บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

สำหรับในบทที่ 2 จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับวิทยานิพนธ์ คือระบบเฮลป์เดสก์ วงจรการพัฒนาซอฟต์แวร์แบบเรชันแนลออปเจกซ์ทอรีโพรเซส แนวคิดการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุ ภาษาโมเดลยูนิฟายต์ การออกแบบสถาปัตยกรรม 3 เทียร์ และ วัตถุเพอซีซเทนท์ และ เฟรมเวอร์ค

2.1 ระบบเฮลป์เดสก์ (Helpdesk System)

ในส่วนของระบบเฮลป์เดสก์นี้จะกล่าวถึงหน้าที่ ระบบเฮลป์เดสก์ ระดับการแก้ไขปัญหา คุณลักษณะ และ วิธีการจัดระดับความสำคัญของปัญหา ดังนี้

2.1.1 หน้าที่ระบบเฮลป์เดสก์

ระบบเฮลป์เดสก์ของหน่วยงานต่างๆนั้นจะแตกต่างกันไป ขึ้นกับขอบเขตของการบริการของศูนย์คอมพิวเตอร์ต่าง ๆ แต่โดยทั่วไปจะมีการกำหนดบริการดังนี้

1) ตอบปัญหาและข้อสงสัย (Answering Questions) ระบบเฮลป์เดสก์จะมีหน้าที่ในการตอบปัญหาในด้านเทคนิคต่างๆให้แก่ผู้ใช้ซึ่งจะมีการจำกัดขอบเขตการให้บริการเฉพาะคำถามที่มีผลต่อการทำงาน โดยจะครอบคลุมอุปกรณ์และซอฟท์แวร์ที่อยู่ภายใต้ความรับผิดชอบของหน่วยงานเฮลป์เดสก์
2) บริการตามคำร้องขอ (Service a Request) ในกรณีนี้ผู้ใช้สามารถร้องขอให้เฮลป์เดสก์ดำเนินการบางอย่างได้ เช่น ปรับปรุงซอฟต์แวร์ให้เป็นรุ่นปัจจุบัน หรือขอย้ายอุปกรณ์ ขอซื้ออุปกรณ์ และ ขอข้อกำหนดมาตรฐานอุปกรณ์ เป็นต้น
3) การดำเนินงานกรณีเร่งด่วน (Handling an Emergency) ในกรณีที่ผู้ใช้มีงานเร่งด่วนต้องการความช่วยเหลือ หรือคำแนะนำ จะสามารถขอความช่วยเหลือจากเฮลป์เดสก์ได้ โดยที่เฮลป์เดสก์เองจะกำหนดขั้นตอน และ วิธีการในการดูแลปัญหาเร่งด่วน
4) การแจ้งปัญหาระบบแก่ผู้ใช้ระบบ (Informing Customer of System Problems) ในกรณีที่ระบบเกิดปัญหาหรือต้องทำการปิดระบบเพื่อซ่อมบำรุง เฮลป์เดสก์จะทำการแจ้งแก่ผู้ใช้ถึงปัญหาดังกล่าว และ วิธีการแก้ไขรวมทั้งเวลาที่คาดว่าจะสามารถเปิดบริการได้ใหม่
5) ระบบรายงาน (Reporting) เฮลป์เดสก์จะมีการเก็บรวบรวมข้อมูลการดำเนินงานเพื่อจัดทำรายงานแก่ผู้ใช้และผู้ดูแลระบบ

6) วิธีการฟื้นสภาพกรณีระบบเกิดข้อผิดพลาด (Disaster Recovery) กรณีที่ระบบเกิดปัญหาร้ายแรงหรือเกิดความเสียหาย ควรจะมีการกำหนดวิธีการในการแก้ไขปัญหาและการฟื้นสภาพ เช่น กรณีหน่วยความจำสำรองเสียหาย ไฟฟ้าดับ หรือ ระบบเครือข่ายล้มเหลว
7) การติดต่อระหว่างระบบเฮลป์เดสก์อื่น ๆ (Commuicating with other Help Desks) กรณีที่ในหน่วยงานมีระบบเฮลป์เดสก์มากกว่า 1 ระบบ หรือติดต่อกับระบบเฮลป์เดสก์ของผู้ขาย หรือมีการซื้อบริการเฮลป์เดสก์บางส่วนจากภายนอก ก็ควรจะมีการกำหนดวิธีการ ในการติดต่อสื่อสารข้อมูลที่ต้องการ และ การรายงานผลการแก้ไขปัญหา
8) ขั้นตอนภายใน (Internal Process) ในระบบเฮลป์เดสก์เองก็ควรจะมีการกำหนดขั้นตอนภายในหน่วยงานไว้เป็นมาตรฐาน เช่น วิธีการส่งต่อปัญหา การส่งเวรการทำงาน การสรุปการดำเนินงานประจำวัน และ ประจำเดือน เป็นต้น

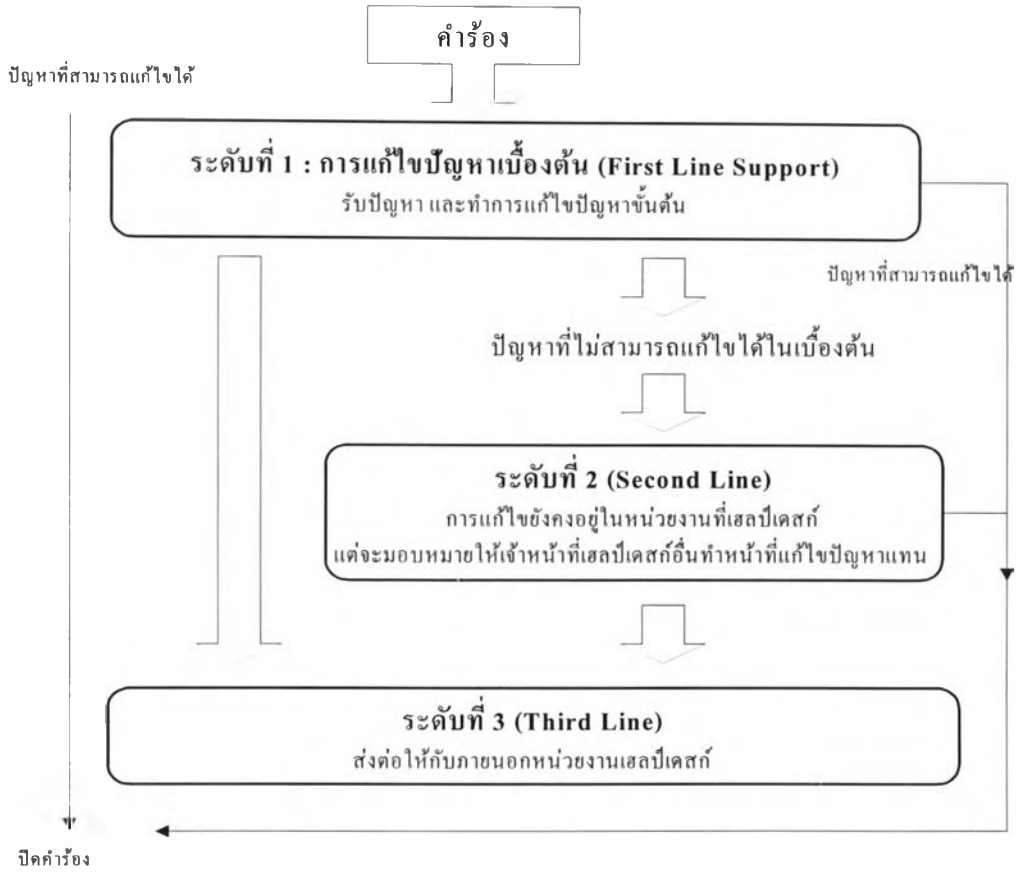
2.1.2 ระดับการแก้ไขปัญหาของระบบเฮลป์เดสก์ [1]

ด้วยสาเหตุข้างต้น หลายองค์กรและหน่วยงานจึงได้มีการจัดตั้งหน่วยงานเฮลป์เดสก์เพื่อทำหน้าที่แก้ไขปัญหาแก่ผู้ใช้ โดยมีขั้นตอนดังรูปที่ 2.1 โดยระบบการแก้ไขปัญหาของผู้ใช้นั้นจะมีอยู่ 3 ระดับคือ

ระดับที่ 1 (First Line Support)

สำหรับการให้บริการในระดับแรก จะทำหน้าที่เป็นหน่วยแรกที่ทำหน้าที่รับปัญหาที่ผู้ใช้โทรศัพท์เข้ามาปรึกษาปัญหาต่างๆ (อยู่ในบริการที่กำหนดไว้ในบริการของศูนย์คอมพิวเตอร์) หากพนักงานสามารถแก้ไขได้จะทำการแก้ไขและปิดคำร้อง หากปัญหาดังกล่าวสามารถแก้ไขได้โดยหน่วยงานเฮลป์เดสก์แต่ต้องใช้เวลามากจะทำการ ส่งต่อไปยังระดับที่ 2 หรือหากกรณีที่คำร้องนั้นเกินขีดความสามารถของพนักงานเฮลป์เดสก์ระดับแรก จะทำการส่งต่อปัญหาไปยังกลุ่มบุคลากรอื่นตามชนิดของปัญหา เช่น ปัญหาด้านข้อผิดพลาดของ ระบบปฏิบัติการจะส่งต่อให้กับวิศวกรระบบ ปัญหาด้านฐานข้อมูลจะส่งต่อให้กับนักบริหารฐานข้อมูล ปัญหาเครื่องคอมพิวเตอร์เกิดความเสียหาย จะส่งต่อให้กับผู้ขาย โดยสามารถจำแนกเฮลป์เดสก์ในระดับที่ 1 ออกเป็น 2 ประเภท คือ

1) ดีสแพทช์ฟรอนท์ไลน์ (Dispatch Front Line) ทำการรับปัญหาแล้วทำการส่งต่อให้ระดับที่ 2 โดยไม่มีการแก้ไขใด ๆ เลย ซึ่งเหมาะกับองค์กรขนาดกลางและขนาดใหญ่ เพราะหากฟรอนท์ไลน์ทำการแก้ไขปัญหาจะทำให้อัตราการโทรศัพท์ติดต่อได้ลดลง หรือทำให้จำนวนปัญหาที่แก้ไขลดลง
2) รีโซว์ฟรอนท์ไลน์ (Resolve Front Line) ทำการรับปัญหาแล้วทำการแก้ไขเบื้องต้น ซึ่งเหมาะกับหน่วยงานหรือองค์กรขนาดเล็กและขนาดกลาง



รูปที่ 2.1 แสดงขั้นตอนของระดับการบริการเฮลป์เดสก์ (Help Desk Service Level) [1]

ระดับที่ 2 (Second Line/Second Level Support)

ทำหน้าที่แก้ไขปัญหาต่าง ๆ ที่เฮลป์เดสก์ระดับแรกไม่สามารถแก้ไขได้ หรือปัญหา ที่ต้องใช้ เวลาในการแก้ไขนาน ซึ่งหากให้เฮลป์เดสก์ระดับที่ 1 ทำการแก้ไขจะส่งผลให้ประสิทธิภาพโดยรวมลดลง จึงส่งให้กับระดับที่ 2 ซึ่งบางหน่วยงานจะกำหนดให้ระดับที่ 2 เป็นกลุ่มบุคลากรผู้เชี่ยวชาญเฉพาะด้านของศูนย์คอมพิวเตอร์ ซึ่งหากระดับที่ 1 ไม่สามารถแก้ไขได้ก็จะทำการส่งต่อให้ระดับที่ 2 ที่เป็นกลุ่มผู้เชี่ยวชาญ

ระดับที่ 3 (Third Line/Third Level Support)

ทำหน้าที่รับปัญหาที่ไม่สามารถแก้ไขได้ในระดับที่ 2 โดยมากจะเป็นกลุ่มผู้เชี่ยวชาญของบริษัทผู้ขาย ทำหน้าที่รับช่วงคำร้องที่ไม่สามารถแก้ไขได้จากระดับที่ 2

2.1.3 คุณลักษณะของระบบเฮลป์เดสก์ [1]

การทำงานของระบบเฮลป์เดสก์สามารถจำแนกออกเป็น 5 ส่วนคือ ข้อมูลพื้นฐาน การเก็บข้อมูลกรณีปัญหาและคำร้องขอบริการ การแก้ไขและการส่งต่อปัญหา ส่วนงานบริหารอุปกรณ์คอมพิวเตอร์ และ การวิเคราะห์ผลการดำเนินงานระบบเฮลป์เดสก์ ดังรายละเอียดต่อไปนี้

2.1.3.1 ข้อมูลพื้นฐาน (HelpDesk Basic Information)

ข้อมูลพื้นฐาน เป็นส่วนที่สำคัญที่สุดของระบบ โดยจะเป็นส่วนจัดเตรียมข้อมูลพื้นฐานสำหรับการทำงานระบบเฮลป์เดสก์ โดยจะประกอบไปด้วย

ก. ข้อมูลนโยบาย อันประกอบไปด้วยบริการของเฮลป์เดสก์ และการจัดลำดับความสำคัญของงาน (Priority)

ข. ข้อมูลผู้ใช้ระบบ (User Information) ประกอบไปด้วยชื่อ และ หน่วยงานของผู้ใช้ หมายเลขโทรศัพท์ ข้อมูลอุปกรณ์ และ ซอฟต์แวร์ที่ใช้งาน เป็นต้น

ค. ข้อมูลเจ้าหน้าที่เฮลป์เดสก์และผู้เชี่ยวชาญ (HelpDesk Staff Information) ข้อมูลชื่อ และ ความชำนาญ

ง. ค่าใช้จ่ายต่าง ๆ ของระบบงานเฮลป์เดสก์ อาทิ ค่าแรงล่วงเวลา เงินเดือน ค่าซอฟต์แวร์ และ ค่าจ้างหน่วยงานภายนอก เป็นต้น

2.1.3.2 การเก็บข้อมูลกรณีปัญหาและคำร้องของผู้ใช้บริการ (Call and Request Logging)

ทำการเก็บปัญหาและคำร้องขอบริการต่าง ๆ ที่ผู้ใช้โทรเข้ามา (รวมคำร้องของหน่วยงานเฮลป์เดสก์) ตามขั้นตอนดังนี้

ก. ทักทายผู้ใช้และแนะนำตัว

ข. เก็บข้อมูลปัญหา อันประกอบไปด้วย ข้อมูลผู้ใช้ รายละเอียดปัญหา วัน-เวลาที่แจ้ง และการแก้ปัญหาเบื้องต้นโดยผู้ใช้

ค. กำหนดระดับความสำคัญของปัญหาหรือคำร้องขอบริการอื่น ๆ

ง. ส่งต่อปัญหา (กรณีของดีสแพทซ์เฮลป์เดสก์) ให้กับผู้เชี่ยวชาญ หรือ เจ้าหน้าที่เฮลป์เดสก์อื่น

2.1.3.3 งานการแก้ไขและการส่งต่อปัญหา (Problems Resolution and Escalation)

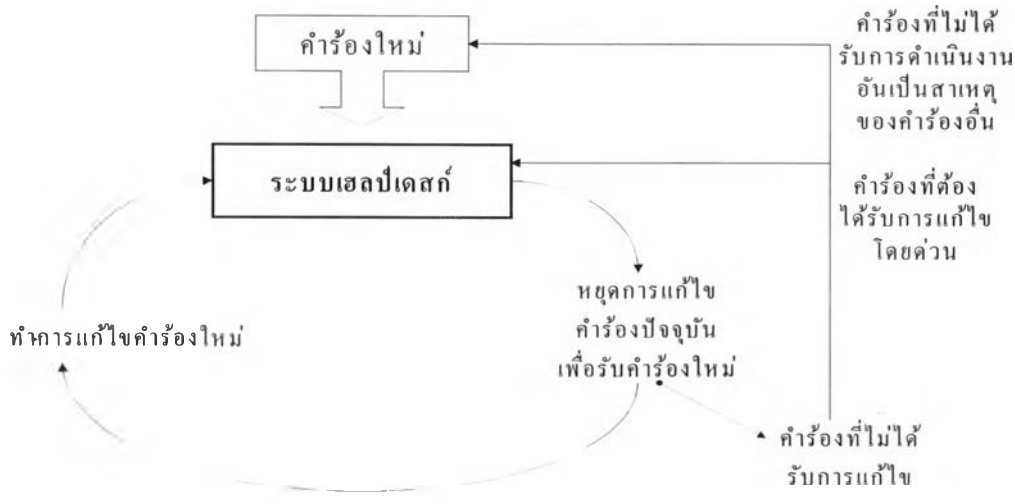
เป็นการทำการแก้ไขปัญหาโดยทำการค้นประวัติปัญหาเดิม เพื่อเป็นแนวทางในการแก้ไขปัญหาดังขั้นตอนต่างๆ ดังนี้

ก. รับปัญหาและทำการตรวจสอบข้อมูลทั้งหมด

ข. ทำการประเมินปัญหา ว่าอยู่ในขอบเขตที่จะสามารถแก้ไขได้หรือไม่ หากไม่สามารถแก้ไขได้ก็ทำการส่งต่อ หรืออาจจะสอบถามข้อมูลจากผู้เชี่ยวชาญอื่น ๆ ในระดับที่ 2 และ 3 ต่อไป

ค. ทำการแก้ไขปัญหาและบันทึกผลการแก้ไข ทั้งนี้ขึ้นอยู่กับปัญหาที่เกิดขึ้นซึ่งควรจะมีการกำหนดวิธีการ (procedure) ในบริการประเภทต่าง ๆ ด้วย

ง. ปิดปัญหา



รูปที่ 2.2 แสดงวงจรของปัญหาที่ไม่ได้รับการแก้ไขอย่างเหมาะสม[1]

ในการจัดการแก้ไขปัญหานั้นมีความสำคัญมากหากไม่มีการจัดการที่ดีจะทำให้เกิดปัญหาขึ้นหรือหากไม่สามารถแก้ไขปัญหาได้รวดเร็วพอ อาจจะทำให้เกิดการสะสมของปัญหาแล้วกลับมาเป็นปัญหาใหม่อีก ดังรูปที่ 2.2

2.1.3.4 งานคลังอุปกรณ์ (Equipment Asset Management)

เป็นส่วนเก็บข้อมูลอุปกรณ์คอมพิวเตอร์ของผู้ใช้ ประวัติการจัดซื้อ การซ่อมแซม การรับประกัน การบำรุงรักษา และข้อมูลผู้ขายต่าง ๆ โดยงานในส่วนนี้มิได้เป็นงานหลักของระบบเฮลป์เดสก์ แต่เป็นส่วนที่ช่วยให้ระบบงานเฮลป์เดสก์ทำงานได้สะดวกขึ้น โดยข้อมูลประวัติการซ่อมบำรุง จะช่วยให้เจ้าหน้าที่เฮลป์เดสก์สามารถประเมินปัญหา และ ค้นหาสาเหตุได้ง่ายขึ้น

2.1.3.5 งานการวิเคราะห์ผลการดำเนินงานของระบบเฮลป์เดสก์ (HelpDesk Performance Analysis) [1]

เป็นส่วนวิเคราะห์ผลการทำงานของระบบเฮลป์เดสก์ โดยเริ่มจากการเก็บข้อมูลต่าง ๆ แล้วนำข้อมูลดังกล่าวมาเปรียบเทียบกับข้อตกลงระดับการให้บริการเฮลป์เดสก์ ดังนี้

3.1.3.5.1 ผลตอบแทนการลงทุน (Return of Investment :ROI)

ผลตอบแทน (Return) เป็นการประเมินผลจากการดำเนินงานของระบบเฮลป์เดสก์ โดยสามารถคิดได้โดยคำนวณจากค่าบริการทั้งหมดที่เฮลป์เดสก์ทำหน้าที่บริการได้ ดังนี้

ผลตอบแทน (R) = T x S ;

โดย T คือ เวลาที่ใช้ในการแก้ไขปัญหา (หน่วยเป็นชั่วโมง) และ S คืออัตราค่าแรงต่อชั่วโมงซึ่งสามารถคำนวณได้จาก

S = SS / TT ;

โดย SS คือผลบวกของเงินเดือนของเจ้าหน้าที่เฮลป์เดสก์ทั้งหมด และ TT คือเวลาทั้งหมด ที่เจ้าหน้าที่เฮลป์เดสก์ปฏิบัติงานในระยะเวลา 1 เดือน

ทั้งนี้ข้อมูลที่ได้ จะนำไปเปรียบเทียบกับค่าใช้จ่ายในการว่าจ้างบริษัท หรือหน่วยงานภายนอกในการรับเหมาช่วงงาน หากค่าใช้จ่ายในการรับเหมาช่วงงานน้อยกว่าและผลการประเมินในประเด็นอื่น ๆ ไม่อยู่ในระดับพึงพอใจ ผู้บริหารอาจจะตัดสินใจให้บริษัทหรือหน่วยงานภายนอกมารับเหมาช่วงระบบงานเฮลป์เดสก์แทน

การลงทุน (Investment) เป็นการคิดค่าใช้จ่ายในการจัดให้มีระบบเฮลป์เดสก์ โดยสามารถคำนวณได้จาก

การลงทุน (I) = ผลรวมของค่าใช้จ่าย ;

โดยค่าใช้จ่ายของระบบเฮลป์เดสก์มีดังนี้

ก. ค่าตอบแทนพนักงาน ทั้งเงินเดือนประจำ และค่าล่วงเวลา

ข. ค่าใช้จ่ายในการรับเหมาช่วงงาน (Outsourcing)

ค. ค่าฝึกอบรมพนักงานเฮลป์เดสก์

ง. ค่าเอกสารในการฝึกอบรมและการกระจายข้อมูลไปยังผู้ใช้

จ. ค่าใช้จ่ายสำนักงาน

ฉ. ค่าซอฟต์แวร์ระบบเฮลป์เดสก์

ช. ค่าฮาร์ดแวร์

ซ. ค่าเช่าอุปกรณ์และอื่น ๆ เช่น ค่าบริการอินเตอร์เน็ตและค่าเช่าคู่สายโทรศัพท์ประจำเดือน

2.1.3.5.2 ประสิทธิภาพในการจัดการภาระของคำร้อง (Effectiveness of Call Load Management)

เป็นการวัด ประสิทธิภาพในการดำเนินงานเฮลป์เดสก์ โดยพิจารณาจากผลการประเมินเชิงปริมาณและเชิงคุณภาพ การประมาณเชิงปริมาณจะเป็นค่าตัวเลขหรือค่าสถิติ

ต่าง ๆ ของการดำเนินงาน แต่การประเมินเชิงคุณภาพนั้นได้จากผลการประเมินความพอใจของผู้ใช้ระบบและเจ้าหน้าที่เฮลป์เดสก์เอง โดยพิจารณาหลาย ๆ ด้านดังนี้

2.1.3.5.2.1 วัตถุประสงค์ (Objectives)

เป็นค่าวัดเชิงปริมาณ โดยพิจารณาจากวัตถุประสงค์ของระบบบริหารงานเฮลป์เดสก์ ดังนี้

ก. จำนวนครั้งหรือร้อยละที่สามารถแก้ไขปัญหาได้ในครั้งแรกที่ติดต่อเฮลป์เดสก์ (Number or percentage of calls resolved at point of call)

ข. จำนวนครั้งหรือร้อยละของปัญหาแต่ละประเภท (Number or percentage of a specific type of call)

ค. เวลาที่ใช้ในการแก้ไขปัญหาที่ไม่สามารถแก้ไขได้ทันที (Resolution times for problems not resolved at point of call)

ง. จำนวนครั้งของปัญหาที่ไม่สามารถแก้ไขได้ (Number of call left unresolved)

จ. เวลาที่ใช้ในการให้บริการอื่นที่ไม่ใช่การแก้ไขปัญหา (Delivery times of services) โดยแยกตามบริการแต่ละประเภท

ฉ. ข้อเปรียบเทียบระหว่างค่าที่ได้สัญญาหรือกำหนดไว้ กับค่าจริงที่ได้จากการดำเนินงาน (Promised versus actual) เช่น กำหนดจะทำการปรับรุ่นของซอฟต์แวร์ไมโครซอฟต์ออฟฟิสจากเวอร์ชัน 95 เป็นเวอร์ชัน 97 จำนวน 100 เครื่อง แต่สามารถทำได้จริง 80 เครื่อง

2.1.3.5.2.2 ข้อตกลงระดับของการให้บริการ (Service Level Agreement : SLA)

พิจารณา ผลการดำเนินงานเปรียบเทียบกับข้อตกลงระดับของการให้บริการ ซึ่งข้อตกลงระหว่างหน่วยงานเฮลป์เดสก์นั้นเป็นข้อตกลงร่วมกันระหว่างผู้ใช้ (ลูกค้า) กับหน่วยงานบริหารระบบเฮลป์เดสก์ ซึ่งในส่วนของผู้ใช้ข้อตกลงนี้เป็นเสมือนการรับประกันประสิทธิภาพการทำงานซึ่งผู้ใช้คาดหวังว่าจะได้รับจากเฮลป์เดสก์ และในส่วนของเฮลป์เดสก์เองก็จะเสมือนเป็นวัตถุประสงค์ที่จะต้องปฏิบัติให้บรรลุผล

2.1.3.5.2.3 ผลการประเมินจากผู้ใช้

เป็นผลการประเมินเชิงคุณภาพ โดยทำการประเมินทัศนคติของผู้ใช้ที่มีต่อระบบโดยใช้แบบประเมิน ซึ่งมักมีข้อถามที่เกี่ยวข้องกับการให้บริการ ดังนี้

ก. ความรวดเร็วและถูกต้องของบริการ (Speed and accuracy of service)

ข. การจัดการบริการเร่งด่วน เมื่อผู้ใช้มีความต้องการ (Provision of emergency service when required)

ค. คุณภาพของเจ้าหน้าที่เฮลป์เดสก์ (Quality of HelpDesk staff)

ง. คุณภาพของการฝึกอบรม (Quality of training)

จ. คุณภาพของการให้บริการ (Quality of services)

ฉ. คุณภาพของการติดต่อสื่อสาร (Quality of communication)

2.1.3.5.2.3 ผลการประเมินจากพนักงานเฮลป์เดสก์ (Staff Evaluation)

ประเมินได้จาก ทัศนคติของผู้ใช้ ลักษณะการโทรแจ้งปัญหา การฝึกอบรมที่ได้รับว่าเพียงพอหรือไม่ ประสิทธิภาพและเครื่องมือที่ใช้ในการทำงาน ภาระงาน ความสามารถและเวลาของกลุ่มผู้เชี่ยวชาญ ประสิทธิภาพของผู้ขาย และ คุณค่าของงานเฮลป์เดสก์ที่ตนเองปฏิบัติอยู่

ก. สถิติในการจัดการระบบเฮลป์เดสก์ (HelpDesk Management Statistics)

ข. การเปลี่ยนแปลงของสิ่งแวดล้อมระบบและภาระงาน (Change in environment and in call load) ตัวอย่างเช่น จำนวนเครื่องคอมพิวเตอร์ส่วนบุคคลที่ต้องดูแล จำนวนครั้งของการแจ้งปัญหาแยกตามประเภท การกระจายของการแจ้งปัญหาประจำวันและค่าเฉลี่ยเพื่อพิจารณาช่วงที่มีภาระงานมาก จำนวนการแจ้งปัญหาต่อคอมพิวเตอร์ส่วนบุคคล จำนวนครั้งของการแจ้งปัญหาต่อเจ้าหน้าที่เฮลป์เดสก์

ค. การเปลี่ยนแปลงของเวลาที่ใช้ในการแก้ไขปัญหา (Change in resolution times)

ง. การเปลี่ยนแปลงเวลาของการตอบสนองผู้ใช้ (Change in responses times)

2.1.3.5.4 ระดับของการปฏิบัติงาน (Level of Proaction)

การประเมินโดยพิจารณาจากความสามารถของระบบเฮลป์เดสก์ และในการแก้ไขปัญหาต่างๆซึ่งสามารถดูได้จาก การวิเคราะห์เพื่อหาข้อบกพร่องหรือข้อด้อยของการปฏิบัติงานแล้วหาแนวทางในการแก้ไข จากนั้นจึงทำการตรวจสอบผลของการแก้ไขดังกล่าว

2.1.4 วิธีการจัดระดับความสำคัญของปัญหา (Priority Setup)

การจัดลำดับความสำคัญของงาน เป็นเรื่องที่มีความสำคัญมากต่อระบบงานเฮลป์เดสก์ ซึ่งในภาวะที่มีการแจ้งปัญหา แล้วสามารถแก้ไขได้ทันทีจะเรียกปัญหาเหล่านั้นว่าเป็นเหตุการณ์ปกติ ทว่าในกรณีที่ไม่สามารถแก้ไขได้จะต้องทำการส่งต่อปัญหานั้น เราจำเป็นที่จะต้องกำหนดระดับความสำคัญของงานโดยพิจารณาจาก

2.1.4.1 ประเภทงานและผลกระทบกับธุรกิจ

ก. ปัญหา (Problems) คือ การขัดจังหวะการทำงานของผู้ใช้อันเนื่องมาจากการผิดปรกติหรือการใช้งานอุปกรณ์ฮาร์ดแวร์ ซอฟต์แวร์และอื่น ๆ เป็นต้น

ข. คำร้อง (Requests) คือการที่ผู้ใช้มีคำร้องขอใช้บริการตามที่ได้กำหนดไว้ในบริการของเฮลป์เดสก์ อาทิ การย้ายเครื่องคอมพิวเตอร์ส่วนบุคคล การสั่งซื้อเครื่องคอมพิวเตอร์ส่วนบุคคลใหม่ การเชื่อมต่อคอมพิวเตอร์ส่วนบุคคลเข้าระบบเครือข่าย การฝึกอบรม เป็นต้น

ค. คำถาม (Questions) คือ ในกรณีที่ผู้ใช้มีความสงสัยในการใช้งานฮาร์ดแวร์หรือซอฟต์แวร์ หรืออื่น ๆ ผู้ใช้ระบบสามารถโทรเข้ามาสอบถามที่เฮลป์เดสก์ได้

ง. งานตามแผน (Planned Work) คืองานที่มีการกำหนดให้ปฏิบัติตามกำหนดการที่แน่นอน เช่น การปรับรุ่นคอมพิวเตอร์ส่วนบุคคล และ การบำรุงรักษาอุปกรณ์คอมพิวเตอร์ตามระยะ เป็นต้น

จ. งานนอกเหนือจากแผน (Unplanned Work) คืองานที่ไม่ได้วางแผนไว้ล่วงหน้าแต่ต้องมีการดำเนินการ ซึ่งบ่อยครั้งมากที่งานดังกล่าวสามารถช่วยป้องกันปัญหาที่อาจจะเกิดขึ้นได้ในระยะยาว อาทิ การปรับรุ่นของระบบปฏิบัติการเมื่อทราบรายงานปัญหาจากผู้ขาย การย้ายเครื่องคอมพิวเตอร์แม่ข่ายสำหรับไปรษณีย์อิเลกทรอนิกส์เมื่อมีผู้ใช้จำนวนมากและการใช้งานเกิดความล่าช้ามาก เป็นต้น

2.1.4.2 ผลกระทบกับธุรกิจ

การพิจารณาว่ามีผลกระทบกับธุรกิจหรือไม่นั้น จะพิจารณาจากความสำคัญขององค์ประกอบ (Component) กับระดับความรุนแรงของปัญหา ดังนี้

2.1.4.2.1 ความสำคัญขององค์ประกอบ (Important of Components)

ก. เทคโนโลยี (ฮาร์ดแวร์และซอฟต์แวร์) โดยพิจารณาว่าเทคโนโลยีนั้นเป็นเทคโนโลยีที่ก่อให้เกิดภาวะวิกฤติต่อองค์กรหรือไม่ หากมีผลกระทบมากจะกำหนดระดับความสำคัญไว้สูง หากมีผลกระทบน้อยจะกำหนดไว้ต่ำ เช่น กรณีของตลาดหลักทรัพย์ หากระบบซื้อขายหุ้นล้มเหลว จะทำให้ไม่สามารถดำเนินธุรกรรมใด ๆ ได้ จึงให้ระดับความสำคัญสูง กรณีร้านค้าปลีกเกิดระบบการขายสินค้าล้มเหลวไม่สามารถดำเนินการได้ แต่ยังคงสามารถขายโดยเก็บเงินและลงรายการสินค้าได้ ทำให้สามารถทำงานได้บ้าง ระดับความสำคัญก็จะต่ำ

ข. โครงการ หากโครงการที่เป็นโครงการสำคัญ ๆ ที่มีผลต่อองค์กรมาก เช่น โครงการเปิดตลาดใหม่ หรือออกสินค้าใหม่ให้ทันคู่แข่ง หากเกิดการหยุดชะงักของโครงการอันเนื่องมาจากเทคโนโลยี จะต้องได้รับการแก้ไขอย่างทันท่วงที จะกำหนดระดับความสำคัญไว้สูง

ค. บุคลากร พิจารณาความสำคัญของบุคลากรที่แจ้งปัญหา หากเป็นผู้บริหารที่มีตำแหน่งสูงที่ต้องทำการตัดสินใจอย่างรวดเร็ว จะได้รับระดับความสำคัญที่สูง

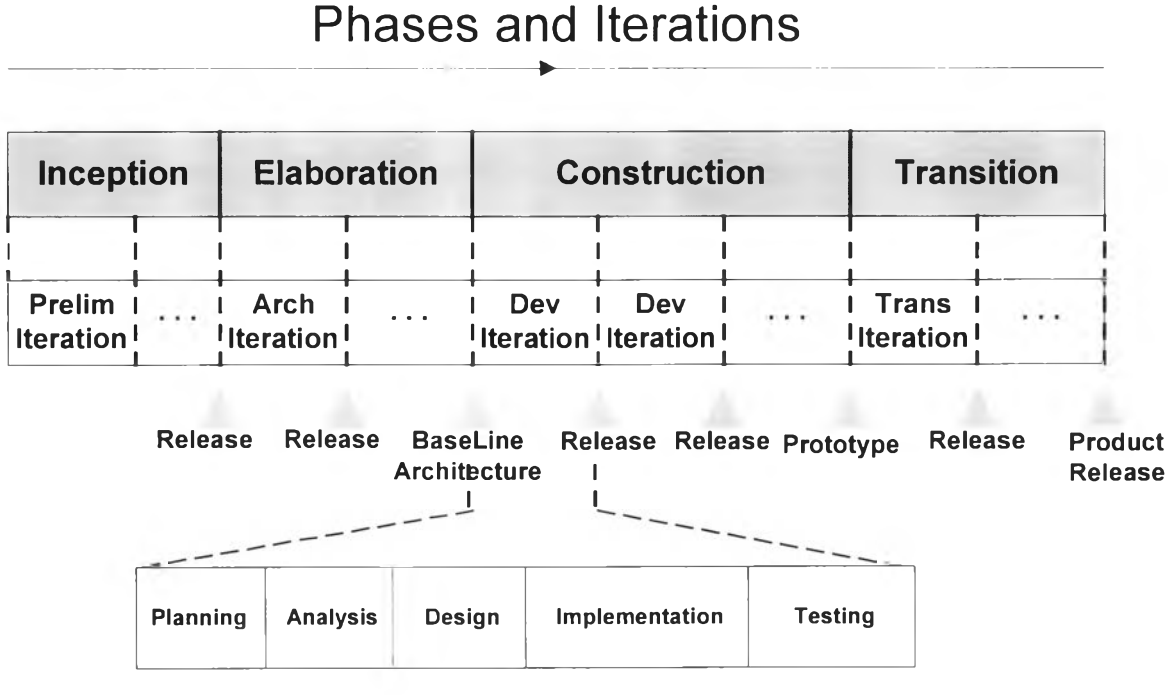
2.1.4.2.2 ระดับความรุนแรงของเหตุการณ์ (Severity of Event)

การพิจารณาความรุนแรงของเหตุการณ์นั้น พิจารณาจากระดับของความล้มเหลว ซึ่งแบ่งเป็นระบบหยุดทำงานและจำนวนคนที่ได้รับผลกระทบซึ่งแบ่งเป็นแบบทั่วทั้งองค์กร ผู้ใช้ส่วนใหญ่ และผู้ใช้บางคน

การจัดระดับความสำคัญของระบบเฮลป์เดสก์ จะพิจารณาจากองค์ประกอบและระดับความรุนแรง โดยกำหนดเป็นตัวเลข 1 – 9 (1 จะมีความสำคัญมาก และ ค่า 9 จะมีความสำคัญน้อย)

2.2 วงจรการพัฒนาซอฟต์แวร์แบบเรชันแนลออปเจกซ์ทอรีโพรเซส (Rational Objectory Process Software Development Life Cycle)[2]

วงจรการพัฒนาซอฟต์แวร์แบบเรชันแนลออปเจกซ์ทอรีโพรเซส เป็นวิธีการในการพัฒนาซอฟต์แวร์โดยใช้แนวคิดของการวนซ้ำ (Iteration) หลายรอบและการพัฒนาแบบเพิ่มพูน (Incremental) โดยทำการแยกซอฟต์แวร์ออกเป็นส่วน ๆ แล้วทำการพัฒนารอบละส่วนจนครบจำนวนรอบเท่ากับจำนวนซอฟต์แวร์ที่แบ่งไว้ โดยแบ่งออกเป็น 4 ระยะ (ดูรูปที่ 2.3 ประกอบ) ดังนี้



รูปที่ 2.3 แสดงวงจรการพัฒนาซอฟต์แวร์โดยวิธีการเรชันแนลออปเจกซ์ทอรีโพรเซส

ระยะที่ 1 อินเซ็ปชัน (Inception)

ระยะนี้จะทำการวางแผนเกี่ยวกับผลิตภัณฑ์ซอฟต์แวร์ทั้งหมด โดยทำการศึกษาเบื้องต้น เพื่อวิเคราะห์หาแนวคิดของโดเมนปัญหา และโครงสร้างเบื้องต้นของผลิตภัณฑ์ แล้วทำการกำหนดหน้าที่ (Function) และความสามารถ (Capability) ของซอฟต์แวร์โดยคร่าว ๆ เพื่อเตรียมการแบ่งซอฟต์แวร์ออกเป็นส่วน ๆ เพื่อพัฒนาในขั้นตอนต่อไป โดยสร้างยูสเคส (ดูภาคผนวก ก.) เพื่อแสดงข้อกำหนดของผลิตภัณฑ์ซอฟต์แวร์

ระยะที่ 2 อีแลบบอเรชัน (Elaboration)

ระยะนี้จะทำการออกแบบสถาปัตยกรรมเบื้องต้นของผลิตภัณฑ์ซอฟต์แวร์ โดยเริ่มจากการวิเคราะห์เพื่อหาแนวคิดของโดเมนและโครงสร้างของผลิตภัณฑ์ โดยการศึกษาความต้องการและโดเมนของปัญหาในขั้นต้น แล้วทำการกำหนดหน้าที่ ความสามารถ ประสิทธิภาพ และคุณสมบัติอื่น ๆ พร้อมกับทำการประเมินความเสี่ยงที่เป็นไปได้ (Risk Feasibility) ในด้านต่าง ๆ อาทิ ด้านเทคนิค ด้านงบประมาณ และด้านนโยบาย เป็นต้น เครื่องมือที่ช่วยในการวิเคราะห์โดเมน (Domain) และ โครงสร้างมักจะเป็น แผนภาพแสดงคลาส (Class Diagram) และ แผนภาพอินเตอร์แอ็กชัน (Interaction Diagram) ซึ่งแสดงทั้งโครงสร้างขององค์ประกอบและพฤติกรรมของผลิตภัณฑ์ซอฟต์แวร์ โดยจะเชื่อมโยงกับยูสเคสที่ได้ศึกษาไว้จากระยะที่ 1 เมื่อสิ้นสุดระยะอีแลบบอเรชันจะได้เป็นสถาปัตยกรรมเบื้องต้นของซอฟต์แวร์ก่อนที่จะทำการพัฒนาในรายละเอียดต่อไป

ระยะที่ 3 คอนสตรักชัน (Construction)

ระยะที่ 3 นี้จะเป็นระยะที่ทำการสร้างต้นแบบของผลิตภัณฑ์ซอฟต์แวร์ โดยใช้แนวคิดการวนซ้ำหลายรอบ กับ การพัฒนาแบบเพิ่มพูน โดยทำการแบ่งส่วนต่าง ๆ ของผลิตภัณฑ์ซอฟต์แวร์ที่ได้จากระยะที่ 2 ออกเป็นส่วน ๆ เพื่อแยกพัฒนารอบละส่วน โดยขนาดของระบบย่อยควรจะมีขนาดเหมาะสมกับทีมงานและใช้ระยะเวลา 3 – 6 เดือน แล้วจึงเริ่มทำการพัฒนาต้นแบบโดยวนซ้ำหลายรอบแต่รอบจะทำการพัฒนาทีละระบบย่อย โดยในแต่ละรอบจะมีขั้นตอนดังนี้

ก. วางแผน (Planning) จะทำการวางแผนเพื่อทำการแจกแจงรายละเอียดของหน้าที่และความสามารถของซอฟต์แวร์แก่ทีมงาน

ข. การวิเคราะห์ (Analysis) ทำการศึกษารายละเอียดโดยใช้แผนภาพแสดงคลาสและ อินเตอร์แอ็กชันไดอะแกรมเบื้องต้น ที่ได้จากระยะที่ 2 เป็นฐานเพื่อจัดทำรายละเอียดของความต้องการภายใต้โดเมนของระบบย่อยส่วนที่ 1 โดยผลลัพธ์จะได้เป็นคลาสไดอะแกรมที่เหมาะสมกับโดเมนส่วนที่ 1 และ แผนภาพอินเตอร์แอ็กชัน ที่สามารถจำลองพฤติกรรมการทำงานของระบบย่อยส่วนนี้

ค. การออกแบบ (Design) ขั้นตอนนี้เป็นการนำคลาสซึ่งได้รับการตรวจสอบว่าสอดคล้องกับโดเมนที่ศึกษาได้ มาทำการออกแบบส่วนที่จะติดต่อกับผู้ใช้ระบบ

(User Interface Class) แล้วทำการออกแบบขั้นตอนการทำงานกับผู้ใช้ รวมทั้งโครงแบบของฐานข้อมูลเชิงตรรกะ (Logical Database Schema)

ง. พัฒนาต้นแบบ (Prototype Implementation) ทำการพัฒนาโปรแกรมโดยทำการแปลงจากไดอะแกรมต่าง ๆ เป็นโปรแกรม และแปลงจากโครงแบบฐานข้อมูลเชิงตรรกะ เป็นโครงแบบฐานข้อมูลเชิงกายภาพ (Physical Database Schema) กำหนดมาตรฐานและรูปแบบของการโต้ตอบระหว่างโปรแกรมและผู้ใช้ระบบ

จ. ทดสอบ (Testing) ทำการทดสอบต้นแบบโปรแกรมเพื่อค้นหาข้อผิดพลาด (Error) เพื่อทำการแก้ไข ซึ่งประกอบไปด้วยการทดสอบหน่วยของโปรแกรม (Program Unit Test) การทดสอบการเชื่อมต่อของโปรแกรม(Integration Test) และการตรวจสอบก่อนการรับมอบระบบ (Acceptance Test) โดยเทคนิคต่างๆ เมื่อได้ดำเนินการพัฒนากับระบบย่อยส่วนที่ 1 จนครบ 5 ขั้นตอนแล้ว จากนั้นจะทำการเริ่มพัฒนาระบบย่อยส่วนที่ 2 ต่อไปจนกระทั่งครบทุกระบบย่อย ซึ่งในระหว่างการพัฒนาแต่ละรอบจะต้องทำการศึกษาและวิเคราะห์คลาสไดอะแกรมเดิมแล้วทำการเพิ่มเติมคลาสที่เป็นยูสเซอร์อินเตอร์เฟสเข้าไป อันส่งผลที่ดีแก่ต้นแบบในระยะสุดท้ายด้วย เนื่องจากมีการพัฒนาต้นแบบโดยได้ศึกษา วิเคราะห์ และ ออกแบบคลาสหลายครั้ง

ระยะที่ 4 ทรานสิชัน (Transition)

ระยะทรานสิชันจะทำการพัฒนาต้นแบบที่ได้จากระยะที่ 3 แล้วทำการแก้ไขข้อบกพร่องต่าง ๆ ที่พบพร้อมกับพัฒนาซอฟต์แวร์ในส่วนที่มิใช่หน้าที่หลัก แต่มีความจำเป็นต่อเสถียรภาพของซอฟต์แวร์ อาทิ ระบบรักษาความปลอดภัย ประสิทธิภาพ และการใช้งานที่ง่ายและสะดวก เป็นต้น เมื่อดำเนินการแก้ไขปัญหาและพัฒนาส่วนสนับสนุนเรียบร้อยแล้ว จากนั้นจะทำการออกแบบวิธีการติดตั้งการรวบรวมคอมโพแนนท์ที่จำเป็นเพื่อบรรจุลงในผลิตภัณฑ์ซอฟต์แวร์ วิธีการติดตั้งและเริ่มต้นระบบ ตลอดจนการบำรุงรักษาเมื่อพบข้อบกพร่องของซอฟต์แวร์

2.3 แนวคิดการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Analysis and Design Concept)

การออกแบบซอฟต์แวร์เชิงวัตถุ จะแตกต่างไปจากการวิเคราะห์ และ ออกแบบเชิงโครงสร้าง (Structural Oriented) โดยการออกแบบเชิงโครงสร้างจะเป็นการออกแบบที่แยกส่วนระหว่างกระบวนการ (Process) และ ข้อมูล (Data) แต่สำหรับการออกแบบซอฟต์แวร์เชิงวัตถุจะเป็นการออกแบบโดยพิจารณาจากวัตถุซึ่งจะประกอบไปด้วยกระบวนการและข้อมูลรวมกัน และในระหว่างกระบวนการวิเคราะห์และออกแบบจะมีการใช้แผนภาพในขั้นตอนต่างๆ อาทิ แผนภาพ

แสดงคลาส (ดู 2.4.2) แผนภาพวัตถุจะทำการพัฒนาคลาสในระดับแนวคิดก่อน เมื่อเข้าสู่ระยะของการพัฒนาโปรแกรมจึงจะทำการพัฒนาเพิ่มในส่วนของยูสเซอร์อินเตอร์เฟส แผนภาพในขั้นตอนต่างๆจะใช้แผนภาพเดิม แล้วทำการพัฒนาเพิ่มเติม ส่วนการวิเคราะห์และออกแบบเชิงโครงสร้างในแต่ละระยะจะทำการพัฒนาแผนภาพใหม่ ทำให้ต้องทำการแปลงจากแผนภาพหนึ่งไปสู่อีกแผนภาพหนึ่ง ซึ่งอาจจะเกิดข้อผิดพลาดในระหว่างกระบวนการแปลง ดังนั้นในการวิเคราะห์ และออกแบบเชิงวัตถุ จะมีการศึกษาแผนภาพซ้ำหลายครั้งเพื่อปรับให้ได้วัตถุต่างๆที่สามารถตอบสนองความต้องการต่างๆของผู้ใช้งานระบบได้ โดยรายละเอียดต่างๆของแนวคิดนี้จะอ้างอิงกับการพัฒนาซอฟต์แวร์แบบเรชันแนลออปเจกซ์ทอรีโพรเซส และส่วนการนิยามระบบจะใช้ภาษาโมเดลยูนิฟายด์ซึ่งเรียกย่อๆว่าภาษายูเอ็มแอล (UML) ซึ่งแนวคิดของการวิเคราะห์และออกแบบเชิงวัตถุจะมีรายละเอียดดังต่อไปนี้

2.3.1 พื้นฐานของแนวคิดเชิงวัตถุ

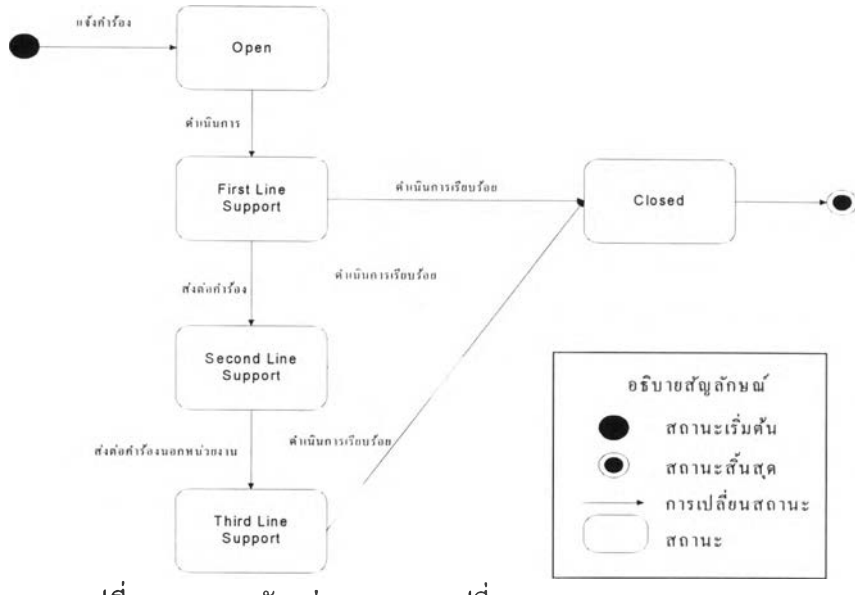
แนวคิดพื้นฐานของแนวคิดเชิงวัตถุในหัวข้อนี้จะประกอบไปด้วยความหมายและนิยามของวัตถุ คลาส และ ความสัมพันธ์ระหว่างคลาส ดังนี้

2.3.1.1 วัตถุ[3] คือสิ่งใด ๆ ก็ตามที่ประกอบไปด้วย สถานะ (State) พฤติกรรม (Behavior) และสามารถจำแนกเอกลักษณ์ (Identify) ได้ดังรายละเอียดต่อไปนี้

ก. สถานะ ประกอบไปด้วยแอตทริบิวท์ของวัตถุบวกกับค่าปัจจุบันของวัตถุนั้น เช่น วัตถุ คำร้องมีสถานะเป็น “เปิด” “การให้บริการระดับที่ 1” “การให้บริการระดับที่ 2” “การให้บริการระดับที่ 3” “ปิด” โดยเมื่อวัตถุมีการเปลี่ยนแปลงสถานะไป ค่าของแอตทริบิวท์ สถานะ (Status) จะถูกเปลี่ยนตามไปด้วย (ดูรูปที่ 2.4 ประกอบ)

ข. พฤติกรรม คือการที่วัตถุกระทำหรือถูกกระทำ ซึ่งจะแสดงโดยการส่งข้อความและสถานะที่เปลี่ยนไป ตัวอย่างเช่น วัตถุคำร้องที่สถานะเป็น “การให้บริการระดับที่ 2” จะเปลี่ยนเป็น “ปิด” เมื่อดำเนินการเรียบร้อย และจะเปลี่ยนเป็น “การให้บริการระดับที่ 3” หากมีการส่งต่อคำร้องไปยังนอกหน่วยงาน (ดูรูปที่ 2.4 ประกอบ)

ค. การจำแนกเอกลักษณ์วัตถุ (Object Identified) คือสิ่งที่ทำให้สามารถแยกแยะให้วัตถุหนึ่งต่างจากวัตถุอื่นๆได้ ตัวอย่างเช่น วัตถุคำร้องสามารถระบุคำร้องโดยเลขที่คำร้อง วัตถุพนักงานสามารถระบุพนักงานแต่ละคนโดยรหัสพนักงาน เป็นต้น



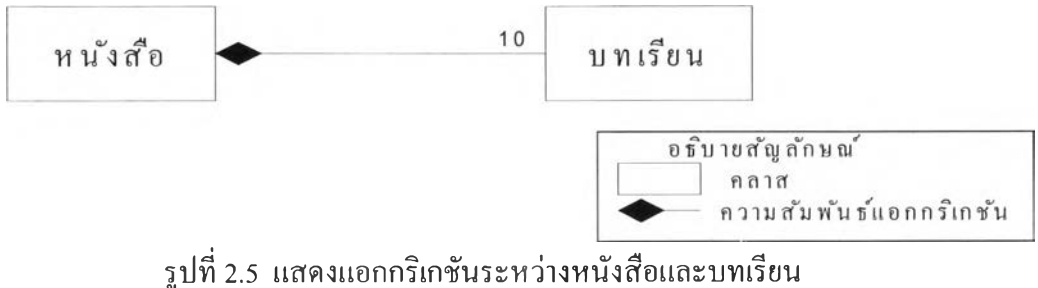
รูปที่ 2.4 แสดงตัวอย่างแผนภาพเปลี่ยนสถานะ (Statechart Diagram)

2.3.1.2 คลาส (Class) และ ความสัมพันธ์ระหว่างคลาส (Class Relationship)

1) คลาส คือเซตของวัตถุซึ่งมีโครงสร้าง (หรือแอตทริบิวท์) และ พฤติกรรมแบบเดียวกัน เช่น คำร้องเลขที่ 100 (แจ้งคำร้องเวลา 10.00 น. โดยนายดุสิต) คำร้องเลขที่ 200 (แจ้งคำร้องเวลา 11.30 น. โดยนายสุรพล) ทั้ง 2 วัตถุจะมีโครงสร้างเหมือนกัน คือ เลขที่คำร้อง เวลาแจ้ง และผู้แจ้ง โดยจะมีพฤติกรรมดังรูปที่ 2.4

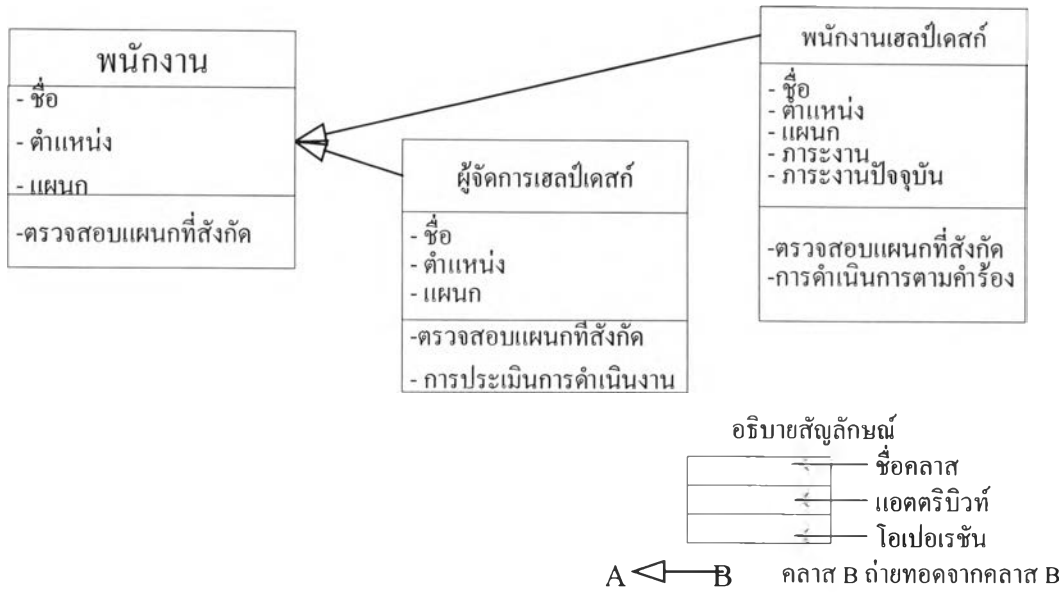
2) ความสัมพันธ์ระหว่างคลาส (Class Relationship)

ก. แอกริเกชัน(Aggregation) คือ การที่วัตถุที่เกิดจากวัตถุอื่นรวมกัน โดยคุณสมบัติ และพฤติกรรมของวัตถุนั้น แตกต่างไปจากวัตถุซึ่งเป็นส่วนประกอบ ตัวอย่างแสดงโดยรูปที่ 2.5 แสดงวัตถุหนังสือซึ่งประกอบไปด้วยบทเรียน 10 บท



รูปที่ 2.5 แสดงแอกกริเกชันระหว่างหนังสือและบทเรียน

ข. การถ่ายทอด (Inheritance) คือการที่วัตถุหนึ่ง ๆ สามารถถ่ายทอดคุณลักษณะ และพฤติกรรมไปยังวัตถุอื่น ๆ ได้โดยคลาสของวัตถุที่เป็นผู้ถ่ายทอดจะเรียกว่าซุปเปอร์คลาส (Superclass) และคลาสที่ถูกถ่ายทอดเรียกว่าสับคลาส (Subclass)



รูปที่ 2.6 แสดงการถ่ายทอดจากพนักงานไปสู่พนักงานเฮลป์เดสก์ และ ผู้จัดการเฮลป์เดสก์

จากรูปที่ 2.6 คลาสพนักงานมีแอตทริบิวท์ รหัสพนักงาน ชื่อ ตำแหน่ง และ แผนกที่สังกัด โดยมีพฤติกรรมคือสามารถตรวจสอบแผนกที่สังกัดได้ ดังนั้น พนักงานเฮลป์เดสก์ และ ผู้จัดการเฮลป์เดสก์ก็จะมีแอตทริบิวท์ รหัสพนักงาน ชื่อ ตำแหน่ง และแผนกที่สังกัดเช่นเดียวกับพนักงาน และ สามารถตรวจสอบแผนกที่สังกัดได้ โดยพนักงานเฮลป์เดสก์จะมีคุณสมบัติและพฤติกรรมอื่นนอกเหนือจาก คลาสพนักงาน อาทิ แอตทริบิวท์ภาระงาน ภาระงานปัจจุบัน และพฤติกรรมการดำเนินการตามคำสั่ง การเปลี่ยนขนาดภาระงาน เป็นต้น

ค. แอสโซซิเอชัน (Association)

เป็นความสัมพันธ์ระหว่างคลาสในระดับที่ไม่ชัดเจนเท่ากับความสัมพันธ์แอกกริเกชัน และ การถ่ายทอดโดยจะแสดงสัดส่วนของความสัมพันธ์เรียกว่า คาดิแนลริตี้ (Cardinality) ซึ่งจำแนกเป็น 3 แบบคือ

1 1 หมายถึง ความสัมพันธ์แบบ 1 ต่อ 1 (one-to-one)

1 1..* หมายถึง ความสัมพันธ์แบบ 1 ต่อ 1 กลุ่ม (one-to-many)

1..* 1..* หมายถึง ความสัมพันธ์แบบ กลุ่ม ต่อ กลุ่ม (many-to-many)

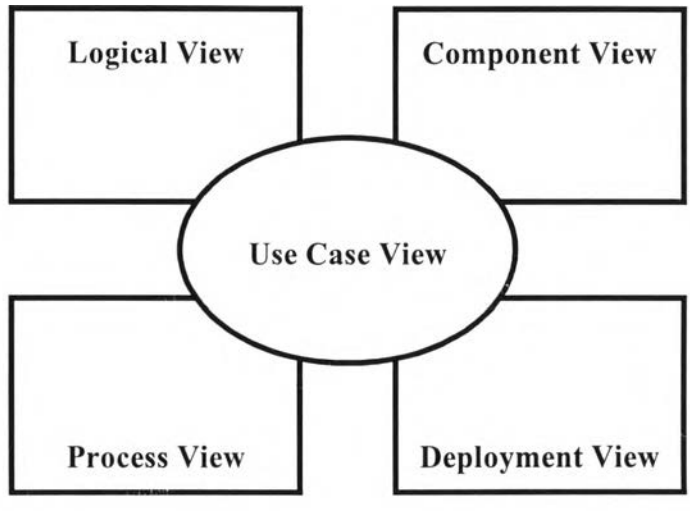
รูปที่ 2.7 แสดงแอสโซซิเอชันระหว่างพนักงานและคำร้อง

จากรูปที่ 2.7 พนักงาน 1 คนสามารถมีคำร้องได้ตั้งแต่ 1 คำร้องขึ้นไป โดยแต่ละคำร้องจะเป็นของพนักงาน 1 คนเท่านั้น

2.3.2 การออกแบบสถาปัตยกรรมของซอฟต์แวร์โดยภาษายูเอ็มแอล (Software Architectural Design using UML)

สถาปัตยกรรมซอฟต์แวร์

คือ เซตของสิ่งที่ใช้กำหนดโครงสร้างของซอฟต์แวร์สำหรับการออกแบบสถาปัตยกรรมซอฟต์แวร์ โดยภาษายูเอ็มแอลนั้นสามารถทำการอธิบายหรือแสดงหรือแทนซอฟต์แวร์ได้ใน 5 มุมมอง (View/Perspective) ดังรูปที่ 2.8 โดยมีแนวคิดที่ว่า ไม่สามารถนำเสนอระบบในทุกแง่มุมโดยใช้ไดอะแกรมหรือในมุมมองเดียว จึงได้ออกแบบให้มี 5 มุมมอง โดยแต่ละมุมมองจะมีวัตถุประสงค์ที่แตกต่างกัน เพื่อนำเสนอกับบุคคลที่แตกต่างกัน



รูปที่ 2.8 แสดงมุมมองของสถาปัตยกรรมซอฟต์แวร์ตามโมเดลของยูเอ็มแอล [2]

ก. มุมมองเชิงตรรกะ (Logical View) เป็นมุมมองซึ่งแสดงถึงองค์ประกอบระดับตรรกะ อันได้แก่ คลาส ความสัมพันธ์ระหว่างคลาส และ กลุ่มของคลาส ซึ่งมุมมองนี้จะถูกสร้างในระยะอีแลบบอเรชัน ในวงจรการพัฒนาซอฟต์แวร์แบบ เรชั่นแนลออบเจ็กท์ทอรีโพรเซส เพื่อแสดงแนวคิดและขอบเขตของระบบ

ข. มุมมองโพรเซส (Process View) หรือมุมมองคอนเคอร์เรนท์ (Concurrent) เป็นมุมมองซึ่งแสดงปฏิสัมพันธ์ระหว่าง โพรเซสในระดับต่างๆโดยจะประกอบไปด้วย โพรเซส เทร็ด (Thread, ดูภาคผนวก ก.) การสื่อสารระหว่างโพรเซส (Process Communication) และ

กลไกแบบอะซิงโครนัส (Asynchronous mechanism) ซึ่งมุมมองนี้จะถูกสร้างในระยะอีแลบบอเรชัน และปรับปรุงในระยะการคอนสตรักชัน

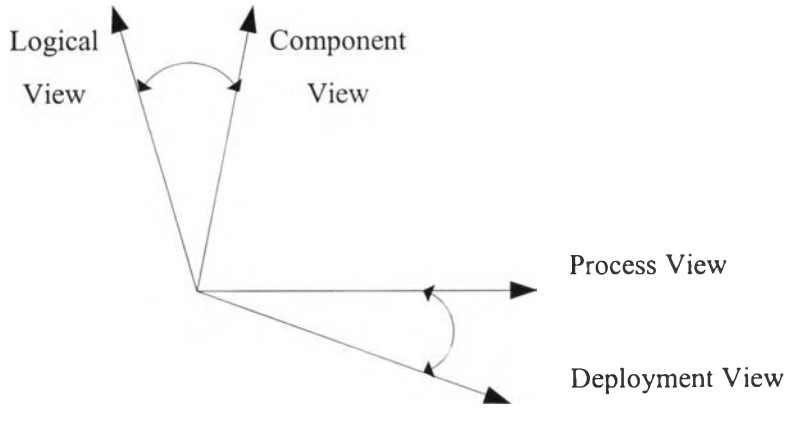
ค. มุมมองดีพลอยเมนท์ (Deployment View) เป็นมุมมองซึ่งแสดงองค์ประกอบทางกายภาพของระบบโดยจะประกอบไปด้วย โหนด (Node) หรือ โพรเซสเซอร์ต่าง ๆ (ดูภาคผนวก ก.) ที่ใช้ในการทำงานของระบบ และ การเชื่อมต่อกันของโหนด

ง. มุมมองคอมโพเนนท์ (Component View) เป็นมุมมองซึ่งแสดงโมดูลต่างๆของระบบในระดับโปรแกรม

จ. มุมมองยูสเคส (Use Case View) เป็นมุมมองซึ่งแสดง ทรานแซกชัน (Transaction, ดูภาคผนวก ก.) ของการทำงานต่างๆ ในมุมมองของผู้ใช้ ซึ่งเป็นการเริ่มต้นวงจรการพัฒนาระบบและเป็นตัวเชื่อมมุมมองที่ 1-4 เข้าด้วยกัน ดังรูปที่ 2.8

ความเหมาะสมในการเลือกมุมมองต่างๆ

ทั้ง 5 มุมมองของกระบวนการออกแบบสถาปัตยกรรมซอฟต์แวร์โดยวิธีการของยูเอ็มแอล (UML Process) นั้นจะมีความสัมพันธ์ระหว่างมุมมองต่างๆดังแสดงโดยรูปที่ 2.9 มุมมองเชิงตรรก และ มุมมองคอมโพเนนท์จะสัมพันธ์กันโดยมุมมองคอมโพเนนท์จะแสดงกลุ่มของคลาสที่สัมพันธ์กันอยู่ในกลุมคลาส (Package) เดียวกัน ส่วนมุมมองโพรเซส และ ดีพลอยเมนท์ก็จะสัมพันธ์กัน ด้วยเป็นการแสดงโพเซสบนโพรเซสเซอร์ต่างๆและการนำเสนอระบบนั้นไม่จำเป็นที่จะต้องนำเสนอทั้ง 5 มุมมอง โดยมีข้อควรพิจารณาดังนี้



รูปที่ 2.9 แสดงความสัมพันธ์ของมุมมอง

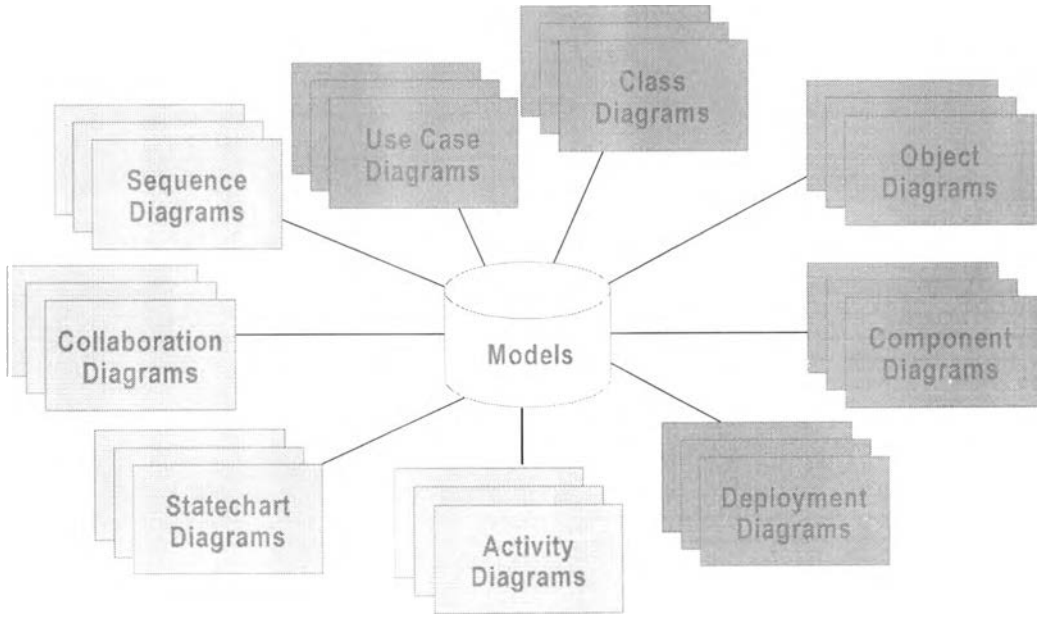
1) กรณีที่ระบบทำงานบนเครื่องเดียว ไม่มีการเชื่อมต่อระหว่างโพรเซสเซอร์จึงไม่จำเป็นที่จะต้องแสดงมุมมองดีพลอยเมนท์

2) กรณีที่มีโพรเซสเดียว ไม่มีการทำงานแบบขนาน (Parallel Processing) ระบบจะทำงานบนเทร็ดเดียว (Single Thread) ไม่มีการสื่อสารระหว่างโพรเซส จึงไม่มีความจำเป็นที่จะต้องแสดงมุมมองโพรเซส

3) กรณีประเมินแล้วขนาดของซอฟต์แวร์ว่าเป็นโปรแกรมขนาดเล็กมีจำนวนคลาสน้อยจนสามารถแสดงทุกๆคลาสได้ด้วยแผนภาพเดียว จึงไม่มีความจำเป็นที่จะต้องแสดงมุมมองคอมโพเนนท์

2.4 ภาษายูเอ็มแอล (UML Language)[4]

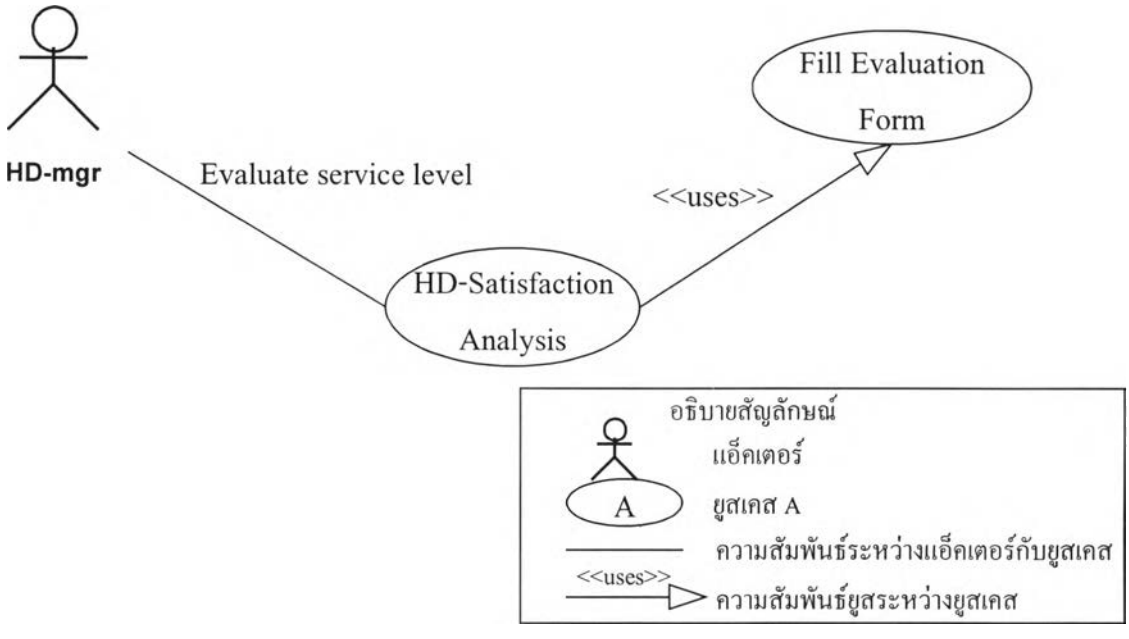
ภาษายูเอ็มแอลเป็นภาษาจินตทัศน์ (Visual Langauge) สามารถทำการกำหนด (Specify) เพื่อใช้ในการสร้างหรือแสดงระบบใดๆ และยังใช้เป็นเอกสารประกอบสิ่งที่กำหนดหรือสร้างได้ ยูเอ็มแอลประกอบไปด้วย 9 แผนภาพ (ดังรูปที่ 2.10) เพื่อใช้ในการโมเดลระบบในมุมมองต่างๆได้ โดยแบ่งออกเป็น 2 กลุ่มคือ มุมมองสเตติก (Static View) ประกอบไป ด้วยแผนภาพยูสเคส คลาส วัตถุ คอมโพเนนท์ และ ดีพลอยเมนท์ กลุ่มที่ 2 คือ มุมมองไดนามิก (Dynamic View) ประกอบไปด้วยแผนภาพ ซีเควน คอลแลบบอเรชัน เสตทชาร์ท และ แอ็กทิวิตี โดยมีรายละเอียดดังต่อไปนี้



รูปที่ 2.10 แสดงแผนภาพต่างๆที่ใช้ในภาษายูเอ็มแอล

2.4.1 แผนภาพยูสเคส (Use Case Diagram)

เป็นแผนภาพที่แสดงฟังก์ชันหรือทรานแซกชัน ในมุมมองของ ผู้ใช้ระบบซึ่งจะเรียกว่า แอ็คเตอร์ (Actor) ยูสเคสจะสร้างเมื่อทำการเริ่มโครงการพัฒนาซอฟต์แวร์ เพื่อใช้ในการศึกษา ทำความเข้าใจ และกำหนดความต้องการของระบบ และ ยังสามารถใช้ในการตรวจสอบและสร้างกรณีทดสอบระบบได้อีกด้วย

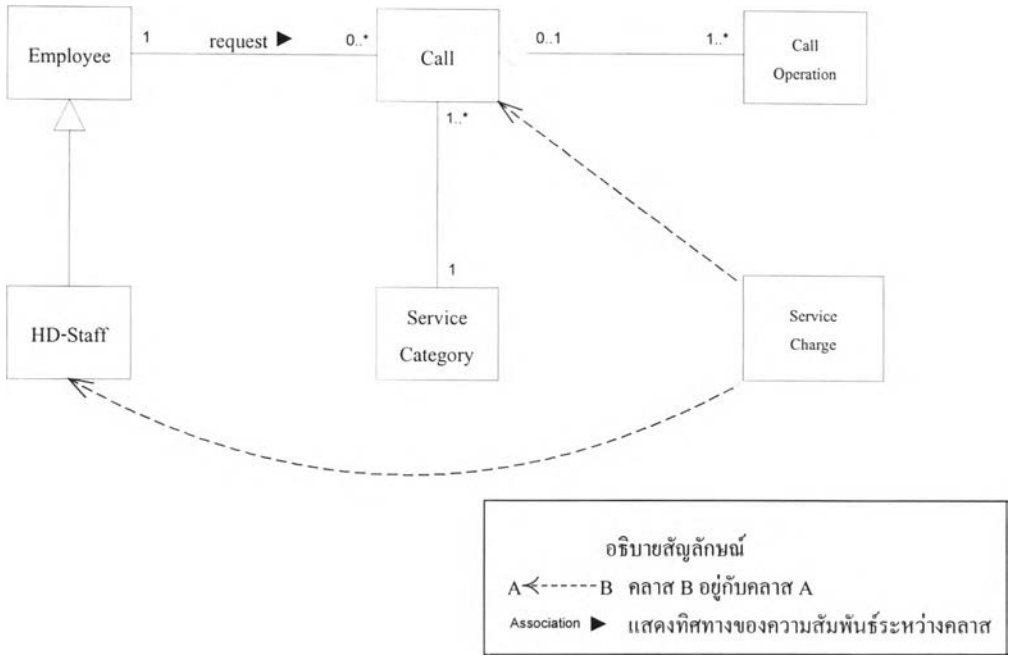


รูปที่ 2.11 แสดงตัวอย่างแผนภาพยูสเคสการวิเคราะห์ความพึงพอใจของผู้ใช้ต่อระบบเฮลป์เดสก์

จากรูปที่ 2.11 ประกอบไปด้วย ยูสเคสการวิเคราะห์ความพึงพอใจ (HD-Satisfaction) และ ยูสเคสการตอบแบบประเมิน (Fill Evaluation Form) ซึ่งยูสเคสการวิเคราะห์ความพึงพอใจจะ สัมพันธ์กับยูสเคสการตอบแบบประเมิน โดยความสัมพันธ์ยูส อันหมายถึงยูสเคสงานการวิเคราะห์ความพึงพอใจนั้น จะรวมไปถึงงานที่เกิดจากยูสเคสการตอบแบบประเมินด้วย

2.4.2 แผนภาพแสดงคลาส (Class Diagram)

เป็นไดอะแกรมซึ่งพัฒนาโดยนักวิเคราะห์ระบบ นักออกแบบ หรือผู้สร้างระบบ เพื่อนิยาม กำหนดแนวคิด ในการทำความเข้าใจระบบ ซึ่งประกอบไปด้วย คลาส และความสัมพันธ์ระหว่างคลาส

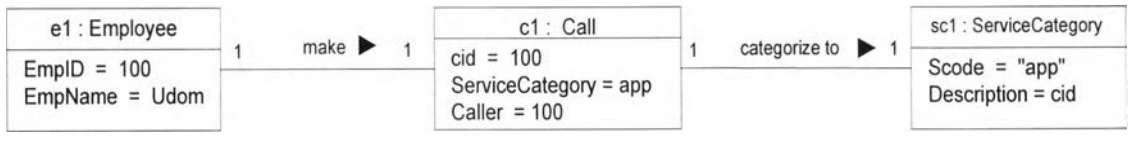


รูปที่ 2.12 แผนภาพแสดงคลาสคำร้อง

แผนภาพแสดงคลาสรูปที่ 2.12 ประกอบไปด้วยคลาสพนักงาน (Employee) พนักงานเฮลป์เดสก์ (HD-Staff) คำร้อง (Call) การดำเนินการตามคำร้อง (Call Operation) ประเภทบริการ (Service Category) ค่าบริการ (Service Charge) โดยพนักงานสามารถทำการออกคำร้องได้หลายคำร้อง โดยแต่ละคำร้อง จะกำหนดประเภทบริการไว้ แต่ละคำร้องสามารถดำเนินการได้หลายครั้ง เมื่อทำการคำนวณจะคำนวณจากค่าบริการของพนักงานเฮลป์เดสก์ และ เวลาที่ใช้ในการดำเนินการตามคำร้อง

2.4.3 แผนภาพแสดงวัตถุ (Object Diagram)

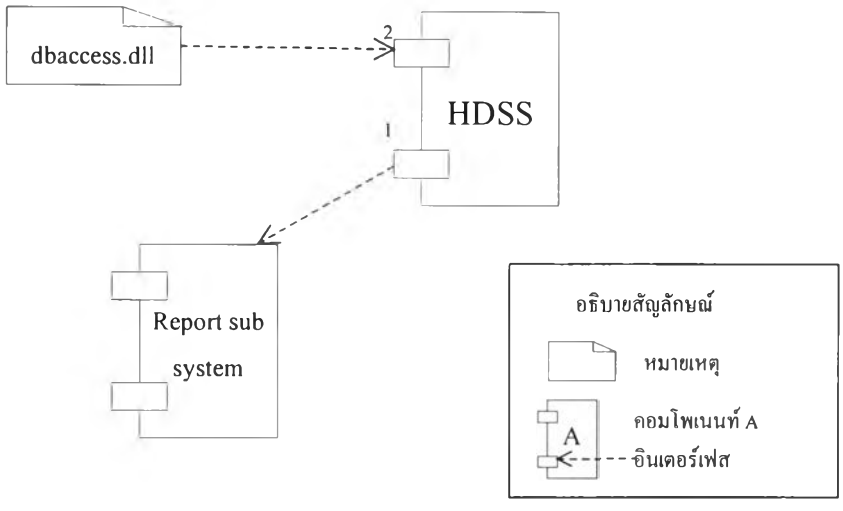
แสดงการสร้างวัตถุและความสัมพันธ์ระหว่างวัตถุ มักใช้ในช่วงการวิเคราะห์ และออกแบบ หรือในการสร้างกรณีข้อมูลสำหรับทดสอบ โดยแผนภาพนี้จะคล้ายคลึงกับ แผนภาพแสดงคลาส โดยต่างกันที่แผนภาพแสดงวัตถุจะระบุค่าที่ใช้ในการระบุวัตถุ ดังตัวอย่างในรูปที่ 2.13 แสดงถึงวัตถุพนักงาน e1 ทำการสร้างคำร้อง c1 ซึ่งจำแนกประเภทบริการเป็นประเภท sc1



รูปที่ 2.13 แสดงวัตถุคำร้องและการเชื่อมกับวัตถุประเภทบริการและพนักงาน

2.4.4 แผนภาพแสดงคอมโพเนนท์ (Component Diagram)

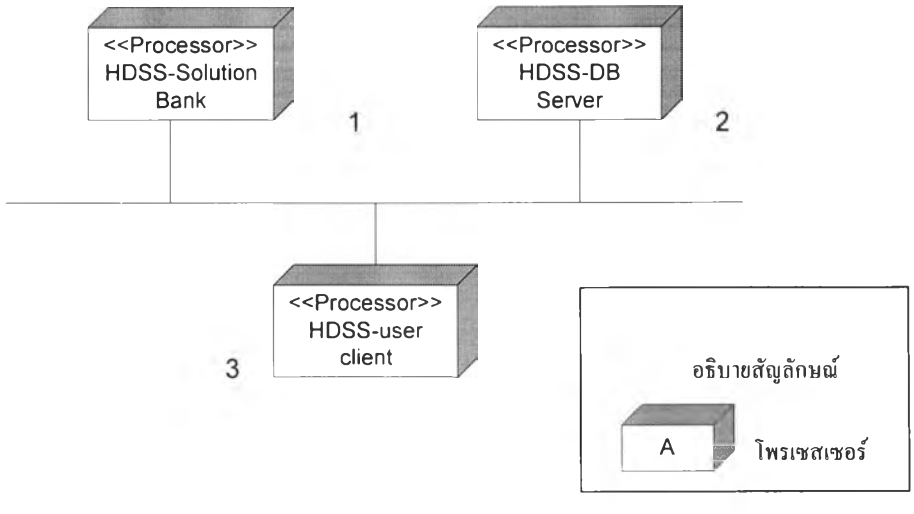
แสดงโครงสร้างของระบบในระดับโปรแกรม หรือ แพ็คเก็จ (กลุ่มคลาส) ความสัมพันธ์ และ การเชื่อมต่อ โปรแกรม จากตัวอย่างในรูปที่ 2.14 คอมโพเนนท์ เฮชดีเอสเอส จะเชื่อมต่อกับ ระบบย่อยสำหรับออกรายงาน (Report Subsystem)



รูปที่ 2.14 แสดงแผนภาพคอมโพเนนท์

2.4.5 แผนภาพดีพลอยเมนท์ (Deployment Diagram)

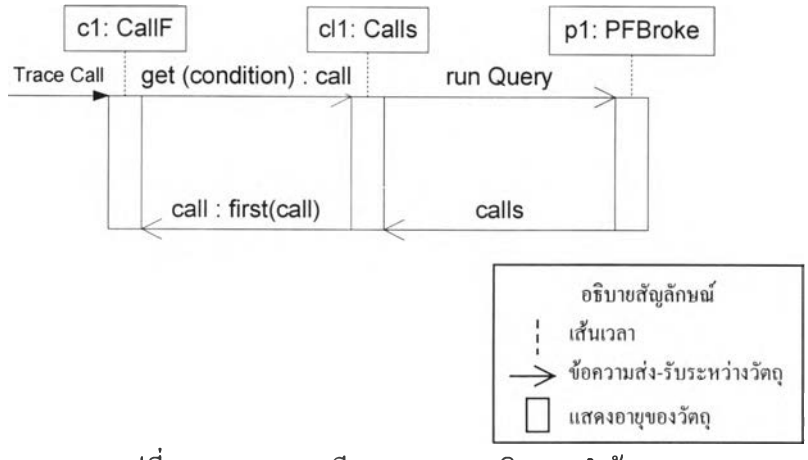
แสดงการกระจายคอมโพเนนท์ที่พัฒนาแล้วลงในฮาร์ดแวร์ต่างๆ ดังรูปที่ 2.15 แสดงการกระจายคอมของโพเนนท์ต่างๆ โดย ดาต้าเบสเซิร์ฟเวอร์ระบบเฮชดีเอสเอส (HDSS-DB Server) อยู่บนโพรเซสเซอร์ 1 โซลลูชันแบงค์ของระบบเฮชดีเอสเอส (HDSS-Solution Bank) อยู่บนโพรเซสเซอร์ 2 และ เฮชดีเอสเอสส่วนของผู้ใช้ (HDSS-user client) อยู่บนโพรเซสเซอร์ 3



รูปที่ 2.15 แสดงแผนภาพดีพลอยเมนท์

2.4.6 แผนภาพซีเควน (Sequence Diagram)

เป็นแผนภาพหนึ่งในกลุ่มไดนามิกโมเดลอันประกอบไปด้วย แผนภาพซีเควน คอลแลบบอเรชัน เสตทชาร์ต และ แอ็คทิวิตี ซึ่งแสดงพฤติกรรมของระบบและการไหลเวียนของระบบ (Flow of Control) ตามลำดับของเวลาที่เกิด

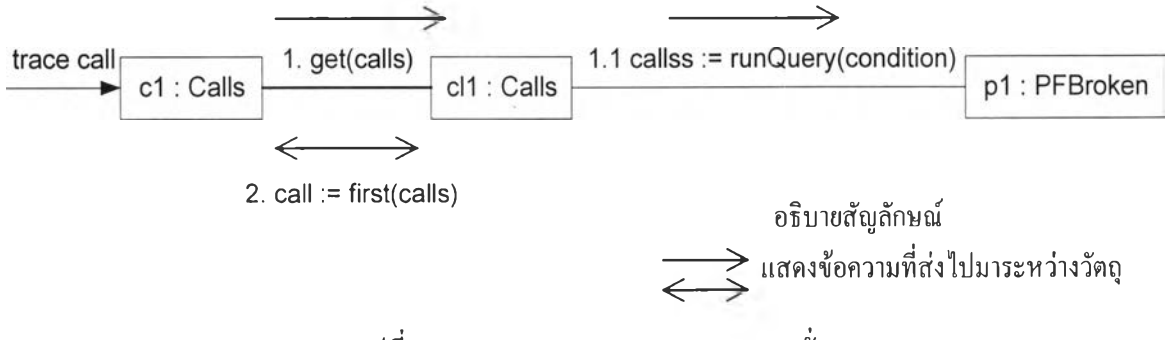


รูปที่ 2.16 แผนภาพซีเควนของการติดตามคำร้อง (Trace Call)

จากรูปที่ 2.16 เมื่อมีคำสั่งติดตามคำร้องผ่านวัตถุคำร้อง c1 ซึ่งเป็น ยูสเซอร์อินเตอร์เฟส คลาสจะทำการส่งข้อความให้สร้างวัตถุรายการคำร้อง cl1 ซึ่งเป็นเซ็ตของคำร้อง แล้วส่งข้อความเพื่อดึงข้อมูลผ่านวัตถุเพอซีซเทนท์โบรกเกอร์ (ดูภาคผนวก ก.) p1 ซึ่งเป็นวัตถุสำหรับการเชื่อมต่อกับฐานข้อมูล เพื่อดึงข้อมูล แล้วเพิ่มข้อมูลลงในเซต cl1 จากนั้นส่งคำร้องที่ได้จาก โอเปอเรชัน first กลับไปเพื่อแสดงในฟอร์มคำร้อง c1

2.4.7 แผนภาพคอลแลบบอเรชัน (Collaboration Diagram)

เป็นแผนภาพที่สร้างขึ้นเพื่อแสดงพฤติกรรมของระบบ ต่างกันที่แผนคอลแลบบอเรชันจะนำเสนอการส่งผ่านข้อความ และ การไหลเวียนของระบบ ส่วนแผนภาพซีเควนจะเน้นที่ลำดับและเวลาของการทำงาน ดังตัวอย่างในรูปที่ 2.17



รูปที่ 2.17 แสดงแผนภาพคอลแลบบอเรชั่น

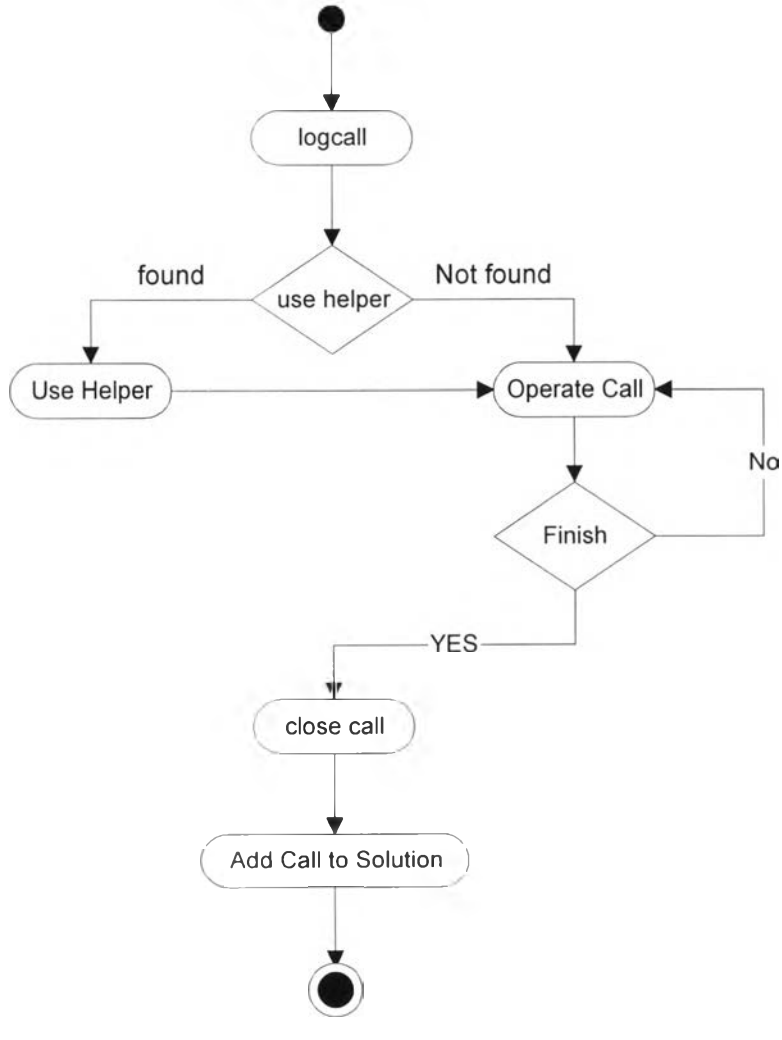
จากรูปที่ 2.17 แสดงวัตถุคำร้อง c1 รายการคำร้อง cl1 และ วัตถุเพอซีซเทนท์โบรกเกอร์ p1 จะติดต่อกันเหมือนกับแผนภาพซีเควนในรูปที่ 2.17

2.4.8 แผนภาพเสตทชาร์ท (Statechart Diagram)

แสดงวงจรชีวิตของวัตถุ (Object Life Cycle) โดยจุดประสงค์เพื่อแสดงพฤติกรรมของวัตถุในเหตุการณ์ต่างๆจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง (ดูรูปที่ 2.4 ประกอบ)

2.4.9 แผนภาพแอ็คทิวิตี (Activity Diagram)

เป็นแผนนภาพซึ่งใช้แสดงผังงานของระบบโดยสามารถแสดงการทำงานร่วมกันของยูสเคสต่างๆ ดังรูปที่ 2.18 จะแสดงทรานแซกชันของการบันทึกคำร้อง (Log Call) จนกระทั่งปิดคำร้อง



รูปที่ 2.18 แผนภาพแสดงไดอะแกรมแอ็คทิวิตี

(Close Call) โดยเริ่มจากการบันทึกคำร้อง จะทำการเรียกใช้ระบบผู้ช่วยเหลือ (Helper) หากพบและสามารถดำเนินการได้ก็จะทำการปิดคำร้อง หากไม่พบจะต้องดำเนินการตามคำร้องเองจนกระทั่งเสร็จสิ้นจึงจะทำการปิดคำร้อง

เหตุผลสนันสนุนการเลือกภาษายูเอ็มแอล

1. ภาษายูเอ็มแอลเป็นภาษาที่เปิด โดยมีมาตรฐานของโอเอ็มจี (OMG) รองรับ
2. สนับสนุนทุกขั้นตอนของวงจรการพัฒนาซอฟต์แวร์
3. สามารถประยุกต์ใช้ได้ในทุก ๆ แอบพลิเคชัน
4. พื้นฐานของภาษาได้มาจากประสบการณ์และความต้องการของกลุ่มนักออกแบบสถาปัตยกรรมระบบ
5. สนับสนุนโดยเครื่องมือและผู้ผลิตเป็นจำนวนมาก อาทิ ไอบีเอ็ม เรชันแนลซอฟต์แวร์ ไอโลจิก ออราเคิล ไมโครซอฟต์ แพลทตินัม และ ยูนิซิส เป็นต้น

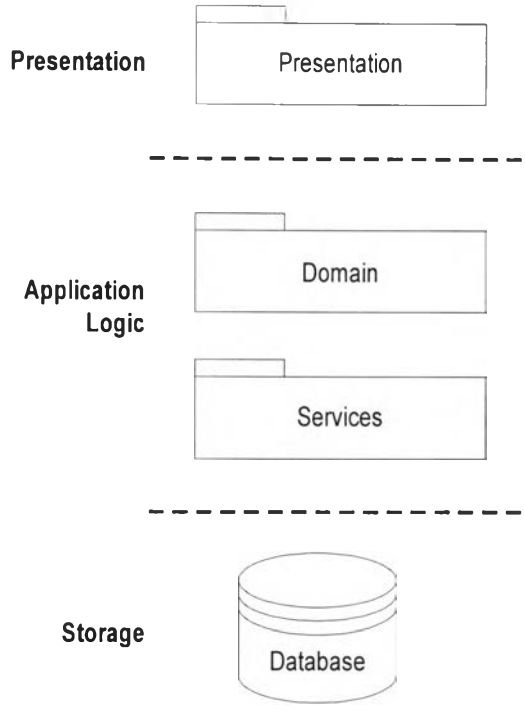
สามารถทำการกำหนดประเภทขององค์ประกอบบางอย่าง ได้ตามใจผู้ใช้ (User Defined) โดยใช้สเตริโอไทด์ (Stereotype) ซึ่งแสดงภายใต้เครื่องหมาย << stereotype>> ทำให้สามารถแสดงแนวคิดบางอย่างที่มิได้กำหนดโดยภาษายูเอ็มแอล ตัวอย่างเช่นต้องการแสดงประเภทของคลาส ซึ่งผู้ใช้จำแนกเป็น คลาสยูสเซอร์อินเตอร์เฟส และ คลาสดาต้าเซอร์วิส ก็สามารถกำหนดสัญญลักษณ์ <<User Interface>> ไว้ที่คลาสที่จัดอยู่ในประเภทยูสเซอร์อินเตอร์เฟส และ <<Data Service>> ไว้ที่คลาสที่จัดอยู่ในประเภทดาต้าเซอร์วิส

2.5 การออกแบบซอฟต์แวร์ตามสถาปัตยกรรม 3 เทียร์ (Three tier software architectural design)[15]

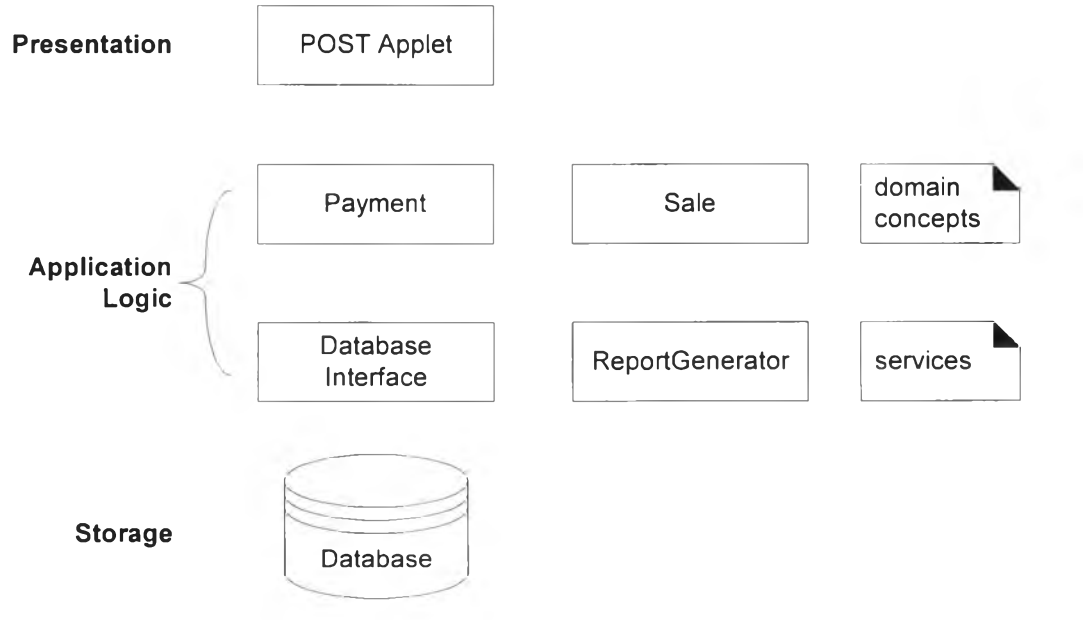
การออกแบบสถาปัตยกรรมซอฟต์แวร์แบบ 3 เทียร์จะแบ่งคอมโพเนนท์ของซอฟต์แวร์ออกเป็น 3 ระดับ (ดูรูปที่ 2.19 และ 2.20 ประกอบ) คือ

2.5.1 ระดับพรีเซนเตชัน (Presentation Layer)

เป็นส่วนของวัตถุยูสเซอร์อินเตอเฟส อาทิ บราวเซอร์ วินโดว์ฟอร์มสำหรับบันทึกข้อมูล และ แอปเพล็ต ตัวอย่างดังในรูปที่ 2.19 เช่น แอปเพล็ตระบบพีโอเอส (POST Applet) โดยวัตถุในระดับนี้จะมีหน้าที่ในการติดต่อระหว่างผู้ใช้กับวัตถุในระดับถัดไป



รูปที่ 2.20 แสดงตัวอย่างสถาปัตกรรมซอฟต์แวร์แบบ 3 เทียร์[13]



รูปที่ 2.19 แสดงสถาปัตยกรรมแบบ 3 เทียร์[12]

2.5.2 ระดับแอปพลิเคชันลอจิก (Application Logic)

เป็นส่วนที่ใช้ในการควบคุมตรรกะของงาน และ ธุรกรรมต่างๆ ซึ่งเลเยอร์นี้จะรองรับโดเมน ของปัญหาหรือของระบบ โดยแบ่งเป็นระดับย่อย (Sublayers) คือ

1) วัตถุโดเมนของปัญหา (Problem Domain Objects)

เป็นกลุ่มของวัตถุที่เป็นตัวแทนแนวคิดของระบบ อาทิ วัตถุรายการชำระเงิน (Payment Objects) และ วัตถุรายการขาย (Sale) ในระบบพีโอเอส (POS) เป็นต้น

2) วัตถุบริการ (Service Objects)

เป็นกลุ่มของวัตถุที่ไม่ได้อยู่ในโดเมนของปัญหา โดยจะทำหน้าที่เป็นส่วนให้บริการในการเชื่อมต่อกับฐานข้อมูล อาทิ ดาต้าเบสอินเตอร์เฟส (Database Interface) และ ออบเจ็กท์สำหรับสร้างรายงาน (Report Generator Objects) เป็นต้น

2.5.3 ส่วนเก็บข้อมูล (Storage)

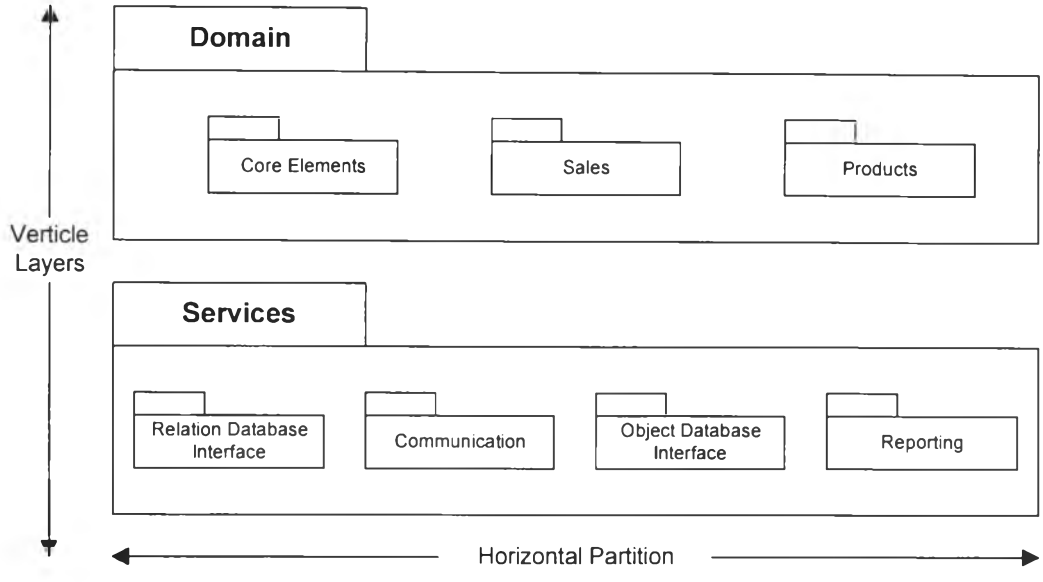
ส่วนเก็บข้อมูล หรือ ส่วนบริการข้อมูล (Data Service) เป็นวัตถุเพอซีซเทนท์ (ดูภาคผนวก ก) ซึ่งมีกลไกในการจัดการกับข้อมูลในฐานข้อมูล อาทิ ระบบฐานข้อมูลเชิงวัตถุ หรือ ระบบฐานข้อมูลเชิงสัมพันธ์ เป็นต้น

การออกแบบโดยใช้สถาปัตยกรรม 3 เทียร์นี้ช่วยทำให้แต่ละส่วนมีความเป็นอิสระและยืดหยุ่นสูง การบำรุงรักษากระทำได้ง่ายและมีผลกระทบ การใช้ภาษายูเอ็มแอลสามารถแสดงสถาปัตยกรรมแบบ 3 เทียร์โดยการใช้แพ็คเก็จ (Package) หรือกลุ่มคลาสซึ่งเป็นการจัดกลุ่มของคลาสในแนวตั้ง (Vertical Layer) โดยแบ่งตามหน้าที่ ขณะเดียวกันหากระบบมีขนาดใหญ่สามารถจัดกลุ่มคลาสในแนวนอน (Horizontal Partitions) ตามกลุ่มการทำงาน

จากรูปที่ 2.21 แสดงการแบ่งกลุ่มคลาสในแนวตั้งเป็นกลุ่มคลาสสำหรับ วัตถุโดเมนของปัญหา (Domain) และ กลุ่มคลาสสำหรับวัตถุบริการ (Services) ซึ่งจะทำการแบ่งกลุ่มวัตถุโดเมนและ บริการ ออกเป็นกลุ่มย่อยอันเป็นการแบ่งตามแนวนอน (Horizontal Partitions) ดังนี้

1) กลุ่มวัตถุโดเมนจะแบ่งออกเป็น กลุ่มงานหลัก (Core Element) การขาย (Sales) และกลุ่มสินค้า (Products)

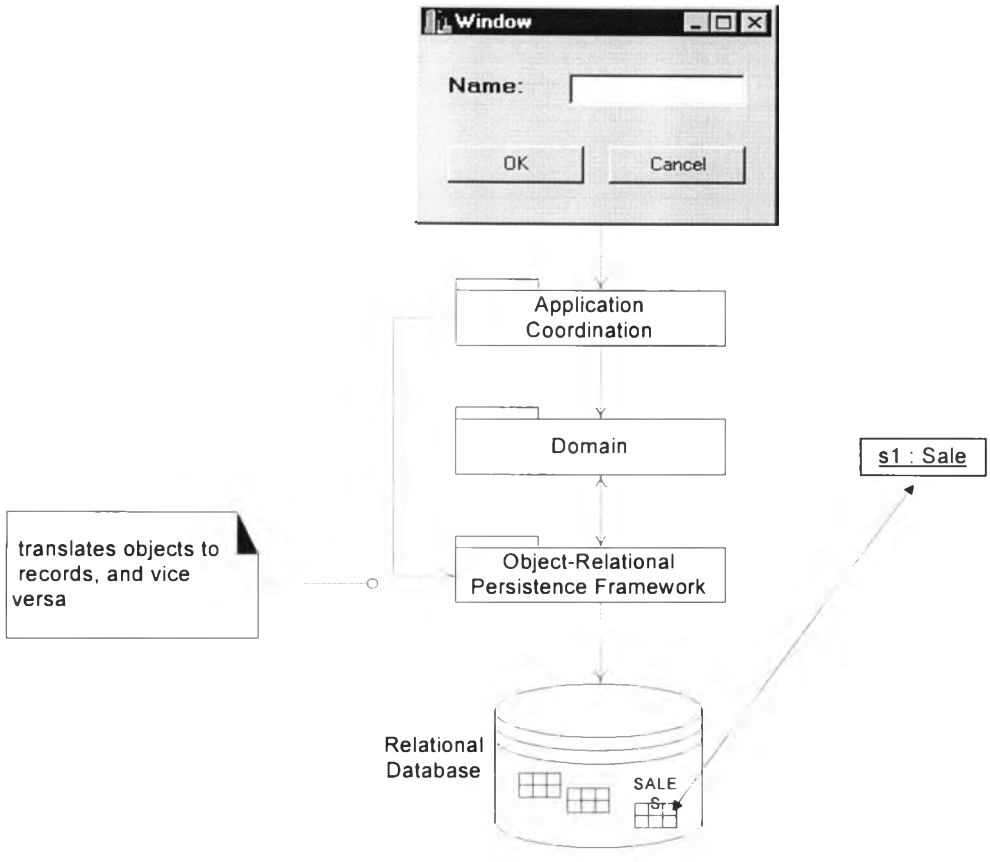
2) กลุ่มวัตถุบริการจะแบ่งออกเป็นกลุ่มอินเตอร์เฟสกับฐานข้อมูลเชิงสัมพันธ์ (Relational Database Interface) กลุ่มสื่อสาร (Communication) กลุ่มอินเตอร์เฟสกับฐานข้อมูลเชิงวัตถุ (Object Database Interface) และ กลุ่มออกรายงาน (Reporting)



รูปที่ 2.21 แสดงการจัดกลุ่มคลาสในแนวตั้งและแนวนอน[14]

2.6 วัตถุเพอซีซเทนท์และเฟรมเวอร์ค (Persistence Object and Frameworks)

ในแอปพลิเคชันที่มีการใช้ข้อมูลมาก ๆ อาทิ ระบบการขายที่ต้องมีการเก็บข้อมูลสินค้าและ ลูกค้าจำนวนมาก หรือ ระบบเฮลป์เดสก์ที่มีการเก็บคำร้องจำนวนมาก ดังนั้นระบบเหล่านี้มีความจำเป็นที่จะทำการดึงข้อมูลและเก็บข้อมูลลงในเพอซีซเทนท์ออบเจ็กท์ ซึ่งอาจเป็นฐานข้อมูลแบบออบเจ็กท์หรือรีเลชันแนล แฟ้มข้อมูลธรรมดา หรือ แฟ้มข้อมูลแบบอินเด็กซ์ (Index) ซึ่งไม่สามารถติดต่อได้โดยตรง เพราะกลไกในการทำงานจะเป็นแบบรายระเบียน (Record-Oriented) มิใช่วัตถุ (Object-Oriented) จึงมีผู้คิดค้นวิธีการในการแก้ไขปัญหานี้โดยใช้ เฟรมเวอร์ค (Framework) ซึ่งเป็นเซ็ตของคลาสซึ่งมีบริการสำหรับวัตถุเพอซีซเทนท์ ซึ่งส่วนมากแล้วจะพัฒนาเป็น เอพีไอ (API) สำหรับเชื่อมต่อกับระบบฐานข้อมูลแบบรีเลชันแนลหรือตัวจัดการข้อมูลแบบระเบียนต่างๆ โดยมีตัวอย่างที่ใช้กันมากก็ คือ โอดีบีซีของบริษัทไมโครซอฟต์ (Microsoft's ODBC) โดยเพอซีซเทนท์เฟรมเวอร์คจะทำหน้าที่แปลงจากวัตถุให้เป็นระเบียน และทำการแปลงจากระเบียนกลับเป็นวัตถุ (ดูรูปที่ 2.22ประกอบ) ในกรณีที่ใช้ระบบฐานข้อมูลเชิงวัตถุแล้วจะไม่มีความจำเป็นที่จะใช้เพอซีซเทนท์เฟรมเวอร์ค



รูปที่ 2.22 แสดงออบเจ็กท์-รีเลชันแนลเพอซีซเทนท์ออบเจ็กท์[3]

2.6.1 หน้าที่ของเพอซีชเทนท์เฟรมเวอร์ มี 2 หน้าที่ คือ

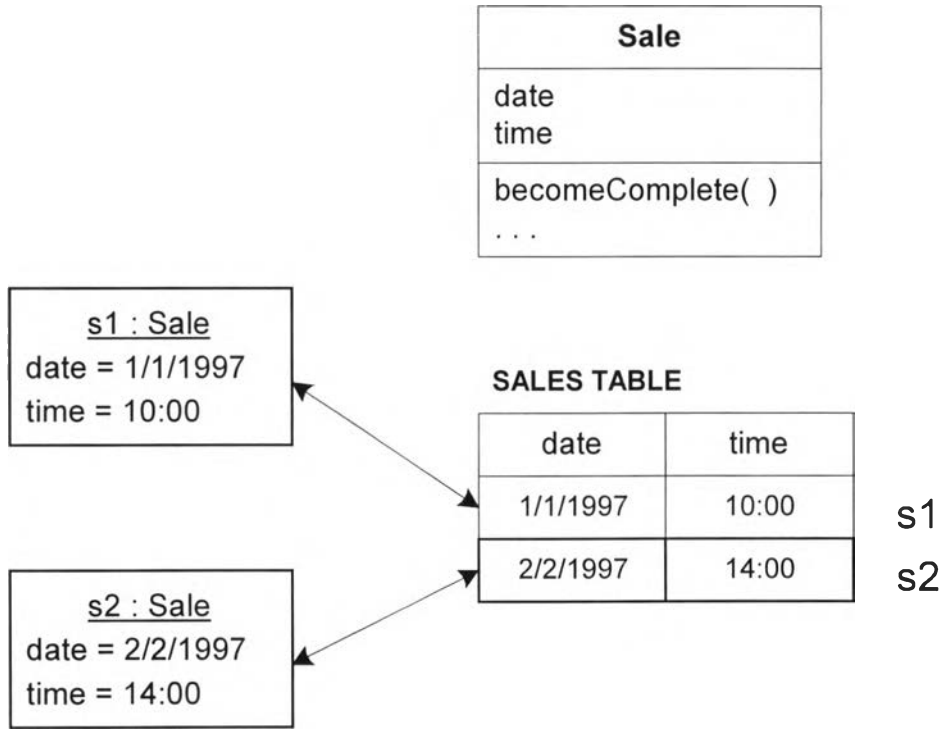
1) จัดเก็บและดึงวัตถุจากเพอซีชเทนออบเจ็กท์

2) ทำการยืนยัน (commit) และ ถอยกลับ (rolllback) ทรานแซกชัน

2.6.2 การแปลงจากระเบียนเป็นวัตถุ (Mapping Records to Objects) จะกระทำได้ดังขั้นตอนดังนี้

1) การแปลงจากระเบียนเป็นวัตถุเป็นเรื่องที่สามารถทำได้โดยง่าย โดยให้พิจารณาตาราง (Table) เป็นเซ็ตของวัตถุ และ ในแต่ละระเบียนจะแปลงเป็น 1 วัตถุโดยแอตทริบิวท์ในตารางจะเป็นแอตทริบิวท์ของวัตถุ ซึ่งในกรณีตรงกันข้ามการแปลงจากวัตถุเป็นระเบียนจะเป็นเรื่องที่ยุ่งยากกว่า โดยจะต้องทำการจับคู่วัตถุให้เป็นระเบียน 1 รายการแล้วจึงทำการแยกระเบียนออกเป็นระเบียนย่อย ๆ เพื่อเก็บลงในตาราง (ดูรูปที่ 2.23 และ 2.24 ประกอบ)

จากรูปที่ 2.23 ตารางรายการขาย (Sale) ประกอบไปด้วย 2 ระเบียนเพื่อแปลงเป็นวัตถุรายการขาย s1 โดยจะแปลงวันที่ และ ไปเป็นแอตทริบิวท์ของวัตถุ s1 (1/1/1997, 10:00) และ สำหรับวัตถุ s2 ก็จะทำการแปลงได้เป็น s2 (2/2/1997, 14:00)

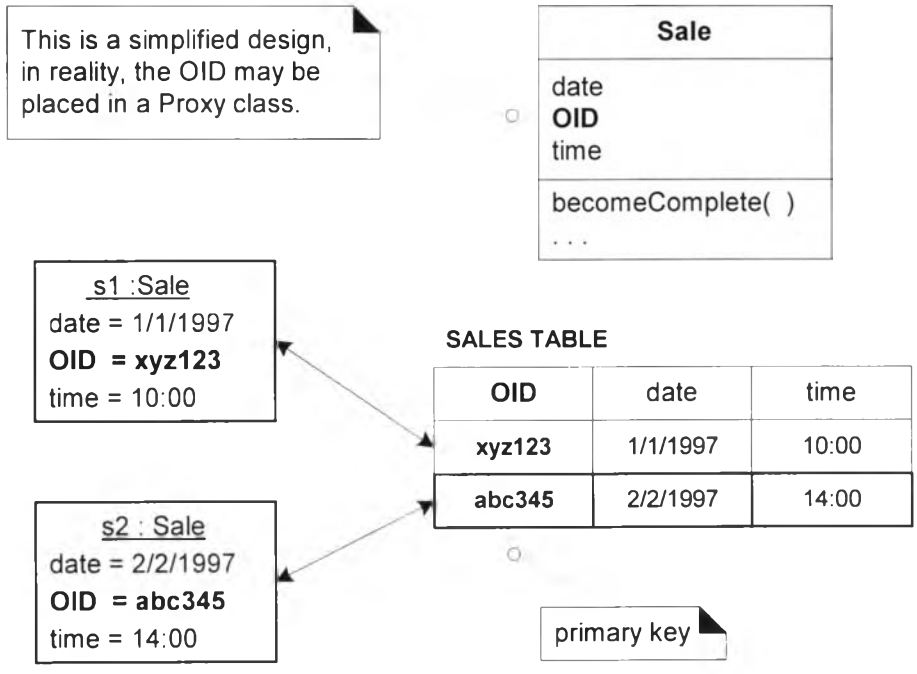


รูปที่ 2.23 แสดงการแปลงระหว่างวัตถุและระเบียน[5]

2) การจำแนกเอกลักษณ์วัตถุ (Object Identify : OID)

การระบุค่าวัตถุจะใช้ค่าที่เรียกว่า โอไอดี (OID) เป็นค่าที่ใช้ในการแยกแยะแต่ละวัตถุออกจากกัน และ การเข้าถึงวัตถุ ในการออกแบบค่าโอไอดี (OID) มักกำหนดให้เป็นตัวอักษรปนตัวเลข (Alpha Numeric) ตัวอย่างเช่น ไมโครซอฟต์ ได้มีการกำหนดค่ายูยูไอดี (Universally Unique Identifier) และได้สร้างเอพีไอสำหรับสร้างค่านี้โดยอัตโนมัติ ซึ่งหากสามารถกำหนดให้ฐานข้อมูลแบบรีเรชันแนลใช้ค่าของโอไอดีแทนค่าคีย์หลัก (Primary Key) แล้วจะสามารถที่จะทำการแปลงค่าโอไอดีไปเป็นระเบียนที่มีโอไอดีเป็นกุญแจหลักได้

จากรูปที่ 2.24 แสดงการแปลงจากระเบียน 2 รายการคือ (xyz123, 1/1/1997, 10:00) ซึ่งมีค่าคีย์หลักเป็น 'xyz123' และ (abc345, 2/2/1997, 14:00) ซึ่งมีค่าคีย์หลักเป็น 'abc345' ไปเป็นวัตถุ s1 ซึ่งจะใช้ค่าคีย์หลัก 'xyz123' เป็นค่าโอไอดี และ วัตถุ s2 ซึ่งใช้ค่าคีย์หลัก 'abc345' เป็นค่าโอไอดี



รูปที่ 2.24 แสดงการใช้โอไอดีในการเชื่อมระหว่างวัตถุและระเบียน[5]