AGILE METHODS – CMMI-SW: LINKING THE TWO EXTREMES

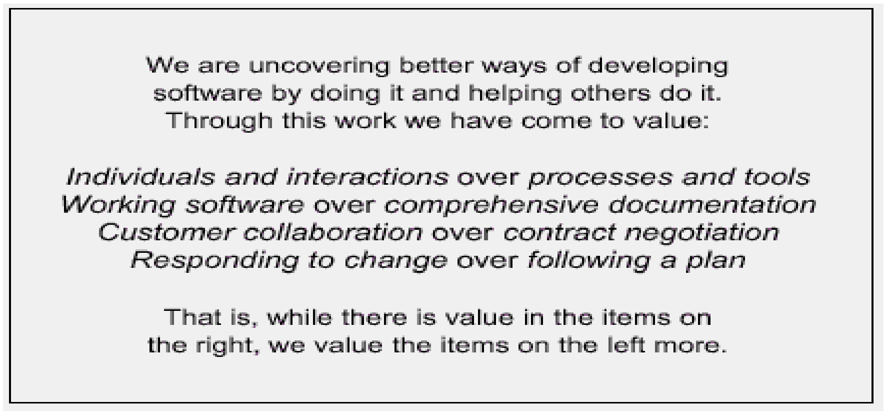
Dr. Ghazi I. Alkhatib
Qatar College of Technology
Doha, Qatar

ABSTRACT

The article explores the possible link between agile methods and CMMI-SW. Major issues incited from such link are: documentation, communication, user involvement, and application types. Interrelationships among these issues are highlighted as well, such as the relationship among the first there issues. Then three strategies for such link are proposed: separatism, integration (hybrid/concurrent, front-end, and back-end), and re-engineering of both methods. Even though current research on the integration of the two focuses on the hybrid approach, this article advocates the concurrent approach, and provides a framework for continuous CMMI-SW improvement towards agility. It is hoped that the research of this paper will encourage the Gulf nations to seek certification and establish local training and certification centers to face globalization challenge in the area of software development.

INTRODUCTION TO AGILE METHEDS AND CMMI-SW

Agile methods (AM) simply threw out old software development life cycle (SDLC) concepts associated with the waterfall model, such as milestones, deliverables, blue prints, road maps, as well as CASE terminologies, such as front-end, back-end, and re-engineering. It then capitalized on newer concepts, which were evolving during the last decade, such as client-led design, client-centered design, participatory design, joint application design, rapid application development, and the different SDLC models, such as prototyping, incremental, and evolutionary. AM eliminated lines of demarcation between traditional SDLC phases through the use of releases and iterations, see (Beck, 1999) for example. AM efforts were culminated by the agile manifesto as shown in Figure 1.


We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Figure 1. The Agile Manifesto

Capability Maturity Model Integration Software (CMMI-SW) evolved from the multiple models of CMM as a single reference model to make it easier for organization to follow model requirements. Most of previous research was done on the link between AM and CMM family of models, while this research used the new CMMI. CMMI has two representations: staged and continuous. The first representation resembles the previous CMM by starting the discussions on the five stages, and to assist organizations who currently use CMM to migrate to CMMI. The continuous representation is developed to bridge CMMI-SW to ISO/IEC TR 15504. This latter approach is critical because it highlights efforts of two major organizations to get closer in their efforts to improve software processes (Kitson, 1999). CMMI is based on moving organizations from one level to a next one and gradually improving software processes. However, an article indicated that ISO may shift emphasis from certification to continuous improvement (Special Report.., 2002). This relationship between CMMI and ISO is critical, because if both move to continuous improvement with less emphasis on certification and levels, organizations will be more encouraged to try newer methods such as AMs. (Since this article deals only with CMMI-SW, CMMI and CMMI-SW are used interchangeably)

Figure 2, as adapted from CMMI-SW continuous model and changed according to the theme of this article, demonstrates model components and their interactions (Capability Maturity …, 2002). The broken lines in Figure 2 represent 4 types of interaction among model components, of which the first three are described in the CMMI-SW document, whereas the last one is proposed by this article. These interactions are explained as follow:

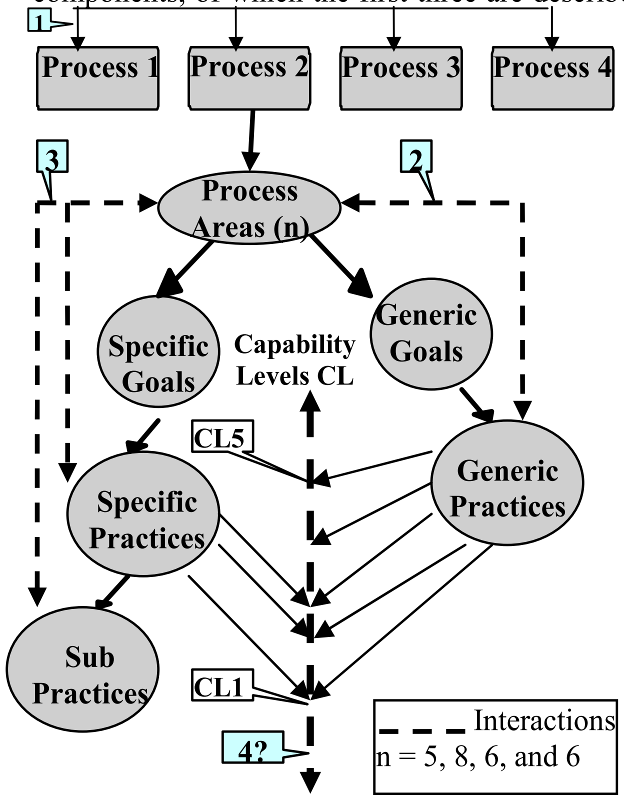


Figure 2. CMMI model components and their interactions (numbered 1, 2, 3, and 4?)

1 This is the intra-process areas interactions, which computationally permits a maximum of 625 interactions (25X25). Some of these interactions, however, may not be feasible or applicable.

2 This interaction is the intersection between generic practices belonging to the five generic goals, which actually represent the five capability levels, and the 25 process areas, allowing for 400 interactions (17X25). All these interactions are explained in the CMMI-SW continuous representation model.

3 This interaction relates the list of specific practices, belonging to each specific goal, of one or more of the five generic goals (or capability levels), permitting a huge number of interactions. This is the case because practices vary from one process area to another, practices vary from one specific goal to another, and sub practices vary from one specific practice to another.

4? This interaction is proposed by this article and will be discussed later in the section dealing with strategies for the link between CMMI and AM, where vertical interaction among capability levels is proposed.

Top

Following this introduction, subsequent sections present discussions on issues raise by the debate over the link between AM and CMMI-SW, and linking strategies.

SOME ISSUES RAISED BY THE CURRENT DEBATE

After comparing agile methods, an article highlighted the need to discuss the issues raised by AM rather than develop new methods, in order to identify common characteristics of AM and their applications (Abrahamsson, P., et. al., 2002). This approach is deemed appropriate to the theme of this paper as well. The following sections present some of these issues and their implications on the link between CMMI and AM, and in the process raise some disputed issues that necessitate further research.

To Document of not to document

The issue of documentation is not new. For example, "Nobody reads documentation" was the title of an article appeared in 1991 (Rettig). Then an extreme (AM style) question will be: do we really need any documentation at all during software development and use? To assist in the analysis, we could distinguish between two different types of documentation: embedded in SDLC activities, and configuration management (CM). For the purpose of our discussion, these two types are referenced as DT1 and DT2, respectively.

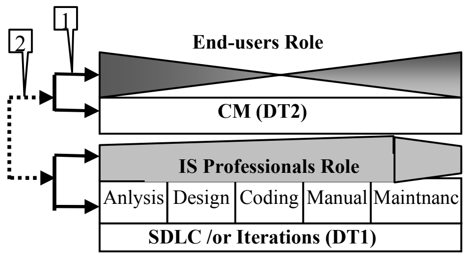


Figure 4. Relationship of documentation types to user and IS professional roles in documentation

In the 70s and 80s, DT1 was the most critical one since developing software without some form of blue prints or roadmap for requirements was considered a recipe for failure. During the 90s, however, IS developers started to deal with different types of IS, where requirements could not be captured at the front-end or they are changing dynamically over time. Here where agile method started to surface justifiably, with shifting emphasis from documentation to strong involvement of end-users. CMMI, on the other hand, continued to stress heavy documentation of both types. Figure 4 shows further classification of DT1 corresponding to SDLC phases, and the position of documentation types in relationship to IS professionals and end-users roles in documentation generation and use. The Figure also implies that end-user role would be heavier and diminishing during the front-end activities (until coding) and lighter and increasing during back-end activities (subsequent to coding). IS role, on the other hand, will be increasing steadily over the SDLC activities, and diminishing lightly after delivery of the system and the start of long term operation and maintenance. The continuous lines (labeled 1) denote the current emphasis of documentation links, where users mainly are associated with DT2. The dotted lines (labeled 2) indicate the desired new link between users and DT1. The desired link implies that SDLC documentation should be understood by users, though to lesser degree than IS counterparts.

To avoid documentation bloat, which AMs are cautioning from, Software Engineers (SEs) could follow the principles of just-in-time knowledge delivery (Cole, et. al. 1997) and documentation minimalism (Orr, 2003), where users and IS professionals can access relevant system information without spending too much time sifting through hundreds of pages or many screens and clicks. Furthermore, the above classification should assist SEs in targeting different types of documentation for reduction, improvement, or elimination, depending on several aspects of the project. To give the example of enterprise resource planning (ERP) implementation projects, detailed database and user interface (UI) designs are not performed since these are dictated by the ERP system.

To involve or not to involve users

The preceding discussions bring in the issue of user involvement of all levels in SDLC. Lack of user involvement and lack of executive support are ranked third and fifth among project termination sources (Boehm, 2000a). SEs should arrive at an adequate level of user involvement throughout the SDLC, as well as assign corresponding appropriate roles for participating users. CMMI stresses this point in the statement "identify stakeholders and check with top management." (Capability Maturity …, 2002). AMs require that knowledgeable end-users actually be part of the development team. The roles may be performed dynamically depending on the stage of software development, as Figure 5 demonstrates.

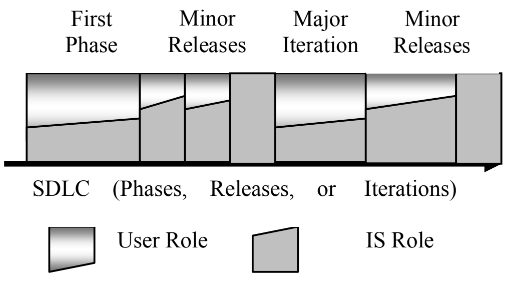


Figure 5. Dynamic User and IS Roles in Software Development

To communicate or not to communicate

The communication issue stems from the fact that a communicate gap exists between users and IS professionals, since user's real world and IS computer world are represented differently. Bridging the gap requires end-users to be computer and IS literate and IS professionals to be business literate. For example, among the top 10 topics with the greatest knowledge gap- where importance most exceeds current knowledge, numbers 1, 3, and 5 are purely management-related topics (Lethbridge, 2000). The top 10 topics include other semi-management topics as well. On the other side, many researches are conducted on user participation, but whether such participation leads to effective communication needs further research. AMs stress continuous involvement of end-users in SDLC, while CMMI recommends planned user involvement. The importance of communication during SDLC and IS use is based on a simple premise: if two people communicate using the same normal language may develop misunderstandings, how about many people communicating through computer languages, and charts such as data flow diagrams, entity-relationship diagrams, screens, class hierarchies, and state charts. An adequate level of IS-user communication is definitely needed to correctly model and satisfy user requirements. Finally, both intra-team and inter-team communication should also be maintained. The following section presents a simple framework to illustrate the relationship among the three previous issues.

Relationship among documentation, user involvement, and communication

Figure 6 illustrates the relationship between the there issues. Current practices of the relationship among them are as follow:

- AMs use strong user involvement as a mean for communication.

- CMMI uses heavy documentation as a mean for communication.

Top

What actually is needed is shown in the middle of chart by the block arrows: a link between user involvement and documentation, where users should understand and/or actually involved in developing documentation in cooperation with IS professionals. We could start doing that in some form in DT2 type of documentation and move gradually into DT1 documentation as well. A candidate DT2 documentation may be the creation of the knowledge management base for lesson learned. DT1 candidate may be process/workflow description using charting tools. To rectify this approach, users may need to be trained on these concepts and methodologies. The equilibrium point is in the middle of the circle (the happy face), where a balance among the three issues is achieved.

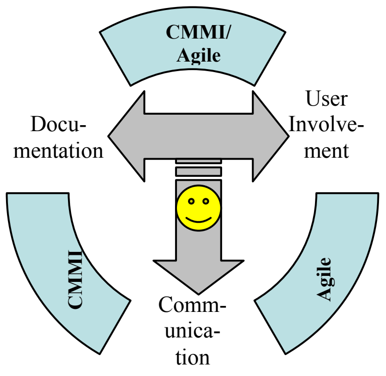


Figure 6. Relationships among the three issues

Application type

The literature identifies emerging set of application types: commercial of-the-shelve software (COTS), I Will Know When I see It (IWKWISI), and rapid change that dictate different requirement modeling techniques (Boehm, 2000). While this classification is appropriate, this article presents a different perspective. During the 70s and 80s, most developed applications were transaction processing systems, where requirements are very much standard across organizations, such as accounting, marketing, production, and human resources systems. These are referred to as horizontal systems. Once systems moved to satisfy requirements of higher management levels, two new types of systems emerged: middle-management systems, such as decision support, expert (DSS), data warehousing (DW), data mining (DM), multidimensional databases (MDB), and knowledge management (KM); and top-management systems, such as executive support systems (EIS), strategic IS (SIS), business intelligence (BI), and competitive intelligent systems (CIS). In this paper, these systems are referred to as vertical systems, and their requirements are not as standardized as horizontal systems.

At the same time, these vertical systems spread throughout the organization horizontally in their respective levels, such as developing DW applications for different functional areas in the business. In addition, browser-based internet applications also evolved, such as e-commerce, collaborative commerce, supply chain management (SCM), mobile commerce, and web services (WS). One major characteristic of these latter types of systems is concerned with the user base: it is interminable, dynamically changing over time, and resides outside the organization’s boundaries.

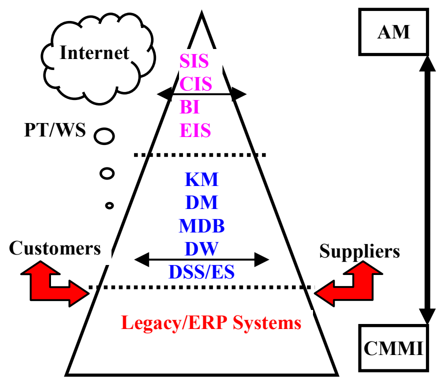


Figure 7. Types of information systems in relationship to management levels, external parties, and CMMI/AM applicability

To map application type to development approaches, we may need a new maturity model, a modified Nolan model perhaps, that shows where an organization currently staged: developing integrated enterprise systems to support its basic IS architecture, re-engineering existing legacy systems or integrating them through enterprise integration architecture, moving upwards through management levels, or moving outwards to multi-platform multi-party applications integration, such as B-to-C, B-to-B, SCM, portals (PT), and WS. It seems natural that the first type of applications, CMMI approach is more suitable than AM, while the reverse holds true for the latter ones. These applications are shown in Figure 7 in relationships to the levels of management. As IS moves vertically and outwardly, the risks and consequences resulting from unsatisfied users, both internal and external, increase. One study reported successful adaptation of agile method in two internet-based application development cases (Ambler, 2002). Finally, Boehm (2002) projected that AM perhaps will be used in medical and air traffic control systems. Whether these systems are low or high risk should be also the subject of further analysis.

STRATEGIES FOR THE LINK

All current research on the link between CMM and AM list CMM areas and map AM principles to them Paulk (2001), for example, relates Extreme Programming (XP) principles to each of the CMM maturity levels requirements. The study concluded that CMM and XP are in many respects complementary. Another research analyzed the relationship between compatibility of agile principles capabilities and process areas and generic goals of CMMI-SW. The article concluded that there is an acceptable level of compatibility, even though not high, and predicted that new methods will evolve in the future (Turner, 2002). The study, however, failed to analyze the mapping of AM to the interaction between process areas and generic goals. It is obvious that process areas (PA) related to "engineering" has the most links to AM principles. Not surprisingly, the same process is the only one with specific and sub- practices associated with CL2 and CL3. In reality advocating the approach "you can have the cake and eat it," current researches on such link escape the difficult issues that will lead to join of disjoin CMMI and AM. Orr (2003) is about the only article that puts these issues up front. The dialogue between Demarco and Boehm (Demarco, 2002) discussed some of these issues, particularity the fact that CMMI mentions the agility process late in the CL5. In the previous section, this article references CMMI and raised issues without going in detail into the mechanics of AM and CMMI. Based on these discussions, this section presents three recommended strategies for the link, as demonstrated in Figure 8.

Separatism strategy

If each party insists on strictly adhering to its principles without any compromise on middle grounds, then this strategy is appropriate and straight forward.

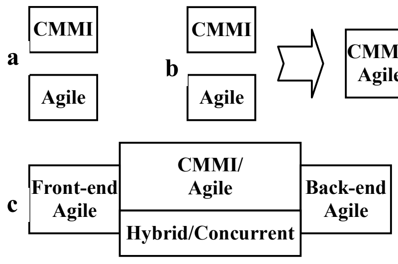


Figure 8. Agile - CMMI linking strategies
b. Separatism Strategy
c. Re-engineering Strategy
d. Integration strategies

Use of CMMI is deemed appropriate in the following cases:

-Department of Defense contractors

-External strategic outsourcing of large projects

-In-house large projects for low level transaction processing system, such as implementing an integrated ERP solution.

AMs methods, on the other hand, are suitable for:

-In-house development for small projects and e-business applications

-Development of management support systems

Integration strategy

In the next level, agile methodologies may be used in support of CMMI processes.

Front-end: This approach suggests the use of AM before implementing the system with CMMI approach. AMs are formally used to identify risk and contract boundaries. This strategy is implied in Boehm's article (2002). Agile team(s) will visit the site for conducting initial interviews and overall system requirements. Once an initial assessment is completed to top-management satisfaction, the CMMI teams finalize the contract and prepare it for signing, and then start executing the contract.

Hybrid/concurrent: Most of the research is on the link between extreme programming, as the first agile method and the most common one, and CMM. One study mapped CMM process areas to XP practices and concluded that XP and CMM in many respect complimentary (Paulk, 2001). One major source of problems when applying this hybrid approach (i.e. embedding AM in CMMI) is human resource assignment to different modules/components in the same project. Agile Methods require different experiences and personalities than CMMI-based methods. This may create a software development human divide similar to the digital divide between agilests and non-agilests. With the current state of the link between AM and CMMI, maintaining organization effectiveness may necessitate the use of concurrent rather than hybrid approach until IS professionals bridge the big gap between the two. The concurrent approach identifies certain modules of a project or complete projects, and then construct appropriate teams accordingly that will employ either AMs or CMMI.

Back-end: Once a system is developed using CMMI approach, AM could be employed to carry on maintenance activities (Poole, 2001). Rajlich (2000) proposes the use of versioning model to control software development by employing a planed major releases and minor updates, similar to system software releases such as MS Windows. IS professional may use

Top

AM for minor releases and CMMI for major releases. Also it may be applied to software patches and minor updates, such as those associated with ERP solutions. Another related area is the study of maintenance activities in ISD using AMs. For example, if a new feature is introduced at iteration number 10, what would the rippling effect on previous nine iterations, if any? The rippling pattern will be even harder to trace if some iterations were conducted concurrently.

Re-engineering strategy

The article by Orr (2003) is the only one who addressed the issue of reengineering AM and CMM (or CMMI for the purpose of this paper). This approach is a long-range plan to effectuate major changes in both AM and CMMI, but perhaps more so in the latter one. It requires a tremendous effort and time, and in both human and cost. For the advancement of software development in the long run, however, it should be done.

Re-engineering CMMI: Orr suggested the following areas for reengineering CMM. Each area is followed by discussions as related to themes of this paper.

- CMM must become less document-centered. As mentioned earlier, balancing the tri-issue factor is very critical for successful completions of software projects.

- CMM must emphasize collaboration. One way of accomplishing this task is through team and software component integration throughout SDLC activities.

- CMM must stress speed as well as quality. Determining factors for balancing speed and quality could include risk, application type, and sourcing decision.

In addition to these three issues, CMMI should distinguish between the guidelines for product manufacturing and software generation. The title of the manual includes the following statement "*Improving processes for better products.*" Also, the manual states that "product or services" may apply to software as well as hardware products [FM102.HDA104.HDB102.T103]. Furthermore, when common terminologies are defined the word "product" is defined but not "software":

The manual was made general so it can be applied to different product types. In the process, the emphasis on, and particulars of, software as the product lost concentration. With software development being human-intensive, project management techniques are different as well as the body of knowledge for software developers. The last issue applies mainly to CL 4 and 5, where quantitative measurements and continuous improvement are discussed. It is a commonly accepted principle that physical process are easier to measure using statistical control packages, and corrective actions for improvement are easier to implement, such as modifying machine set-ups. SEs need to differentiate between software systems that manage and control physical product processes and software systems that are used as performance measurement for software product processes. More research is needed in the area of software quality and metrics in relationship to the scale of measurements (ratio, ordinal, etc.), and the statistical reliability of the generated statistics, when used as the basis for software process continuous improvement.

Re-engineering agile methods: Orr suggests that attention must be paid to some basic management issues:

Top

- Plan at the beginning to build a system's architecture that can be used to break the whole of large projects into a series of small iterations that can be integrated at the end or in stages to form a larger whole. There is no pre-determined formula that will apply to all projects and to all organizations. Regardless of the development methodology employed, large projects should be divided into smaller modules with time table presented graphically using tools such as PERT. If this could not be accomplished, software project may experience budget and time overruns, as well as unsatisfied users.

- Give up its revulsion against the use of tools. For example, some tools are introducing upgrades that will facilitate and speed-up agile development, such as Borland development tools (Krill, 2002).

- Generate more design artifacts and documentation beyond code. As mentioned in the discussion on the tri-issues factor: each organization must arrive at a balance between the three issues.

A framework for CMMI continuous improvement (CI) towards agility

This section is based on the less drastic continuous improvement approach rather than the reengineering approach. The word "agile" and "innovation" are mentioned in CMMI under GP5.1. [GP125.N101]. Does this imply that after building an elephant, start trimming it to a horse? Why organizations have to wait until level 5 to capitalize on the agility process? Can the whole organization become agile? Then, CMMI states at the beginning of the manual that achieving CL 4 and 5 would be the exception rather than the norm. [FM121.HDA104.T105]

It seems that ISD may never construct "empowered workforce" or reap "a cycle of continual improvement" at Capability Level 4 and 5. Will software development ever becomes "… stable for an extended period of time" for IT infrastructure and IS architecture that are characterized by dynamic and continual change? Later the manual links between GL3, 4 and 5 with five advance process areas. This link coupled with the previous statements practically cancels most of the intersections between the advanced process areas and CL4 and 5 generic goals. Based on the discussions thus far, and aside from reengineering it, CMMI-SW future may take one of three alternatives:

- Cancel CL4 and 5 and related process areas, which leaves us with 13 generic goals and about 20 process areas bringing the total number of interactions down to 260 from the current 400. The model becomes more manageable and easier to implement and follow-up.

- Maintain status quo and let organizations chase the *ignis fatuus;* achieving CL4 and 5.

- Keep CL4 and 5 and allow the agility process to be instituted at CL1. This approach is based on the basic philosophy behind the continuous model of CMMI. Rather than using a strict level-based practices, this article suggest the use of gradual phase-in of required practices of higher levels at lower levels and gradual phase-out of undesired lower level practices when moving to higher levels. This will apply mainly to agile processing, quantitative measures, and continuous process improvements associated with advanced process areas. For example, it is impractical to expect that organizations will become suddenly proficient in using quantitative measures in managing the IS function at CL4. They need to use some form of measurements and software metrics and establish data collection mechanisms. The same will apply to continuous improvement (CI) goals, where organizations need to start such practices at lower CLs but perfecting the practice, both its depth and breadth, as it moves to a higher level. The objective is to move organizations to agility and CI in vertical chunks rather than targeting the whole organization horizontally. Figure 10 presents pictorially this scenario that allows vertical interaction between capability levels, as also initially introduced in Figure 2 - interaction "4?" The enclosed irregular shape in the middle of the CLs hierarchy represents an organization that applied limited agile methods and CI and certified at CL2. The objective of CMMI CI strategy is not to transfer the whole organization to agility and continuous improvements; it is to encourage organizations to perpetually rethink their software development strategies.

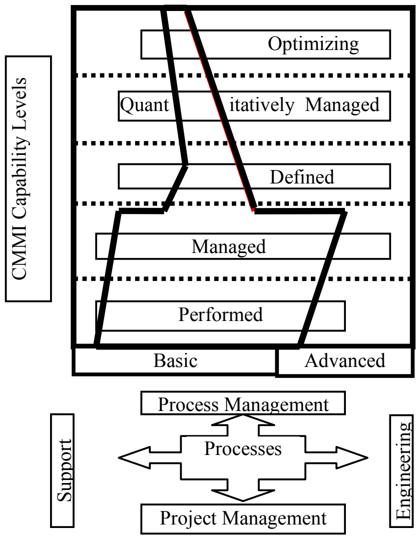


Figure 10. CMMI Capability levels interactions
for continuous process improvement

CONCLUSIONS

This article presented suggestions and guidelines for establishing the link between CMMI and agile methods. Before the two become compatible, several research issues still need to be addressed and resolved. CMMI should address the issue of agility at all levels, rather than have organizations incorporate limited use of AM in CL2. Currently, separatism, front-end, back-end, and concurrent linking strategies are recommended. A hybrid strategy for the link could be delayed unit some form of reengineering of both methods is achieved. The Link between the two sides – CMMI and AM- should address two areas: the use of CMMI and AM in large projects and contracted projects. Meanwhile, organization could employ suggested mechanisms to have both methods more adoptable and innovative. This in the long run may stimulate efforts to reengineer these methods. Furthermore, it is hoped that this article highlighted critical issues and proposed recommendations that will promote further research on this subject. This research should be issue-focused, particularly human factors issue, and the applicability of methods, rather than the detailed mechanism of these methods. For researches who advocated the embedded approach, more detail analysis on how agile methods could be used in CMMI, particularly the intersection between the "Engineering" process area and the Generic Goal "Managed."

Finally, research such as this paper should encourage IT organization the Gulf and the in the Middle East in general to seek certification and to establish local training and certification centers that provide certification to IT department.

REFERENCES

Ambler, S. (2002). Lessons Learned in Agility From Internet-Based Development. IEEE Software, 19:2, pp. 66-73.

Beck, Kent (1999). Embracing Change with Extreme Programming. *Computer*, 32:10, pp. 70-73.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer*, 35:1, pp. 64-69.

Boehm, B. (2000a). Project Termination Doesn't Equal Project Failure. *Computer*, 33:9, pp. 99- 103.

Boehm, B. (2000b). Requirements that Handle IKIWISI, COTS, and Rapid Change. *Computer*, 33:7, pp. 99-103.

Capability Maturity Model® Integration (CMMI^{SM}), Version 1.1 CMMI^{SM} for Software Engineering (2002). (CMMI-SW, V1.1) Continuous Representation CMU/SEI-2002-TR-028 ESC-TR-2002-028 CMMI August.

Cole, K., Fischer, O., and Saltzman, P. (1997). Just-in-time Knowledge Delivery. *Communication of the ACM*, 40:7, pp. 49-53.

DeMarco, T., and Boehm, B. (2002). The Agile Methods Fray. *Computer*, 35:6, pp 90-92.

Kitson, D. (1999) "The Impact of ISO/IEC TR 15504 on the Capability Maturity Model® (CMM®) Integration Effort," Vol. 1, Iss. 1 PILOT, August 1999, http://www.spiceworld.com.

Krill, P. (2002). Borland Eyes Agile Development with Upgraded Tools. Inforworld.com. November 4, 6:06 am PT
Lethbridge, T. (2000). What Knowledge Is Important to a Software Professional? *Computer*, 33:5, pp. 44-50.

Orr, K. (2003). CMM Versus Agile Development: Religious Wars and Software Development. 3:7, http:// www.cutter.com/project/index.html.

Paulk, M. (2001). Extreme Programming form CMM Perspective. *Paper for XP Universe*, and later published in *IEEE Software*, November-December, 2001, pp. 19-26.

Poole, C. and Hulsman, J. W. (2001). Using Extreme Programming in a Maintenance Environment. *IEEE Software*, 18:6, November-December, pp. 56-61.

Rajlich, V. and Bennett, K. (2000). A Staged Model for the Software Life Cycle. *Computer*, 33:7, pp. 66-71.

Special Report: Continual Improvement and the Consistent Pair. (2002). *ISO Management Systems*, November-December, pp. 27-38.

Rettig, M. (1991). Nobody Reads Documentation. *Communications of the ACM.*, 34:7, pp. 19-24.

Top

Turner, R., and Jain, A. (2002). Agile meets CMMI: Culture clash or common cause? Lecture Notes on Computer Science, Electronic Edition, pp. 153-165. available through www.informatik.com.

Top