บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

ในการพัฒนากระบวนการพัฒนาซอฟต์แวร์เพื่อเตรียมความพร้อมองค์กรเข้าสู่มาตรฐาน CMM Level 2 ผู้ศึกษาได้ค้นคว้าเอกสารและงานวิจัยที่เกี่ยวข้องโดยมีรายละเอียดตามหัวข้อที่กำหนดตามลำดับดังนี้

2.1 CMM (Capability Maturity Model)

2.1.1 SEI (Software Engineering Institute)

สถาบันวิศวกรรมซอฟต์แวร์ หรือ SEI (Software Engineering Institute) ได้ถือกำเนิดขึ้น โดยทำการศึกษาเกี่ยวกับขั้นตอนการพัฒนาซอฟต์แวร์ให้เป็นมาตรฐานที่ใช้ทั่วโลก ซึ่งจะมีขั้นตอนที่เกี่ยวกับการประเมินและการจัดขั้นตอนการพัฒนาซอฟต์แวร์ ในเวลาต่อมาสถาบันวิศวกรรมซอฟต์แวร์ได้สร้างมาตรฐานของการพัฒนาซอฟต์แวร์ออกมา ซึ่งเรียกว่า CMM (Capability Maturity Models)

สถาบันวิศวกรรมซอฟต์แวร์ก่อตั้งขึ้นในปี ค.ศ. 1984 โดยมีเจ้าหน้าที่และผู้เชี่ยวชาญจากมหาวิทยาลัย จากภาครัฐ และ ภาคอุตสาหกรรม เข้ามาทำงานร่วมกัน โดยมีภารกิจสำคัญที่จะเป็นผู้นำในการผลักดันระดับความสามารถทางด้านวิศวกรรมซอฟต์แวร์ให้ก้าวหน้ามากขึ้น และ เพื่อปรับปรุงระบบต่าง ๆ ที่ต้องอาศัยซอฟต์แวร์เป็นพื้นฐานการทำงาน งานหลักของสถาบันวิศวกรรมซอฟต์แวร์ก็คือการเปลี่ยนการพัฒนาซอฟต์แวร์จากรูปแบบการทำงานที่ใช้คนจำนวนมาก และ ทำงานอย่างไม่มีหลักเกณฑ์ให้กลายเป็นงานที่มีระบบและกฎเกณฑ์อีกทั้งยังสามารถควบคุมและจัดการให้เป็นไปตามที่เหมาะที่ควรได้ ภารกิจของสถาบันวิศวกรรมซอฟต์แวร์มีอยู่สองด้านคือ

1. เพิ่มขีดความสามารถในการจัดการ งานส่วนนี้เน้นในด้านการทำให้องค์การสามารถพยากรณ์ และ ควบคุม คุณภาพ กำหนดเวลา ต้นทุน และ ผลผลิต ในการจัดหา สร้าง หรือปรับปรุงระบบซอฟต์แวร์
2. พัฒนาแนวทางด้านเทคนิควิศวกรรม งานส่วนนี้เน้นในด้านความสามารถของวิศวกรในการวิเคราะห์ พยากรณ์ และ ควบคุม สมบัติบางประการของระบบซอฟต์แวร์ งานส่วนนี้

เกี่ยวกับการกำหนดทางเลือกสำคัญ ๆ ระหว่างการจัดหา สร้าง หรือ ปรับปรุงระบบซอฟต์แวร์

2.1.2 กระบวนการซอฟต์แวร์

คำว่า กระบวนการซอฟต์แวร์ (software process) เป็นคำที่ใช้กันมากสำหรับระบุงานต่าง ๆ ที่เกี่ยวข้องกับกระบวนการพัฒนาซอฟต์แวร์ Pressman กล่าวว่า กระบวนการซอฟต์แวร์นั้นหมายถึงกรอบของงานต่าง ๆ ที่จำเป็นสำหรับการพัฒนาซอฟต์แวร์ที่มีคุณภาพสูง คำว่ากระบวนการซอฟต์แวร์นี้มีความหมายทั้งเหมือนกันและแตกต่างกันจากคำว่าวิศวกรรมซอฟต์แวร์ คำว่ากระบวนการซอฟต์แวร์นั้นหมายถึงแนวทางที่ใช้ระหว่างการดำเนินการวิศวกรรมกับซอฟต์แวร์ แต่คำว่าวิศวกรรมซอฟต์แวร์นั้นรวมเทคโนโลยีต่าง ๆ ที่นำมาใช้ในกระบวนการซอฟต์แวร์ ตลอดจนเครื่องมือต่าง ๆ ทางด้านวิศวกรรมซอฟต์แวร์ด้วย กระบวนการซอฟต์แวร์ของบริษัทและหน่วยงานต่าง ๆ นั้นมีความแตกต่างกันมาก สุดแท้แต่ความสามารถในการบริหารงานซอฟต์แวร์ และของบุคลากรผู้พัฒนาซอฟต์แวร์ บริษัทและหน่วยงานบางแห่งสามารถพัฒนาซอฟต์แวร์ได้ดี แต่บริษัทและหน่วยงานอื่นที่มีสมบัติคล้ายกันในแทบจะทุกด้านกลับไม่สามารถพัฒนาซอฟต์แวร์ได้ดีเท่า ด้วยเหตุนี้จึงเกิดคำถามว่าเหตุใดจึงเป็นเช่นนี้ มีปัจจัยอะไรที่ก่อให้เกิดความแตกต่าง เมื่อเดือนพฤศจิกายน ปี 1986 สถาบันวิศวกรรมซอฟต์แวร์ร่วมกับ Mitre Corporation ได้เริ่มดำเนินการพัฒนาแบบจำลองสำหรับใช้ในการระบุว่าหน่วยงานต่าง ๆ มีความสามารถในการพัฒนาซอฟต์แวร์ถึงระดับไหนบ้าง ในเดือนกันยายนปีต่อมา สถาบันได้ประกาศคำอธิบายสั้น ๆ เกี่ยวกับกรอบงานสำหรับวัดความเจริญก้าวหน้าในด้านการพัฒนาซอฟต์แวร์ออกมาให้ผู้สนใจรับทราบ การวัดนี้ใช้วิธีการสามแบบ แบบแรกเรียกว่าการประเมินกระบวนการซอฟต์แวร์ซึ่งเป็นการใช้ทีมงานซอฟต์แวร์มืออาชีพที่ได้รับการฝึกมาเป็นพิเศษในการกำหนดระดับความก้าวหน้าด้านซอฟต์แวร์ขององค์กร มาศึกษาประเด็นสำคัญต่าง ๆ เกี่ยวกับการพัฒนาซอฟต์แวร์ที่องค์การกำลังประสบอยู่ แบบที่สองคือ การประเมินความสามารถของซอฟต์แวร์ เป็นการใช้ทีมงานซอฟต์แวร์มืออาชีพในการประเมินหาผู้รับจ้างพัฒนาซอฟต์แวร์ที่มีความสามารถมากพอที่จะรับงานดูแลกระบวนการซอฟต์แวร์ที่องค์กรกำลังดำเนินงานอยู่ และ แบบที่สามเป็นการใช้แบบสอบถามที่มีคำตอบสำหรับประเมินค่าห้าระดับในการประเมินความเจริญก้าวหน้าของกระบวนการซอฟต์แวร์หลังจากใช้กรอบงานข้างต้นในการวัดความเจริญก้าวหน้าของกระบวนการซอฟต์แวร์เป็นเวลานานถึงสี่ปี ในที่สุดสถาบันก็สามารถจัดทำแบบจำลองความเจริญก้าวหน้า

ทางด้านความสามารถทางด้านซอฟต์แวร์ขององค์กรออกมาได้ในปี 1991 แบบจำลองนี้เรียกย่อ ๆ ว่า CMM (ย่อมาจาก Capability Maturity Model for Software)

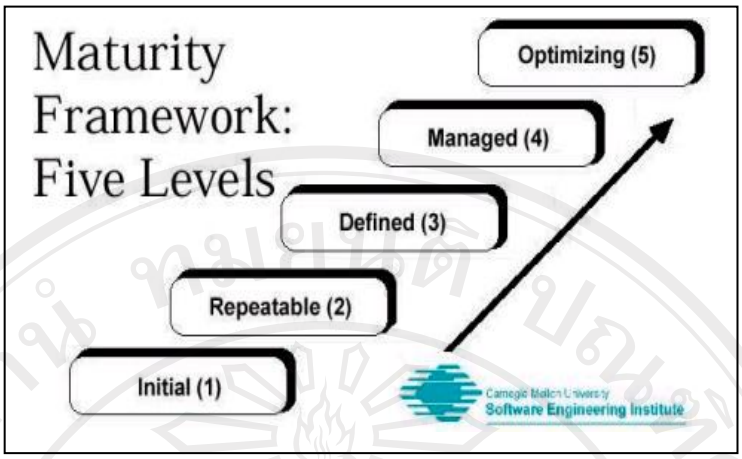
2.1.3 ความหมายของ CMM

CMM คือต้นแบบแบบหนึ่งของการพัฒนากระบวนการซอฟต์แวร์ (Software Process Improvement) ที่ได้รับความนิยมมากทั่วโลก เนื่องจากเป็นแนวทางที่เหมาะสมกับงานพัฒนาซอฟต์แวร์ และยังช่วยเพิ่มความสามารถขององค์กรในการดำเนิน Project ซอฟต์แวร์ ทำให้ซอฟต์แวร์ที่ผลิตได้มีคุณภาพ และยังสามารถสร้างภาพลักษณ์ของบริษัทให้เป็นที่เชื่อถือของลูกค้าได้อีกด้วย

สำหรับบริษัทหรือนักพัฒนาซอฟต์แวร์ไทยที่ต้องการพัฒนาซอฟต์แวร์ให้ได้ระดับมาตรฐานสากลนั้น คงรู้จักกับมาตรฐานใหม่ที่ถูกนำมาใช้ในวงการซอฟต์แวร์ระดับสากล คือ มาตรฐาน CMM (Capability Maturity Model) ถือกำเนิดจาก Software Engineering Institute (SEI) ของมหาวิทยาลัย Carnegie Mellon (http://www.sei.cmu.edu) โดยเป็นการนำข้อดีของมาตรฐาน TQM (Total Quality Management) มาปรับใช้กับเรื่องการพัฒนาซอฟต์แวร์ (Software Development) ตัวมาตรฐาน CMM นี้เรียกได้อีกอย่างหนึ่งว่า SW-CMM (The Capability Maturity Model for Software) ใช้ในการวัดความเชื่อมั่นและคุณภาพของกระบวนการพัฒนาซอฟต์แวร์ขององค์กร และใช้ในการกำหนดวิธีการปฏิบัติงานที่สำคัญในอันที่จะเพิ่มความเชื่อมั่นและคุณภาพต่อกระบวนการเหล่านั้น ซึ่งทางสถาบันวิศวกรรมซอฟต์แวร์ตั้งใจที่จะผลักดันให้มาตรฐาน CMM ได้รับการยอมรับมากขึ้นเทียบเท่ากับ ISO ในปัจจุบันถือว่าเป็น De Facto Standard จากบริษัทและนักพัฒนาซอฟต์แวร์ทั่วโลก

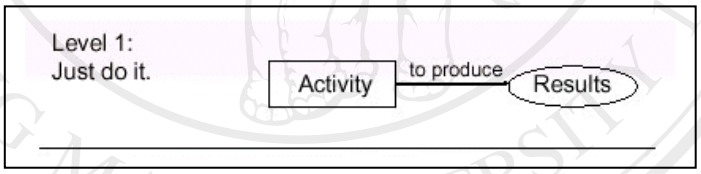
2.1.4 ลักษณะมาตรฐานของ CMM

มาตรฐานCMM จัดเป็นมาตรฐานที่ได้รับความนิยมระดับสากลในเรื่องของซอฟต์แวร์ที่บริษัทพัฒนาซอฟต์แวร์ สามารถนำไปใช้ เพื่อเป็นแนวทางในการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ โดยมาตรฐาน CMM แบ่งระดับความสามารถในการพัฒนาซอฟต์แวร์ของบริษัทผู้พัฒนาซอฟต์แวร์ไว้ 5 ระดับ ดังนี้



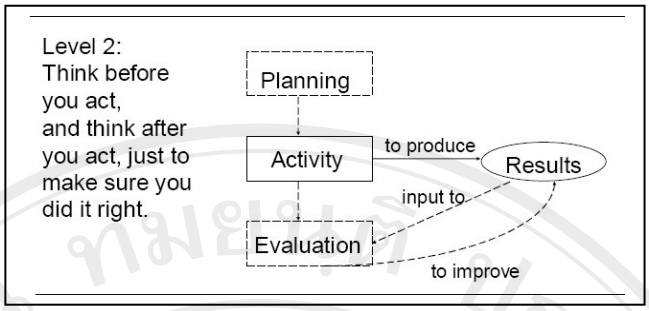
รูป 2.1 ลักษณะของโครงสร้างมาตรฐาน CMM ทั้ง 5 Level

- **CMM Level 1 หรือระดับเริ่มต้น เรียกว่า Initial Level** เป็นการพัฒนาเพียงด้านเดียว เป็นระดับที่บริษัทผู้พัฒนาซอฟต์แวร์ต้องอาศัยความสามารถของบุคลากรเพียงอย่างเดียว ลักษณะการทำงานไม่เป็นทางการมากนัก ยังไม่มีการควบคุมที่ดี ไม่มีการวางแผนงานที่เป็นระบบ จึงไม่สามารถประเมินคุณภาพของผลงานที่ได้ว่าจะมีคุณภาพดีหรือไม่ และซอฟต์แวร์ที่พัฒนาขึ้นส่วนใหญ่ไม่มีการนำไปพัฒนาต่อ



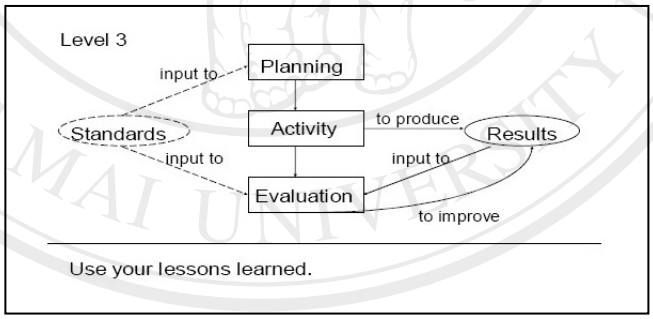
รูป 2.2 ลักษณะมาตรฐานของ Level 1

- **CMM Level-2 หรือระดับจัดทำ Project เบื้องต้น เรียกว่า Repeatable Level** ในระดับนี้มีการนำการบริหารการจัดการ Project เบื้องต้น (Basic Project Management) มาใช้ มีการวางแผนการทำงานอย่างเป็นระบบ มีการจัดทำเอกสารและสามารถตรวจสอบได้ บริษัทผู้พัฒนาซอฟต์แวร์ที่สามารถเข้าสู่ระดับนี้ได้ จะสามารถพัฒนาซอฟต์แวร์ในแต่ละ Project ที่มีลักษณะแบบเดียวกันให้ประสบผลสำเร็จได้เช่นเดียวกับ Project ที่ทำสำเร็จไปแล้ว



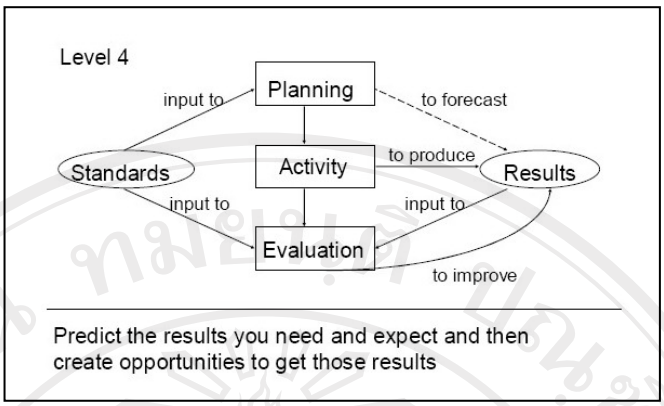
รูป 2.3 ลักษณะมาตรฐานของ Level 2

- **CMM Level-3 หรือระดับที่มีการกำหนดขึ้นอย่างชัดเจน เรียกว่า Defined Level** ในระดับนี้เป็นการพัฒนาเพิ่มขึ้นจาก Repeatable Level การเข้าสู่ระดับบริษัทผู้พัฒนาซอฟต์แวร์จะต้องมีการกำหนดแนวทางในการปฏิบัติงานด้านการจัดทำเอกสารและกำหนดมาตรฐานในการปฏิบัติงาน ทั้งในส่วนของการบริหาร Project และด้านการพัฒนาซอฟต์แวร์ ได้อย่างเหมาะสม โดยมาตรฐานดังกล่าวต้องมีแนวปฏิบัติแบบเดียวกันทั้งองค์กร นั้นคือ องค์กรเริ่มมีระเบียบวิธีการปฏิบัติงานเป็นมาตรฐานของตนเอง



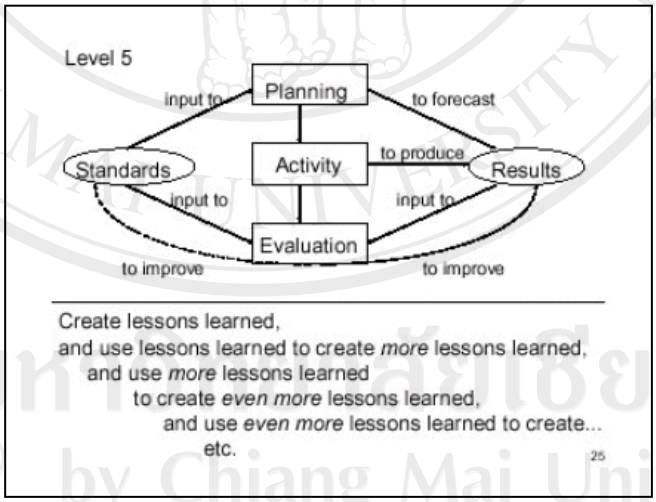
รูป 2.4 ลักษณะมาตรฐานของ Level 3

- **CMM Level-4 ระดับมีการจัดการ เรียกว่า Managed Level** เป็นการพัฒนาเพิ่มขึ้นจาก Defined Level ลักษณะการปฏิบัติในระดับนี้ผู้จัดทำต้องมีการรวบรวมข้อมูลรายละเอียดการปฏิบัติงานต่างๆ ที่เกิดขึ้นไว้ในรูปของสถิติ (Statistical Process Control) เพื่อนำข้อมูลนั้นมาใช้ในการศึกษาวิเคราะห์ผลการทำงาน สามารถวัดผลและควบคุมกระบวนการทางซอฟต์แวร์ได้



รูป 2.5 ลักษณะมาตรฐานของ Level 4

- **CMM Level-5 ระดับปรับปรุงให้เหมาะสมที่สุด เรียกว่า Optimizing Level** เป็นระดับที่ได้นำเอาหลักการจัดการคุณภาพ(Continuous Process Improvement) มาใช้เพื่อป้องกันไม่ให้เกิดข้อบกพร่องในการปฏิบัติงาน และนำไปสู่การพัฒนาอย่างต่อเนื่อง รวมถึงเพื่อให้บริษัทผู้พัฒนาซอฟต์แวร์สามารถปรับเปลี่ยนตัวเองให้สอดคล้องกับการเปลี่ยนแปลงทางด้านเทคโนโลยีได้



รูป 2.6 ลักษณะมาตรฐานของ Level 5

2.1.5 กลุ่มกระบวนการหลัก

สถาบันวิศวกรรมซอฟต์แวร์ได้กำหนดกลุ่มกระบวนการหลัก (Key Process Area หรือ KPA) สำหรับระดับความสามารถแต่ละระดับเอาไว้ด้วย กลุ่มกระบวนการหลักเหล่านี้ใช้อธิบายฟังก์ชันต่างๆ ทางด้านวิศวกรรมซอฟต์แวร์ที่จะต้องมีในแต่ละระดับ การกำหนดกลุ่มกระบวนการหลักแต่ละรายการจะต้องจำแนกตามสมบัติต่อไปนี้

- เป้าหมาย วัตถุประสงค์หลักที่กลุ่มกระบวนการหลักแต่ละรายการจะต้องบรรลุให้ได้
- ข้อตกลง ข้อกำหนดที่ระบุให้หน่วยงานต้องดำเนินงานให้บรรลุเป้าหมาย และเป็นการยืนยันว่าจะพยายามทำตามเป้าหมาย
- ความสามารถ สมบัติที่จำเป็นจะต้องมีทั้งทางด้านองค์กรและในเชิงเทคโนโลยีเพื่อให้หน่วยงานดำเนินงานตามข้อตกลง
- กิจกรรม ภารกิจที่จำต้องทำเพื่อให้บรรลุกลุ่มกระบวนการหลัก
- วิธีการตรวจการดำเนินงาน แนวทางในการตรวจกิจกรรมว่าดำเนินไปเช่นใด
- วิธีการตรวจสอบผลการดำเนินงาน แนวทางในการตรวจสอบการดำเนินงานกลุ่มกระบวนการหลักว่าดำเนินการได้ถูกต้องเหมาะสม

ซึ่งแต่ละระดับมีกลุ่มกระบวนการหลักดังนี้

CMM Level-1

ไม่มีกลุ่มกระบวนการหลัก

CMM Level-2

มี 6 กลุ่มกระบวนการหลัก ดังนี้

1. Requirements Management (RM)
2. Software Project Planning (SPP)
3. Software Project Tracking and Oversight (SPTO)
4. Software Subcontract Management (SSM)
5. Software Quality Assurance (SQA)
6. Software Configuration Management (SCM)

CMM Level-3

มี 7 กลุ่มกระบวนการหลัก ดังนี้

1. Peer reviews

2. Intergroup coordination

3. Software product engineering

4. Integrated software management

5. Training program

6. Organization process definition

7. Organization process focus

CMM Level-4

มี 2 กลุ่มกระบวนการหลัก ดังนี้

1. Software quality management

2. Quantitative process management

CMM Level-5

มี 3 กลุ่มกระบวนการหลัก ดังนี้

1. Process change management

2. Technology change management

3. Defect prevention

2.2 ลักษณะมาตรฐานของ CMM Level 2

CMM Level 2 มีกลุ่มกระบวนการหลักที่จะต้องทำให้เป็นไปตามข้อกำหนดของ CMM อยู่ด้วยกัน 6 เรื่องคือ

1. Requirement Management (RM) เป็นกุญแจที่ตรวจสอบในเรื่องการค้นหาความต้องการในการทำ Project โดยจะดูจากวิธีการหา การยอมรับความต้องการที่เขียนขึ้นของผู้ใช้ เป็นต้น
   - การสร้างความเข้าใจของลูกค้ากับ Project Team Member ให้ตรงกัน
   - การทำเอกสาร (Document)และควบคุม (Control) ความต้องการของลูกค้า
   - ถ้ามีการเปลี่ยนแปลงของลูกค้า Project Team Member ต้องรับทราบ ทำเอกสารใหม่
   - ทำแผนงาน (Plan) ระยะเวลา รายละเอียด Project ให้ตรงกับ Requirement
   - RM จะควบคุมการใช้ Tool, Module, Component ให้ตรงกัน

2. Software Project Planning (SPP) แบบแผนและทิศทางการดำเนินงานของ Project ทั้งหมด ซึ่งจะต้องได้รับการออกแบบอย่างรอบคอบ
   - ต้องมีการประเมิน จำนวนคน เวลา เงิน และความเป็นไปได้
   - Project Team Member ต้องมีการสื่อสารกันให้ได้ข้อสรุปร่วมกัน รวมถึงผู้บริหารด้วย
   - กำหนดรายละเอียดในแผนการทำงาน ช่วงเวลาไหน ทำอะไร
   - การตั้งชื่อแผน (Software Development Plan)
   - แผนต้องระบุ Software Development Life Cycle คือ Project ต่างๆอาจใช้ Software Development Life Cycle ต่างกัน เช่น Waterfall, Spiral เป็นต้น
   - ความเสี่ยงของ Project นั้นมีหรือไม่ ถ้ามีต้องระบุไว้ในแผนด้วย เช่น ลูกค้ากำลังจะโดนย้าย เป็นต้น
   - วางแผน Size, Effort, Cost, Schedule, Resources
     - ถ้ามีข้อมูลที่ใช้ในการวางแผนงานเช่น ข้อมูลอดีตอาจบอกเราได้ว่า Project ลักษณะแบบนี้ใช้คนเท่าไหร่
     - ถ้าเป็น Project ใหม่ๆให้ดูว่าคนอื่นทำกันอย่างไร ใช้เวลาเท่าไหร่
3. Software Project Tracking and Oversight (SPTO) การตรวจสอบและเฝ้าติดตามขั้นตอนการทำงาน จะต้องมีรูปแบบการตรวจสอบที่ชัดเจน และไม่เข้าไปขัดขวางขั้นตอนการทำงานเดิม
   - ตรวจสอบการทำงานจริงที่ทำกับแผนว่าตรงกันไหม
   - สิ่งที่เข้าไปตรวจสอบ
     - Product Size
     - Project Effort
     - Cost
     - Schedule
     - Activities
     - Risk
   - เมื่อมีการเบี่ยงเบนไปจากแผนเดิม ต้องทำการบันทึกลงในเอกสาร Corrective Action แล้วทำการเปลี่ยนวิธีการทำงาน หรือเปลี่ยนแผนให้เหมาะสมกับงาน
4. Software Subcontract Management (SSM) กุญแจที่คอยดูในเรื่องการบริหารส่วนการทำงานที่แยกออกไป เมื่อ Project ใหญ่ขึ้น บางครั้งเราจำเป็นต้องดึงงานบางส่วนที่

สามารถให้คนอื่น หรือหน่วยงานอื่นทำได้ทำแทนเพื่อเร่งหรือรักษาเวลาเสร็จสิ้น Project เอาไว้ ตรงจุดนี้จึงจำเป็นต้องมีกลวิธีในการตรวจสอบและบริหารการทำงานของผู้รับผิดชอบส่วนนอกด้วย

- เลือกบริษัทผู้รับงานรายย่อย โดยการตรวจสอบบริษัทรายย่อย (Subcontract) โดยดูที่ Process Capability เช่น ถ้าบริษัทได้ CMM ก็ควรเลือกบริษัทที่มี CMM เหมือนกัน เคยทำงานด้านไหน มีความรู้ด้านไหน
- บอกรายละเอียดให้บริษัทผู้รับงานรายย่อย มีอะไรบ้างในงาน (Statements of Work)
  - ความต้องการของบริษัท
  - มาตรฐานที่ต้องทำตามบริษัท
  - ขั้นตอนการดำเนินงาน
  - ผลิตภัณฑ์ที่ต้องทำการส่งมอบ
- การตรวจการทำงานของบริษัทผู้รับงานรายย่อย เช่น ใช้สีอะไร ไฟล์เป็นระเบียบหรือไม่ โดยมีวิธี ดังนี้
  - การสัมภาษณ์
  - การทำงานของ SQA บริษัทรายย่อย
  - การทำงานของ SCM บริษัทรายย่อย เช่น การเก็บ การควบคุมเอกสารต่างๆ

5. Software Quality Assurance (SQA) กุญแจที่จะคอยรับประกันในเรื่องคุณภาพของซอฟต์แวร์ว่าถูกต้องตรงกับความต้องการของผู้ใช้หรือไม่
   - คอยตรวจสอบว่าผิดหรือถูกขั้นตอนอะไรบ้าง
   - การทำงานและซอฟต์แวร์ตรงตามที่กำหนดไว้หรือเปล่า
   - ทำรายงาน SQA ส่งตรงต่อผู้บริหาร
   - SQA เป็นหน่วยงานอิสระ แยกในผังองค์กร
   - SQA ต้องเริ่มตั้งแต่แรก และต้องเข้าใจแผน มาตรฐานขององค์กร
6. Software Configuration Management (SCM) กุญแจที่คอยตรวจสอบเรื่องการทำงานด้านเทคนิคในส่วนของการระบุค่าข้อมูลต่างๆ
   - มีระบบจัดเก็บโปรแกรมและเอกสาร
   - ดูแลการเปลี่ยนแปลงอย่างเป็นระบบ
   - Baseline หรือพื้นฐานของงานที่ทำอยู่ ซอฟต์แวร์เวอร์ชันล่าสุด
   - ตกลงว่า Baseline ควรเก็บอะไรบ้าง เช่น Source Code
   - Baseline สามารถเปลี่ยนได้ ถ้า Requirement เปลี่ยน

- SCM ดู Version Control ตลอด Life Cycle
- การเปลี่ยนแปลงแต่ละ Version ต้องเขียนลงในเอกสาร
- การสร้าง Directory ต้องตกลงกันให้เป็นมาตรฐานเดียวกันหมดทั้งบริษัท เช่น ไฟล์ที่เก็บ Project ใช้ชื่อตามรหัส Project
- มีการจัดทำที่เก็บเอกสารทั้งหมดของบริษัท

2.3 ประโยชน์ของ CMM

1. การทำงานเป็นระบบมากขึ้น ทุกขั้นตอนต้องการจดบันทึกรายละเอียดระหว่างการทำงานไว้เป็นเอกสาร หรือมีหลักฐานการทำงานที่ตรวจสอบได้โดยง่าย เช่น การบันทึกการเจรจากับลูกค้า
2. เมื่อการทำงานเป็นระบบโอกาสที่จะประสบผลสำเร็จในการทำงานก็มากขึ้น เป็นการสร้างชื่อเสียงให้หน่วยงานได้ และสร้างโอกาสในการรับงานจากลูกค้าเพิ่มขึ้นด้วย
3. การทำงานของหน่วยงานจะมีวัฒนธรรมการทำงานที่เป็นแบบเดียวกัน มีวิธีการปฏิบัติที่เป็นมาตรฐาน ที่สามารถยืดหยุ่น และปรับตัวให้เข้ากับความเปลี่ยนแปลงได้ตลอดเวลา ผู้บริหารมองเห็นสภาพการปฏิบัติงานของ Project ที่เป็นนามธรรมได้อย่างชัดเจน สามารถแก้ปัญหาที่เกิดขึ้นแล้วได้อย่างมีประสิทธิภาพ และสามารถเตรียมตัวแก้ปัญหาที่อาจจะเกิดในอนาคตได้เป็นอย่างดีด้วย
4. ประโยชน์ต่อประเทศชาติ หากไทยสามารถพัฒนาบริษัทซอฟต์แวร์ไทย ให้มีวุฒิภาวะความสามารถมากขึ้น จะสามารถรับงานจากต่างประเทศ และทำรายได้เข้าประเทศได้อีกมาก

2.4 สรุป

CMM เป็นโมเดลที่ใช้ในการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ ประกอบด้วยระดับความเติบโตห้าระดับ ระดับ 5 Optimizing เป็นระดับสูงสุดของโมเดลนี้ แต่ละระดับจะมีเป้าหมายที่องค์กรต้องบรรลุถึงอยู่จำนวนหนึ่ง เรียกว่ากลุ่มกระบวนการหลัก (KPA) องค์กรมีอิสระในการหาวิธีมาดำเนินการเพื่อให้บรรลุเป้าหมายดังกล่าว องค์กรที่ใช้ CMM มีแนวโน้มที่จะผลิตซอฟต์แวร์ที่มีคุณภาพสูงขึ้น ส่งผลงานได้ตรงต่อเวลา และสามารถคาดการณ์ล่วงหน้าถึงความสำเร็จในการพัฒนา ทำให้สามารถลดค่าใช้จ่าย เวลา และเพิ่มคุณภาพให้กับซอฟต์แวร์ที่พัฒนาขึ้นมา ระบบการปรับปรุงกระบวนการที่ดีจะต้องมีการปรับปรุงระบบดังกล่าวให้พัฒนาขึ้นไป ให้สอดคล้องกับเทคโนโลยี และให้เข้ากับยุคสมัยที่เปลี่ยนแปลงไป