Chapter 2
Multidisciplinary Systems Engineering

2.1 Multidisciplinary Engineering for Modern Systems

Multidisciplinary Engineering will be a requirement for designers of modern systems in the near future. Much of the undergraduate Systems Engineering education is not effective in preparing engineers for the multidisciplinary approaches that will be required for system-of-systems integration and system optimization in the future. Commercial companies are becoming increasingly aware of the need for systems engineers, particularly systems engineers that understand a host of disciplines required to design, implement, and manage complex Information Technology systems. Engineering students must learn and adopt a breadth of disciplines to be ready for the systems engineering challenges of the future. Not only will Systems Engineering skills be required, but a host of other skills like Business Intelligence, Human Factors, Technology Integration, along with a working knowledge of Science, Technology, Engineering, and Math (STEM) skills. Systems Engineers can no longer become a stovepipe of systems engineering knowledge, and assume someone else will handle making sure their designs conform to the needs of other disciplines. Figure 2.1 below illustrates the confluence of these skills into the field of Multidisciplinary Systems Engineering [25].

Question to think about:
What actually happens when Systems Engineering does not provide a robust system solution?

The study of Multidisciplinary Systems Engineering begins the journey of understanding the difference between studying engineering and becoming a practicing engineer. The purpose of this introductory Multidisciplinary Systems Engineering textbook is to help the engineering student begin this journey.

J.A. Crowder et al., *Multidisciplinary Systems Engineering*,
DOI 10.1007/978-3-319-22398-8_2

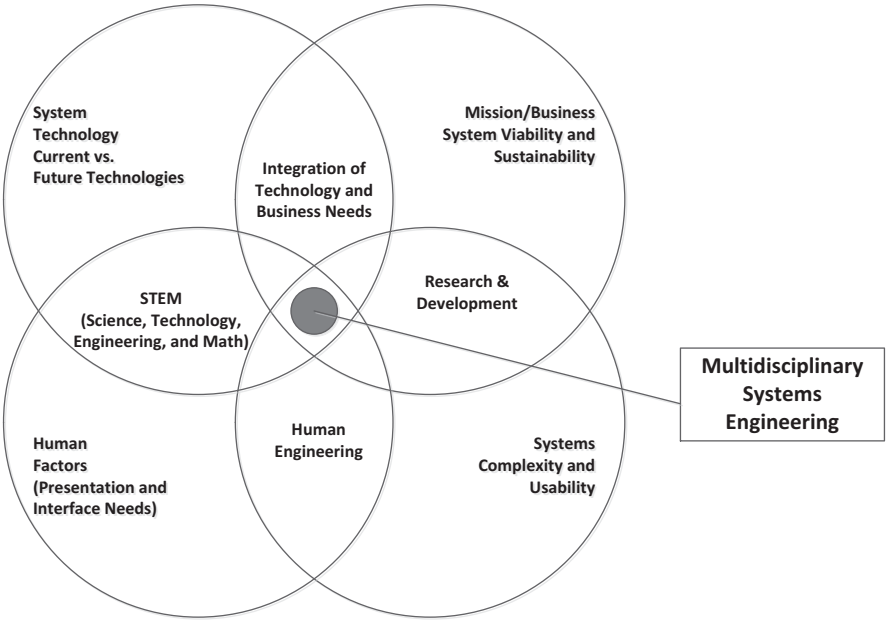

Fig. 2.1 Multidisciplinary systems engineering as a convergence of multiple skills

2.1.1 Multidisciplinary Approaches to Knowledge

Research shows that new knowledge is accomplished via natural human means and observations. For example: mental thought about observation or insights, the scientific inquiry process, physical human sensing of their surroundings, learning from results of actions, and experiences, while context is information, which adds to the characterization of knowledge, using terms and sentences, and gives it additional detail to focus the meaning. This knowledge is generally acquired via established forms of scientific research requiring the focused development of an established set of domain specific criteria, uniform approaches, detailed designs, and domain appropriate analysis, as inputs into what can be the potential solutions.

Multidisciplinary Engineering research literature clearly argues for development of strategies that can transcend and bridge the knowledge of any one given engineering discipline to another and engineers to enhance research collaboration between disciplines. Increasingly, cross and multi-domain research is more commonplace, made possible through a wide-range of available web based search engines accessible from ubiquitous numbers of devices, information content, and digital toolkits all part of the emerging Internet of Things (IoT) [26]. This environment creates the condition where large amounts of inadvertent cross-domain information content are exposed to wider audiences. Researchers expecting specific results to specifically focused queries usually end up acquiring somewhat ambiguous additional results and hence additional responses much broader in scope than was originally planned by the instigating query. Consequently,

a resulting lengthy iterative learning process ensues along with numerous attempts at query refinement, until the truly sought after knowledge happens to be discovered. As a multi-disciplinary example Biomedical and Health care systems [27] generally access vast stores of medical research and clinical information content in attempts to gather detailed information and research about a particular topic or disease or condition.

Often digital and non-digital book searches can yield thousands of possible sources, most of which are not relevant to the context that a medical professional or non-medically inclined patient is seeking. Queries across many types of symptoms can lead to potentially hours of work and contemplation before an answer can be reached. This recursive-like refinement of knowledge and context occurs as user cognitive system interaction, over a period in time, where the granularity of information content results are analyzed, followed by the formation of relationships and related dependencies. These dependencies exist, even for a medical professional cross a number of domain boundaries. Even a doctor who has learned a great deal at medical school does not commit all medical content to memory and has to reach continually across discipline boundaries to achieve appropriate and many times life changing information content. Hence, underlying the ability to perform true data fusion within a domain is a significant challenge to create actionable knowledge from a continually expanding environment of vast, exponentially growing structured and unstructured sources of complex cross-domain data.

Therefore, Systems Engineering, as well as all engineering domain(s) requires new multidisciplinary methodologies to deal with the challenges of cross-domain content to more efficiently and more effectively discover and assimilate the most appropriate content for creating designs and architecture for systems that are viable enough to continue to advance research and compete in future complex environments in need of minimizing ambiguity and maximizing clarity.

2.1.2 Multidisciplinary Engineering as on Overriding Guide for the Design

As we discussed in Chap. 1, current and future system designs require System Engineering to embrace and utilize a multidisciplinary approach. The balance between depth of Systems Engineering principles and practices and a breadth of knowledge of the multiple disciplines (refer to Fig. 1.5) required to adequately design modern, complex systems must be evaluated throughout the system development lifecycle. Figure 2.2 illustrates this balance.

As expected, the overall goal of Multidisciplinary Systems Engineering is the successful design, implementation, delivery, and operation of a given engineering system. Project Management texts will frame success in terms of timescales, budget profiles, and resource utilization. And while not unimportant, these criteria do not reflect the delivery of a successful, viable, usable system [28]. Here, we couch success in terms of different entities: the Chief Multidimensional Systems Engineer, the Design and Development Team, the Multidimensional Systems Engineering Process, the overall design, and the end product (system). Figure 2.3 below illustrates the interactions between each of these entities.

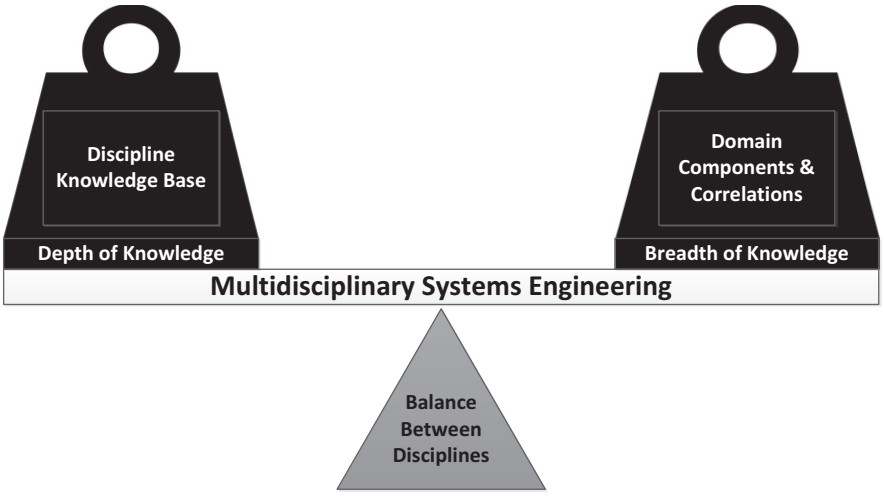

Fig. 2.2 Balance between systems engineering knowledge depth and a multidisciplinary knowledge breadth

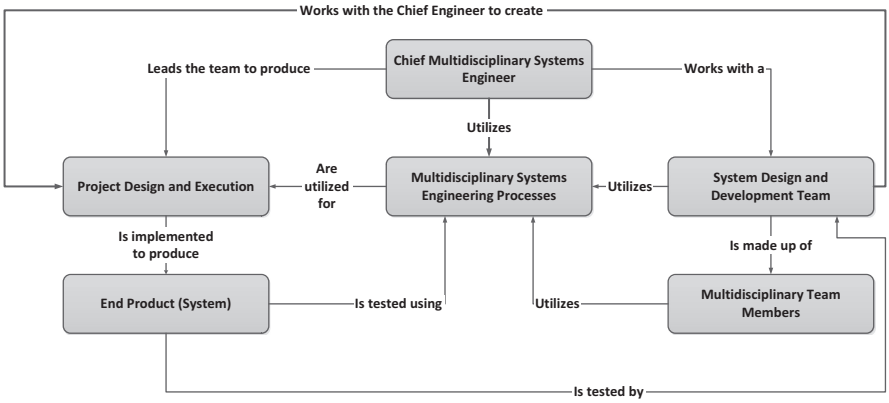

Fig. 2.3 The multidisciplinary success process dynamics

For such a multidisciplinary design and implementation process, success depends on the behavior of the development team and the overall ability of the team members to work together in the creation of the various system architectures (e.g., system, information, data, software, etc.), as well as the inherent way in which the design team employs a recursive mentality in their design process. For the Multidisciplinary Systems Engineer, project success is seen in terms of:

- An overall design process, not just an end product.
- The interrelationship between each of the steps in the design and implementation process.

- The balance between each of the engineering disciplines, making sure that all aspects of the design are equally considered throughout the system lifecycle (e.g., Human Engineering, Security Engineering, Systems Engineering, etc.).
- The overall level of team work and engineering skills, including communication.

These are inherently important to measuring the overall success of a project, since the measure of these factors is useful in determining the effectiveness of the team and whether this team makeup should be kept for other projects. The result is that success in Multidisciplinary Systems Engineering does not just depend on good designs and architectures, buy also on the socialization of team members across each project/design and the interactions between teams and team members to communicate lessons learned.

2.1.3 Multidisciplinary System of Systems Engineering

System of Systems Engineering began within DoD-related system development; however commercial companies are embracing the concept as they try to bridge legacy systems across companies and across geographically disperse stand-alone systems [29]. This may be through company acquisition, or across intracompany divisions. This drives the high-level definition of System of Systems characteristics: operational independence of System of Systems elements, evolutionary behavior as the systems are interconnected, and emergent behavior as coherent information flows drive the System of Systems architecture to become more than the sum of the system of systems elements [30]. The operational independence characteristic describes the boundary behavior of the connected System of Systems elements, while the other characteristics define the overall behavior, once the system is connected and operational. The last characteristic, emergent behavior develops due to complex interconnectivity of System of Systems elements that do not manifest themselves when each System of Systems element is operated in isolation. Understanding the dynamics of emergent behavior presents one of the primary challenges that Multidisciplinary Systems Engineering must face when designing and architecting System of Systems. The Multidisciplinary Systems Engineer must learn to understand the mechanics of evolutionary and emergent behavior, develop the metrics necessary to detect them, and establish methodologies for managing such behaviors. To that end, there are many types of System of Systems that the Multidisciplinary Systems Engineer may be involved in designing and implementing: the Directed System of Systems, the Collaborative System of Systems, the Acknowledged System of Systems, and the Virtual System of Systems.

2.1.3.1 The Directed System of Systems

Directed System of Systems are implemented for very specific purposes. It is usually build for long-term operations (e.g., satellite ground control system). The Directed System of Systems is intended to fulfill its specific purpose, but is expected

to be expandable to encompass other purposes as the mission/business needs arise and are dictated in the form of changing requirements. Here, the Multidisciplinary Systems Engineer must design each element and bring the elements together to form a complete System of Systems. The elements of the System of Systems cooperate and are dedicated to the specific purposes specified by the overall customer.

2.1.3.2 The Collaborative System of Systems

Within the Collaborative System of Systems, system elements work together (or collaborate) to fulfill specific purposes that may be different from any of the individual elements of the System of Systems. Each element of the System of Systems decides separately how they will provide or deny service, allowing each element independent control over enforcing and maintaining standards. An example of the Cooperative System of Systems would the World Wide Web. In this System of Systems, the Multidimensional Systems Engineer from each element must collaborate and cooperate to create the System of Systems behavior required to fulfill its goals and objectives. There may be a formal agreement or an unwritten agreement such that it benefits all parties to keep the System of Systems goals and objectives fulfilled long-term.

2.1.3.3 The Acknowledged System of Systems

For the Acknowledged System of Systems, there are high-level objectives that must be fulfilled through the collective use of the Elements of the System of Systems and these must be agreed to and maintained by the collective System of Systems. In this model, however, each element remains independent, with separate ownership of each element, and each element retains its own objectives and purposes, operations, maintenance, etc. Any changes required to repurpose the entire Collaborative System of Systems must be based on collaboration between the over-arching System of Systems and each of the elements. Here, the Multidisciplinary Systems Engineers for each element carry a dual role, where each element has the responsibility to maintain its own goals and objectives that it must fulfill, while at the same time ensuring that the larger System of Systems also meets its goals and objectives. Here the Multidisciplinary Systems Engineer must manage a global system optimization, while continuing to manage their element-level system optimization.

2.1.3.4 The Virtual System of Systems

In the Virtual model of System of Systems, each element maintains complete control and there is no overall central management structure. The System of Systems required emergent behavior is dependent on relative mechanisms that the Multidisciplinary Systems Engineers together maintain, even if behind the scenes.

The emergent behavior may or may not be what is required to fulfill the requirement, needs, and objectives of the System of Systems customer. In many cases, they may wait to observe the emergent behavior that ensues, and then modify their objectives to match the observed emergent trends in the System of Systems. The Multidisciplinary Systems Engineer is much more concerned with their own element-level system, while behind the scenes trying to manipulate their elements to drive the overall emergent behavior of the larger System of Systems. This model is a very difficult System of Systems to facilitate.

2.1.4 Establishing an Effective Frame of Reference: The Heart of Multidisciplinary Engineering

Understanding the overall operational system environment is essential to understanding how to decompose the system requirements, needs, and goals into a viable System of Systems architecture. Based on the environment, the requirements, the performance needs, the external interfaces, the concepts of operations, constraints, and other applicable design considerations, a basic set of reference frames should be developed that guide the architecture design, system development, transition and operations of the System of Systems [2].

2.1.4.1 Analysis Frame of Reference

2.1.4.1.1 Logical Analysis

During Logical Analysis, the required System of Systems functionality and overall system behavior is analyzed. This analysis defines, selects, and designs the logical architecture required to create a Logical Framework from which design can be assessed for compliance with all system requirements for all operational scenarios required and for long-term System of Systems operations. The Logical Analysis will include Trade-off analyses between requirements and proposed solutions. This Logical Analysis will include creation of Functional Models (functional decomposition) of the system requirements, Semantic Models that describe the relationships between information and data required within the system (e.g., Data Flow Diagrams, Entity-Relation Diagrams, etc.), and Dynamic Models that describe state transition diagrams and activity required to describe and meet the overall system requirements. The Logical Architecture that flows from this analysis is a set of technical concepts and principles that support the logical operations of the systems [31].

Question to ponder:
What is the difference between the logical architecture lay out of the systems vs. the physical instantiation of the architecture?

Improper handling of the Logical Analysis and subsequent Logical Architecture is the reason for many system design and implementation failures. Some of the common problems with improper Logical Analysis are:

- **Improper handling of Mission/Business needs**: the Logical Analysis should take into account the customers' operational concepts. If these are not well understood or folded into the overall Logical Architecture, the resulting developed system may not be useable for the purposes for which it was originally intended.
- **Improper handling of system requirements**: there are times Multidisciplinary Systems Engineers will pay attention to some requirements more than others because they are well versed in designs using some requirements and not well-versed in others. The consequences are that the ensuing Logical Architecture defines a system the Multidisciplinary Systems Engineer is comfortable with, not one that meets the requirements and needs of the customer and system operators.
- **Improper Functional Decomposition**: if the Multidisciplinary Systems Engineer creates of too many functions within a given level of the architecture, or too many levels of architecture, the number of internal interfaces and data/information flows increases, overly complicating the overall design, implementation, operations, and maintenance of the system.
- **Improper consideration of inputs/outputs**: Many systems architects consider only the functionality and subsequent actions needed to support the functionality while not considering the system inputs and outputs as part of the overall, integrated design, deciding, instead, to deal with the system inputs/outputs as an afterthought later. The Multidisciplinary Systems Engineer must realize that inputs and outputs are an integral part of the overall design and must be included at every step in the architectural design process.
- **Improper emphasis on static functional decomposition**: static decomposition of the system is important, but is a small part of the overall architectural design and should only be performed after the functional scenarios have been created and folded into the Logical Architecture. The static decomposition should be one of the last operations that the Multidisciplinary Systems Engineer deals with in the architecture design.
- **Improper handling of Governance and System Management**: Strategic Monitoring and Orchestration of the system (Governance) and Tactical Monitoring of the system (System Management) are often mixed and handled as one set of system functionality. Behavioral management as well as temporal management of the System of Systems are related but must be managed within the design individually. These are very distinct functions and mixing them will result in improper use of both within the overall Functional Design.

Question to think about:
What do you think the effect is on the usability of the system, when system monitoring is not included in the design?

2.1.4.1.2 Risk Analysis

Risk analysis is important to assess the potential technical risks associated with the System of Systems environments, technical constraints, proposed new technologies, or anything that could pose technical issues throughout the program lifecycle. It is impossible to assess 100 % of the potential risks to the system (e.g., predicting a category 5 hurricane hitting New Orleans). At the System of Systems level, any legacy system that will be used in the overall design must be evaluated for impacts to the optimization and overall operations of the System of Systems. For the Multidisciplinary Systems Engineer, risk analysis must involve Systems Engineers from each of the elements of the System of Systems, along with Subject Matter Experts (SMEs) from each of the technologies to be employed throughout the System of Systems in order to adequately assess the technical risks to the program. Both the likelihood of occurrence and overall system impacts posed by the use of the legacy system are important risks to be determined and understood and then decisions must be made as to how to handle the potential risks. More will be discussed in later Chapters and Risk Management is normally its own course within the overall Multidisciplinary Systems Engineering curriculum.

Question to think about:
What effects are there on the risk posture of a project when Inputs/Outputs are ignores?

2.1.4.1.3 Decision Analysis

Decision Analysis involves determining the methodologies and metrics that will be utilized throughout the System of Systems to assess performance against the customers' objectives. Some of the issues to be analyzed are:

- What metrics and measures of effectiveness (MOEs) should be collected?
- When are the appropriate times to take metrics?
- Which system nodes should be monitored throughout the System of Systems?
- What kind of root cause analyses should be utilized for problem identification/resolution?
- What types of Alerts, Warnings, and Event notification should be utilized throughout the System of Systems?
- What changes might impact the System of Systems, to include changes in technology as well as mission/business need changes?

2.1.4.1.4 Data/Information Analysis

The purpose of Data/Information Analysis (which is part of the discipline of Information Engineering) is to determine the data/information objects that are required to support the overall system-level requirements, objectives, goals, user

scenarios, etc. Understanding the types, quantity, and quality of the data that are required by the system can radically change the overall design of the system. "*Big Data*" has become a major field of study in engineering over the last two decades, as the availability of analysis techniques increases, and the need for more detailed analysis grows (e.g., cybersecurity), more and more data are required to feed the analysis and decision engines of today and the future. If you are building a toaster, the quality of data required by sensors is trivial compared to, let's say, our GPS systems. In GPS, quality of data/information is paramount since the world relies on the accuracy of GPS data for everything from how get to the nearest Starbucks, to precision timing of transactions on Wall Street, to military operations; Starbucks being the most imperative one on the list, of course.

Not only does data volume needs to be considered, but the heterogeneity of the data must be considered, as well as who needs access to which types of data. In performing Data/Information Analysis, the Multidisciplinary Systems Engineer should be determining a host of data/information characteristics that must be taken into account by the Multidisciplinary Systems Engineer in the overall architectural design and implementation of the System of Systems. The System of Systems Data/information ontology should be developed with subsequent data/information taxonomies for each element of the System of Systems. Figure 2.4 is an example of an ontology created for multi-media data/information. Some of the considerations are:

- Types of data (textual, scientific, etc.)
- Data representations (e.g., structured, semi-structured, unstructured, etc.)
- Precision required for each type of data/information
- Overall data throughput for the System of Systems
- Message types and quantities required across the System of Systems information system

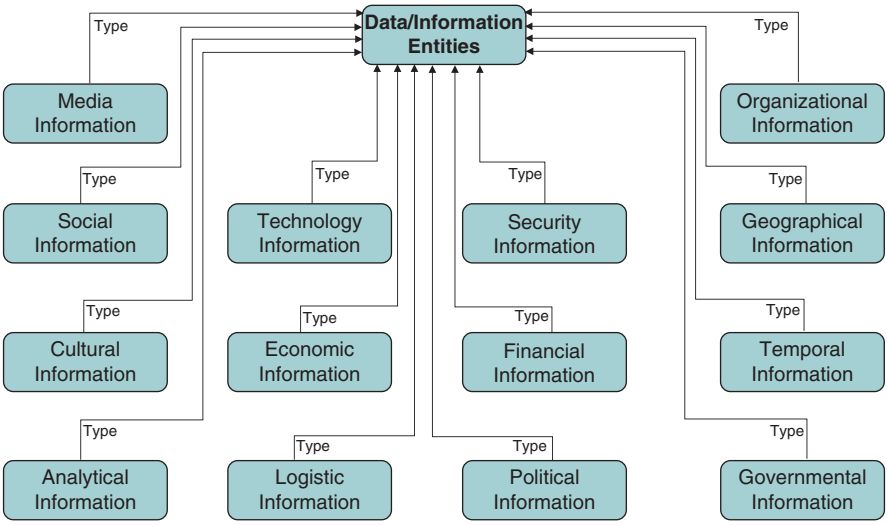

Fig. 2.4 Example of a data/information ontology

- Usage of each data type
- Data/Information interfaces required (both external and internal interfaces)
- Data/Information protocols that are needed/required for the System of Systems
- Data transformations required; possibly due of the use of legacy systems, or due to external interface requirements
- Data/information storage requirements
- Data/Information sources (where does all the data come from, both internally generated data/information as well as externally generated data/information)
- Data management and maintenance
- Data/Information presentation requirements

Question to think about:
What are the key differences between data ontology vs. storage structures for data?

2.1.4.2 Technical Frame of Reference

For System of Systems design, implementation, integration, and testing generally requires a distributed development, with each element of the architecture having a separate development team; while the Multidisciplinary Chief Systems Engineer must oversee the distributed efforts to ensure a smooth integration of the overall system. In order to accomplish this, a common technical frame of reference must be established among the various elements. The common technical frame of reference ensures the element-level technical solutions meet all requirements, goals, and mission/business needs in terms of technical performance, data/information needs, which then allows Integration, Verification and Validation, Transition, Operations, and Maintenance to be accomplished without major rework between elements. To that end, there are a number of technical frames of reference that must be established by the Multidisciplinary Systems Engineering organization for the System of Systems program.

Question to think about:
Without common technical frame(s) of reference, will the system actually integrate and operate without extensive cost and schedule impacts?

2.1.4.2.1 Requirements Decomposition and Allocation

The challenge with distributed development is ensuring that requirements are properly decomposed and derived for each element, that requirements are not missing or misunderstood, and there exists a clear roadmap that establishes how the element-level requirements and their subsequent decomposition can be used to demonstrate compliance with the System of Systems-level requirements.

Table 2.1 Requirements decomposition and derivation criteria

Characteristic	Description
Requirement prioritization	Comparison weighting based on overall mission/business needs
Requirement source	Where did the requirement come from: customer requirements, concept of operations, constrants, laws, and/or agency restrictions (e.g., FAA)
Requirement usage	Technical functionality alternatives that can be used to execute this requirements
Requirement impact	Performance consequences, potential risks the requirements pose on the system, architecture impacts
Requirement user scenario	How will this requirement be utilized by the operational system and how the people who will operate the system will benefit by this requirement

Requirements decomposition and derivation is one of the most important aspects of system design, where bad requirements decomposition and derivation is almost impossible to overcome during the development phase without a major amount of rework, including redesign of the various architectures and new development efforts. Bad requirements will have major impacts on the overall cost and schedule of the project. Creating a common requirements framework and management scheme for all requirements decomposition and derivation among each element, therefore is paramount for program success. Table 2.1 illustrates an example template for all requirements work. There are many electronic tools to manage requirements; the list below shows a few:

- IBM—Rational DOORS Next Generation®
- Intland Software GmbH—codeBeamer Requirements Management®
- Hewlett-Packard—HP Agile Manager®
- Polarion Software—Polarion Requirements (2014 SR2)®
- TechExcel—DevSpec®

Question to think about:
Are requirements necessary for a system development?

2.1.4.2.2 Data/Information Decomposition and Allocation

In order to understand how the data/information in the system should be allocated and utilized across the System of Systems, as well as the overall capabilities required by the data/information at each level in the architecture, the Multidisciplinary Systems Engineer must understand the following:

- **Data/Information Quality**: here we define quality in terms of the accuracy, compliance, and completeness of data/information objects and their overall ability to satisfy the system requirements, goals, objectives, etc. This includes topics like data usage, data rate of change within the systems, and synchronous vs. asynchronous data issues.

- **Data/Information Structure**: How will the data/information be organized across the System of System architecture, including the structure to be utilized at each level?
- **Data/Information Architecture**: This includes information structures, processes, and organizations (who is responsible), as well as storage alternatives for data at the System of Systems Enterprise level.
- **Data/Information Governance**: The roles, responsibilities, policies, procedures, and governing laws that are required to manage all data/information across the System of Systems.
- **Data/Information Security**: This involves identifying the security requirements on the data in the system, and who has access to which data/information in the system, and how to manage authentication and authorization for user/process access to protected information.

2.1.4.3 Technical Management Frame of Reference

The role of Technical Management within the System of Systems development is to understand the technology base for the program. For long-term programs, Technology Management is a continuous set of processes and strategies that the Multidisciplinary Systems Engineer utilizes to manage the use of technology during development, with a goal of optimizing the overall system technology to meet the Quality, Cost Profile, and Schedule for the program. Technical Management utilizes many concepts within its overall frame of reference:

- **System of Systems Technology Roadmap**: what technologies are available when, and how will they be introduced into the system over the life of the program.
- **System of Systems Technology Strategy**: the role technology plays in the overall success of the System of Systems program.

2.1.4.3.1 Planning Management

Technical Planning Management involves organizing, controlling, and executing the processes, procedures, protocols, and activities required meet the requirements, goals, needs, and performance of the System of Systems. For the Multidisciplinary Systems Engineer, technical planning requires constant attention from the early beginnings of a program, to its ultimate retirement when a new system is built and transitioned. This includes optimizing the allocation of the necessary inputs, functions, and outputs to integrate them to meet all pre-defined System of System objectives. Planning Management has been around for thousands of years and was probably put in place by early Egyptian Pyramid builders. Figure 2.5 below is a picture of a carving discovered from Rome in 113 AD, depicting roman soldiers building a fortress.

Fig. 2.5 Picture of carving showing a roman building project from 113 AD

2.1.4.3.2 Requirements Management

As we have discussed several times and will continue to discuss, because it is ***that*** important, the handling of requirements is one of the single-most important activities the Multidisciplinary Systems Engineer must clearly understand and handle well if program success is at all a priority (of course, it ***always*** is). Managing the Technical Requirements is necessary to define the technical issues that the Multidisciplinary Systems Engineer must deal with in creating the overall architectures and designs required for successful delivery of a System of Systems that meets all requirements, goals, and needs. Part of Requirements Management is to make sure the requirements at every level in the systems are:

- **Defined Precisely**: Technical Requirements (all requirements really) should be clearly worded, sufficiently detailed to eliminate ambiguity, necessary (related to the mission/business needs), and testable in relation to the tier of the system.
- **Analyzed for Feasibility**: Part of Requirements Planning is to determine the feasibility of the requirements at each level of the System of Systems, ensuring that there are no conflicts in the requirements that will make execution of the system problematic. This includes an impact analysis of the major requirements and their impacts on the overall architecture.

2.1.4.3.3 Risk Management

Once program risks have been determined and categorized, as discussed earlier, program risks must be managed throughout the lifecycle of the system to ensure system success in the event any of the identified risks are realized. Risk Management involves determining the processes, methodologies, metrics, tools and Risk

Management infrastructure which is appropriate for each System of Systems design and implementation. Having a standard methodology for Risk Management across each element in the System of Systems design allows seamless communication of mitigation strategies for each program risk. There are many Risk Management COTS S/W that is available, including freeware packages. Several are listed below:

- Metric Stream—Risk Management Software Solutions®
- SAI GLOBAL—Enterprise Risk Management (ERM)®
- Integrated GRC—Acuity Risk Management®
- SwordActiveRisk—Active Risk Manager®

2.1.4.3.4 Data/Information Management

Data/Information Management runs through the entire system lifecycle, from initial requirements decomposition/derivation/allocation through system retirement (cradle-to-grave). Whether all data is contained within relational databases, contained as individual data items, or as part of flat files, all data must be adequately managed throughout the System of Systems in order to track and control data changes as the design and or system environment changes.

Data and information management must the integrated into the Configuration and Change Management processes used by Multidisciplinary Systems Engineers. This ensures that all of the data/information, archive information, database schemas, and data/information flows throughout each architecture level are controlled and managed during the development cycles. This management and control ensures that a central repository is available to all elements in the System of Systems development. This includes the date of change for the data formats on programs with multiple development periods. Note: This can be mitigated through some of the modern data definition schemas (e.g. xml), as long as the usage rules are properly defined.

The Multidisciplinary Systems Engineer creates methodologies and constructs to create and manage Metadata,[1] which is utilized to provide identification and organization of System of Systems data/information. Formal Data/Information Management allows enhanced collaboration across System of Systems elements by identifying uses and formats required by each element and storing data in one place, transforming it to its intended format as required by the element.

2.1.4.3.5 Configuration and Change Management

Configuration and Change Management is involved in managing changes to the System of Systems. This includes:

- Managing change requests
- Planning changes—application of effective change dates
- Implementing
- Assessing change impacts

[1] Metadata is information used to identify and manage data/information. Metadata is used to organize data/information and resources across the system.

The main goals of Configuration and Change Management provide traceability of changes across the System of Systems.

Questions to think about:
Even with Change Management utilized within a system development, does the technical frame of reference ensure that Change Management results are valid?

Even with Change Management utilized within a system development, does its existence ensure that the technical frame of reference will be adhered to?

What must happen to ensure Configuration and Change Management are effective from an MDSE point of view?

2.1.4.3.6 Interface Management

Management of all internal and external interfaces is critical for the Multidisciplinary Systems Engineer. Understanding all of the interface interactions, properties, protocols, and data/information formats is necessary not only for architectural design, but also is essential for integration of the System of Systems elements and testing of the System of Systems. The challenge for the Multidisciplinary Systems Engineer is specifying and designing interfaces in a way that can be utilized by all elements in the System of Systems, while adhering to industry standards. Interface Management is utilized to track all System of System boundaries and how they interact with each other (both external and internal). Interface interaction descriptions describe the operational entities used by services during all System of Systems operations.

Question to think about:
What happens in system development when interface management and requirements are not defined in terms of Systems Engineering, but in terms of Software Engineering?

2.1.5 Creating a Unifying Lexicon

Ensuring term definitions are unified across the entire System of Systems is essential to ensure that there is no confusion as data flows between elements and within element of the System of Systems [32]. The Multidisciplinary Systems Engineer needs to establish a team early on to establish a unifying term and data lexicon before architecture design has begun. When terms and definitions are not properly defined, risk to the development increases as well as potential system failure(s). A good example from history occurred with the Mars Probe in 1999.

One team was using Metric units for their element, while another team (on another element of the System of Systems) was using English Units. Unfortunately data passed between these elements for a critical operation on the Mars probe, which led to a catastrophic failure during the landing operation (a tragic mistake since *AAA* doesn't go out that far).

2.2 Software's Role in Multidisciplinary Systems Engineering

Software plays in integral part in Multidisciplinary Systems Engineering. The Software Systems Analyst role is necessary to determine whether the System of Systems architecture is implementable in software such that all the System of Systems requirements can be met. This includes analysis of hardware required to host the software, memory usage, external and internal interface traffic, and data/information flows throughout the System of Systems. This includes development of the Service Architecture and the Software Fault Architecture that will be required to assess the Reliability, Maintainability, and Availability (RMA) for the delivered System of Systems. The Software Analyst Engineering role within Multidisciplinary Systems Engineering is a very distinct role, which is to elicit, prioritize, suggest, and negotiate the decomposed/derived/allocated requirements in order to simplify and optimize the envisioned software design as much as possible. The Multidisciplinary Systems Engineer must be able to think like a Software Systems Analyst in order to create a System of Systems Architecture that not only can be realized in software, but also seamlessly integrates across the System of Systems elements [15]. The Software Systems Analyst role within Multidisciplinary Systems Engineering facilitates collaboration between systems and software engineers and allows creation of overriding principles for integration, interoperability, modifiability, and expandability of the System of Systems final product. Figure 2.6 below illustrates the Systems vs. Software roles as well as the combined roles the Multidisciplinary Systems Engineer must embrace [33].

2.2.1 Computational Thought and Application

Multidisciplinary Systems Engineering developed out of the need for foundational science in engineering across disciplines/domains, and invariably the systems developed and employed specifically for those domains. Understand that many disciplines reach out and touch other disciplines out of a desired need. The desire to automate biological and chemical science required software. The desire for automated mechanical systems in the auto and oil industries required software. Over time, through observation and implementation, the software engineering discipline instituted pattern development and recognition techniques for common and optimize development and implementation. Software engineers today, with longevity,

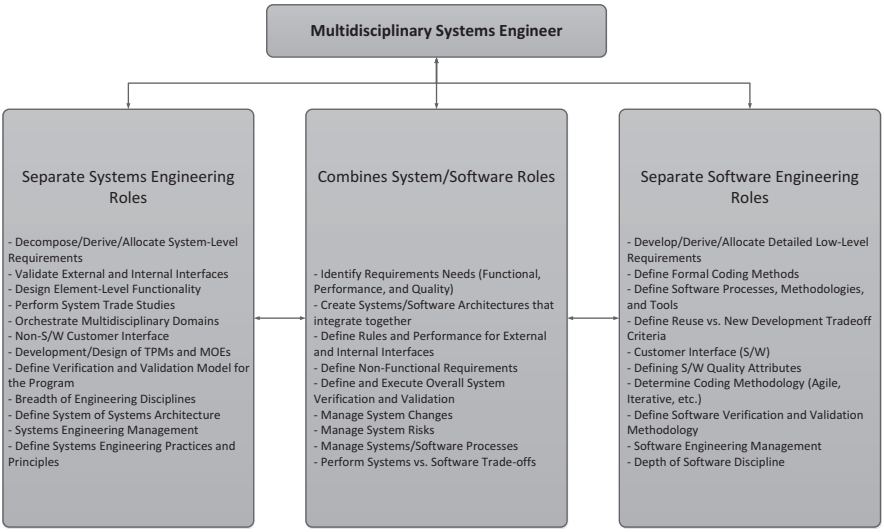

Fig. 2.6 Systems vs. software roles of the multidisciplinary systems engineer

have an uncanny ability honed over years to rapidly find the common denominators needed in a design and provide a design solution. The thought processes and mechanisms together are known as Computational Thinking (CT). Simultaneously, over the last few decades, via the Moore's Law expansion, many other disciplines developed their own improvements from the inside-out or Interdisciplinary, as seen in the Chap. 1, Fig. 1.4, as opposed to common improvements across disciplines (e.g. Multi- Trans-). This emphasizes the difference: Multidisciplinary is the act of applying methods from two or more disciplines to single topic within one discipline, whereas, Transdisciplinary, a higher form of Multidisciplinary, focuses on the science and engineering required to engineer for commonality across two or more disciplines. This fine-grained nuance is essential to the development of the right frame of reference to solve difficult problems and develop complex systems (System of Systems) within a discipline you might not initially fully comprehend.

2.2.1.1 Recursion's Role in Layering Abstraction

One of the difficult concepts for software engineers to initially understand is the time where they are exposed to the difficult concepts required for effective recursive engineering. Although not a difficult topic to understand, recursion can be a struggle to implement effectively. You will find many definitions of recursion. Recursion simply put, is an iterative process. The end state of a recursive process is an optimal solution based upon a set of thresholds defined as part of the criteria for starting the

process in the first place. The inherent artistic nature of a good recursive process is one that most effectively processes all of the items and parameters while the process continues to unfold. Recursion is well-known in many disciplines where repetition is required. Repeating the same method over and over without the need to continuously understand the output of each of the iterations is what we would liken to the simplest form of recursion. We will discuss what is at the core of achieving successful optimization of any Multidisciplinary Systems Engineering design, a concept called Computational Thinking (CT).

2.2.1.2 Application of Computational Analysis and Thinking

The successful and most optimized applications of recursion, which includes iteratively optimized processes through the use of *abstraction*, requires determining the relationships between the different layers as the process recursively iterates through them, while simultaneously adjusting the parameters of optimization at each iterative layer. Abstraction is at the core of what is known as Computational Thinking, which is an inherently humanistic inquisitive process like 5WH, a term used by many for Who? What? When? Where? Why? and How? To achieve effective CT we apply 5WH and research in Cognitive Psychology to help you understand and apply Computational Thinking processes to be effective during engineering multidisciplinary systems on any project. As we have discussed, engineering across disciplines requires one to take into account many parameters and layers of abstraction in order to successfully achieve a solution. This cognitive process is known as Recombinant kNowledge Assimilation (RNA), comprising a set of instructions underlying learning algorithms [17], each using 5WH: discovery, decomposition and reduction, compare and contrast, association and normalization. Figure 2.7 depicts the high-level process occurring as one evolves through the set of cognitive RNA instructions analyzing, iterating and recombining abstraction layer content.

Discovery sub-process obtains content from the information domain; the content is decomposed and reduced to a threshold of understanding within the temporary knowledge domain. Subsequently, it is compared and contrasted to additional iteratively discovered content. Subsequently, relationships of understanding are made via association combined with normalization which finalizes the understanding of the new abstracted layer within the Knowledge domain. Effective recursive and

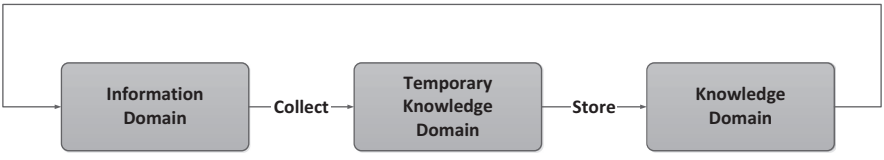

Fig. 2.7 Knowledge recombination domains

iterative optimization ends up as a feedback loop, with new Information Domain content utilized for the next iterations. This emphasizes the most important question a Multidisciplinary Systems Engineer working in any domain must ask, "How do I know when I have achieved an optimal level of abstraction for my system?", "How do I know when I am done?"

Questions to think about:
What happens when Systems Engineering is not complete relative to:

(a) *The system design?*
(b) *Hardware and software development?*
(c) *Cost and schedule?*
(d) *Risk?*

2.2.1.3 System Optimization Via CT Automation

The previous paragraphs introduced you to the underlying constructs and processes for using Computational Thinking. However, to answer the questions posed in the previous paragraph in terms of what is occurring during System Design optimization we will focus on the last RNA instruction, Normalization. As each layer of abstraction is defined and the relationships between the parameters of each of the iterations are understood a certain simplification or generalization naturally occurs. The evolving generalization improves after each of the iterations and is evolving into a common way of approaching the process until that final iteration matches the design and engineering thresholds one is attempting to attain. In a physical system design instantiation this might include a set of tolerances or design range in which a machine must operate within. The CT infused systems engineering design actions you are performing are automating the layered abstractions into a common approach, when implemented, become the optimized common system processes. Therefore, whether optimizing System Engineering designs within a discipline or across multiple disciplines evolves to enhanced set of common automatable solutions mechanized by Computational Thinking.

2.3 The Psychology of Multidisciplinary Systems Engineering

Multidisciplinary Systems Engineering requires a paradigm shift in the classical Systems Engineer. As programs become more and more complex, and as System of Systems become more prevalent, we must embrace change (even though it seems change is hard for engineers in general). The psychology of Multidisciplinary Systems Engineering is all about a change in Systems Engineering which will address the Multidisciplinary aspects of modern systems.

2.3.1 Resistance to Change

If change has always been an integral part of life, why do we resist it so? Why, in every generation, do we have major resistance to change? *It seems obvious that there should be thought put into change theories as we ask for people and environments to change.*

Some key components to encourage change are empowerment and communication. People need time to think about expected change. As we discussed earlier, people are good at change in order to master or improve their world or environment. When people remain agile they are better at being agile. When change is part of the regular process then one becomes agile. Alternatively, when one is used to doing something exactly the same way or systematically then it becomes the way one likes to operate.

Why, in particular, do some engineers not like change? When asking this question it seems obvious to consider education. What is it that has been required of engineers in the past and present and what will be required of them in the future, particularly for Multidisciplinary Systems Engineers. Juan Lucena [34] hypothesizes connecting engineers' educational experiences with their response to organizational change and offers a curriculum proposal to help engineers prepare for changing work organizations. As our technology increases and our work world becomes more agile it makes sense also that soft skills will become more and more important for engineers. Multidisciplinary Systems Engineers must organize themselves to optimize performance with soft skills to embrace the multiple domains they must utilize.

Trust is the most important factor in change. Trust helps to balance fear which is often the root of most resistance. Who in their right mind will blindly make changes without trusting others? Agile methods increase trust by increasing transparency, accountability, communication and knowledge sharing [35]. Iteration/sprint planning methods give members visibility on requirements, individual assignment, and agreed estimates. People get the information at the same time. The daily stand up provides visibility and transparency so issues can be addressed immediately. People will know if someone is behind. The sprint/iteration retrospective provides transparency and visibility regarding goals. This agile design builds trust among the team member as they are able to see others trustworthiness and competencies as they continue working together.

Crawford et al. [36] suggest that employees who had higher creativity, worked on complex challenging jobs and had supportive non-controlling leadership, produced more creative work. If workers see that their ideas are encouraged and accepted they are more likely to be creative. Empowerment and trust encourage creativity which encourages change which encourages agility [35].

Think about the complexities that people bring to projects. Think about projects of many teams made up of many people, virtual teams, many sites, many locations, many levels of expertise and experience, many cultures and places that people live. Then add in technology, many technologies, changing technologies, developing and revolutionary technologies. It becomes more and more clear that soft skills are incredibly valuable to engineers and particularly to agile teams. Azim [37], shows that up to 75 % of project complexity has to do with the human

factor or the people in the project. They claim that soft skills are important in the implementation of plans. Soft skills are clearly important in any complex project and in all phases of change previously discussed, particularly in the commitment and transition of change.

Soft skills include organizational, teamwork, communication, and other people based skills. As technology matures and new technologies emerge it is imperative that the teams and people in the teams become more agile. Not only is technology constantly changing so are people. It seems ever so important that the soft skills come with the hard skills. Multidisciplinary Systems Engineers must be willing to embrace significant change in the industry in order to be effective with large System of Systems designs.

Question to think about
What happens to the company that fails to address change?

2.4 Case Study: Application Multidisciplinary Systems Engineering (MDSE) for Information Systems Applications to Biology

This case study is used to show how Multidisciplinary System Engineering practices should be applied to solution sets for something other than a traditional information management system, weapon system, power system, etc. The tools needed to develop a system used by biologists are relatively the same as another other system. The need for top down analysis, development of requirements and architectural solution remain the same, and the application of MDSE will enhance the outcome for the biologist's system. The following case study outlines a premise/argument for the use of multi-, trans-disciplinary engineering techniques to implement and then upgrade a system solution in the field of Systems Biology. The following materials provide the background needed to assess if MDSE can be/should be applied for the development of a system used in biological engineering.

Biological Engineering has only recently emerged from an applied science to its current status as a legitimate field of engineering professional practice. Biological systems exhibit a number of properties that challenge our traditional ways of thinking. Among the most dramatic are time and age dependent changes in form, composition and behavior, including evolution, adaptation, competition, emergent properties and others. Many biological processes respond to external stimuli such as light, temperature, pressure, population density, nutrition, etc. in sometimes-unpredictable ways through self-regulation, aggression, and learned or genetically derived anticipatory actions. Rarely can we design one element of a process without deep connections to surrounding elements [38].

Today, there exists much discussion about the need for transdisciplinary application of today's engineering solutions [27], as well as, the lack of use of transdisciplinary engineering theories in enhancing and augmenting the field of Systems Biology [39].

Hence, the goal of this section is to explore multi-, trans-disciplinary options for Systems Biology, and use those options to provide an analysis and potential suggested solutions to a specific set of Systems Biology requirements and problem set.

In biology, inclusive systems and interactions tend to produce more robust and sustainable outcomes—in biology the operative word is AND versus the traditional design choice OR in engineering. In spite of the great advances in biology and biological engineering science, designers are faced with high levels of uncertainty and many unknowns in their design efforts. Chaotic behavior is "normal" in biology. Every day we are surprised by discovery of previously unknown interactions between organisms and their environment, and among organisms in a population. The biological world is constantly affected by environmental and ecological pressures within and beyond what ecologists call the "natural range of historic conditions." With all this chaos, uncertainty and unknowable stuff, there are fundamental principles that we can use as anchors to form trends to follow function in biological engineering design."

Systems Biology is wrought with high levels of uncertainty and many unknowns as well. Antonsson [40], stated that the advancement of design theory requires "investigation into aids for the design process." The need for disciplines like bioscience and design science to investigate beyond the boundaries of their existing design concepts to create aids in advancing the design processes is great. Nowhere is this more self-evident than within the bioscience discipline. Almost to the point of understatement, are the stimulating discussions that come from scientific methods and concepts currently under debate in bioscience, Evolution versus Intelligent Design (ID). Scientific methods when applied to design theory, as in creating data and design process abstraction, do not stimulate the kind of fervor generated by Evolution versus ID. However, they commonly stimulate discussion surrounding the transdisciplinary paradigm concepts regarding mutually sharing of discipline methods and subjects between them.

Hence, we introduce multidisciplinary concepts and apply them to the problem set to achieve a specific set of qualitative solutions. Engineering disciplines which are either more mature, or have generated solutions to similar problem sets within their domain, can potentially provide added efficiencies and benefits because of similarities to requirements, designs, and problems in other domains.

2.4.1 Discipline Background Investigation

Examining the use of specific theories, methods, and practices known as First Principles lays the foundation of understanding. This must be performed before delving deeper in processes and methodologies, as well as addressing potential solutions to specific discipline problem sets. Once a foundational layer of First Principles is laid a specific set of requirements can be undertaken. In the following example we will examine a specific architecture; processes, methodologies and mechanisms used in a system developed to perform comparative System Biology analysis and look for potential optimizations.

A common mechanism for performing System Biological study is the use of micro-cell arrays to analyze cell growth to help determine cause and effect. A standard way of performing oncology research without the use of human cells is the use of less complex and more readily available yeast cells. A yeast processing system has several key components: Cell Array Analysis, an Experiment Management System, and Lab Automation in the form of robotics. A cursory analysis of disciplines in use within this system yields: image analysis, software, and hardware, along with command and control systems for the robotics. Each of these disciplines now requires a threshold of study of first principles to obtain a reasonable understanding in order to determine an initial system-level Discovery and Decomposition.

As part of our increased transdisciplinary study of foundational content relative to this new unknown system we look to recognize/identify which disciplines could provide certain optimizations related to our system. Upon doing so, we find that our study time should include theory and applications from the fields of Photogrammetry for the potential optimization of the micro-array system, Software Engineering for the software architecture and potential specialized mathematics for the imagery and analysis. Next, we begin using the application and theories from within the disciplines: software and hardware Systems Architecture, analyzing current workflow or business logic processes in use, and perhaps examining and augmenting the current mathematical formulations and methodologies to improve the accuracy and efficiency of the current cell array system design [41].

Photogrammetry [42] is a remote sensing technology which incorporates methods from many disciplines, including optics and projective geometry, to determine the geometric properties of objects from photographic images. Upon detailing the software system it becomes apparent that the potential optimization benefits of applying Software Engineering methods and theories to this Systems Biology analysis system can derive from componentization theory and Object Oriented Design (OOD) [16], which has been on the forefront of the information age since the late 1980s. Additionally, it also well known that the use Service Oriented Architectures (SOA's) to administer functionality, organize data and normalize workflow or business processes within a system, can have potential benefits for this system. Both Object Oriented Design and Service Oriented Architectures being methodologies key to the Multidisciplinary Systems Engineering approaches to System of Systems designs.

2.4.2 Current System Design: Experimental Approaches

An example system diagram of a semi-automated yeast processing system is depicted in (Fig. 2.8).[2] From left to right the graphics below describe an example system design within a systems biology lab. The system uses a number of paradigms and approaches to as a goal to increase the ability to determine phenotype's

[2] Depicts a representation of processing environment from the University of Alabama, Birmingham Hartman Genetics Lab (circa 2007).

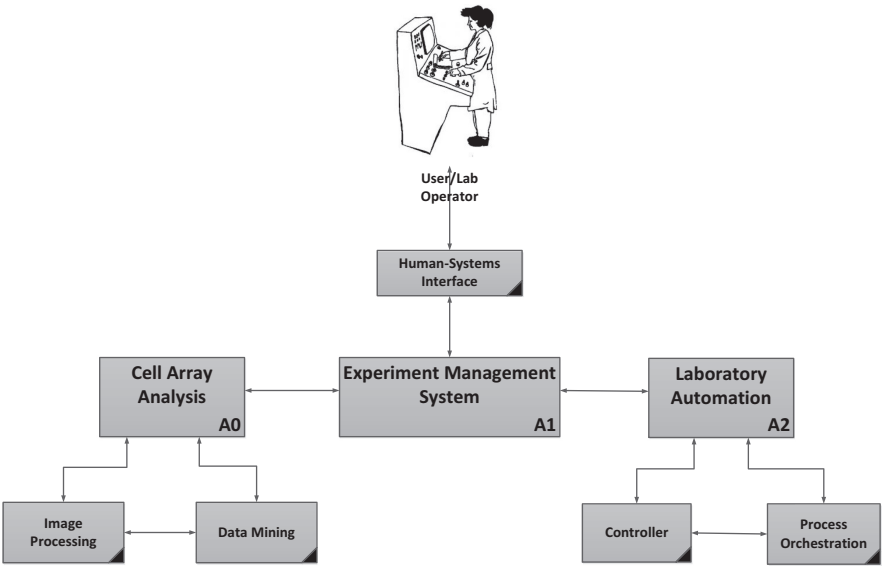

Fig. 2.8 Yeast processing system diagram

(specific outcomes) thru a systematic analysis approach of Haploid deletion strains of yeast (monitoring of cell growth), Cellular Array Technology Development, using Perturbation strategies (pair-wise approach to Chemical sensitivity and Genetic mutation combinations) and by Quantification of Interactions; determining the effect of gene deletion on phenotypic response to perturbation [43]. An analysis of the processing part of the system yields an example UML diagram depicting the state transitions of the current cell array yeast processing (Fig. 2.9).

2.4.3 Current Hardware and Infrastructure

The collection and processing, accuracy and measurement of the data are our focus with respect to the hardware. The hardware consists of a scanner, USB interface, PC hardware, and agar plates made up of Petri dish and nutrient based gelatin like solution. An expressions scanner is used for data collection. The data samples are scanned at 140 DPI for agar plates containing 8×12 culture arrays or 280 DPI for agar plates containing 16×24 culture arrays. The scanner comes with an accessory unit for scanning transmitted light images. Data collection is always performed using transmitted (film setting) light, equivalent to camera images with backlighting source. Concerning the PC, it was assumed that the commercial processor platform was sufficient to execute a Java program and save the results. Concerning agar plates, heated/liquid agar (like jelly) is poured into a plastic rectangular Petri dish with sidewalls, after it cools and/solidifies, yeast are spotted onto the surface. Light passes from above through the yeast cells, the agar, and the plastic bottom of the Petri dish and the glass platform of the scanning instrument to the detector.

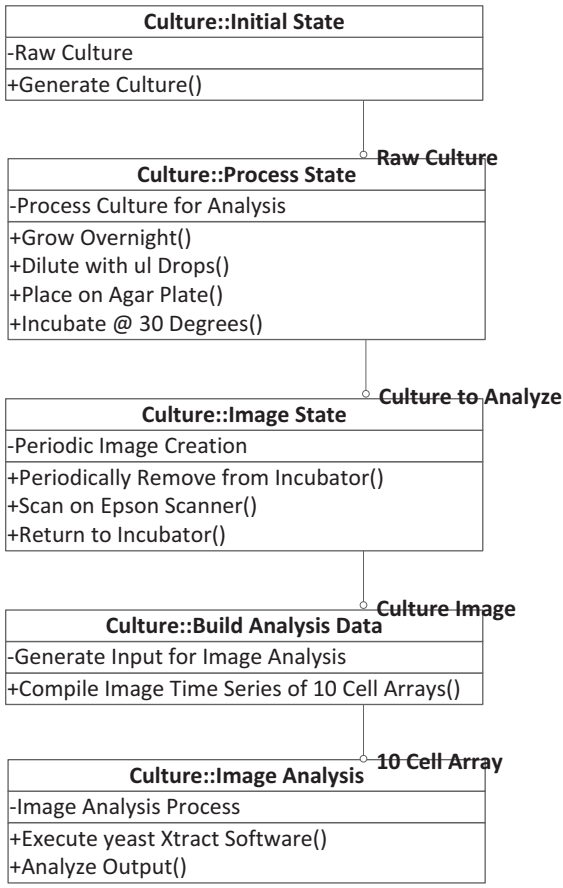

Fig. 2.9 System UML class diagram

A first look investigation of the method and device used to collect the imagery was performed. The basic collection method appears to experience normal image quality distortion problems in terms of focus, and lens distortion. The result of this distortion can cause variation in culture collects in terms of size and density based on the position of the culture spot near the center versus the edge of the scanner platform. These problems are inherent in any image collection that can be corrected by applying distortion correction mechanisms.

2.4.4 Current Software Environment

An analysis of the current Java software shows use of current Java libraries and Java2d only classes to perform automated analysis of the yeast experiments on cell array scans to determine whether changes have occurred and to what magnitude. This is accomplished by modeling an elliptical region slightly smaller than the yeast growth area and determining pixel recognition of values within the ellipse

and outside the ellipse. Each "Culture Image" object holds the growth in pixels at a given moment in time. The author attempts to compensate for some of the error at the edges of each specimen by shifting pixels, et al., re-centering the ellipse and in all cases is summing up the intensity of all pixels for a later comparison against other specimens.

In the proposed example system changes below, we can show that with added algorithmic modifications for error correction due to scanner glass, etc. we can alter the software effectively. Additionally, we can then describe some overall software architectural changes to the system as a whole to speed analysis and data fusion in the existing lab, as well as, among external partners with data to share.

2.4.5 Current Experiment Algorithms

The program expects an initial input parameter file to set the scanner and other common parameters. Then it starts reading the input tiff file data that contains the image scans of ten agar plates containing 16 by 24 culture samples. Figure 2.10 [44] is an example of the input file.

Finally the software creates a temporary folder within a folder and copies this file over to temp. Then it is ready to start analysis processing. For each agar plate output the program creates a directory for each plate and then copies each annotated spot detection image into the proper directory. Figure 2.11 shows an example of an annotated spot detection file. Then as a final step the Main.java program creates three output ascii text files containing the spot intensities, and area for each culture spot by scan, row and column [44].

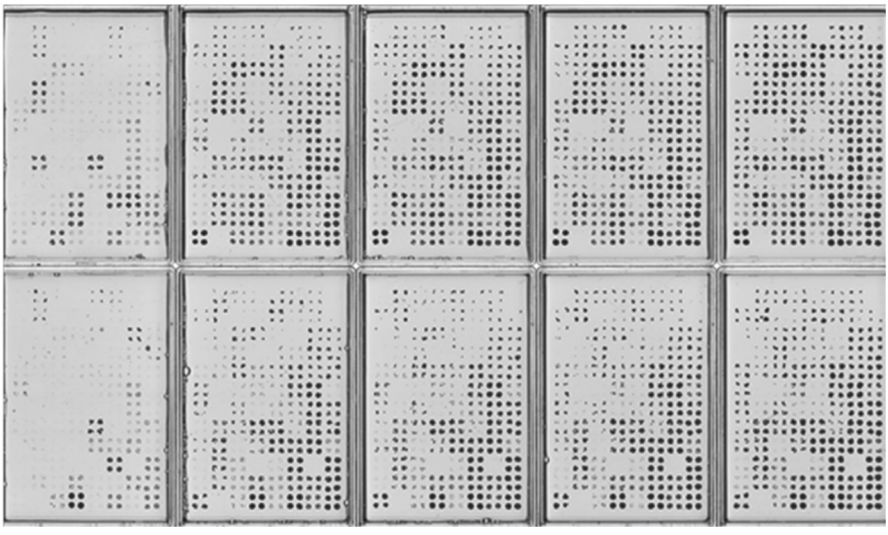

Fig. 2.10 Ten agar plate input scan

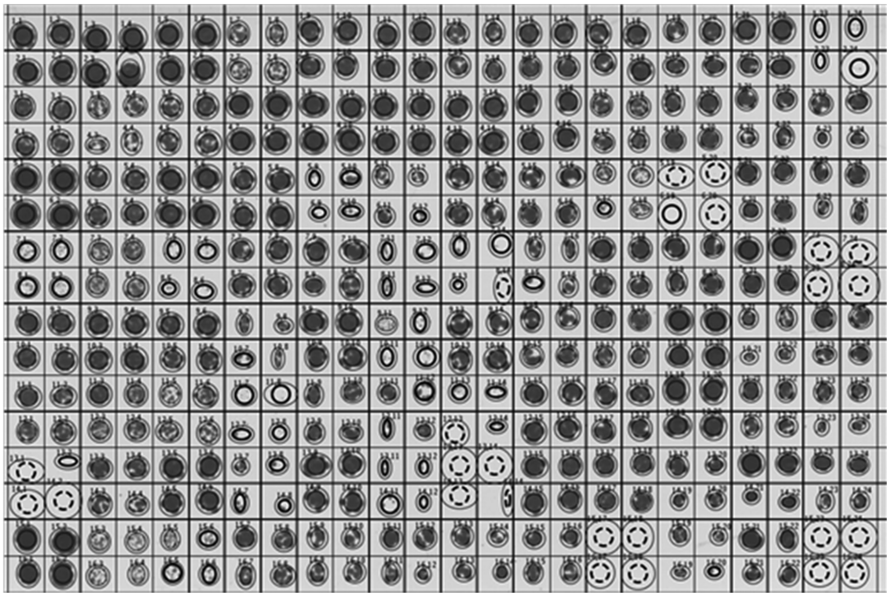

Fig. 2.11 Sample spot detection file

A Main.java function performs the major calculations. It considers background pixel content and figures out the ellipse and where the spot lies as an analog of the function it overlays. Then a function to detect the ellipse simply uses the values from the each set of spot passed-in values for all the other time points. Next the spots are aligned to the object's image so as to minimize the difference between it and an external image the reference image is moved accordingly, −x…+x and also −y…+y. Lastly, the software returns the average pixel intensity within each ellipse.

2.4.6 System Upgrades

2.4.6.1 Hardware and Infrastructure

The Systems Biology infrastructure field is changing from one of disjoint, sequential processes to an area that is centered on biological properties. These properties consist of models that incorporate sequence information, high throughput data, pathway models, etc. These biological properties are also scalable to data quantities [45]. We have realized through our analysis that causal error can be introduced into the system by the scanner, and agar plate. Specifically, the errors associated with the scanner are the fixed focal length, the linear distortion of the scanner lens and the radial distortion on the edge of the scanner lens. The agar plate error is comprised of the depth or thickness of Petri dish, agar gelatin constitution, and the reflectance of both. Due to these factors, there exists a density distortion factor of the agar.

2.4.6.2 Software Enhancements

Source Code Enhancements

Changes should be made to the algorithm as currently implemented to take into account the above mentioned error introduced by the scanner and the agar plate into the decision making process. The scanner focal length must be accounted for in the software as well as the radial distortion of the scanner lens. New algorithms should replace the current set that take into account summation of light and dark pixels to arrive at a value used to make the magnitude decisions. Hence, when implementing the lens modeling and algorithm implementations edge distortions should be removed from the equation or at least abated to a great degree.

Making the suggested modifications the magnitude output value will be much more exact and will also provide a value respective of the total volume of growth in three dimensions. The current implementation achieves only a 2D representation of magnitude. The Z-axis cell growth analysis encompasses potentially many more growth cells than perceived in 2D. Hence, potentially important to the classification and decision making process. Figure 2.12 illustrates this difference [41].

2.4.6.2.1 Software Architecture Enhancements

An analysis of the software architecture revealed that a stovepipe approach was used to develop the original system. For future scalability and system pliability we propose a potential re-architecting of the system involving implementing a true J2EE or otherwise known as Java Enterprise architecture. The following describes the components of a J2EE architecture template we will use in attacking the Lab architectural issues and use it to assist in organizing the lab components being analyzed [46]:

- Client Tier
  - Presentation layer

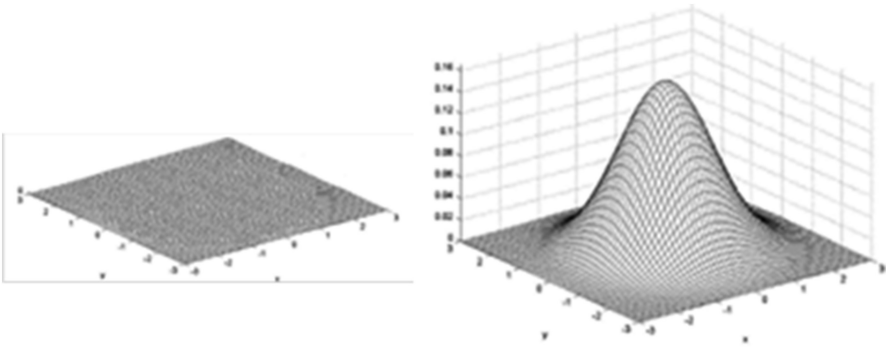

Fig. 2.12 2D vs. 3D sampling

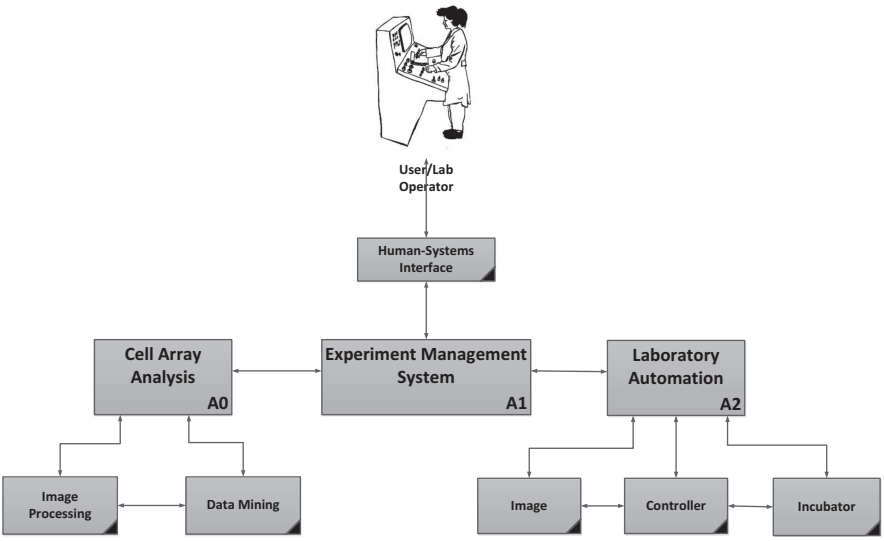

Fig. 2.13 Before J2EE

- Web Tier
  - Web Services
  - Web Applications
- Business Tier
  - Business Workflow Logic
- Enterprise Information System (EIS) Tier
  - Enterprise Data Tier

The design view of Fig. 2.13 illustrates the architecture before the J2EE enhancements, Fig. 2.14 presents a view of the architecture after the insertion of the J2EE technology and incorporates the following Architecture Tenets: Web-enabled, N-Tiered, Service Oriented (SOA), Component-based, Network-Centric, COTS-based, Collaborative Web GUI, Metadata Tagging, Seamless Data Access, Open Standards.

2.4.6.3 Algorithms Modifications

In order to relate changes in the real world to changes in the image collection sequences, we need parametric models that describe the real world environment and the image generation process. For this problem it is assumed that the real world environment is controlled and there is very limited number of variables that needs to be accommodated. In terms of image processing, the most important models the need to be considered are scene, object, camera, and illumination models. The YeastXtract program attempted to account for scene (background) and object

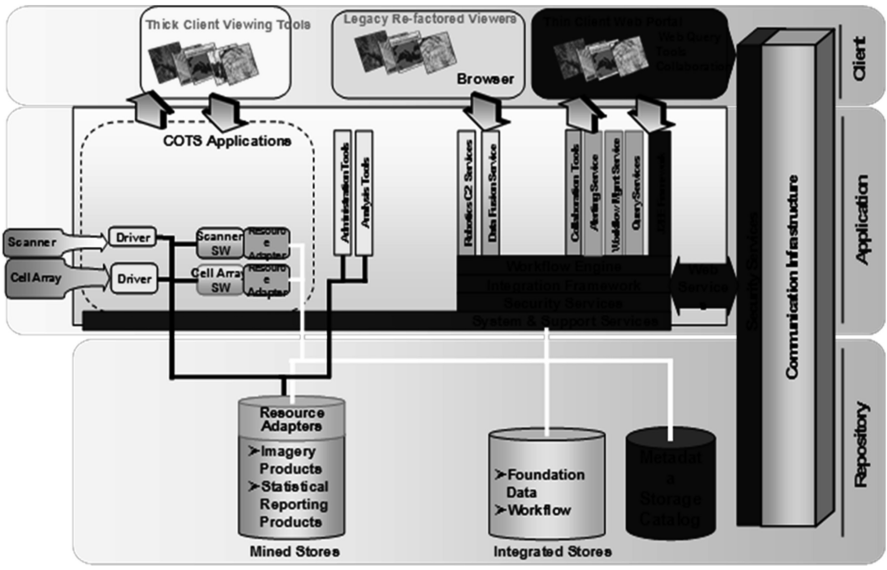

Fig. 2.14 After J2EE

Table 2.2 Naming conventions for real-world model entities

Real world	Parametric model	Model world
Real scene	Scene model	Model scene
Real object	Object model	Model object
Real texture	Texture model	Model texture
Real shape	Shape model	Model shape
Real camera	Camera model	Model camera
Real image	Image model	Model image
Real illumination	Illumination model	Model illumination

(culture spot) but did not provide any consideration adjustment in camera or illumination. Further, by using a scanner vice a digital camera (pinhole model solution) then the creation of a parametric model to compensate for collector distortion becomes much more difficult. Table 2.2 shows the terms that we use to name the parametric models, their corresponding real world entities, and the entities reconstructed according to the parametric model and presented in the real world [47].

At this point each model is covered starting with the camera model followed by illumination, object and finally scene.

2.4.7 Camera Model

The camera model describes the projection of real objects in the real scene onto the image plane of the real camera. The imaging plane is also referred to as the camera target. The image on the image plane is then converted into a digital image.

Due to the proprietary nature of commercial scanner venders, instead of a scanner, if a high quality digital camera was used then camera modeling becomes straight forward. A widely used approximation of the projection of real objects onto a real camera target is the pinhole camera model. This corresponds to an ideal pinhole camera. The geometric process for image formation in a pinhole camera has been nicely illustrated by. The process is completely determined by choosing a perspective projection center and a retinal plane. The projection of a scene point is then obtained as the intersection of a line passing through this point and the center of projection C with the retinal plane R. Most cameras are described relatively well by this model. In some cases additional effects (e.g. radial distortion) have to be taken into account. In the simplest case where the projection center is placed at the origin of the world frame and the image plane is the plane Z = 1, the projection process can be modeled as follows:

Image Projection

$$x = \frac{X}{Z} \quad y = \frac{Y}{Z} \tag{2.1}$$

For a world point (X,Y,Z) and the corresponding image point (x,y). Using the homogeneous representation of the points a linear projection equation is obtained:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This projection is illustrated in Fig. 2.15. The optical axis passes through the center of projection C and is orthogonal to the retinal plane R. Its intersection with the retinal plane is defined as the principal point c.

With an actual camera the focal length f (i.e. the distance between the center of projection and the retinal plane) will be different from 1; the coordinates of (2.1) should therefore be scaled with f to take this into account. In addition the coordinates in the image do not correspond to the physical coordinates in the retinal plane. With a CCD camera the relation between both depends on the size and shape of the pixels and of the position of the CCD chip in the camera. With a standard photo camera it depends on the scanning process through which the images are digitized. The transformation is illustrated in Fig. 2.15.

The image coordinates are obtained through the following equations

Image Transformation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & (\tan\alpha)\frac{f}{p_y} & c_x \\ & \frac{f}{p_y} & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ 1 \end{bmatrix} \tag{2.2}$$

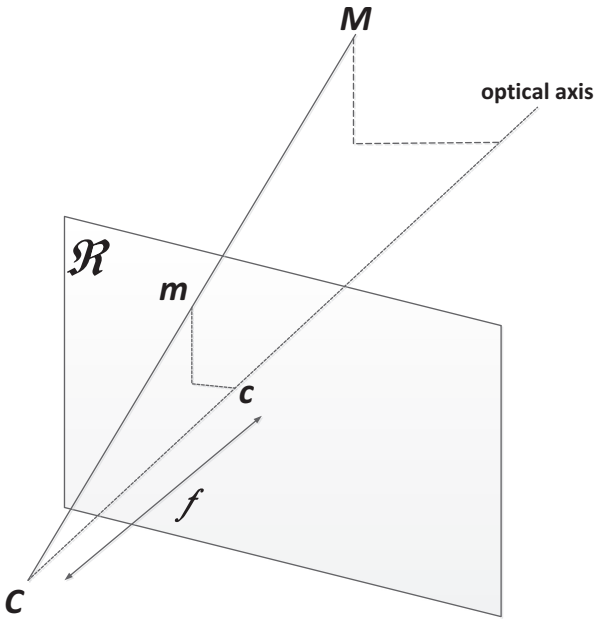

Fig. 2.15 Perspective projection

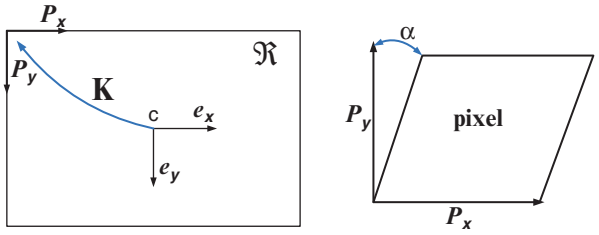

Fig. 2.16 Retinal coordinates to image coordinates

where p_x and p_y are the width and the height of the pixels, T is the principal point and α the skew angle as indicated in Fig. 2.16. Since only the ratios and are of importance the simplified notations of the following equation will be used in the remainder of this text:

Simplified Equation of (2.2)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ 1 \end{bmatrix} \tag{2.3}$$

with and being the focal length measured in width and height of the pixels, and a factor accounting for the skew due to non-rectangular pixels. The above upper triangular matrix is called the calibration matrix of the camera; and the notation will be used for it. So, the following equation describes the transformation from retinal coordinates to image coordinates.

$$m = Km_{\Re}$$

For most cameras the pixels are almost perfectly rectangular, and thus are very close to zero. Furthermore, the principal point is often close to the center of the image. These assumptions can often be used, certainly to get a suitable initialization for more complex iterative estimation procedures. For a camera with fixed optics these parameters are identical for all the images taken with the camera. For cameras which have zooming and focusing capabilities the focal length can obviously change, but also the principal point can vary. An extensive discussion of this subject can for example be found in the work of Willson [48]. At this point it is unclear which type of camera Dr. Hartman's laboratory team may migrate to in the future.

2.4.8 Illumination Model

There are two types of light sources: illuminating and reflecting. An illumination model describes how the light incident on the object influences the reflected light distribution, which is what we see. There are several such models. Illumination models can be divided into spectral and geometric models. Spectral illumination models are used if we want to model changes of color resulting from several colored light sources, indirect illumination with objects of different colors, or colored reflecting surfaces. Geometric models describe the amplitude and directional distribution of incident light. Geometric illumination models are available for ambient and for point light sources. The problem covered in this paper centers around a geometric illumination model for a point light source projecting through a semi-transparent agar background. Djokic has proposed an efficient analytical procedure for using an extended ideally diffuse light source (IDLS) with one point light source [49]. He provides example luminance calculations (horizontal, vertical and vector/magnitude) for a rectangular IDLS. Some aspects of the proposed procedure are considered in terms of photometric measurements and practical applications [49]. Hank Weghorst describes a set of algorithmic procedures that have been implemented to reduce the computational expense of producing ray-traced image models [50]. Concerning the semi-transparent agar background, YeastXtract program calculates the value of the semi-transparent background for boundary areas containing each yeast culture spot it is unclear where that value is used to adjust the calculate gray scale value associated with the yeast culture spot. During data investigation we calculated the background across all sample data files and determined that the average agar color value 225.4 on the gray scale (0 = max black to 255 = max white). Further the lowest agar color value was 222 and the highest value was 228.

This implies two things. First the light color frequency provides a gray scale value greater than 228. Second, the light color values below the gray scale value 221 belongs to yeast culture spots. Based on our data and source code inspection, it is recommended that the background value be calculated once, set in a value, and then applied it to the yeast culture spot. Pixel density values vice being calculate over and over for boundary areas containing each yeast culture spot.

2.4.9 Object Model

The object model describes assumptions about the real objects. In this problem the objects are yeast cultures in different phases of maturation. Texture models describe the surface properties of the object. In yeast cultures, if a yeast culture can exhibit different textural characteristics then these textural characteristics must be considered for inclusion in the textural components of the object model. Here, we assume that the texture of each yeast culture point is described by a color parameter. The YeastXtract program calculates the gray value for each object pixel belonging to each yeast culture spot and calculates and average gray scale value. The YeastXtract program equates each average gray scale value as a density value. To obtain a real density value for each yeast culture spot area one should create a gray scale to micron thickness value table for the valid range of gray scale (estimated to be gray scale values between 100 and 222) than modify the YeastXtract program to calculate the actual dimensions for each pixel belonging to the each yeast culture spot over time.

2.4.10 Scene Model

Having discussed how to model the illumination source, object, and camera, now the focus is on modeling the image scene. The scene model describes the world with illumination sources, objects, and cameras. Depending on the models used for different components, one arrives at different scene model. Since there should be one simple camera, illumination, and object modal the scene model is just applying the sum of the corrections of the three models discuss above.

2.4.11 Example of Discovery Research for Systems Biology Infrastructure

Current work in Systems Biology provides information on an up and coming new research paradigm. Advances in the biology discipline as DNA sequencing, the Genome project are key new areas in this field. These advances have changed the research paradigm that had been used in biology. A key attribute of this new research paradigm are the transdisciplinary research teams that work together to design

experiments, collect and process data for these focused biological areas termed "-omics". Examples of this distinct research areas are genomics (study of whole genomes) and proteomics (study of all proteins an organism makes).

Systems Biology utilizes a computing infrastructure designed to apply data management and analysis techniques to process data from areas such as gene/protein clusters to their existing functional context recorded in public databases such as GenBank, KEGG, GO and OMIM. This area within Systems Biology is specifically called Bioinformatics. The influx of data is exploding with new types of data and large amounts of the currently collected data. New techniques are being developed to integrate, manage, interpret and visualize data from diverse sources to continue advances in the Biology discipline.

In the past, the main focus of the Bioinformatics field is to develop algorithms to analyze genomic sequences in massive databases [51]. Advances in this field have necessitated a Bioinformatics infrastructure change from an analytical interpretation of data to one that deals with biological properties. These biological properties consist of cross domain models of biological processes that incorporate "sequence information, high throughput data, ontological annotation, pathway models and literature sources." This better allows the cross discipline teams' access to all of the data in a form that is meaningful to their particular areas of expertise. It has been brought about by the need to have data and application integration on diverse execution platforms. The technology base for this is current state of the art for service oriented architecture and web services. Cheng provides a succinct goal for Systems Biology infrastructure: "Infrastructure's primary task is to reverse engineer biological circuits [24]. But we also expect it to apply to bioengineering projects by supporting the design and synthesis of complex sets of molecular interactions with a particular computational, biomedical or biosynthetic purpose [52].

The initial discussion centers on two major areas of concern related to infrastructure requirements. Data integration as related to database and application integration is the first requirement discussed. Both require the retrieval and storage of data distributed across heterogeneous local and public data sources. Both relate to a single key concept. Data is unstructured and mainly text fields. Sources for this data are on the increase. The data is continually being analyzed and cross-referenced. These new outputs are then stored in new organism specific, domain specific or disease specific repositories. DiscoveryNet efforts resolved this disparity issue by using a uniform interface for queries that span relational and XML repositories. It is based on a wrapper concept that allows selection of data widgets which are customized to each of the various repository formats.

The second issue relates to application integration. A survey conducted with Biology professionals performed for DiscoveryNet resulted in an efficiency workflow concept. In other words, the need is to provide a workflow logic that will run independent tasks in parallel and dependent tasks in the correct sequence. The former will also necessitate combining the results of these parallel tasks at the end of the process. Currently, tools to meet these requirements are being developed by researchers and made available over the web. As these tools proliferate, a rapid integration framework is necessary to enable the inclusion of these new tools as they

are needed by the researchers. The functional elements of DiscoveryNet are primarily geared toward supplementing the knowledge discovery process for the Systems Biology discipline. This process is based on defining architecture for the explicit and implicit application pipelines that provide the complex analysis associated with Systems Biology. It is extended to include external applications along with the many data repositories utilized in the analysis tasks. These include the following:

- Component platform supporting deployment of functional components
- Execution management that includes Meta information to control parallel task execution and serial task execution in the proper sequence.
- Data modeling to represent the data structures that are being passed between the components
- Component registry that will make these components available to the research community in general.
- Deployment issues such as location, instantiation, etc.
- Collaboration support for the design and execution of components.
- Service engine treating the components as available services for the research community.

The field of Systems Biology focuses on analysis of molecular systems [53]. The infrastructure provides computational support to the biologists and other scientists who are performing the analysis for this data. That basic key roadmap for a systems biology project identifies four steps:

- Identify all of the data types that are to be handled by the system (genes, proteins, etc.)
- Provide high throughput technologies such as micro-arrays used for analysis of complex biological systems
- Generate a global model that provides a biological, mathematical and computational representation of the complex biological system.
- Model analysis to provide a new hypothesis for testing.

A key concept in building an infrastructure such as this is scalability. Each component needs to be scalable in its biological objects, processes and intermediate interfaces amongst the components [53].

2.4.12 Multidisciplinary Application Discussion

Justification of Multidisciplinary processes used to augment Systems Biology and the yeast processing lab have been detailed above. The existing system was analyzed using a multi-tiered architectural approach used in software engineering. The Presentation Tier described the possible visual aids for the user, while the Application Tier described modifications and adaptations to the existing system (algorithms, software) which enhance the quality and efficiency of generating and sharing the data. Additionally, a future enhancement should be to develop a specific camera model to handle all of the data collection errors described above. The data layer was analyzed for standardization to enhance the sharing of the data [48].

Further work is needed in the area of accurate, precise modeling of cell proliferation kinetics from time-lapse imaging and automated image analysis of agar yeast culture arrays [54]. One area is in data collection. There is a need to create a method to correct agar plate data collections for environment induced errors due to collector and other environment conditions. This could be achieved by creating an imagery distortion correction processing algorithm and associate calibration parameter table in Java to be use in performing image correction on the data prior to quantitative systems level analysis processing of gene interactions. A second area is in creating a more rigorous experiment setup process. A third area is in the automated image analysis processing itself. Currently the software program is a complex compute intensive program that can be restructured and streamlined and the precise processing is performing very fine calculations on very crudely collected data.

2.5 Discussion

Modern systems require modern thinking. Multidisciplinary Systems Engineering provides a new approach and paradigm for Systems Engineering organizations; a necessary shift to take System of Systems design into the future. Even though many organizations recognize that Systems Engineering is inherently a multidisciplinary approach, most do not really embrace the concept and train their engineers to think along multidisciplinary lines. Each of the authors, having spent decades designing, architecting, implementing, and delivering large-scale system have come to appreciate the need for this book. The case study provided in Chap. 2 illustrated one example of the multidisciplinary approach as applied the study of Biology. MDSE's emphasis on bringing all disciplines to the table at the same time holds the promise of more robust System of Systems architectures, holding the possibilities of improvement in achieving overall goals and constraints placed on modern system designs and a reduction in the variability created by designing system disciplines separately. Recognizing that MDSE is inherently a feedback driven process, and not being pushed by "just get it done" attitudes will, in the end, produce architectures that better meet the demands of ever more challenging system designs.

Question to think about:
Is there any inherit difference in MDSE between a traditional information management system and the Systems Biology problem?

http://www.springer.com/978-3-319-22397-1

Multidisciplinary Systems Engineering
Architecting the Design Process
Crowder, J.A.; Carbone, J.N.; Demijohn, R.
2016, XXIII, 297 p. 105 illus., 22 illus. in color.,
Hardcover
ISBN: 978-3-319-22397-1