Using the Shlaer-Mellor Method to Support a Level 4 Software Process

Sanjiv Gossain

Project Technology Inc.
10940 Bigge Street
San Leandro, CA 94577-1123
510 567-0255
http://www.projtech.com

1. Introduction

1.1. Software Process

A software process is a set of activities, methods, and practices that guide people in the production of software. It is widely recognized that organizations with a mature software development process are able to consistently produce high quality, reliable software. In addition to the technical benefits, the business case for a mature process cannot be ignored, with one organization recently reporting significant financial advantages in improving their software development process [Dion93]. Improving a software process within an organization can also aid in helping understand, and achieve, the quality requirements of software quality certification, such as ISO 9000-3.

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) [Humphrey89, Paulk93a] is a recognized framework that organizes the process maturity of software development into five levels. These levels provide a scale with which organizations may measure their software process capability, and also provide a means to identify their strengths and weaknesses and hence prioritize improvement efforts.

There are a number of factors that contribute to an organization's process maturity. They are: the development method used, the structure of the organization, the motivation and commitment of the people involved, and the discipline of the software process management within the organization. The development method used is only one factor, but is also one of the most crucial. It is essential, therefore, that the development method is able to provide the support necessary for a defined, repeatable process.

This paper describes how the Shlaer-Mellor Method for system development supports a software development process that is at the *Managed Level* (level 4) of the CMM. The unique properties of rigor and formality in the Shlaer-Mellor method make it suited to producing software in a highly repeatable way, and we shall see this through a detailed examination of each of the key areas up to level 4 as defined by the CMM.

1.2. The Capability Maturity Model

Each maturity level is a step towards achieving a mature software process, providing a set of goals which, when satisfied, places an organization at the next level of maturity. Except for level 1, each level is decomposed into several key process areas that an organization should focus on to improve its software process. Each key area is described in terms of key practices [Paulk93b] that contribute to meeting the goals. These key practices require that certain activities be followed in order to establish a process capability.

The five levels, their characterizations, and their key process areas are:

1. Initial. The software process is characterized as ad hoc. Few processes are defined, and success depends on individual effort. Key process areas: none.

2. Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. Key process areas: Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance, Software Configuration Management.

3. Defined. The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. Key process areas: Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, Peer Reviews.

Published in *Software Engineering Notes*, ACM Press, Volume 19, Number 3, July 1994.

4. Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. Key process areas: Quantitative Process Management, Software Quality Management.

5. Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. Key process areas: Defect Prevention, Technology Change Management, Process Change Management.

For an organization to improve its software process, it must first recognize its current level, identify areas for process improvement (using the key process areas list), and then go about improving those process areas. To do this, the key practices are used as a guideline for establishing improvement, and the activities required to implement a process capability are undertaken.

1.3. The Shlaer-Mellor Method and the Levels of Maturity

In order to show how the Shlaer-Mellor Method [Shlaer88, Shlaer91, Shlaer93a] supports a level 4 software process, we shall examine each of the activities of the key process areas for levels 1 to 4. A separate section is devoted to each key area, and at the beginning of the section there is a summary table characterizing those activities that are method related and those that are not. For each activity that is directly method related, we detail those aspects of the Shlaer-Mellor method that provide direct support in realizing the activity. Pointers to the literature where the reader may find more details are provided where necessary. Those activities that are not method related are more directly affected by the procedure and policy in place in the organization rather than by the method used[1]. The activities lists are extracted verbatim from Appendix C of [Paulk93b].

2. Level 2: Repeatable

2.1. Requirements Management

The goals of requirements management are to allocate requirements to the software system, to establish a baseline to control software engineering and management, and to ensure that software plans, products, and activities are kept consistent with the software requirements.

Activities Supported

2. *The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.*

The requirements are used as a basis for beginning the analysis process using the Shlaer-Mellor Method. It is these requirements that drive the creation of the work products, and assist in establishing the correctness of the analysis and design work products. The use of these requirements permeates the entire analysis process, and are reflected in the work products created. See [Shlaer93a].

During the design stages, the requirements that affect issues such as system performance are addressed. Work products created during design include the task communication model, thread of control chart, and various architecture-specific models. See [Shlaer93a].

The project matrix, a project management and planning tool, is structured around the work products of the method. With the link between work product and requirements clearly defined it is possible to trace the meeting of requirements to plans. [Shlaer84].

1. It is important to note, however, that the rigors of the Shlaer-Mellor Method provide the infrastructure necessary to more easily introduce the procedures and policies required in each key area.

Requirements Management

Activity	Method Related	Procedure & Policy
1. Software engineering group reviews requirements before they are incorporated into the software project.		◆
2. Software engineering group uses requirements as basis for plans.	◆	
3. Changes to allocated requirements are reviewed and incorporated into the software project.		◆

2.2. Software Project Planning

The goals of software project planning are to document software estimates, project activities, and commitments for planning and tracking the project, and to ensure that affected groups and individuals agree to their project commitments.

Activities Supported

2. *Software project planning is initiated in the early stages of, and in parallel with, the overall project planning.*

The domain chart depicts the dependencies between the different subject matters in the system and the information workflow diagram of the project matrix [Shlaer84] describes the dependencies within one domain. They can, therefore, both be used in assisting in the software project planning in parallel with the overall project planning. For further information see Chapters 7 and 8 of [Shlaer91].

5. *A software lifecycle with predefined stages of*

Software Project Planning

Activity	Method Related	Procedure & Policy
1. Software engineering group participates in proposal team.		◆
2. Planning initiated early.	◆	
3. Software engineering group participates with other affected groups in overall project planning.		◆
4. Commitments are reviewed with senior management.		◆
5. Software lifecycle is defined	◆	
6. Software development plan is developed according to procedure.		◆
7. Plan is documented.	◆	
8. Software work products are identified.	◆	
9. Estimates for size are derived according to procedure.	◆	
10. Estimates for efforts and costs are derived according to procedure.	◆	
11. Estimates for project's critical computer resources are derived according to documented procedure.	◆	
12. Schedule derived according to documented procedure.	◆	
13. Risks associated with cost, resource, and schedule are identified, assessed, and documented.	◆	
14. Plans for software engineering facilities and support tools are approved.		◆
15. Software planning data are recorded.		◆

manageable size is identified or defined.

The Shlaer-Mellor Method is founded upon the belief that a software development strategy requires a clearly-defined lifecycle, and it therefore has a well-defined software lifecycle in which it can be used. It is described, in overview form, in [Shlaer93a].

7. *The plan for the software project is documented.*

The domain chart [Shlaer91], along with the domain mission statements and bridge descriptions formalize the documentation of the project's purpose, scope, and goals, which are a key part of the software plan.

The project matrix, a project management model of a software development project, and its associated models provide a means of documenting the plan for the software project. It clearly describes each stage in the process and establishes the inter-relationships and dependencies between the various tasks. The information flow diagram represents the fundamental information flow constraints of the work and brings out possibilities for parallel work. The matrix is useful for establishing accountability and control, for tracing requirements and planning. It is described in considerable detail in [Shlaer84].

8. *Software work products that are needed to establish and maintain control of the software project are identified.*

In the Shlaer-Mellor Method every step in the process has a well-defined work product which is required to be produced, and the relative completion of these work products is a good indication of the progress of the project. During the analysis phase, these work products are: the domain chart for a system, the project matrix, the object information models, state models, and process models. Other models created at the domain, subsystem, and object levels during OOA are derivable from existing models. In addition to these graphical models, there is extensive supporting documentation that must be created. The primary OOA models (information models, state models, and process models) are the key work products used in establishing control of the software project in tandem with the use of the project matrix.

During the design phase they include the task communication diagram, task templates, dependency diagrams, inheritance diagram, class diagram, class structure charts, and class templates. See [Shlaer93a, Shlaer93c] for more details on work products of the method.

9. *Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.*

Guidelines on the use of the method, and estimates upon the length of time needed to produce work products have been prepared using data collected over the numerous projects that have used the Shlaer-Mellor Method. This data, which can be used in estimating work product size, can be found in [Montrose93]. Such guidelines can be incorporated into a procedure to be used by the organization.

10. *Estimates for the software project's effort and costs are derived according to a documented procedure.*

Guidelines on estimating a software project's effort and costs have been prepared and can be found in [Montrose93]. They are based, in part, on the guidelines in estimating size of work products as this will affect their size and cost. These guidelines can be incorporated into a documented procedure.

11. *Estimates for the project's critical computer resources are derived according to a documented procedure.*

    The separation of the software architecture from the analysis means that the subject matter of critical computer resources in the target environment can be studied and estimated effectively. The CPU time, memory capacity, and communications channel capacity, for example, are all areas that are pertinent to the software architecture. The well-defined relationships between the architecture and the other domains means that the size of work products, the operational processing load, and communications traffic an all be taken into consideration when arriving at estimates. Such activities are all part of the Recursive Design process.

12. *The project's software schedule is derived according to a documented procedure.*

    The project matrix and information flow diagram clearly describe the dependencies between different steps and work products in the Shlaer-Mellor Method. Using these tools it is possible to construct a schedule based on these dependencies.

13. *The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.*

    The partitioning of the system into different subject matters (domains) means that the risks associated with each of the different subject matters can be isolated and assessed on their own basis. The dependencies between domains, as highlighted on the domain chart, help to identify the potential dependencies between these risks and can thus be used as tool to minimize risks.

2.3. Software Project Tracking and Oversight

The goals of the software project tracking and oversight key area are to track actual results and performances against the software plans, taking corrective actions when there is significant deviation, and changes to software commitments are agreed to by the affected groups and individuals.

Activities Supported

1. *A documented software development plan is used for tracking the software activities and communicating status.*

    The project matrix can be used to effectively communicate the status of a project. Each cell of the matrix identifies one or more work products that must be created during the development in conjunction with completion of a stage of the development. Therefore, the status of each aspect of the development can be recorded in a corresponding cell, thus providing a visual representation of the project status. [Shlaer84]

5. *The size of the software work products (or size of the changes to the software work products) are tracked, and corrective actions are taken as necessary.*

    Each cell of the project matrix represents one or more work products of the development method and it can therefore be used to track the size of the software work products. By placing the work product size for each work product in the corresponding cell, it is possible to closely monitor their size and take corrective actions as necessary.

6. *The project's software effort and costs are tracked,*

Software Project Tracking and Oversight

Activity	Method Related	Procedure & Policy
1. Software development plan used for tracking.	◆	
2. Software development plan revised according to a documented procedure.		◆
3. Project commitments and changes to commitments are reviewed with senior management according to a documented procedure.		◆
4. Approved changes to commitments are communicated to members of the software engineering and other related group.		◆
5. Size of software work products and size of changes to work products are tracked.	◆	
6. Software effort and costs are tracked.	◆	
7. Critical computer resources are tracked.	◆	
8. Software schedule is tracked.	◆	
9. Software engineering activities are tracked with corrective actions taken as necessary.	◆	
10. Risks associated with cost, schedule, resource, and technical aspects are tracked.	◆	
11. Actual measurement data and replanning data are recorded.		◆
12. Software engineering group conducts periodic reviews to track progress against the software development plan.		◆
13. Formal reviews are conducted at selected milestones.		◆

and corrective actions are taken as necessary.

Each cell of the project matrix represents one or more work products of the development method and it can therefore be used to track the software effort and costs. By placing the estimated effort for each stage in the corresponding cell, and then entering in the actual effort as the project proceeds, it is possible to closely monitor the project's cost, and thus take corrective actions as necessary.

7. *The project's critical computer resources are tracked, and corrective actions are taken as necessary.*

   The project matrix can be used to track computer resources. Through use of the cells to represent usage of computer resources, it is possible to track their usage, and take corrective actions as necessary.

8. *The project's software schedule is tracked, and corrective actions are taken as necessary.*

   The project matrix can be used to track the software schedule and determine if the project is on schedule or not. This is done by entering the expected, and actual, completion dates of work products in the appropriate cells of the project matrix.

9. *Software engineering technical activities are tracked, and corrective actions are taken as necessary.*

   The method has a well-defined set of activities for each stage of the lifecycle [Shlaer93c]. Using the project matrix as a basis for tracking overall progress (based on work products and stages of the lifecycle) means that the activities of each stage can also be monitored.

10. *The software risks associated with cost, resource, schedule, and technical aspects of the project are tracked.*

    The project matrix organizes activities around domains and work products in each domain. This means that the software risks, separated into their respective subject matters in the domain chart, can be easily tracked on a work product by work product basis.

2.4. Software Subcontract Management

The goals of software subcontract management are to select qualified software subcontractors, to agree to commitments, and to ensure that the subcontractor's results and performance are tracked against those commitments.

Activities Supported

None of the activities performed for Software Subcontract Management are method related, but are affected by procedure and policy issues of the organization, and are therefore not listed.

2.5. Software Quality Assurance

The goals of the software quality assurance (SQA) key area are to plan SQA activities and to verify that software products and activities adhere to the applicable standards, procedures, and requirements. Non-compliance issues that cannot be resolved within the software project should be addressed by senior management.

Software Quality Assurance

Activity	Method Related	Procedure & Policy
1. SQA Plan is prepared according to documented procedure	◆	
2. SQA group activities performed in accordance to SQA plan		◆
3. SQA group participates in preparation and review of software development plan, standards and procedures		◆
4. SQA group reviews software engineering activities to verify compliance		◆
5. SQA group audits designated software work products to verify compliance		◆
6. SQA group reports results to the software engineering group		◆
7. Deviations in work products and activities are documented and handled according to a documented procedure		◆
8. SQA group conducts periodic reviews of activities and findings with customer's SQA personnel		◆

Activities Supported

1. *An SQA plan is prepared for the software project according to a documented procedure.*

   The Shlaer-Mellor Method is based on a formalism, a set of rules [Lang93], allowing its work products to be verified both statically and dynamically. The product development and product verification activities that need to be included as part of the SQA plan are therefore based on assuring that the models conform to these rules. The formalism addresses the technical aspects of the SQA plan, leaving only the organizational issues to be addressed, by the management.

2.6. Software Configuration Management

The goals of the software configuration management key area are to identify those software work products to be controlled, to ensure that changes are tracked, and to ensure that affected groups and individuals are informed of the status and content of software baselines.

Activities Supported

All the activities performed for Software Configuration Management are organizational issues that are reflected in the procedure and policy in place within the organization, and are therefore not listed.

3. Level 3: Defined

3.1. Organization Process Focus

The goals of the organization process focus key area are to coordinate software process development, identify strengths and weaknesses of the software process, and plan improvement activities across the organization.

Activities Supported

All of the activities performed for this key areas are organizational activities that are reflected in the procedure and policies in place within the organization, and so are not listed here.

3.2. Organization Process Definition

The goals for the organization process definition key area are to develop and maintain a standard software process for the organization, and to collect, review, and make available information on the process.

Organization Process Definition

Activity	Method Related	Procedure & Policy
1. Organization's standard software process is developed and maintained according to documented procedure	◆	
2. Organization's standard software process is documented		◆
3. Descriptions of software lifecycles approved for use by the project are documented and maintained		◆
4. Guidelines and criteria for tailoring of the organization's standard process are developed and maintained		◆
5. Organization's software process database is established and maintained		◆
6. A library of software process-related documentation is established and maintained		◆

Activities Supported

1. *The organization's standard software process is developed and maintained according to a documented procedure.*

The Shlaer-Mellor Method is based on a repeatable, clear, well-defined software process. Each stage in the process is formal and has distinct completion criteria. Therefore, the internal interfaces between the various software disciplines (e.g. design and code, coding and testing) are clear and unambiguous. Analysis consists of a number of models, of which the most pivotal are the domain chart, object information models, state models, and process models. The relationships between each of these models are clear and described in detail in [Shlaer91].

The method is based upon a translation of analysis work products into design and implementation, using a common software architecture, such that there is a uniform structure across all domains [Shlaer93a, Shlaer93b].

3.3. Training Program

The goals of the training program key area are to plan, and arrange the provision of, training activities necessary for individuals in the software engineering group and software-related groups.

Training Program

Activity	Method Related	Procedure & Policy
1. Each project develops and maintains a training plan	◆	
2. Training plan is developed and revised according to a documented procedure		◆
3. Training is performed in accordance with training plan		◆
4. Courses prepared are developed and maintained according to organization standards		◆
5. Waiver procedure is established to determine whether individuals already possess the knowledge and skills		◆
6. Records of training are maintained		◆

Activities Supported

1. *Each software project develops and maintains a training plan that specifies its training needs.*

An established method that has been proven throughout the industry, the Shlaer-Mellor Method is thoroughly supported by authoritative and specialized training programs. With these programs, a defined training path can be established for all persons involved with the project, with training directed to the level appropriate for specific personnel (i.e., one level for practitioners and another level for managers).

3.4. Integrated Software Management

The goals of the integrated software management key area are to provide a defined software process for the project, and to ensure that this is adhered to.

Integrated Software Management

Activity	Method Related	Procedure & Policy
1. Defined software process is developed by tailoring standard process		◆
2. Process is revised according to a documented procedure		◆
3. Software development plan is developed according to procedure	◆	
4. Project managed in accordance with software process		◆
5. Organization's process database used for planning and estimating		◆
6. Size of software work products is managed according to documented procedure	◆	
7. Software effort and costs are managed according to documented procedure	◆	
8. Computer resources are managed according to documented procedure	◆	
9. Dependencies and critical paths managed according to documented procedure	◆	
10. Risks are identified, assessed, and managed according to documented procedure		◆
11. Reviews are periodically performed to bring results in line with projected needs	◆	

Activities Supported

3. *The project's software development plan, which describes the use of the project's defined software process, is developed and revised according to a documented procedure.*

   Refer to activities 6 & 7 of Section 2.2 Software Project Planning, and activities 1 & 2 of Section 2.3 Software Tracking and Oversight.

6. *The size of the software work products (or size of changes to the software work products) is managed according to a documented procedure.*

   The partitioning strategy used in the Shlaer-Mellor Method can take advantage of reuse of whole subject matters (domains) across a number of projects, providing great leverage of the analysis. As such, the size of significant work products could be affected by the amount of reusability based on work from previous projects. See Chapter 6 of [Shlaer91].

   Refer also to activity 9 of Section 2.2 Software Project Planning and activity 5 of Section 2.3, *Software Tracking and Oversight.*

7. *The project's software effort and costs are managed according to a documented procedure.*

   Refer to activity 6 of Section 2.3 *Software Project Tracking and Oversight.*

8. *The project's critical computer resources are managed according to a documented procedure.*

   Refer to activity 7 of Section 2.3 *Software Project Tracking and Oversight.*

9. *The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.*

   Due to the explicit software development strategy employed using the Shlaer-Mellor Method, it is a straightforward task to identify the critical dependencies, and critical paths of a software project. The domain chart identifies dependencies between different domains (system wide partitions), and the project matrix identifies the dependencies between the various subsystems, and between the required analysis models of each subsystem. These dependencies can thus be managed effectively. See also [Shlaer84, Shlaer93a].

11. *Reviews of the software project are periodically performed to determine the actions needed to bring the software project's performance and results in line with the current and projected needs of the business, customer, and end users, as appropriate.*

    The analysis models, because of their formalism and comprehensibility, can be effectively used to communicate the analysis results with customers and end-users as appropriate.

3.5. Software Product Engineering

The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.

Activities Supported

1. *Appropriate software engineering methods and tools are integrated into the project's defined software process.*

   The Shlaer-Mellor Method has a clear set of tasks that are integrated into the development process. The sequence of steps and the work products that they generate are clearly defined, and documented. The

Software Product Engineering

Activity	Method Related	Procedure & Policy
1. Software engineering methods are integrated into process	◆	
2. Requirements are developed, maintained and verified by systematically analyzing allocated requirements	◆	
3. Design is developed, maintained, and verified according to process	◆	
4. Code is developed and maintained according to process	◆	
5. Software testing is performed according to process	◆	
6. Integration testing is performed according to process	◆	
7. System and acceptance testing are performed	◆	
8. Documentation is developed according to process	◆	
9. Data on defects are collected and analyzed according to process	◆	
10. Consistency maintained across work products	◆	

domain chart is created first, which structures the system into different subject matters (domains). From these an analysis is carried out of each domain, noting the requirements and dependencies that domains place upon each other. Within each domain the analysts build an object information model, state models, process models, and a number of other engineered work products. There are numerous CASE tools that support the construction of these models.

Upon completion of the analysis of each domain, the design architecture is applied uniformly across all domains to provide a uniform system structure, and management of control and data. From these structures, the software can be written, integrated, and tested as appropriate. See [Shlaer93c].

2. *The software requirements are developed, maintained, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.*

The Shlaer-Mellor Method requires that the problem space be analyzed using a systematic, rational process. In order to achieve this there are well-defined work products produced at distinct stages of the method (e.g. domain chart, object information models, state models, process models, software architecture, design rules, code templates).

The meeting of functional requirements is verified through execution of the analysis models. These models can be reviewed with customers and end users to verify correctness prior to moving to design. Once implemented, the code can be tested to provide final verification of the system.

3. *The software design is developed, maintained, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.*

The software design architecture is a separate subject matter in the Shlaer-Mellor Method, and, as such, can be analyzed early within the constraints of the software lifecycle. The architecture is designed to meet the code space, performance, and implementation requirements of the system, and forms the basis for detailed design and coding. The architecture addresses those requirements that deal with performance of the system. For more details see Chapter 9 of [Shlaer91].

4. *The software code is developed, maintained, documented, and verified, according to the project's defined software process, to implement the software requirements and software design.*

The Shlaer-Mellor Method advocates a translation of the analysis models to code using a software architecture, which by definition means that there is an order imposed in which the code is produced. The sequence in which units of code are arrived at, tested, and integrated is specified by the Recursive Design process which addresses issues such as domain dependence, integration, and test issues. For more information see [Shlaer93a, Shlaer93b].

5. *Software testing is performed according to the project's defined software process.*

Testing plays a distinct role in the Shlaer-Mellor Method. There is the verification of analysis through execution of the analysis models to verify correct functionality with customers and end users. When moving to an implementation the Recursive Design process can assist in the definition of testing of code, and test results are compared to their counterparts in the simulation of the models (where appropriate) to verify correct functionality.

6. *Integration testing of the software is planned and performed according to the project's defined software process.*

The software design architecture and the Recursive Design aspect of the Shlaer-Mellor Method means that there is a plug compatibility between the analysis work products and the architecture. The architecture, which can be modeled using OOA, prescribes how the analysis models of the various domains will be implemented through a set of rules that map from an analysis to an implementation. Recursive Design also advocates an order in which the domains are implemented.

The effect of the Recursive Design process means that when constructing the software, integration testing has been broken down into three logical parts: testing of the application and other domains, testing of the architecture, and testing of the mapping rules. This makes the integration testing process much more manageable, and also means that integration testing can conform to the development process being used.

7. *System and acceptance testing of the software are planned and performed to demonstrate that the software satisfies its requirements.*

The method allows an early verification of functional requirements through the simulation of analysis models. This means that system and acceptance testing of the software can be planned with these earlier tests in mind, and the results can be compared against the original simulations and the requirements.

8. *The documentation that will be used to operate and maintain the software is developed and maintained according to the project's defined software process.*

The exact amount of documentation needed when using the Shlaer-Mellor Method, and the work products they are associated with, is well-defined and unambiguous. Each piece of documentation is there to explicitly capture some piece of analysis or design information and provide a view of some aspect of the system.

There are different items of documentation at the different levels of a system. At the domain level, models are at a high enough level to provide a good understanding of the system partitioning. At the subsystem level, the models describe the various subsystem relationships. Documentation at the object level is detailed enough that all the necessary information needed to fully understand a system is available. For more information see [Shlaer91].

9. *Data on defects identified in peer reviews and testing are collected and analyzed according to the project's defined software process.*

The unique properties of rigor and formality of the method mean that the definitions of the defects are unambiguous and can clearly be measured against the rules of the method [Lang93]. These defects can be collected, categorized, and analyzed against method rules and also against the functional requirements. See [PT93].

10. *Consistency is maintained across software work products, including the software plans, process descriptions, allocated requirements, software requirements, software design, code, test plans, and test procedures.*

A cornerstone of the Shlaer-Mellor Method is a requirement for consistency across all software work products including software design, requirements, code, test plans, etc. As a result each work product represents a distinct view of some aspect of the system with the inter-relationships between work products being clearly defined [Shlaer93a] so that consistency can be maintained [Lang93].

3.6. Intergroup Coordination

The goals of the intergroup coordination key area are to ensure that the customer's requirements and the commitments between engineering groups are agreed to by all the affected groups.

Intergroup Coordination

Activity	Method Related	Procedure & Policy
1. Software engineering group participates with the customer and end users to establish system requirements		◆
2. Software engineering group works with other groups to monitor and coordinate technical activities and resolve issues		◆
3. A documented plan used to communicate intergroup commitments and to coordinate and track work performed		◆
4. Critical dependencies between groups are tracked	◆	
5. Work products produced as input to other groups are reviewed by representatives of receiving groups		◆
6. Intergroup issues not resolvable by individual representatives are handled according to procedure		◆
7. Representatives of groups conduct periodic reviews and interchanges		◆

Activities Supported

4. *Critical dependencies between engineering groups are identified, negotiated, and tracked according to a documented procedure.*

The method, with its well defined work products and system partitions, can assist greatly in the identification of critical dependencies. The relationships between the dependencies and the engineering groups can be used for tracking purposes. See also Section 3.4 Integrated Software Management, activity 9.

3.7. Peer Reviews

The goals are to plan peer review activities identifying and removing defects in the software work products.

Peer Reviews

Activity	Method Related	Procedure & Policy
1. Peer reviews are planned and the plans are documented		◆
2. Peer reviews are performed according to a documented procedure	◆	
3. Data on conduct and results of reviews are recorded		◆

Activities Supported

2. *Peer reviews are performed according to a documented procedure.*

   Shlaer-Mellor analysis models require that analysts adhere to the formalism of OOA. Fortunately, such checks can be checked for by CASE tools, thus leaving analysts to concentrate on checking the semantics of the problem domain being modeled. For more information see [PT93].

4. Level 4: Managed

4.1. Quantitative Process Management

The goals of the quantitative process management key area are to plan, control, and quantitatively measure the project's defined software process.

Activities Supported

3. *The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's defined software process.*

   The Shlaer-Mellor Method has a well-defined set of work products and a well-defined software process. This means that the process control points and data collection points are clearly identifiable. Some examples of data collection points include the completion of the construction of the object information model and completion of state models and the object communication model. More details can be found in [Montrose93], which describes what measurements were taken and when, using the Shlaer-Mellor Method based on real-world experiences.

4. *The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.*

   Because the Shlaer-Mellor Method has a well-defined set of work products, and because each work product's contents are created using a set of rigorous rules [Lang93], it is possible to arrive at a set of data that needs to be measured to control the software process quantitatively. A definitive list is beyond the scope of this paper, but the reader is recommended to consult [Montrose93] for more details.

Quantitative Process Management

Activity	Method Related	Procedure & Policy
1. Quantitative process management plan developed according to a documented procedure		◆
2. Activities performed in accordance with plan		◆
3. Strategy for data collection and analyses based on process	◆	
4. Measurement data are collected according to a documented procedure	◆	
5. Defined software process is analyzed and brought under quantitative control according to a documented procedure	◆	
6. Reports documenting results are prepared and distributed		◆
7. Process capability baseline for organization's process is established and maintained according to procedure		◆

5. *The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.*

   The Shlaer-Mellor Method has been used successfully on numerous real-world projects [Lee93], and as a result the data analysis activities required are well-defined. They describe the input data required, the data manipulations performed, and also what analysis techniques were used. More details can be found in [Montrose93].

4.2. Software Quality Management

The goals of the software quality management key area are to plan, define, quantify, and manage software quality activities.

Activities Supported

2. *The project's software quality plan is the basis for the project's activities for software quality management.*

   With its well-defined process the Shlaer-Mellor Method has clearly defined points in the process where software quality is measured and identifies activities to measure software product quality. See

Software Quality Management

Activity	Method Related	Procedure & Policy
1. Software quality plan is developed and maintained according to a documented procedure.		◆
2. Quality plan is the basis for quality management activities	◆	
3. Quantitative quality goals for products are defined, monitored and revised throughout the software lifecycle	◆	
4. Quality of products is measured, analyzed, and compared to quantitative quality goals on an event driven basis		◆
5. Quantitative quality goals are allocated appropriately to subcontractors delivering software products to the project		◆

[Montrose93].

3. *The project's quantitative quality goals for the software products are defined, monitored, and revised throughout the software life cycle.*

For each stage in the Shlaer-Mellor development lifecycle there are distinct work products to be produced, and combined with the quantitative criteria with which these are to be measured, it is possible to arrive at a set of well-defined quality goals. More details can be found in [Montrose93].

5. Summary

A mature software development process is essential to the production of high-quality and dependable software. This is the basis for the Software Engineering Institute's (SEI's) Capability Maturity Model, whose five-leveled maturity scale allows organizations to evaluate their software process capability and to target areas for improvement. Critical to achieving software process maturity is the support for a clearly defined and repeatable process that is inherent in the Shlaer-Mellor Method for system development.

The Shlaer-Mellor Method, with its rigor and formality, supports all method-related activities for the key process areas up to SEI Level 4. The method is particularly supportive in the process areas of Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Integrated Software Management, and Software Process Engineering. The method also provides extensive support in the remaining process areas, even though these are mainly centered around organizational and management issues.

6. References

[Dion93]
Raymond Dion, “Process Improvement and the Corporate Balance Sheet,” *IEEE Software*, Volume 10 (4), pp28-35, July 1993.

[Humphrey89]
Watts S. Humphrey, *Managing The Software Process*, Addison-Wesley, 1989.

[Lang93]
Neil Lang, “Shlaer-Mellor Object-Oriented Analysis Rules,” in *ACM SIGSOFT Software Engineering Notes*, Volume 18(1), January 1993.

[Lee93]
Michael M. Lee, “Object-Oriented Analysis in Large Scale Projects,” *Object Magazine*, Volume 3 (4), pp45-49, November-December 1993.

[Montrose93]
Rodney C. Montrose, “Software Metrics Using the Shlaer-Mellor Method,” *in preparation*, November 1993.

[Paulk93a]
Mark C. Paulk, Bill Curtis, Mary Beth Chrissis and Charles V. Weber, *Capability Maturity Model for Software*, Version 1.1, Technical Report CMU/SEI-93-TR-24, February 1993.

[Paulk93b]
Mark C. Paulk, Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis and Marilyn Bush, *Key Practices of the Capability Maturity Model*, Version 1.1, Technical Report CMU/SEI-93-TR-25, February 1993.

[PT93]
Project Technology Inc., Course Notebooks: Object-Oriented Analysis--Domains and Objects, Object-Oriented Analysis--States and Processes, Recursive Design, 1993.

[Shlaer88]
Sally Shlaer and Stephen J. Mellor, *Object Oriented Systems Analysis: Modeling the World in Data*, Prentice-Hall, 1988.

[Shlaer91]
Sally Shlaer and Stephen J. Mellor, *Object Lifecycles: Modeling The World In States*, Prentice-Hall, 1991.

[Shlaer93a]
Sally Shlaer and Stephen J. Mellor, *Real-Time Recursive Design*, Project Technology, Berkeley, CA, 1993.

[Shlaer93b]

Sally Shlaer and Stephen J. Mellor, “A Deeper Look at the Analysis-Design Transition,” *Journal of Object-Oriented Programming*, pp16-21, February 1993.

[Shlaer84]
Sally Shlaer, Diana Grand, and Stephen J. Mellor, “The Project Matrix: A Model for Software Engineering Project Management,” in *Proceedings of the Third Software Engineering Standards Application Workshop*, October 1984.

[Shlaer93c]
Sally Shlaer and Stephen J. Mellor, “The Shlaer-Mellor Method,” Project Technology, Berkeley, CA, 1993.