ABSTRACT

MARKOW, TANYA THAIS. A Knowledge Maturity Model: An Integration of Problem Framing, Software Design, and Cognitive Engineering. (Under the direction of DR. THOMAS LYNN HONEYCUTT).

The Knowledge Maturity Model (KMM) is a new model proposed as an alternative to an existing software engineering evaluation model, the Capability Maturity Model (CMM). The KMM is offered as a solution to some key weaknesses of the CMM. The CMM was developed in the early 1980s, when highly structured programming and business practices were the standard. In the current agile methods computer science environment, it is often difficult to evaluate a company which employs agile methods using the CMM methodology. The CMM consists of five levels; in order to claim the next higher level, all tasks of that level must be accomplished. Many companies operating with agile software engineering and management practices tend to be performing at many levels within the CMM, making it difficult to assign such an organization an appropriate CMM level designation. The KMM proposes instead an evaluation of the actual inner processes the company uses to develop software, rather than it's ability to achieve a given set of tasks, as required by CMM.

It will be shown that the KMM bridges the gap between the CMM and agile methods by employing the Knowledge Insight Model (KIM). The KIM is an iterative process that employs four key roles: Framer, Maker, Finder and Sharer. The Framer is responsible for the "big picture" of project management, including defining requirements and scope. The Maker must create new concepts and code for solving the problem. The Finder seeks out existing knowledge and information to help solve the problem. The

Sharer must create and maintain a database of the project and ensure that all involved get the information they need. The Knowledge Maturity Model incorporates the concept of levels or states of maturity from the CMM, and the core fundamentals of the KIM: the roles and an iterative process. The synergy of these concepts gives rise to the four state model of the KMM: recognition and use of the Plan, Do, Check, Act cycle, use of the four roles of KIM, use of an iterative process, and finally, the fully working inner mechanism or sharing mechanism of the KIM. The KMM allows an organization to choose any traditional software engineering methodology for a given project by providing the roles-based structure to make shifting between software engineering methodologies easier, allowing companies to tailor their process for specific projects.

KMM ties together the three fundamental centers that comprise the process of developing software: systems engineering, software engineering, and cognitive engineering. The KMM solves the systems engineering problem by providing a generalized process that is a superset of any given software engineering methodology. Because KMM provides a superset to all existing software engineering methodologies, it frees up an organization to choose the one that best suits a given project, rather than always having to use one standard approach, therefore addressing the software engineering aspect. At the heart of KMM are the four roles, which addresses the need to completely incorporate people into the process, thus bringing in the cognitive engineering side of the discipline.

A Knowledge Maturity Model: An Integration of Problem Framing, Software Design, and Cognitive Engineering

By

Tanya Thais Markow

A thesis submitted to the Graduate Faculty in partial fulfillment of
the requirements for the Degree of

Master of Science in **COMPUTER SCIENCE**

Department of Computer Science

North Carolina State University

2004

Approved By _____

Dr. Thomas L. Honeycutt, Chairperson of Advisory Committee

_____

Dr. Christopher G. Healey

_____

Dr. Robert St. Amant

Date: _____

To my husband, Pete and my daughter, Morgan,

for their tremendous love, support and patience.

To my mother, Annette Fogarty,

for raising me believing that with perseverance,

ALL things are possible.

And above all, I thank God for

all the blessings in my life.

BIOGRAPHY

Tanya Thais Markow (nee Tolles) was born in Mayfield Heights, Ohio on August 19, 1973. She attended Lakewood High School (1987-1990), and the United States Military Academy at West Point (1991-1995), where she earned a Bachelor of Science in Mechanical Engineering (Aerospace) and was commissioned as a Second Lieutenant in the United States Army. She graduated from the Aviation Officer Basic Course as an Honor Graduate in June 1996, designated an Army Aviator, and became an AH-64A Apache Pilot in October 1996. She then served as an Attack Platoon Leader in A Troop, 1-6 Cavalry, in the Republic of Korea (1996-1997). She returned to the United States to 3-229th Aviation Regiment (Attack), Ft. Bragg, North Carolina, where she served as Battalion Intelligence Officer. She deployed with her battalion to Bosnia-Herzegovina in support of Stabilization Force IV, Operation Joint Endeavor (1998-1999). Tanya then attended the Aviation Captain's Career Course (2000). She served as Battalion Adjutant Officer for 1-210th Aviation Regiment before being selected for Company Command of HHC, 1-14th Aviation Regiment at Ft. Rucker, Alabama. She earned the designation of AH-64A Apache Instructor Pilot in August 2001. Tanya is an active duty Army Captain, with over 1000 hours in various aircraft, and is a Senior Army Aviator. Selected in 2001 to teach Computer Science at the United States Military Academy, Tanya is currently pursuing a Master of Science in Computer Science at North Carolina State University in Raleigh, North Carolina.

ACKNOWLEDGEMENTS

I wish to thank Dr. Thomas L. Honeycutt for recognizing the potential for this thesis in my eighteen-page answer to a homework problem he assigned in Software Engineering. Thanks for your guidance and wisdom. I would also like to thank my committee members, Dr. Christopher G. Healey and Dr. Robert St. Amant for their suggestions and assistance during the course of my thesis.

I thank my husband for his love, patience, and help, and for being my best friend. I thank my daughter for her unconditional love. I thank my mother for instilling the belief in me that anything is possible. Finally, I wish to thank my friends at NCSU, specifically Sam Krishna, Michael Dougherty, Neha Katira, David Wright, and Sarat Kocherlakota for their help and friendship.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

1.1 Research Motivation:

There is “no silver bullet.” The truth of this statement is highly evident in the constantly evolving world of software engineering. This fact has not hampered the attempts by many to claim the latest “silver bullet” that would solve all of the discipline’s problems in one magic methodology or process. Extreme programming, SCRUM, Rational Unified Process, or even the classic waterfall and spiral models have all been touted at one time or another as the preverbal “silver bullet” of software engineering. The plain and simple truth is that all of these methodologies are employed, with various degrees of success, throughout the industry on any given day. The heart of the problem is that the way in which software engineers view these concepts is fundamentally flawed. They are tools, not magical concepts that will radically transform the industry. Just as it would be foolish for a carpenter to attempt to build a house using only a screwdriver, it is just as ridiculous for a software engineering company to try to create all of its software with only one methodology or process. A far more elegant solution for software engineering would be to employ a layer above software engineering methodologies or tools that allows an organization to use any of the tools it chooses, and move easily between them with a minimum of disruption. There is also the question of evaluation. There must be a way to determine the effectiveness of a particular methodology or tool. It is difficult to analyze and compare the success of the myriad of methodologies from

company to company. The Capability Maturity Model is one of the best-known ways to evaluate the potential success of a software engineering firm. Using the Capability Maturity Model (CMM), however, is also problematic. The CMM provides only a checklist of key process areas, which must be accomplished to claim a particular level on the five-level model. It does not provide the methodology, or "how" of process improvement. A more generalized layer may not only aid in software development, but might also provide the answer to the "how" of process improvement.

1.2 Statement of the Problem:

There are a multitude of software engineering methodologies to choose from. They vary greatly from the early, highly structured Rational Unified Process, Waterfall and Spiral Models, to the more modern "agile" methods, which include models such as XP and SCRUM. The problem is deciding which is best to use for any given problem frame. Rather than being forced to use the same method all the time, a mature and highly agile organization could employ a layer above these methodologies that would allow them to decompose any method and use it, as they choose. Such a layer could also be used to accomplish process improvement and evaluation, using a method such as the CMM. The CMM provides a checklist for process improvement evaluation, but does not provide a method by which to accomplish the tasks required to move up the levels. The CMM does not accurately portray the abilities of many companies using agile methods. The CMM requires an organization to accomplish a list of set tasks in order to claim a particular level. However, many companies using agile methods find that they employ some

methods that would be rated at level five and some that would put them at level two. Strict adherence to the CMM would require such an organization to claim a level two status, but does not portray their higher skills in other areas. This can result in a company being underrated. True maturity of a software engineering company should be based on the inner workings, how the company actually develops software and how able that company is to adapt as needed to accommodate the needs of the customer, without such changes imposing great stress to the rest of the organization.

1.3 Goals for this thesis:

This paper will demonstrate a new paradigm for software engineering, the Knowledge Maturity Model (KMM). The foundation for the KMM, the Knowledge Insight Model (KIM), will be introduced and validated by demonstrating its use as a process improvement methodology by showing how it can be employed to climb the levels of the CMM. The KMM will be shown to represent another level of maturity in software engineering. The KMM will also be demonstrated to serve as a superset of current software engineering methodologies, allowing organizations to employ the aspects of KMM to customize their own method(s), as appropriate for a given problem frame. The KMM paradigm will provide the layer above software engineering methodologies and process improvement that can aid in orchestrating both activities in an organization. The KMM does not call for an upheaval of current software engineering practices, but rather offers a paradigm shift to a simpler way of executing software engineering and process improvement.

1.4 Knowledge Maturity Model (KMM):

The Knowledge Maturity Model (KMM) is a new paradigm that employs the concept of levels from the CMM, but also integrates the idea of roles-based software engineering. The KMM provides a layer above software engineering and process improvement methodologies, providing a method of control for these activities. At the heart of the KMM is the KIM, a generalized process that is capable of multiple instances. The KIM incorporates four roles: Framer, Maker, Finder, and Sharer. The KIM employs a means by which to transition between these states: the Plan, Do, Check, Act (PDCA) cycle. The KIM takes into account both internal and external factors that can influence an organization.

Each of the four roles allow team members to understand what their mission is, based on which role they are assigned at any given time. The Framer is responsible for defining the "big picture," determining project scope, requirements and goals. The Maker establishes new concepts that support the project goals. The Finder seeks out existing knowledge that can aid in solving the problem at hand. The Sharer establishes a database that all in the organization can access, to ensure that efforts are not duplicated.

The KMM consists of four maturity levels: Process Cycle, Roles, KIM, and Inner Mechanism. In the KMM framework, a company at the Process Cycle level employs a cycle, such as the Plan, Do, Check, Act (PDCA) cycle. An organization at the Roles level, assigns roles, with clear tasks, to project team members. An organization at the KIM level employs both roles and an iterative process model to accomplish software projects. An organization at the highest level, Inner Mechanism, has achieved a superior

level of communication both inside and outside the company. Such an organization maintains a constantly updated database of all available knowledge that pertains to a particular project. All team members have access to the database and it is maintained for use on future projects and is considered a "living" database. The reason for the Inner Mechanism as the highest level is that a lack of communication within companies is the leading cause of duplicated efforts and stagnation of projects, due to the fact that one department may not realize that another has already discovered the solution to a show-stopping problem. An organization with a highly developed Inner Mechanism avoids such pitfalls, and is at the pinnacle of maturity. The levels of the KMM may be used to determine the overall maturity of an organization, however, the KMM levels are always a part of an organization's framework. An organization that has achieved the Inner Mechanism level will not always need to apply this level to every project. A simple, straightforward problem may be quickly solved, stored in a database and never require the high level of communication of extremely large or complex problems. However, a company at this level always has the ability to handle such problems, should they arise.

1.5 Objectives of this thesis:

The objectives of this paper are:

(1) Review relevant literature related to the Capability Maturity Model and existing methodologies being used to climb the CMM "ladder."

(2) Introduce the Knowledge Insight Model and propose it as a solution to integrating people into the problem of "climbing" the CMM "ladder."

(3) Validate the KIM by applying it to the CMM, demonstrating how it can be employed by a software engineering organization to achieve the desired level on the CMM, given the organization's current level on the CMM and the target level.

(4) Introduce the Knowledge Maturity Model (KMM), a new paradigm that provides a control layer situated above both software engineering methodologies and process improvement concepts. The KMM also offers another way to evaluate software engineering process maturity.

1.6 Thesis Layout:

Chapter 2 covers the Capability Maturity Model, its uses and evolution since its introduction. Chapter 3 provides a review of literature and gives an overview of methods that may be used for process improvement. Chapter 4 introduces the Knowledge Insight Model. Chapter 5 validates the KIM by demonstrating its use in process improvement within the framework of the Capability Maturity Model. Chapter 6 introduces the Knowledge Maturity Model as a new paradigm that provides another way to evaluate process maturity, as well as a new approach to employing existing software engineering methodologies thereby allowing organizations to customize their own method(s). Chapter 7 consists of conclusions from this research and proposes possible future developments and research in the areas of process improvement and evaluation models.

2. CAPABILITY MATURITY MODEL (CMM)

2.1 History of the Capability Maturity Model (CMM):

The desire to establish a measure for software engineering products as well as the firms that create them, led to the development of the Capability Maturity Model by the Software Engineering Institute (SEI) at Carnegie Melon University [12]. The need for the Capability Maturity Model arose in November of 1986 from a request on the part of the United States government, looking for a way to measure the performance of various software engineering contractors [12]. The Department of Defense (DOD) had noticed a disturbing trend among its software engineering contractors. Delays in product delivery were often measured in terms of months and even years [12]. Often, projects were scrapped altogether due to undelivered software products [12]. In September 1987, SEI, in conjunction with the MITRE Corporation, debuted a "software process maturity framework [12]" which consisted of a questionnaire (which is often mistaken for the entire process itself), and a "software process assessment and software capability evaluation [12]." Since then, the CMM has undergone several changes and refinements to better bring it in line with the standards of the software engineering community and is intended to serve as a "living document [12]."

2.2 CMM Overview:

One common misconception about the CMM is that it is a process by which an organization may improve its software engineering methodologies. This is not the case. The CMM is simply a measure of the organization's current software engineering process and provides a set of goals, a checklist, of what the organization must accomplish in order to advance up the CMM "ladder" of levels. The CMM was derived from the basics of Total Quality Management (TQM) [12]. The CMM consists of five "levels" that are may be equated to a ladder. Each of the levels corresponds with an improved state of the company's overall software engineering process.

1. Initial – a few "smart guys" devise solutions to software engineering requests from customers. No organized, company-wide process in place. Such an organization may produce quality software, but the timeframe and cost are often unknown at the outset of the project. Even if the organization makes some predictions of time and cost, they are likely to miss these goals.

2. Repeatable – the organization establishes a simple process that can be used to develop software. A method of project management is established to ensure that the organization can track project cost, time, and effectiveness (ie – does the software meet the customer's specifications?).

3. Defined – a standard operating procedure (SOP) is established for software engineering activities across the organization. An SOP is also emplaced for managerial activities. The level 3 organization assigns a group to oversee software engineering processes. A training program is implemented across the organization to ensure that all

employees are using the same methodologies, to ensure standardization. Because the process is so well-defined, management is better able to track where any project is at any time, within the organization, making it easier for management to communicate time, cost, and functionality changes to the customer in a timely fashion.

4. Managed – the organization not only tracks project data such as cost, timeliness and customer satisfaction, but goes a step further by establishing a set of standards for these product measures. In this way, the organization is now able to determine how effective it is in meeting goals and deadlines. The level 4 organization also attempts to limit variations in its product, based on its standards. By further controlling the quality of the final product, a level 4 organization is known for its reliability.

5. Optimized – an organization at this level seeks to optimize the very processes it uses to create software products. A level 5 organization analyzes its process, finds flaws, corrects them, seeks out the reason such flaws were introduced in the first place, and ultimately improves the overall process. An organization at this level uses this process on its software products as well. A level 5 organization strives for a zero-defect product. Level 5 organizations correct errors, fix the process that allowed the errors to be generated, and evaluates and corrects the validation/verification process that allowed the errors to slip by, optimizing the overall software engineering process [12].

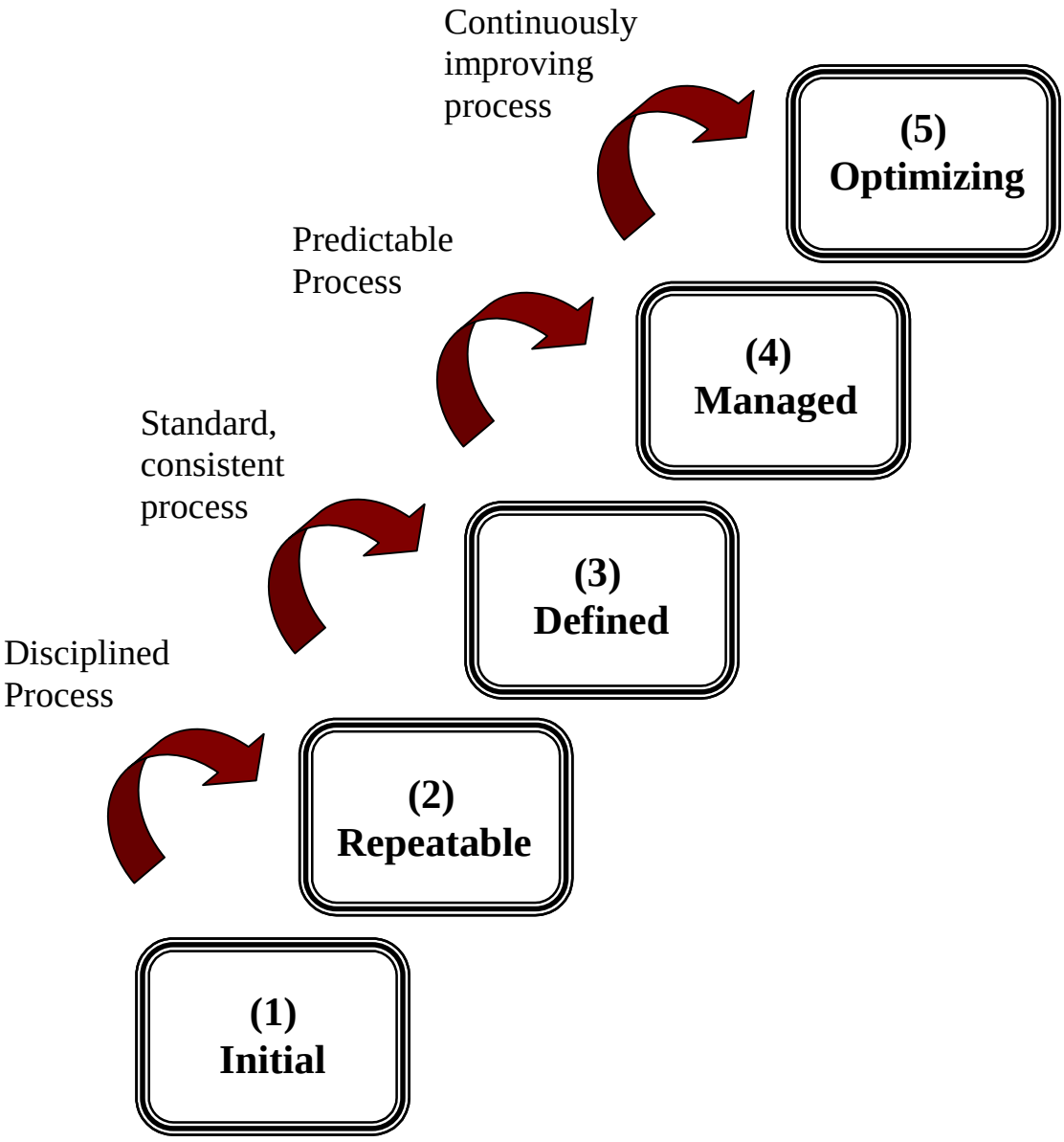

Figure 1. Diagram of the five levels of the Capability Maturity Model [12].

2.3 Evolution of the CMM, Rise of the CMMI:

The Capability Maturity Model for Software Engineering has evolved over the years, and the Software Engineering Institute at Carnegie Mellon University was poised to release version 2.0 of the SW-CMM, and the other CMM categories, which include P-CMM (People), IPD-CMM (Integrated Product Development), SA-CMM (Software Acquisition), and SE-CMM (Systems Engineering). However, there has been a growing desire in the industry, and within the Department of Defense, to see the entire CMM suite of products brought into one suite that would be generalized enough to accommodate all of the separate disciplines, allowing organizations to use one plan and reduce confusion between departments. Thus, SEI scrapped the release of CMM 2.0 and instead the Capability Maturity Model Integration (CMMI®) has come into being and will eventually replace all the categories of the CMM on the market [4]. There was clearly a desire on the part of SEI to attempt to provide potential users of the CMMI with examples of organizations that have succeeded with CMM (called “best practices” in the CMMI model) to help new using organizations of CMMI improve their processes while applying the CMMI checklist. The specific examples, while applicable to some, will not always be relevant to all potential users. The new CMMI suite of products makes a good attempt to provide a process improvement methodology, but continues to lack something fundamental to improving anything: how to integrate the people involved into the process improvement methodology.

3. PROCESS IMPROVEMENT METHODS

3.1 Review of Literature:

The literature review of this paper is divided into four sections. The first section deals with the Capability Maturity Model; its application in the computer science world as well as some key issues to consider when using the CMM. The second section addresses some other processes being used today to climb the CMM ladder of levels. The third section discusses advantages and disadvantages of using the Knowledge Insight Model to climb the CMM ladder. The fourth section provides an introduction to the Knowledge Insight Model, the foundation for the KMM.

3.2 Application of the Capability Maturity Model:

The Capability Maturity Model is by no means the only measure of software engineering organizations, but it is certainly one of the most recognized models in the industry. Many of the lucrative software engineering contracts today are sponsored by the Department of Defense, for whom the CMM was devised [4]. This certainly makes the CMM an attractive tool for many software engineering organizations.

3.3 Methods for Advancing within the CMM Framework:

IDEAL – IDEAL is "a life-cycle model for software process improvement based upon the Capability Maturity Model ® (CMM®) for software [3]." IDEAL is comprised of the following phases:

- Initiating – lay the groundwork for improvement
- Diagnosing – determine where the organization is and what the goal is
- Establishing – come up with a plan for process improvement
- Acting – Execute the plan
- Learning – Get feedback from the improvement process to help refine the new process or to use for future iterations

Advantages:

1. Extremely generalized method; may be used for many areas of organization.
2. Already well known and established in the computer science world.
3. Iterative process that may be used over until the end goal is reached.

Disadvantages:

1. Does not give guidance on the roles of personnel in the improvement process.
2. May be *too* generalized.

CMMI – The Capability Maturity Model Integration is an attempt by SEI to 1. Consolidate its CMM products into one suite that may be used for several different types of work (Software, People Management, Systems Engineering, Software Acquisition, and Integrated Product Development), and 2. To introduce "best practices," that were employed by successful CMM organizations, in hopes that these will help show

organizations how to climb the CMM ladder. There are two different methodologies within the CMMI, staged and continuous. The staged representation is a more static methodology that is very similar to the original CMM; it is best represented by the stair-step depiction of the CMM. The organization generally improves or develops the required traits to move up each step of the ladder and generally will do this in sequential fashion. The continuous method is better suited for organizations who wish to make improvements at different time frames from different categories, rather than taking the step-by-step approach of the staged method. The continuous method may be best for organizations who look at the CMMI and recognize that they have achieved level 2 in one arena, level 4 in another, level 3 in yet another, and they are best to continue to develop in all areas at different paces, than to slow progression in other areas, while the remaining catch up and the like.

Advantages:

1. One product may be potentially used across an organization for process improvement.
2. Based on an established, well-known process in the industry (CMM).
3. Employs “best practices” learned over the years from organizations who have used CMM.
4. Ability to use continuous or staged methods, as preferred by the organization.

Disadvantages:

1. More complicated than the original CMM; requires a “transition” for organizations already using CMM.

2. Does not explicitly address the "people" aspect of process improvement.

Consultants – Consultants are a wonderful way to climb the CMM/CMMI; if your business can afford it!

Advantages:

1. Very tailored, specific plans for your organization.
2. Consultants tend to focus only the particular contract they are currently working (ie – they are generally totally devoted to your problem or may only be working for a few organizations at a given time, allowing them to spend a lot of time focusing on your problem). When attempting to improve an organization internally, while also trying to continue to produce (ie – multiple distracters), there is less likelihood of completion of the process improvement within the desired timeframe.
3. Generally, outsiders, such as consultants, are less likely to be defensive of current practices. They are able to take a more objective view of where the organization currently is and what must be done to get them where they want to be.
4. An experienced consultant in the field of process improvement is likely to have seen an organization similar to the one currently desiring help, and therefore already has a basic "plan" that can be tailored to the particular organization in need. Such a consultant may be able to recognize certain patterns within a company and knows best how to achieve their goals, based

on past experiences. Most organizations have not had the ability to look into what other similar organizations have done in the line of process improvement, and therefore do not have a body of knowledge of "best practices" to use.

Disadvantages:

1. Can be VERY expensive; may be cost prohibitive for small companies that really need it.
2. Some consultants may try to "sell" their idea, even if it is not what's truly best for the organization.
3. Consultants may tell the organization exactly what they want to hear, in order to appease the management that hired them.

3.4 Advantages and Disadvantages of KIM:

Advantages of the Knowledge Insight Model are:

1. Assigning well-defined roles to all members of the project team (Framer, Finder, Maker, Sharer), thus reducing ambiguity.
2. It is an iterative process, allowing for multiple runs to achieve the organization's ultimate goals [7]
3. KIM does not lock the organization into any specific programming methodology (ie – Waterfall, Spiral, SCRUM, XP, RUP), as it is highly adaptable to different scenarios.
4. Communication, both within and outside the organization is a key factor in the

Knowledge Insight Model; this is one of the weaknesses of many current software engineering organizations.

5. Existing processes may be easily mapped to the KIM; thus organizations with existing processes may make use of the KIM and potentially optimize with it.

A disadvantage of the KIM is that it is a new process. Typically, those in both the corporate world and academia are resistant to change. However, the Knowledge Insight Model is easy to learn and is applicable in many areas of study, which may help overcome the usual resistance to new ideas and models [7]

3.5 Existing Process Analysis:

The review of literature shows that there are several possible ways to tackle the problem of climbing up the CMM ladder. However, there is a lack of definition in the "how" of process improvement. Roles within the organization are not well-defined as they pertain to software engineering process improvement. While consultants certainly take a lot of the problem off the back of the organization seeking to improve their process, it is just not cost effective for many smaller software engineering companies looking to improve their CMM standing. Once the consultants leave, there may be a lack of understanding of what was done and therefore require some dependency on the consultants for future improvements or changes. The CMMI is an attempt to give more of the "how" to climb the ladder, but is far too specific and cannot be viewed as a generalized process improvement vehicle. While IDEAL is an attractive way to execute

process improvement, there is no real personnel direction included. The usual problem of "what should I be doing right now?" is still a problem. The IDEAL model does meet the goal of a generalized process improvement method, but may actually be too generalized, to the point of almost seeming like common sense. By employing the Knowledge Insight Model, we seek to demonstrate a generalized process that meets the end goal of the organization: a higher level within the CMM framework. The introduction of roles to those participating in the process improvement has many positive benefits, to include the ability to adapt over to other types of projects in the organization (ie – the introduction of roles in software engineering projects).

4. KNOWLEDGE INSIGHT MODEL (KIM)

4.1 Knowledge Insight Model (KIM)

The Knowledge Insight Model (KIM) was created in an effort to construct "an instrument for acquiring knowledge [7]". The KIM also addresses the need to collect and share knowledge. The KIM is based off a Japanese idea, "Kaizen," meaning to continually analyze an organization's current position and work through a continuous improvement process [10]." People truly are any organization's greatest asset. In order to fully utilize this asset, an organization must ensure that the knowledge base of its employees is compiled and distributed throughout the organization to help prevent duplication of effort. This type of knowledge sharing also helps organizations through the departure and/or absence of key personnel. The foundation of the KIM lies in the four states or roles; Framer, Maker, Finder and Sharer. The mechanism that allows for transitions between these states is provided by the Plan, Do, Check, Act (PDCA) cycle. The KIM also takes into account both external and internal factors which affect the development of a company, process or product. The process begins in the discover phase, moves to refinement, and continually iterates between the two states to improve ideas and concepts. This method was derived from the Japanese term "Hoshin" meaning "continuous process improvement [10]."

4.2 Plan, Do, Check, Act (PDCA) Cycle

The Plan, Do, Check, Act (PDCA) Cycle plays a key role in the KIM. PDCA provides the transitions between the states of Framer, Maker, Finder and Sharer. The PDCA is four cycle methodology used to help identify and solve problems, and is especially well suited to process improvement.

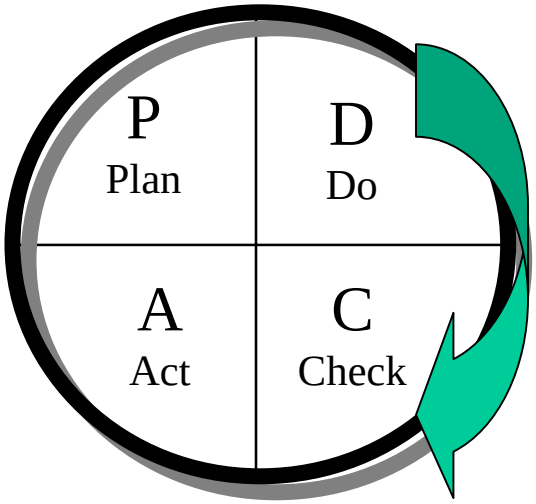

Figure 2. Plan, Do, Check, Act (PDCA) Cycle

- **Plan** to make changes to achieve process improvement; determine what the problem is and devise a way to try to solve the problem(s).
- **Do** changes in a series of small increments. A good way to do this is select one group within the company to serve as a test bed, allowing them to work under the changed process, while all other groups continue under the legacy process, so as not to disrupt business on a large scale.
- **Check** to see if the small changes are having the desired effect before moving on

to larger scale changes. Compare the test group with the norms throughout the organization, and with historical data.

- **Act** to make the changes on a large scale and continue to monitor across the organization to determine if the changes in process should be made permanent [13].

4.3 External, Internal, Discover, Refine Transition Matrix

The KIM considers the external and internal influences on a project, and how an organization transitions between the effects of both. A process will begin in the discovery mode, move into refinement, then continuously iterate between the two stages. Discovery and refinement may occur due to internal or external forces or ideas, and thus both must be considered by the organization. The four quadrant matrix in the figure below demonstrates the relationship between these factors.

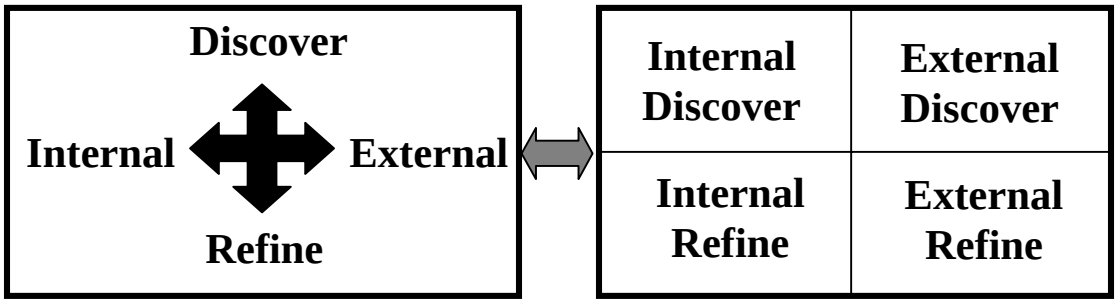

Figure 3. Four Quadrant Matrix

Solution increments occur when transitioning from discover to refinement, while consolidation is indicated by a transition from refinement back to discover. These transitions are continued iteratively until the refinement goal is achieved. Transitioning from external to internal and vice versa is accomplished in order to discover and incorporate both ideas from within an organization and without. Generally, an organization will look outside for inspiration, devise a new concept, and then turn inward for ideas on how to implement the new idea. The organization may then again look outside to help refine their innovation. Iteration between external and internal influences continues until the concept reaches a desired maturity level.

The final mature concept is then consolidated and used as input for further development (Refinement Evolution), or it may also be used to work toward a new creative goal (Creative Evolution).

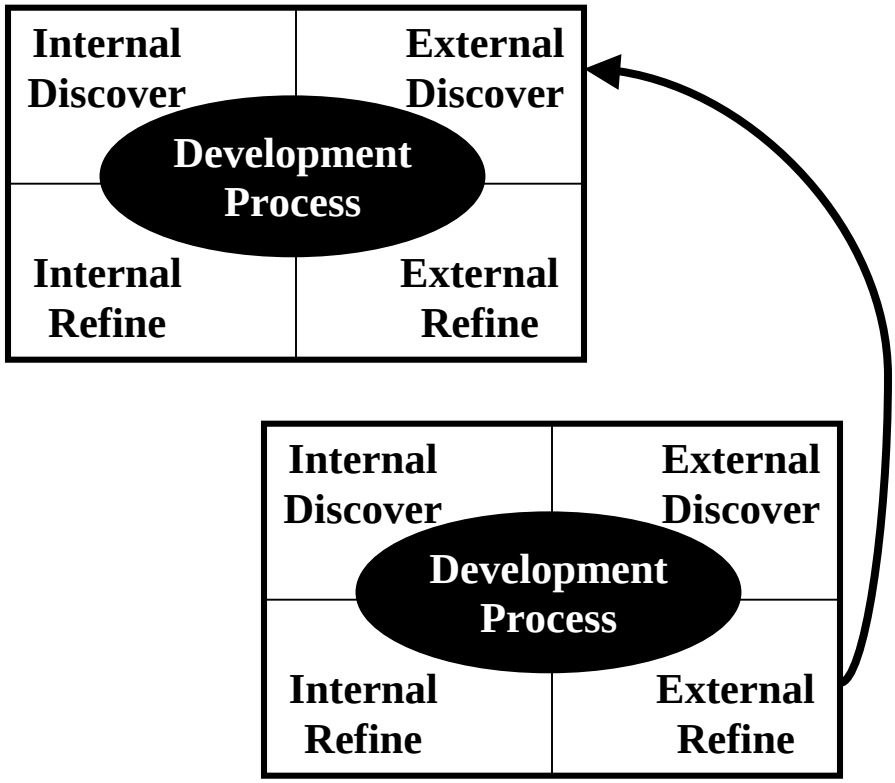

Figure 4. Refinement Evolution: Creative phase to development phase of design process

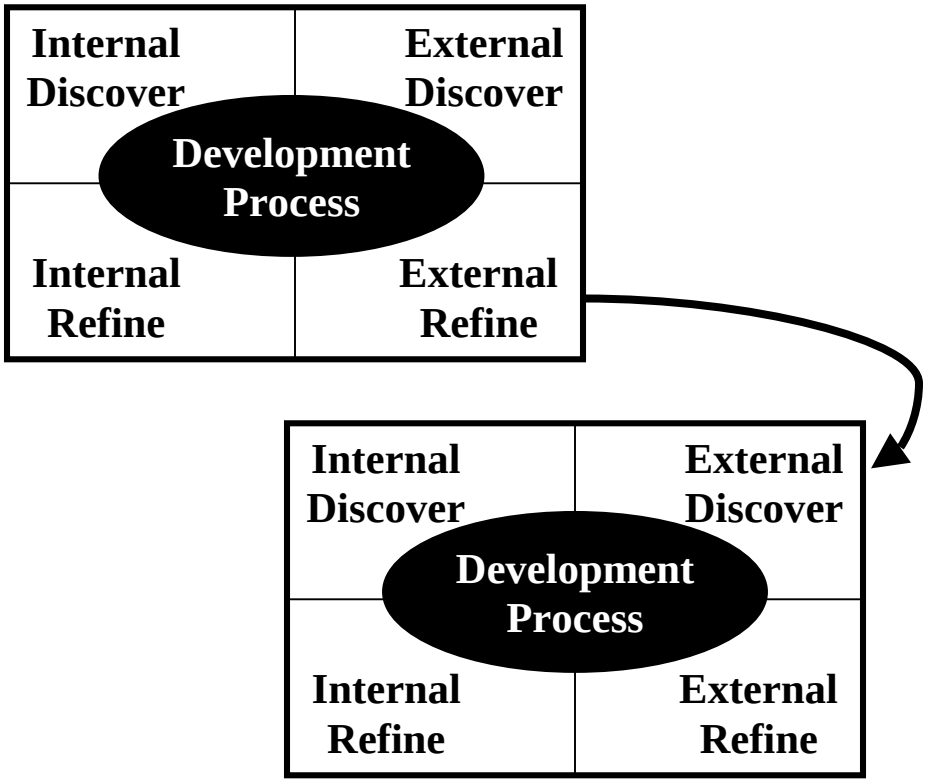

Figure 5. Creative Evolution: Creative phase of design process to a new design process

The KIM is generalized process that combines all of the facets of the PDCA cycle and the External/Internal, Discover/Refinement transitions. The synthesis of these concepts is illustrated in the overall graphic of the Knowledge Insight Model in Figure 6 below:

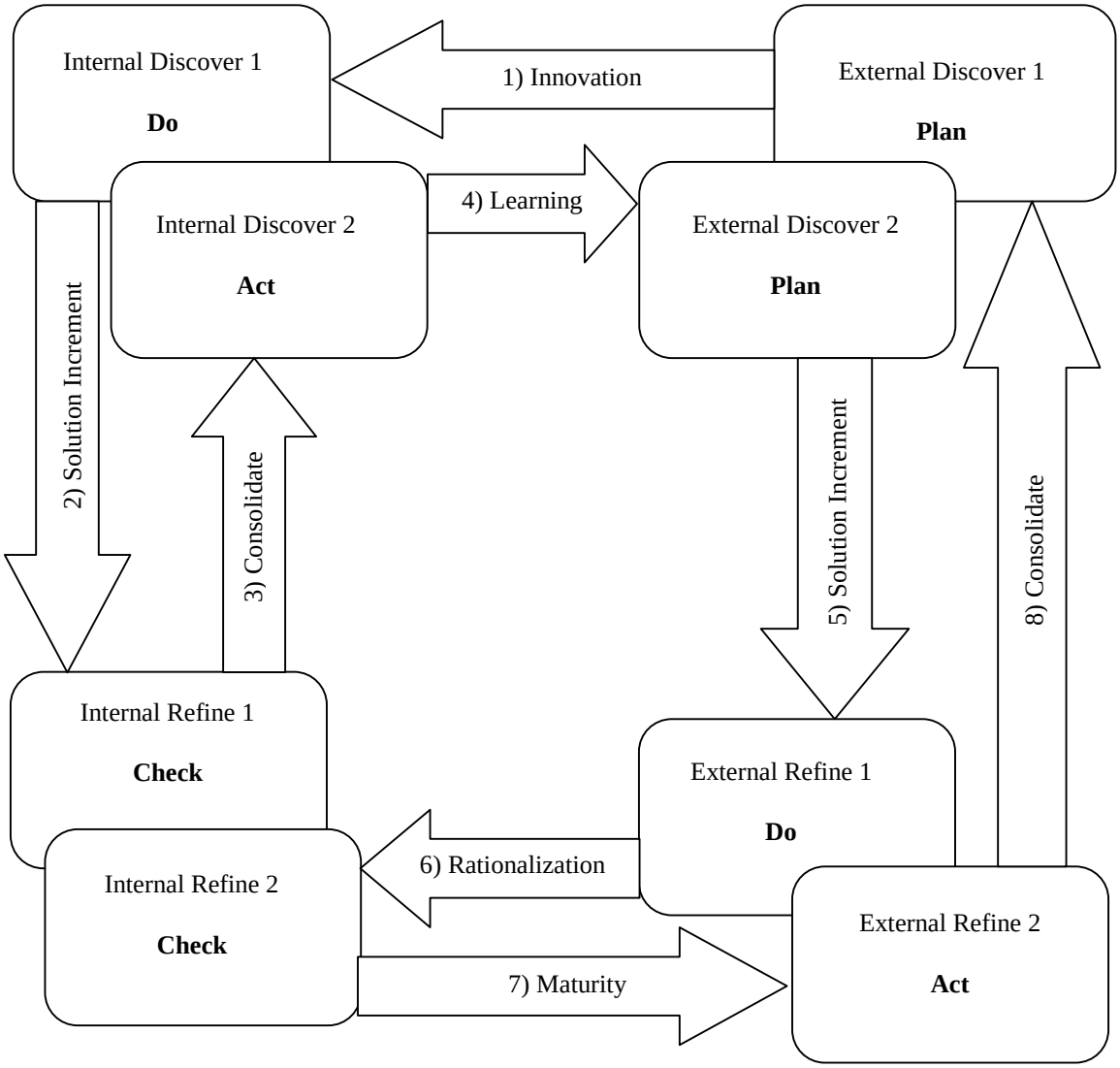

Figure 6. Knowledge Insight Model

4.4 Framer:

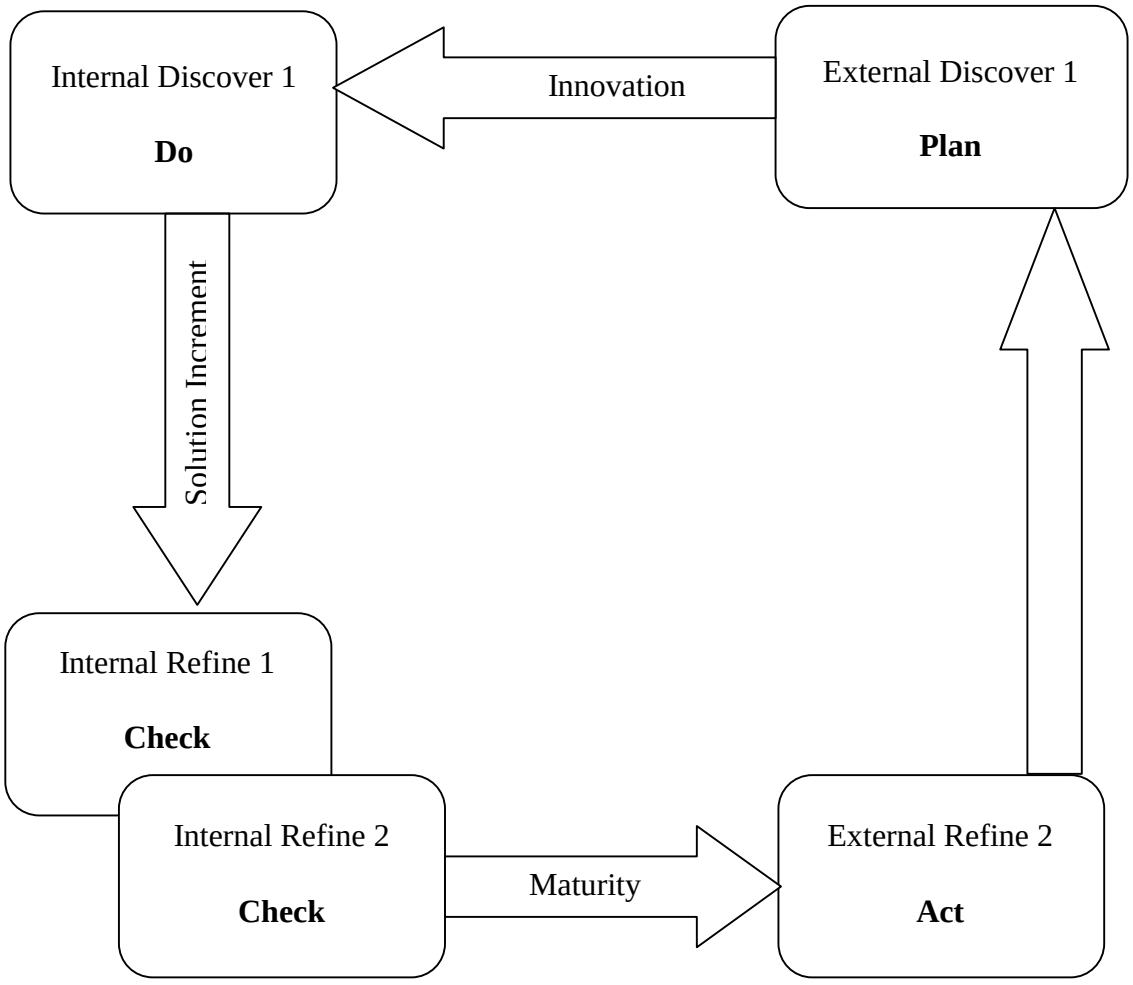

Figure 7. Framer

The Framer is concerned with devising an overall plan for how to accomplish the task at hand. As the name suggests, and as evidenced in Figure 2, the Framer provides the “frame” for the entire KIM process. The Framer must define the problem, establish a

timeline for the project, provide guidelines and requirements of the particular problem, and offer potential solutions that the group should tackle to solve the problem [8]. The Framer corresponds to the Plan phase of the Plan, Do, Check, Act cycle [7]. As such, the Framer devises plans to carry out the project or to solve a problem. The focus of the framer during each of the eight steps is as follows:

- **External Discover 1:** In this plan stage, the Framer must ask: "what problem does the group seek to solve?"
- **Internal Discover 1:** In this do stage, the Framer must break the problem down into smaller tasks to be tackled
- **Internal Refine 1:** In the first check stage, the Framer needs to determine and compile a list of the potential risks in solving
  the problem
- **Internal Refine 2:** In this second check phase, the Framer must determine, based on initial estimates, if the group possess the ability to accomplish the tasks needed to solve the problem. The Framer may also determine what additional resources and/or additional study may be required to complete the project.
- **External Refine 2:** In this act phase, the Framer must establish an initial timeline for the project [10].

4.5 Maker:

The Maker's primary focus is on the physical creation of a product or process. The Maker is responsible for creating any new knowledge or product that is required to

accomplish the objectives, based on initial analysis. The Maker comprises much of the creative force behind the project, and must work closely with the Finder to ensure that there is not a sufficient existing process or product that can be used, thus avoiding "reinventing the wheel."

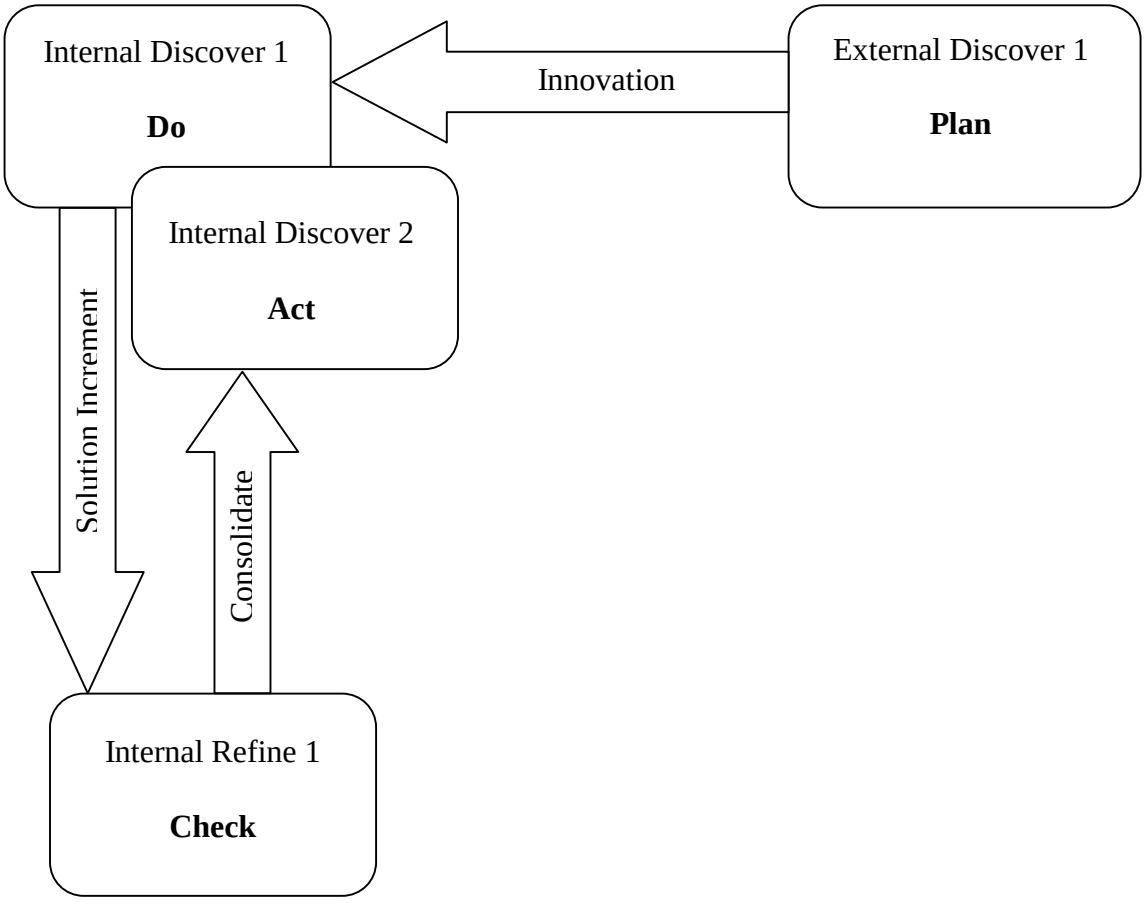

Figure 8. Maker

The steps of the KIM that comprise the Maker are:

- **External Discover 1:** In this plan stage, the Maker must determine what is already in existence that may solve the problem or may contribute to the development of a new concept for the project.
- **Internal Discover 1:** In this do stage, the Maker needs to compile the requirements that the new product or process must meet.
- **Internal Refine1:** In the check phase, the Maker must perform a risk analysis must be performed to assess the risks as determined by the Framer and possibly add risks as seen by the Maker. The Maker will also try to find ways to mitigate risks in this step.
- **Internal Discover 2:** In this act phase, the Maker creates a new product or process [10].

4.6 Finder:

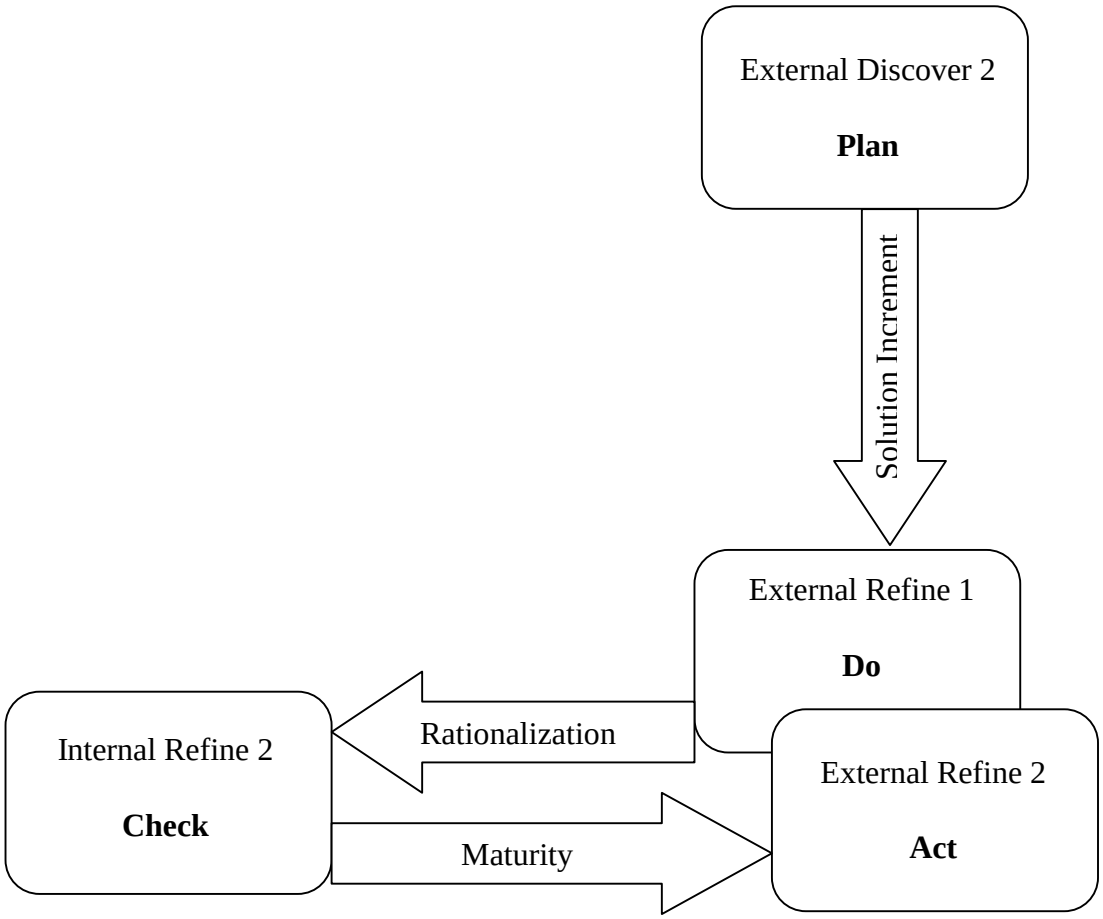

Figure 9. Finder

The Finder must scour all existing knowledge in the field being considered. In doing this, the Finder keeps the Maker from creating "new" concepts that are already proven and in existence elsewhere. In some cases, the contributions of the Finder may be research-oriented, offering ideas to the Maker for how to tackle an existing solution in a new way. The Finder is also responsible for seeking out a need or market for the new

product or process that the group is creating. The details of the steps that make up the Finder phase are as follows:

- **External Discover 2:** In this plan stage, the Finder must determine what information must be found by the Finder.
- **External Refine 1:** In the do stage, the Finder researches information external to the organization; the Finder may at this stage refine his search based on information discovered.
- **Internal Refine 2:** The Finder in the check stage will look internally for past projects or other efforts that may aid in the current project.
- **External Refine 2:** The Finder in the act stage again looks outside the organization to get all remaining information that may aid in project completion; at this point, most of the data has been collected, and this is simply a final look to ensure nothing that may be of assistance has been misse

4.7 Sharer:

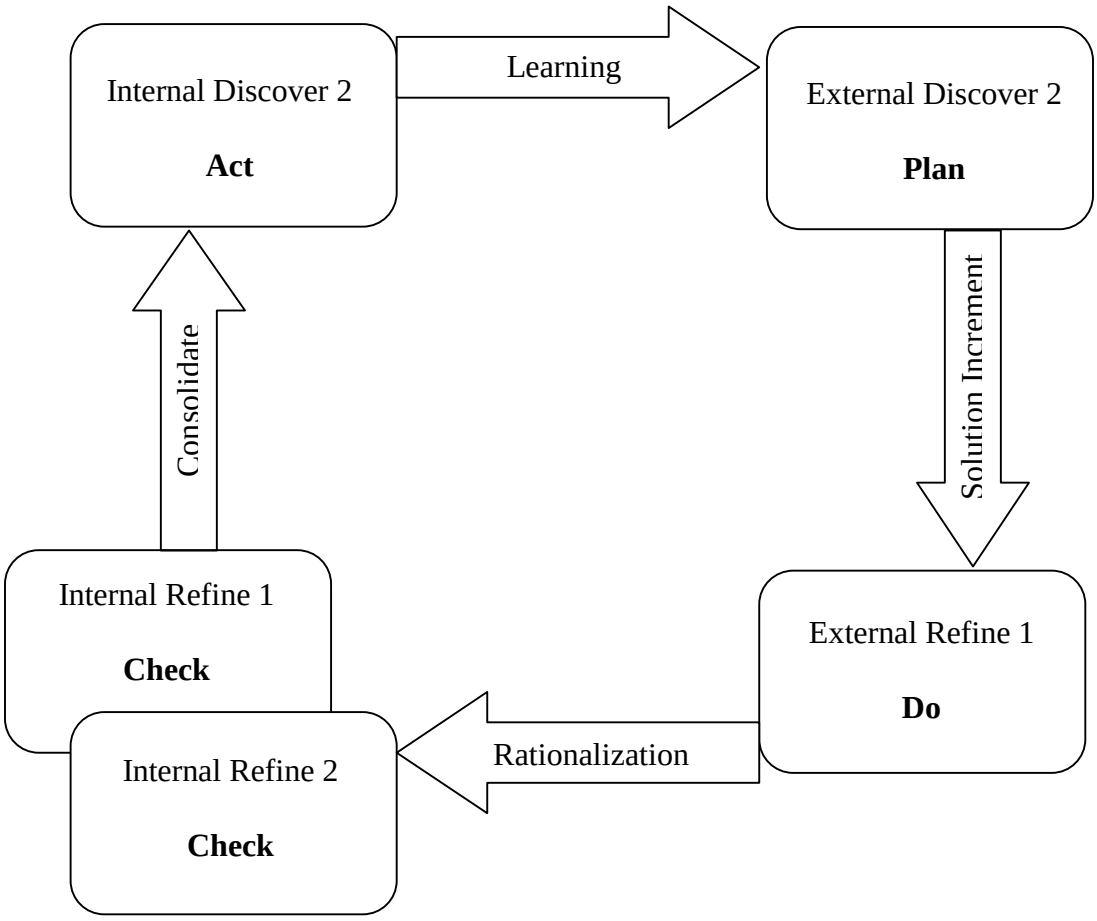

Figure 10. Sharer

The mission of the Sharer is to create and maintain an accurate and up-to-date database of the data collected or created by all other members of the team. The Sharer must also ensure that all involved have appropriate access to the information, to avoid duplicated efforts. The role of the Sharer is the most important of all, as the Sharer is at the center of all internal and external communications. One of the most common problems in many major corporations today is a lack of communication within as well as with outside organizations that contribute toward a common effort (ie – contractors,

companies involved in partnerships, etc…) [2]. In order for an organization to make the Sharer concept work, it must instill within the company a core value of information sharing. Oftentimes, managers or project leaders may possess information that is vital to another department of the company, but are reluctant to share this information, thinking they will lose some real or imagined advantage over a peer (the proverbial "knowledge is power" mentality). If this way of thinking can be eradicated, organizations can more effectively exploit the use of the Sharer concept. The steps that comprise the Sharer mechanism are as follows:

- **External Discover 2:** In this plan stage, the Sharer must determine what types of data will likely need to be collected and who it should be shared with.
- **External Refine 1:** In this do stage, the Sharer builds the first cut of the database
- **Internal Refine 2:** In this check phase, the Sharer may poll those on the project to see if the database provides the information they truly need.
- **Internal Refine 1:** The Sharer alters the database according to feedback from Internal Refine 2, and then checks again to see if the updated database meets the needs of those on the project.
- **Internal Discover 2:** In this act phase, the Sharer seeks out what major changes may be required of the database (ie – if some of the original core parameters that were being distributed have completely changed due to the discovery of new information).

5.0 CLIMBING THE CMM USING THE KIM

5.1 Method Introduction:

The CMM may be viewed as a series of steps to be climbed by an organization. The KIM may be used to iterate through the steps of the CMM, and the KIM may be applied multiple times just to climb one level, or may only need to be applied once to achieve similar results. This chapter shows how each of the personnel filling the four roles of the KIM can work toward the goal of process improvement, and a higher CMM level. It also demonstrates the KIM process in action; how an organization considers both internal and external influences. While the CMM is the focus of this chapter, the KIM may be used for any process improvement, as it is general enough to adapt to various situations.

5.2 CMM Level 1 to Level 2 using KIM:

The transition from level 1 to level 2 is the first time a company makes an attempt at some kind of order and framework for the software engineering process. This transition is illustrated in Figure 11 below:

Each of the four KIM patterns play key roles throughout this transition:

Framer: The Framer's role is to lay the groundwork for process improvement to move up to a higher level of the CMM. The Framer will select a method to organize the software engineering process within the company. The Framer may request research to find an

appropriate method being successfully used in the industry, which may be modified, or appended to with ideas generated from within the organization.

- The Maker supports the Framer during this transition by trying to come up with new ideas and software engineering methodologies internally.
- The Finder supports the Framer during this transition by researching methodologies used by other organizations or those used currently within the organization, but not on a company-wide basis. This research will aid the Framer in making a decision on which method is most appropriate.
- The Sharer supports the Framer during this transition by tracking all the potential process improvement methodologies in a database.

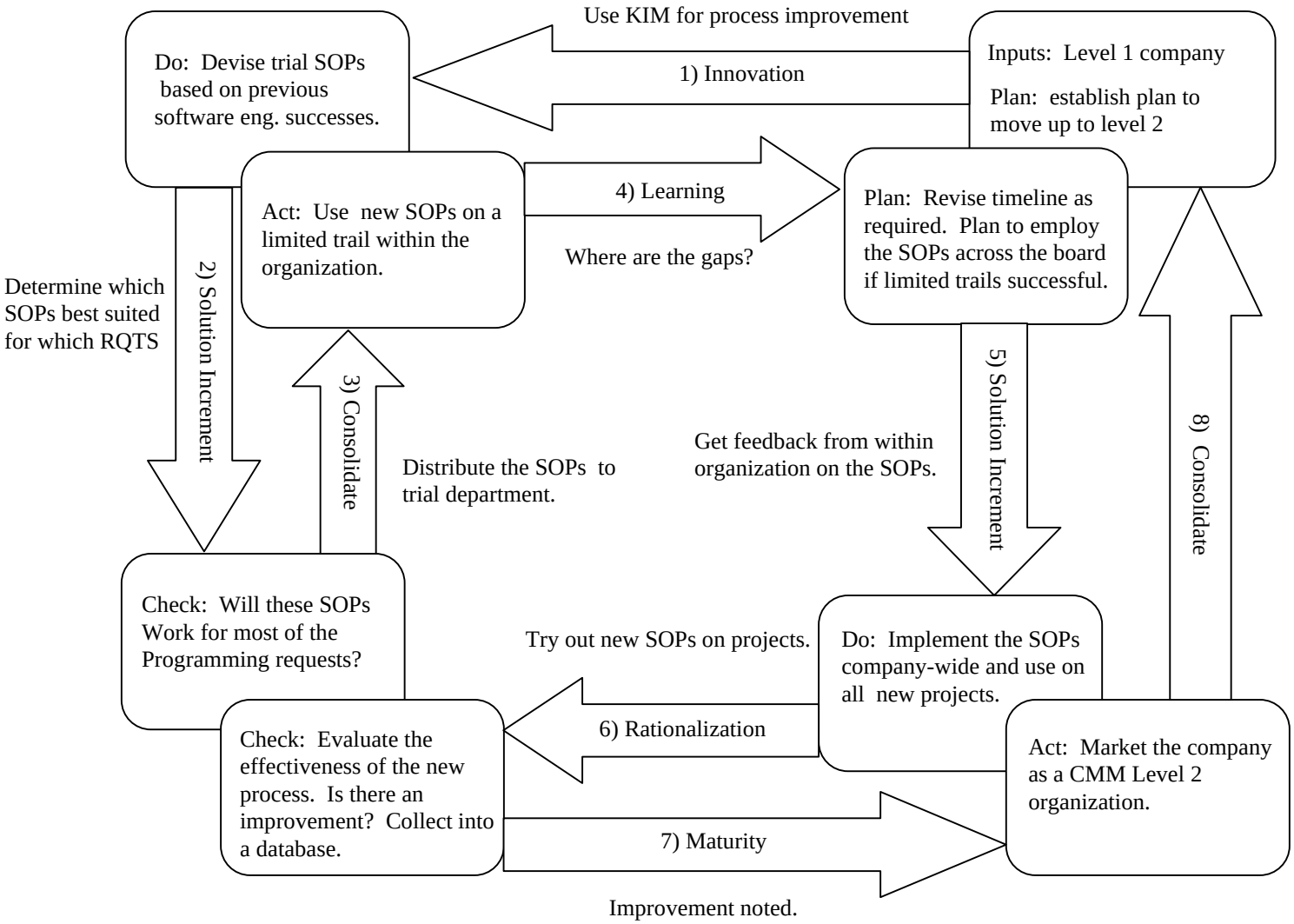

Figure 11. Transition from CMM Level 1 to Level 2 using KIM

Maker: Responsible for devising any new software engineering procedures that may be needed in order to implement a company-wide standardized software engineering process that may be used for various types of software engineering problems. The Maker may create new processes in their entirety, or may generate part of the overall process to compliment existing processes as researched by the Finder.

- The Framer supports the Maker during this transition by providing the final decision on which process improvement method will be used, as well as the tentative timeline, and cost allocated for the process improvement project.
- The Finder supports the Maker during this transition by providing information on methods being used throughout the industry to solve similar software engineering problems.
- The Sharer supports the Maker during this transition by providing the Maker with a database appropriate for collecting information about the various software engineering processes that the Maker creates.

Finder: Attempts to discover all available SE process information for use within the company.

- The Framer supports the Finder during this transition by incorporating the new methods of SE into the overall SE process plan
- The Maker supports the Finder during this transition by providing new information that they create to solve the current problem
- The Sharer supports the Finder during this transition by all the new SE information, both created and existing that was used to solve the

current problem is incorporated into a database for future use

Sharer: Responsible for codifying all compiled information on the SE process used by the company for solving problems and completing projects. Starts a database of SE methods.

- The Framer supports the Sharer during this transition by determining what information should be included in the SE database
- The Maker supports the Sharer during this transition by providing information about which SE processes worked to solve the problem
- The Finder supports the Sharer during this transition by processes and packages the existing SE processes which were used on the project

5.3 CMM Level 2 to Level 3 using KIM:

The transition of a company from level 2 to level 3 is marked by the codifying of the both the software engineering process and management procedures. This transition is shown in Figure 12 .

Framer: Devises a plan to codify the SE policies created in the level 1 to level 2 transition; plan to design standard software process and management procedures

- The Maker supports the Framer during this transition by creating methods to "fill in the blanks" of the standard software process; especially if those being used in industry do not perfectly fit the needs of the company
- The Finder supports the Framer during this transition by collecting both internal and external knowledge of standard software processes that may be incorporated into the final product for the company to use
- The Sharer supports the Framer in this transition by ensuring that all the gathered information on standard software processes is distributed to all throughout the company for approval for a codified system

Maker: Throughout this transition, the Maker must create solutions for the "holes" in a potential standard software process; while a company may select an "off the shelf" process that is used throughout industry, there will usually need to be slight or major modifications to make the system work for the particular company in question

- The Framer supports the Maker during this transition by updating the plan for a standard software process and management system to accommodate the innovations in these areas made by the Maker

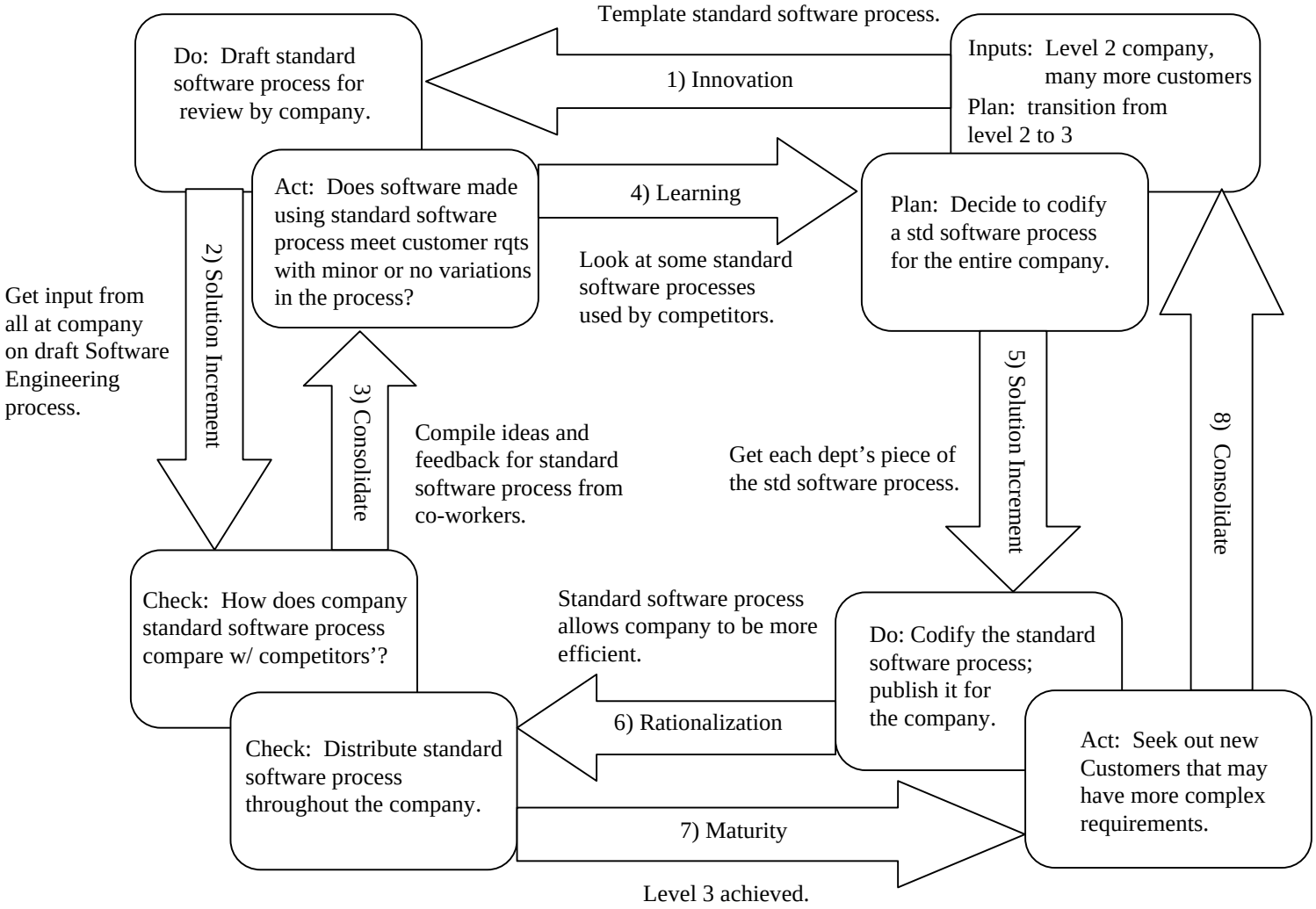

Figure 12. Transitioning from CMM Level 2 to Level 3 using KIM

- The Finder supports the Maker during this transition by checking to see what standard software processes are being used in industry so that the Maker knows what "holes" must be filled via creation of new processes
- The Sharer supports the Maker during this transition by getting feedback from others both inside and outside the company on what may need to be created to ensure a usable standard software process. Others within the company may have ideas that must be shared among all involved in the evolution of processes

Finder: The Finder concerns himself with ensuring that the standard software process being developed will allow the company to produce products that fall within the acceptable range of the average customer

- The Framer supports the Finder during this transition by refining the standard software process and management system to ensure that it will meet the industry standard for software
- The Maker supports the Finder during this transition by creating new processes to ensure the company's standard software process produces quality software products
- The Sharer supports the Finder during this transition by passing along all refinements and process changes until the final process is established

Sharer: The Sharer must ensure that all data on the candidate standard software process as well as a management process is passed around to all within the company. This is the

only way in which all members may have their ideas heard, debated, and incorporated or rejected for the final process.

- The Framer supports the Sharer in this transition by providing the proposed framework for the candidate standard software processes as well as the proposed management processes
- The Maker supports the Sharer during this transition by creating new knowledge, as required, to fill in the blanks and round out the data to be passed around the company
- The Finder supports the Sharer during this transition by collecting all the ideas, both internal and external, for both processes

5.4 CMM Level 3 to Level 4 using KIM:

The transition from level 3 to level 4 is a step toward a truly mature process. A level 4 company thoroughly understands the workings of the company, the software and management processes, and has collected enough data on performance, cost, etc… to predict how changes, such as producing on a new product, or trying out a new process, may affect the company. This can be especially useful in weathering out "bumps" in the corporate road, such as stock drop-offs, poor sales due to weak economy, and the incorporation of a new product line. If these "spikes" can be ridden out, a company can thrive long-term. The process is illustrated in Figure 13.

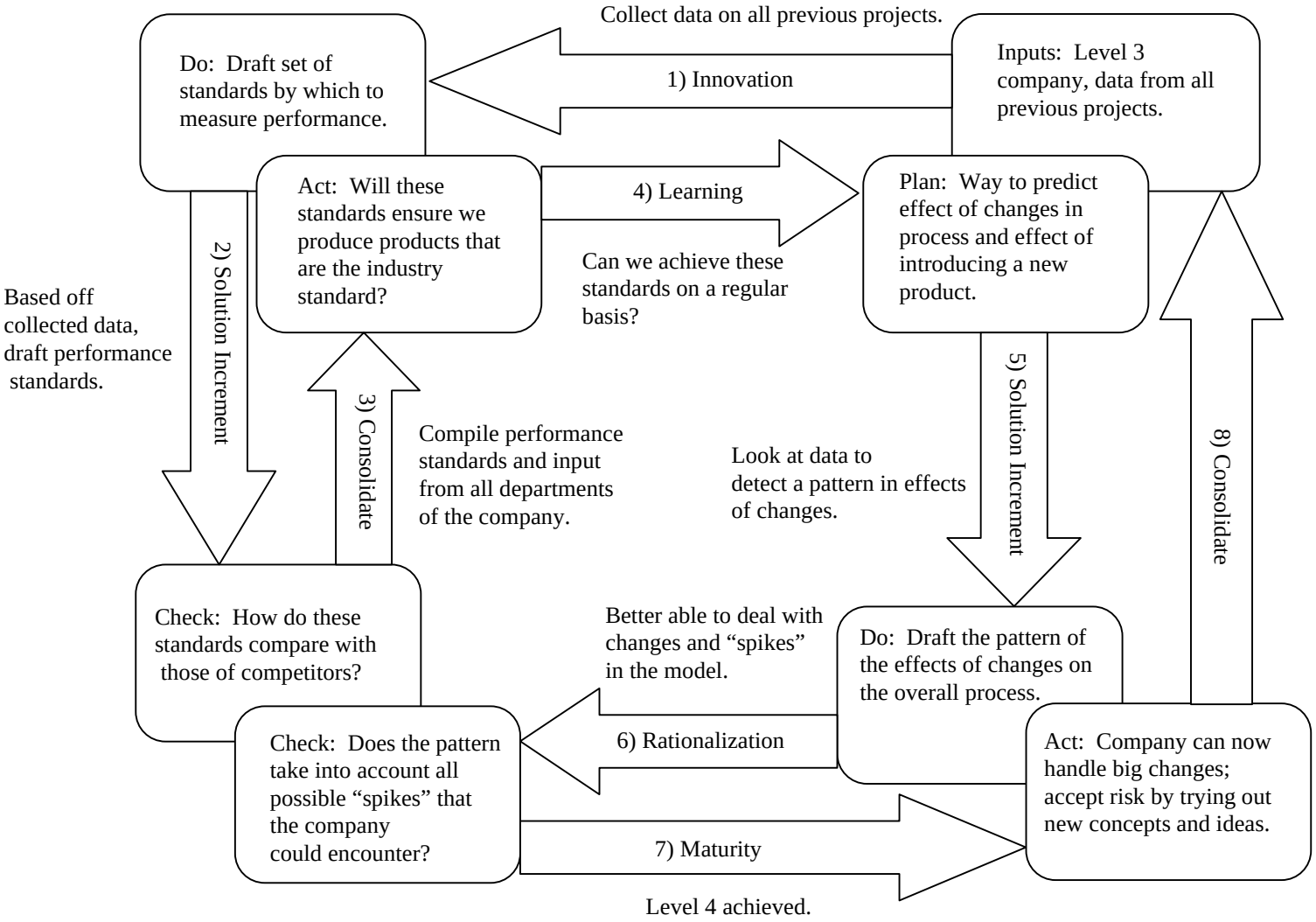

Figure 13. Transitioning from CMM Level 3 to Level 4 using KIM

Framer: The Framer is primarily concerned with establishing what types of performance data to track from previous projects, such as product performance, cost, completion time against industry standards.

- The Maker supports the Framer during this transition by devising new ways to track the various performance measures of the software engineering process.
- The Sharer supports the Framer during this transition by passing along all previous data on the projects completed by the company and compiles a database of this information. The Sharer is also responsible for modifying the database to capture new information, as determined by the other members of the process improvement team.
- The Finder supports the Framer during this transition by researching the industry standards for the type of software the company creates to compare the performance, cost, etc… data against.

Maker: The Maker helps establish a set of standards by which to measure the company based on all previous projects and the external information on the industry standard; also looks for trends in the performance data (ie – effects of changes on the process over time)

- The Framer supports the Maker during this transition by providing the overall plan for measuring performance, cost, project completion time, and the like.
- The Sharer supports the Maker during this transition by collecting and compiling all data, both internal and external, to devise the standards of measure for the company, and guide the Maker as to what data should be collected on a regular basis.

- The Finder supports the Maker during this transition by providing research on what other companies use to measure their products, as well as what measures the organization has used internally in the past.

Sharer: The Sharer must ensure that all data, both that collected internally, and that researched externally, is compiled and placed in a database so that all may see it and decide on what performance aspects of the company should be measured and what those standards should be.

- The Framer guides the Sharer by continuously updating the overall plan for what data to collect; the "plan" for data collection should be a living document until all involved in the process improvement team have reviewed and given input on the matter.
- The Maker supports the Sharer during this transition by reviewing what the sharer compiles and collects and may add or subtract from that to ensure the final tracking methodology will live up to industry standards.
- The Finder supports the Sharer during this transition by providing all collected performance data, both internally and externally, so that it may be distributed to all for review.

Finder: The Finder must continuously seek out ideas on what performance data to collect, both within and outside the company. Research on what is done throughout the industry will ensure that the company stays on the cutting edge

- The Framer supports the Finder during this transition by refining the overall performance data collection plan to incorporate the new knowledge collected

by the Finder

- The Maker supports the Finder during this transition by supplying all the ideas that were generated internally for how and what data to collect.
- The Sharer supports the Finder during this transition by supplying all the previously collected performance data.

5.5 CMM Level 4 to Level 5 using KIM:

Level 5 has been achieved by only a select few in the software engineering field. The company transitioning from level 4 to level 5 takes the data collected and analyzed as a level 4 company and seeks ways to reduce discovered errors. If they find an error rate of say two errors per million lines of code, a level 5 company will attempt to modify their process to get that down to one error per million. This transition is demonstrated in Figure 14.

Framer: The Framer introduces a plan to overhaul the current software and management processes to minimize existing errors.

- The Maker supports the Framer during this transition by supplying ideas that may allow the company to change the current software engineering process to optimize the product and reduce errors
- The Sharer supports the Framer during this transition by ensuring all new ideas for optimizing the process are passed along within the company; creates a database of ideas for optimization

- The Finder supports the Framer during this transition by obtaining all possible information both internally and externally, on how to optimize the software engineering and management processes

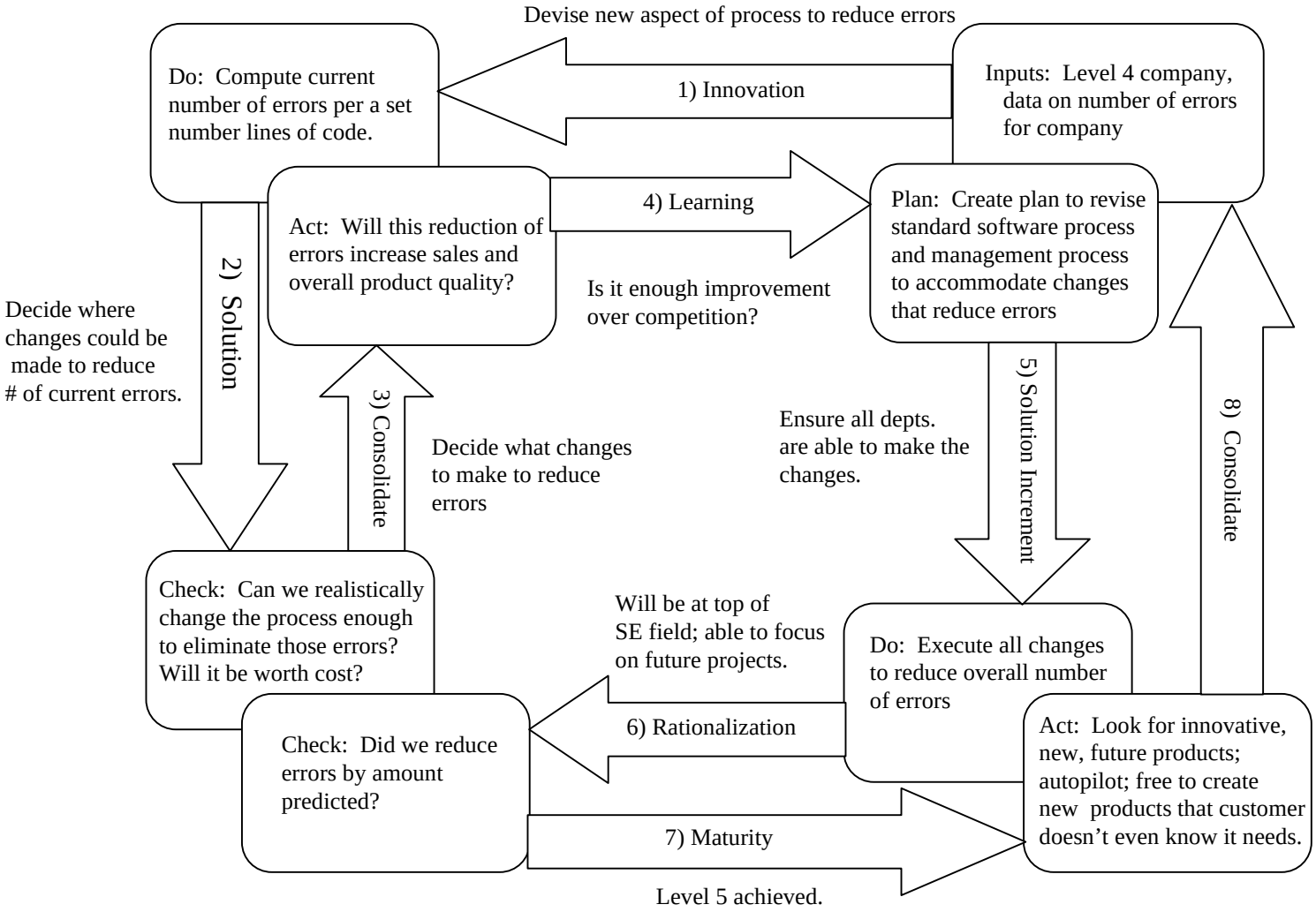

Figure 14. Transitioning from CMM Level 4 to Level 5 using KIM

Maker: The Maker is concerned with finding a way to revise the software engineering process to optimize the product and reduce/eliminate errors

- The Framer supports the Maker during this transition by providing the overall plan and goals of optimizing
- The Sharer supports the Maker during this transition by providing methods used by other companies to optimize their process so the Maker can better decide what changes should be made to the current process
- The Finder supports the Maker during this transition by seeking out information on optimizing processes that have worked for other companies

Sharer: The Sharer must ensure the constant flow and availability of process optimizing data that is found by the Finder

- The Framer supports the Sharer during this transition by providing the goals of the optimization and plan for optimization
- The Maker supports the Sharer during this transition by creating new ideas for optimizing based on information provided by the sharer
- The Finder supports the Sharer during this transition by seeking out the information that the sharer will pass along to the rest of the company to optimize the company's processes

Finder: The Finder checks to see if the optimization will improve sales and demand for the product. This may aid in the final decision of whether to optimize or not based on the investment required to do so

- The Framer supports the Finder during this transition by re-evaluating the plan

to optimize based on reports on whether the optimization will improve demand for the product; the Framer may even scrap the plan altogether if costs to optimize outweigh potential benefits

- The Maker comes up with the actual changes that would have to be made to achieve the next step of optimization
- The Sharer supports the Finder during this transition by providing research information on the cost to optimize based on the potential changes supplied by the Maker as well as potential customer demand changes

6.0 KNOWLEDGE MATURITY MODEL

6.1 Introduction:

The Knowledge Maturity Model (KMM) is a new paradigm, which introduces a layer of control above software engineering and process improvement methodologies. It also introduces a new way to evaluate an organization's level of maturity. The KMM does not address an organization's ability to accomplish a set of key tasks, as the CMM does, but instead looks at the inner workings of how a company tackles software engineering projects. The KMM provides companies a model that allows them to decide what software engineering methodology is best for the particular problem frame they are presented with, customize a method based on the best practices of existing methods, or create a tailored process for the use of the organization. Rather than an organization choosing to use the Waterfall or Spiral Models, XP, RUP, or SCRUM all the time, they would be free to employ any of these methods, or create their own, adjusting from project to project, should they choose. KMM makes this possible by providing the tools of the four roles: Framer, Maker, Finder, and Sharer; a way by which to iterate through any of the software engineering methodologies, the 8-step process of the KIM; and the four states of the problem solving frame: Process Cycle, Roles, KIM, and Inner Mechanism. The KMM provides an essential link between three key aspects of software design: Cognitive Engineering, Software Engineering, and Systems Engineering [9]. KMM addresses cognitive engineering by assigning roles to personnel involved in software engineering or process improvement tasks. KMM includes the aspect of software

engineering by providing a process (KIM) that allows an organization to employ any software methodology to work in a particular problem frame. The KMM addresses the final aspect, systems engineering, by giving an organization a framework that allows them to shift from one software engineering methodology to another, even in the execution of the same project, adding another level of agility to the software development process.

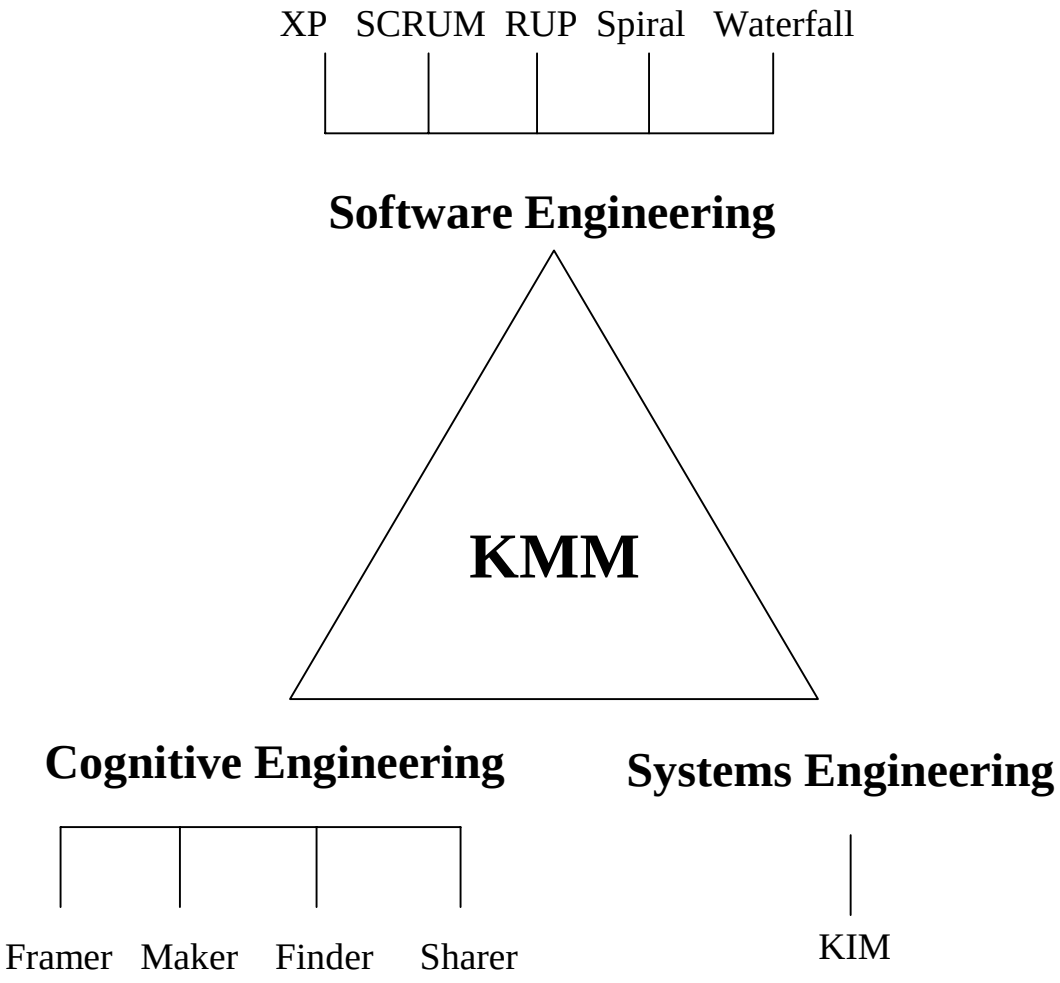

Figure 15. The Three Aspects of Software Design and KMM

6.2 KMM Overview

Dr. Nancy Leveson, in her paper "Software Engineering: A Look Back and A Path to the Future," discusses the fact that modern software engineering has progressed to the point where complex problem frames require more than simply sitting down and hacking out some code. The complexity and scope of modern problems often requires large teams of people to write, integrate, test and validate. Dr. Leveson argues that software engineering can no longer be viewed as an entity unto itself. Modern software design must integrate three key engineering disciplines: Systems, Software and Cognitive Engineering. The argument for employing Systems Engineering can be made by simply reviewing a definition of the discipline itself: "Systems engineering is the branch of engineering concerned with the development of large and complex systems, where a system is understood to be an assembly or combination of interrelated elements or parts working together toward a common objective [19]." Software engineering is an obvious factor in software design, bringing the concepts of programming, verification, validation, and software methodologies, such as the Waterfall and Spiral models, RUP, XP, and SCRUM to the overall system. Cognitive engineering is the least intuitively obvious of the three aspects of software design, but is coming quickly into the forefront. Cognitive engineering includes the people, and their mental abilities and limitations, in the overall equation of software design. This can include the ability of people to work together in teams, the assignment of roles to team members, as well as the limitations of programmers. Some in the software field once argued that software engineering, unlike more "concrete" forms of engineering, that were clearly limited by the materials they

employed or the laws of physics, was only limited by the imagination and skill of the programmer [9]. This romantic notion is not true. There are very real human limitations that do drive how software products are developed [9].

The KMM takes agility to the next level, by allowing companies to select the method that works best for a particular problem frame, by employing the four roles. The KMM is a superset of any software engineering methodology. The four roles, used in conjunction with the KIM generalized process and the awareness of and proper use of the four states of the software design process allow an organization to quickly assume the use of existing models by assigning roles to team members. The tasks and processes of current software engineering methodologies can be decomposed and performed by each of the four roles of the KMM. The KIM process is employed to execute the process of the method of choice. The KMM evaluation scale consists of four level: Process Cycle, Roles, KIM and Inner Mechanism, which are used to assess an organization's abilities.

6.3 KMM Evaluation

The four levels of the KMM consist of the Process Cycle, Roles, KIM, and the Inner Mechanism. Many companies work at the Process Cycle level. These organizations may employ a method such as the Waterfall or Spiral Models. There is a logical, straightforward process to follow for project completion. An organization at the Roles level has taken the next step and defined named roles that all team members understand and employ. Each role has a clear-cut set of duties to perform, and all employees know what basic tasks they must carry out with little or no guidance, for the

problem frame. The third level, KIM, describes a company that employs both roles and an iterative process for software engineering problems. The highest level of maturity in KMM, Inner Mechanism, defines organizations with a highly developed Sharing process. A company at the Inner Mechanism level communicates very effectively both internally and externally. All knowledge for a given project and any knowledge that may even pertain to the project is readily available in a continuously updated database, thus avoiding costly duplication of effort as well as project delays caused by not knowing that another department has already solved a daunting problem. It is also possible for a mature company to move between these levels, as required, to complete the project at hand. Some projects are simple, only requiring the use of a simple Process Cycle, such as PDCA or CAPD. A more complex problem frame may call for the assigning of roles to team members to clarify duties. The KIM state provides both roles and an iterative process. The Inner Mechanism state is employed when a high degree of communication is needed. The Inner Mechanism represents a highly efficient Sharer. Such a level is certainly always desirable, but just as attaining CMM Level 5 is difficult and expensive, maintaining a fully developed Inner Mechanism at all times is not necessary.

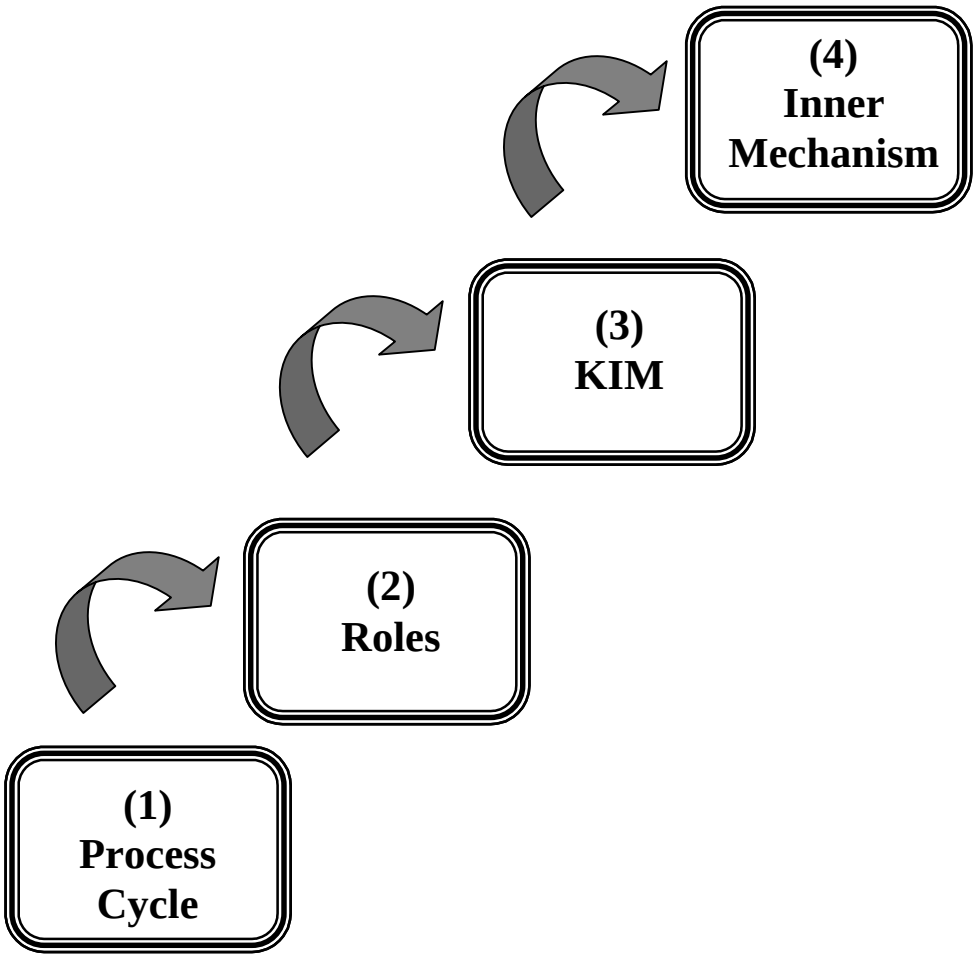

Figure 16. KMM Levels

6.4 KMM as a Superset of The Waterfall Model

The Waterfall Model is one of the classic software engineering methodologies, which has served as the basis for many of the agile methods currently being practiced and developed. The Waterfall Model employs a simple, logical process for software engineering. To demonstrate that KMM is a superset of this methodology, each of the aspects of the PDCA cycle are assigned to each part of the Waterfall Model:

- Requirements Analysis: Plan
- Design: Plan/Do
- Implementation: Do/Act
- Testing: Check
- Integration and Maintenance: Act

The interpretation of these roles is fairly straightforward. The requirements and design phases are attributed to the Plan aspect of the cycle, which correlates to the Framer process of the KIM. Design and Implementation are a part of the Do cycle, which is encompassed by the Maker process. The implementation and integration and maintenance phases are correctly placed into the Act phase of the cycle, which falls into the Finder process. Finally, the testing phase is considered part of the Check cycle, which is controlled by the Sharer process.

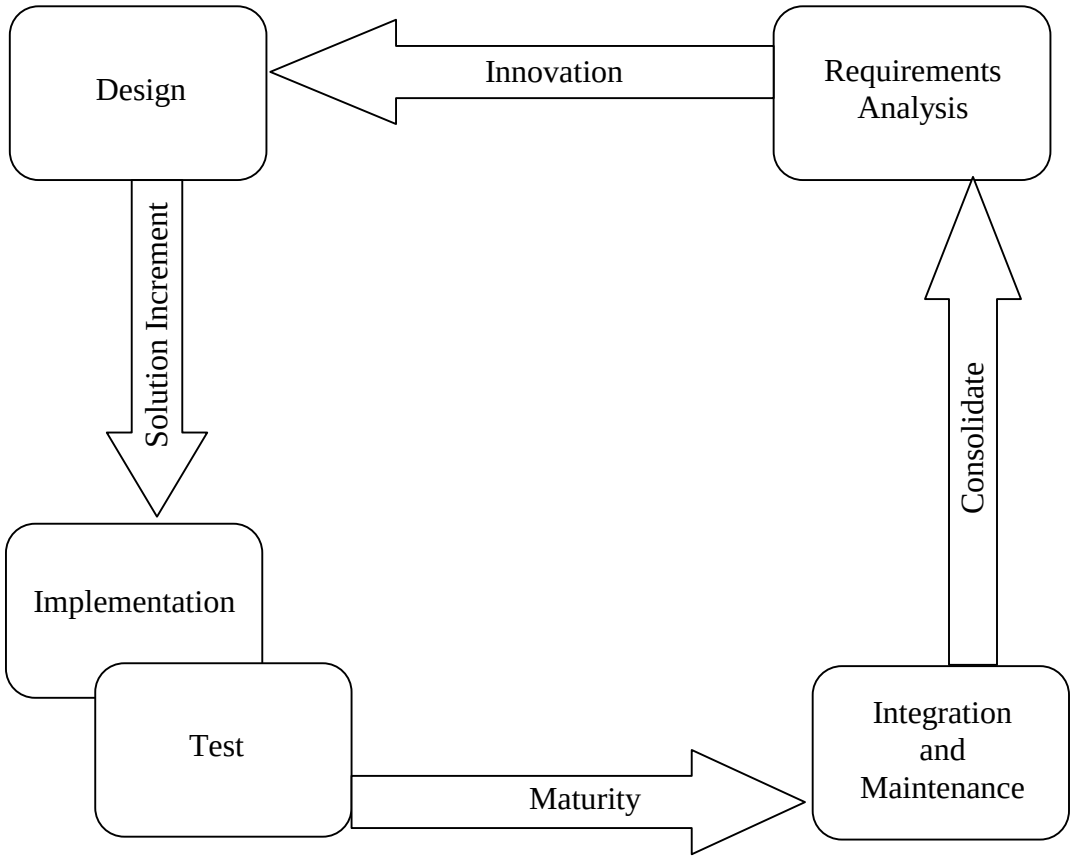

Figure 17. Waterfall Model Instance of KMM

The Waterfall Model only employs the Process Cycle level of the KMM evaluation model. Most organizations using it do not assign roles. Roles have been assigned here, for the purpose of demonstrating it as an instance of the KMM.

6.5 KMM as a Superset of The Spiral Model

The Spiral Model introduced the concept of a truly iterative process for software engineering. Rather than simply completing a given set of steps or phases in a particular order, the Spiral Model may be applied to any phase of the software development process. The Spiral Model consists of six "task regions" which are traversed starting from the center of the diagram (figure 20), and works outward in a clockwise fashion. The blocks in the diagram represent potential starting points for different kinds of projects [14].

The six phases of the Spiral Model can be assumed by the parts of the PDCA cycle as follows:

- Customer Communication: Check
- Planning: Plan
- Risk Analysis: Act
- Engineering: Do/Plan
- Construction & Release: Do/Check
- Customer Evaluation: Act

The spiral model does add iteration to the software engineering process, but is still basically at the Process Cycle level. The Spiral Model employs the Check, Act, Plan, Do (CAPD) cycle, rather than the PDCA that the Waterfall Model uses. The Spiral Model's CAPD cycle is the Sharer pattern, which is a positive step toward an Inner Mechanism.

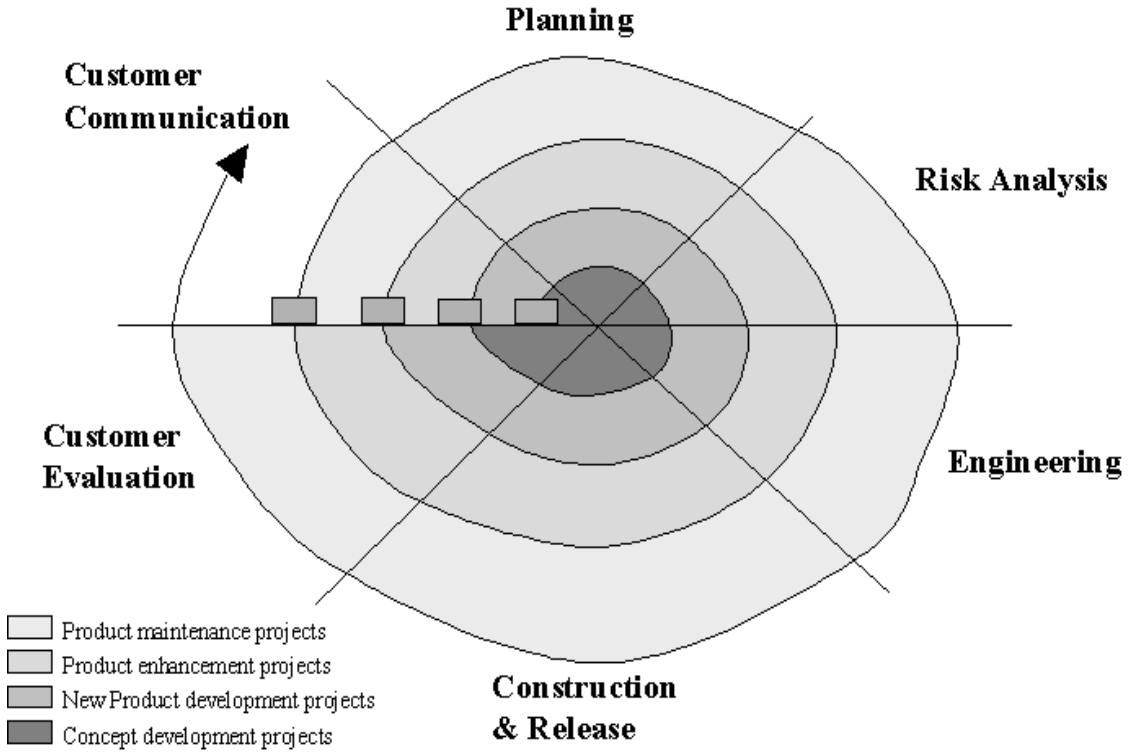

Figure 18. Spiral Model

The Spiral Model starts in the Check phase, obtaining customer communication, employing the Sharer mechanism of the KIM. It then moves into the Plan phase, which correlates to the Planning aspect of the Spiral Model. Risk Analysis appropriately falls into the Act category, and is best suited for the Finder process. Engineering is primarily an aspect of the Do and Plan phases, which is controlled by both the Maker and Framer processes. Construction and Release are part of the Do and Check phases, which are

driven by the Maker and Sharer mechanisms. Finally, the Customer Evaluation phase is part of the Act phase and are also encompassed by the Finder process.

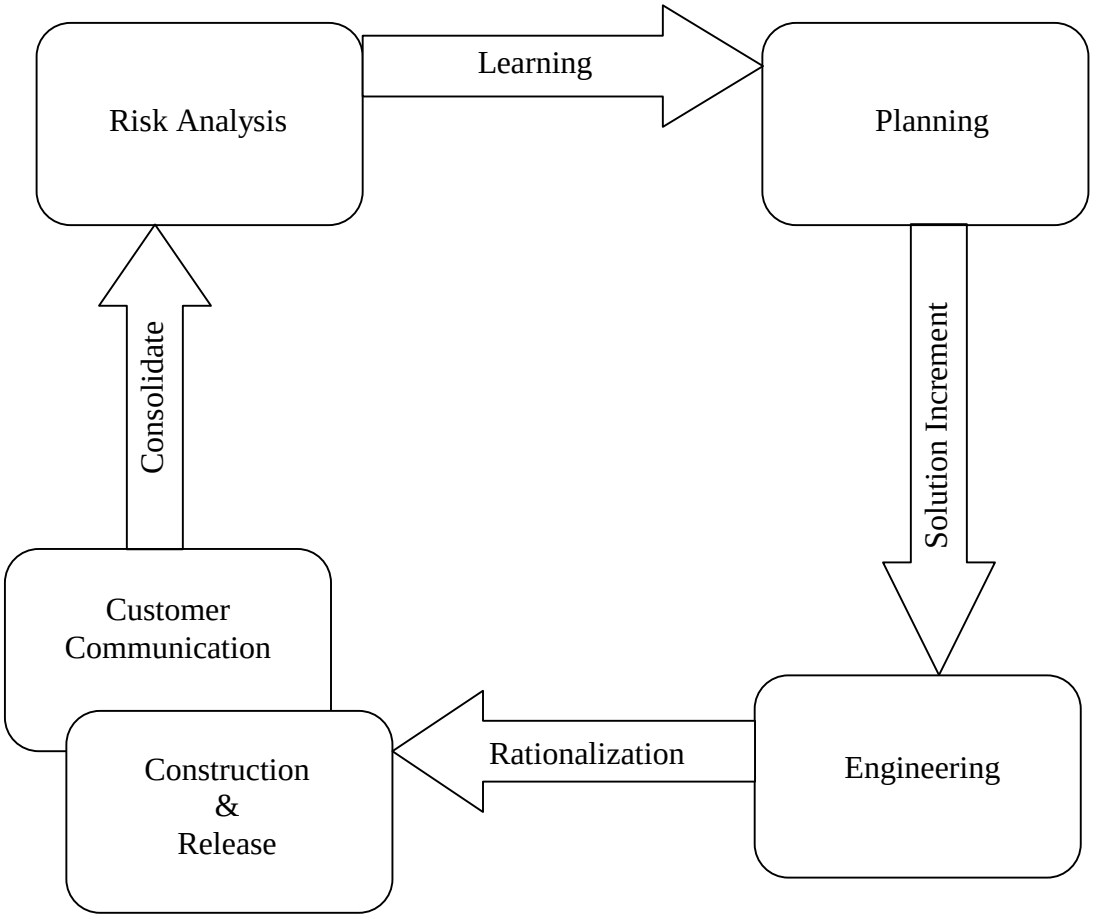

Figure 19. KMM as a Superset of the Spiral Model

6.6 KMM as a Superset of the Rational Unified Process

Much of the software engineering world has moved away from the structured programming concepts of the past. The Rational Unified Process (RUP) was devised in

the structured, procedural environment of the past few decades. Such projects were "big picture," and business based projects. RUP consists of four phases, with each phase having multiple iterations that must be completed before moving onto the next phase [16]. The four phases are: inception, elaboration, construction and transition (figure 15). During the inception phase, the project size is defined, and the "big picture" is established. The elaboration phase consists of defining the problem and analyzing requirements. During the construction phase, the developers actually write the programs to complete the project. Finally, in the transition phase, the software product is delivered to the customer for use [16].

The different aspects of the RUP can be linked to the different phases of the PDCA cycle. They correlate as follows:

- Business Modeling: Plan
- Requirements: Plan
- Analysis and Design: Do
- Implementation: Do
- Test: Check
- Deployment: Act
- Configuration and Change Management: Check
- Project Management: Plan
- Environment: Plan

The KIM is well-suited to representing the RUP, as it can capture the iterative nature of the RUP. Reviewing figure 20, it is clear that different iterations of the RUP

are dominated by one of the four roles of the KIM. The Initial and Elaboration 1 iterations, are dominated by the Framer process. Elaboration 2 and Construction 1 are defined by the Maker phase. Construction 2 and the final Construction iterations are best defined by the Sharer phase. Transition 1 is captured by the Finder process. Finally, Transition 2 is defined by a return to the Sharer phase.

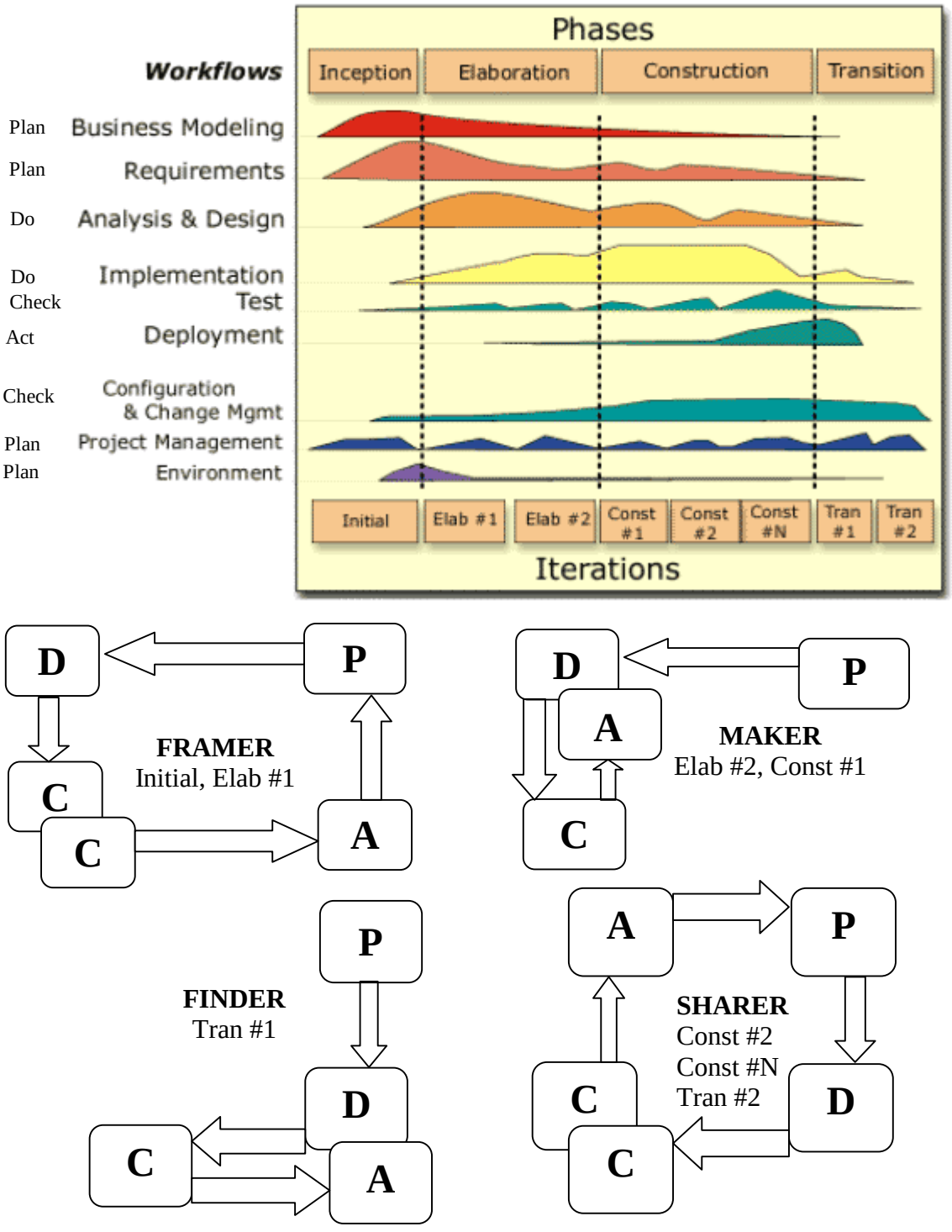

Figure 20. KMM Applied to RUP

6.7 KMM as a Superset of Extreme Programming (XP)

Extreme programming (XP) takes a bottom-up approach to programming projects. XP is agile enough to allow for major changes to be made very late in the project development. XP does not need every requirement be known and completely understood in the first phase of the project, as in RUP. XP consists of twelve practices that help ensure its flexibility. These twelve practices can be assigned to each of the four aspects of the PDCA cycle as follows:

- The Planning Game: Plan
- Small Releases: Check
- System Metaphor: Plan
- Simple Design: Plan, Do
- Continuous Testing: Act
- Refactoring: Do
- Pair Programming: Do
- Collective Code Ownership: Do
- Continuous Integration: Act
- Forty Hour Work Week: Plan
- On-Site Customer: Check
- Coding Standards: Plan, Do

When XP is represented in the KMM framework, the cycle and the twelve practices must be separated. The cycle consists of Planning, Implementation, Testing, and Prototype Release. The twelve practices act as attributes of steps in the cycle.

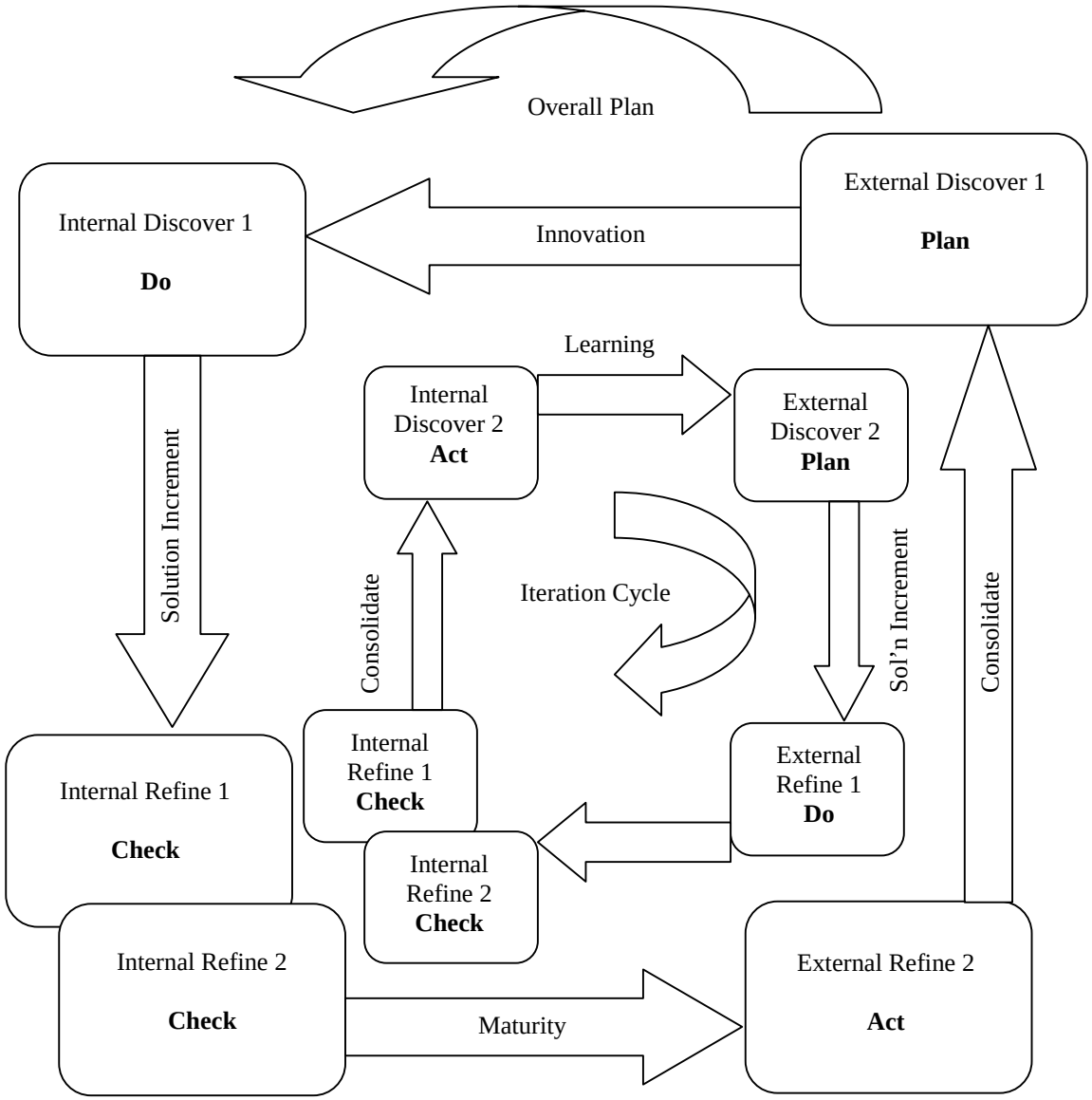

Figure 21. KMM Applied to XP

Figure 21 demonstrates the XP cycle as a subset of the KMM. XP consists of one overall project plan with multiple iterations lasting from one to three weeks, each

ending with the delivery of a working prototype, until the final product is produced and delivered. XP is depicted in KMM as two processes, the Framer for the overall plan or "outer loop," and the Sharer, which defines the multiple iterations, occurring on the "inner loop." Some of the twelve practices are attributes of the cycle, rather than comprising the cycle itself. The practices of pair programming and refactoring are simply attributes of how coding will be accomplished. Likewise, the forty-hour workweek, coding standards, and collective code ownership are simply guidelines that the Framer and Maker use to ensure a healthy and orderly work environment. The On Site Customer attribute describes the relationship between customer and client, but is not a part of the software development cycle, and so is also not included in the actual KMM model.

6.8 KMM as a Superset of SCRUM

The concept of SCRUM has its origins in the sport of Rugby. In Rugby, SCRUM is the term used to describe the group of players that move together down the field. SCRUM is based around the use of a daily SCRUM or 15 minute meeting of the team members, which is intended to allow members to share what they have done since the last SCRUM, any obstacles that they need to work through, and what they intend to do before the next SCRUM [18]. SCRUM takes all the customer requirements and desired functionality and compiles them into a product backlog. The product backlog is then expanded into tasks by team. SCRUM uses "sprints" or 30 day increments in which the team produces a working prototype for evaluation by the customer. This allows for

changes to occur as the product is being developed, rather than trying to make major changes in functionality after all the coding is already complete, as is often the case in more structured methods, such as RUP. During the sprints, the team members work through sprint backlog in order to accomplish all the desired requirements of the customer, as defined in the product backlog. SCRUM may be used independently, or may be used as a "wrapper" around other engineering processes, such as XP [18]. The workings of SCRUM are illustrated in figure 21 below:

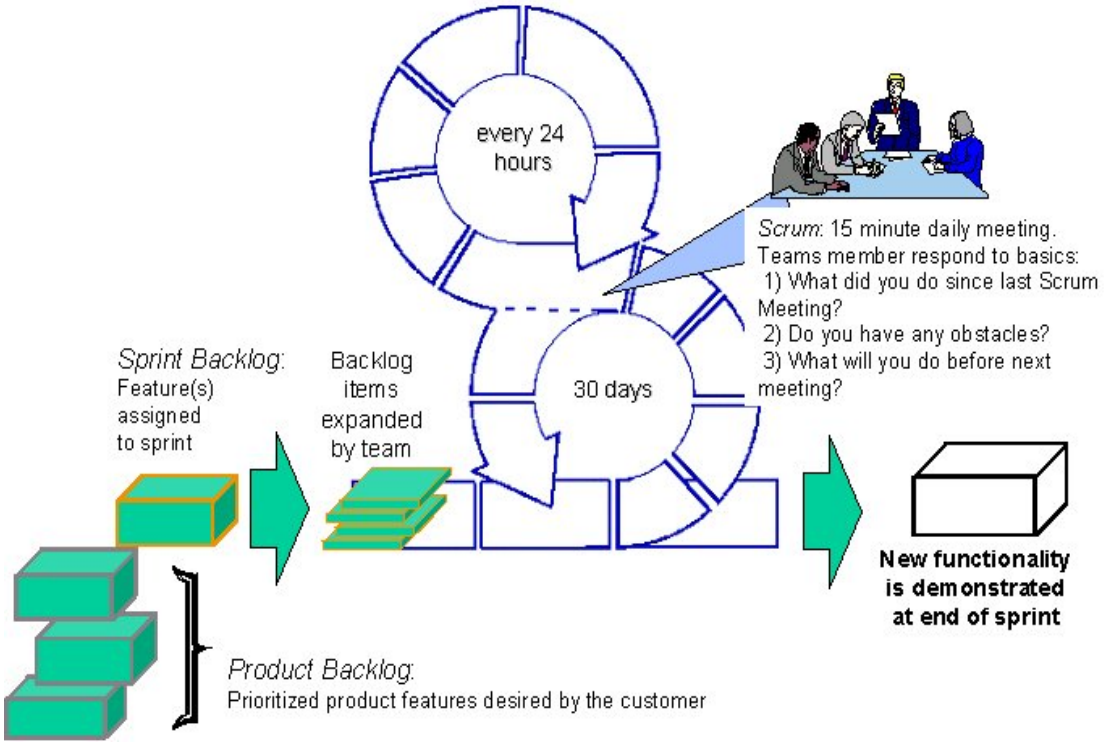

Figure 22. SCRUM [18]

SCRUM can be modeled by the KMM as depicted in Figure 23. The outer cycle, the Framer model, represents the activities of the overall plan for the project. The inner

cycle, the Sharer model, depicts the activities of the 30-day sprint. This Sharer model is iterated through repeatedly until the final product is released.

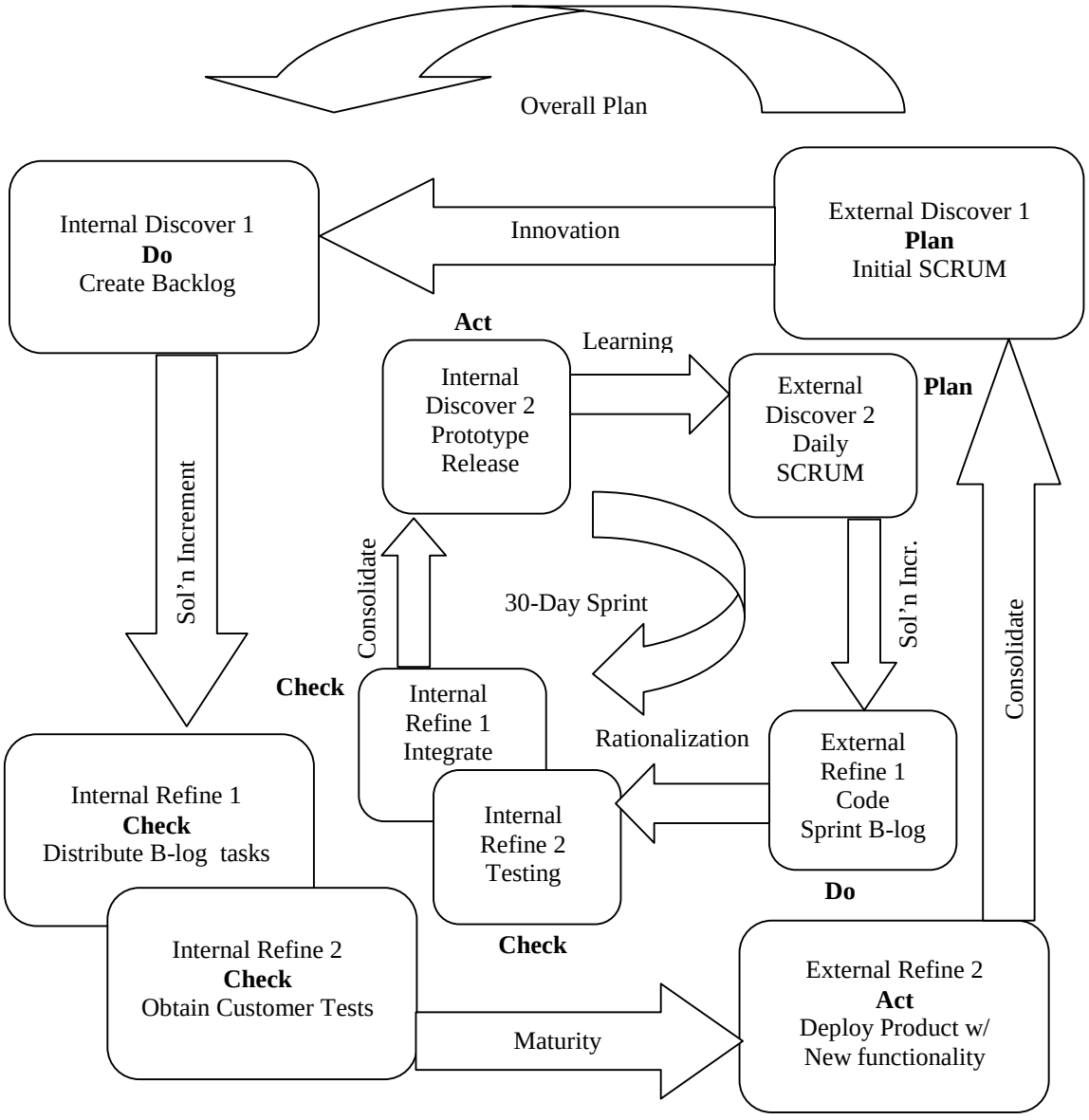

Figure 23. KMM as Superset of SCRUM

7. NASA AND THE KMM

7.1 *Columbia*

The loss of the Space Shuttle *Columbia*, STS-107 and her crew, on 1 February 2003 revealed fundamental flaws in the very fabric of the National Aeronautics and Space Administration (NASA). Many would argue that the loss of the vehicle and crew were due to foam from the External Tank striking the leading edge of the left wing, thus creating a hole in a Reinforced Carbon-Carbon (RCC) panel, allowing extreme heat to penetrate into the wing and ultimately to burn and break up the entire vehicle. This is certainly the physical cause, but the root of the problem lies in the culture of the NASA workforce, an extremely poor communication network within the organization, and the lack of a working, fully accessible knowledge system across NASA. In spite of the fact that many of the departments of NASA, and their contractors, are rated in the top of their fields, (the Onboard Shuttle Software Group maintains a CMM Level 5 status), it is clear that NASA is not at the peak of maturity when considered on the Knowledge Maturity Model (KMM). NASA lacks the Inner Mechanism, the easy flow of information and knowledge within an organization, so that all who require information can obtain in, with little or no bureaucracy.

7.2 *Challenger / Columbia* and the NASA Culture

The Columbia Accident Investigation Board came to a startling conclusion. The culture of NASA prior to and during the *Columbia* tragedy was almost identical to that

before the loss of the *Challenger*, STS-51L, and her crew, on 28 January 1986. The board that investigated the loss of *Challenger*, known as the Rogers Commission, determined that over many successful flights of the Space Shuttle fleet, NASA had slowly begun to accept what were once launch-aborting problems as simple maintenance issues to be studied, rather than used to assess a GO/NO-GO for launch. A contributing cause was that these problems would occur, and yet missions were successful in spite of them. However, many engineers at all levels were adamant about the need to look at the O-Ring design and the effect of cold weather on them. The engineers who attempted to communicate their concerns were silenced by the management of NASA, under pressure to maintain a demanding schedule of Shuttle launches. The breakdown of communication between the engineers and management at NASA ultimately led to the loss of *Challenger* and her crew. The Rogers Commission recommended that the Space Shuttle Program be moved from Johnson Space Center to Washington D.C. "with the aim of preventing communication deficiencies that contributed to the *Challenger* accident [2]." However the non-communicative culture of pre-*Challenger* NASA was pervasive. The changes instituted following *Challenger* were fought at every level. "Cultural norms tend to be fairly resilient…the norms bounce back into shape after being stretched or bent. Beliefs held in common throughout the organization resist alteration [2]." Thus NASA found itself back in the pre-*Challenger* culture. The key was a severe lack of communication, "the structure of NASA's Shuttle Program blocked the flow of critical information up the hierarchy [2]." The communication problems were not only from bottom to top, but also the other way around. The loose knit group of engineers which

formed via email the day after the launch of Columbia, the "Debris Assessment Team," desired further data about previous missions involving foam strikes, but were denied access to it based on their low "paygrade" and the fact that the departments some of them worked in were technically outside of the Shuttle Program and were thus not privy to such data. In fact, this group of engineers went so far as to contact the Department of Defense to obtain satellite imagery of the *Columbia* while in orbit, to determine if there was damage, but this request was stopped by NASA, and the engineers involved were reprimanded for going around the NASA hierarchy. Communication was not the only flaw, there was much confusion about the roles of personnel within the organization. "Also, the Board found many safety units with unclear roles and responsibilities that left crucial gaps [2]." This lack of clear roles caused "ambiguous working relationships [2]" within NASA and its contractors.

7.3 New Culture with KMM

Certainly, there are many aspects of the NASA culture which must be corrected, and they cover the entire spectrum, from psychological issues, to business and financial practices, and even undue political pressures. The KMM cannot solve the problems in all of these areas, but it does provide a template for a working knowledge system that makes knowledge transfer simpler and less bureaucratic. The KMM introduces the four roles, Framer, Maker, Finder, and Sharer. Employing these roles at NASA will greatly reduce job ambiguity and define the relationships between people in the organization. However, the greatest benefit would be gleaned from the role of the Sharer. An organization with a

fully functioning Inner Mechanism, where the Sharer function is operating at the highest possible level, ensures that knowledge is made readily available to all who may require it. The Sharing function would become the backbone at NASA. Introducing an extensive, "living" database, accessible to all within the organization, would eliminate the problem encountered by low-level engineers during the *Columbia* flight, denied data based on paygrade. A fundamental change of culture is needed, however, the KMM will establish the framework for the rise of a new culture, steeped in knowledge sharing, distinct roles, and clear communication lines at every level.

8. CONCLUSIONS AND FUTURE WORK

8.1 Conclusions:

Software design and process maturity are rapidly changing areas of interest in the computer science world. There exist a myriad of software engineering methodologies; treating these as tools, rather than the proverbial “silver bullet” of the discipline, requires that there be a level above, to make efficient use of all these potential tools. Many current process maturity models are not well suited to properly evaluate organizations employing agile methods, thereby not giving a true picture of the potential abilities of such companies.

In this thesis, a layer above software engineering methodologies, the Knowledge Maturity Model, was developed. This layer provides a needed tie between three aspects of software design which have, to date, been addressed independently: software engineering, cognitive engineering, and systems engineering. Simply recognizing the requirement for these three aspects to be executed together enables the shift of focus needed to address many of the problem areas in software design. An organization that adopts the KMM ensures that all team members understand the functions and interaction between the four basic cognitive roles of Framer, Maker, Finder and Sharer. This understanding allows such an organization to look at the various software engineering methodologies and decompose their steps and phases into the four roles, allowing the team members who assume the duties of the roles to know what tasks they are required to perform, and how they should interact with other team members, as part of an overall

systems engineering style plan. In doing this, the organization reduces ambiguity and makes it much simpler to decipher methodologies which may otherwise appear daunting to adopt. This thesis also demonstrated the use of the Knowledge Insight Model (KIM) as a method by which to climb the Capability Maturity Model (CMM). This verified the ability to use the KIM, which is at the heart of the KMM, for process improvement. The KMM took this concept one step further by demonstrating a new, four-level maturity model. The new model better reflects true maturity within a software engineering organization, by measuring the ability of such a company to employ roles, processes, and ultimately communicate knowledge, in order to create the best possible software products. It essentially measures the potential of an organization to tie together software engineering, cognitive engineering, and systems engineering effectively.

The KMM was shown to be beneficial to organizations in need of improved knowledge management processes. By analyzing the culture of NASA, before two tragedies, the loss of the *Challenger* and *Columbia*, the importance of the Sharer function and a fully working Inner Mechanism was examined. Many departments and contractors of NASA are considered at the pinnacle of their fields, and yet they suffered from two key weaknesses: the lack of defined roles and the lack of a fully-accessible knowledge system led to the loss of two Space Shuttles and 14 crew members. It is clear that NASA would benefit from the use of the KMM, in clarifying roles, and implementing an effective knowledge system that would ensure best possible dissemination of knowledge to engineers and management.

8.2 Future Work:

Experiments could be conducted using an actual software engineering company, providing one team with the tools of the KMM, and using a control team, which would employ their normal methodologies. The experiment would require that the teams create multiple software products using a different software engineering methodology each time, to see if the KMM provides a measurable time and/or efficiency improvement. A similar experiment could test the process improvement aspect of the KMM by employing test and control teams, to determine if such a project is simplified by using the KMM.

A very interesting test would be the introduction of the KMM to NASA to see if such a complex and very technically competent organization can benefit from its use to build a more effective culture.

Today's software engineering environment is in a constant flux. New methodologies seem to be developed overnight. Many organizations desire a way to achieve process improvement, without using costly consultants. Abstracting these concepts to a higher level allows the same basic concepts of the four roles, the PDCA Cycle and a simple iterative process to be employed for both of these tasks. Making use of a simple layer, above existing methodologies for software design and process improvement, makes these complex tasks far less daunting.

The KMM shows potential to be employed in the realm of knowledge management, as briefly explored in this thesis, in conjunction with the culture at NASA. Further research and experimentation may lead to breakthroughs in knowledge management which could be employed across multiple fields in academia and industry.

LIST OF REFERENCES

[1] Brooks, Frederick P. JR. "No Silver Bullet: Essence and Accidents in Software Engineering." *IEEE Computer Society Press.* Los Alamitos, California 1987.

[2] "Columbia Accident Investigation Board." *National Aeronautics and Space Administration.* 2003.

[3] "Capability Maturity Model® for Software (SW-CMM®)." *Carnegie Mellon Software Engineering Institute*. 24 July 2002. Carnegie Mellon University. 13 February 2003. <www.sei.cmu.edu/cmm/cmm.html>.

[4] "Capability Maturity Model Integration® (CMMI®)." *Carnegie Mellon Software Engineering Institute.* 18 August 2003. < www.sei.cmu.edu/cmmi>.

[5] Fishman, Charles. (Dec 96/Jan 97). "They Write the Right Stuff." *Fast Company*. 06, 95.

[6] Gremba, Jennifer, and Chuck Myers. "The IDEAL^{SM}Model: A Practical Guide for Improvement." *Bridge.* Issue 3, 1997.

[7] Honeycutt, Thomas L. *Knowledge Enabling Organon: Knowledge Executive Officer*. North Carolina: KEI LLP, 2001.

[8] Honeycutt, Thomas L., and Sarat M. Kocherlakota. *A Knowledge Insight Model Framework for Knowledge Discovery and Data Mining.* North Carolina North Carolina State University. 2002.

[9] Leveson, Nancy. *Software Engineering: A Look Back and a Path to the Future.* Massachusetts Institute of Technology. No date. http://sunnyday.mit.edu/cacm.html.

[10] Menjoge, Zehlam. *Software Development using the Knowledge Insight Approach.* North Carolina: North Carolina State University, 2003.

[11] Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. "Capability Maturity Model, Version 1.1." *IEEE Software*. 10.4 (1993): 18-27.

[12] Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. "The Capability Maturity Model: Guidelines for Improving the Software Process." Addison-Wesley Publishing Company, New York, New York, 1995.

[13] "PDCA Cycle." *HCi.* 20 December 2003. <http://www.hci.com.au/hcisite2/toolkit/pdcacycl.htm#Plan-Do-Check-Act>.

[14] Pressman, Roger S. Software Engineering: A Practitioner's Approach. McGraw-Hill Higher Education, New York, New York, 2001.

[15] "Reaching CMM Levels 2 and 3 with the Rational Unified Process: Rational Software White Paper." *Rational*. No date. <http://www.rational.com/media/whitepapers/rupcmm.pdf> .

[16] "What is the Rational Unified Process?" *RevMedia Inc.* http://www.revmedia.com/process_what_is_rup.php. 2003.

[17] "RUP." *Novasoft.* http://nsuml.sourceforge.net/zebra/zebra3.html. 2003.

[18] "SCRUM Development Process." *Advanced Development Methods, Inc*. www.controlchaos.com. 2003.

[19] "System Engineering Definitions." *University College London*. http://www.ucl.ac.uk/syseng/pages/sedef.html. No date.