cyclical does not mean that we cannot plan for that and anticipate the need for a replacement or renovation of the existing system. It is important to consider though the different things that can be done to lower the frequency and severity of renovating the system. When renovating a system there are three activities:

- **Restructuring** seeks to improve the overall logical structure of the system without any modifications to its functionality.
- **Reverse Engineering** seeks to recreate the design specifications or analysis model that the system conforms to.
- **Re-Engineering** creates a new version of the system by using new technology, often in combination with the results of reverse engineering.⁸

2.3.12.1 Restructuring

This can be avoided by proper initial design, considering properly all of the entities and their relationships. The problem with realizing this though is a continual change in the current

⁸ Olle et al., pg. 253

method of organizing data and systems. Figure 2.3 shows that there has been a progression from the Hierarchical Model to the Object-Oriented Model. The two models share the characteristic that they are concentrated more on hierarchy, but the newer object oriented architecture also allows for integrated functionality.

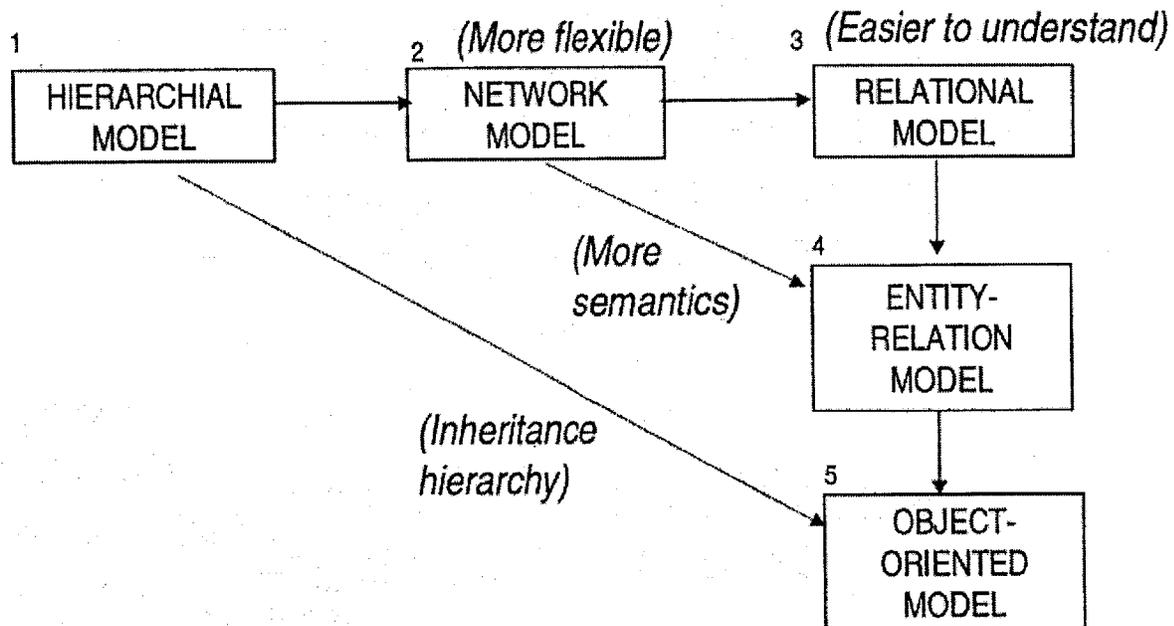


Figure 2.3- Data Model Progression

This progression of data modeling is normal but the problem is that each implementation is different and therefore as new modeling techniques are developed the previously implemented systems must be restructured to embody the current model structures. This problem is generally unavoidable because there is no way to anticipate the data and system models that will be used in the future.

2.3.12.2 Reverse Engineering

Reverse engineering is necessary when an implemented system must be analyzed so that a set of models and specifications can be derived. This must occur when the current condition of the system is not fully understood and there is a push to add new technology through re-engineering or to widen the scope of the system.

This problem of reverse engineering can be avoided by keeping proper documentation of the system and by not allowing the system to just run without considering the organization behind it. By keeping proper record of the system, new functionality can be added easily. The only other issue is an increase of scope. The system can not be design to be ready for all possible new functionality, but by keeping the structure object oriented with proper abstraction, less reverse engineering will be required.

2.3.12.3 Re-Engineering

Re-Engineering is the most common problem that a system must face because new technology is continually developed further optimizing systems and data organization. But of the types of system renovation this is the easiest to handle and the most desired. Re-engineering is the driving force in the cyclical life cycles in Figure 2.2.

2.3.13 Cost

2.3.13.1 Cost of Implemented System

Given the Current stage of the System it would not be advantageous to explore the costs of the systems researched because of the limitation of scope and the continual change in costs.

2.3.13.2 Cost Returns

There is substantial research and documentation on the returns of intelligent infrastructure. That discussion of economic and financial analysis is beyond the scope of this report.

Chapter 3 – Proposed Task Division Architecture

The task division architecture of the system has been divided into 8 separate layers, each with their own function and operation. As seen in Figure 3.1, each layer is defined both by an absolute and relative layer number. The absolute nomenclature is bounded by level 1- Data Collection and level 8- Total System Design and Requirements. For the Relative nomenclature any layer can be defined as N, so that the layer above is N+1 and the layer below is N-1. Overlaying all 8 layers is the systems controls. The system controls must be separate because it impacts all layers equally. Within System Controls are the functions: Optimization, Failure Control, and Reliability.

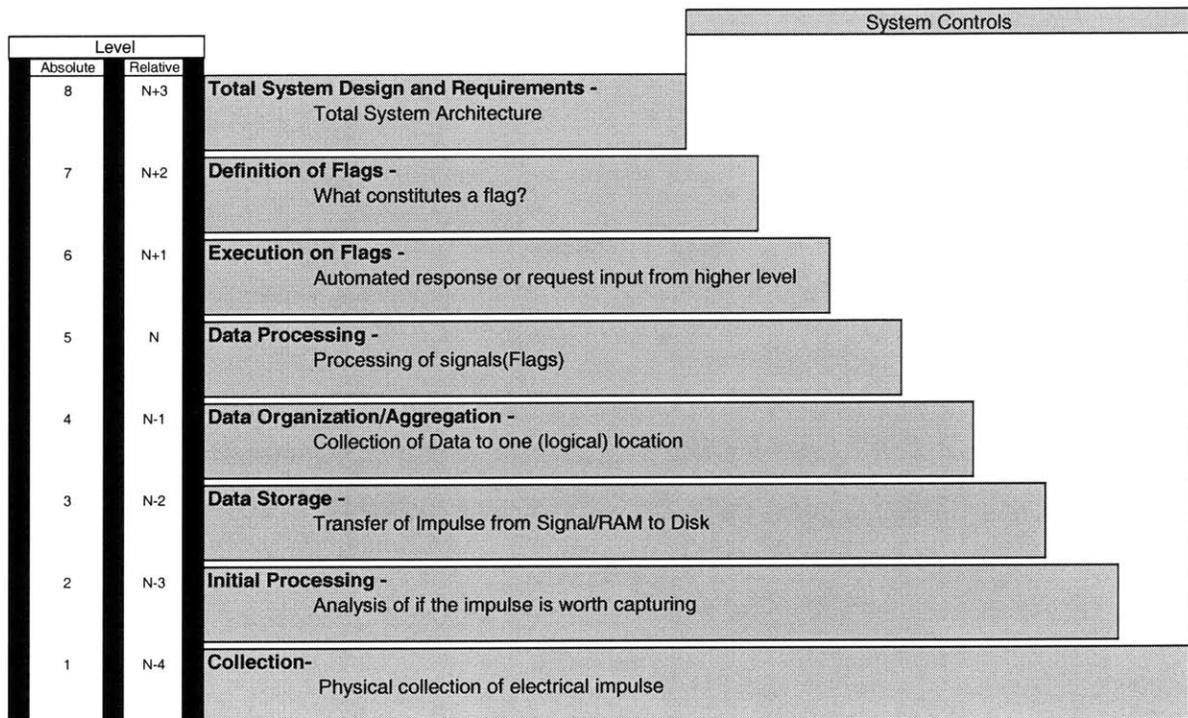


Figure 3.1 – Total System Architecture

This chapter will describe the specific responsibilities of each layer. During the discussion, abstraction is still observed, because this architecture can be applied to a system of any scope and scale in Intelligent Integrated Infrastructure.

The 8 layers that have been defined for the architecture are completely separated from physical implementation. There are some situations when one physical object could execute operations from all 8 layers of the architecture, or there could be situations when every single layer is handled by separate equipment. Also because of the rapid development of technology it is important to separate the layers of the system from the physical implementation.

All information will be passed between layers as a Flag. To maintain the object-oriented architecture of the system each layer can be considered an object. The information flow between occurs then not as open data but as an encapsulated flag. These flags are sent when a layer determines using its inherent criteria that it must send a flag. The flag can be a notification or function call to the N, N+1, or the N-1 Layer.

3.1 COLLECTION

The lowest level in the system, data collection is only the capture of electrical impulses. This could be a pressure actuator, a light sensor, an electronic fuse, or any piece of equipment design to collect information. There will certainly be new methods of data collection so they too will be considered within this list. This information size will be very small on the order of magnitude of bits.

3.2 INITIAL PROCESSING

Once the capture of data has occurred it must be initially processed to determine whether it is worth keeping. Since there are different uses of the data, this layer is required. Depending on the criteria set by the following four types of collection, the data will be collected or it will be disregarded.

3.2.1 Baseline

The first and most important type of collection required is the baseline data. There is no way to create initial conditions defining emergencies, lower reliability, and changed conditions if a baseline of information is not collected. The initial baseline calculation is

necessary but even more important is the ability to monitor if there has been a slip in the baseline information over time or use. The other three types of processing can not be properly interpreted without this information.

3.2.2 Reliability

The second case of initial processing could be gradual fault or reliability assurance. In this case a continual state of readiness must be maintained but the time period or monitoring and calculating is much longer, ranging from seconds to the life of the object monitored.

3.2.3 Emergency

The third use of the data could be an emergency case, when fault or failure is detected. This case would require immediate action with a time period of microseconds to seconds and must have a continual state of readiness.

3.2.4 Specific Monitoring

The final case for data collection could be specific monitoring criteria passed from the system administration that is intended to collect data for a specific purpose facilitating system stability and control. When running specific tests, special information is required and can not be automated into the system.

3.3 DATA STORAGE

Once it is determined that the data should be collected and processed because it is useful it is necessary to store it. This introduces a large amount of data storage capacity into the system and could be seen as unnecessary, but because of system robustness and fault control it is required. Since it has already been determined that the data is important and worthy of collection then it must be protected. If the data remains in temporary memory and a fault occurs in the system then the data will be lost. It is therefore important to convert the data from temporary memory to permanent storage, or disk. This is the only real way to always ensure no loss of important data.

Another way of handling the loss of data is through data checking and algorithms that can calculate the lost data but this should not be part of the system architecture. It could be used in a specific case of the implementation under certain conditions, but because it is assuming data is lost, it should not be in the design. The design should be built to assume that no data is lost.

3.3.1 Location

The most important issue of data collection to consider is the location of the stored data and the way in which is stored. These two topics though are dependent on technology. Where the data is actually stored is the critical issue though. It is important to store the data away from the point of capture so that if a failure occurs then the system can recover and no data is lost. However, by enforcing remote data storage, the system is much harder to implement and the issues of technology are essential in remediating the problems.

It could also be a possibility to not locate the database completely remotely from the object, but in a scattered way, so that a distributed database is created. This could again introduce irrecoverable data loss if a failure occurs, but still should be researched for system architecture.

3.3.2 Duration of Storage

The length of time the data would need to be stored before erasing is dependent on the size, type, use and nature of the implemented system. If the system was a critical system, such as a hydroelectric electricity generation plant, then it would be essential to allow for more robustness in data storage than in a single unit personal residence. It would also be important to keep the data for longer periods in the power plant than the residence to allow for system monitoring, and optimization.

3.4 DATA ORGANIZATION

Once the data is collected and stored, it can be organized for processing. In this system design, the organization of the data has been separated from the processing of the data

because although they are often performed by the same object they are two distinct operations. The process of collecting the data across the network is different then making a decision on the data at one hub.

Depending on who is actually using the data, the aggregation of the data can be vastly different. If the operation is overall quality assurance, the data must come from all of the Level 1 layers. If the operation though is a specific probe into the performance of one small component of the system then the data must only from one of the Level 1 layers.

Intelligent data connections or middleware push the responsibility of the aggregating the data lower in the system organization but is dependent again on the current technology. A decentralized processing ability would be required within the network in order for many different components to collect information from different locations.

3.5 DATA PROCESSING

With the data aggregated, the information can be processed. The processing of the data can not occur before the actual operation is passed from the N+1 layer, Execution on Flags. Once the processing has occurred then the result from the processing can be handed back to the execution level.

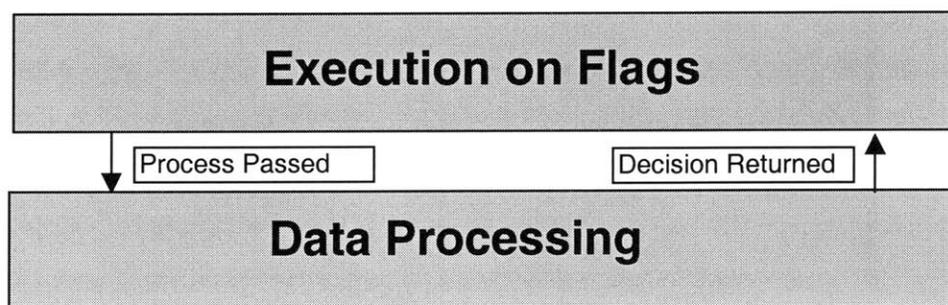


Figure 3.2 – Data Processing

3.6 EXECUTION ON FLAGS

With the information aggregated and processed, the execution level must act on results returned from the Data Processing. The type of actions required are bounded by the definition of flags at the N+1 layer.

- **Do nothing/Wait**
If the output passed from the N-1 does not substantiate any action then the action can be to do nothing. If nothing is done then the output is either discarded or passed back to the data storage level. The output could also substantiate no immediate action but continual monitoring. An example of this could be if a condition in a component reaches levels that require alert but not action, or non-failure conditions.
- **Invoke automated response**
If the output calls for an action then this layer must be able to act on that automatically. It can only act automatically within the predetermined actions defined by the N+1 layer, because the system has finite boundaries and can not infer decisions past its limits.
- **Request higher level intervention**
If the action required is outside of the bounds of the predetermined actions then the execution level turns to the N+1 layer and waits for an intervention. This is a critical point in the architecture because it inherently builds delay into the system. It is required though because not every system can be completely automated. There are conditions within any system, which require human judgment and additional information external to the limits of the system.

3.7 DEFINITION OF FLAGS

A flag is any piece of information or request for action sent between layers and objects. The reason they are called flags is because they are only required when a specific condition has

been met and a notification of information or for action must be passed. As mentioned it would be expected that there are two types of flags: data and process. But in our design there is actually only one type: data and process combined. This is because of the object-oriented architecture and the encapsulation of both data and methods together.

3.8 TOTAL SYSTEM DESIGN AND REQUIREMENTS

This layer plays the largest role during the design and development of the implemented system. Each system must have a set of rules in which to operate. These are defined primarily at the initial stages but can be restructured and redesigned at any time during operation through system renovation.

3.9 SYSTEM CONTROLS

Overlaying the system are functions, which must be integrated at all layers to ensure a stable, responsive and active system.

3.9.1 System Failure Control

System failure control can be monitored at two levels. The first type is where components at the N level monitor their peers and detect failure. If they detect a failure then they pass a flag to the N+1 layer. This type of failure control is effective only for localized failures in a system and is not able to handle macro failures, which disable a complete level

The second level is at the N+1 level. This is primarily for identifying macro failures. A macro failure is characterized by a level's functionality being comprised. The only way that this failure can be detected is from a level, which has not been disabled by the failure.

3.9.2 System Optimization

Although a system will continue to run without inherent optimization, it would be finite and static by nature, having a calculable service life always using the same technology and processing. Our system architecture requires that it not be static or finite, but completely

responsive to the current technology, the current operating conditions, and the current use. Optimization can not occur across all aspects of the system. When considering cost, flow, safety, stability, speed of reaction, and many other aspects it is clear that one or two aspects must be selected for optimization because ultimately the optimization of one impacts all others.

The first type of system optimization is automated. It can be either completely automated or require specific external discussion support from a system administrator. An example of a completely automated optimization operation could be water flow and supply within a grid cities water supply system. There are rules and boundaries defined for the system, and depending on the system conditions it can automatically optimize supply and flow throughout the network. An example of a semiautomated optimization could be in a traffic corridor where the number of travel lanes in each direction is controlled by shifting of the margin during different time and capacity conditions. The system could determine that a change in margin should take place, but the execution of that request must be authorized by an administrator to the system.

The second type of system optimization is from system renovation. As discussed earlier this can come through restructuring, reverse engineering, or re-engineering. Although these three types of renovation are different in their level of impact on the system, they are always performed to optimize the operations of one or many aspects of the system. This level of system operation can only be performed from the system administrations level.

3.9.3 Reliability

When considering the reliability of the system you accept that the system your are architecting will inherently have faults and failures. These faults and failures must be from those caused by the implementation of the system and not from an improper system architecture. In order to maintain the quality over time or the reliability of the system the following steps can be integrated into the system.

- Inspection

Candidate Architecture for Intelligent Integrated Infrastructure

A continual look at the components of the system and their ability to perform their assigned task or function.

- Surveillance

An occasional in depth analysis of the system and the inspection layer.

- Audit

A full probe into the components of the system, their ability to function, and an inspection of the actual task performed

Chapter 4 – Semicentralized Object Network

Now that the tasks have been defined, the organization of the components that actually are the system will be discussed. In this section though, it is again difficult to discuss the separation of the tasks without defining the objects themselves and introducing the limits of the current technology into the discussion. The solution is to then discuss the principles and methods that must be considered in any implementation of the architecture.

4.1 ARCHITECTURE CAN BE APPLIED AT ANY LEVEL

The power of this architecture is that it can be applied for any range of size and scope in the implementation. The reference of N level is completely abstracted in this section and it must be understood that any reference to objects at N level does not correspond to the task layer definitions of Chapter 3.

4.2 PROCESSING AT POINT OF CAPTURE

4.2.1 Discussion of Bottleneck calculations

A purely hierarchical structure to the flag passing and information flow is shown in Figure 4.1. This type of data flow forces centralized computations about the N layer to take place at the N+1 layer. This causes considerable bottlenecking at the N+1 layer. Consider that there are on average 2 orders of magnitude greater number of N objects than N+1 objects. It is a poor judgement to believe that the N+1 layer will always be able to handle the information flow from the N layer even if special considerations are made by increasing computational ability of an N+1 layer component.

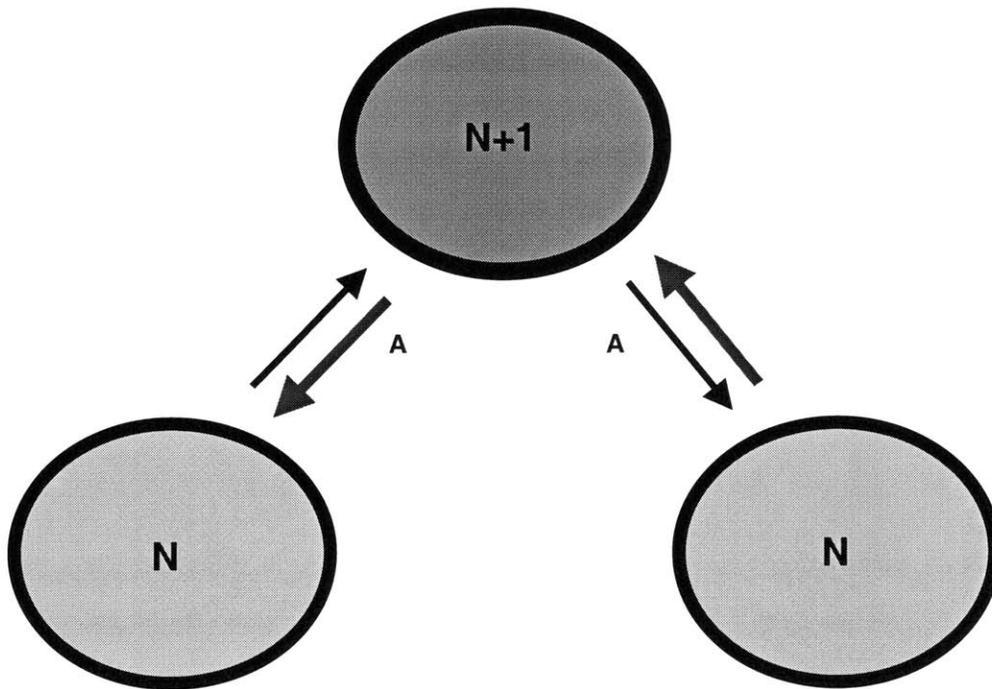


Figure 4.1 – Pure Centralized Computing

4.2.2 Lower Bandwidth required

Depending on the preferred means of connecting the objects and layers, a N level computation scheme can substantially optimize bandwidth usage. This is the case if scheme 1 as discussed in Section 4.5. If the data processing takes place within the N level, then there will less information passed to the N+1 layer. A large amount of information will be passed within the N layer but because there are many objects, it can be isolated to that private network whose only role is to handle that specific traffic. The N+1 layer is then able to communicate with multiple different nets of the N layer without restricted bandwidth. This is because the N to N+1 connection is only used for information from a decision, not a pure impulse or retrieval.

4.2.3 Flag processing at N level

Using a decentralized computing approach to the components, all decision making is made at the N level commutatively by all members at the N level. Flags are passed among the members and a decision can be made by any of the members. Once the decision is made it is passed to the N+1 layer.

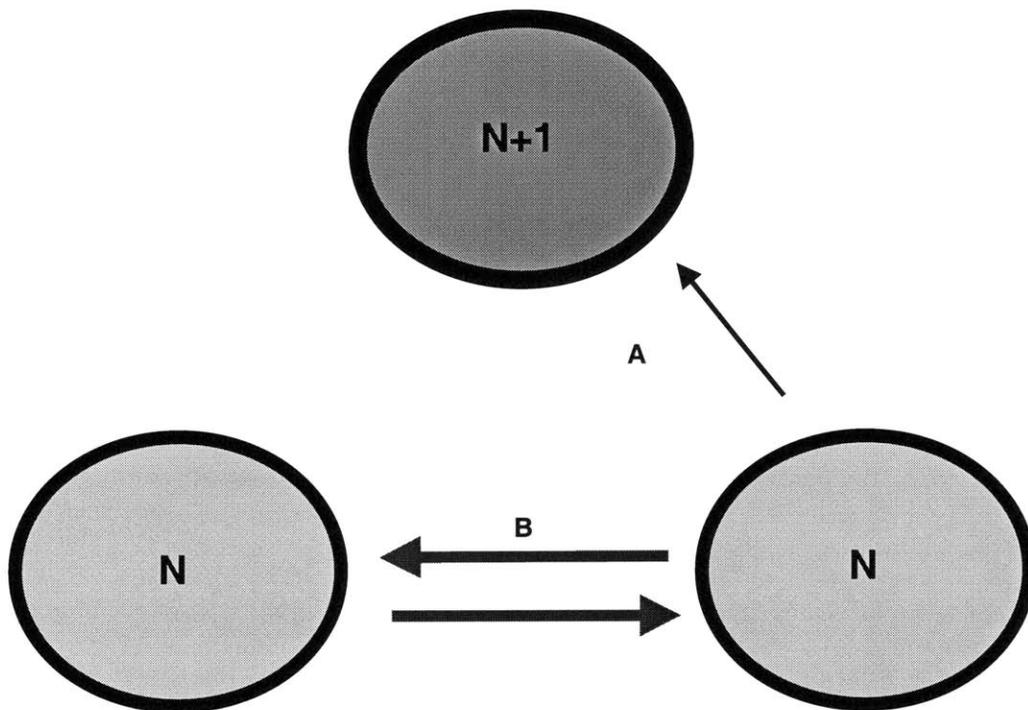


Figure 4.2 – Decentralized Computing with N+1 Flag passing

This approach avoids the occurrence of bottlenecking of flags to the N+1 layer, but has its problems as well. How do independent and equal objects come to a consensual decision? They must pass information from one to the other until a critical amount of data has been received by any one member, and which point it can pass a decision upwards.

Another problem is that if the N layer has a failure, then there will be no way for the N+1 to receive the flag notifying it of the failure. To clarify, if the N layer goes into failure it must decide that it is failed before it sends a flag to the N+1 layer informing it thereof. This is

counterintuitive and checks against that must then be installed. The N+1 layer can ask for status of the N layer continually and if it gets no response, it can take course of action. But this creates an unnecessary overhead, and it does not provide the N+1 with almost any information that would be required to handle the failure.

4.2.4 Both Server and Client can be fat

A solution to this problem is one that introduces redundancy into the system, but one which increases robustness and stability. If the N and the N+1 layer are both able to execute, perhaps only on certain flags, then the system would be able to handle failure more appropriately. N level processing could occur during normal conditions, but if the N layer loses its ability to execute properly, it is acceptable to have the N+1 layer handle that task.

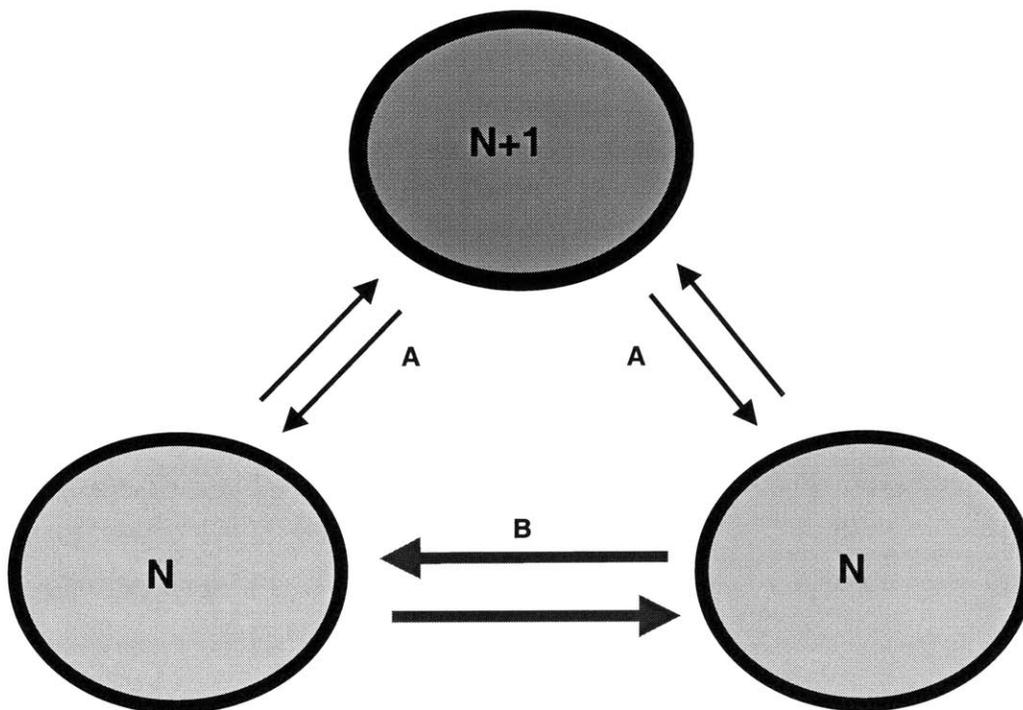


Figure 4.3 – Modified Dual Computation Ability

4.2.5 Sharing tasks through Load Balancing across N tier

By allowing computation to occur at the N level, load balancing can be obtained by utilizing non active, or underutilized components. This avoids the bottleneck of going to the N+1 level and also enables more efficient processing at the N level.

4.3 EACH COMPONENT IS SELF MANAGING

Each component should be able to exist at no cost to other components. If failure occurs in a component or object it should fail safe and neither cause other components to fail, or cause the system to fail.

4.4 DATA HIDING

A strength of the Decentralized computing within and Object Oriented Organization is the ability to hide data. All information that is passed within the N layer and that either is passed up to the N+1 layer or down to the N-1 layer is sent wrapped. This wrapping is the notion of a flag or an invocation or notification of a method, which modifies or passes data, or alerts about a decision made at the N layer.

4.5 NETWORK CONNECTION

There are different ways to physically connect the objects within the system. This physical connection is of course based on the logical connection between the objects. There are two schemes that can be used.

Scheme 1

The objects of the N layer are interconnected with each other and communicate with the N+1 layer on a different network. This scheme will allow the true separation of layers in their physical and informational connectivity.

Scheme 2

All objects at the N and N+1 layer communicate across one open network. At strategic points in the network sub-nets will be declared and partitions will be determined by their global role.

Chapter 5 – Conclusion

5.1 TRANSPARENCY OF STRUCTURAL AND INFORMATION OBJECTS

As discussed in Section 1.2 because of the power of abstraction and inheritance associated with Object Oriented Organization, the architecture can be used for both purely structural or informational, or any combination between.

5.2 REMOVAL OF CURRENT TECHNOLOGY

The removal of current technology is essential in order to properly develop a system architecture, which adheres to principle of its design and organization, not the way in which it could be implemented. In order to outline the architecture for this paper and not create a definite end to the utility of the paper, any mention of current technology was avoided. By discussing organizational issues, which are not affected by technology, a feasible architecture could be outlined.

5.3 USE OF TECHNOLOGY IN THE FUTURE

Many technologies of the future will not change the actual architecture of the III, but it could have drastic effects in the way it is implemented.

5.3.1 Wireless

Wireless communications would remove the need for actual physical connections and could allow a more efficient flow of information based more on the content and purpose itself. By removing the information flows from the structure, if a physical failure occurs then a zonal failure is less likely to occur and the informational channels can still be available. The system is thus that more robust and stable.

5.3.2 Faster Computing Speed

By having much higher computing speeds and methods of processing information it will be possible to move data processing lower and lower within the physical objects hierarchy. If the actual mechanism which collects the data can do the processing as well, benefits from distributed computing power can be realized.

5.3.3 Different Storage of Data Procedures

The largest issue when discussing data handling is the storage of it in an organized and usable fashion. If the use of standardized distributed databases can be fully implemented in III, then the problems of centralized data storage and bottlenecks can be avoided.

The process of writing information to disk is the only mechanical process involved with data handling. There is no way to permanently save data safely except to physically write it to a specific location. No matter how quickly the data write/read/copy can occur it is magnitudes slower than electronic data handling and even is inherently different. Because of this difference a change in the procedures could impact the method of implementation of the system and change to organizational structure of the data storage level.

5.4 OBTAINABLE BENEFITS

Although many see I³ as an unobtainable, the realization would produce specific and obtainable benefits that make its development substantiated.

5.4.1 Controlled Modeling

Modeling of a system is only as reliable as the data on which it is based. With III, real data that is not from scaled versions but the real city can be used in real time conditions to analyze the condition of the systems and create a real representation.

5.4.2 Total systems optimization and safety

With the controlled modeling capabilities, different tests on optimizations and safety can be routinely and easily performed. Utilities systems and their interaction with not only each other, but the infrastructure in which they are embedded can be optimized and tested for worst case scenario catastrophes to analysis system and personal risks and safety.

5.5 DIRECT FUTURE OF RESEARCH AND DEVELOPMENT

There are two main areas of further development required. Within each area there are examples given which are intended to outline the type of research that is needed and not define the actual point to be researched.

- What needs to be calculated (based on known premises)
Given the framework of the system architecture as outlined in this paper there is quantifiable information that must be developed. Some examples are: The actual size of the data collected, the computational speed required to meet the required processing time, the differences in centralized and distributed computing power.
- What needs to be researched (new premise development)
It is expected that new technology is developed everyday. The new technology does not need to be directly researched and developed to push the project forward. What is needed is revolutions in technology, which actually change not only the technology but the organization behind it. An example as mentioned in Section 5.3.3 is implementable wireless technology. Removing the flow of information from the physical flow of the structure is a fundamental difference which will certainly impact the implementation and could theoretically impact the system architecture

Chapter 6 References

1. Brill, Alan E. Building Controls into Structured Systems. Yourdon Press, New York, NY; 1983.
2. Chorafas, Dimitris N. System Architecture and System Design. McGraw-Hill Book Company. New York, New York; 1989.
3. Cox, S. J. and Tait N. R. S. Reliability, Safety and Risk Management; An integrated approach. Butterworth-Heinemann Ltd. Jordan Hill, Oxford; 1991.
4. Goode, Harry H., System Engineering; an introduction to the design of large scale systems. McGraw-Hill Book Company. New York, New York; 1957.
5. Jamshidi, Mohammad. Large Scale Systems, Modeling and Control. North Holland. New York, NY; 1983.
6. Klaassen, Klaas B. and van Peppen, Jack C. L. System Reliability, Concepts and Applications. Edward Arnold of Hodder & Stoughton. New York, NY, 1989.
7. Lancaster, John. Engineering Catastrophes, Causes and effects of major accidents. Abington Publishing. Cambridge, England; 1996.
8. Madnick, Stuart. Global Information Systems: Communications and connectivity among information systems, MIT. Class lecture notes, 2000.
9. Olle, William T. et al. Information Systems Methodologies, a Framework for Understanding. Addison-Wesley Company, Wokingham, England; 1991.
10. Rabin, Jack and Jackowski, Edward M. Handbook of Information Management. Marcel Dekker, Inc. New York, NY; 1988.
11. Taylor, David A. Object-Oriented Information Systems. John Wiley & Sons, Inc, New York, NY; 1992.

12. Thierauf, Robert J. Effective Information Centers, Guidelines for MIS and IC Managers. Quorum Books. Westport, Connecticut; 1988.
13. Tricker, R. I. Effective Information Management, Developing Information Systems Strategies. van Nostrand Reinhold, New York, NY; 1982.
14. van Steenis, Hein. How to Plan, Develop & Use Information Sytems. Dorset House, New York, NY; 1990.
15. Ward, Paul T. Systems Development with Pain. Yourdon Press, New York, NY; 1984.
16. Wetherbe, James C. Executive's guide to Computer-Based Information Systems. Prentice-Hall, Inc. Englewood Cliffs, NJ; 1983.
17. Wood, Jane and Silder, Denise. Joint Application Development. John Wiley & Sons, Inc, New York, NY; 1995.