OSMA Software Program
Domain Analysis Guidebook

Victor R. Basili
Institute for Advanced Computer Studies and Department of Computer Science, UMCP
Fraunhofer Center for Experimental Software Engineering – Maryland

Carolyn Seaman
Department of Information Systems, University of Maryland at Baltimore County (MBC)
Fraunhofer Center for Experimental Software Engineering – Maryland

Roseanne Tesoriero
Department of Computer Science, University of Maryland at College Park (UMCP)

Marvin V. Zelkowitz
Institute for Advanced Computer Studies and Department of Computer Science, UMCP
Fraunhofer Center for Experimental Software Engineering – Maryland

Abstract

Domain analysis is the process of identifying and organizing knowledge about a class of problems. This guidebook presents a method of performing *experience domain analysis* in software development organizations. The purpose of the guidebook is to facilitate the reader in characterizing two given development environments, applying domain analysis to model each, and then applying an evaluation process, based upon the Goal/Metric/Paradigm, to transfer a given development technology from one of the environments to the other. This guidebook describes this process and gives an example of its use within NASA.

Foreword

This document was produced by the authors for the National Aeronautics and Space Administration (NASA) Office of Safety and Mission Assurance (OSMA) Software Program, an agency-wide program designed to address and mitigate software-related risk within NASA programs. The goals and strategies of this program are traceable to the NASA Strategic Plan, the NASA Software Strategic Plan, and the OSMA Strategic Plan.

The present document is one element of the set of standards, guidebooks, and reports produced in support of the OSMA Software Program. The process described in this document is based upon the work of Rose Pajerski (Fraunhofer Center for Experimental Software Engineering – Maryland). Preliminary efforts on understanding experience application domains were performed by Frank McGarry (Computer Sciences Corporation and Lionel Briand (Fraunhofer Institute for Experimental Software Engineering – Germany). We appreciate the help of David Petri (NASA/JSC) for allowing us to develop this model with the help of his Rapid Development Laboratory. We greatly acknowledge the comments from Sira Vegas and Maurizio Morisio, who read an earlier draft of this report. This activity was partially sponsored by NASA grant NCC5170 to the University of Maryland.

Contents

Chapter 1. Introduction

Domain analysis is the process of identifying and organizing knowledge about a class of problems. The goal is to recognize standard concepts, functions, and architectural characteristics within a software development application area. Domain analysis has taken on greater importance recently as a means to facilitate product reuse and improve both the productivity and quality of the final products by allowing knowledge from other domains to be applied easily and reliably to a new domain.

Often domain analysis refers to specific software applications, such as payroll systems, fluid dynamics, weather predictions, or in the case of NASA software, to applications such as onboard computation, flight dynamics, and ground support. In this report we take a broader view of a domain. We also consider the environment which develops the software: the people, the computers, the automated tools, and the processes that are used. This provides a context for domain analysis in order to enable not only product reuse (i.e. source code), but also for reuse of other software-related artifacts, including process models (e.g., best practices), cost estimation models, defect baselines, and software techniques and methods. In other words, a “domain” may be more than an application domain. We use the term “experience domain analysis” to refer to the gaining of a domain understanding in order to reuse experience of any kind.

Therefore, in addition to product reuse issues, we view *experience domain analysis* as a means of identifying areas and groups of systems for the reuse and sharing of experiences where:

- similar development or maintenance standards and processes may be applied (e.g. identify systems for which DOD-STD-2167 is applicable within NASA/GSFC’s Information Systems Center (ISC))

- quality and productivity data and models are comparable (e.g., identify systems for which the error rates are comparable within all the branches of NASA/GSFC’s ISC

- similar development environments may be used (e.g., identify systems where C++ would be an effective development environment and identify systems where Ada would be an effective development environment)

The goal of this document is to provide some practical guidance related to domain analysis in order to define a context for the reuse of software experiences. Once domains have been identified, common processes, standards and databases of experience may be shared with confidence by various software organizations within a broader organizational structure. Also, when similarities are numerous in terms of standards and models, these software development and maintenance organizations can share a common body of experience. Hence, software domains may be defined, not solely *a priori* on the basis of some organizational partitioning, but according to the factors that characterize the specific development processes, technologies, products, constraints, goals, and risks associated with the projects. Thus, because organizations can share data, lessons learned, and best practice information, they can improve faster and further than they could in isolation.

If domains can be identified, then there is a possibility for experience sharing and mutual collaboration for refining processes and evaluating new technologies. However, an organization is needed in order to identify these domains, and to collect and package empirical experience for each of the domains. It is also needed to embed the experience domain analysis process without having to relearn this activity each time it is required.

This organization is referred to as the Experience Factory [Basili1994]. For practical reasons, two kinds of Experience Factories are actually needed. The first one, referred to as the *Local Experience Factory,* is specific to a development environment and is responsible for the local data collection and packaging. It also provides support and consulting to the development organization. The second one, referred to as the *Consolidated Experience Factory,* is responsible for identifying commonalties across organizations and packaging them into common models. It also helps insure consistent data collection across organizations when possible, and is supported, fed, and used by multiple organizations. The Consolidated Experience Factory is shown in Figure 1-1.

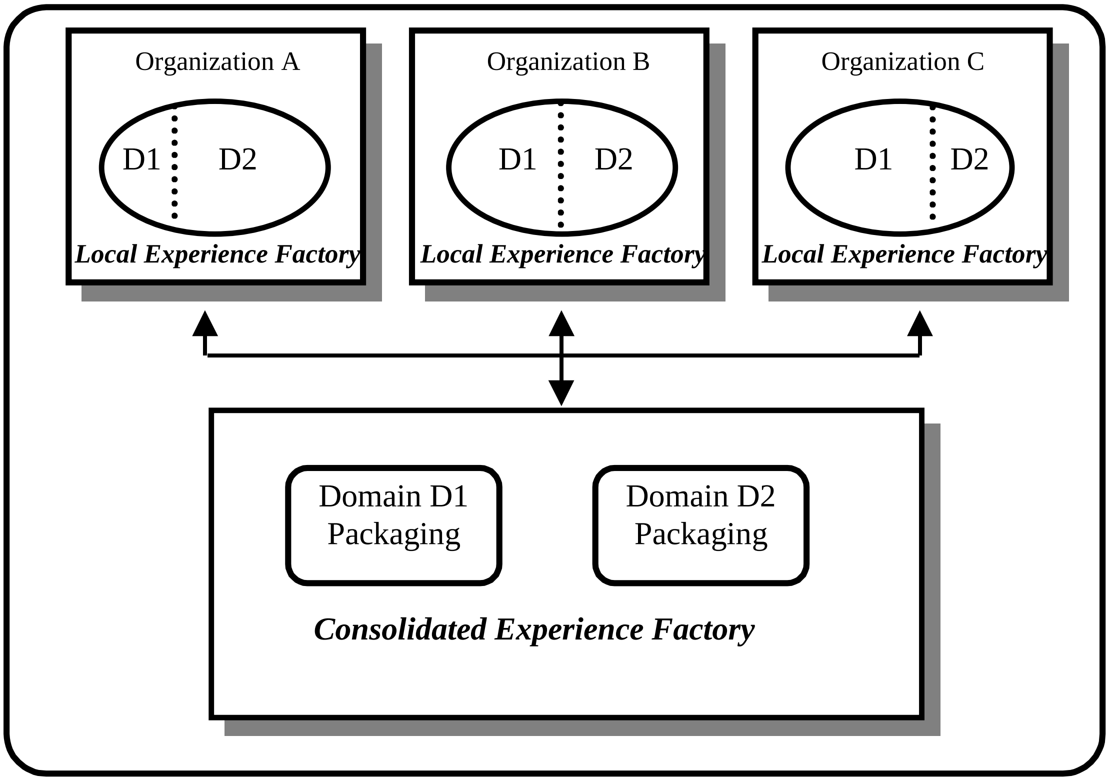

Figure 1-1. Consolidated Experience Factory

It is toward the development of such a Consolidated Experience Factory that this experience domain analysis guidebook is oriented. By formalizing the steps necessary to understand a domain, the transference of that domain experience to another domain is made that much simpler.

In order to implement experience domain analysis, three processes must be undertaken:

- We must **characterize** the original and the new environment in order to determine what we can and want to reuse. What technologies are amenable for reuse, and what factors in these environments will affect the feasibility of reuse?

- We must **model** the existing environments so that we are able to understand the constraints on our analysis.

- We must **evaluate** the model with respect to the technology we wish to reuse.

Chapters 2 through 4 of this document present the activities of the experience domain analysis process. Chapter 5 presents a method for extending this process for long-term environment improvement with experience domain analysis as one component of the method.

Chapter 2. Characterizing a Domain

We can summarize the basic concepts of domain analysis with the following definitions:

> **Definition 1:** *Domain Analysis*
>
> Domain analysis is the process of identifying and organizing knowledge about a class of problems.
>
> **Definition 2:** *Experience Domain Analysis*
>
> Experience domain analysis is the process of identifying and organizing knowledge about a set of software artifacts within or across organizations.

The goal of experience domain analysis is to assess the feasibility of reusing or sharing a set of software artifacts within or across organizations. For example, can we effectively use a cost model developed in a new environment that is based on a different set of past projects than those of our current development? Will the cost model make the same predictions about the future in the new environment that it made in the older environment?

> **Definition 3:** *Domains*
>
> With respect to a particular domain analysis goal, domains are "types" of software development artifacts (e.g., software development organizations, projects, software components) for which certain common experiences may be reused or shared.
>
> **Definition 4:** *Domain Characteristics (Factors)*
>
> Domain characteristics represent those factors that determine whether or not one or more experiences can be reused or shared within or across organizations.

For example, in a given organization, large Ada flight simulators may represent a domain with respect to cost modeling and prediction. In this case, the project characteristics involved in the domain characterization function are the project size (large), the programming language (Ada), and the application domain (flight simulation).

The following sections describe how domains can be characterized by these definitions.

2.1 Transferable Artifacts

The first step in experience domain analysis is to determine the kinds of experience one wants to reuse or share. We characterize experience as an artifact usable by the developers of a software product. As an example, we present below a generic taxonomy of software artifacts that can conceivably be reused or shared within or across organizations. The experiences presented by the artifacts in the following taxonomy and in the rest of the paper are project level experiences, since we think the project level is the appropriate level of application for experience domain analysis. However, we could similarly address the issues related to other kinds of experiences associated with other levels of artifacts (e.g. development organizations such as branches, directorates, or NASA centers).

It is important to note that the taxonomy presented here encompasses more than just software products. Also, one could further refine the taxonomy within each specific organization.

Taxonomy of Experience Domain Analysis Technologies

Artifacts, which are reusable from one project to another, are indicated by the following taxonomy. Note that we consider much more than simply source code or design documents as the experiences that may be shared. All packageable information is a candidate artifact for experience domain analysis.

- **Data and Models**
    - Descriptive models
    - Predictive models
        - Cost models
        - Schedule models
        - Reliability growth models
        - Error models
        - Change models
        - Performance models
    - Quality evaluation models
    - Lessons learned

- **Standards and Processes**
    - Requirements
    - Specifications
    - Design
    - Coding
    - Testing, Inspections, Evaluations
    - Change management
    - Process improvement (e.g., GQM)
    - Configuration management
    - Quality assurance

- **Products**
    - Requirements, Specifications
    - Architecture

- Design
- Code
- Test plans and data

One must determine the goals desired of reuse in order to identify the factors that will be relevant to the reusability of the predefined software artifact or experience. The following list of questions attempts to identify some the most important goals for motivating experience domain analysis. This list is not intended to be complete, but rather a sample of the important questions that are not product-related and that can be addressed by an experience domain analysis procedure.

1. Should a common standard, process, or best practice be used across the entire organization? Are there domains of projects that require different technical solutions and, as a consequence, different standards (i.e , does flight or embedded software require a different design process than ground support systems because of real-time and memory constraints)?

2. When can one reuse a particular standard, process, or best practice that has been used successfully in another organization (i.e., can the Cleanroom development process as originally designed at IBM be used by a NASA organization)?

3. When can one compare the particular quality metric values for a set of projects (e.g., productivity, error density, reliability, reuse level) across a organization? What are the project domains across which this particular quality aspect is not comparable?

4. When can one trust lessons learned from another organization (e g., information about the usefulness of a development tool)?

5. When can one reuse data and models (e.g., a cos model) from an ther organization?

Certain kinds of experiences are *a priori* more likely to be reusable or sharable than others because they naturally have a broader realm of application. For example, many high-level concepts are universally applicable, such as the tracking of project progress across the development life cycle through data collection to monitor or control schedules and resource consumption. Other kinds of experiences may have a somewhat more restricted realm of application. For example, a waterfall process model is only applicable as long as the application domain is well known and the solution space is reasonably well understood. Furthermore, some kinds of experiences have a very narrow realm of application, such as experiences related to application-specific programming languages and operating systems (e.g., a real- ime UNIX variant for real-time applications).

2.2 Experience Domain Analysis Factors

Based upon the artifacts identified by the previous section, different project and environmental factors need to be considered to help identify the appropriate domain of projects. These factors provide the goals for transferring the artifact. For each factor deemed relevant in our domain, we need to identify the quantitative values that the factor has in both the old and new environment. This will allow us to decide if the artifact can be safely reused in the new environment.

Taxonomy of Domain Characteristics (Factors)

The following is a taxonomy of factors (potential domain characteristics) to be considered.

- **Product**
    - Functionality
    - Requirement stability
    - Concurrent software
    - Memory constraints
    - User interface complexity
    - Safety and reliability requirements
    - Lifetime requirements
    - Size
    - Programming language
    - Intermediate product quality
    - Product reliability

- **Process**
    - Process model (e.g., waterfall, spiral)
    - Process conformance
    - Environment constraints (e.g., schedule, budget, funding)
    - Productivity

- **Personnel**
    - Application domain experience or training
    - Platform experience
    - Process experience
    - Education
    - Motivation
    - Development team organization
    - Turnover
    - Willingness to change

- **Environment**
    - Application domain
    - Development domain
    - Manager’s role
    - External contract support

Chapter 3. Modeling a Domain

Given the set of technologies and factors that can be used to characterize a given domain, the next step in experience domain analysis is to model the domain. To formalize the domain information, a domain modeling scheme developed by Andreas Birk at the Fraunhofer Institute for Empirical Software Engineering in Kaiserslautern, Germany is described. The reader is referred to [Birk1997] for a detailed explanation of the modeling strategy.

It is important to realize that the domain model depends upon the viewpoint from which it has been derived [Birk1997]. Developing a model that characterizes project management issues will be different from one that characterizes software development issues.

The domain model consists of four parts: header, intuitive domain model, operationalization, and operational domain model. Appendix A gives a detailed example of this domain model.

3.1 Header

The header relates the domain model to its technology and environment and simply contains identifying information. It contains five components: technology, task, goal, viewpoint, and environment. The task and goal are particularly important, as these must match the task and goal of the project that wants to import the technology. This should be obvious, but is sometimes overlooked.

3.2 Intuitive Domain Model

The Intuitive Domain Model, the second part of the model, describes the domain factors that have been identified at a conceptual level. The domain factors are also assigned a set of possible values; the model is actually a faceted classification scheme. Each facet is a domain factor listed in section 2.2. For example, for “programming language,” we would indicate what language is being used. For quantitative factors, such as the personnel factor “platform experience,” we could use numerical values such as years of experience, or ordinal values such as “many,” “some,” or “few”. The domain factor and its value is called a *domain characteristic.*

3.3 Operationalization

The third section of the model, the Operationalization, defines the domain factors more formally. Three kinds of mappings can be used for these definitions:

1. *Direct operationalization,* which can be used for explicit intuitive domain models. For example, for “programming language,” we could use the explicit language name (e.g., Ada, C++)

2. *Functional operationalization,* which can be used for intuitive domain factors that can be broken down into more elementary domain factors. The domain factors in Appendix A are generally defined as simple algorithms from more primitive factors. For example, for the factor “Attitudes toward measurement” described in Appendix A, the functional operationalization is given as

if **goals** is *no* then
    attitudes toward measurement is ***low***
else

managers → **senior managers** ↓	*high*	*medium*	*low*
high	***high***	***medium***	***medium***
medium	***high***	***medium***	***low***
low	***medium***	***medium***	***low***

Based upon the attitudes of both managers and senior managers, then the attitudes toward measurement can be specified.

3. *Vague operationalization,* which can be used for domain factors that can be traced to more elementary factors, but cannot be specified precisely. The factor “Experience of developers” is one such factor that cannot be given explicitly. We could refer to “Experience of developers” as a direct operationalization by specifying the number of years of experience, but research has consistently shown that the number of years of experience is a poor predictor of capabilities. In such a case it is best to leave this sort of factor as a vague operationalization rather than a direct operationalization.

3.4 Operational Domain Model

Finally, the Operational Domain Model defines the measurement procedures and instruments to be used to apply the measures, which are in turn used to determine the values of the domain factors. These procedures and instruments are to be applied to an environment wishing to use the technology. In our domain model, the procedure used to collect all the data from the new environment can be an interview, with the instruments being the interview questions. Survey forms or data collected from previous projects are other aspects of the operational domain model.

3.5 The Goal/Question/Metric Paradigm

This section identifies how one might analyze the relationship between domain characteristics and reuse goals based upon experience. We have argued that domain characteristics strongly depend on the particular goal to be achieved by the experience domain analysis. Ideally, in a mature engineering discipline, the importance of all possible domain characteristics within the context of each relevant goal should be established. With respect to the discipline of software engineering, this is still difficult, or in some cases even impossible, to achieve.

Without objective data, the reader must decide, based on personal experience, which characteristics (within the taxonomy presented previously) affect the particular goals for the organization. A process, the *Goal/Question/Metric (GQM) Paradigm*, has been developed as a means to help in this activity. Based upon a specific goal, a set of questions can be posed that help address understanding the particular goal. From these questions, a series of metrics can be defined so that appropriate measurements can be made from the development process in order to be able to address the underlying goal. From this analysis, we can address the specific questions needed to build the domain model described in the previous sections of this chapter. The application of this process is given in Chapter 4.

Software development is an engineering discipline, and measurement is an ideal mechanism for feedback and evaluation. However, for an organization to measure in a purposeful way requires that it (1) specify the goals for itself and its projects, (2) trace those goals to the data that are intended to define these goals operationally, and (3) provide a framework for interpreting the data to attain the goals. The Goal/Question/Metric paradigm is a mechanism for defining and evaluating a set of operational goals, using measurement. It represents a systematic approach for tailoring and integrating goals with models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization. The GQM paradigm is pictured in Figure 3-1.

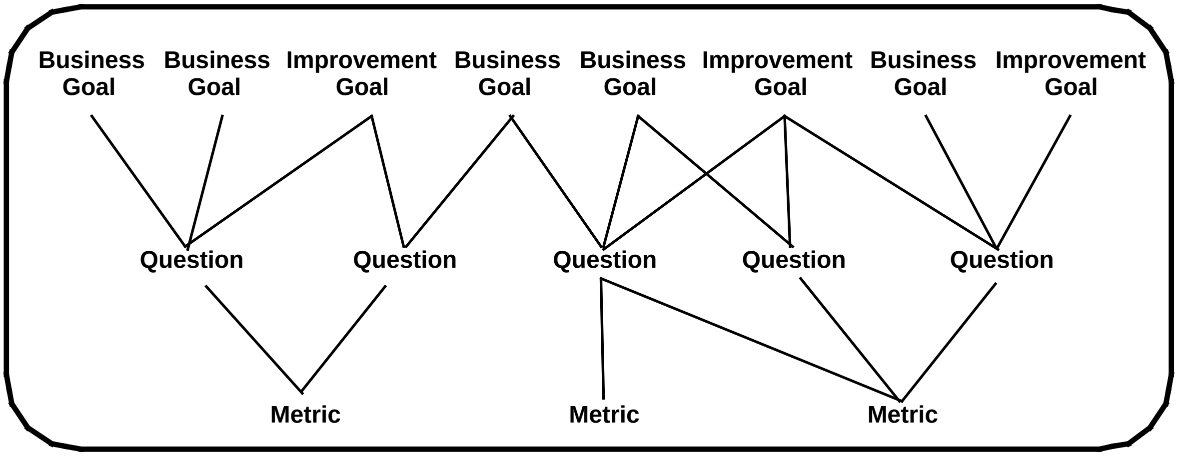

Figure 3-1. Goal/Question/Metric Paradigm

The goals are defined in an operational and tractable way by refining them into a set of quantifiable questions that are used to extract the appropriate information from the models. The questions and models define the metrics that, in turn, specify the data that needs to be collected. The models provide a framework for interpretation. As shown in the figure, the flow from the goals to the metrics in the GQM paradigm can be viewed as a directed graph, with the flow from the goal nodes to the question nodes to the metric nodes. Eight goals are shown, and each goal generates a set of quantifiable questions that attempt to define and quantify the specific goal that represents an entry node in the directed graph. These questions are based upon a particular set of process, product, and quality models that are not explicitly represented in the graph. Although there may be many goals and even many questions, the metrics do not grow at the same rate as the goals and questions. Thus a single set of metrics can be collected that allows us to answer many goal-based process and product questions.

Applying GQM involves five steps:

1. *developing* a set of corporate, division and project goals for productivity and quality (e.g., customer satisfaction, on-time delivery, improved quality)

2. *generating* questions (based upon models) that define those goals as completely as possible in a quantifiable way

3. *specifying* the measures needed to be collected to answer those questions and to track process and product conformance to the goals

4. *developing* mechanisms for data collection

5. *collecting, validating and analyzing* the data in both real time (e.g., to provide feedback to projects for corrective action) and in a post-mortem fashion (e.g., to assess conformance to the goals and make recommendations for future improvements)

Chapter 4. Evaluating a Domain

We now describe an analysis procedure aimed at performing experience domain analysis. This procedure is not an exhaustive presentation of all the techniques that could be involved in identifying domains. However, a framework is given which should help the reader identify the steps, their goals, and the kinds of techniques that can be used to perform them.

4.1 Identifying Domain Characteristics

There are four steps in identifying domain characteristics.

Step 1 *Identify the experience to be used (goal analysis)*

What experiences do you want to use and why? The specific goals of the experience domain analysis to be performed have to be defined (e.g., define a testing and inspection standard process across a NASA engineering center). Surveys of current practices in a given organization might help identify issues and relevant reuse or sharing opportunities in order to define goals for experience domain analysis.

Step 2 *Identify the relevant factors and generate metrics (domain characteristics' analysis)*

Step 2.1 *Determine relevant factors*

What factors affect the use of this experience? A set of factors likely to characterize domains within the context of each experience domain analysis goal must be defined. GQM can be used at this stage. Once the goals have been stated, identify the factors which may characterize different domains, and thereby require different solutions with respect to the goals. For example, the level of system criticality will have an impact on testing and inspection standards, while a high requirement instability might require specific inspection and consistency analysis procedures. As a consequence, these important variations in the development process are also likely to create the need for different cost models for systems with high requirement instability and high level of criticality. These two factors are examples of domain characteristics.

Step 2.2 *Assess availability of factors*

What measures, direct or indirect, are available for the given factors? Determine which factors are available for helping with the selection of adequate standards, processes and models once they have been defined or built. Let us assume that requirement stability is a crucial factor that determines the various domains where requirement and specification strategies should differ. However, requirement stability happens to be unknown and not easily predictable when starting the requirement and specification process. As a consequence, it becomes difficult to know which standard process to apply beforehand.

Step 2.3 *Determine operationalization of factors*

What appropriate measurement scale must be used for a given factor? When their level of measurement is either ordinal or nominal, one has to define the value domain of the identified domain characteristics. For example, an ordinal scale may be defined for levels of criticality according to the expected loss if a failure occurs during system operation. This scale is by definition subjective and could be defined along the following scale: minor financial loss, major financial loss, human life at risk, likely loss of human life. Categories should always imply significant differences in the reusability or applicability of the experience of interest; that is, all four categories of the factor level of criticality match different sets of solutions with respect to testing and inspection processes.

Step 2.4 *Obtain values for those factors in both domains*

What values for each identified factor can be determined? Interviews and feedback can be used to reach a consensus from all involved. In some cases, questionnaires may be distributed to project managers and the collected subjective information about past and current projects may be used as a rough substitute to actual project data.

When adequate data are available, the following steps can be performed in a straightforward manner.

Step 3 *Identify the relevant project set and associated data for the particular goal (ample analysis)*

Which projects can validly be used to perform the experience domain analysis? One should be careful not to systematically use all the data available within the organization's historical database. For example, some projects may be experiments aiming at evaluating new technologies, while others may be classified as pure maintenance tasks. These projects might not be relevant to the models/analyses to be built/performed and, in this case, they should be removed from the data set. Also, the data collection may be of varying quality from one project to another. The projects showing evidence of poor data collection can be discarded.

Do the factors in the "relevant project set" display sufficient variation to be of any use? Based on the value domains defined for each factor, the analyst must determine whether or not there is variability within the studied organization(s). For example, if requirement definitions appear somewhat stable over all the projects, then it becomes an irrelevant factor for experience domain analysis even though it may be a theoretically relevant factor. This step may involve the use of statistical analysis techniques such as analysis of variance, non-parametric rank-based tests, and tests for proportion.

Step 4 *For two different environments, compare the domain characteristics*

This step identifies similarities with respect to the factors of interest across the organization's projects. Similar projects are clustered in domains. Differences between domains must be significant enough to justify their differentiation with respect to the stated goal. On the other hand, consistency within domains must be high enough to allow the reuse or sharing of experience between the domain's projects. The level of confidence associated with the decisions that will be made based on domain comparisons will get higher as the domain definitions become more precise and the domain projects become more homogeneous. This step can be performed based on intuition (e.g., Delphi method) but is usually more accurate when based on data collected at the project level.

Compare the values of the domain factors for the new environment to those for the original environment and identify the domain factors with different values in the two environments. For each domain factor that represents a difference between the new environment and the old, determine whether the technology can be modified in the new environment or whether the difference will not affect its performance in the new environment. From this analysis, develop a plan to transfer the technology, or to decide the technology cannot be reused in the new environment.

4.2 Evaluating the Domain Model

The objective of experience domain analysis is to learn how one environment might affect the successful use of a technology by comparing the environment to another environment that already uses the technology. As an aid to describing this process, we outline the application of the steps from the previous section to describe the applicability of moving one technology (i.e., the *new* technology) from a *source* environment to a *target* environment[1].

In particular, we apply the previously described process with the following steps:

1. Identify the new technology that is to be reused in the target environment [Step 1].

2. Identify characteristics of the source environment that had an effect on the use of the new technology (i.e., the domain characteristics). For the source environment, both the domain factors and the values of those factors must be determined [Step 2.1 and Step 2.2].

3. Formalize the set of domain factors into a domain model [Step 2.3].

4. Determine the values of those domain factors in the target environment [Step 2.4 and Step 3].

5. Compare the values of the domain factors for the target environment to those for the

---

[1] This example is based upon moving the GQM technology from the NASA/GSFC Software Engineering Laboratory to the Rapid Development Laboratory at NASA/JSC.

source environment and identify the domain factors with different values in the two environments [Step 4]. For each domain factor that represents a difference between the target environment and the source environment, determine whether

- the new technology could be modified to accommodate the difference, or
- the way that the new technology is transferred to the target environment could be modified to accommodate the difference, or
- the difference cannot be accommodated and will prevent the successful use of the new technology in the target environment.

6. Develop a transfer plan for the new technology itself according to the analysis in step 5.

Of course, it is hoped that step 6 does not identify any differences in domain factor values that will prevent the successful transfer of the new technology. Each of the above steps is further elaborated in the following sections.

4.2.1 Identify the New Technology

The first phase is to identify the technology that is to be reused. In this example, the NASA/GSFC Software Engineering Laboratory (SEL) (the source environment) was to work with the NASA/JSC Rapid Development Laboratory (RDL) (the target environment). The RDL was embarking on a metrics program for its projects. The SEL has been collecting measurement data for over twenty years, and has been using the GQM method successfully for over ten years as a way to organize concepts on what data needs to be collected. The GQM paradigm has been adopted by many organizations worldwide. Thus, this seemed like a reasonable candidate technology that the RDL could use in its measurement program. From experiences in the SEL, GQM should be transferable to other NASA centers. The decision was made early that this technology (the new technology) would be the candidate technology to try.

4.2.2 Collecting Domain Factors

Applying the previously given model, one first gathers qualitative data on possible domain factors affecting the new technology use in the source environment. This data can be collected by way of open-ended interviews with a number of people who have used the new technology extensively in the source environment both to help design studies and to help manage projects. Interviews may be held singularly or in group settings. They can also be conducted via email or as surveys. The interviews center on the following open-ended questions:

- Has the use of the new technology in the source environment been successful? In what ways?
- What characteristics of the source environment have facilitated the successful use of the new technology?

- Can you think of anything about the source environment that, if it were not true, might have made it difficult or impossible to apply the new technology?

From the interview notes, a preliminary set of domain factors is identified. This preliminary set is then presented to a subset of the original interviewees, who provide comments and optionally add new factors.

In this example, interviews were the major source of data on domain factors. However, other information can also be accessed. Given an historical database, the costs, reliability, and size of projects built using the new technology can be compared to projects that did not use the new technology. In this example, this was not done. Although the source environment had extensive data for many of these domain factors, since the target environment was only beginning a measurement program, these factors could not be applied to the target environment, as the evaluation process requires.

For our example process, the following domain factors were included in the final set:

> ***Personnel turnover.*** Low turnover of managers and experienced analysts was cited as important to the successful use of the new technology. That is, the high-level people who drive the use of the method have been the same, more or less, over time. Also, low turnover at the developer level is also helpful, because it lessens the need for constant training in the method.
>
> ***Skills of the manager.*** It had been important that the managers of the source environment had been people who were both committed to evolutionary improvement and who knew the higher levels of the organization. They could use this knowledge to get support for continued use of technologies such as the new technology. At the same time, the source environment had not undergone tremendous amounts of scrutiny from higher levels of management. That is, the leaders of the source environment knew how to manipulate higher management to get what they needed without drawing unwanted attention.
>
> ***Role of the manager.*** It was important that the manager of the source environment was also involved in project management.
>
> ***Application domain homogeneity.*** It had been helpful for the source environment that its projects had consistently been in a very homogeneous application domain (ground support software for unmanned satellites).
>
> ***Outside researchers.*** The role of the local university (University of Maryland, College Park) in the source environment had been an advantage because it provided an outsider's view to projects. It lent a measure of outside credibility to the source environment's work, and in some cases it provided a source of inexpensive labor.
>
> ***Attitudes toward measurement.*** There should be a belief that measurement is important

amongst the project management and one level of management above that. It also helps if the managers already have some experience with measurement. They must also have a specific focus, goal, or need that they want to address through measurement, other than just to collect data.

Feedback. There must be feedback on the data collected to both developers and managers.

Trust. There must be trust between developers and managers that the data will not be used in nefarious ways.

Funding. There must be adequate funding to support the new technology use, including the building of measurement infrastructure and the useful analysis of measurement data, which includes formulating and testing hypotheses. Further, the nature of the projects should ensure that adequate funding continues into the future.

4.2.3 Determining Domain Values in the Target Environment

The questions in the Operational Domain Model are sent to the management of the target environment, and the responses are gathered (e.g., telephone call, email, video conference, field trip). The responses are recorded qualitatively, then coded into the defined possible values for each variable in the Operational Domain Model. Once each variable has a value, the Operationalization domains are applied to get the values of the domain factors as they applied to the target environment. The values of the operational variables, as well as the resulting domain factor values for both the target environment and the source environment, are shown in Table 4-1.

4.2.4 Analysis of the Two Domain Models

As can be seen from Table 4-1, there are four domain factors that have different values in the two environments: application domain, outside researchers, feedback, and funding. These are the areas that need to be given special consideration when planning the transfer of one technology (i.e., the GQM technology in this case) from the source to the target environment.

The source environment historically had dealt with development projects within the same application domain: ground support software for unmanned spacecraft. In addition, most of the projects had been complete development projects, including the entire development cycle from requirements through all phases of testing. The projects had been fairly homogeneous in this respect. Source environment personnel interviewed in the domain modeling process said that this characteristic aided the use of the new technology because the results of that new technology could often be reused between projects with little modification. In the target environment, on the other hand, projects have mostly been within the same general application domain (i.e., guidance, navigation, control of manned spacecraft), but have taken on different types of work (i.e. analysis, development, testing) They were not, for the most part, complete development projects.

Operational Variables	**Values for Target Environment**	**Domain Factors**	**Values for Target Environment**	**Values for Source Environment**
Advocate turnover	low	Personnel turnover	low	low
High-level turnover	low			
Low-level turnover	low			
Ability to gain support	high	Skills of the manager	high	high
Scrutiny	low			
Role of the manager	yes	Role of the manager	yes	yes
Application domain homogeneity	medium	***Application domain homogeneity***	***medium***	***high***
Outside researchers	funded	***Outside researchers***	***funded***	***involved***
Managers	high	Attitudes toward measurement	high	high
Senior managers	medium			
Goals	yes			
Manager feedback	sometimes	***Feedback***	***medium***	***high***
Developer feedback	sometimes			
Evaluation	no	Trust	high	high
Developer trust	yes			
Uncollected data	no			
Present	no	***Funding***	***unsure***	***secure***
Future	no			

Table 4-1. Values of Operational Variables and Domain Factors [2]

In this initial phase, the transfer of the new technology was not greatly affected by this domain difference because the experience domain analysis study concentrated on only one project. In other words, the study team was concerned primarily with introducing the new technology concepts on one particular project rather than spreading the use of that technology to all new projects. However, in later phases of this technology transfer effort, this could become a concern. One possible way to address the heterogeneity of the target environment's projects is to use higher-level, more abstract, experience domain analysis goals and questions that each project can tailor to their own needs and use to choose appropriate metrics.

Another domain factor that differed was the availability and role of outside researchers. Faculty and students from the University of Maryland have played an active role in measurement,

---

[2] Bold italicized items represent domain factors that differed between the source and target environments.

including the development and use of the new technology, at the source environment. The role of these academics in the source environment had been an advantage because it provided an outsider's view of projects. It lent a measure of outside credibility to the source environment's work, and it often provided expertise and personnel for measurement planning and analysis, thus allowing NASA personnel to focus on meeting project requirements. The target environment had no such collaborative contacts, and no academics or other outside researchers were involved in its work, although their division funded some academic research. However, the experience domain analysis study team filled this role for the target environment's work with the new technology. The study team served as an outside source of information and guidance, and also provided much of the effort needed to do the analysis on the new technology.

Measurement process feedback to both developers and managers is essential to the successful use of the new technology. The feedback is important for two reasons. First, it helps developers and maintainers see the benefits of the measurement process, securing their support for measurement. Second, it helps to ensure that the data being gathered is relevant to those it is meant to help. Developers and managers who come to rely on measurement data will speak up when it is no longer meeting their needs. In the target environment, feedback was not always provided to developers and managers. To overcome this difference, the study team viewed feedback as part of the new technology approach, rather than a necessary condition for its use. In other words, the importance of, and mechanisms for, providing feedback are packaged as part of the technology being transferred. For instance, analyses and reports that provide such feedback were included in the recommendations that resulted from the initial new technology analysis.

The last difference between the two domains was the security of funding. The source environment personnel that were interviewed acknowledged that a measurement program (including the use of GQM) required resources, and the funding needed to be secure to allow GQM analyses to be thorough and complete. Funding was not so secure within the target environment and was dependent on projects being assigned to them with enough resources to support measurement planning. In this initial phase, this difference was mitigated by the fact that the study team provided the resources for much of the work required. However, this could become a major factor in the long-term success of the new technology at the new location.[3]

---

[3] In fact, the lack of funding in the target environment indeed turned out to be a problem in the continued transfer of this technology from the source environment to the target environment.

Chapter 5. Conclusion

As indicated in the introduction, if domains can be identified, then there is a possibility for experience sharing and mutual collaboration for refining processes and evaluating new technologies. This organization is referred to as the Experience Factory [Basili1994]. The role of experience domain analysis in the context of the experience factory can be summarized as follows:

- Experience domain analysis allows the Experience Factory to identify the limitations of reusing or sharing process information (e.g. process evaluation, tool evaluation); in other words, the domain of validity of process-related information.

- Experience domain analysis allows the Experience Factory to determine if different projects are comparable with respect to quality evaluation criteria (e.g., two rojects belong to two different domains characterized by a very different level of requirement stability).

- Experience domain analysis allows the Experience Factory to determine if data can be pooled together to build better quantitative models of software development.

Appendix A. GQM Domain Model

The model presented in this document describes the various factors that have affected the successful use of GQM at the SEL. This data was gathered, through interviews, with several of the SEL principals. The model is organized according to the formalism defined by Andreas Birk of the Fraunhofer IESE for domain modeling for software technologies.

The model has four parts. The first part is a header, which contains some identification information about the model. The second part, the intuitive domain model, lists a set of domain factors, which are attributes of the SEL environment that have impacted the use of GQM. In the intuitive domain model, each domain factor is given an intuitive, conceptual definition and a set of possible values. The third part, the operationalization, defines how the domain factors defined intuitively map to concrete values that can be measured. The fourth and final part, the operational domain model, defines the actual data that must be collected from an organization in order to perform a domain comparison with this model.

1 Header

Technology: Goal/Question/Metric method
Task: Design of a software measurement program
Goal: Appropriateness and efficiency of metrics set
Viewpoint: Project manager/planner
Environment: Software Engineering Laboratory at NASA/GSFC

2 Intuitive Domain Model

Actual values for each domain factor are in bold.

Domain factor: Personnel turnover
Definition: The rate at which personnel are added to, and leave, the organization
Possible values: high, medium, ***low***

Domain factor: Skills of the manager
Definition: The ability of the manager to obtain needed support and resources from upper management without drawing unwanted scrutiny and attention
Possible values: ***high***, medium, low

Domain factor: Role of the manager
Definition: Indicates whether or not the manager of the organization

has a role in project management
Possible values: ***yes***, no

Domain factor: Application domain homogeneity
Definition: The degree to which the different projects in the organization are similar in terms of application domain
Possible values: ***high***, medium, low

Domain factor: Outside researchers
Definition: The degree to which outside researchers (e.g., cademics) are involved in any way in the organization
Possible values: none, funded, ***involved***

Domain factor: Attitudes toward measurement
Definition: The degree to which the management of the organization believe in and understand the importance of measurement
Possible values: ***high***, medium, low

Domain factor: Feedback
Definition: The degree to which data is fed back to developers and managers
Possible values: ***high***, medium, low

Domain factor: Trust
Definition: The amount of trust between developers and managers that the data will not be used in nefarious ways
Possible values: ***high***, medium, low

Domain factor: Funding
Definition: The adequacy of present and future funding to support use of the new technology
Possible values: ***secure***, unsure

3 Operationalization

Domain factor: Personnel turnover
Operationalization: Functional
Based on: advocate turnover, high-level turnover, low-level turnover
Model:

```
if advocate turnover is high then
    personnel turnover is high
else
    if high-level turnover is high then
        personnel turnover is medium
    else
```

```
if low-level turnover is high then
    personnel turnover is medium
else
    personnel turnover is low
```

Rationale: If the advocates of measurement in the organization tend not to be around very much (i.e , advocate turnover is high), then this has a big effect on the success of the measurement program, so the effect of personnel turnover is said to be high. If the advocates stick around, but there is a great deal of turnover at the management (i.e., high-level turnover) and developer (i.e., low-level turnover) levels, then personnel turnover has a medium effect on success. Finally, if all types of turnover are low, then the impact of personnel turnover is low.

Domain factor: Skills of the manager
Operationalization: Functional
Based on: ability to gain support, scrutiny
Model:

Ability to gain support → **Scrutiny** ↓	*high*	*medium*	*low*
low	***high***	***medium***	***low***
medium	***high***	***medium***	***low***
high	***medium***	***low***	***low***

Rationale: The amount of unwanted scrutiny that the organization suffers is not really significant unless it is high. Otherwise, the skills of the manager factor is determined by the manager's ability to gain support from upper levels.

Domain factor: Role of the manager
Operationalization: Direct

Domain factor: Application domain homogeneity
Operationalization: Direct

Domain factor: Outside researchers
Operationalization: Direct

Domain factor: Attitude toward measurement
Operationalization: Functional

Based on: managers, senior managers, goals
Model: if **goals** is *no* then
    attitude toward measurement is ***low***
else

managers → **senior managers** ↓	*high*	*medium*	*low*
high	***high***	***medium***	***medium***
medium	***high***	***medium***	***low***
low	***medium***	***medium***	***low***

Rationale: If there are no specific measurement goals (i.e., goals is no), then this indicates that management really does not understand the importance of measurement, and thus the attitude toward measurement factor is rated low, no matter what the managers' attitudes are. Otherwise, both sets of managers' and senior managers' attitudes are important because first-line managers must provide the motivation, while senior managers provide the resources. However, some measurement can be done without the senior managers' support, so the managers' attitude is weighted slightly higher.

Domain factor: Feedback
Operationalization: Functional
Based on: manager feedback, developer feedback
Model:

manager feedback → **developer feedback** ↓	*always*	*sometimes*	*never*
always	***high***	***medium***	***medium***
sometimes	***medium***	***medium***	***medium***
never	***medium***	***medium***	***low***

Rationale: If both manager feedback and developer feedback from measurement occurs on a regular basis, then feedback is rated high. Similarly, if neither group ever gets feedback, then feedback is low. Any other combination is medium.

Domain factor: Trust
Operationalization: Functional

Based on: evaluation, developer trust, uncollected data

Model:

```
if (evaluation is no) and (developer trust is yes) and
(uncollected data is no) then
    trust is high
else
    if (evaluation is yes) and (developer trust is no) and
    (uncollected data is yes) then
        trust is low
    else
        trust is medium
```

Rationale: If data is not used to evaluate developers (evaluation is no) and the developers believe that (developer trust is yes) and lack of trust has never affected decisions about data collection (uncollected data is no), then trust is rated high. If exactly the opposite of that is true, then trust is low. Anything in between is rated medium.

Domain factor: Funding
Operationalization: Functional
Based on: present, future

Model:

```
if (present is yes) and (future is yes) then
    funding is secure
else
    funding is unsure
```

Rationale: Funding is considered secure only if it is currently adequate to support measurement and it is reasonably likely to continue to be adequate.

4 Operational Domain Model

Domain factor: Personnel turnover
Operationalization: Functional
Instruments: Interview questions:

- **high-level turnover**
  How high is turnover in the leadership of the organization and their managers? (*high/low*)
- **low-level turnover**
  How high is turnover among developers and analysts in the organization? (*high/low*)
- **advocate turnover**
  Who are the people who would be important in applying GQM in the future and what is the likelihood that they will turn over in the near future? (*high/low*)

Domain factor: Skills of the manager
Operationalization: Functional
Instruments: Interview questions:

- **ability to gain support**
  Is the leadership knowledgeable about his/her superior levels of management and skillful in working those levels to gain support for the organization? (*high/medium/low*)
- **scrutiny**
  What level of scrutiny does the organization get from upper levels of management? (*high/medium/low*)

Domain factor: Role of the manager
Operationalization: Direct
Instruments: Interview questions:

- **role of the manager**
  Is the manager involved in project management? (*yes/no*)

Domain factor: Application domain homogeneity
Operationalization: Direct
Instruments: Interview questions:

- **application domain homogeneity**
  How homogeneous is the application domain of projects within the organization? (*high/medium/low*)

Domain factor: Outside researchers
Operationalization: Direct
Instruments: Interview questions:

- **outside researchers**
  Are there opportunities for university collaboration in the organization? Are there other independent research organizations with which the organization could collaborate? To what extent could the organization itself be considered an independent research organization? (*none, funded, involved*)

Domain factor: Attitudes toward measurement
Operationalization: Functional
Instruments: Interview questions:

- **managers**
  Do the managers of the organization believe that measurement is important? (*high/medium/low*)
- **senior managers**
  Does the level of management above the organization believe that measurement is important?

(*high/medium/low*)

- **goals**
  Are there specific goals or needs to be addressed through measurement? (*yes/no*)

Domain factor: Feedback
Operationalization: Functional
Instruments: Interview questions:

- **manager feedback**
  Do managers currently get feedback on the results of data collection and analysis on a regular basis? (*always/sometimes/never*)
- **developer feedback**
  Do developers currently get feedback on the results of data collection and analysis on a regular basis? (*always/sometimes/never*)

Domain factor: Trust
Operationalization: Functional
Instruments: Interview questions:

- **evaluation**
  Does the management have any plans or desire to use the collected data to evaluate individual developers? (*yes/no*)
- **developer trust**
  Do the developers trust that the data will not be used to evaluate them personally or in any other way that might cause them harm? (*yes/no*)
- **uncollected data**
  Is there otherwise useful data that is not collected mainly because of staff concern that it will be misused? (*yes/no*)

Domain factor: Funding
Operationalization: Functional
Instruments: Interview questions:

- **present**
  Do current project budgets, in general, include resources for developing, maintaining, and using a measurement program? (*yes/no*)
- **future**
  How do the projects fit into the larger organization’s mission? Do they provide revenue or other concrete benefits to the larger organization? Are they perceived to be critical projects? Can it be expected that future project budgets, in general, will include resources for developing,

maintaining, and using a measurement program? (*yes/no*)

Abbreviations and Acronyms

GQM	Goal/Question/Metric
GSFC	Goddard Space Flight Center
IBM	International Business Machines
ISC	Information Systems Center
JSC	Johnson Space Center
NASA	National Aeronautics and Space Administration
OSMA	Office of Safety and Mission Assurance
RDL	Rapid Development Laboratory
SEL	Software Engineering Laboratory

References

[Basili1994] Basili V., et al, The Experience Factory, Encyclopedia of Software Engineering, Wiley & Sons, Inc., 1994.

[Birk1997] Birk, A., "Modeling the Application Domains of Software Engineering Technologies”, IESE-Report No. 014.97/E, August 1997.