

Using Multimedia Domain Specific Techniques

Dan Port, Ebru Dincel

Center for Software Engineering
Henry Salvatori Computer Science SAL 320
University of Southern California
Los Angeles, CA 90089-0781
{dport, edincel}@sunset.usc.edu

Experiences Using Domain Specific Techniques Within Multimedia Software Engineering

ABSTRACT

Domain Specific Techniques take advantage of the commonalities among applications developed within a certain domain. They are known to improve quality and productivity by incorporating domain knowledge and previous project experiences and promote reuse. This paper describes six domain specific software engineering techniques for developing multimedia applications within the digital library domain. We provide examples of each technique from several projects in which they were used, how the techniques are used within general software engineering practice (in particular MBASE), how the techniques address some of the particular challenges multimedia software engineering, and the positive impacts we have measured resulting from their use within a graduate level software engineering course.

1. Introduction

Over the past four years we have been refining and measuring the effectiveness of introducing six software engineering techniques specifically oriented to deal with developing multimedia applications within a digital library domain. This is done primarily within our 2-semester, team based, real-client project course (CS577) that is a core course in the USC MSCS-SE degree program. The library information services division (known as ISD) is the primary source for project customers. Generally the projects fall within ISD's digital library initiatives with such as managing digital access to specialized library collections of books, music, rare manuscripts, and art. The projects tend to be multimedia applications with sophisticated distributed (usually web-based) user interfaces utilizing a variety of consumer off the shelf (COTS) products and often must integrate with legacy systems such as SIRSI. To date student teams have developed over 57 multimedia systems. As a result, we have amassed a significant amount of experience in developing multimedia applications with a wide variety of developers, customers, and project considerations.

The course provides general software engineering guidance through our Model Based [System] Architecting and Software Engineering (MBASE) framework [Boehm 1998c; Boehm and Port 1999d]. The MBASE approach is based on the view that a narrow focus on technical aspects of system architecture is wholly insufficient to promote project success. Instead MBASE advocates a holistic treatment of issues in four dimensions: product (e.g., domain model, architecture, requirements, code); process (e.g., tasks, activities, milestones); property (e.g., cost, schedule, performance); and success (stakeholder satisfaction, e.g., business case, legal requirements). The idea is that in each dimension, activities and expectations are guided by models, and that all of the selected models have to be well matched. The approach emphasizes model mismatch as a source of failure, model reconciliation as a key activity, and model synergy as a source of advantage. The approach provides guidance for identifying and resolving conflicts iteratively over the course of a project. The Win-Win spiral model is used to protect stakeholders' interests within dynamically evolving models. Success criteria, requirements, processes, and other elements are continually adjusted and refined throughout the project. MBASE is further elaborated in section 5 where we describe how it is used as a framework for the evolution of the effectiveness of our domain specific techniques.

While MBASE provides a basis for relatively inexperienced developers to meet the typical demands of project development, multimedia applications present development challenges that require additional, more specific guidance. Some of these are the heavy dependency on COTS, a lack of standards, and a "two cultures" problem [Snow 1959] where customers with extensive, but specialized domain expertise may have unfeasible technical expectations. To this end we have evolved a number of techniques specifically for the digital library domain that directly address these challenges and have measured a significant improvement with respect to early project success through their use. While our techniques may be useful outside the digital library domain, they are not general software engineering techniques. The primary distinction from general software engineering techniques (such as Schedule as an Independent Variable, risk management,

or object orientation) is that our techniques directly address and are based on the digital library domain. For an example of this, note that in Table 1 and Table 2 the entries are organized around digital library “sub-domains”. Informally a technique that is dependent on or utilizes directly a particular application domain is a *domain specific* technique. Although such techniques may be based on general principles independent of any application domain, they are typically “empty” unless expressed within the parameters of a specific application area. It is not uncommon for a technique to apply (analogously) to multiple domains and tempting to assume the existence of a general principle. Indeed this may be so, however great care and consideration must be applied when abstracting from specific domains.

We will not be concerned with such efforts for the moment and be content to describe six domain specific software engineering techniques that have been introduced into our CS577 course for developing multimedia digital library applications. They are:

1. **Sub-domain Identification:** given a project description, identify a likely sub-domain from a list of common digital library sub-domains.
2. **Pre-architecture Selection:** validate and specialize the top-level architecture associated with a chosen sub-domain.
3. **Initial Requirements Mapping:** organize and refine requirements with respect to the sub-domain taxonomy.
4. **Top-N Risk Identification:** identify top risk factors for sub-domain within top risks for domain.
5. **Simplifier and Complicator Analysis:** analyze and specialize tradeoff issues associated with sub-domain.
6. **Experience Factory:** draw from previous project experiences within sub-domain.

We provide examples from successfully completed projects for each technique, illustrate how the process of how the techniques are used within a general software engineering practice (i.e. MBASE), discuss how the techniques address particular challenges of digital library projects, and measure the positive impacts we have measured resulting from their use within the CS577 course.

2. Challenges of Multimedia Software Engineering

A multimedia system is a system capable of integrating and processing multimedia data by utilizing both hardware (capture/storage/display devices) and software (network related). Developing such systems present many challenges and issues [Dan et al. 1998], some of which will be briefly discussed in this section. The six techniques listed above were developed in part to address such challenges by leveraging a high degree of domain knowledge and experiences.

2.1. Media Integration and Interactivity

Multimedia systems are said to have certain basic characteristics. They have to give an *integrated* presentation of a variety of media (audio, video, images, data) simultaneously. The presentation to the user may involve both spatial (virtual or physical) and temporal organization. The temporal and spatial relationships between many forms of media need to be managed properly. This includes media representation and presentation, retrieval

and storage of data, as well as the sequencing and synchronizing media elements. Each media type presents its own set of challenges such as streaming video over limited bandwidth, manipulation of large images within a browser, synchronizing text and audio, or simply managing, multiple graphics formats. While there are many workable approaches to these challenges, considered as integrated media applications, the problems of integrated applications are naturally more complex as compared to standalone, homogeneous media applications. Part of the problem is a lack of standards where the adage “Standards, I have many of them” seems to apply all too often. There are few universally accepted multimedia standards. The few that do exist are volatile and only are just starting to be deployed today, although significant efforts in this area have started to emerge [AHDS 2001].

The typical multimedia system also has to be *interactive*. However, due to their size and limited bandwidth, the transfer of these media objects to clients from a repository might encounter unacceptable latency or require large client memory buffers. The overhead caused by storage, transfer, and processing must be minimized by complex resource optimization techniques taking into account network, server, operating system, and processing power constraints.

2.2. Rapid Application Development (RAD) Challenges

Like many other software applications, multimedia systems are built with the intent of delivering a product as quickly as possible. This may be due to the need of acquiring market share or very limited budget and schedule (as with our CS577 projects). Multimedia applications are particularly susceptible to these issues and it is relatively easy to get something working quickly, but not working completely or at an acceptable level of service. The pressure to reduce delivery schedule has evolved itself into a popular process model called Rapid Application Development (RAD). However, without a disciplined approach accompanying RAD, the systems are destined to suffer quality problems such as security vulnerabilities, inability to scale, expensive and unrealistic support expectations on end users and management. If RAD has to be used for multimedia applications, it is important to choose the right kind backed up with a disciplined approach [Boehm 1999a].

2.3. Lack of Multimedia Standards

Due to a lack of standardization, diverse formats for each type of media exist. This, in return, decreases the potential for interchangeability of elements, integration of heterogeneous multimedia systems, and support of user-friendly tools to handle various multimedia design applications. For example, there is no file format standard to allow for streaming compression and decompression in real-time.

Another characteristic of multimedia systems is that the information being handled must be represented digitally, often being translated from an analog source. Unfortunately, translation is costly (in effort) and the resulting multimedia objects require a large amount of storage. Therefore, issues of efficient digitizing, storage management, and retrieval often play a critical role when designing multimedia systems. Currently there seems to be no general way to address these problems as techniques on efficient retrieval

based on distribution (e.g. load balancing), compression, caching, and network utilization by buffering are still active research areas [Serpanos et al.1998; Dan et al.1997].

2.4. Heavy Dependency on COTS

In today's quick-to-market applications there is an intense desire to implement functionality through use of COTS products. There is a large, increasing, and unstable base of COTS products. RAD tends to dictate the use of such components to achieve complex required functionality within a demanding schedule. Demanding multimedia capabilities such as streaming video requires highly specialized technical expertise and is often infeasible to develop for a single application. Although using COTS is attractive as a means to save development effort, it comes with a range of new risks and tradeoff considerations such as architectural mismatches, requirements infeasibility, and late development cycle breakage. The technical pitfalls and remedies of utilizing COTS have been discussed by various architectural research groups such as [Garlan et al. 1994] and remain an active research area. There are many more non-technical challenges such as legacy system integration and backward compatibility (or transitioning), license fees, vendor dependency, etc. [Boehm and Abts 1999c] that further complicate multimedia development.

2.5. Level of Service Issues

When it comes to making decisions about multimedia system design, non-functional requirements often play a critical role compared to non-multimedia applications. Users generally unaware of possible technical limitations often have high level of service expectations. From a user's perspective, a system has to perform fast, accurately, and naturally. The interface must be easy to use yet provide a large variety of sophisticated features. From a system perspective, it has to be secure, available, extensible, robust, and a host of other difficult to achieve "ilities". The aggregate set of expectations are often in conflict or are unfeasible to achieve. For example, a customer specified a web based real-time video playback capability for the USC student film school archives. Within USC's high bandwidth local network with a very small numbers of users this was feasible. However, the customer expected equivalent performance for a large number of non-local network users. In operation for only a few days the system rapidly bogged down and was subsequently closed to non-local network users.

2.6. Specialized Domain Knowledge

A myriad of non-technically focused industries such as entertainment, publishing, advertising, and library information systems depend on multimedia for managing and presenting information using multiple media types. Often deep and specialized knowledge of the activities, processes, and demands of the particular organization's domain in which a system is to be used is required to construct effective applications. For example, digital library applications necessitate an understanding of library systems. Our student developers tend not to have library science degrees, yet often must resolve (or at least understand well) library science problems in order to address their customer needs.

In addition developers may incur technically challenging problems as a result of customer demands with respect to their domain specific problems. Some examples are Natural Language Processing for search capabilities, Image Processing or Pattern Recognition for content-based image retrieval [Rui et al. 1999], and Speech Synthesis and Recognition for data output and input. Many of these areas have major open problems and are still actively being researched. As a result, multimedia applications may be limited within such capabilities, yet the customer's expectations are not. As an example of this, a customer specified a natural language interface analogous to "Ask Jeeves" for an otherwise simple query system. The project was cancelled after the natural language interface caused a factor-of-5 overrun in project budget and schedule as the developers struggled to understand and scope down the range of possible queries.

2.7. Reuse Challenges

Developing high quality reusable multi-media software components is a challenge due to the dynamic market with a growing range of multimedia formats, lack of proper standardization, competing COTS packages, and unrealistic non-functional requirements (such as portable, available, 100% reliable systems). For such challenges, one needs to adopt a systematic reuse approach [Schmidt 1999] over an ad-hoc "I'll reuse it if I remember it" approach. Adopting an object-oriented or component oriented technology does not automatically provide reuse, it must be planned for in the early stages of the life cycle. In addition to technical challenges, there are myriad of management concerns such as documented in [Smart et al. 1988].

2.8 Modeling and Specification

Another difficult problem is modeling and specification of multimedia applications. This is mainly due to our lack of experience in the general design of such systems and effective means to communicate them. There is some promising work in this area. One such is the exploitation of software architectures for the specification of systems through an object oriented architecture description language [Tsai99]. Another is the identification of design patterns of for multimedia design [Lyar98]. Both these approaches attempt to provide some form of system modeling outside of implementation.

For multimedia applications, levels of service issues are (e.g. performance, security) crucial. Specialization of this by itself brings many different dimensions; specification of time, security, reliability, etc. Specifications of these qualities often pose a challenge, particularly with modeling complex multimedia timing logic. Additionally, there are issues of consistency across different aspects and composition to analyze the overall specification and for performance validation and analysis [Blair].

Owing to the above issues, there is a strong need to adapt standard software engineering techniques (such as specification, modeling, verification, and validation) to multimedia applications.

3. Domain Specific Software Engineering Techniques

A goal of Software Engineering as a discipline is to achieve "surprise free" software that satisfies user's needs and is delivered on time and within budget. The issues and

challenges of this are approached in many different ways, one of them through domain engineering and domain specific techniques [Tracz 1995, Hayes-Roth et al. 1994, Gamma et al. 1994].

A *Domain* is defined as an area of knowledge or activity characterized by a set of concepts and terminology understood by the practitioners in that area [Booch et al. 1999]. A *Sub-domain* is a more specialized area of concern that specifies particular characteristics within a given parent domain. For the techniques that will be described here the domain is digital library systems. Our techniques will all be expressed in terms of sub-domains for this domain. Domain specific techniques improve quality and productivity by incorporating previous project experiences and domain knowledge. As such, domain specific techniques are expressed in terms of domain elements, whereas general software engineering techniques are expressed in terms of a generic project. For example, selecting a pre-architecture necessitates having an established list of pre-architectures to choose from, whereas using schedule as an independent variable [Boehm et al. 2001] addresses project elements independent of any particular domain. Although domain specific techniques might incur some overhead initially, their benefits far outweigh their cost over the various activities within the life cycle, especially in design. Domain specific techniques are semi-iterative and are extended and refined each time they are used. Eventually refinement is expected to stabilize to the point where only new project experience references are added. This will be exemplified in the sections below.

Domain Specific Software Engineering is gaining importance recently well after the need was recognized [SEI 1990, Tracz 1994]. These processes and architectures provide an organizational structure customized to a family of applications such as embedded systems [Dager 2000], medical imaging product family [Pronk 1999], avionics [Sharp 2000] and can provide tremendous benefit for relatively little effort. Some of these benefits will be listed next.

3.1. Benefits of domain specific software engineering

Domain Specific Techniques generally take advantage of the commonalities among applications developed within a certain domain. Some of the benefits to using domain specific knowledge when designing a member of an application family can be summarized as follows:

1. Reduced costs and risks, since the analysis, design and development space is bounded.
2. Higher reuse as variability is contained within in a specific domain. Provides reuse on multiple levels such as architecture, requirements, COTS mappings, development process, documentation, etc. as compared to simple code reuse.
3. Improved predictability for strategic, organizational level decisions based on accumulated domain specific experiences (e.g. common pitfalls, typical risks).
4. Reduced time-to-market and known focus areas for RAD by leveraging common domain patterns (within analysis and design).
5. Higher maintainability due to mature documentation and previous domain experience.

6. Greater identification and use of standards by matching to domain needs and prior use in domain.
7. Better, more meaningful performance modeling, analysis, and testing by utilizing established domain specific baselines and comparisons.
8. Rapid evolution of skilled staff that requires less training time – smaller, tangible areas to focus on are more easily digested and have more identifiable relevance.
9. More effective, rapid communication between project stakeholders owing to a reduced, shared domain scope.
10. Encourages “product line” engineering practices.

4. Domain specific techniques for library multimedia systems

Here we detail some useful domain specific techniques for addressing the multimedia challenges of digital library systems. For each technique, we have provided some background, a brief explanation, the steps that outline the technique and an illustrated example for how it was used for a sample multimedia project completed in our cs577 course. In Section 5, we will discuss how these techniques evolved and where they fit into a general software engineering framework. Generally the techniques are described as sequence of steps and decision. Each technique description will conclude with an example of applying the technique taken from a CS577 project.

4.1. Sub-domain Identification

The purpose of this technique is to identify a pre-existing sub-domain classification for the application. This classification will be used as a basis for using most of the other domain specific techniques and as such it is critical to be judicious in its application.

Performing Sub-domain Identification

We have compiled a list of sub-domain descriptions based on previous projects. Given a project description, first choose the closest matching sub-domain from Table 1. Next, provide a rationale for the classification choice. Each sub-domain from Table 1 has an elaboration containing possible stakeholder roles and responsibilities in addition to system boundaries (see **Figure 1** for an example). Adapt this as needed to the project. In the event that no sub-domain matches well, perform the following activities:

1. Validate that the application falls within the digital library domain by attempting to achieve a consensus of stakeholders, particularly domain experts. If no consensus can be reached, the digital multimedia library domain specific techniques do not apply to this project. Explore other domains.
2. Attempt to extend an existing sub-domain to create a satisfactory match. If you cannot easily extend an existing sub-domain, create a new sub-domain description.
3. Validate that the new sub-domain falls within the digital library domain as per activity 1. Have this added to the sub-domain list if successful.

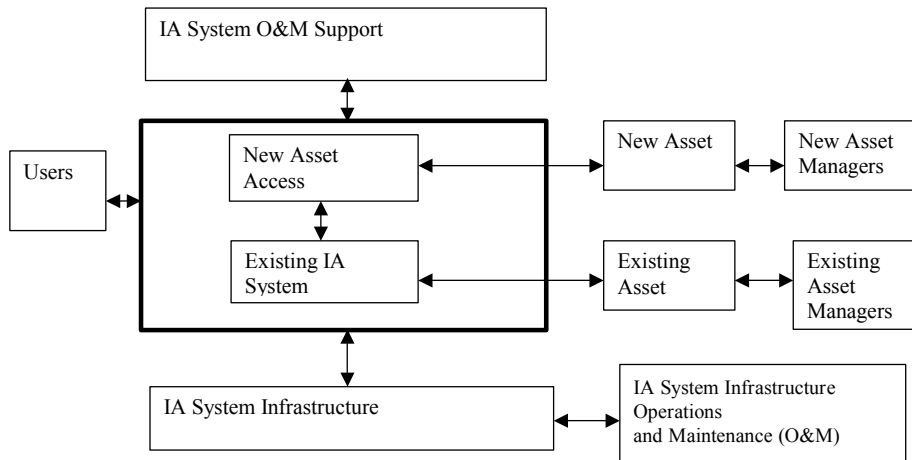
Table 1. Sub-domain descriptions

Sub-domain	Description
1) Multimedia Archive	Provides a user interface for a collection of

	multimedia content)
2) Selective Dissemination of Information	Distributes content according to user interests and selection rules
3) Data Analysis	Process data from multiple sources and reporting
4) Activity Monitoring and Control	Implements agents that invoke policy in response to events and provides status reporting and activities management
5) Automated Reference Services	Provides a uniform source for frequently requested, relatively static information
6) Data Migration	Aggregates and converts data from one format to another
7) Virtual Access to Special Collection	Provides a virtual environment that implements access policies for multimedia content
8) COTS Package Extension	Externally adds capabilities to COTS
9) Distributed Borrowing	Provides an interface to manage non-digital assets and implements a borrowing policy
10) Interactive Communication	Provides interactive user interface for rich media access

System Block Diagram:

This diagram shows the usual block diagram for extensions providing access to new information archive assets from an existing information archive (IA) System:



The system boundary (marked by thicker bound) focuses on the automated applications portion of the operation, and includes such entities as users, operators, maintainers, assets, and infrastructure (campus networks, etc.) as part of the system environment. The diagram abstracts out such capabilities as asset catalogues and direct user access to O&M support and asset managers.

Some Stakeholder Roles and Responsibilities

- Asset Managers. Furnish and update asset content and catalogue descriptors. Ensure access to assets. Provide accessibility status information. Ensure asset-base recoverability. Support problem analysis, explanation, training, instrumentation, and operations analysis.
- Operators. Maintain high level of system performance and availability. Accommodate asset and services growth and change. Protect stakeholder privacy and intellectual property rights. Support problem analysis, explanation, training, instrumentation, and operations analysis.
- Users. Obtain training. Access system. Query and browse assets. Import and operate on assets. Establish, populate, update, and access asset-related user files. Comply with system policies. Provide feedback on usage.
- Application Software Maintainer. Perform corrective, adaptive and perfective (tuning, restructuring) maintenance on software. Analyze and support prioritization of proposed changes. Plan design, develop, and verify selected changes. Support problem analysis, explanation, training, instrumentation, and operations analysis.
- Service providers (e.g. network, database, or facilities management services). Similar roles and responsibilities to Asset Managers.

Figure 1 Sub-domain Model.

Example use of Sub-domain Identification:

The project description for the Asian Film Database is as follows. Representatives from the film scholars and film industries of China, India, Japan, Korea, and Taiwan want to jointly create an Asian film database containing basic information as well as film clips for each of their cinema cultures. Their film production ranges from 10 per year from Taiwan to 800 films per year from India. The information must be translated and made available in four languages: English, Chinese, Japanese, and Korean. The database will

ultimately be housed and maintained by the Information Services Division (ISD). The Asian Film database will allow film critics, film programmers, film scholars, and independent filmmakers to choose the language they wish to view the database web pages in, then browse and/or search for information by film title, language, genre, keywords, and whether film clips or full-text film review articles are included in the database. Basic information about all films as well as additional information (such as translations of full-text film reviews, still images from the movies, and digitized video clips of film excerpts) for selected films will be included. The database must be easy to use and not require the latest technology (such as esoteric plug-ins), so that as many people as possible can access the information. In addition to the general public, there are several primary user groups: film critics for newspapers and magazines, film programmers who schedule film screenings at film festivals or film archives, and film scholars at research universities.

More information about selected stakeholders is provided as a result of the stakeholder meetings:

Film Critics: Film critics need background and contextual information about Asian films in a very short time frame. A film critic in the U.S. has very little time to view an Asian film and write a review (could be 24 hours). Information must be easy to access and quick to read. Perhaps a two-fold view (brief synopsis and more detailed information) can be used.

Film Programmer: Film programmers need authoritative reference information to help them select which movies to screen and current contact information to help them acquire the selected film for their program. Fans generate most U.S. movie websites and lack well researched reference information on Asian films. Getting a film to screen is difficult because contact information is not always present, correct, or updated.

Film Scholar: Multilingual capability is important so that users can read material published in the home country of the film (for example, read Korean film reviews of Korean films, and also read the Korean film review in Chinese, English, or Japanese). It is also important to know if a film has subtitled versions, and if so, in what language(s).

Choice of sub-domain rationale:

Based on past experiences and the meeting discussions, there was consensus among stakeholders that the application is an instance of a multimedia archive. The main proposed capabilities included storage of the massive multimedia assets in a structured format through an Internet-based user interface so that the information can be browsed as desired. There is a close correspondence of the system description with the system block diagram in **Figure 1**. Although unstructured, there is currently an existing assets of Asian films managed by library staff. New assets (more diversity of Asian films) will be integrated as a result of archiving these assets. Users such as film critics, programmers, scholars will query and browse the system. Operator and service provider roles will be covered by ISD.

4.2. Pre-architectural Selection

This technique aims at giving a quick lead on the application architecture by providing a high-level block diagram of a generic architecture associated with the selected sub-domain.

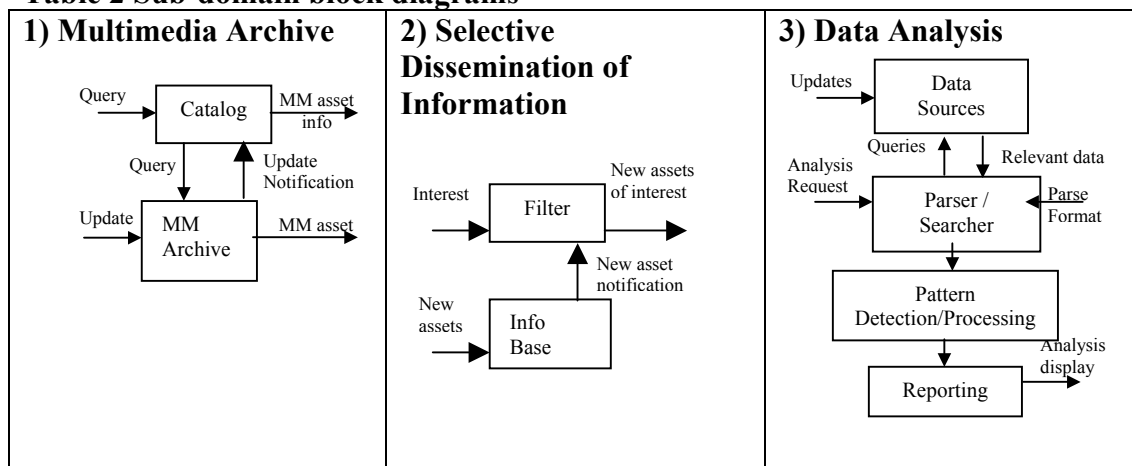
Performing Pre-architectural Selection

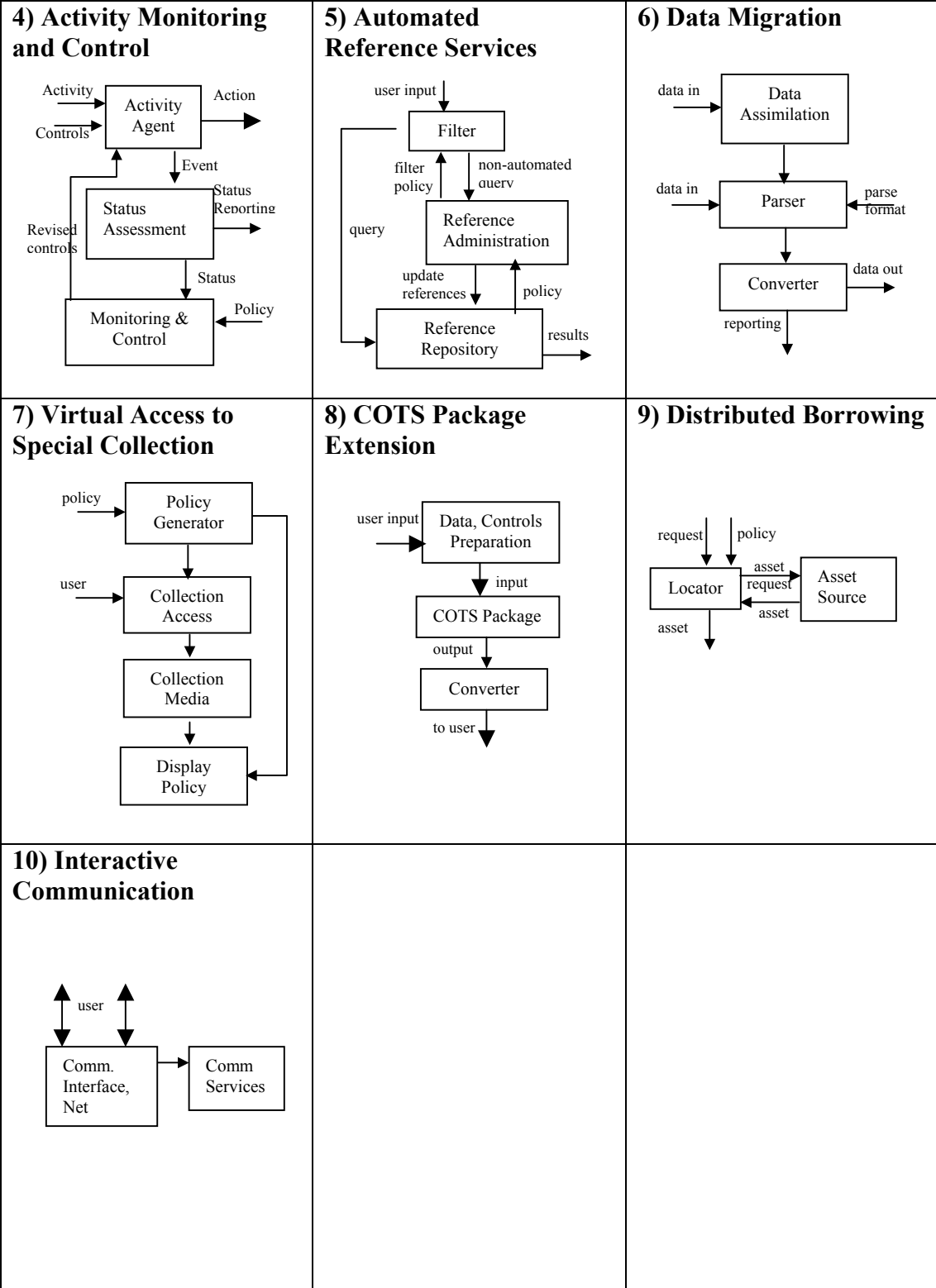
The activities are as follows:

1. Based on the sub-domain classification, locate the associated block diagram from Table 2.
2. Specialize this generic block diagram to the application and provide rationale for the specializations made.

The resulting specific block diagram for the proposed system should be derived from the generic block diagram. If specializing this diagram for the selected sub-domain proves difficult, it may indicate an inappropriate sub-domain has been chosen. The specific block diagram should serve as a high level starting point to the application architecture. If the sub-domain was newly added, then an appropriate block diagram for the new sub-domain should be added to Table 2. Some further explanation of Table 2 is in order. Consider a project classified in the Multimedia Archive sub-domain. The first cell in Table 2 provides a typical top-level block diagram for a multimedia archive. It indicates that users can query the Catalog for multimedia asset information. When they find assets of interest, they can query the multimedia Archive to receive electronic copies of the assets. Administrators can update the Archive and transmit update notifications for the Catalog.

Table 2 Sub-domain block diagrams





Example of Pre-architectural Selection: From Table 2, we locate the first sub-domain block diagram since the Asian Film Database was classified earlier as a kind of Multimedia Archive. The film catalog consists of roughly 1200 films per year produced in China, India, Japan, Korea and Taiwan. Through the catalog, the Asian Film Database will allow film critics, film programmers, film scholars, and independent filmmakers to choose the language they wish to view the database web pages in, then browse/retrieve and/or query for information by film title, language, genre, keywords. The archive will enable updating and displaying of film clips/stills. Meanwhile, as new film clips/stills are updated, the related asset information will be updated in the catalog. Based on the needs of this particular application, here is a specialization of the Multimedia Archive block diagram from Table 2:

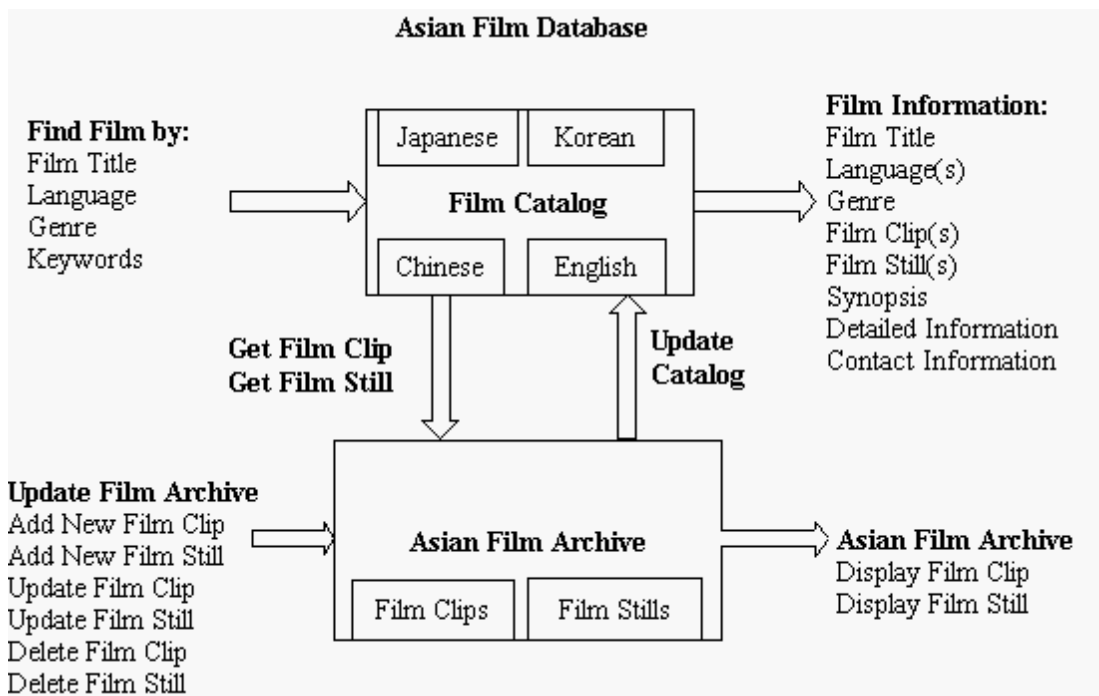


Figure 2. Specific block diagram for Asian Film Database.

4.3. Initial Requirements Mapping

The stakeholder WinWin approach to requirements engineering [Boehm et al. 1994] provides a way to help reconcile stakeholder expectations and system capabilities. For example a hard-to-achieve customer or user Win Condition will conflict with the developer's Win Condition to minimize the risk of delivering an acceptable product within budget and schedule. In the WinWin approach, this conflict is identified as an Issue needing resolution before the formal WinWin equilibrium state (all Win Conditions covered by Agreements and no unresolved Issues) can be achieved [Lee 1996]. Eventually WinWin negotiations lead to requirements.

However, our experience with the WinWin approach has taught us that this reconciliation is much smoother if the majority of win conditions are expressed early on. The difficulty with this is that stakeholders may not recognize all their areas of concern from the outset.

This is often exacerbated as stakeholders may only express concerns in reaction to something raised rather than independently. This frustrates and delays the requirements process, leading to inaccuracies and critical oversights.

Performing Initial Requirements Mapping

This technique provides a more accurate and rapid input to the requirements engineering processes. Table 3 provides an example sub-domain WinWin taxonomy for Multimedia Archives. There are sub-domain taxonomies for each entry listed in Table 2. The steps to follow are straightforward:

1. Customize your sub-domain's WinWin taxonomy [Boehm et al. 1997] by adding items specific to your project and removing irrelevant items.
2. Map the initial requirements from the project (if known) to the customized taxonomy.

Table 3. WinWin Taxonomy for Multimedia Archive Domain.

1. Operational Modes
1.1 Classes of Service (research, education, general public)
1.2 Training
1.3 Graceful Degradation and Recovery
2. Capabilities
2.1 Media Handled
2.1.1 Static (text, images, graphics, etc.)
2.1.2 Dynamic (audio, video, animation, etc.)
2.2 Media Operations
2.2.1 Query, Browse
2.2.2 Access
2.2.3 Text Operations (find, reformat, etc.)
2.2.4 Image Operations (zoom in/out, translate/rotate, etc.)
2.2.5 Audio Operations (volume, balance, forward/reverse, etc.)
2.2.6 Video/Animation Operations (speedup/slowdown, etc.)
2.2.7 Adaptation (cut, copy, paste, superimpose, etc.)
2.2.8 File Operations (save, recall, print, record, etc.)
2.2.9 User Controls
2.3 Help
2.4 Administration
2.4.1 User Account Management
2.4.2 Usage Monitoring and Analysis
3. Interfaces
3.1 Infrastructure (SIRSI, UCS, etc.)
3.2 Media Providers
3.3 Operators
4. Quality Attributes
4.1 Assurance
4.1.1 Reliability/Availability
4.1.2 Privacy/Access Control
4.2 Interoperability
4.3 Usability
4.4 Performance
4.5 Evolvability/Portability
4.6 Cost/Schedule
4.7 Reusability
5. Environment and Data
5.1 Workload Characterization
6. Evolution
6.1 Capability Evolution
6.2 Interface and Technology Evolution
6.3 Environment and Workload Evolution

As a result of this technique, the WinWin taxonomy should have been refined and eventually populated with the initial requirements by a requirements gathering method.

Example Initial Requirements Mapping: Table 4 shows a snippet of a customized WinWin taxonomy for the Asian Film Database. Taxonomy items in *italics* are additions and strikeouts are deletions. There are specific requirements on privacy and access control (4.1.2) since such systems require legal implications, e.g. copyright, or want to prevent unauthorized access due to the inappropriate or confidential material. For this system, the interoperability requirement (4.2) is taken out since there is no need of another system’s exchanging or using information with this system within this project. All the capabilities will be managed through a standalone database application and an Internet user interface. Under usability (4.3) there are specific requirements. The first one is an Ease of Learning requirement (4.3.1), which is needed to keep the expected learning or training time low due to the insufficient digital multimedia systems expertise. In this case the approach will be to implement content specific help (4.3.1.1). Ease of Use (4.3.2) is specified to promote the system’s comprehensive, well organized/researched information on multimedia assets to a degree the system is easy to use. There are performance requirements (4.1) due to the immediate need of film critics to retrieve background and contextual information about Asian films. The Reusability (4.7) requirement is taken out due to the architectural/financial decisions based on organizational policy.

Table 4. (Partial) Customized Multimedia Database taxonomy for Asian Film.

4. Quality Attributes
4.1 Assurance
4.1.1 Reliability/Availability
4.1.2 Privacy/Access Control
4.1.2.1 <i>Integrity</i>
4.1.2.2 <i>Audit</i>
4.2 Interoperability
4.3 Usability
4.3.1 <i>Ease of Learning</i>
4.3.1.1 <i>Context Specific Help</i>
4.3.2 <i>Ease of Use</i>
4.3.3 <i>Help Requirements</i>
4.1 Performance
4.1.1 <i>Network Bandwidth Usage</i>
4.1.2 <i>Workload</i>
4.5 Evolvability/Portability
4.6 Cost/Schedule
4.7 Reusability

After the taxonomy is customized, activity 2 is mapping the initial requirements to the taxonomy. For the sake of brevity, we will provide only one such example.

Requirement ID: R-02.

Type: Ease of Learning (Taxonomy 4.3.1).

Title: Expected learning time of system for Film Critics.

Description: Due to their heavy schedule, film critics cannot afford to spend significant amount of time for learning how to use the system.

Measurable: The average film critic should be able to view a brief synopsis within a minute without resorting to the manual.

Achievable: By introducing one click user interface operations for simple, typical activities.

Relevant: Higher usability means higher acceptance and therefore recognition of Asian Film Industry.

Specific:

4.4. Top-N Risk Identification

A multimedia software project that does not consider proactively sources of risk will run into serious difficulties if exposed to such risks. While it is unfeasible to identify and plan contingencies for all possible risks, an effective starting point is the well-known list of the 10 most frequent software risks from [Boehm 1989].

Table 5 is a specialization of this list that shows the more likely to occur risks within the Multimedia Archive sub-domain. This list helps developers continuously assess risk throughout the project, and make all stakeholders aware of what to expect. In addition to this, developers have found it easier to prioritize risks by assessing risk exposure (probability of loss times magnitude of loss). Each week, the risks are reassessed to see if its priority has changed or to determine how much progress had been made in mitigating them. For example a common mitigation technique is the “design to schedule” strategy in which a feasible core set of capabilities is implemented, and optional features to be implemented as schedule permits. Based on experience we’ve compiled a set of common risks particular to each sub-domain and suggested mitigation strategies. It is remarkable how much the common risks vary between the sub-domains. For example, one might expect that the “excessive bells and whistles” risk would appear often in the Multimedia Archive sub-domain, but it actually appears more frequently in “Virtual Access to Special Collection”.

Performing Top-N Risk Identification

The activities for performing top-n risk identification are as follows:

1. Using the top-n risks for your sub-domain (such as in Table 5) as a guideline, identify the specific risks for your project (including the domain specific risks as well as the general project risks).
2. Perform risk assessment as per [Boehm 1989]. Rank your projects top-n risks and compare to previous rankings, track number of weeks risk remains on top-n, describe current state of risk resolution.
3. Apply the specific top n risk item list on a regular basis by highlighting these in regular project reviews, focusing on new entries and slow progress items. .

Table 5. Top-N Risk Item List for Multimedia Archive Sub-domain.

Source of Risk	Risk Management Techniques
1) Performance risks for image/video distribution systems	Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
2) Finding a proper search engine	Software evaluation of search engines, prototyping, experience factory investigation

3) Legacy software integration	Reengineering; code analysis; interviewing; wrappers; incremental deconstruction
4) Information Systems Division limitations	Interviewing, alternative analysis, benchmarking
5) Effective indexing and access of assets	Technical analysis, prototyping, modeling, tuning
6) Digitizing complex/fragile assets	Effort/schedule estimation, equipment analysis, benchmarking, instrumentation; tuning

Example of Top-N Risk Identification: By using Table 5 as a guideline for Asian Film Database Project, perform a risk assessment, choose mitigation strategies, and track risk resolution progress. Below in Table 6, we show a top-n risk tracking for a specific week. The highest risk is in COTS availability. It is shown that that this was the highest risk last week as well, and has been tracked for a total of five weeks. The current state of resolution of this risk is finding out whether there is an appropriate COTS package to do the multilingual natural language processing that is easily integrated, low cost, and efficient and accurate enough. The results can be obtained through COTS surveys, and the financial problems can be overcome obtaining academic discounts. After preparing the project specific rankings, these need to be brought up during subsequent stakeholder meetings updated accordingly for the next week's tracking list.

Table 6. Top N Risk Tracking for Asian Film Database.

Risk Items	Weekly Ranking			Risk Resolution Progress
	Current	Previous	# Weeks	
COTS availability	1	1	5	Performed multilingual natural language processing COTS survey. Applied for academic discount.
Personal shortfalls	2	4	4	Two members are not available for this week. Other team members will put extra effort this week.
Effective access for Asian Films	3	3	3	Performed an initial analysis on efficient storage management techniques. Will need to do look more into this matter in the following weeks.
Digitizing Indian Films	4	2	3	Due to its fast growing rate, there is a need for additional effort to digitize Indian films. Ask staff to assign a work study for this task.
Storage space	5	5	2	Plan for additional budget for more storage space to store. Discuss alternatives, tradeoffs with weekly customer reviews.

4.5. Simplifier & Complicator Analysis

This technique came out of the intent to reduce frequency of the initial project review shortfalls. Many of these shortfalls were due to unmanaged customer expectations, known as “Two Cultures” problem. These are not adequately addressed by the risk assessment efforts, and [Snow 1959] found that science and technology policymaking was extremely difficult because it required the combined expertise of both scientists and politicians, whose two cultures had little understanding of each other's principles and practices.

Our approach involved the following main steps to manage digital library project stakeholders' expectations:

- For each of the major digital library application sub-domains, we developed a characterization of the sub-domain, including lists of simplifiers and complicators (S&C's) that would make the applications easier or harder to implement.
- We provided the S&C's to Library clients, with explanations in librarian terms by the Library project coordinators.
- We highlighted the S&C's in class lectures on risk management.
- We made the S&C's the subject of an early student homework exercise: to pick two candidate projects from previous years, to identify their sub-domains, and analyze their S&C's.

Below in Table 7 are the S&C's for digital library sub-domain. An explanation of the initial row (Multimedia Archive) is given following the tables.

Table 7. Developer side simplifiers and complicators.

Simplifiers	Complicators
1) Multimedia Archive	
<ul style="list-style-type: none"> • Use standard query languages • Use standard or COTS search engine • Uniform media formats 	<ul style="list-style-type: none"> • Natural language processing • Automated cataloging or indexing • Digitizing large archives • Digitizing complex or fragile artifacts • Rapid access to large Archives • Access to heterogeneous media collections • Automated annotation, description, or meanings to digital assets • Integration of legacy systems
2) Selective Dissemination of Information	
<ul style="list-style-type: none"> • Use of existing or standard information base • Well defined distribution points • COTS notification and event processing • WWW/internet based • Restricted interests vocabulary and filtering structures • Single information base 	<ul style="list-style-type: none"> • Volatile or ill-defined interest or filtering criteria • Complex distribution • Multiple distribution formats • Heterogeneous information sources • Complex filter reasoning • Automated interest update
3) Data Analysis	
<ul style="list-style-type: none"> • Use of data analysis packages (statistics, etc.) • Implement using interpreted or script languages. • Data stored in an RDBMS • COTS reporting packages (for graphics, etc.) • Simple, consistent data formats 	<ul style="list-style-type: none"> • Natural language processing • Highly unstructured data • High degree of formatting or conversion • Computationally intensive reporting • Spatial data analysis • Complex pattern recognition
4) Activity Monitoring and Control	
<ul style="list-style-type: none"> • Standards based agent interfaces • Simple, well-defined policies • Uncoupled controls 	<ul style="list-style-type: none"> • Real time or embedded monitoring • Synchronization of monitoring & control activities • Concurrency management of activities • Distributed monitoring of activities • Natural language processing of policy • Policy learning
5) Automated Reference Services	
<ul style="list-style-type: none"> • Single repository • COTS repository • WWW interface • Restricted query vocabulary and format • Stable FAQ's 	<ul style="list-style-type: none"> • Natural language processing of query • Multiple domains • Heterogeneous repositories • Complex or volatile filter policies • Volatile FAQ's

6) Data Migration	
<ul style="list-style-type: none"> • Single data source; uniform data • Simple, stable parse formats 	<ul style="list-style-type: none"> • Heterogeneous or distributed data sources • Context sensitive conversions • Large number of exceptions • Highly relational data • Complex computational dependencies
7) Virtual Access to Special Collection	
<ul style="list-style-type: none"> • Fixed virtual architecture • Stable collection 	<ul style="list-style-type: none"> • Non-homogenous physical architecture • Immerse virtual reality (VR) • Heterogeneous and/or volatile collections
8) COTS Package Extension	
<ul style="list-style-type: none"> • Clean, well-defined API's • Single COTS package • Simple mappings of interface inputs and outputs 	<ul style="list-style-type: none"> • Dynamic API's • Natural language processing • Multiple, incompatible COTS packages • Complex exception handling • Volatile COTS packages
9) Distributed Borrowing	
<ul style="list-style-type: none"> • Homogeneous asset sources • Simple asset source interfaces • Few asset sources 	<ul style="list-style-type: none"> • Complex borrowing policies and requests • Organization politics and economics
10) Interactive Communication	
<ul style="list-style-type: none"> • Internet/WWW interface • COTS communication services 	<ul style="list-style-type: none"> • High degree of integration • Real-time • Synchronous • Concurrency • Rich media (video, voice, NLP, etc.)

The numbered rows indicate sub-domains. The first column identifies project decisions, which would generally simplify the development of the archive capability. Their use needs to be balanced with other project considerations, particularly fixed requirements. For example, some standard query languages are not very user-friendly, but it may be better to provide a user-friendly front end to a standard query system than to develop a totally new query capability. The list of simplifiers will necessarily evolve with experience. For example, some projects in 1998 experimented with the IBM Digital Library COTS package. Had it met the Information Services Division's acceptance criteria, its use would have become a major simplifier. The second column (first row) in Table 7 identifies project decisions that would generally complicate the development of a multimedia archive application. We also developed an initial list of client-side simplifiers and complicators (Table 8). These are conditions that make the library clients' job easier or harder, which are generally beyond the awareness of computer science specialists. For example, we have had some student teams add what they thought would be helpful extensions to their systems and to be totally surprised when their clients told them that the extensions made the system less useful, because they created dependencies on organizations outside the client's scope of control.

We should also mention that the two cultures situation also frequently produces pleasant surprises. These happen when the library client does not ask for capabilities because they appear too difficult, but the student teams bring these up as candidate improvements. For example, The EDGAR corporate data project presented with a financial table reformatting problem identified a number of Web-based extensions to simplify querying for the sources of the tables, and to provide links to related Web-available information such as the Web home page of the company being analyzed. The client was amazed at

how much more capability she was able to obtain in the project's short duration [Boehm et al. 1997].

Performing a Simplifier & Complicator Analysis

Manage the expectations of different stakeholders by performing the following tasks:

1. Using the list of generic developer side simplifiers and complicators in Table 7, provide specific simplifiers and complicators customized towards for the project.
2. For each simplifier/complicator, explain the overall implied risks and trade-offs (Table 9,10).
3. Select relevant client side simplifiers and complicators and specialize them to your project to help you gain an understanding of your client's perspective.
4. For each client side simplifier and complicator provide the trade-off relationships between the client side and developer simplifiers and complicators (Table 8).

As a result of this technique, developer and client side specific simplifiers and complicators should have been identified as well as the associated risks and tradeoffs. Providing stakeholders with a better mutual understanding of domain specific simplifiers and complicators will reduce over expectations from both developers and customers and the frequency of projects specifying unachievable requirements [Boehm et al. 1999b].

As a rule of thumb, you should be able to find a specific simplifier/complicator for every generic simplifier/complicator respectively. Note that for every complicator, there is an inverse simplifier. But for every simplifier, there might not be an inverse complicator.

Example of a Simplifier & Complicator Analysis: Activity 1,2 suggests providing specific simplifiers and complicators as indicated in Table 9 and Table 10 together with the risks and tradeoffs associated using Table 7 as a guide. For example, in Table 10, digitizing a large number of Asian films is listed as a specific complicator for this project. The risk is due to the enormous amount of storage needed, both initially and from the anticipated growth rate. At this point, there has been no discussion of tradeoff issues. Next, Activity 3,4 involve focusing on the client side simplifiers and complicators. This analysis is done based on stakeholder meetings, past experiences, existing documentation, organizational policies and client feedback or concern on certain matters. Table 8 shows the result of such activities with the Leavey Library (a library at USC) staff. An example simplifier is as the application falls under digital library domain, the project scope fits well given the Leavey Library staff is authorized to make decisions on all such applications within the USC Library system.

Table 8. Client-Side Simplifiers and Complicators.

Simplifiers	Complicators
Project scope fits within client's authority scope	Scope crosses organizational boundaries
Solution reduces job tedium, reduces procedural delays	Solution creates more user work, dehumanizes personal interactions
Solution reduces organizational friction, infrastructure clashes	Solution shifts power, confuses lines of authority, puts outside parties on critical path
Task-tailorable user interface	Mismatches between user interface and user tasks,

	Capabilities
COTS product features anticipate direction of growth	COTS product features evolving toward different Marketplace
	Hidden costs: licenses, data entry, conversion
	Mismatches with existing legacy-system constraints
	Single-criterion optimization: speed; correctness
	Creeping (baroque) elegance

Table 9. Asian Film Database Developer Simplifiers Analysis.

Simplifiers	Risks and Trade-offs
<p>Generic Uniform Media Formats</p> <p>Specific All video clips are stored using an open file format for video/audio (e.g., MPEG). All film stills are stored using an open image file format (e.g., JPEG). The inverse complicator is to store film clips using streaming video technologies</p>	<p>This means that we may have to convert existing digital assets or digitize the original media, which may be costly.</p> <p>A unique file format limits the user base to those who have viewers for that particular file format</p> <p>The chosen file format may not be the most efficient for the various types of media (in terms of compression rates, quality, etc...)</p>
<p>Generic Use Standard Query Languages</p> <p>Specific Organize catalog and archive relationally so that queries will be limited to standard search formats,: match exactly by value on any of the fields with or without using boolean combinations (AND, OR, NOT, etc...), or using pattern matching (SQL <i>LIKE</i> keyword)</p>	<p>May not be as effective for "discovering" assets in the archive: users must know what they're looking for, in order to search for it</p>
<p>Generic Use Standard COTS</p> <p>Specific Use a standard Relational Database Management System (RDBMS) that supports storing multi-media assets</p>	<p>A Relational Database Management System may not be most suited for archival of multi-media assets.</p> <p>A Relational Database Management System may have a high initial cost, high implementation, and high administration cost (requires specialized knowledge skills)</p>

Table 10. Asian Film Database Developer Complicator Analysis.

Complicators	Risks and Trade-offs
<p>Generic Natural Language Processing</p> <p>Specific Store the information only in one language (e.g., English) and provide dynamic translation into Chinese, Japanese and Korean The inverse simplifier is to store the same information in 4 different languages (English, Chinese, Japanese and Korean).</p>	<p>The first approach is a complex, error-prone, expensive natural language processing issue</p> <p>The second approach will require more storage space, in addition to acquiring the translations</p>
<p>Generic Digitizing Large Archives</p> <p>Specific Digitizing film clips from the entire collection of films (which grows at a very fast rate of 800 films per year for Indian films alone)</p>	<p>If each film clip requires around 10 MB, then the rate of growth of the database will be of 8GB a year (exclusive of catalog information)</p>
<p>Generic Integration of "Legacy" Systems</p>	<p>We cannot use more effective multi-media formats, which are becoming standard technologies</p>

Specific Do not require Real-Video plug-in for Web browsers to allow users to view streamed film clips	
--	--

4.6. Experience Factory

The experience factory (EF) is an infrastructure aimed at the capitalization and reuse of life cycle experiences and artifacts. This technique relies on the fact that our domain specific process experiences are recorded and project artifacts are archived. The EF establishes a process for the continual refinement and application of a domain specific knowledge base. In this it plays two roles. First as a repository of particular project experiences, and second as a methodology that establishes a continual demand for experience inputs. It is similar to, but narrower in scope to the Experience Factory method as described in [Basili et al. 1994].

For a given project, each domain specific technique as described in Sections 4.1-4.5 work within the digital library domain experience factory to provide inputs. In turn, the EF provides developers examples of previous projects, COTS utilized, domain specific design patterns and common pitfalls. The EF repository is organized according sub-domains. After a project is completed, experience data is added to the repository as indicated in Table 11.

Table 11. Experience Factory Repository CS577 Projects 1996-2000

Sub-domain	Projects
1) Multimedia Archive	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 31, 32, 35, 36, 37, 39, 47, 48, 50, 55, 57
2) Selective Dissemination of Information	38, 41, 49
3) Data Analysis	23, 28, 29, 43, 44, 54
4) Activity Monitoring and Control	46
5) Automated Reference Services	18, 24, 28, 33, 34, 45, 51
6) Data Migration	2, 22, 29
7) Virtual Access to Special Collection	19, 40, 52, 53
8) COTS Package Extension	2, 23, 27, 30, 56
9) Distributed Borrowing	16, 42
10) Interactive Communication	21, 25

Using the EF is straightforward:

1. Identify the project sub-domain as described in Section 4.1.
2. Review previous projects within the given sub-domain. Look for analogous projects.
3. Refine project WinWin taxonomy (Section 4.3), top risks (Section 4.4), pre-architecture (Section 4.2), simplifier and complicators (Section 4.5) and other project models as inspired by analogous projects and EF common pitfalls for the project sub-domain.
4. Review the domain specific design patterns (DSDP) for the project sub-domain and apply patterns to the project's specialized pre-architecture (Section 4.2) as appropriate.
5. Identify and evaluate possible COTS for sub-domain from the EF repository as indicated by DSDPs and other pre-architecture components.

6. Map appropriate COTS onto the specialized pre-architecture.
 7. Evaluate analogous projects' code base and prototypes for possible reuse.
- Example:** After Activity 1, we have identified the project sub-domain as Multimedia archive. We have looked at the example projects as listed below, and selected the closest projects to Asian Film Database and refined Asian Film Database accordingly reusing simplifiers and complicators and architectural diagrams taking into common pitfalls into consideration (Activity 2,3). Model View Controller and Façade were used as design patterns for user interface (Activity 4). COTS products from below were evaluated to see the best fit for the specific project. Mr. Sid has been chosen for viewing large images, however MS Access wasn't chosen as a database due to the platform limitations. After identifying the COTS products, mapping of the architectural components is performed such as repository component by the chosen database, Mr. Sid as image viewer.

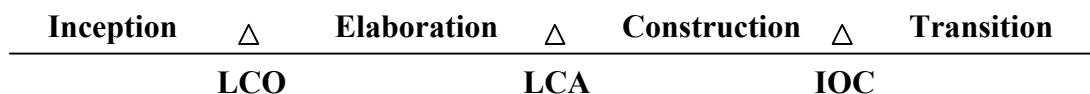
Multimedia Archive Experience Repository

Multimedia Archive		
Example Projects: 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 31, 32, 35, 36, 37,39, 47, 48, 50, 55, 57		
COTS:		
Name	Vendor address or URL	Project Uses
Mr. Sid	http://www.roktech.net /ROK/docs/ Products/MrSid.cfm	<ul style="list-style-type: none"> Serving and viewing of large images in a web browser. Provides image manipulation functions such as zoom, rotate, select region. Used in projects 47,..
MS Access	Microsoft	<ul style="list-style-type: none"> A database management system. Provides facility to organize data, find and retrieve information effectively. Used in projects 10,..
BANS	http://www.bans.com	<ul style="list-style-type: none"> Operates by connecting to your mail server and filtering each waiting message, searching for what you specify. Used in projects 41,..
...	...	<ul style="list-style-type: none"> ...
<p>DSDP: Model View Controller, Façade, Composite, Reactor</p> <p>Common Pitfalls: Dependency on the use of cgi scripts, ISD limits the use of these scripts due to security reasons; Plan your schedule taking into delays in the negotiation activity; Restricted usage of legacy software due to unexpected occasions, Integrated Library System's test server could not be used for prototyping because it was needed in transitioning from old to new system; Lack of required expertise in library information systems; Doing software evaluation for Search Engine without investigating the experience factory,</p>		

5. Evolution and Application of Domain Specific Techniques

The techniques described in Section 4 evolved to meet the challenges encountered as described in Section 2 in our six years of developing digital library products for the USC Libraries. From an individual project perspective, we have been evolving an approach called Model-Based (System) Architecting and Software Engineering (MBASE) that incorporates a high-degree of domain engineering. MBASE involves early reconciliation of a project's success models (correctness, business case, stakeholder win-win, ...); product models (domain, requirements, architecture, ...); process models (waterfall, evolutionary, spiral, ...); and property models (performance, reliability, ...). It extends

Figure 3. Rational/MBASE Phases and Milestones



the classic spiral model as described in [Boehm 1998b] by establishing critical milestones that are used to evaluate project progress. The milestones are the Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC) and are ordered as illustrated in Figure 3.

The LCO and LCA milestones involve the concurrent achievement of the following six system definition elements:

- An Operational Concept Description, including system goals and boundary, current system shortfalls and operational scenarios for the proposed system;
- Prototypes and/or key executing core capabilities of the system;
- A Requirements Definition, including capability, interface, quality attribute, and evolution requirements;
- An Architecture Definition, including choices of COTS and reuse components;
- A Life Cycle Plan, including key software life cycle milestones, schedules, deliverables, organizational roles, and risk management;
- A Feasibility Rationale, including a business case analysis, and demonstrating satisfaction of the key success criterion: that a system built to the architecture will support the concept of operations, be compatible with the prototypes, satisfy the requirements, and be buildable within the budgets and schedules in the plan.

The primary success criteria for the LCO milestone are that the project has demonstrated a viable cost-effectiveness business case, has shown at least one architecture that can satisfy the Feasibility Rationale success criterion, and has achieved stakeholder concurrence on the key system parameters. The primary success criteria for the LCA milestone are that the project has committed to a specific architecture which satisfies the Feasibility Rationale success criterion, that the project has identified all its critical risks, and that these risks are either resolved or covered by a risk management plan. Later we will use the LCO and LCA success criteria to evaluate the effectiveness of introducing the domain specific techniques. Here we indicate where the challenges with respect to MBASE occur and how the domain specific techniques address them.

For our digital library projects, we have roughly 11 weeks to go from a short problem statement (typically 4-5 sentences) to an LCA package, consisting of various prototypes and about 190 pages of documentation. This includes about 6 weeks for the Inception phase to develop the LCO version of the package. To reduce the risk that the various projects (typically 15-20) will go off in totally incompatible directions, we have developed the domain-specialized Rapid Application Development of the MBASE approach for the digital library projects utilizing the techniques described in Section 4.

Sub-domain Identification

Figure 4 shows the MBASE approach for the Inception phase of the digital library projects. For the projects, the beginning point at the top of Figure 4 is a domain model. While these are not difficult to produce, they require a significant amount of time from all stakeholders (e.g. interviewing customers). The sub-domain model furnished to the

students as identified by the technique in Section 4.1 includes the system boundary, its major interfaces, and the key stakeholders with their roles and responsibilities. This greatly narrows down what must be focused on to rapidly generate an appropriate domain model and also satisfies the MBASE LCO identification of Stakeholders and Primary Win Conditions. In doing this we address the challenges described in Sections 2.1 and 2.5 for a particular project where the developers do not possess specialized domain knowledge and are working under a constrained schedule.

Initial Requirements Mapping

As indicated in Figure 4, MBASE requires establishing a WinWin taxonomy, which is used as a checklist and organizing structure for the WinWin requirements negotiation process. As shown at the left of Figure 4, this taxonomy is also used as the table of contents for the requirements description, ensuring consistency and rapid transition from WinWin negotiation to requirements specification. This involves rapidly acquiring a deep and broad understanding of the potential requirements, here again facing the challenges of Sections 2.1 and 2.5. For each sub-domain model we have established a taxonomy that provides a well-scoped and manageable head start on potential requirement areas. This is rapidly customized and validated by utilizing the initial requirements mapping technique of Section 4.3.

Pre-architecture Identification

Pre-architectures (Section 4.2) evolved from a need to rapidly identify a system's basic concept of operation, manage "I know it when I see it" (IKIWISI) and prototype expectations, and to identify early viable architecture options as required within the MBASE LCO milestone elements. The evolution is simple, over time we noticed several recurring patterns in the system block diagrams that are created within the CS577 MBASE projects. After identifying their project sub-domain (Section 4.1) students attempt to specialize the associated generic block diagram to their project. This provides a head start on possible system operations and tangible framework to discuss a host of architectural issues before any implementation. It rapidly identifies unclear and high-risk areas that may be resolved through prototyping and provides a basis to manage expectations and discuss feasibility tradeoffs. The pre-architecture diagram typically serves as the framework for simplifier and complicator (Section 4.5) discussions.

Experience Factory

The experience factory described in Section 4.6 touches many areas with respect to the MBASE LCO milestone elements. In particular, the EF repository helps developers get a jump-start on the following LCO elements:

- Viable architecture options by examining analogous prior project architectures, and EF sub-domain domain specific design patterns, common pitfalls, and COTS usage.
- Environment models by examining analogous prior project domain descriptions.
- Property models by examining EF sub-domain metrics, risk assessments, and analogous prior project effort models.
- Prototypes through analogous prior project Prototype models (an MBASE artifact), and EF sub-domain COTS usage, and domain specific design patterns. Sometimes code from previous projects is used to prototype a new system.

After identifying the project sub-domain (Section 4.1), our students will browse the EF repository to find previous projects to use as a basis for their project. Application of the EP is usually straightforward. Typically the students will be “inspired” or directly make use of information from several related projects. We do not consider this plagiarism, rather a form of “enlightened reuse” that we actively encourage. For example, it is often difficult to locate an appropriate COTS package to provide web based large image viewing. After an exhaustive search and analysis, project 47 found the Mr. Sid product would meet their oversized book image viewing needs well. Later, other projects needed a similar capability to handle viewing large slide images. After consulting the EF, they were quickly able to assess that Mr. Sid would also work for their project. So long as they provide clear references and are careful to thoughtfully apply information from prior projects, we find that the projects are completing faster, with more capabilities, at a higher level of service, and increasingly satisfied (if not more demanding) customers. Clearly the EF addresses COTS, reuse, and standards challenges. That the EF addresses all of the challenges listed in Section 1 is not a surprise as it is intimately related to all the domain specific techniques we have introduced as well as greater particular project techniques.

Risk Identification

The technique in Section 4.4 indicates the most frequent risks involved in developing applications in that sub-domain. This satisfies the MBASE LCO milestone for the identification of Frequent Risks. This is a specialization of the list of 10 most frequent software risks in [Boehm 1989] as exemplified for the Multimedia Archive sub-domain which includes performance risks for image and video distribution systems; risks that users could not fully describe their win conditions, but would need prototypes; and risks that users would expect more capabilities than could be developed in the 12-week spring-semester development period. Risks are often tricky to establish and assess early without a great deal of previous experience in the particular project domain (the “learn by being burned” phenomenon). It is easy to overlook areas of considerable risk and not have the ability to recover. By leveraging frequent risks with respect to particular sub-domain the likelihood of this can be reduced. Greater project assurance is achieved by continuously tracking frequent risks on a weekly basis.

Simplifiers and Complicators

The top-n risk item checklist discussed above enabled the teams to identify some of the high expectations as risk items and to resolve them early. However, the risk items list is not sufficiently detailed or domain-specific to catch all the unrealistic expectations. As a result, during 1996 and 1997, we found that about 25% of the projects failed to satisfy the primary LCO Feasibility Rationale success criterion, i.e., to demonstrate at least one architecture that could satisfy the conditions in the Operations Concept Description, Requirements Description, and Life Cycle Plan. Following the LCO Architecture Review Board feedback, these projects were able to produce satisfactory LCA packages, however with more effort and less success than if the problems had been detected earlier. The problems were due largely to the two-culture gap. For example, 4 out of the 15 projects in 1996 exhibited the following difficulties:

- A student film archive project focused on cataloguing and querying issues, and underestimated the performance problems involved in digital film distribution across a campus network.
- A multimedia museum project underestimated the range of variability required of the cataloguing, query, and distribution subsystems in dealing with heterogeneous media. The project assumed that a single set of generic subsystems would handle all of the media.
- A financial table reformatting system project underestimated the range of variability involved in dealing with a wide variety of heterogeneous word processing systems.
- A large photographic archive project focused on the issues of automating the archive's operation, and seriously underestimated the amount of effort required to digitize the photos.

In 1997, we had 4 of 16 projects exhibit similar LCO package difficulties:

- A Virtual Reference Librarian project underestimated the complexity of handling natural language queries, and of building artificial intelligence like capabilities to infer frequently asked questions from a set of natural language queries.
- A project to develop a front-end query capability for heterogeneous set of art-collection databases underestimated the complexity of developing a common query language and translator for the query inputs. The project did not address the even more difficult problem of handling the heterogeneous query outputs, including error responses.
- A famous-writer-archive project underestimated the difficulties of cleaning up archaic audiotapes, synchronizing audio and text, and providing an English-German multilingual capability.
- A project to convert legacy periodical records into a new set of common formats underestimated the complexity of creating a general set of record translators across a wide variety of periodical characteristics (variable publication frequency, volume numbering discontinuities, etc).

The technique described in Section 4.5 provides stakeholders with a better mutual understanding of application simplifiers and complicators and reduces over expectations and the frequency of projects specifying unachievable requirements. On a sub-domain level this touches nearly all the challenges listed in Section 2 especially level of service expectations, standards, and use of COTS. For the 1998 and beyond series of projects, we managed to reduce the frequency of these LCO package shortfalls.

The application of the domain specific techniques pre-positions the project to satisfy the MBASE LCO success criteria by leveraging previously known successful approaches. Although the techniques evolved within the MBASE framework, it is clear that they would apply equally well independent of MBASE. The inputs and relationships between the techniques do not depend on the MBASE framework and can be used in a self-contained way as indicated in Figure 5.

Figure 4. MBASE Model Integration: Inception Stage ending with the LCO milestone

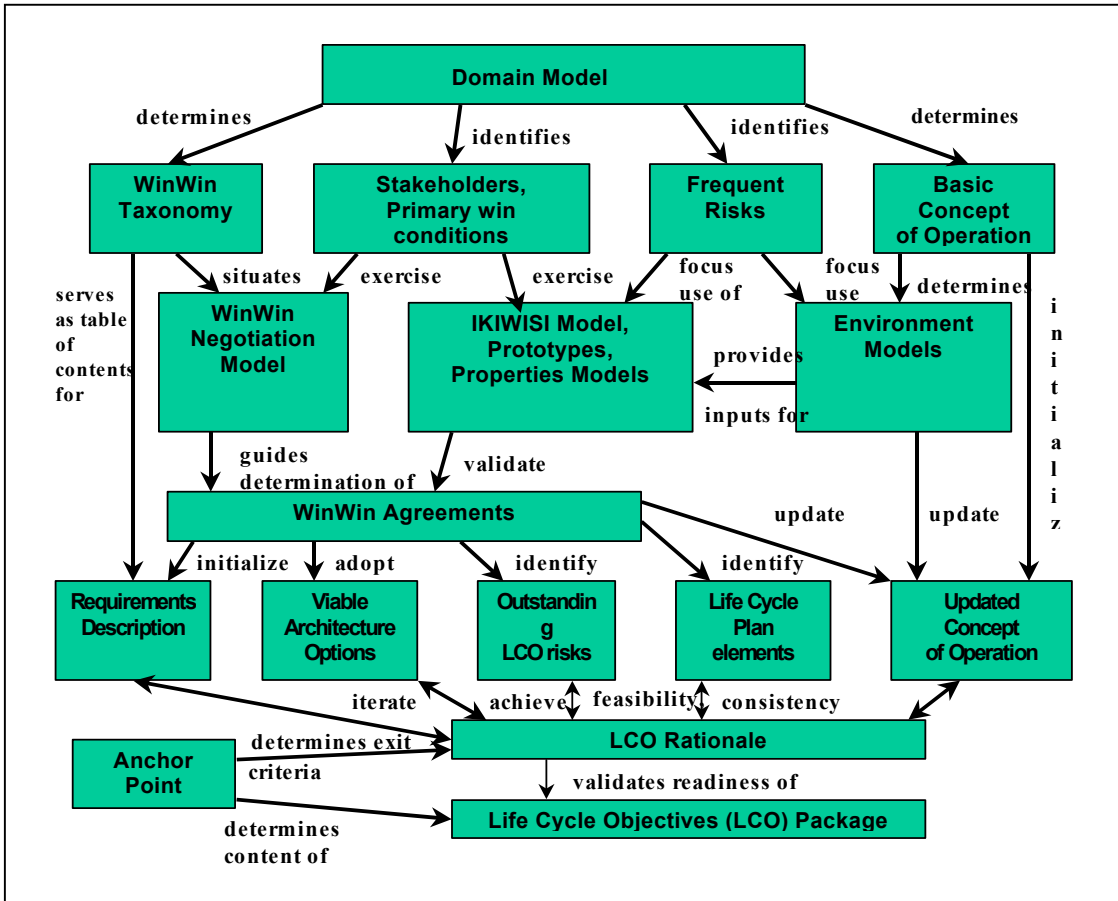
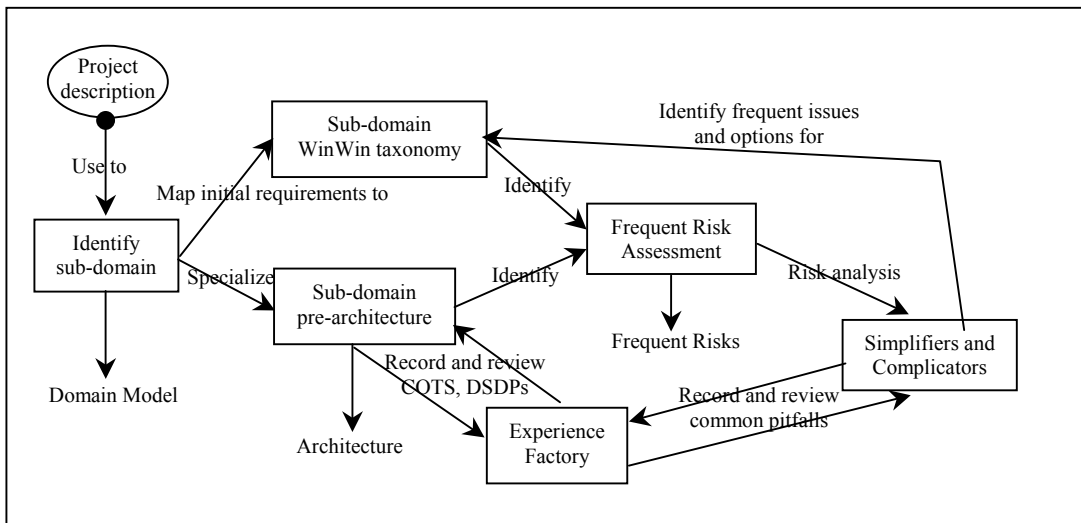


Figure 5. Integrated techniques Map



6. Effectiveness of Techniques

We used “percentage of projects failing the LCO Feasibility Rationale success criterion” as described in section 5 to measure the effectiveness of the domain specific techniques. Given that projects, technology and student capabilities (e.g., with Web-based applications) change from year to year, and that other factors (e.g., personnel shortfalls or incompatibilities) may cause LCO package failures, this approach is necessarily imprecise. Also, it is incomplete, as it does not test the relative effectiveness of S&C’s vs. prototypes or other methods of expectations management. However, the number (15-20 per year) and similarity of the projects provides a stronger-than-usual opportunity to test hypotheses about real-client software engineering.

None of the 23 2000 projects failed its LCO Feasibility Rationale success criterion. This is a 5% failure rate in 1998 just after (most of) the techniques were introduced and the 25-27% failure rates in 1996 and 1997 prior to their introduction. For the 1998 series of digital library/applications, the techniques successfully reduced the number of projects having serious Life Cycle Objective milestone feasibility problems, from 4 in 12 for 1996, 4 in 17 for 1997, and 1 in 19 for 1998, to 1 in 22 for 1999, and 0 in 23 for 2000. Client Evaluations after the projects were completed rose from 4.15 in 1996 to 4.75 in 1999. Additionally, the majority of students and clients stated that they have found the domain specific techniques quite useful in their evaluations.

There are many factors that contribute our improving LCO success rate beyond the domain specific techniques. To name a few, our maturing MBASE framework, refined instructions, repeating clients, and greater university support. While these certainly have had an impact, there are also a number of counter factors that just as easily could have reduced our success. Each year we have been dealing with larger enrollments, more complex and diverse projects (including many non-multimedia, outside the digital library domain), more demanding customers, and a host of other issues that introduce new challenges. We are confident that we can attribute the significant drop in the LCO failure rates to the introduction (as subsequent refinements) of the domain specific techniques. This comes primarily from the fact that the first significant drop was in 1998 and we did not introduce many changes outside of the domain specific techniques that year. Each subsequent year the rates continued to drop and hence this was likely not a fluke.

Table 12. List of Multimedia Project Examples

1996-1997

1. Cinema-TV Moving Images
2. EDGAR Corporate Data
3. Hancock Library Image Archive
4. ITV Course Material
5. Korean-American Museum
6. Latin American Pamphlets
7. Digital Maps
8. Medieval Manuscripts
9. Planning Documents
10. Photographic Archives
11. Stereoscopic Slides
12. Technical Reports Archive

1998-1999

30. Data Mining the Library Catalog
31. California Virtual University DB
32. Dissertations
33. Education Reference Assistant
34. Business Q&A’s
35. Asian Film Database
36. WWI - Record Enhancement
37. Digital Document Creation
38. Current Awareness Service
39. Hispanic Digital Archive
40. Book Locator for Doheny Stacks
41. New Booklist
42. Web Document Delivery

1997-1998

13. Architecture & Fine Arts Databases
14. Bella Lewitsky Database
15. Business School Working Papers
16. Inter-Library Loan
17. Engineering Technical Reports
18. General Library FAQ's
19. Hancock Museum Virtual Tour
20. Lion Feuchtwanger Archive
21. Network Consultation Support
22. Serial Control Records
23. Statistical Charts
24. Education Reference Assistant
25. Cross-Cultural Teaching Model
26. Business Reference Assistant
27. Online Catalog Search Strategies
28. Forms development for data ingest
29. Forms Development

43. Data Input Into the Digital Library
44. Voice Metadata Ingest
45. Authoring tool for ADE system
46. Seaver Auditorium Scheduling

1999-2000

47. Viewing Utility for Oversized Objects
48. Digitizing & Access Asian Materials
49. Automated New Books List Generator
50. Latino Serials Microfilm Collection
51. Business/Reference Q&A
52. Doheny Library Virtual Tour
53. Virtual Education Reference Assistant
54. Daily Summary for Japanese Press
55. Hispanic Digital Archive Enhancements
56. Managing Multimedia Databases
57. OCLC Record Retrieval

Note: not all CS577 projects are listed here

7. CONCLUSION

Abstracting experiences within multimedia digital library systems into concrete techniques and applying them in a disciplined way has helped our developers rapidly become more effective in areas that typically require a great depth of experience and skill such as risk management, client interaction, requirements negotiation, expectations management, software and system architecting, project organizing and planning, and product validation and transition. In addition, they have helped all stakeholders converge more rapidly and effectively on mutually satisfactory and achievable system requirements.

The techniques provide specific and tangible guidance within a general software engineering framework and enable more rapid development by “kick-starting” difficult areas. This guidance evolved from a need to address particular challenges with respect to multimedia software engineering. The key aspect of the techniques is identifying commonalities of systems within sub-domains through a “closed loop feedback” approach that continuously refines the techniques. While currently oriented to the digital library domain, it is straightforward to see how analogous techniques could be developed for other domains.

The effectiveness of domain specific techniques has been demonstrated anecdotally and through early project success assessments. It is clear that such techniques are practical and should continue to be refined and expanded to other domains.

References

- AHDS (2001), “Arts and Humanities Data Service”, <http://ahds.ac.uk/resource/standards.html>
- Basili, V.R., G. Caldiera and H.D. Rombach (1994), "The Experience Factory," *Encyclopedia of Software Engineering*, 1, pp 469-476.

- Booch, G., J. Rumbaugh, I. Jacobson (1999), "The Unified Modeling Language User Guide," Addison Wesley, pp. 461.
- Boehm, B. (1989), "Software Risk Management," IEEE-CS Press, 1989.
- Boehm, B., P Bose, E. Horowitz and M.J. Lee (1994), "Software Requirements As Negotiated Win Conditions," In Proceedings of 1st the International Conference on Requirements Engineering, pp.74-83.
- Boehm, B., A. Egyed, J. Kwan and R. Madachy (1997), "Developing Multimedia Applications with the WinWin Spiral Model", In Proceedings of the 8th European Software Engineering Conference, Springer Verlag, pp. 20-39.
- Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah and R. Madachy (1998a), "Using the WinWin Spiral Model: A Case Study," IEEE Software, 31, 7, , pp. 33-44.Boehm, B., A. Egyed, D.Port, A.
- Shah, J. Kwan and R. Madachy (1998b), "A Stakeholder WinWin Approach to Software Engineering Education," Annals of Software Engineering, 6, pp. 295-321.
- Boehm, B. (1998c), "A Spiral Model of Software Development and Enhancement," IEEE Computer, 21, 5, pp. 61-72.
- Boehm, B. (1999a), "Making RAD Work for Your Project," IEEE Computer, 32, 3, pp. 113-117.
- Boehm, B., M. Abi-Antoun, J. Kwan, A. Lynch and D. Port (1999b), "Requirements engineering, expectations management, and the two cultures," Proceedings of the IEEE International Conference on Requirements Engineering, pp. 14-22 .
- Boehm, B. and C. Abts (1999c), "COTS Integration: Plug and Pray?,"IEEE Computer, 32, 1, pp. 135-138.
- Boehm, B. and D. Port (1999d), "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them," ACM Software Engineering Notes, 4, 1, pp. 36-48.
- Boehm, B. and W. Brown (2001), "Mastering rapid delivery and change with the SAIV process model," Technical Report TR-01-01, Department of Computer Science, University of Southern California, Los Angeles, CA.
- Dager, J. (2000), "Cummin's Experience in Developing a Software Product Line Architecture for Real-time Embedded Diesel Engine Controls," 1st Software Product Line Conference.
- Dan, A., S. I. Feldman and D. N. Serpanos (1998), "Evolution and challenges in multimedia," Multimedia systems, 42.
- Dan, A. and D. Sitaram (1997), "Multimedia Caching Strategies for Heterogeneous Application and Server Environment", Multimedia Tools and Applications, 4, 3.
- Garlan, D., R. Allen and J. Ockerbloom (1994), "Architectural Mismatch: Why Reuse is So Hard," IEEE Software.
- Gamma, E., R. Helm, R. Johnson and J. Vlissides (1994), "Design Patterns Elements of Reusable Object-Oriented Software", Addison Wesley.
- Hayes-Roth, R. and W. Tracz (1994), "DSSA Tool Requirements for Key Process Functions,"ADAGE Technical Report, ADAGE-IBM-93-13B.
- Lee, M.J., (1996), "Formal Modeling of the WinWin Requirements Negotiation System," Ph.D. Thesis, Department of Computer Science, University of Southern California, Los Angeles, CA.
- Port, D. and B. Boehm (2001), "Educating Software Engineering Students to Manage Risks," Proceedings, International Conference on Software Engineering.
- Pronk, B.J. (1999), "Medical Product Line Architectures," 1st Working IFIP Conference on Software Architecture , pp. 357-367.
- Rui, Y., T. S. Huang and S. Chang (1999), "Image Retrieval: Current

Techniques, Promising Directions and Open Issues,” Journal of Visual Communication and Image Representation, 10, pp. 1-23.

Schmidt, D. (1999), “Why Software Reuse has Failed and How to Make It Work for You,” C++ Report Magazine January 1999. SEI (1990), Proceedings of the Workshop on Domain Specific Software Architectures, Software Engineering Institute, Pennsylvania.

Serpanos, D., L. Georgiadis, and A. Bouloutas (1998), "MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers," IEEE Transactions on Circuits and Systems for Video Technology, 8, 1, pp. 13-17.

Sharp, D. (2000), “Component Based Product Line Development of Avionics Software,” 1st Software Product Line Conference.

Smart, P.F., S.N. Woodfield, D.W Embley and D.T Scott (1988), “An empirical investigation of the effect of education and tools on software reusability,” In Proceedings of 7th Annual International Phoenix Conference on Computers and Communications, March 1998, IEEE Computer Society Press, Washington, DC, pp. 224-228.

Snow, C.P. (1959), ”The Two Cultures and the Scientific Revolution,” Cambridge University Press, 1959.

Tracz, W. (1994), “Collected overview reports from the DSSA project,” Loral Federal Systems, Owego.

Tracz, W. (1995),”DSSA (Domain-Specific Software Architecture) Pedagogical Example”, ACM SIGSOFT Software Engineering Notes, July 1995.

[Tsai99] J. P. Tsai, "Knowledge-based Software Architecture", IEEE Trans. on Knowledge and Data Engineering, Vol. 11, No. 1, Jan 1999.

[Lyar98] D. Lyardet, G. Rossi, D. Schwabe, "Using Design Patterns in Educational Multimedia Applications", Proc. of ED-MEDIA'98, Friburg, June 1998.

[Blair] Composition in Multi-Paradigm Specification Techniques