# บทที่ 2

# แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงแนวคิดและทฤษฎี และงานวิจัยที่เกี่ยวข้อง ได้แก่

2.1 ความรู้เบื้องต้นเกี่ยวกับระบบ SAP R/3

2.2 วงจรการพัฒนาระบบ

2.3 Capability Maturity Model Integration (CMMI)

2.4 แผนผังสาเหตุและผล

2.5 การปรับปรุงประสิทธิภาพของโปรแกรม ABAP/4

2.6 การบำรุงรักษาซอฟต์แวร์

## 2.1 ความรู้เบื้องต้นเกี่ยวกับระบบ SAP R/3

### 2.1.1 ระบบ SAP R/3

ระบบ SAP R/3 เป็นระบบที่มีสถาปัตยกรรมแบบรับ-ให้บริการ 3 ชั้น (Three-tier Client/Server) ซึ่งแบ่งการทำงานออกเป็น 3 ส่วน ดังภาพที่ 2.1 (Buk-Emden and Galimow, 1996; Doppelhammer, Hoppler, Kemper & Kossman, 1997; Schneider, 1999; Will, Hienger, Strassenburg, Himmer, 1996)

1.1 ชั้นการแสดงผล (Presentation Layer) เป็นชั้นของการทำงานที่ให้บริการทางด้านการแสดงผลที่หน้าจอด้วยส่วนต่อประสานกราฟิกกับผู้ใช้ (Graphical User Interface: GUI) ซึ่งทำงานอยู่บนเครื่องคอมพิวเตอร์ของผู้ใช้งาน และติดต่อกับส่วนบริการประยุกต์ (Application Server) ผ่านข่ายงานบริเวณเฉพาะที่ (Local Network: LAN) หรือข่ายงานบริเวณกว้าง (Wide-Area Network: WAN)

1.2 ชั้นการประยุกต์ (Application Layer) เป็นชั้นของการทำงานที่ให้บริการด้านตรรกะประยุกต์ (Application Logic) คือ ประมวลผลคำสั่งในการทำงานและคำสั่งในการคำนวณ เช่น IF, CASE, DO, WHILE, คำสั่งบวก ลบ คูณ หาร เป็นต้น และด้านตรรกะข้อมูล

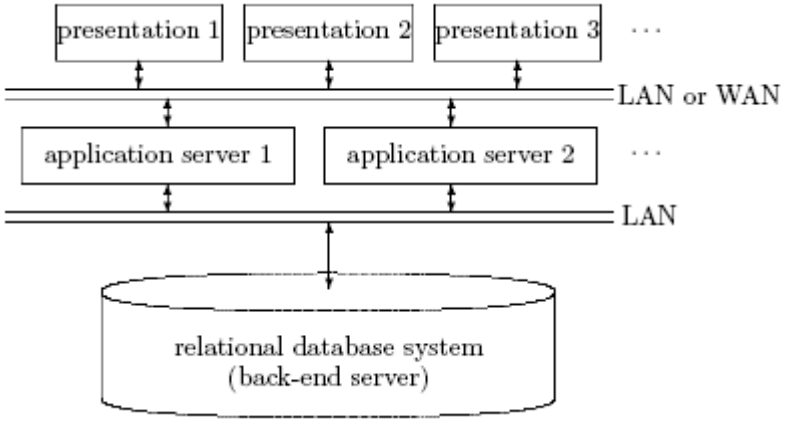
(Data Logic) คือ ประมวลผลคำสั่งภาษาสอบถามเชิงโครงสร้าง (Structured Query Language: SQL) โดยชั้นการทำงานนี้ติดต่อกับชั้นฐานข้อมูลผ่านข่ายงานบริเวณเฉพาะที่

1.3 ชั้นฐานข้อมูล (Database Layer) เป็นส่วนที่ให้บริการทางด้านการจัดการข้อมูลในระบบฐานข้อมูล ซึ่งทำงานร่วมกับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System: RDBMS)

ภาพที่ 2.1

สถาปัตยกรรมของระบบ SAP R/3

(Three-tier Client/Server)



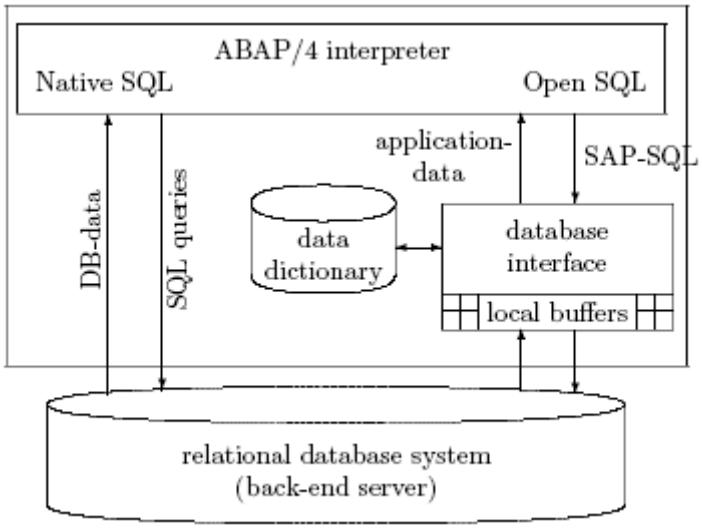

ที่มา: “Performance Tuning for SAP R/3,” by Kemper, Zeller, and Kossmann, 1999, The 25th International Conference on Very Large Databases, p. 3.

### 2.1.2 ภาษา ABAP/4

ภาษา ABAP/4 (Advanced Business Application Programming Language) ที่ใช้ในการพัฒนาโปรแกรมประยุกต์ของระบบ SAP R/3 ภาพที่ 2.2 แสดงให้เห็นว่าภาษา ABAP/4 สามารถรองรับการเข้าถึงฐานข้อมูลได้ 2 ทาง คือ Native SQL และ Open SQL การเข้าถึงฐานข้อมูลด้วย Native SQL คือการใช้คำสั่ง SQL ของระบบการจัดการฐานข้อมูลโดยตรง เช่น ถ้าระบบการจัดการฐานข้อมูลเชิงสัมพันธ์เป็นระบบ ORACLE การเข้าถึงฐานข้อมูลจะต้องเขียน

คำสั่งด้วยโครงสร้างของภาษา SQL ตามมาตรฐานของระบบ ORACLE โดยเฉพาะ เป็นต้น ส่วนการเข้าถึงฐานข้อมูลด้วย Open SQL จะเขียนคำสั่ง SQL ตามมาตรฐานของระบบ SAP R/3

ภาพที่ 2.2
การเข้าถึงฐานข้อมูลของภาษา ABAP/4




ที่มา: “Performance Tuning for SAP R/3,” by Kemper, Zeller, and Kossmann, 1999, The 25th International Conference on Very Large Databases, p. 3.

ในระบบ SAP R/3 นั้น การเขียนโปรแกรม ABAP มีรูปแบบการเขียนโปรแกรมหลักๆ อยู่ 4 แบบด้วยกัน คือ

2.1 ABAP Report คือ การเขียนโปรแกรม ABAP เพื่อออกรายงาน

2.2 Dialog Program คือ การเขียนโปรแกรมเพื่อทำงานในลักษณะของรายการเปลี่ยนแปลง (Transaction) หรือ (Screen Processing) เช่น รายการเปลี่ยนแปลงใช้ในการบันทึกรายการบัญชี (GL Document Posting)

2.3 SAPscript หรือ Smart Forms Program คือ การเขียนโปรแกรม ABAP เพื่อแสดงข้อมูล ในลักษณะของฟอร์มเอกสาร เช่น ใบกำกับภาษี หรือ ใบเสร็จรับเงิน

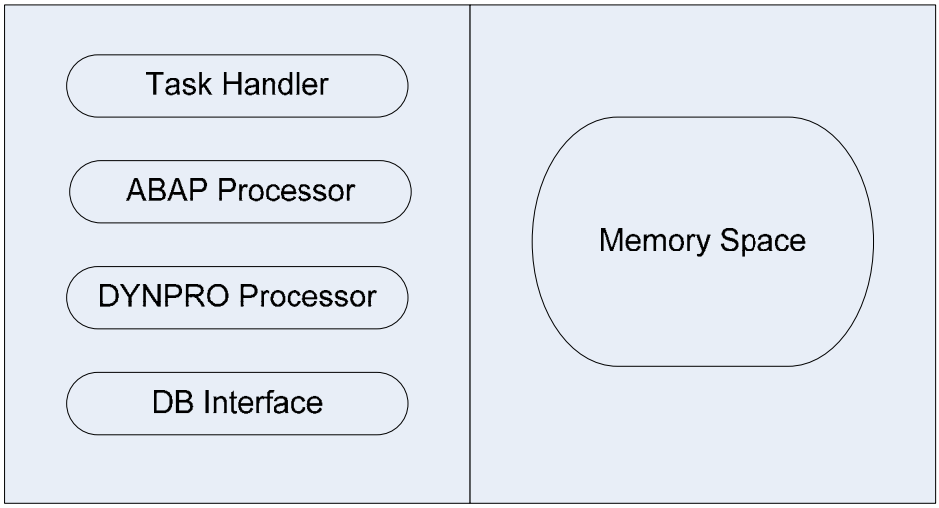
2.4 BDC Program คือ การเขียนโปรแกรม ABAP เพื่อใช้ในการแปลงผันข้อมูล (Data Conversion) หรือโปรแกรมต่อประสาน (Interface Program) โดยมีการนำข้อมูลจากระบบภายนอกเข้าสู่ระบบ SAP R/3

ภาษา ABAP เป็นภาษาที่ใช้ในการพัฒนาระบบ SAP โดยมีเบื้องหลังการทำงานของระบบ SAP ที่เรียกว่า Work Process หรือ Kernel ซึ่งถูกพัฒนาขึ้นมาจากภาษา C++ สำหรับหน้าที่ของ Work Process คือ เป็นตัวประเมินผลโปรแกรม ABAP ที่ถูกเรียกใช้โดยจะทำการประเมินผลที่ระบบประยุกต์

Work Process ประกอบด้วยองค์ประกอบ 4 ส่วน ได้แก่ Task Handler, ABAP Processor, DYNPRO Processor และ DB Interface ดังภาพที่ 2.3

ภาพที่ 2.3

องค์ประกอบของ Work Process




ที่มา: “SAP R/3 ABAP Programming,” โดย ประพจน์ สุขมานนท์, 2547, น. 33

Task Handler เป็นส่วนที่ทำหน้าที่พิจารณาโปรแกรม ABAP ที่ถูกเรียกใช้ว่ามีคำสั่งประเภทใดบ้างในโปรแกรม ABAP โดยพิจารณาทีละคำสั่งตามลำดับ เมื่อพบว่าคำสั่งอยู่ในประเภท ABAP ส่วน Task Handler จะทำการส่งคำสั่งนั้นไปให้กับส่วนของ ABAP Process ทำการประมวลผล และถ้าคำสั่งเป็นส่วนที่เกี่ยวกับ Screen Processing (Dialog Program) ส่วนของ Task Handler จะโอนการทำงานในส่วนนี้ให้กับ DYNPRO (Dynamic Program Processor) ประมวลผล ถ้า Task Handler พบว่าคำสั่งเป็นคำสั่งประเภท Open SQL มันก็จะส่งคำสั่ง Open SQL ไปที่ส่วนบริการฐานข้อมูล (Database Server)

สำหรับส่วนหน่วยความจำเฉพาะที่ (Local Memory) นั้นจะเป็นพื้นที่หน่วยความจำของ Work Process ที่มันจะใช้ในการเก็บข้อมูลที่ได้จากการประมวลผลคำสั่ง ABAP ต่างๆ โดยจะมีส่วนที่เรียกว่า Memory Space ที่จะเป็นพื้นที่เก็บข้อมูลพวกตัวแปร (Data Object) ต่างๆ ของโปรแกรม ABAP ที่มันกำลังทำงานให้อยู่ ซึ่งพื้นที่หน่วยความจำเฉพาะที่นี้จะถูกลบทิ้งเมื่อจบการทำงานของโปรแกรม ABAP (ประพจน์ สุขมานนท์, 2547, น.31-34)

### 2.1.3 เครื่องมือในการตรวจสอบการทำงานของโปรแกรม

เครื่องมือในการตรวจสอบการทำงานของโปรแกรมเป็นเครื่องมือที่ระบบ SAP R/3 ได้มีรองรับไว้อยู่แล้ว ซึ่งเครื่องมือเหล่านี้ก็เป็นโปรแกรมอันหนึ่งที่จะช่วยตรวจสอบการทำงานของโปรแกรมที่เกิดปัญหา

3.1 Performance trace เป็นเครื่องมือที่มีประสิทธิภาพสูงสำหรับการวิเคราะห์เวลาการทำงาน (Run time) ของโปรแกรม ABAP โดยจะแสดงเวลาของการทำงาน 4 หมวด ได้แก่ การทำงานที่เรียกใช้ฐานข้อมูล (Database access หรือ SQL statements) การทำงานที่เรียกใช้ Remote Function Call (RFC call สำหรับระบบ SAP R/3 Version 4.0 ขึ้นไป) การทำงานประเภท Enqueue (Enqueue operation สำหรับระบบ SAP R/3 Version 4.0 ขึ้นไป) การทำงานที่เรียกใช้บัฟเฟอร์ (SAP buffer access สำหรับระบบ SAP R/3 Version 4.5 ขึ้นไป)

การเรียกใช้ Performance trace สามารถเรียกใช้ได้โดยไปที่เมนู System, Resources และ Performance Trace ตามลำดับ หรือเรียกใช้ผ่านรหัสรายการเปลี่ยนแปลง (Transaction code) ST05 เมื่อเข้ามาที่หน้าโปรแกรมดังภาพที่ 2.4 จะมีปุ่มที่ใช้สำหรับ Start, Stop และ Evaluate Performance Trace และมีรายการให้เลือกได้ว่าต้องการวิเคราะห์เวลาการทำงาน (Run time) ของโปรแกรมในแต่ละหมวด ได้แก่ SQL trace, Enqueue trace, RFC trace และ Buffer trace ซึ่งหมวดที่แนะนำให้วิเคราะห์ คือ SQL trace, Enqueue trace และ RFC trace

การเปิดใช้ Performance Trace ทำได้เพียง 1 การทำงานต่อ 1 Application server ณ เวลาหนึ่งเท่านั้น กล่าวคือ เมื่อเปิดใช้ Performance Trace และเลิกใช้แล้ว จะต้องทำการปิดมันเสียก่อน จึงเปิดใช้ในครั้งต่อไปได้อีก เพราะไม่สามารถเปิดใช้พร้อมกันได้ เมื่อกดปุ่ม Trace on จะพบกับหน้าจอที่แสดงว่าจะวิเคราะห์เวลาการทำงานของโปรแกรมภายใต้ชื่อผู้ใช้ (User Name) นั้น ซึ่งปกติจะเป็นตามชื่อผู้ลงบันทึกเปิด (Logon User) ที่เปิดใช้ Performance

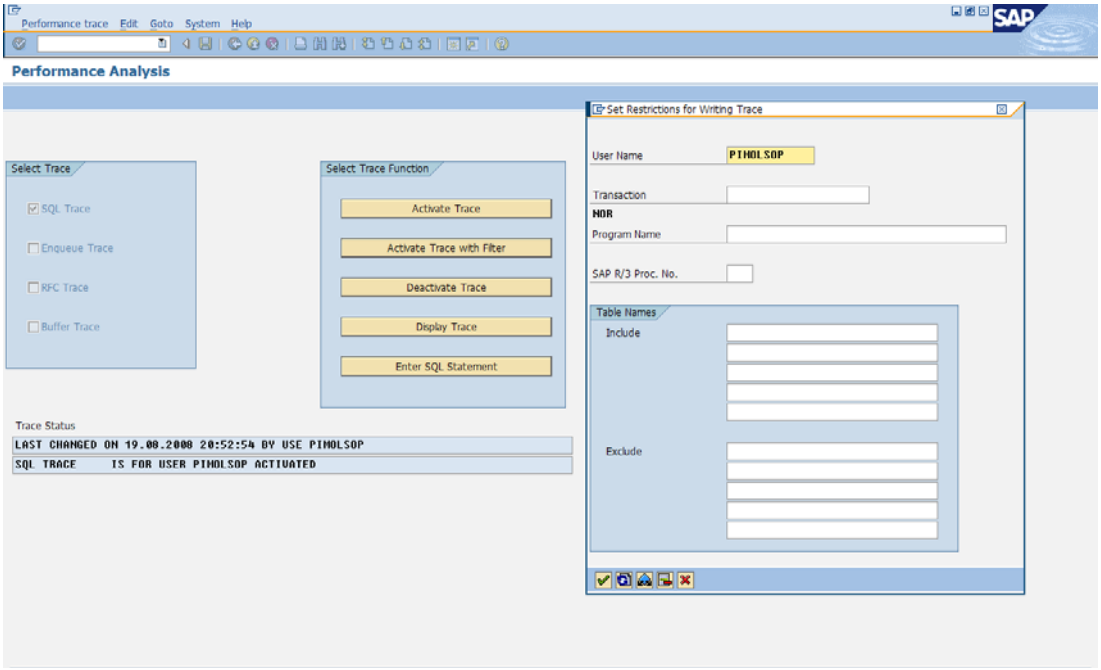
trace นั่นเอง หากต้องการวิเคราะห์เวลาการทำงานของโปรแกรมภายใต้ชื่อผู้ใช้อื่นก็ย่อมสามารถทำได้ตามต้องการ และการแสดงผลของหน้าจอการ Trace แสดงได้ดังภาพที่ 2.5

ข้อเสนอแนะเกี่ยวกับการเปิดใช้ Performance Trace สำหรับชื่อผู้ใช้ที่ต้องการวิเคราะห์การทำงานของโปรแกรมควรมีการทำงานเพียง 1 การทำงานเท่านั้นในขณะที่เปิด Performance Trace และต้องไม่มีการทำงานในโหมดภาวะพื้นหลัง (Background Mode) รวมทั้งการทำงานที่เป็นลักษณะของ Update request ภายใต้ชื่อผู้ใช้นี้ เพื่อให้ได้ข้อมูลที่ต้องการวิเคราะห์ออกมามีความชัดเจน

เงื่อนไขที่สามารถระบุได้สำหรับการ Trace และข้อมูลเวลาที่แสดงออกมาหลังจากการปิด Trace ในแต่ละหมวด มีดังตารางที่ 2.1 ถึงตารางที่ 2.4 (Schneider, 2006, p.163-172)

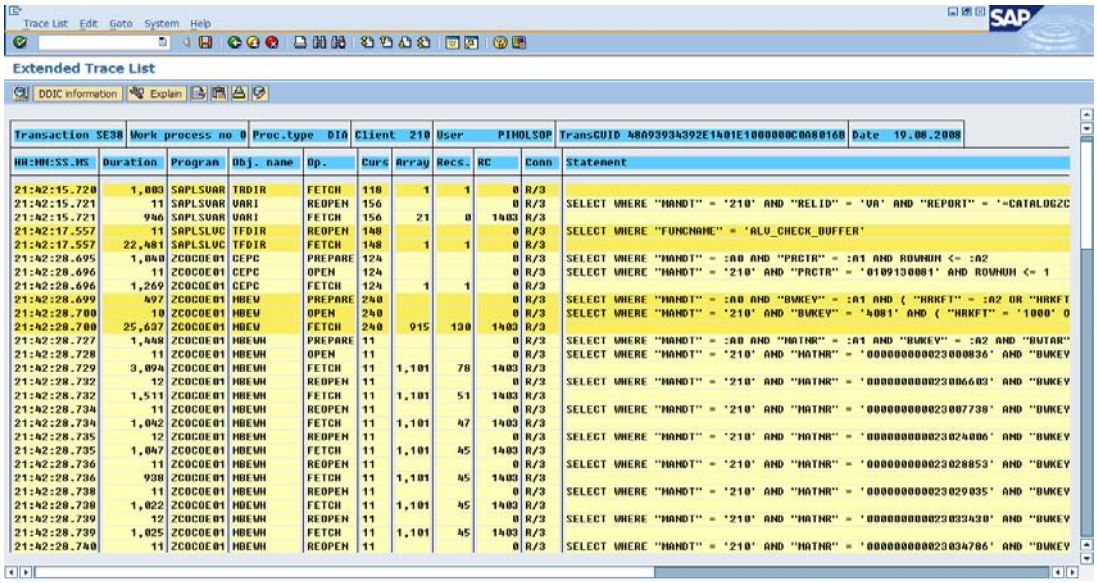
ภาพที่ 2.4

ตัวอย่างจอภาพ Performance Trace



ภาพที่ 2.5

ตัวอย่างจอภาพของผลจาก Performance Trace




ตารางที่ 2.1

เงื่อนไขที่สามารถระบุได้สำหรับการ Trace

| Field | คำอธิบาย |
|---|---|
| Trace Filename | ชื่อของ Trace file |
| Trace Mode | SQL trace, RFC trace และ Enqueue trace<br>โดยปกติหมวด SQL trace จะถูกเลือกไว้ให้อยู่แล้ว ถ้าต้องการ Trace ในหมวดอื่น ให้เลือกเพิ่มเติมได้ |
| Trace Period | ช่วงเวลาที่ทำการ Trace |
| Username | User ที่ต้องการ Trace |
| Object name | ชื่อของตารางที่ต้องการ Trace |
| Duration | ระบุช่วงเวลาที่จะทำการ Trace |
| Operation | การทำงานของฐานข้อมูลที่ต้องการ Trace |

ที่มา: “Performance Optimization Guide,” by Schneider, 2006, p. 165

ตารางที่ 2.2

ข้อมูลของ SQL Trace

| Field | คำอธิบาย |
|---|---|
| Duration | เวลาในการทำงานของคำสั่ง SQL หน่วยเป็น ไมโครวินาที (Microsecond) ถ้าเวลาในการทำงานมากกว่า 150,000 ms รายการนั้นจะเป็นสีแดง หมายถึง ใช้เวลาในการทำงานนานมาก |
| Object | ชื่อของตาราง |
| Oper | การทำงานบนฐานข้อมูล เช่น Prepare หมายถึงเตรียมการส่งคำสั่ง, Open หมายถึง คำสั่งที่ใช้เปิดฐานข้อมูล, Fetch หมายถึง การส่งผ่านข้อมูลมาจากฐานข้อมูล |
| Rec | จำนวนรายการของข้อมูลที่อ่านจากฐานข้อมูล |
| RC | Database-specific return code |
| Statement | รูปย่อของคำสั่ง SQL |
| hh:mm:ss:ms | Time stamp |
| Program | ชื่อของโปรแกรม |
| Curs | Database cursor number |

ที่มา: “Performance Optimization Guide,” by Schneider, 2006, p. 166

ตารางที่ 2.3

ข้อมูลของ RFC Trace

| Field | คำอธิบาย |
|---|---|
| Duration | เวลาในการทำงานของ RFC หน่วยเป็นไมโครวินาที |
| Object | ชื่อระบบของผู้รับการเรียก RFC |
| Oper | บอกว่าการ Trace นี้ทำที่ระบบของผู้ส่งการเรียก RFC หรือระบบของผู้รับการเรียก RFC |
| Rec | Field นี้ไม่ได้ใช้งานสำหรับ RFC Trace |
| RC | Return code เช่น 0 หมายถึง สำเร็จ |

ตารางที่ 2.3 (ต่อ)

ข้อมูลของ RFC Trace

| Field | คำอธิบาย |
|---|---|
| Statement | ข้อมูลเพิ่มเติม เช่น ชื่อระบบผู้ส่งและผู้รับการเรียก RFC, ชื่อ RFC module และปริมาณข้อมูลที่ส่งผ่าน |
| hh:mm:ss:ms | Time stamp |
| Program | ชื่อของโปรแกรมที่ใช้ RFC |
| Curs | Field นี้ไม่ได้ใช้งานสำหรับ RFC Trace |

ที่มา: “Performance Optimization Guide,” by Schneider, 2006, p. 171

ตารางที่ 2.4

ข้อมูลของ Enqueue Trace

| Field | คำอธิบาย |
|---|---|
| Duration | เวลาในการทำงานของ คำสั่ง Enqueue และ Dequeue หน่วยเป็นไมโครวินาที |
| Object | ชื่อของ Object ที่ถูก Lock อยู่ |
| Oper | คำสั่ง ENQUEUE, DEQUEUE และ DEQ ALL |
| Rec | จำนวนครั้งของการ ENQUEUE และ DEQUEUE |
| RC | Return code เช่น 0 หมายถึง สำเร็จ |
| Statement | ข้อมูลเพิ่มเติมของคำสั่ง ENQUEUE |
| hh:mm:ss:ms | Time stamp |
| Program | ชื่อของโปรแกรมที่ใช้คำสั่ง ENQUEUE |
| Curs | Field นี้ไม่ได้ใช้งานสำหรับ RFC Trace |

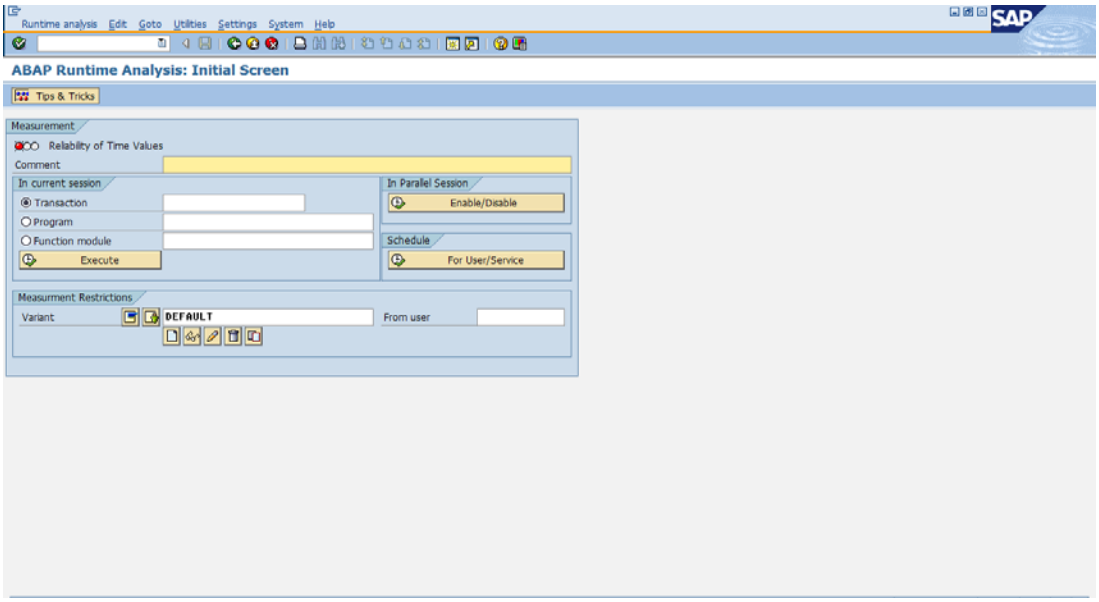
ที่มา: “Performance Optimization Guide,” by Schneider, 2006, p. 172

3.2 ABAP trace เป็นโปรแกรมที่ใช้วิเคราะห์เวลาการทำงานของโปรแกรมทั้งในส่วนของคำสั่ง SQL (Database access) และคำสั่ง ABAP (ABAP statement) ซึ่งสามารถแสดงเวลาการทำงานของส่วนย่อยในโปรแกรมได้ด้วย เช่น ชุดคำสั่ง MODULE, PERFORM, CALL FUNCTION และ SUBMIT รวมไปถึงคำสั่งที่ทำงานกับตัวแปร Internal table เช่น APPEND, COLLECT, SORT และ READ TABLE

การเข้าใช้งาน ABAP trace สามารถเข้าผ่านเมนู System, Resource, Runtime analysis และ Execute ตามลำดับ หรือเข้าผ่านรหัสรายการเปลี่ยนแปลง SE30 จะพบกับจอภาพดังภาพที่ 2.6 ในการ Trace สามารถระบุเงื่อนไขได้ดังนี้ รหัสรายการเปลี่ยนแปลง ชื่อโปรแกรม และชื่อฟังก์ชันโมดูล (Function module) ข้อมูลที่แสดงหลังจากการ Trace เมื่อกดปุ่ม Analysis จะแสดงเป็นกราฟแท่งให้เห็นภาพรวมว่าในส่วนใดที่ใช้เวลามาก และแสดงรายละเอียดได้ดังภาพที่ 2.7 และภาพที่ 2.8 ตามลำดับ

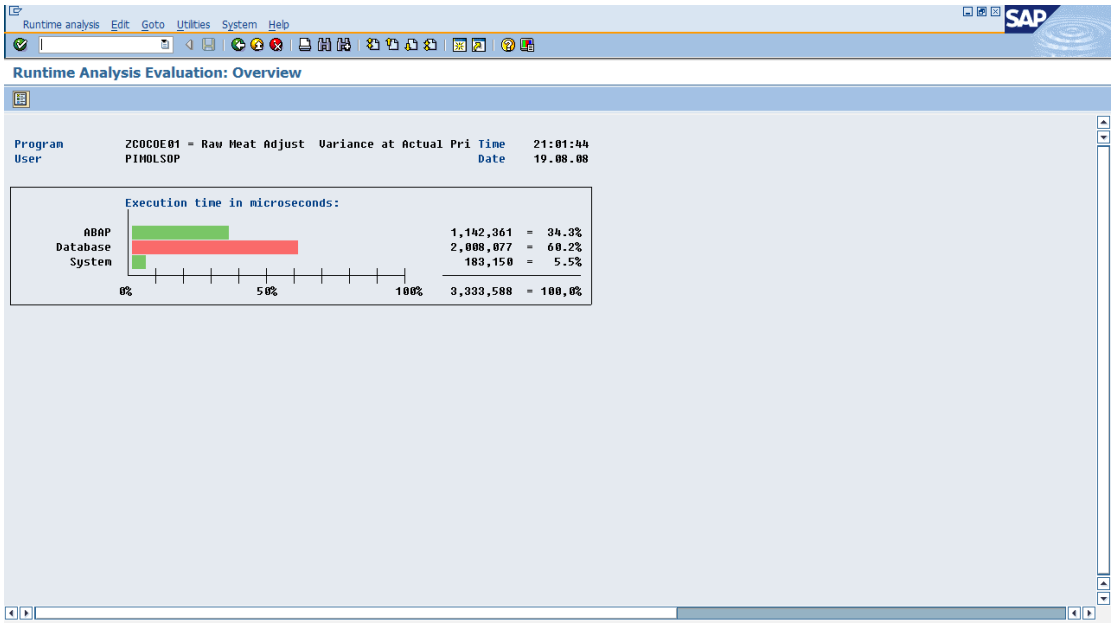
ภาพที่ 2.6

ตัวอย่างจอภาพ ABAP Trace



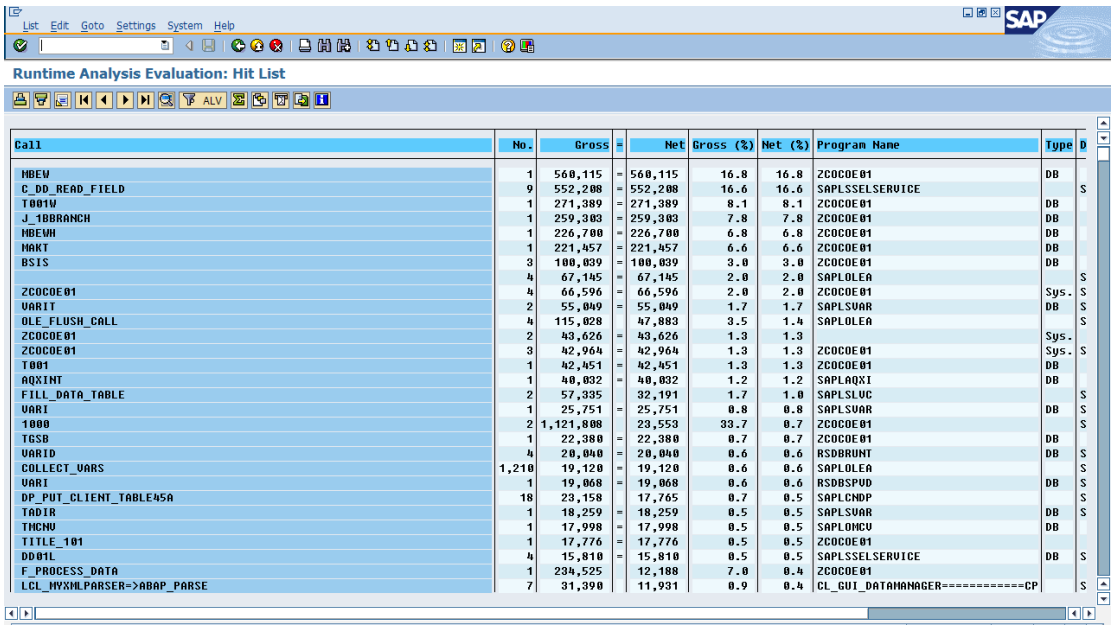
ภาพที่ 2.7

ตัวอย่างจอภาพแสดงภาพรวมของผลจาก ABAP Trace




ภาพที่ 2.8

ตัวอย่างจอภาพรายละเอียดของผลจาก ABAP Trace



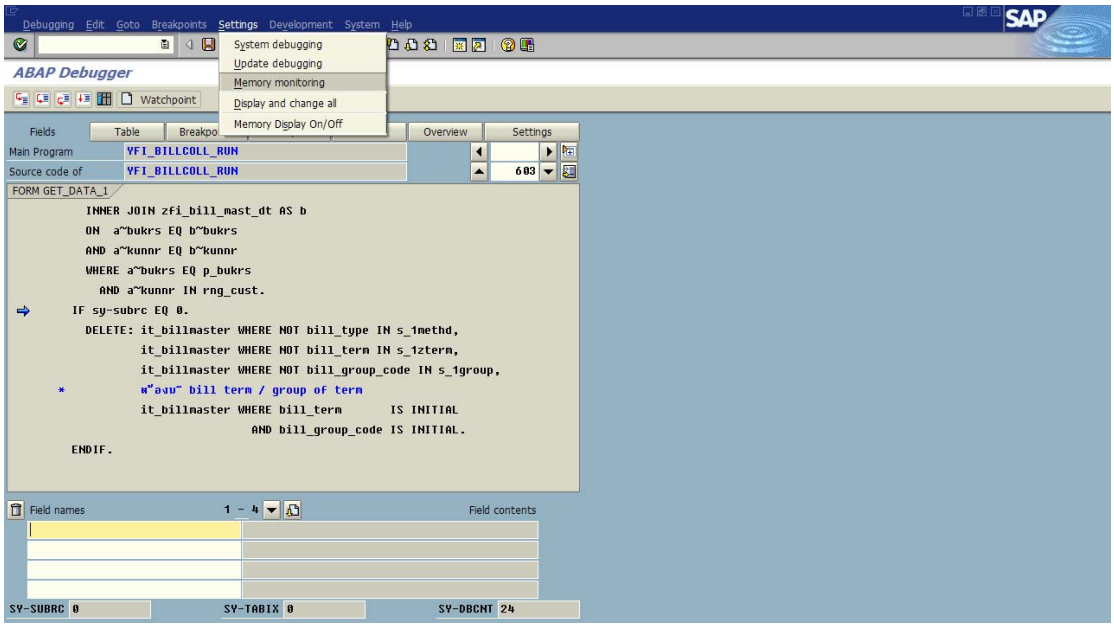

| Call | No. | Gross | = | Net | Gross (%) | Net (%) | Program Name | Type | D |
|---|---|---|---|---|---|---|---|---|---|
| MBEW | 1 | 560,115 | = | 560,115 | 16.8 | 16.8 | ZCOCOE01 | DB | |
| C_DD_READ_FIELD | 9 | 552,208 | = | 552,208 | 16.6 | 16.6 | SAPLSSELSERVICE | | S |
| T001W | 1 | 271,389 | = | 271,389 | 8.1 | 8.1 | ZCOCOE01 | DB | |
| J_1BBRANCH | 1 | 259,303 | = | 259,303 | 7.8 | 7.8 | ZCOCOE01 | DB | |
| MBEWH | 1 | 226,700 | = | 226,700 | 6.8 | 6.8 | ZCOCOE01 | DB | |
| MAKT | 1 | 221,457 | = | 221,457 | 6.6 | 6.6 | ZCOCOE01 | DB | |
| BSIS | 3 | 100,039 | = | 100,039 | 3.0 | 3.0 | ZCOCOE01 | DB | |
| | 4 | 67,145 | = | 67,145 | 2.0 | 2.0 | SAPLOLEA | | S |
| ZCOCOE01 | 4 | 66,596 | = | 66,596 | 2.0 | 2.0 | ZCOCOE01 | Sys. | S |
| VARIT | 2 | 55,049 | = | 55,049 | 1.7 | 1.7 | SAPLSVAR | DB | S |
| OLE_FLUSH_CALL | 4 | 115,028 | | 47,883 | 3.5 | 1.4 | SAPLOLEA | | S |
| ZCOCOE01 | 2 | 43,626 | = | 43,626 | 1.3 | 1.3 | | Sys. | |
| ZCOCOE01 | 3 | 42,964 | = | 42,964 | 1.3 | 1.3 | ZCOCOE01 | Sys. | S |
| T001 | 1 | 42,451 | = | 42,451 | 1.3 | 1.3 | ZCOCOE01 | DB | |
| AQXINT | 1 | 40,032 | = | 40,032 | 1.2 | 1.2 | SAPLAQXI | DB | |
| FILL_DATA_TABLE | 2 | 57,335 | | 32,191 | 1.7 | 1.0 | SAPLSLVC | | S |
| VARI | 1 | 25,751 | = | 25,751 | 0.8 | 0.8 | SAPLSVAR | DB | S |
| 1000 | 2 | 1,121,808 | | 23,553 | 33.7 | 0.7 | ZCOCOE01 | | S |
| TGSB | 1 | 22,380 | = | 22,380 | 0.7 | 0.7 | ZCOCOE01 | DB | |
| VARID | 4 | 20,040 | = | 20,040 | 0.6 | 0.6 | RSDBRUNT | DB | S |
| COLLECT_VARS | 1,210 | 19,120 | = | 19,120 | 0.6 | 0.6 | SAPLOLEA | | S |
| VARI | 1 | 19,068 | = | 19,068 | 0.6 | 0.6 | RSDBSPVD | DB | S |
| DP_PUT_CLIENT_TABLE45A | 18 | 23,158 | | 17,765 | 0.7 | 0.5 | SAPLCNDP | | S |
| TADIR | 1 | 18,259 | = | 18,259 | 0.5 | 0.5 | SAPLSVAR | DB | S |
| TNCNV | 1 | 17,998 | = | 17,998 | 0.5 | 0.5 | SAPLONCV | DB | |
| TITLE_101 | 1 | 17,776 | = | 17,776 | 0.5 | 0.5 | ZCOCOE01 | | |
| DD01L | 4 | 15,810 | = | 15,810 | 0.5 | 0.5 | SAPLSSELSERVICE | DB | S |
| F_PROCESS_DATA | 1 | 234,525 | | 12,188 | 7.0 | 0.4 | ZCOCOE01 | | |
| LCL_MYXMLPARSER=>ABAP_PARSE | 7 | 31,390 | | 11,931 | 0.9 | 0.4 | CL_GUI_DATAMANAGER==========CP | | S |

3.3 Debugger เป็นเครื่องมือที่ช่วยในการหาข้อผิดพลาดของโปรแกรม แต่สามารถนำมาใช้สำหรับตรวจสอบประสิทธิภาพการทำงานของโปรแกรมในทางอ้อมได้ เพื่อช่วยตรวจสอบในเรื่องของหน่วยความจำที่ใช้ โดยเฉพาะอย่างยิ่งเมื่อทำงานกับตัวแปร Internal Table ที่มีขนาดใหญ่ ไม่ว่าจะเป็นการอ่าน เรียง หรือค้นหาข้อมูล

การเรียกดูปริมาณหน่วยความจำที่ใช้ สามารถเข้าผ่านเมนู Settings และ Memory monitoring ตามลำดับ เพื่อเปิดใช้ฟังก์ชันในการแสดงหน่วยความจำ และไปที่เมนู Go to, Status Display และ Memory Use ตามลำดับ ดังภาพที่ 2.9 และภาพที่ 2.10

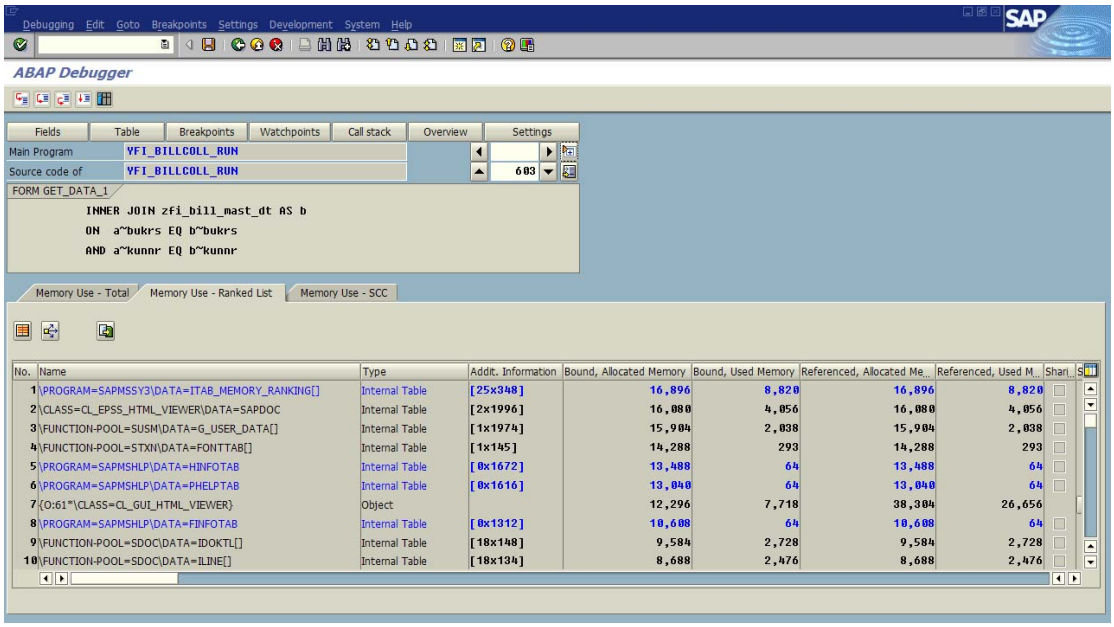
ภาพที่ 2.9

ตัวอย่างจอภาพการเรียกดูปริมาณหน่วยความจำที่ใช้



ภาพที่ 2.10

ตัวอย่างจอภาพแสดงปริมาณหน่วยความจำที่ใช้

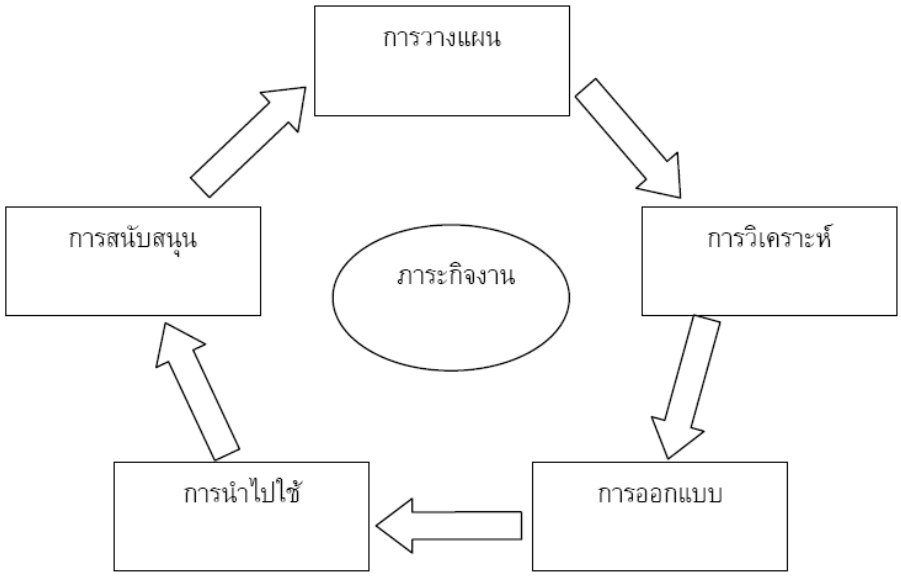



## 2.2 วงจรการพัฒนาระบบ

Gary B. Shelly (1999) ได้นำเสนอวงจรการพัฒนาระบบ (System Development Life Cycle: SDLC) ไว้ดังภาพที่ 2.11 ซึ่งประกอบไปด้วย (1) ขั้นการวางแผนประกอบไปด้วย การพิจารณาถึงความต้องการของโครงการ กำหนดทรัพยากรที่สนันสนุน เช่น งบประมาณ บุคลากร เครื่องมือ เป็นต้น และกำหนดทีมงานในการพัฒนาโครงการ (2) ขั้นการวิเคราะห์ ประกอบด้วย การศึกษาความเป็นไปได้ในการที่จะใช้ระบบเพื่อแก้ปัญหา การวิเคราะห์ภารกิจในรายละเอียด ได้แก่ ศึกษาว่าระบบทำงานได้อย่างไร และการสนองตอบต่อความต้องการของผู้ใช้ (3) ขั้นการออกแบบ ประกอบด้วย การคำนึงถึงการได้มาของฮาร์ดแวร์ และซอร์ฟแวร์ การพัฒนารายละเอียดทั้งหมดของระบบ (4) ขั้นการนำไปใช้ ประกอบด้วย การพัฒนาโปรแกรม การติดตั้ง การทดสอบระบบใหม่ การฝึกอบรม และการให้คำแนะนำแก่ผู้เริ่มต้นใช้ระบบ การเปลี่ยนเพื่อเข้าสู่ระบบที่ใหม่ขึ้น (5) ขั้นการสนับสนุน ประกอบด้วย การพิจารณาภารกิจหลังการใช้ระบบ การระบุข้อผิดพลาดและปรับปรุงให้ดีขึ้น การดูแล และสังเกตการณ์การทำงานของระบบ

ภาพที่ 2.11

วงจรการพัฒนาระบบ




ที่มา: Shelly, 1999

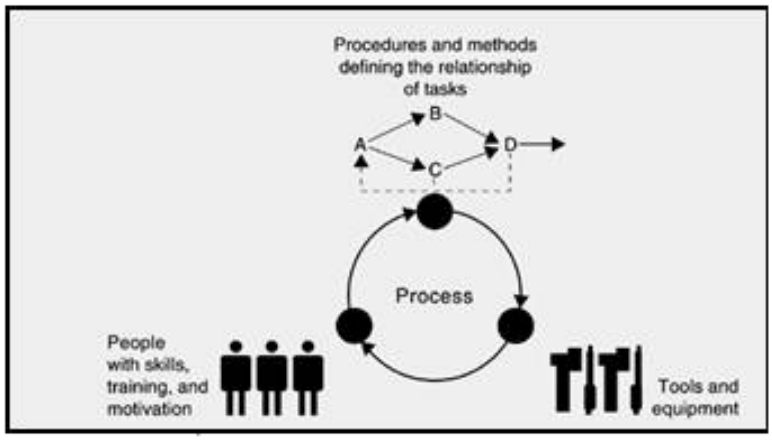
## 2.3 Capability Maturity Model Integration (CMMI)

การที่จะไปถึงมาตรฐาน CMMI จะต้องเริ่มจากการมีกระบวนการซอฟต์แวร์ (Software Process) ที่ชัดเจน ในการพัฒนาซอฟต์แวร์หรือเขียนโปรแกรมนั้น แต่เดิมไม่ค่อยมีใครสนใจในกระบวนการพัฒนามากนัก ผู้พัฒนาแต่ละคนอาจจะมีขั้นตอนต่างกัน เมื่อเขียนโปรแกรมแต่ละครั้งก็อาจดำเนินการไม่เหมือนกัน ดังนั้นผลที่ได้รับจึงไม่คงเส้นคงวา บางครั้งอาจเขียนโปรแกรมได้ผลดี แต่บางครั้งก็อาจจะไม่ได้ผล ด้วยเหตุนี้จึงมีผู้ผลักดันให้เกิดกระบวนการซอฟต์แวร์ขึ้น โดยเชื่อว่ากระบวนการซอฟต์แวร์ที่กำหนดขึ้นอย่างรอบคอบจะช่วยให้เขียนโปรแกรมในแต่ละครั้งมีขั้นตอนที่ชัดเจนและให้ผลลัพธ์ตามที่คาดหมายได้

CMMI เป็นมาตรฐานในการปรับปรุงคุณภาพซอฟต์แวร์ให้มีประสิทธิภาพ เป็นที่รู้จักและยอมรับของสากล CMMI จะมีวิธีการหรือขั้นตอนการปรับปรุงกระบวนการ (Process Improvement) เพื่อปรับปรุงคุณภาพของผลิตภัณฑ์และบริการให้มีประสิทธิภาพตั้งแต่กระบวนการออกแบบ จนถึงการรีลีส (Release) และการบำรุงรักษา (Maintenance) เพื่อให้ทุกองค์กรนำไปใช้ปรับปรุงคุณภาพซอฟต์แวร์ ซึ่งมิติสำคัญสำหรับการปรับปรุงคุณภาพซอฟต์แวร์มี

ด้วยกัน 3 มิติ ได้แก่ (1) คน (2) กระบวนการ และ(3) เครื่องมือ โดยมิติที่เป็นตัวกลางเชื่อมทั้ง 3 มิติเข้าด้วยกัน คือ กระบวนการ แสดงไว้ดังภาพที่ 2.12

ภาพที่ 2.12
มิติสำคัญสำหรับการปรับปรุงคุณภาพซอฟต์แวร์

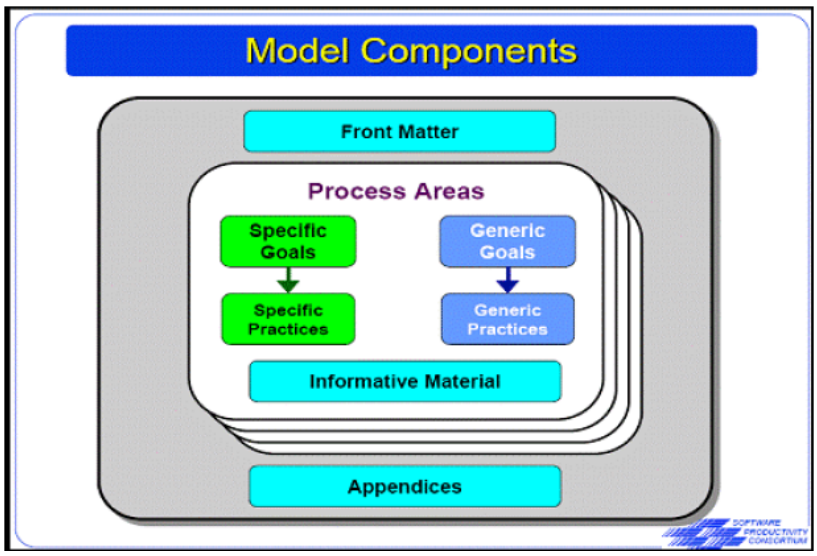



ที่มา: “การตีความ CMMI สำหรับองค์กรการบริการ-วิศวกรรมระบบและตัวอย่างการบริการแบบบูรณาการ” โดย เบญจพล ออประเสิรฐ, 2551, น. 4

CMMI เวอร์ชันที่ใช้กันอยู่ในปัจจุบันเป็นเวอร์ชัน 1.2 ถูกพัฒนาขึ้นโดย Software Engineering Institute (SEI) ในเดือนสิงหาคมปี ค.ศ. 2006 ประกอบไปด้วย 22 กลุ่มกระบวนการ (Process Area) กลุ่มกระบวนการ หมายถึง กลุ่มของข้อปฏิบัติที่ดีที่สุดที่ต้องทำให้เกิดผลแล้วจะทำให้บรรลุวัตถุประสงค์ของงานนั้นๆ หรืออาจมองว่าเป็นแนวทางการปรับปรุงกระบวนการทำงานด้านต่างๆ ซึ่งแต่ละงานอาจต้องทำหลายกลุ่มกระบวนการก็ได้ ซึ่งจะเป็นแนวทางที่ดีที่จะช่วยให้องค์กรที่ต้องการทำ CMMI นำไปปฏิบัติ ภาพที่ 2.13 แสดงองค์ประกอบในแบบจำลอง CMMI ซึ่งมีกลุ่มกระบวนการเป็นองค์ประกอบหลัก โครงสร้างหลักภายในกลุ่มกระบวนการ ประกอบด้วย เป้าหมาย (Goals) และข้อปฏิบัติ (Practices) โดยที่ประเภทของเป้ามายและข้อปฏิบัติแบ่งเป็นทั่วไป (Generic) และเฉพาะ (Specific)

ภาพที่ 2.13

องค์ประกอบแบบจำลอง CMMI




ที่มา: “การตีความ CMMI สำหรับองค์กรการบริการ-วิศวกรรมระบบและตัวอย่างการบริการแบบบูรณาการ” โดย เบญจพล ออประเสิรฐ, 2551, น. 4

กลุ่มกระบวนการทั้ง 22 กลุ่มประกอบด้วย

1. กลุ่มกระบวนการวิเคราะห์สาเหตุและการแก้ไขปัญหา (Causal Analysis and Resolution: CAR)

2. กลุ่มกระบวนการจัดการโครงแบบ (Configuration Management: CM)

3. กลุ่มกระบวนการวิเคราะห์ การตัดสินใจและการแก้ปัญหา (Decision Analysis and Resolution: DAR)

4. กลุ่มกระบวนการจัดการบูรณาการโครงการ (Integrated Project Management: IPM)

5. กลุ่มกระบวนการวัดและการวิเคราะห์ (Measurement and Analysis: MA)

6. กลุ่มกระบวนการนวัตกรรมองค์กรและการนำไปใช้ (Organizational Innovation and Deployment: OID)

7. กลุ่มกระบวนการข้อกำหนดของกระบวนการองค์กร (Organizational Process Definition: OPD)

8. กลุ่มกระบวนการจุดมุ่งหมายของกระบวนการองค์กร (Organizational Process Fogus: OPF)

9. กลุ่มกระบวนการปฏิบัติงานของกระบวนการองค์กร (Organizational Process Performance: OPP)

10. กลุ่มกระบวนการฝึกอบรมขององค์กร (Organizational Traning: OT)

11. กลุ่มกระบวนการบูรณาการผลิตภัณฑ์ (Product Integration: PI)

12. กลุ่มากระบวนการเฝ้าติดตามและควบคุมโครงการ (Project Monitoring and Control: PMC)

13. กลุ่มกระบวนการวางแผนโครงการ (Project Planning: PP)

14. กลุ่มกระบวนการประกันคุณภาพของกระบวนการและผลิตภัณฑ์ (Process and Product Quality Assurance: PPQA)

15. กลุ่มกระบวนการจัดการทรัพยากรโครงการ (Quantitative Project Management: QPM)

16. กลุ่มกระบวนการพัฒนาความต้องการ (Requirements Development: RD)

17. กลุ่มกระบวนการจัดการความต้องการ (Requirements Management: REQM)

18. กลุ่มกระบวนการบริหารความเสี่ยง (Risk Management: RSKM)

19. กลุ่มกระบวนการจัดการข้อตกลงกับผู้จัดหา (Supplier Agreement Management: SAM)

20. กลุ่มกระบวนการแก้ปัญหาเชิงเทคนิค (Technical Solution: TS)

21. กลุ่มกระบวนการตรวจสอบความถูกต้อง (Validation: VAL)

22. กลุ่มกระบวนการทวนสอบ (Verification: VER)

กลุ่มกระบวนการเหล่านี้ได้ถูกจัดกลุ่มตามรูปแบบของการนำเสนอ CMMI ซึ่งมีด้วยกัน 2 รูปแบบ คือ (1) รูปแบบ Stage แสดงระดับการปรับปรุงด้วยระดับวุฒิภาวะ (Maturity Level) และ (2) รูปแบบ Continuous แสดงระดับการปรับปรุงด้วยระดับความสามารถ (Capability Level) โดยในแต่ละรูปแบบการนำเสนอ CMMI กลุ่มกระบวนการจะถูกจัดกลุ่มด้วยโครงสร้างต่างกัน กล่าวคือ รูปแบบ Stage กลุ่มกระบวนการจะถูกแบ่งตามระดับวุฒิภาวะ ส่วนรูปแบบ Continuous กลุ่มกระบวนการจะถูกแบ่งตามประเภท มี 4 ประเภท คือ การบริหารโครงการ (Project Management) การจัดการประบวนการ (Process Management) วิศวกรรม

(Engineering) และสนับสนุน (Support) ตารางที่ 2.5 การเปรียบเทียบรูปแบบการนำเสนอ CMMI ตารางที่ 2.6 การเปรียบเทียบระดับการปรับปรุงกระบวนการ และตารางที่ 2.7 การจำแนกประเภทกลุ่มประบวนการ (เบญจพล ออประเสิรฐ, 2008, น.1-7)

ตารางที่ 2.5
การเปรียบเทียบรูปแบบการนำเสนอ CMMI

| Stage Representation | Continuous Representation |
|---|---|
| 1) เป็นรูปแบบเดิมที่ใช้ในโมเดล CMM | 1) องค์กรสามารถเลือกกลุ่มกระบวนการมาเพียง 1 กลุ่มและปรับปรุงเฉพาะประสิทธิภาพของกลุ่มกระบวนการนั้นให้ดีขึ้น |
| 2) มีการกำหนดเป็นระดับวุฒิภาวะขององค์กร (ระดับ 1 - ระดับ 5) | 2) มีการกำหนดเป็นระดับความสามารถของงานแต่ละด้าน (ระดับ 0 – ระดับ 5 ) |
| 3) แต่ละระดับต้องมีการปรับปรุงกลุ่มกระบวนการที่กำหนด | 3) มีความยืดหยุ่นกว่ารูปแบบ Stage เพราะสามารถเลือกปรับปรุงกลุ่มกระบวนการที่ต้องการปรับปรุงได้ |
| 4) มีเส้นทางในการทำ CMMI เมื่อทำตามกลุ่มกระบวนการที่มีให้ครบถ้วน จะถือว่าผ่าน | 4) หรืออาจเลือกหลายๆกลุ่มกระบวนการที่ตรงกับวัตถุประสงค์เชิงธุรกิจ |

ที่มา: “การตีความ CMMI สำหรับองค์กรการบริการ-วิศวกรรมระบบและตัวอย่างการบริการแบบบูรณาการ” โดย เบญจพล ออประเสิรฐ, 2551, น. 6

ตารางที่ 2.6
การเปรียบเทียบระดับการปรับปรุงกระบวนการ

| Level | Stage Representation: Maturity Levels | Continuous Representation: Capability Levels |
|---|---|---|
| 0 | N/A | Not Performed |
| 1 | Performed | Performed |
| 2 | Managed | Managed |

ตารางที่ 2.6 (ต่อ)

การเปรียบเทียบระดับการปรับปรุงกระบวนการ

| Level | Stage Representation: Maturity Levels | Continuous Representation: Capability Levels |
|---|---|---|
| 3 | Defined | Defined |
| 4 | Quantitatively | Quantitatively |
| 5 | Optimizing | Optimizing |

ที่มา: “การตีความ CMMI สำหรับองค์กรการบริการ-วิศวกรรมระบบและตัวอย่างการบริการแบบบูรณาการ” โดย เบญจพล ออประเสิรฐ, 2551, น. 6

ตารางที่ 2.7

การจำแนกประเภทกลุ่มกระบวนการ

| ระดับวุฒิภาวะ | ประเภท กลุ่มบริหารงานโครงการ | ประเภท กลุ่มการจัดการกระบวนการ | ประเภท กลุ่มวิศวกรรม | ประเภท กลุ่มสนับสนุน |
|---|---|---|---|---|
| ระดับ 5 | | OID | | CAR |
| ระดับ 4 | QPM | OPP | | |
| ระดับ 3 | IPM, RSKM | OPF, OPD, OT | RD, TS, PI, VER, VAL | DAR |
| ระดับ 2 | PP, PMC, SAM | | REQM | CM, MA, PPQA |

ที่มา: “การตีความ CMMI สำหรับองค์กรการบริการ-วิศวกรรมระบบและตัวอย่างการบริการแบบบูรณาการ” โดย เบญจพล ออประเสิรฐ, 2551, น. 7

กลุ่มกระบวนการประเภทกลุ่มสนับสนุนเป็นกลุ่มที่ครอบคลุมกิจกรรมที่ใช้ในการพัฒนาและการบำรุงรักษาผลิตภัณฑ์ และกลุ่มกระบวนการนี้ได้ถูกกล่าวถึงในการนำไปใช้ปฏิบัติงานภายใต้บริบทของกระบวนการอื่นๆ ด้วย ยกตัวอย่างเช่น กลุ่มกระบวนการประกัน

คุณภาพของกระบวนการและผลิตภัณฑ์สามารถนำไปใช้ได้ในทุกกระบวนการ เพื่อสร้างวัตถุประสงค์ของการประเมินกระบวนการและชิ้นงานของกระบวนการนั้นๆ เป็นต้น

กลุ่มกระบวนการประเภทสนับสนุนประกอบไปด้วย (1) กลุ่มกระบวนการจัดการโครงแบบ (2) กลุ่มกระบวนการวัดและการวิเคราะห์ (3) กลุ่มกระบวนการประกันคุณภาพของกระบวนการและผลิตภัณฑ์ (4) กลุ่มกระบวนการวิเคราะห์ การตัดสินใจและการแก้ปัญหา และ (5) กลุ่มกระบวนการวิเคราะห์สาเหตุและการแก้ไขปัญหา

กลุ่มกระบวนการประเภทสนันสนุนขั้นพื้นฐาน คือ การสนับสนุนที่สำคัญที่ถูกนำไปใช้โดยทุกกลุ่มกระบวนการ ถึงแม้ว่าสิ่งนำเข้าของกลุ่มกระบวนการประเภทสนับสนุนจะมาจากกลุ่มกระบวนการอื่น แต่กลุ่มกระบวนการนี้ก็นำมาซึ่งข้อปฏิบัติทั่วไปที่ใช้ในการสนับสนุน การทำงานของกลุ่มกระบวนการประเภทสนันสนุนขั้นพื้นฐานแสดงดังภาพที่ 2.14

กลุ่มกระบวนการจัดการโครงแบบสนับสนุนทุกกลุ่มกระบวนการโดยสร้างและบำรุงรักษาความน่าเชื่อถือของชิ้นงานด้วยโครงแบบการระบุ โครงแบบการควบคุม โครงแบบบัญชีสถานะ และโครงแบบการสอบ ชิ้นงานจะอยู่ภายใต้การจัดการโครงแบบที่ประกอบไปด้วยผลิตภัณฑ์ที่จะส่งมอบให้ลูกค้า การตั้งชื่อชิ้นงานภายใน การได้มาซึ่งผลิตภัณฑ์ เครื่องมือ และอื่นๆ ที่ใช้ในการสร้างชิ้นงานเหล่านี้ ตัวอย่างของชิ้นงาน ได้แก่ แผนงาน คำจำกัดความของกระบวนการ ความต้องการ ข้อมูลการออกแบบ งานเขียนแบบ รายละเอียดของผลิตภัณฑ์ ชุดคำสั่ง โปรแกรมแปลชุดคำสั่ง ไฟล์ข้อมูลผลิตภัณฑ์ และเอกสารเชิงเทคนิคของผลิตภัณฑ์

กลุ่มกระบวนการวัดและการวิเคราะห์เป็นกลุ่มกระบวนการที่จัดเตรียมข้อปฏิบัติเฉพาะเพื่อเป็นแนวทางให้กับโครงการและองค์กรถึงสิ่งจำเป็นสำหรับการวัด วัตถุประสงค์และวิธีการวัดเพื่อให้ได้ผลของการวัด ซึ่งผลเหล่านี้จะเป็นสิ่งที่สามารถนำไปใช้ในการตัดสินใจและเลือกแนวทางการแก้ไขปัญหาที่ถูกต้องเหมาะสม

กลุ่มกระบวนการประกันคุณภาพของกระบวนการและผลิตภัณฑ์เป็นกลุ่มกระบวนการที่จัดเตรียมข้อปฏิบัติเฉพาะสำหรับการประเมินกระบวนการทำงาน ชิ้นงาน การบริการ มาตรฐานและระเบียบปฏิบัติ กระบวนการประกันคุณภาพของกระบวนการและผลิตภัณฑ์เป็นกระบวนการที่สนับสนุนการส่งมอบผลิตภัณฑ์และการบริการที่มีคุณภาพสูงซึ่งเกิดจากความร่วมมือกันของทีมงานและผู้จัดการทุกระดับด้วยทรรศนะที่เหมาะสม รวมไปถึงผลการตอบกลับของงาน กระบวนการทำงานและชิ้นงานที่เกี่ยวข้องตลอดระยะเวลาของโครงการ

กลุ่มกระบวนการประเภทสนันสนุนขั้นสูงจัดเตรียมการปรับปรุงพัฒนาความสามารถในการสนับสนุนให้กับโครงการและองค์กร และในแต่ละกลุ่มกระบวนการมีสิ่งนำเข้าที่

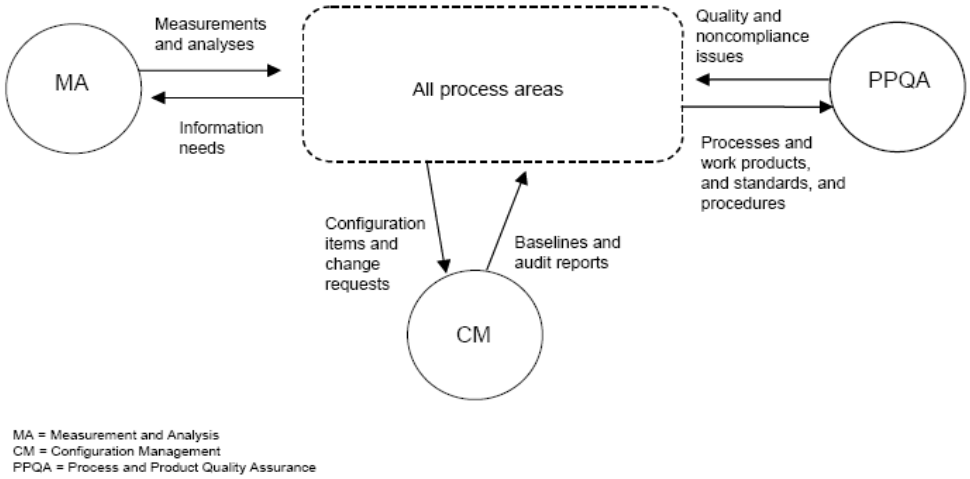
เฉพาะเจาะจงมาจากกลุ่มกระบวนการอื่นๆ การทำงานของกลุ่มกระบวนการประเภทสนันสนุนขั้นสูงแสดงดังภาพที่ 2.15

กลุ่มกระบวนการวิเคราะห์สาเหตุและการแก้ไขปัญหา ในกลุ่มกระบวนการนี้ทีมงานในโครงการเป็นผู้ระบุสาเหตุจากข้อบกพร่องที่เลือกมารวมทั้งปัญหาอื่นๆ ด้วย และลงมือปฏิบัติการป้องกันเพื่อไม่ให้ข้อบกพร่องหรือปัญหานั้นเกิดขึ้นอีกในอนาคต เมื่อกระบวนการของโครงการถูกกำหนดขึ้นเป็นหลักการในการระบุปัญหาจากข้อบกพร่อง และข้อเสนอในการปรับปรุงกระบวนการ แล้วจึงค่อยสร้างมาตรฐานกระบวนการสำหรับองค์กร เพื่อป้องกันการเกิดข้อบกพร่องในระดับองค์กร

กลุ่มกระบวนการวิเคราะห์ การตัดสินใจและการแก้ปัญหาสนับสนุนกลุ่มกระบวนการอื่นๆ ในการระบุปัญหาซึ่งอยู่ภายใต้ระเบียบของกระบวนการประเมิน ซึ่งผลของกระบวนการนี้ถูกนำไปดำเนินงานต่อด้วยกระบวนการประเมิน (CMMI Product Team, 2006, p.62-64)

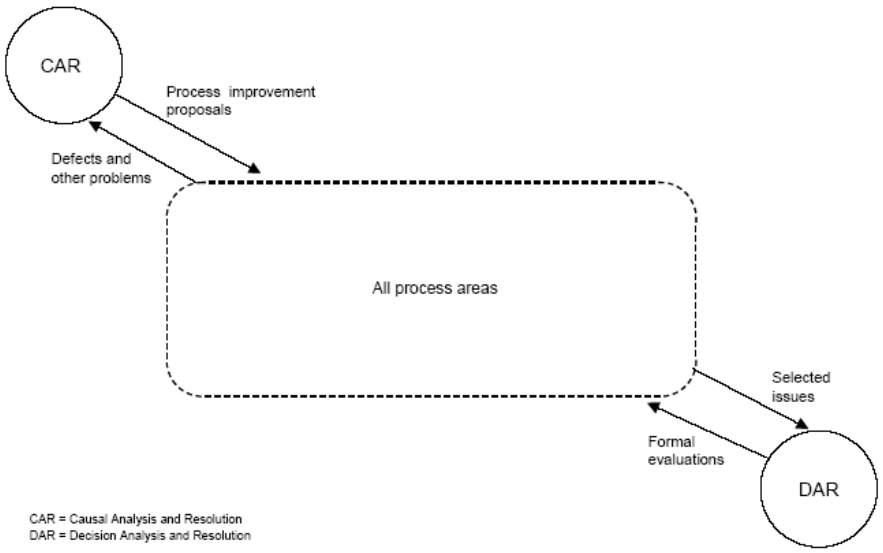
ภาพที่ 2.14

การทำงานของกลุ่มกระบวนการประเภทสนันสนุนขั้นพื้นฐาน




ที่มา: “CMMI for Development, Version 1.2” by CMMI Product Team, 2006, p. 63

ภาพที่ 2.15
การทำงานของกลุ่มกระบวนการประเภทสนันสนุนขั้นสูง




ที่มา: “CMMI for Development, Version 1.2” by CMMI Product Team, 2006, p. 64

## 2.4 **แผนผังสาเหตุและผล** (Cause and Effect Diagram)

แผนผังสาเหตุและผลหรือเรียกอีกชื่อหนึ่งว่า ผังก้างปลา (Fish Bone Diagram) เนื่องจากหน้าตาแผนภูมิมีลักษณะคล้ายปลาที่เหลือแต่ก้างหรือหลายๆ คนอาจรู้ จักในชื่อของแผนผังอิชิกาว่า (Ishikawa Diagram) ซึ่งได้รับการพัฒนาครั้งแรกเมื่อปี ค.ศ. 1943 โดยศาสตราจารย์คาโอรุ อิชิกาว่า แห่งมหาวิทยาลัยโตเกียว แผนผังสาเหตุและผลเป็นแผนผังที่แสดงถึงความสัมพันธ์ระหว่างปัญหา (Problem) กับสาเหตุทั้งหมดที่เป็นไปได้ที่อาจก่อให้เกิดปัญหานั้น (Possible Cause) เราจะใช้แผนผังสาเหตุและผลเมื่อต้องการค้นหาสาเหตุแห่งปัญหาและทำความเข้าใจ หรือทำความรู้จักกับกระบวนการอื่น ๆ เพราะว่าโดยส่วนใหญ่พนักงานจะรู้ปัญหาเฉพาะในพื้นที่ของตนเท่านั้น แต่เมื่อมีการ ทำผังก้างปลาแล้ว จะทำให้เราสามารถรู้กระบวนการของแผนกอื่นได้ง่ายขึ้น นอกจากนี้ยังใช้เพื่อต้องการให้เป็นแนวทางในการระดมสมอง ซึ่งจะช่วยให้ทุกๆ คนให้ความสนใจในปัญหาของกลุ่มซึ่งแสดงไว้ที่หัวปลา

สิ่งสำคัญในการสร้างแผนผัง คือ ต้องทำเป็นทีม เป็นกลุ่ม โดยใช้ขั้นตอน 4 ขั้นตอน ดังต่อไปนี้

1. กำหนดประโยคปัญหาที่หัวปลา

2. กำหนดกลุ่มปัจจัยที่จะทำให้เกิดปัญหานั้นๆ

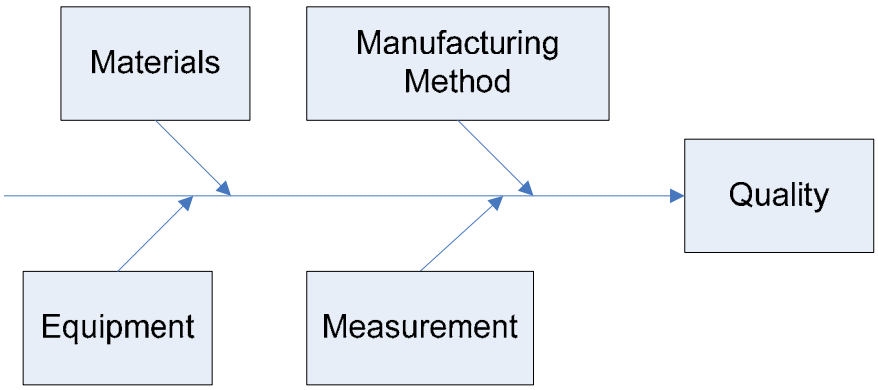
3. ระดมสมองเพื่อหาสาเหตุในแต่ละปัจจัยและหาสาเหตุหลักของปัญหา

4. จัดลำดับความสำคัญของสาเหตุและเขียนสาเหตุย่อยในแต่ละสาเหตุหลัก

ปัจจัยที่จะช่วยให้เราแยกแยะและกำหนดสาเหตุต่างๆ ได้อย่างเป็นระบบ และเป็นเหตุเป็นผล ปัจจัยเหล่านั้น ได้แก่ วัตถุดิบ (Materials) สิ่งอำนวยความสะดวกหรือเครื่องมือต่างๆ (Facilities or Equipment) และกระบวนการในการผลิต (Manufaturing Method) ตัวอย่างแผนผังสาเหตุและผลแสดงได้ดังภาพที่ 2.16

ภาพที่ 2.16

ตัวอย่างแผนผังสาเหตุและผล




ที่มา: “Guide to Quality Control” by Ishikawa, 1972, p.20

## 2.5 การปรับปรุงประสิทธิภาพของโปรแกรม ABAP/4

ประสิทธิภาพในเชิงเทคนิคของระบบ SAP R/3 หมายถึง ความสามารถในการประมวลผลข้อมูลของระบบ เพื่อตอบสนองต่อความต้องการของผู้ใช้ในเรื่องของเวลาในการตอบสนอง (Response Time) และปริมาณงาน (Throughput) ที่ทำได้ ยกตัวอย่างเช่น ผู้ใช้ต้องการให้ระบบสามารถประมวลผลใบแจ้งราคาสินค้า (Invoice) จำนวน 10,000 ใบภายในเวลา 1 ชั่วโมง หรือจะเป็นการใช้เวลาในการประมวลผลใบสั่งซื้อจากลูกค้าภายในเวลาที่น้อยกว่า 1 วินาที เป็นต้น ประสิทธิภาพการทำงานของระบบจึงไม่ได้ขึ้นอยู่กับคุณสมบัติที่ดีของระบบเพียงอย่างเดียว แต่รวมไปถึงการตอบสนองต่อความต้องการของผู้ใช้ด้วย

การปรับปรุงประสิทธิภาพของระบบถูกแบ่งออกเป็น 2 ส่วน คือ การปรับปรุงประสิทธิภาพด้านกระบวนการทางธุรกิจ (Business Process Tuning) และการปรับปรุงประสิทธิภาพด้านเทคนิค (Technical Tuning)

การปรับปรุงประสิทธิภาพด้านกระบวนการทางธุรกิจเริ่มต้นด้วยการวิเคราะห์กระบวนการทางธุรกิจ (Business Process) โดยกระบวนการหลักทางธุรกิจคือโปรแกรมที่มีการทำงานเหมือนกับขั้นตอนของธุรกิจการค้า ตัวอย่างเช่น ในกระบวนการธุรกิจต้องมีสร้างเอกสาร ได้แก่ ใบสั่งซื้อจากลูกค้า ใบส่งของ และใบแจ้งราคาสินค้า ซึ่งกระบวนการสร้างเอกสารเหล่านี้จะต้องผ่านการตรวจสอบทางการบัญชีและการบริหาร โดยทำงานผ่านโปรแกรมต่างๆ เป็นต้น เมื่อทราบการทำงานต่างๆ ของกระบวนการทางธุรกิจแล้ว จึงมาพิจารณาว่าโปรแกรมหรือรายการทำงาน (Transaction) ใดที่ใช้ทรัพยากรของระบบมากที่สุด ส่วนใดของโปรแกรมเป็นสาเหตุที่ทำให้เกิดภาระงานหนักในฝั่งฐานข้อมูล การเข้าถึงฐานข้อมูลส่วนใดที่ใช้เวลานานหรือใช้ทรัพยากรหน่วยความจำ (Memory) หรือหน่วยประมวลผลกลาง (Central Processing Unit: CPU) มากเกินไป คำถามเหล่านี้เป็นสิ่งที่ต้องตระหนักถึงในขั้นตอนการจัดทำระบบ SAP R/3 ให้เป็นไปตามความต้องการของผู้ใช้ (Customizing Phase) เพราะเป็นสิ่งที่มีอิทธิพลต่อประสิทธิภาพการทำงานของระบบ SAP R/3 ต่อจากนั้นใช้การปรับปรุงประสิทธิภาพโปรแกรมประยุกต์ (Application Tuning) คือ การปรับปรุงประสิทธิภาพของโค้ดโปรแกรม ABAP ซึ่งมีประโยชน์สำหรับการพัฒนาโปรแกรมตามความต้องการของลูกค้า การดัดแปรของลูกค้า (Customer Modification) เข้าไปที่มาตรฐานของระบบ SAP R/3 รวมไปถึงการใช้งานโปรแกรมทางออกสำหรับผู้ใช้ (User Exit Program: เป็นส่วนที่ระบบ SAP R/3 จัดไว้ให้สำหรับแต่ละองค์กร เพื่อแทรกการทำงานที่เพิ่มเติมจากการทำงานมาตรฐานของระบบ) การปรับปรุงประสิทธิภาพของโค้ดโปรแกรม ABAP ทำได้โดย (1) การปรับเปลี่ยนคำสั่งโปรแกรมโดยใช้รูปแบบคำสั่ง SQL ที่มีประสิทธิภาพแทนการทำงานของคำสั่งโปรแกรมที่ซับซ้อน ซึ่งทำให้ฝั่งฐานข้อมูลมีภาระงานหนัก (2) ถ้าระบบการจัดการฐานข้อมูลเชิงสัมพันธ์มีภาระงานหนัก ควรย้ายการทำงานของส่วนฐานข้อมูลโดยใช้การประมวลผลข้อมูลมาไว้ที่ Application Server เช่น การเรียงลำดับข้อมูล และการจัดกลุ่มข้อมูล เป็นต้น (3)ปรับเปลี่ยนคำสั่งโปรแกรมจากการใช้คำสั่ง Open SQL มาใช้คำสั่ง Native SQL (4) สร้าง Index ให้กับตารางในฐานข้อมูล (Kemper, Kossmann, & Zeller, 1999, p.9-10)

สิ่งสำคัญที่ต้องคำนึงถึงเมื่อจะทำการปรับปรุงประสิทธิภาพด้านกระบวนการทางธุรกิจ คือ จะต้องไม่กระทบกับกระบวนการทำงานมาตรฐานของระบบ SAP R/3 โดยหลังจากการ

ปรับปรุงจะต้องทดสอบทั้งกระบวนการทำงานมาตรฐานและโปรแกรมที่พัฒนาเพิ่ม ว่ายังคงทำงานได้ถูกต้องตามเดิม การปรับปรุงประสิทธิภาพด้านกระบวนการทางธุรกิจควรเริ่มทำตั้งแต่เนิ่นๆ ตั้งแต่ก่อนเริ่มช่วงของการจัดทำระบบ และการพัฒนาโปรแกรมเพิ่ม โดยผู้รับผิดชอบในส่วนนี้จะเป็นหน่วยงานที่รับผิดชอบในกระบวนการทางธุรกิจนั้นๆ และผู้จัดทำระบบในส่วนของโปรแกรมประยุกต์และการพัฒนาโปรแกรมเพิ่ม

การปรับปรุงประสิทธิภาพด้านเทคนิคเกี่ยวพันกับโครงแบบในทุกส่วนประกอบของระบบ SAP R/3 เพื่อให้การทำงานของผู้ใช้เป็นไปอย่างเหมาะสม และไม่เกิดปัญหาคอขวด ส่วนประกอบเหล่านั้น ได้แก่ ระบบปฏิบัติการ (Operating System: OS) ฐานข้อมูล กระบวนการทำงานของ SAP (SAP Work Process) บัฟเฟอร์ของ SAP (SAP Buffer) การจัดการหน่วยความจำ (Memory Management) และข่ายงาน ล้วนแล้วเป็นเรื่องของฮาร์ดแวร์ ผู้รับผิดชอบในส่วนนี้ ได้แก่ ผู้บริหารระบบ (System Administrator) ผู้บริหารฐานข้อมูล (Database Administrator) และผู้จัดทำระบบในด้านเทคนิค (Schneider, 2001, p. 333-336)

ข้อมูลมีปริมาณเพิ่มมาขึ้นทุกวัน ส่วนที่จะเกิดปัญหาคือการจัดการฐานข้อมูล วิธีทางเทคนิคที่จะช่วยแก้ไขปัญหาปริมาณข้อมูลที่เพิ่มมากขึ้น คือ เทคนิคการแบ่งแยกตามแนวนอน (Horizontal Partitioning) ซึ่งเป็นเทคนิคในเรื่องของการจัดการฐานข้อมูลที่นำมาใช้แทนเทคนิคการสร้างดรรชนี (Index) ให้กับตารางในฐานข้อมูล ซึ่งเป็นการแก้ไขปัญหาแบบดั้งเดิม โดยดรรชนีจะถูกสร้างตามเงื่อนไขในการอ่านข้อมูลของโปรแกรม แต่ข้อเสียคือ การแก้ไขและการเพิ่มข้อมูลจะช้าลง และยากต่อการบำรุงรักษาฐานข้อมูล เพราะถ้าตารางมีขนาดใหญ่ขึ้น ดรรชนีย่อมเพิ่มขนาดตามไปด้วย และเมื่อใช้การค้นหาแบบทวิภาค (Binary Search) (Zeller & Kemper, 2002, p.3-4) ดรรชนีมักจะไม่ทำงาน เพราะระบบวางแผนทรัพยากรทางธุรกิจโดยรวมส่วนใหญ่ จะใช้การค้นหาตามลำดับจากน้อยไปมากแทน (Cesarini and Soda, 1983, p.13-16)

### 2.5.1 การเลือกใช้คำสั่งเพื่อปรับปรุงประสิทธิภาพของโปรแกรม

1. จัดกลุ่มการอ่านข้อมูล

ควรวางแผนการอ่านข้อมูลจากฐานข้อมูล โดยส่วนที่เป็นกระบวนการทำงานที่ซ้อนกันอยู่ (Nested Process) พยายามจัดกลุ่มคำสั่งที่อ่านข้อมูลจากตารางในฐานข้อมูลเดียวกันให้อยู่ด้วยกัน เพื่อหลีกเลี่ยงการอ่านข้อมูลที่ซ้ำซ้อน (G. Propfe, 2001, p.212; SAP Online Help)

2. อ่านข้อมูลตามหลักความสัมพันธ์ของตารางในฐานข้อมูล

วิธีการที่ทำให้โปรแกรมทำงานได้เร็วขึ้น ก่อนอื่นให้พิจารณาที่คำสั่ง SELECT จะต้องปฏิบัติตามกฎในเรื่องความสัมพันธ์ของข้อมูลที่ว่า ต้องอ่านข้อมูลจากตาราง Header ก่อนแล้วจึงอ่านข้อมูลที่เกี่ยวข้องจากตาราง Item แต่ถ้าหากความต้องการของผู้ใช้ทำให้ต้องอ่านข้อมูลจากตาราง Item ก่อน ก็ควรจะ COLLECT ข้อมูลก่อน แล้วจึงอ่านข้อมูลจากตาราง Header ด้วย FOR ALL ENTRIES ส่วนคำสั่งหรือกระบวนการทำงานที่ซ้อนกัน ควรเปลี่ยนไปใช้คำสั่งในรูปแบบอื่นเพื่อช่วยปรับปรุงประสิทธิภาพการทำงานของโปรแกรม เช่น ใช้การ JOIN หรืออ่านข้อมูลขึ้นมาพักไว้ในตัวแปร Internal table ก่อน แล้วจึงอ่านข้อมูลอีกตารางหนึ่ง นอกจากนี้สิ่งที่ควรระวังอีกอย่างหนึ่งคือ ความสัมพันธ์แบบลำดับขั้นของตารางในฐานข้อมูล เช่น MARA, MARC และ MARD จัดเป็นกลุ่มของตารางที่มีความสัมพันธ์กันแบบลำดับขั้น ตาราง MARA เก็บข้อมูลของ Material master ตาราง MARC เก็บข้อมูลของ Material ว่ามีอยู่ที่ Plant ใดบ้าง และตาราง MARD เก็บข้อมูลของ Material ว่ามีอยู่ที่ Plant และ Storage location ใดบ้าง ส่วน MARA, MARC และ MVKE ไม่ถือเป็นกลุ่มของตารางที่มีความสัมพันธ์กันแบบลำดับขั้น เพราะ ตาราง MVKE เก็บข้อมูลทางด้านการขาย ความสัมพันธ์แบบลำดับขั้นของตาราง สามารถดูได้จาก Key ของตารางที่สนใจ และค้นหาว่ามันถูกเรียกใช้ที่ตารางใดบ้าง หรือจะดูจาก Logical database ที่มีในระบบ ถ้าระบบอ่านข้อมูลจากฐานข้อมูลอย่างไร นั่นคือวิธีที่ดีที่สามารถนำมาเลียนแบบใช้ได้ ถึงแม้ว่า Standard ไม่ได้อ่านข้อมูลด้วย Key ของตาราง แต่จะไปใช้ประโยชน์จาก Index ของตารางแทน การเพิ่มเงื่อนไขใน WHERE clause ด้วย field ที่ไม่ใช่ Key หรือ Index จะเป็นการทำให้โปรแกรมทำงานได้ช้าลง (G. Propfe, 2001, p.212)

3. อ่านข้อมูลจากตารางที่เหมาะสม

หากทราบว่าข้อมูลที่ต้องการใช้มีอัตราการเพิ่มของข้อมูลในปริมาณเท่าๆกันต่อหน่วยเวลา เช่น ในแต่ละเดือนจะมีปริมาณข้อมูลเท่าๆกัน และข้อมูลที่ต้องการใช้เป็นข้อมูลตลอดปี ควรออกแบบการดึงข้อมูลให้เป็นแบบรายเดือน เพื่อเป็นการลดปริมาณข้อมูลที่เข้ามาผ่านการทำงานในโปรแกรม หรือดึงข้อมูลจากตารางที่เก็บค่าเป็นยอดรวมต่อเดือนซึ่งระบบ SAP มีไว้ให้อยู่แล้ว ก็จะช่วยลดเวลาในการทำงานของโปรแกรมลงได้ (Vamsi)

4. อ่านข้อมูลด้วยคำสั่ง SELECT อย่างมีประสิทธิภาพ

4.1 การอ่านข้อมูลเพื่อทำการแสดงข้อมูลในรายงานอย่างมีประสิทธิภาพ ไม่ควรอ่านทุก Field หรือใช้คำสั่ง SELECT * เพราะจะทำให้มีการส่งผ่านข้อมูลจาก Database server มาที่ Application server ในปริมาณที่มาก ดังนั้นจึงควรเลือกเฉพาะ Field ที่ต้องการใช้ หรือมีอีกทางเลือกหนึ่งคือใช้การอ่านข้อมูลจาก Database View แทน Database table แต่ไม่ควร

ใช้การอ่านข้อมูลแล้วจัดกลุ่มเอง ควรใช้ Aggregate function มาช่วย (ประพจน์ สุขมานนท์, 2547, น.123; SAP Online Help; Vamsi; Agnihotri and Horacio) และลำดับของ Field ที่อ่านควรเรียงตามลำดับตาม ABAP Dictionary ส่วนการพักข้อมูลไว้ใน Intenal table ใช้คำสั่ง SELECT field1 field2 INTO TABLE ซึ่งลำดับของ Field ในการอ่านและลำดับของ Field ใน Internal Table ที่มารับข้อมูลจะต้องตรงกัน (Ghose)

4.2 อ่านข้อมูลจากฐานข้อมูลให้ได้ปริมาณน้อย โดยการอ่านข้อมูลควรระบุเงื่อนไขทั้งหมดไว้ที่ WHERE Clause โดยลำดับของ Field ที่ใช้เป็นเงื่อนไขควรเรียงตามลำดับของ Key หรือ Index ใน ABAP Dictionary และควรใช้ AND และการเปรียบเทียบด้วยเครื่องหมายเท่ากับ (=) เพราะการเปรียบเทียบด้วย NOT, OR, IN เครื่องหมายไม่เท่ากับ (<>) เครื่องหมายน้อยกว่า (<) เครื่องหมายมากกว่า (>) และ LIKE % จะไม่เข้าเงื่อนไขการค้นหาของ Index ข้อห้ามอีกข้อหนึ่งคือไม่ควรอ่านข้อมูลขนาดใหญ่และใช้คำสั่ง CHECK ในการตรวจสอบเงื่อนไข (SAP Online Help; Agnihotri & Horacio)

4.3 คำสั่ง SELECT ที่ซ้อนกัน (Nested SELECT) ควรใช้เฉพาะกับกรณีที่ชุดคำสั่ง SELECT ชุดนอกอ่านข้อมูลปริมาณน้อยเท่านั้น ซึ่งวิธีการที่จะหลีกเลี่ยงการใช้คำสั่ง SELECT ที่ซ้อนกัน ได้แก่ ใช้การ JOIN ใน FORM clause หรือจะใช้ View ที่เกิดจากการ Join กันซึ่งมีอยู่แล้วใน ABAP Dictionary หรือใช้สั่ง SELECT…FOR ALL ENTRIES

การใช้คำสั่ง JOIN ควรใช้เป็น INNER JOIN แต่ไม่ควรคำสั่ง JOIN ตารางมากกว่า 2 ตารางขึ้นไป และเงื่อนไขในการ JOIN ควรเป็นไปตาม Key ของตาราง ไม่เช่นนั้นควรเปลี่ยนไปใช้ FOR ALL ENTRIES

การที่จะใช้ประโยชน์ของคำสั่ง SELECT… FOR ALL ENTRIES ได้อย่างเต็มประสิทธิภาพ ควรทำการเรียงข้อมูลใน Internal table ต้นทางด้วย Column ที่จะนำมาใช้เป็น Key และทำให้ Key นั้นไม่มีค่าซ้ำกัน ก่อนทำการอ่านข้อมูลของตารางปลายทางจากฐานข้อมูล (SAP Online Help; Agnihotri & Horacio; Ghose)

4.4 การใช้คำสั่ง SELECT SINGLE ให้ใช้กับกรณีที่ต้องการอ่านข้อมูลเพียงรายการเดียวและสามารถระบุเงื่อนไขได้ครบตาม Key ของตารางในฐานข้อมูล หากทราบเงื่อนไขไม่ครบควรใช้คำสั่ง SELECT UP TO 1 ROW (Vamsi; Ghose)

4.5 ควรใช้คำสั่ง SELECT INTO TABLE แต่ถ้าหากข้อมูลที่ได้มาเป็นจำนวนมาก เช่น 500,000 รายการขึ้นไป ควรใช้ SELECT UP TO N ROWS เพื่อระบุจำนวนรายการ (Ghose)

4.6 หลีกเลี่ยงการใช้คำสั่ง SELECT…DISTINCT ควรใช้คำสั่ง DELETE ADJACENT DUPLICATES… (Agnihotri & Horacio)

5. ปรับปรุงฐานข้อมูลอย่างมีประสิทธิภาพ

เลือกการทำงานกับฐานข้อมูลให้น้อยครั้ง เมื่อต้องการ INSERT, UPDATE, MODIFY และ DELETE ให้ใช้การทำงานแบบทีละชุดข้อมูล แทนที่จะทำงานทีละรายการ เพราะวิธีนี้ช่วยให้มั่นใจได้ว่า รายการที่เพิ่ม แก้ไข หรือลบ เป็นรายการที่ซ้ำกัน และยังช่วยลดภาระงานในฝั่งฐานข้อมูลด้วย ส่วนการแก้ไขข้อมูลในฐานข้อมูล ยกตัวอย่าง เช่น การแก้ไขข้อมูลในฐานข้อมูลจาก Internal Table ถ้าหากต้องการแก้ไขเพียงบาง Field ควรใช้คำสั่ง MODIFY…TRANSPORTING f1…fn เป็นต้น และถ้าหากต้องการแก้ไขเฉพาะ Field ให้ใช้คำสั่ง UPDATE…SET วิธีนี้ช่วยลดเวลาการทำงานได้ดี (SAP Online Help) แต่อย่างไรก็ตาม ควรใช้คำสั่ง UPDATE หรือ INSERT แทนการใช้คำสั่ง MODIFY (Ghose)

6. คำสั่งที่ใช้จัดการกับตัวแปร Internal Table

6.1 คำสั่ง READ TABLE…WITH KEY… เป็นคำสั่งที่ใช้อ่านข้อมูลจากตัวแปร Internal table โดยอ่านข้อมูลขึ้นมาทีละหนึ่งรายการแล้วเปรียบเทียบกับเงื่อนไขไปจนกว่าจะตรงตามเงื่อนไขที่อ่าน (Sequential read) ซึ่งใช้เวลานานหากตัวแปร Internal table มีขนาดใหญ่ จึงควรใช้ READ TABLE…WITH KEY…BINARY SEARCH เพราะจะใช้วิธีการอ่านตามเงื่อนไขแบบ Binary search ก่อนจะใช้คำสั่งนี้ต้องทำการเรียงข้อมูลในตัวแปร Internal table นั้นก่อน สำหรับ SAP Version 4.0 ขึ้นไป สามารถเพิ่มประสิทธิภาพการทำงานกับตัวแปร Internal table ด้วยการกำหนดให้ตัวแปร Internal table นั้นเป็น SORTED TABLE หรือ HASHED TABLE โดยการอ่านข้อมูลจากตัวแปร Internal table ที่เป็น SORTED TABLE จะทำการอ่านด้วยเงื่อนไขที่ระบุไว้แบบ Binary search ให้เลย โดยไม่ต้องใช้คำสั่ง BINARY SEARCH ส่วนตัวแปร Internal table ที่เป็น HASHED TABLE ก็เช่นกัน แต่มีข้อแม้ว่าจะต้องระบุเงื่อนไขการอ่านให้ครบทุก Key ของ Internal table ไม่เช่นนั้นแล้ว ก็จะเป็นเหมือนกับการอ่านแบบ Sequential read ตามปกติ (Schneider, 2006, p.181-182; Ghose)

ถ้าข้อมูลใน Internal table มีโอกาสซ้ำกันได้ ให้ใช้การอ่านแบบ Binary search ก่อนในครั้งแรก และเก็บตำแหน่งของรายการที่อ่านพบ (SY-TABIX) แล้วจึงเริ่มทำงานจากตำแหน่งนั้นเป็นต้นไปด้วยคำสั่ง LOOP แต่ถ้าข้อมูลใน Internal table มีปริมาณไม่มาก เช่น มีจำนวนอยู่ที่หลักร้อยรายการ จะใช้คำสั่ง COLLECT และ APPEND ก็ย่อมได้ การเลือกใช้ประเภท

ของ Internal table ก็ช่วยในเรื่องของประสิทธิภาพการทำงานได้เช่นกัน แต่ต้องทราบข้อกำหนดของแต่ละประเภทด้วยว่าใช้งานอย่างไรจึงจะเหมาะสม (G. Propfe, 2001, p.213)

6.2 ให้ใช้ตัวแปร Field Symbol แทนการใช้ Header Line ในการอ่านข้อมูลในคำสั่ง LOOP ที่ตัวแปร Internal Table เพราะเป็นการทำงานที่เนื้อข้อมูลโดยตรง โดยไม่มีการคัดลอกข้อมูลจากเนื้อข้อมูลไปที่ตัวแปร Header Line (ประพจน์ สุขมานนท์, 2547, น.123-124; Agnihotri and Horacio) ดังตัวอย่าง

```
FIELD-SYMBOLS <wa> TYPE tab.
LOOP AT tab ASSIGNING <wa>.
WRITE: / <wa>-id, <wa>-name, <wa>-city.
ENDLOOP.
```

6.3 การใช้คำสั่ง LOOP ซ้อนกัน (Nested Loop)

คำสั่ง LOOP ซ้อนกัน มักจะใช้เพื่อต้องการประมวลผลข้อมูลในตัวแปร Internal Table 2 ตารางที่ขึ้นต่อกัน เช่น

```
LOOP AT HEADER INTO WA_HEADER.
LOOP AT POSITION INTO WA_POSTION WHERE KEY = WA_HEADER-KEY.
“Processing …
ENDLOOP.
ENDLOOP.
```

ในแต่ละรายการของตาราง HEADER จะทำการอ่านทุกรายการในตาราง POSITION ตามเงื่อนไขที่ระบุไว้ใน WHERE clause ขึ้นมาประมวลผลทีละรายการ ถ้าทั้งสองตารางมีขนาดใหญ่ ชุดคำสั่งนี้จะกลายเป็นสาเหตุให้ใช้เวลามากในการทำงาน ถึงแม้ว่าจะเปลี่ยนมาเป็น SORTED TABLE เพื่อทำให้การอ่านรายการในตาราง POSITION ใช้วิธีการ Binary Search แต่ก็สามารถใช้ได้เพียง Key เดียว และยังต้องระบุ Key นั้นไว้ที่ลำดับแรกของ WHERE Clause ด้วย

ทางเลือกอีกทางหนึ่ง คือ สามารถใช้การระบุ Index เข้ามาช่วยลดเวลาในการทำงานได้ โดยต้องทำการเรียงข้อมูลในตาราง HEADER และ POSITION ด้วย Key ที่สัมพันธ์กัน และใช้ชุดคำสั่งดังนี้

```
I = 1.
LOOP AT HEADER INTO WA_HEADER.
LOOP AT POSITION INTO WA_POSITION FROM I.
IF WA_POSITION-KEY <> WA_HEADER-KEY.
I = SY-TABIX. EXIT.
ENDIF. “Processing…
ENDLOOP.
ENDLOOP.
```

6.4 การใช้คำสั่งกำหนดค่าเริ่มต้นของตัวแปร Internal Table ควรใช้คำสั่ง OCCURS n ควรปล่อยให้เป็นหน้าที่ของระบบโดยให้ n = 0 การกำหนดตัวเลขให้กับ n > 0 จะใช้ในกรณีที่ทราบจำนวนรายการของข้อมูลที่แน่นอนที่จะอ่านได้ เพราะถ้าหากกำหนดตัวเลขที่แน่นอนลงไปแล้ว แต่ข้อมูลที่อ่านได้เกินจำนวนที่จองเอาไว้ การจองครั้งที่ 2 และครั้งต่อๆไปจะจองเท่ากับตัวเลขที่กำหนดในครั้งแรก เช่น กำหนดว่า OCCURS 10 แต่รายการที่อ่านได้มี 15 รายการ การจองเนื้อที่ครั้งที่ 2 จะยังคงจองอีก 10 แต่ก็ใช้เพียงแค่ 5 รายการที่เกินมา ทำให้มีเนื้อที่ส่วนที่ไม่ได้ใช้งานซึ่งจองไว้เกินจำเป็น ต่างกับการกำหนด OCCURS 0 ซึ่งระบบจะจองแบบยืดหยุ่นตามจำนวนรายการของข้อมูลที่อ่านได้จริง (SAP Online Help)

6.5 การจัดการเกี่ยวกับ Index ของ Internal table เมื่อใช้คำสั่ง INSERT, DELETE หรือ SORT กับตัวแปร Internal table ที่มีขนาดใหญ่ สามารถทำให้ใช้เวลาในการทำงานเพิ่มขึ้นได้ เพราะทำให้เกิดการเปลี่ยนแปลง Index ของ Internal table ดังนั้นการเพิ่มข้อมูลเข้าไปในตัวแปร Internal table ควรใช้คำสั่ง APPEND แทนคำสั่ง INSERT เพื่อเพิ่มข้อมูลทีละรายการ หรือถ้าต้องการเพิ่มข้อมูลทีละชุดให้ใช้คำสั่ง INSERT LINES OF itab1 [FROM i] [TO j] INTO itab2 [INDEX k]. หรือ APPEND LINES OF itab1 [FROM i] [TO j] TO itab2. (SAP Online Help)

6.5 การคัดลอกข้อมูลในกรณีที่ Internal Table มีโครงสร้างเหมือนกัน ให้ใช้วิธีการคัดลอกทั้งตาราง ด้วยคำสั่ง TAB_DESTINATION[] = TAB_SOURCE[]. (Ghose)

6.6 การแก้ไขค่าในตัวแปร Internal Table ด้วยค่าและเงื่อนไขเดียวกัน ให้ใช้คำสั่ง MODIFY…TRANSPORTING… สำหรับการแก้ไขข้อมูลครั้งละ 1 รายการ และให้ใช้คำสั่ง MODIFY…TRANSPORT…WHERE…สำหรับการแก้ไขข้อมูลคราวละมากกว่า 1 รายการ (Ghose)

6.7 การลบรายการจากตัวแปร Internal Table สามารถทำได้หลายแบบ ได้แก่ การลบแบบระบุ Index ด้วยคำสั่ง DELETE…INDEX การลบมากกว่า 1 รายการด้วยเงื่อนไขเดียวกัน ด้วยคำสั่ง DELETE…WHERE

6.8 การเปรียบเทียบจำนวนรายการของ Internal Table 2 ตัวแปร ให้ใช้คำสั่ง CHECK itab1[] = itab2[]. หรือ IF itab1[] = itab2[] … ENDIF.

6.9 การย้ายค่าจากตัวแปร Internal Table หนึ่งไปยังอีก Internal Table หนึ่ง ซึ่งทั้งสอง Internal Table มีโครงสร้างเหมือนกัน ให้ใช้คำสั่ง APPEND LINE OF itab1 to itab2.

7. คำสั่งอื่นๆ

7.1 ตัวแปร Internal Table, Work Area และตัวแปร Global ทั้งหลาย ควรใช้คำสั่ง FREE หรอื REFRESH เมื่อไม่มีการใช้งานแล้ว โดยเฉพาะอย่างยิ่งการโปรแกรมที่มีการอ่านข้อมูลปริมาณมากมาไว้ที่ตัวแปร Internal Table หรือโปรแกรมที่มีลักษณะเป็น Batch Input Program ถ้าหากไม่มีการใช้คำสั่ง REFRESH หรือคำสั่ง FREE ทำให้ Memory ยังคงถูกจองใช้งานอยู่ถึงแม้ว่าไม่จำเป็นต้องใช้แล้วก็ตาม (Schneider, 2006, p.181; Ghose) และควรใช้คำสั่ง CLEAR กับ Field หรือ Structure ทุกครั้งก่อนใช้งาน (G. Propfe, 2001, p.213)

7.2 พยายามประกาศใช้ตัวแปรเท่าที่จำเป็น เพื่อเป็นการประหยัดการใช้งานส่วน Memory (Agnihotri and Horaci) ลดการใช้ตัวแปรที่เป็น Global variable ควรใช้ตัวแปรที่เป็น Local variable ซึ่งประกาศใช้เฉพาะใน Subroutine (Aveek) และใช้หลักการตัวแปร Formal Parameter เพื่อส่งผ่านข้อมูลจากตัวแปร Local ของ Subroutine หนึ่งไปยังอีก Subroutine หนึ่ง (G. Propfe, 2001, p.213)

7.3 หลีกเลี่ยงการใช้คำสั่ง INTO CORRESPONDING FIELDS (Agnihotri and Horacio)

7.4 หลีกเลี่ยงการใช้คำสั่ง COLLECT (Agnihotri and Horacio)

7.5 ควรใช้คำสั่งตรวจสอบ Return code (SY-SUBRC) ว่าทำงานสำเร็จหรือไม่ก่อนที่จะทำขั้นตอนต่อไป (G. Propfe, 2001, p.213)

7.6 ระวังการใช้ชื่อตัวแปรที่ทับซ้อนกัน (G. Propfe, 2001, p.213) เช่น

```
DATA: ABC(10) TYPE C.
FORM TEST. DATA: ABC(5) TYPE N. ENDFORM.
```

7.7 ระวังการ Loop ที่ไม่รู้จบ เพราะ ไม่เข้าเงื่อนไขของการจบ Loop (G. Propfe, 2001, p.213)

7.8 หลีกเลี่ยงการมี ชุดคำสั่งที่ไม่ได้ถูกเรียกใช้ (Dead code) อยู่ในโปรแกรม โดยใช้เครื่องมือ Extended Program Check เพื่อตรวจจับชุดคำสั่ง ตัวแปร และ Subroutine ที่มีประกาศไว้แต่ไม่ได้ใช้งาน และส่วนต่างๆ ที่อาจทำให้โปรแกรมมีข้อผิดพลาด เพื่อนำสิ่งเหล่านี้ออกจากโปรแกรม (Ghose)

7.9 การใช้คำสั่ง SORT ควรระบุแบบชัดแจ้งว่าเป็นการเรียงข้อมูลจากน้อยไปมาก (SORT…ASCENDING) หรือมากไปน้อย (SORT…DESCENDING) และไม่ควรใช้คำสั่ง SORT ในชุดคำสั่ง LOOP (Ghose)

7.10 การประกาศตัวแปรควรใช้คำสั่ง TYPE แทน LIKE (Ghose)

7.11 ควรใช้คำสั่ง CASE กับการเปรียบเทียบเงื่อนไขที่มากว่า 1 เงื่อนไข แต่เป็นการเปรียบเทียบแบบเท่ากับเท่านั้น (Ghose)

7.12 ไม่ควรใช้คำสั่ง SELECT ภายใต้ Event GET (Ghose)

7.13 การประกาศตัวแปร Internal Table ใช้คำสั่ง TYPE STANDARD TABLE OF และไม่ควรประกาศให้มี Header Line ให้ใช้ตัวแปร Work Area แทน (Ghose)

7.14 สำหรับโปรแกรม Smartform และ SAPscript ไม่ควรมีการอ่านข้อมูลที่ซ้ำซ้อนกับข้อมูล Interface ของ Form

### 2.5.2 เทคนิคการพัฒนาโปรแกรมด้วยหลักการ Running Task Parallel

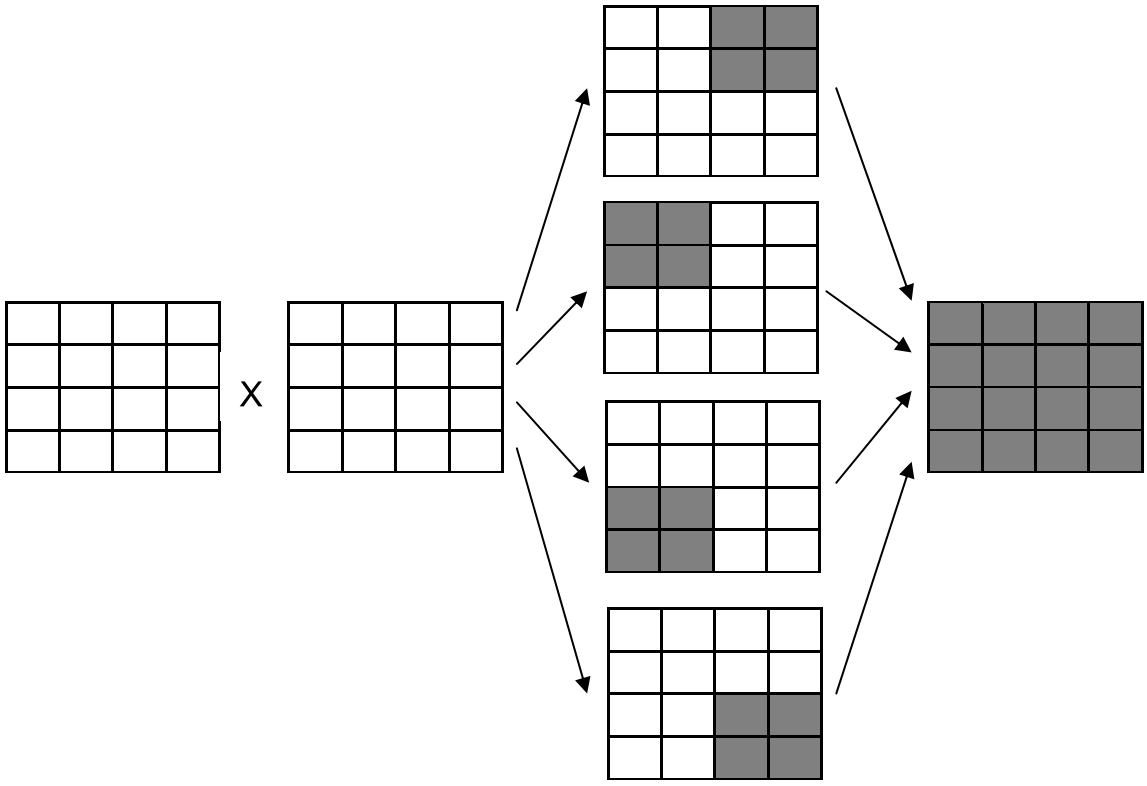
เทคนิคหรือวิธีการอีกอันหนึ่งที่สามารถนำมาช่วยได้ คือ Running Task Parallel ดังภาพที่ 2.17 กระบวนการบางอย่างที่สามารถแตกเป็นส่วนย่อยได้ โดยส่วนย่อยเหล่านั้นต้องไม่ขึ้นต่อกัน และนำส่วนย่อยเหล่านั้นมาปล่อยให้ทำงานไปพร้อมๆกัน เพื่อช่วยปรับปรุงประสิทธิภาพการทำงานของโปรแกรม ในการทำงานจริง เทคนิคนี้ไม่ค่อยนำมาใช้กันมากนัก เนื่องมาจากขาดความเข้าใจที่แท้จริง และมีความยุ่งยากในการเขียนโปรแกรมและการบำรุงรักษา แต่ถ้าหากใช้เครื่อง Server ที่ใช้เป็นแบบ Multi-Processor เทคนิคนี้จะเป็นประโยชน์มาก หลักการทำงานของเทคนิคนี้ คือ ปล่อยให้ส่วนย่อยทำงานไปพร้อมๆกันโดยทำงานผ่าน Function Module ที่เรียกใช้แบบ Asynchronous และสร้างเป็นงานย่อยอันใหม่ ซึ่งแต่ละงานย่อยจะต้องมีที่ชื่อต่างกัน หลังจากงานย่อยถูกปล่อยไปทำงานแล้ว โปรแกรมหลักที่เป็นตัวควบคุมจะต้องรอเก็บรวบรวมผลการทำงานของงานย่อยเหล่านั้น และทำงานต่อไป ข้อควรระวังของเทคนิคนี้คือ งานที่แตกย่อย

ออกมาจะต้องเป็นอิสระจากส่วนอื่นอย่างแท้จริง ไม่เช่นนั้นปัญหาเรื่อง Enqueue หรือ Deadlock อาจเกิดขึ้นได้

ตัวอย่าง Application ที่สามารถนำเทคนิคนี้มาประยุกต์ใช้ได้ ได้แก่ การสร้าง BDC ที่ไม่มีการ Post, อ่านข้อมูลจากตารางต่างกัน เช่น VBAK และ VBAP, การคำนวณที่ซับซ้อนของแต่ละ Characteristic (QM Module), การคำนวณ ASTM ของชุดตัวแปรที่ต่างกัน (IS-OIL Module) และ การคำนวณ Repricing Simulation สำหรับเอกสารที่ต่างกัน (SD and MM Module) (G. Propfe, 2001, p.222-223)

ภาพที่ 2.17

การทำงาน Running Task Parallel




ที่มา: “ABAP with Style” by G. Propfe, 2001, p. 223

## 2.6 การบำรุงรักษาซอฟต์แวร์

การบำรุงรักษาซอฟต์แวร์ (Software maintenance) คือ การเปลี่ยนแลงแก้ไขระบบซอฟต์แวร์หลังจากการส่งมอบงาน เพื่อต้องการแก้ไขข้อผิดพลาด ปรังปรุงประสิทธิภาพ หรือดัดแปลงให้เหมาะสมกับสิ่งแวดล้อมที่เปลี่ยนไป (ANSI/IEEE, 1983) การบำรุงรักษาซอฟต์แวร์เป็นเรื่องที่ทำได้ยาก และยากต่อการจัดการอย่างประสิทธิภาพ ปัญหาที่เกิดขึ้นสำหรับการบำรุงรักษาระบบ เป็นผลมาจากการที่ใช้วิธีการในการพัฒนาซอฟต์แวร์อย่างไม่เหมาะสม (Schneidewind, 1987) ดังนั้นจึงควรเริ่มปรับปรุงการพัฒนาซอฟต์แวร์ตั้งแต่ขั้นตอนแรกๆ กล่าวคือ การออกแบบ หรือตอนที่ซอฟต์แวร์ยังอยู่ในกระบวนการวิศวกรรมย้อนกลับ (Re-engineering) เพื่อส่งผลต่อการบำรุงรักษาซอฟต์แวร์ที่ได้ผลดีในระยะยาว ความซับซ้อนของซอฟแวร์ หมายถึง ลักษณะของโครงสร้างข้อมูลและการทำงานภายในซอฟต์แวร์ที่ยากต่อการทำความเข้าใจและปรับปรุงแก้ไข ถือเป็นปัจจัยสำคัญที่ส่งผลต่อการบำรุงรักษาซอฟต์แวร์ (Banker, Davis, & Slaughter, 1998)

การปรับปรุงประสิทธิภาพจัดเป็นส่วนหนึ่งของการบำรุงรักษาระบบ บริษัท SAS System เป็นอีกองค์กรหนึ่งซึ่งประสบกับปัญหาประสิทธิภาพ สิ่งที่มีผลทำให้เกิดปัญหาประสิทธิภาพของระบบ ได้แก่ แอพพลิเคชันโค้ด (Application Code), แบบจำลองข้อมูล (Data Models), และการออกแบบกระบวนการ (Process Design) ซึ่งสิ่งเหล่านี้ส่งผลต่อการทำงานของสถาปัตยกรรมโครงข่าย (Network Architecture) และโครงแบบ (Configuration) ปัญหาที่เกิดขึ้นได้แก่ (1) การทำงานของงานหนึ่งๆ ไม่สามารถประมวลผลสำเร็จภายในเวลาที่กำหนด (2)งานหนึ่งๆใช้ทรัพยากรปริมาณมากในการประมวลผลแต่ไม่สามารถประมวลผลสำเร็จได้ในเวลาที่กำหนด (3) การทำงานของระบบล้มเหลวบ่อยครั้งหลังจากมีการใช้งานมากเกินไป และ (4) ระบบไม่สามารถรองรับกับปริมาณการใช้งานตามที่คาดหวังหรือไม่คาดหวัง โดยปราศจากผลของปัญหาที่กล่าวมาข้างต้นได้ ทางบริษัทจึงได้มีกระบวนการที่จะระบุสาเหตุของปัญหาและแก้ไขปัญหาอย่างมีประสิทธิภาพไว้ดังนี้ (Customer Technology Center, 2001)

1. กำหนดและบันทึกคำร้องเกี่ยวกับประสิทธิภาพการทำงานของระบบ
2. ระบุสาเหตุของปัญหาโดย
    2.1 วัดประสิทธิภาพของระบบในส่วนของ SAS
    2.2 ทำความเข้าใจกับโครงแบบของระบบที่เกิดปัญหา
    2.3 วัดประสิทธิภาพของระบบในระดับ Server

2.4 วัดประสิทธิภาพของระบบในระดับระบบ

3. แก้ไขปัญหาตามสาเหตุที่ระบุได้

3.1 ปรับ Server หรือโครงข่าย

3.2 ปรับแอพพลิเคชัน กระบวนการ และการจัดการข้อมูล

3.3 พิจารณาประโยชน์ที่เกิดจากการจัดการปริมาณงาน

3.4 เพิ่มเติมทรัพยากรของระบบ