International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018

A Novel Model for the Software Engineering Using Process Mapping Techniques

Rambabu Mudusu & Dr. M.V.Siva Prasad
Assoc.Professor, Department of Computer Science & Engineeng, KG Reddy College of Engineering & Technology, Hyderabad,
E-Mail: rams.crypto@kgr.ac.in
B.E. M.Tech., Ph.D., MISTE, Principal Department of Computer Science and Engineering Anurag Engineering College, Kodada,Suryapet,Telangana
E-Mail: Prasad@magantis.com

ABSTRACT:
The software engineering process Knowledge Area has witnessed dramatic growth over the last decade. This was partly due to recognition by major acquirers of systems where software is a major component that process issues can have an important impact on the ability of their suppliers to deliver. Therefore, they encouraged a focus on the software engineering process as a way to remedy this. Furthermore, the academic community has recently pursued an active research agenda in developing new tools and techniques to support software engineering processes, and also empirically studying these processes and their improvement. These process models are exercising mappings of data symbols which are used in every software engineering process. It should also be recognized that many software engineering process issues are closely related to other disciplines, namely those in the management sciences, albeit they have used a different terminology. The industrial adoption of software engineering process technology has also been increasing, as demonstrated by a number of published success stories. Therefore, there is in fact an extensive body of knowledge on the software engineering process.

Key Points: *Software Process, Knowledge Area, Measurements, Software Engineering Cycle, Process Mapping,*

1. INTRODUCTION

The software engineering process Knowledge Area (KA) can potentially be examined at two levels. The first level encompasses the technical and managerial activities within the software engineering process that are performed during software acquisition, development, maintenance, and retirement. The second is the meta-level, which is concerned with the definition, implementation, measurement, management, change and improvement of the software engineering process itself. The latter we will term *software process engineering*. The first level is covered by the other KA's of this Guide to the Software Engineering Body of Knowledge.

This Knowledge Area is concerned with the second: **software process engineering**.

1.1 Scope

This Knowledge Area does not explicitly address the following topics:

- Human resources management (for example, as embodied in the People CMM [30][31])
- Systems engineering processes

While important topics in themselves, they are outside the direct scope of software process engineering. However,
where relevant, interfaces (or references to interfaces) to HRM and systems engineering will be addressed.

1.2 Currency of Material

The software process engineering discipline is rapidly changing, with new paradigms and new models. The breakdown and references included here are pertinent at the time of writing. An attempt has been made to focus on concepts to shield the knowledge area description from changes in the field, but of course this cannot be 100% successful, and therefore the material here must be evolved over time. A good example is the on-going CMM Integration effort (see http://www.sei.cmu.edu/cmmi/products/models. html for the latest document suite) and the Team Software Process effort [71], both of which are likely to have a considerable influence on the software process community once widely disseminated, and would therefore have to be accommodated in the knowledge area description. In addition, where Internet addresses are provided for reference material, these addresses were verified at the time of press. However, there are no guarantees that the documents will still be available on-line at the same location in the future.

1.3 Structure of the KA

To structure this KA in a way that is directly related to practice, we have defined a generic process model for software process engineering. This model identifies the activities that are performed in a process engineering context. The topics are mapped to these activities. The advantage of such a structure is that one can see, in practice, where each of the topics is relevant, and provides an overall rationale for the topics. This generic model is based on the PDCA (plan-do-check-act) cycle.

2. LITERATURE SURVEY

In software engineering, a **software development process** is the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a **software development life cycle**. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.[1]

Most modern development processes can be vaguely described as agile. Other methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming.

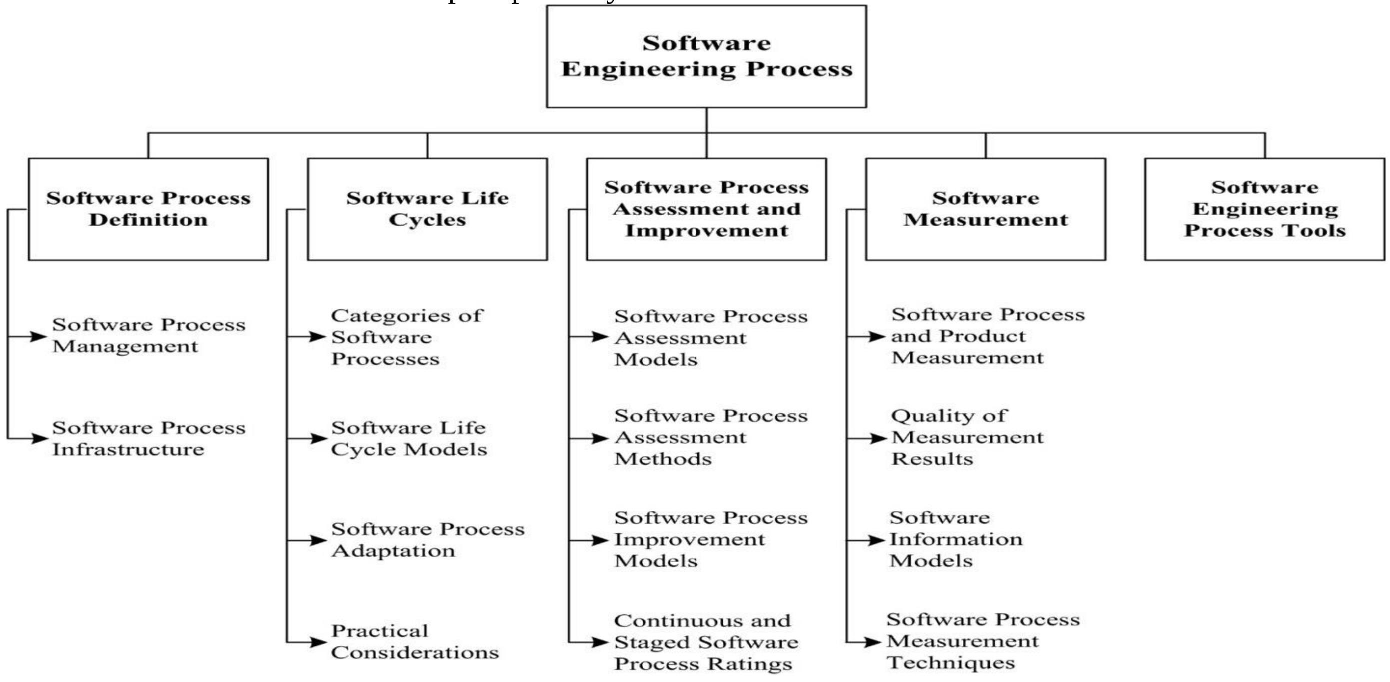
Some people consider a life-cycle "model" a more general term for a category of methodologies and a software development "process" a more specific term to refer to a specific process chosen by a specific organization. For example, there are many specific software development processes that fit the spiral life-cycle model. The field is often considered a subset of the systems development life cycle.

Software development is the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components. Software development is a process of writing and maintaining the source code, but in a broader sense, it includes all that is involved between the conception of the desired software through to the final manifestation of the software, sometimes in a planned and structured process. [1] Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products. [2] Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a perceived need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a scientist may write software to automate a mundane task). Embedded software development, that is, the development of embedded software, such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product. System software underlies applications and the programming process itself, and is often developed separately. The need for better quality control of the software development process has given rise to the discipline of software engineering, which aims to apply the systematic approach exemplified in the engineering paradigm to the process of software development.

A software development process (also known as a software development methodology, model, or life cycle) is a framework that is used to structure, plan, and control the process of developing information systems. A wide variety of such frameworks has evolved over the years, each with its own recognized strengths and weaknesses.

There are several different approaches to software development: some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations. Most methodologies share some combination of the following stages of software development:

- Analyzing the problem
- Market research
- Gathering requirements for the proposed business solution
- Devising a plan or design for the software-based solution
- Implementation (coding) of the software
- Testing the software
- Deployment
- Maintenance and bug fixing


Software Engineering Process
Software Process Definition
Software Process Management
Software Process Infrastructure
Software Life Cycles
Categories of Software Processes
Software Life Cycle Models
Software Process Adaptation
Practical Considerations
Software Process Assessment and Improvement
Software Process Assessment Models
Software Process Assessment Methods
Software Process Improvement Models
Continuous and Staged Software Process Ratings
Software Measurement
Software Process and Product Measurement
Quality of Measurement Results
Software Information Models
Software Process Measurement Techniques
Software Engineering Process Tools

These stages are often referred to collectively as the software development lifecycle, or SDLC. Different approaches to software development may carry out these stages in different orders, or devote more or less time to different stages. The level of detail of the documentation produced at each stage of software development may also vary. These stages may also be carried out in turn (a “waterfall” based approach), or they may

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 12
April 2018

be repeated over various cycles or iterations (a more "extreme" approach). The more extreme approach usually involves less time spent on planning and documentation, and more time spent on coding and development of automated tests. More “extreme” approaches also promote continuous testing throughout the development lifecycle, as well as having a working (or bug-free) product at all times. More structured or “waterfall” based approaches attempt to assess the majority of risks and develop a detailed plan for the software before implementation (coding) begins, and avoid significant design changes and re-coding in later stages of the software development lifecycle planning.

3. EVALUATING SOFTWARE ENGINEERING PROCESS CONCEPTS

3.1.1 Themes

Dowson [35] notes that “All process work is ultimately directed at ‘software process assessment and improvement’”. This means that the objective is to implement new or better processes in actual practices, be they individual, project or organizational practices. We describe the main topics in the software process engineering (i.e., the meta-level that has been alluded to earlier) area in terms of a cycle of process change, based on the commonly known PDCA cycle. This cycle highlights that individual process engineering topics are part of a larger process to improve practice, and that process evaluation and feedback is an important element of process engineering.

At the centre of the cycle is the “Process Experience Base”. This is intended to capture lessons from past iterations f

the cycle (e.g., previous evaluations, process definitions, and plans). Evaluation lessons can be qualitative or quantitative. No assumptions are made about the nature or technology of this “Process Experience Base”, only that it

be a persistent storage. It is expected that during subsequent iterations of the cycle, previous experiences will be adapted and reused. It is also important to continuously re-assess the utility of information in the experience base to ensure that obsolete information does not accumulate.

With this cycle as a framework, it is possible to map the topics in this knowledge area to the specific activities where they would be most relevant. This mapping is also shown in **Figure 1**. The bulleted boxes contain the Knowledge Area topics. It should be noted that this cycle is not intended to imply that software process engineering is relevant to only large organizations. To the contrary, process-related activities can, and have been, performed successfully by small organizations, teams, and individuals.

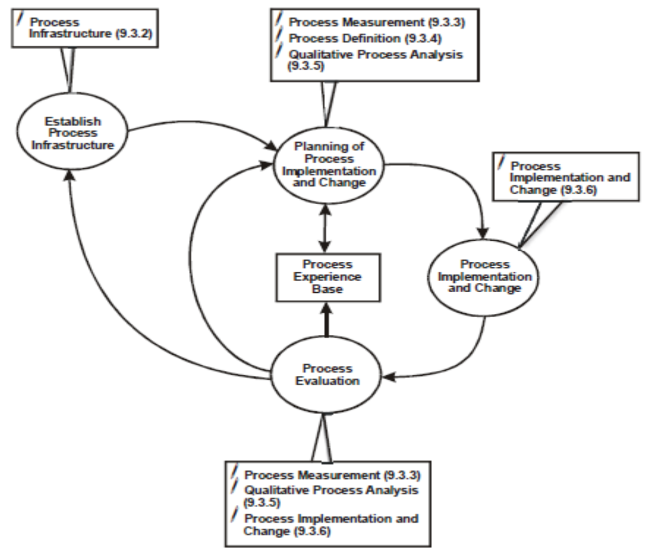

Process Infrastructure (9.3.2)
Process Measurement (9.3.3)
Process Definition (9.3.4)
Qualitative Process Analysis (9.3.5)
Establish Process Infrastructure
Planning of Process Implementation and Change
Process Implementation and Change (9.3.6)
Process Experience Base
Process Implementation and Change
Process Evaluation
Process Measurement (9.3.3)
Qualitative Process Analysis (9.3.5)
Process Implementation and Change (9.3.6)

Figure 1 A model of the software process engineering cycle.

Thedifferent levels in this KA are as follows:

Process Infrastructure: This is concerned with putting in place an infrastructure for software process engineering.

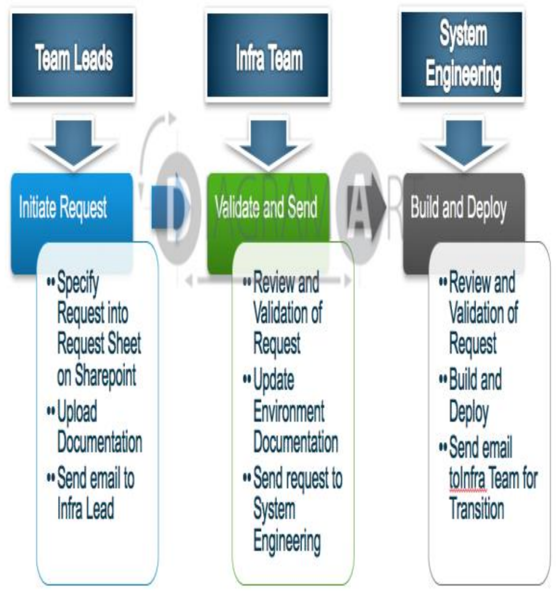

Team Leads
Infra Team
System Engineering
Initiate Request
Validate and Send
Build and Deploy
Specify Request into Request Sheet on Sharepoint
Upload Documentation
Send email to Infra Lead
Review and Validation of Request
Update Environment Documentation
Send request to System Engineering
Review and Validation of Request
Build and Deploy
Send email toInfra Team for Transition

Figure 2 A model of the software process Infrastructure.

It is widely recognized that a team separate from the developers/maintainers must be set up and tasked with process analysis, implementation and change [16]. The main reason for this is that the priority of the developers/maintainers is to produce systems or releases, and therefore process engineering activities will not receive as much attention as they deserve or need. This, however, should not mean that the project organization is not involved in the process engineering effort at all. To the contrary, their involvement is essential. Especially in a small organization, outside help (e.g., consultants) may be required to assist in making up a process team.

Process Measurement: This is concerned with quantitative techniques to diagnose software processes; to identify strengths and weaknesses. This can be performed to initiate process implementation and change, and afterwards to evaluate the consequences of process implementation and change.

Process performance measures are the 'vital signs' of organizational health, providing a current status assessment, giving clues to possible health issues, and showing progress towards recovery.

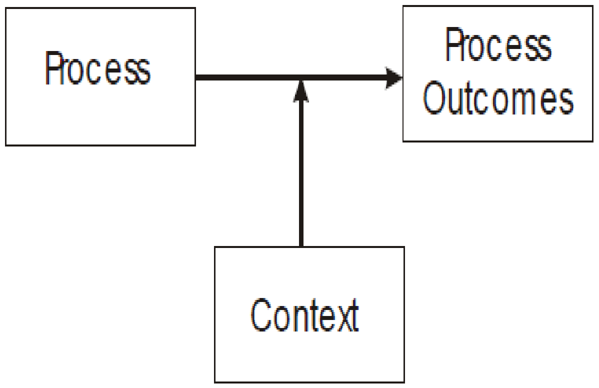


Figure 3 A model of the software process Measurement.

Two days later, all were relieved that the launch had gone well; two weeks later, there was a slowly rising murmur of complaint; two months later, service delivery was in virtual gridlock and the murmur was a cacophony. A month after that, it was finally discovered that a key process in the service delivery was often failing, causing significant rework and introducing inordinately long delays. Further analysis found ways to reduce the delays by 95%.

For this to be possible, there must be regular assessment of the effectiveness of the changes made. Process practitioners are not in the business of just making recommendations; their purpose is to make positive change—and to prove that they have done so. Measuring business process performance delivers many benefits:

1. factual evidence of customer-service levels
2. better understanding of cross-functional performance
3. enhanced alignment of operations with strategy
4. evidence-based determination of process improvement priorities
5. detection of performance trends
6. better understanding of the capability range of a process
7. uncovering actual and latent problems
8. changing behavior based on factual feedback

9. improved control over the risks that really matter.

Process Definition: This is concerned with defining processes in the form of models, plus the automated support that is available for the modeling task, and for enacting the models during the software process. Software engineering processes are defined for a number of reasons, including: facilitating human understanding and communication, supporting process improvement, supporting process management, providing automated process guidance, and providing automated execution support [29][52][68]. The types of process definitions required will depend, at least partially, on the reason. It should be noted also that the context of the project and organization will determine the type of process definition that is most important. Important variables to consider include the nature of the work (e.g., maintenance or development), the application domain, the structure of the delivery process (e.g., waterfall, incremental, evolutionary), and the maturity of the organization. There are different approaches that can be used to define and document the process. Under this topic the approaches that have been presented in the literature are covered, although at this time there is no data on the extent to which these are used in practice.

Qualitative Process Analysis: This is concerned with qualitative techniques to analyze software processes, to identify strengths and weaknesses.

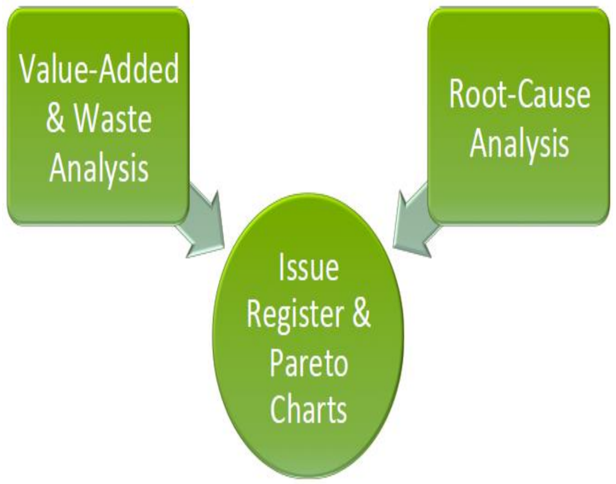


Figure 4 A model of the software Engineering Qualitative process

Process implementation and change is an instance of organizational change. Most successful organizational change efforts treat the change as a project in its own right, with appropriate plans, monitoring, and review. Guidelines about process implementation and change within software engineering organizations, including action planning, training, management sponsorship and commitment, and the selection of pilot projects, and that cover both the transition of processes and tools, are given in [33][92][95][104][114][120][127][130][133]. An empirical study evaluating success factors for process change is reported in [46]. Grady describes the process improvement experiences at Hewlett-Packard, with some general guidance on implementing organizational change [61].

4. STUDY RESULTS

A process map is a planning and management tool that visually describes the flow of work. Using process mapping software, process maps show a series of events that produce an end result. A process map is also called a flowchart, process flowchart, process chart, functional process chart, functional flowchart, process model, workflow diagram, business flow diagram or process flow diagram. It shows who and what is involved in a process and can be used in any business or organization and can reveal areas where a process should be improved.

4.1 Purpose of process mapping

The purpose of process mapping is for organizations and businesses to improve efficiency. Process maps provide insight into a process, help teams brainstorm ideas for process improvement, increase communication and provide process documentation. Process mapping will identify bottlenecks, repetition and delays. They help to define process boundaries, process ownership, process responsibilities and effectiveness measures or process metrics.

4.2 Understanding processes

One of the purposes of process mapping is to gain better understanding of a process. The flowchart below is a good example of using process mapping to understand and improve a process. In this chart, the process is making pasta. Even though this is a very simplified process map example, many parts of business use similar diagrams to understand processes and improve process efficiency, such as operations, finance, supply chain, sales, marketing and accounting.

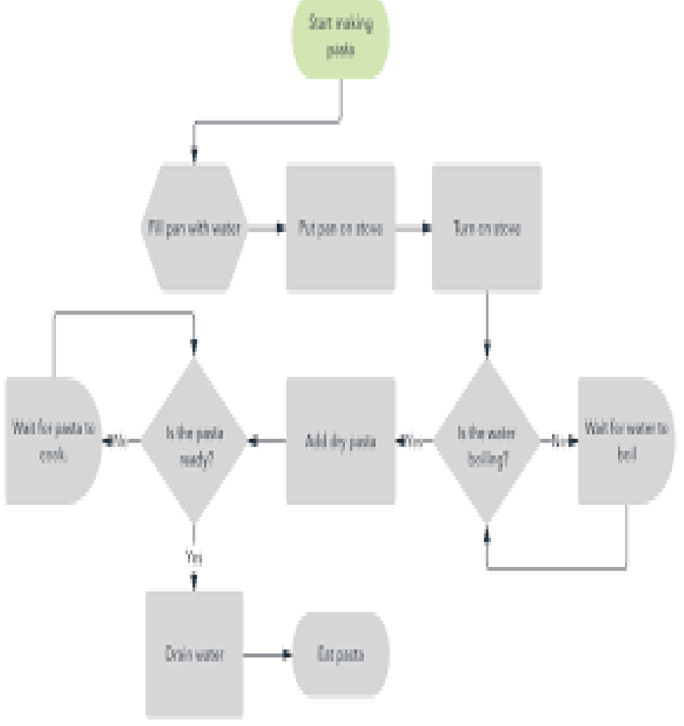


Figure 5 A Design for Mapping the Understanding processes

4.3 Benefits of process mapping

Process mapping spotlights waste, streamlines work processes and builds understanding. Process mapping allows you to visually communicate the important details of a process rather than writing extensive directions.

Flowcharts and process maps are used to:

- Increase understanding of a process
- Analyze how a process could be improved
- Show others how a process is done
- Improve communication between individuals engaged in the same process
- Provide process documentation
- Plan projects

Process maps can save time and simplify projects because they:

- Create and speed up the project design
- Provide effective visual communication of ideas, information and data
- Help with problem solving and decision making
- Identify problems and possible solutions
- Can be built quickly and economically
- Show processes broken down into steps and use symbols that are easy to follow
- Show detailed connections and sequences
- Show an entire process from the beginning to the end

Process maps help you to understand the important characteristics of a process, allowing you to produce helpful data to use in problem solving. Process maps let you strategically ask important questions that help you improve any process.

4.4 Types of process mapping

Process mapping is about communicating your process to others. You can build stronger understanding with process maps. The most common process map types include:

- **Activity Process Map**: represents value added and non-value added activities in a process
- **Detailed Process Map**: provides a much more detailed look at each step in the process
- **Document Map**: documents are the inputs and outputs in a process
- **High-Level Process Map**: high-level representation of a process involving interactions between Supplier, Input, Process, Output, Customer (SIPOC)
- **Rendered Process Map**: represents current state and/or future state processes to show areas for process improvement
- **Swimlane (or Cross-functional) Map**: separates out the sub-process responsibilities in the process
- **Value-Added Chain Diagram**: unconnected boxes that represent a very simplified version of a process for quick understanding
- **Value Stream Map**: a lean-management technique that analyzes and improves processes needed to make a product or provide a service to a customer.
- **Work Flow Diagram**: a work process shown in "flow" format; doesn't utilize Unified Modeling Language (UML) symbols.

4.5 Process mapping symbols

Key elements of process mapping include actions, activity steps, decision points, functions, inputs/outputs, people involved, process measurements and time required. Basic symbols are used in a process map to describe key process elements. Each process element is represented by a specific symbol such as an arrow, circle, diamond, box, oval or rectangle. These symbols come from the **Unified Modeling Language** or **UML**, which is an international standard for drawing process maps.

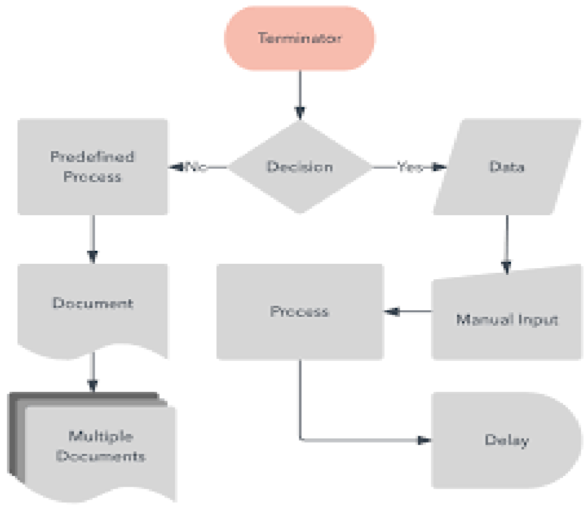


Figure 6 A Designs for Process mapping symbols

4.6 Business process mapping

In business, a process is a group of interrelated tasks that happen as a result of an event. These tasks produce a desired result for the customer. Process mapping can be used in many areas of business: business process improvement, business process redesign, reengineering, training, quality improvement, simulation, information technology, work measurement, documentation, process analysis, operational process design, process integration, acquisitions, mergers and selling business operations. Business process mapping can also be helpful for complying with manufacturing and service industry regulations, such as the common **ISO 9000 (International Organization for Standardization)** or **ISO 9001**.

How to create a process map

Process mapping has become streamlined because of software that provides a better understanding of processes. Process maps can be created in common programs like Microsoft Word, PowerPoint or Excel, but there are other programs more customized to creating a process map. Process mapping is about communicating your process to others so that you achieve your management objectives. Knowing how to map a process will help you build stronger communication and understanding in your organization.

Step 1: Identify the problem

- What is the process that needs to be visualized?
- Type its title at the top of the document.

Step 2: Brainstorm activities involved

- At this point, sequencing the steps isn't important, but it may help you to remember the steps needed for your process.
- Decide what level of detail to include.
- Determine who does what and when it is done.

Step 3: Figure out boundaries

- Where or when does the process start?
- Where or when does the process stop?

Step 4: Determine and sequence the steps

- It's helpful to have a verb begin the description.
- You can show either the general flow or every detailed action or decision.

Step 5: Draw basic flowchart symbols

Each element in a process map is represented by a specific flowchart symbol. Lucidchart makes it simple to create and rearrange shapes, add labels and comments and even use custom styling in your process map.

- Ovals show the beginning of a process or the stopping of a process.
- Rectangles show an operation or activity that needs to be done.
- Arrows represent the flow of direction.
- Diamonds show a point where a decision must be made. Arrows coming out of a diamond are usually labeled yes or no. Only one arrow comes out of an activity box. If more than is needed, you should probably use a decision diamond.
- A parallelogram shows inputs or outputs.

Step 6: Finalize the process flowchart

- Review the flowchart with others stakeholders (team member, workers, supervisors, suppliers, customers, etc.) for consensus.
- Make sure you've included important chart information like a title and date, which will make it easy to reference.
- Helpful questions to ask:
    - Is the process being run how it should?
    - Will team members follow the charted process?
    - Is everyone in agreement with the process map flow?
    - Is anything redundant?
    - Are any steps missing?

Process maps provide valuable insights into how a businesses or an organization can improve processes. When important information is presented visually, it increases understanding and collaboration for any project.

5. CONCLUSSION

This paper presents how to design process models what are the best models in that levels os the software engineering process, and also empirically studying these processes and their improvement. These process models are exercising mappings of data symbols which are used in every software engineering process. Future enhancement is to in engineering education evaluation procedure are mapped with data symbols. In this paper provides the most up-to-date review of evidence supporting process assessment and improvement, as well as a historical perspective on some of the levels using with analysis and measurements.

6. REFERENCES

[1] T. Abdel-Hamid and S. Madnick, *Software Project Dynamics: An Integrated Approach*, Prentice-Hall, 1991.

[2] W. Agresti, "The Role of Design and Analysis in Process Improvement," in *Elements of Software Process Assessment and Improvement*, K. El-Emam and N. Madhavji (eds.), IEEE CS Press, 1999.

[3] L. Alexander and A. Davis, "Criteria for Selecting Software Process Models," in *Proceedings of COMPSAC'91*, pp. 521-528, 1991.

[4] J. Armitage and M. Kellner, "A Conceptual Schema for Process Definitions and Models," in *Proceedings of the Third International Conference on the Software Process*, pp. 153-165, 1994.
[5] S. Bandinelli, A. Fuggetta, L. Lavazza, M. Loi, and G. Picco, "Modeling and Improving an Industrial Software Process," *IEEE Transactions on Software Engineering*, vol. 21, no. 5, pp. 440-454, 1995.
[6] R. Barbour, "Software Capability Evaluation - Version 3.0 : Implementation Guide for Supplier Selection," Software Engineering Institute, CMU/SEI-95-TR012, 1996.
(available at http://www.sei.cmu.edu/publications/documents/95. reports/95.tr.012.html)
[7] N. Barghouti, D. Rosenblum, D. Belanger, and C. Alliegro, "Two Case Studies in Modeling Real, Corporate Processes," *Software Process - Improvement and Practice*, vol. Pilot Issue, pp. 17-32, 1995.
[8] V. Basili, G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory," *ACM Transactions on Software Engineering and Methodology*, vol. 1, no. 1, pp. 53-80, 1992.
[9] V. Basili, G. Caldiera, F. McGarry, R. Pajerski, G. Page, and S. Waligora, "The Software Engineering Laboratory - An Operational Software Experience Factory," in *Proceedings of the International Conference on Software Engineering*, pp. 370-381, 1992.
[10] V. Basili, S. Condon, K. El-Emam, R. Hendrick, and W. Melo, "Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components," in *Proceedings of the 19th International Conference on Software Engineering*, pp. 282-291, 1997.
[11] B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, vol. 21, no. 5, pp. 61-72, 1988.
[12] T. Bollinger and C. McGowan, "A Critical Look at Software Capability Evaluations," *IEEE Software*, pp. 25-41, July, 1991.
[13] L. Briand, V. Basili, Y. Kim, and D. Squire, "A Change Analysis Process to Characterize Software Maintenance Projects," in *Proceedings of the International Conference on Software Maintenance*, 1994.
[14] L. Briand, W. Melo, C. Seaman, and V. Basili, "Characterizing and Assessing a Large-Scale Software Maintenance Organization," in *Proceedings of the 17th International Conference on Software Engineering*, pp. 133-143, 1995.
[15] L. Briand, C. Differding, and H.D. Rombach, "Practical Guidelines for Measurement-Based Process Improvement," *Software Process Improvement and Practice*, vol. 2, pp. 253-280, 1996.
[16] L. Briand, K. El Emam, and W. Melo, "An Inductive Method for Software Process Improvement: Concrete Steps and Guidelines," in *Elements of Software Process Assessment and Improvement*, K. El-Emam and N. Madhavji (eds.), IEEE CS Press, 1999.
[17] F. Budlong and J. Peterson, "Software Metrics Capability Evaluation Guide," The Software Technology Support Center, Ogden Air Logistics Center, Hill Air Force Base, 1995.
[18] I. Burnstein, T. Suwannasart, and C. Carlson, "Developing a Testing Maturity Model: Part II," *Crosstalk*, pp. 19-26, September, 1996. (available at http://ww w.stsc.hill.af.mil/crosstalk/)