บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะอธิบายแนวคิดและทฤษฎีที่เกี่ยวข้องกับงานวิจัยนี้ โดยส่วนแรกเป็นแนวคิดและทฤษฎีที่เกี่ยวข้อง ซึ่งมีจุดประสงค์เพื่อให้เข้าใจหลักการและทฤษฎีต่างๆที่เกี่ยวข้องกับงานวิจัยนี้ ได้แก่ เครื่องมือวิศวกรรมซอฟต์แวร์ใช้คอมพิวเตอร์ช่วย วิธีการเชิงวัตถุ และภาษายูเอ็มแอล ส่วนที่สองอธิบายถึงเอกสารและงานวิจัยที่เกี่ยวข้อง ซึ่งมีจุดประสงค์เพื่อให้เห็นถึงงานวิจัยที่ได้เคยทำมาแล้วและข้อจำกัดของงานวิจัยในอดีต โดยได้กล่าวถึงการสำรวจการใช้งานเคสทูล การใช้เคสทูลเพื่อการพัฒนาซอฟต์แวร์เชิงวัตถุ การนำเสนอคุณสมบัติของเคสทูล การเปรียบเทียบเคสทูลที่ใช้ภาษายูเอ็มแอลในการวิเคราะห์และออกแบบระบบ และแนวคิดในการนำเสนอรูปแบบของเคสทูลในอนาคต

2.1 แนวคิดและทฤษฎี

2.1.1 เครื่องมือวิศวกรรมซอฟต์แวร์ใช้คอมพิวเตอร์ช่วย (Computer Aided Software Engineering (CASE) Tool)

เคสทูล (CASE Tool - Computer Aided Software Engineering Tool) คือเครื่องมือช่วยในกระบวนการต่างๆในการพัฒนาซอฟต์แวร์ตามแนวทางวิศวกรรมซอฟต์แวร์ (Software Engineering) (Software Engineering Institute, 2004) เพื่อเพิ่มประสิทธิภาพของกระบวนการพัฒนาซอฟต์แวร์ให้รวดเร็วและถูกต้องมากยิ่งขึ้น เคสทูลได้ถูกนำเสนอในปีคริสต์ศักราช 1970 โดยจุดประสงค์หลักของการนำเคสทูลมาใช้คือ ช่วยเพิ่มผลผลิต ลดเวลา และลดค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ รวมถึงการเพิ่มคุณภาพของกระบวนการพัฒนาซอฟต์แวร์ โดยเครื่องมือนี้จะช่วยให้การพัฒนาซอฟต์แวร์แต่ละขั้นตอนเป็นไปโดยอัตโนมัติมากยิ่งขึ้น

พัฒนาการของเคสทูลนั้นเริ่มจากการพัฒนาเครื่องมือสำหรับการสร้างและจัดทำเอกสารและพัฒนาการใช้เทคนิคทางกราฟิกเข้ามาช่วย โดยมีการพัฒนาอย่างต่อเนื่องดังนี้ (Barclay & Padusenko, 2004)

ช่วงต้นปีคริสต์ศักราช 80:

- เริ่มพัฒนาในรูปของคอมพิวเตอร์ช่วยการจัดทำเอกสาร (Computer Aided Documentation) ซึ่งเน้นการทำงานด้านเอกสารและเริ่มมีเครื่องมือในรูปแบบของ

การวิเคราะห์แผนภาพ (Diagramming Analysis) และเครื่องมือการออกแบบ (Design Tool)

ช่วงกลางปีคริสต์ศักราช 80:

- มีการพัฒนาเคสทูลที่สามารถตรวจสอบการวิเคราะห์และออกแบบอย่างอัตโนมัติ (Automated Analysis and Design Checking)

ช่วงปลายปีคริสต์ศักราช 80:

- มีการพัฒนาเคสทูลที่สามารถสร้างรหัสจากข้อกำหนดการออกแบบ (Design Specification) อย่างอัตโนมัติ (Automated Code Generation)

ช่วงปีคริสต์ศักราช 90:

- มีการพัฒนาเคสทูลให้มีความฉลาดในการตรวจสอบความผิดพลาดของการออกแบบและมีการสนับสนุนคุณสมบัติบางประการของการพัฒนาซอฟต์แวร์เชิงวัตถุ

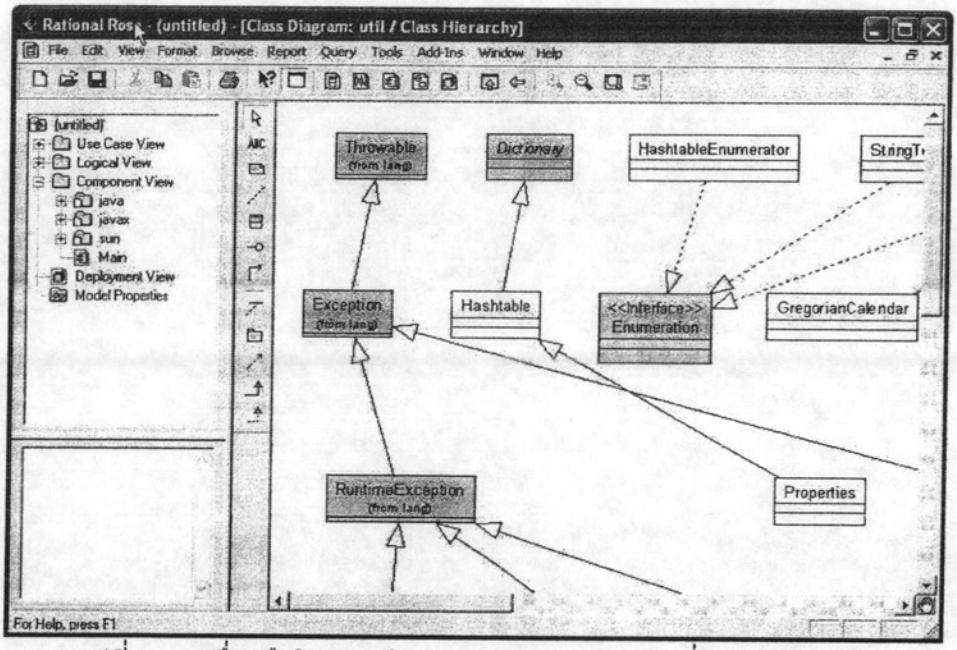
องค์ประกอบของเคสทูล (Components of CASE Tool)

องค์ประกอบของเคสทูลที่สำคัญ (Hoffer et al; 2002) ได้แก่

- เครื่องมือสร้างแผนภาพ (Diagramming Tools)
- เครื่องมือสร้างฟอร์มและรายงาน (Form and Report Generator)
- เครื่องมือในการวิเคราะห์ (Analysis Tools)
- รีโพสิโทรีกลาง (Central Repository)
- เครื่องมือสร้างเอกสาร (Document Generator)
- เครื่องมือสร้างรหัส (Code Generator)

เครื่องมือสร้างแผนภาพ (Diagramming Tools)

เป็นเครื่องมือที่ช่วยงานด้านการสร้างแผนภาพต่างๆ เพื่อช่วยควบคุมโครงสร้างของซอฟต์แวร์ โดยจะแสดงในรูปของกราฟิกหรือรูปภาพซึ่งทำให้งานเป็นรูปธรรมขึ้น โดยแผนภาพต่างๆ สามารถนำมาใช้เป็นแนวทางในการพัฒนาซอฟต์แวร์ต่างๆได้ รูปที่ 2-1 แสดงเครื่องมือในการสร้างแผนภาพของเคสทูลเรชั่นนัลโรส 2003



รูปที่ 2-1: เครื่องมือในการสร้างแผนภาพของเคสทูลเรชั่นนัลโรส 2003

เครื่องมือสร้างฟอร์มและรายงาน (Form and Report Generator)

เป็นเครื่องมือเพื่อใช้สร้างฟอร์มและรายงานอัตโนมัติ ดังนี้

1 ช่วยสร้าง ปรับปรุง และทดสอบฟอร์มและรายงานที่เป็นต้นแบบของหน้าจอคอมพิวเตอร์และรายงานต่างๆ

2 กำหนดการจัดเก็บ และการแสดงรายการข้อมูล (Data Item) ในแต่ละฟอร์มหรือรายงาน

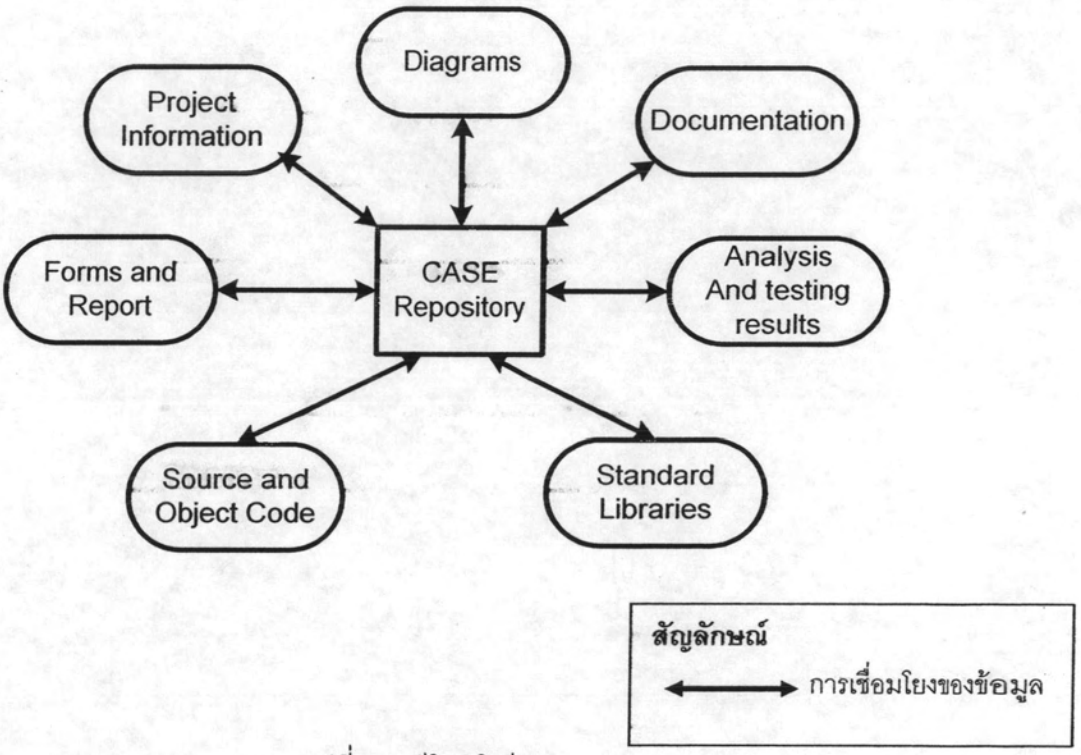
เครื่องมือในการวิเคราะห์ (Analysis Tools)

จุดประสงค์หลักของเคสทูลอีกอย่างคือการนำมาใช้ในการสร้างซอฟต์แวร์ที่มีขนาดใหญ่และมีความซับซ้อน ดังนั้นเมื่อสร้างแผนภาพต่างๆจากเครื่องมือการสร้างแผนภาพและสร้างฟอร์มและรายงานแล้ว เคสทูลยังมีเครื่องมือในการวิเคราะห์ ซึ่งเป็นตัวช่วยในการวิเคราะห์ความถูกต้องของแผนภาพว่ามีความไม่ถูกต้องตรงกัน (Inconsistencies) และความซ้ำซ้อน (Redundancies) หรือไม่ และสิ่งที่นักวิเคราะห์และออกแบบซอฟต์แวร์อาจจะละเลยไปในการสร้างแผนภาพ โดยจะแสดงผลเป็นรายงานของการวิเคราะห์

รีโพสิโทรีกลาง (Central Repository)

รีโพสิโทรีเป็นอีกส่วนหนึ่งที่มีสำคัญและเป็นประโยชน์อย่างมากเมื่อมีการใช้เคสทูล รีโพสิโทรีเป็นที่เก็บข้อมูลต่างๆไว้ด้วยกัน ซึ่งในเคสทูลหนึ่งๆจะสามารถแลกเปลี่ยนข้อมูลระหว่างเครื่องมือต่างๆได้โดยไม่ต้องแปลงรูปแบบของข้อมูลที่แตกต่างของเครื่องมือแต่ละส่วนให้อยู่ในรูปเดียวกัน

การเก็บข้อมูลไว้ที่เดียวกันในรีโพสิโทรี (Repository) เพื่อให้เครื่องมือต่างๆสามารถแลกเปลี่ยนข้อมูลกันซึ่งจะสนับสนุนในวงจรการพัฒนาซอฟต์แวร์ให้เป็นไปได้อย่างต่อเนื่องและเป็นระบบ ดังแสดงในรูปที่ 2-2



รูปที่ 2-2: รีโพสิโทรีกลาง

(ดัดแปลงมาจาก Hoffer et al, 2002)

รีโพสิโทรีกลาง (Central Repository) ของเคสทูลประกอบด้วยส่วนสำคัญ 2 ส่วน คือ รีโพสิโทรีของสารสนเทศ (Information Repository) และพจนานุกรมข้อมูล (Data Dictionary)

รีโพสิโทรีของสารสนเทศ (Information Repository) เป็นเครื่องมืออัตโนมัติซึ่งใช้ในการจัดการข้อมูลที่เกี่ยวกับสารสนเทศของธุรกิจ (Business Information) และกลุ่มของระบบงาน

(Application Portfolio) โดยสารสนเทศของธุรกิจเป็นข้อมูลขององค์กร ในขณะที่กลุ่มของระบบงานประกอบด้วยโปรแกรมประยุกต์ที่ใช้เพื่อจัดการสารสนเทศของธุรกิจ

พจนานุกรมข้อมูล (Data Dictionary) เป็นส่วนที่ใช้จัดการและควบคุมการเข้าถึงรีโพสิโทรีของสารสนเทศ โดยเป็นเครื่องมือที่ช่วยอำนวยความสะดวกในการบันทึก เก็บรักษา และประมวลผลรายละเอียดของข้อมูลที่สำคัญขององค์กรและทรัพยากรที่ใช้ในการประมวลผลข้อมูล (Hoffer et al, 2002) โดยรายการในพจนานุกรมมีมาตรฐานของคำนิยาม ดังนี้

1) ชื่อของข้อมูล (Element) และชื่อเหมือนอื่นๆ
2) คำอธิบายความหมายของข้อมูล
3) ข้อมูลที่เกี่ยวข้องกัน
4) ประเภทของข้อมูล
5) ขนาดและค่าข้อมูลต่างๆที่ยอมรับได้
6) ข้อมูลสำคัญอื่นๆที่ใช้ในการประมวลผลข้อมูล

เครื่องมือสร้างเอกสาร (Document Generator)

ในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์นั้นจะมีการจัดสร้างเอกสารต่างๆ และมีการจัดเก็บการสร้างเอกสาร เอกสารสามารถสร้างโดยขึ้นอยู่กับองค์ประกอบที่มีอยู่ในรีโพสิโทรี โดยปกติเอกสารของการพัฒนาระบบจะประกอบด้วย ข้อกำหนดความต้องการ (Requirements Specification) แผนภาพของข้อมูล ข้อกำหนดของโปรแกรม (Program Specification) คู่มือสำหรับผู้ใช้งาน (User Manual) รวมถึงโปรแกรมประยุกต์และเอกสารอ้างอิงอื่นๆ (Application and Reference Material) ซึ่งปัญหาที่พบจากการที่ไม่ได้ใช้เคสทูลคือความไม่สมบูรณ์ของเอกสารที่สร้าง ทำให้การบำรุงรักษาซอฟต์แวร์นั้นยุ่งยากและเอกสารส่วนใหญ่ถูกสร้างหลังจากการพัฒนาระบบแล้ว ทำให้ขาดการตรวจสอบและไม่มีคุณภาพ

เครื่องมือสร้างเอกสารในสภาพแวดล้อมของเคสทูลจะช่วยสร้างเอกสารดังกล่าว โดยการจัดสร้างแม่แบบที่เป็นมาตรฐานให้สำหรับเอกสารแต่ละประเภท ซึ่งช่วยให้การทำเอกสารมีความสมบูรณ์และมีมาตรฐาน ส่งผลถึงการช่วยให้การบำรุงรักษาซอฟต์แวร์นั้นเป็นไปได้ง่ายและถูกต้องยิ่งขึ้น

เครื่องมือสร้างรหัส (Code Generator)

เครื่องมือสร้างรหัส (Code Generator) คือโปรแกรมอัตโนมัติที่สร้างรหัสต้นฉบับจากแผนภาพและฟอร์ม โดยรหัสที่สร้างได้นั้นจะช่วยให้การเขียนโปรแกรมของทีมพัฒนามีมาตรฐานและยังช่วยแก้ไขในกรณีที่สภาพแวดล้อมของระบบเปลี่ยนแปลงตามแพลตฟอร์มของฮาร์ดแวร์หรือระบบปฏิบัติการ ดังนั้นเครื่องมือสร้างรหัสส่วนมากถูกออกแบบมาเพื่อสร้างรหัสต้นฉบับ (Source Code) และการสร้างคำสั่งเพื่อสร้างฐานข้อมูล (Database Script)

2.1.2 **วิธีการเชิงวัตถุ** (Object Oriented Approach)

แนวคิดของวิธีการเชิงวัตถุถูกนำเสนอในการพัฒนาซอฟต์แวร์ตั้งแต่ปีคริสต์ศักราช 1966 แต่ได้รับความนิยมอย่างจริงจังในปีคริสต์ศักราช 1990 เป็นต้นมา ซึ่งวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุนั้นมีวิธีการมององค์ประกอบต่างๆของซอฟต์แวร์เป็นวัตถุ แทนการมองเชิงโครงสร้างที่จะมองกระบวนการและข้อมูลแยกกัน

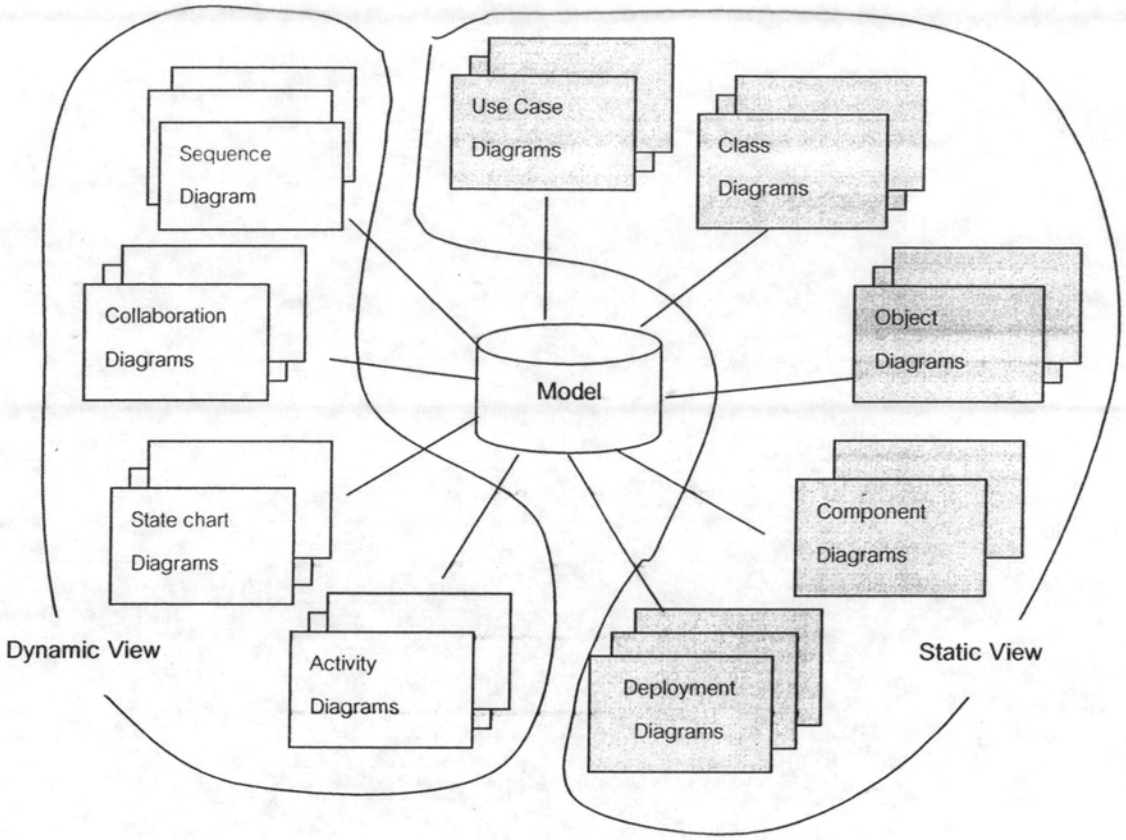
วัตถุ (Object) หมายถึง สิ่งของหรือเหตุการณ์ที่มีสภาวะสารสนเทศ (Information State) และมีการนิยามการปฏิบัติการหรือพฤติกรรม (Operation หรือ Behavior) ที่กระทำกับวัตถุ สภาวะของวัตถุถูกแสดงด้วยคุณสมบัติของวัตถุ (Object Attribute) ปฏิบัติการต่างๆของวัตถุอาจถูกจัดเตรียมสำหรับให้บริการกับวัตถุอื่นที่ร้องขอ เพื่อต้องการให้วัตถุดำเนินกิจกรรมบางอย่าง หรือเพื่อให้บริการเฉพาะภายในวัตถุ ดังนั้นลักษณะของวัตถุถูกกำหนดโดยกลุ่มปฏิบัติการและสภาวะต่างๆที่มีในวัตถุ ตัวอย่างเช่น บัญชีเงินฝาก (Bank Account) เป็นวัตถุประกอบด้วยข้อมูลที่เกี่ยวกับบัญชีเงินฝาก คือ ชื่อเจ้าของบัญชี (Account Name) และยอดเงินในบัญชี (Balance) เป็นต้น และอีกส่วนคือการปฏิบัติการหรือพฤติกรรม เช่น การถอนเงิน (Withdraw) และการฝากเงิน (Deposit) เป็นต้น (สมนึก คีรีโต, 2545)

เนื่องจากการใช้หลักการมองปัญหาต่างๆเป็นวัตถุ โดยแต่ละวัตถุมีหน้าที่การทำงานเฉพาะและแตกต่างกัน ทำให้สะดวกในการพัฒนาซอฟต์แวร์ นอกจากนั้นการพัฒนาซอฟต์แวร์เชิงวัตถุยังมีคุณสมบัติที่รองรับการพัฒนาซอฟต์แวร์ที่ดีอีกหลายประการ เช่น แนวคิดที่จะพยายามนำซอฟต์แวร์ที่ได้รับการพัฒนามาแล้วมาปรับปรุงหรือพัฒนาต่อให้สามารถนำกลับมาใช้ใหม่ (Reuse) ซึ่งส่งผลให้วิธีการพัฒนาซอฟต์แวร์เชิงวัตถุนั้นมีแนวโน้มถูกนำมาใช้ในปัจจุบัน และคาดว่าจะเป็นรูปแบบของการพัฒนาต่อไปในอนาคต (สุชาย ธนวเสถียร และคณะ, 2542)

2.1.3 **ภาษายูเอ็มแอล** (UML - Unified Modeling Language)

จากแนวคิดการพัฒนาซอฟต์แวร์เชิงวัตถุที่สามารถช่วยแก้ปัญหาในการพัฒนาซอฟต์แวร์ที่มีขนาดใหญ่และมีความซับซ้อนได้อย่างมีประสิทธิภาพ ทำให้แนวคิดเชิงวัตถุได้รับความนิยมจากนักพัฒนาซอฟต์แวร์ในปัจจุบัน จึงมีผู้คิดค้นวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุขึ้นหลายวิธี ทำให้ผู้ใช้เกิดความสับสนว่าควรใช้วิธีใดเนื่องจากวิธีของนักพัฒนาระบบจากวิธีต่างๆนั้นใช้สัญลักษณ์ที่ซับซ้อนและสื่อความหมายที่แตกต่างกันไป ซึ่งเมื่อเลือกวิธีการวิเคราะห์ออกแบบของผู้ใดก็จำเป็นต้องใช้วิธีการนำเสนอผลลัพธ์ของการวิเคราะห์และออกแบบตามที่ผู้สร้างกำหนดด้วย โดยแต่ละวิธีนั้นต่างก็มีการกำหนดสัญลักษณ์รูปภาพ (Notation) แตกต่างกันไป ซึ่งรวมทั้งจุดเด่นและจุดด้อยของแต่ละวิธีการก็แตกต่างกันไปด้วยเช่นกัน ทำให้เกิดสภาพที่เป็นสงครามวิธีการ (Method War) คือ จึงเป็นการยากที่จะค้นหาวิธีการที่ดีที่สุด

ดังนั้นผู้พัฒนาระบบที่มีประสบการณ์มักจะใช้หลายวิธีการร่วมกันอันเป็นที่มาของการรวมกันของวิธีการต่างๆ ในช่วงแรกเป็นการรวมกันระหว่างวิธีการบูช (Booch Method) ของ Grady Booch และโอเอ็มที (OMT - Object Modeling Technique) ของ James Rumbaugh โดยมีเป้าหมายให้ได้มาซึ่งกระบวนการพัฒนาซอฟต์แวร์เชิงวัตถุที่เป็นหนึ่งเดียว (Unified Method) โดยนำเอาวิธีการของบูชและโอเอ็มทีมารวมกันและปรับปรุงใหม่ โดยมีความตั้งใจให้คล้ายกับการสร้างภาษาพีแอลวัน (PL/1) คือ รวบรวมเอาภาษาทั้งหลายมาเป็นภาษาเดียว ต่อมา Ivar Jacobson ผู้พัฒนากระบวนการโอโอเอสอี (OOSE - Object Oriented Software Engineering) ได้เข้าร่วมโครงการดังกล่าวทำให้ได้ภาษาในการโมเดล (Modeling Language) ขึ้นใหม่เรียกว่าภาษายูเอ็มแอล (UML - Unified Modeling Language) เป้าหมายของภาษายูเอ็มแอลก็เพื่อจะเป็นสัญลักษณ์ที่ช่วยในการวิเคราะห์และออกแบบซอฟต์แวร์เชิงวัตถุที่เป็นมาตรฐานกลาง ซึ่งช่วยให้ผู้พัฒนาสามารถใช้แผนภาพไดอะแกรมสำหรับอธิบายและสื่อสารความต้องการของระบบในระดับ (Level) ต่างๆ และแสดงระบบในมุมมอง (View) ต่างๆกัน โดยมีแผนภาพแสดงระบบในมุมมองต่างๆ โดยแบ่งออกเป็น 2 มุมมองคือ มุมมองสถิต (Static View) และมุมมองพลวัต (Dynamic View) ดังแสดงในรูปที่ 2-3



รูปที่ 2-3: แผนภาพต่างๆที่ใช้ในภาษายูเอ็มแอล

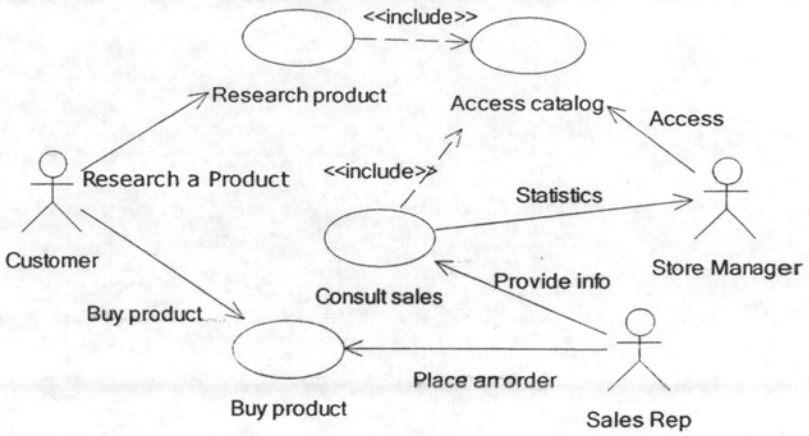
(ดัดแปลงมาจาก http://www-306.ibm.com/software/rational/uml/, ปี 2004)

มุมมองสถิต (Static View)

คือแผนภาพ (Diagram) ที่แสดงคุณสมบัติในเชิงสถิต (Static) ของระบบ ซึ่งมุมมองสถิตประกอบด้วย แผนภาพยูสเคส (Use Case Diagram) แผนภาพคลาส (Class Diagram) แผนภาพวัตถุ (Object Diagram) แผนภาพคอมโพเนนท์ (Component Diagram) และแผนภาพดีพลอยเมนท์ (Deployment Diagram)

1.) แผนภาพยูสเคส (Use Case Diagram)

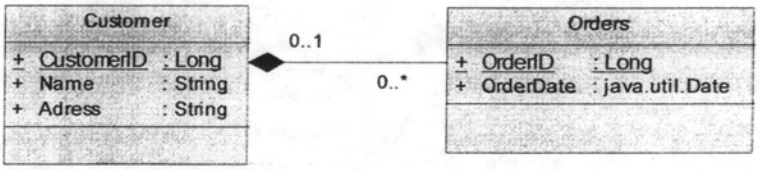
เป็นแผนภาพที่แสดงฟังก์ชันในมุมมองของผู้ใช้ระบบ ซึ่งจะเรียกว่า แอ็คเตอร์ (Actor) ยูสเคสจะสร้างขึ้นมาเพื่อใช้ในการศึกษาทำความเข้าใจ และกำหนดความต้องการของระบบ และยังสามารถใช้ในการตรวจสอบและสร้างกรณีทดสอบระบบได้อีกด้วย โดยในรูปที่ 2-4 แสดงตัวอย่างของแผนภาพยูสเคส



รูปที่ 2-4: แผนภาพยูสเคส

2.) แผนภาพคลาส (Class Diagram)

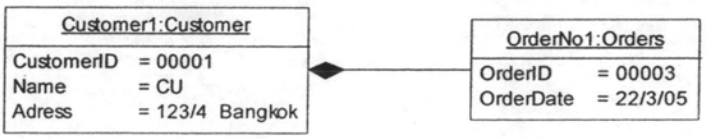
เป็นแผนภาพที่แสดงการมีอยู่ของคลาส (Class) ต่างๆ ซึ่งเปรียบเหมือนแม่แบบ (Template) ในการสร้างวัตถุ (Object) และความสัมพันธ์ของคลาสเหล่านั้น ซึ่งประกอบไปด้วยคลาสและความสัมพันธ์ระหว่างคลาส โดยในรูปที่ 2-5 แสดงตัวอย่างแผนภาพคลาส



รูปที่ 2-5: แผนภาพคลาส

3.) แผนภาพวัตถุ (Object Diagram)

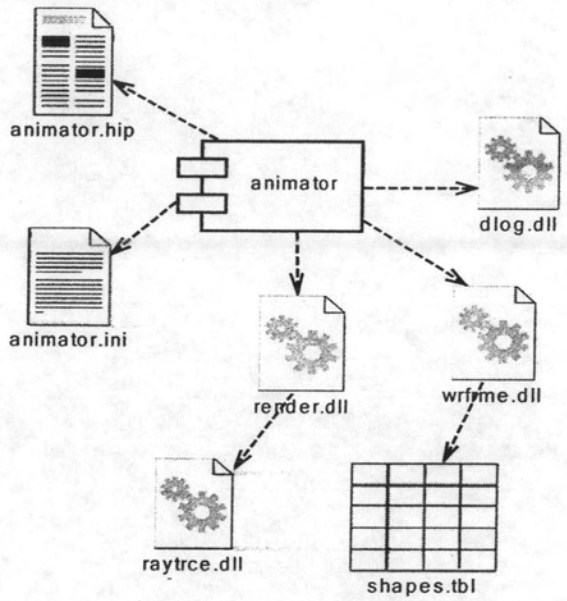
เป็นแผนภาพที่แสดงวัตถุและความสัมพันธ์ระหว่างวัตถุ โดยในรูปที่ 2-6 แสดงตัวอย่างแผนภาพวัตถุ



รูปที่ 2-6: แผนภาพวัตถุ

4.) แผนภาพคอมโพเนนท์ (Component Diagram)

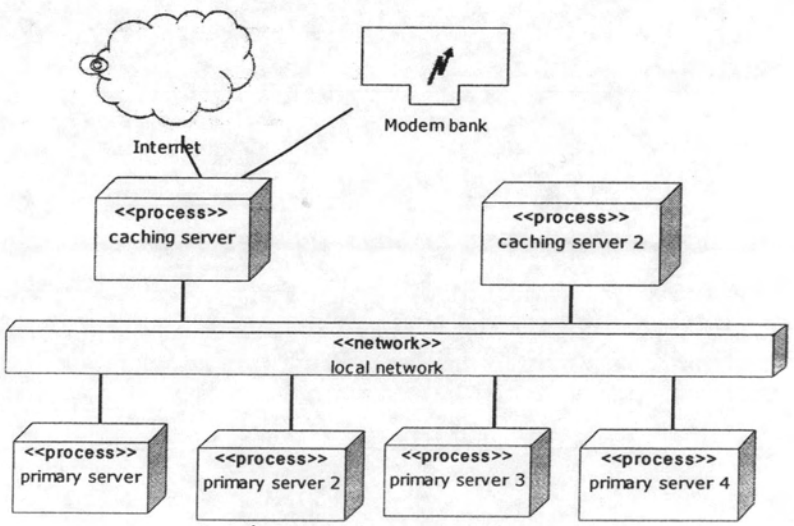
เป็นแผนภาพที่แสดงโครงสร้างของระบบในระดับโปรแกรม หรือ แพ็กเกจ (กลุ่มคลาส) ความสัมพันธ์และการเชื่อมต่อโปรแกรม โดยในรูปที่ 2-7 แสดงตัวอย่างของแผนภาพคอมโพเนนท์



รูปที่ 2-7: แผนภาพคอมโพเนนท์

5.) แผนภาพดีพลอยเมนท์ (Deployment Diagram)

เป็นแผนภาพที่แสดงการกระจายส่วนประกอบที่พัฒนาแล้วลงในฮาร์ดแวร์ต่างๆ โดยในรูปที่ 2-8 แสดงตัวอย่างของแผนภาพดีพลอยเมนท์



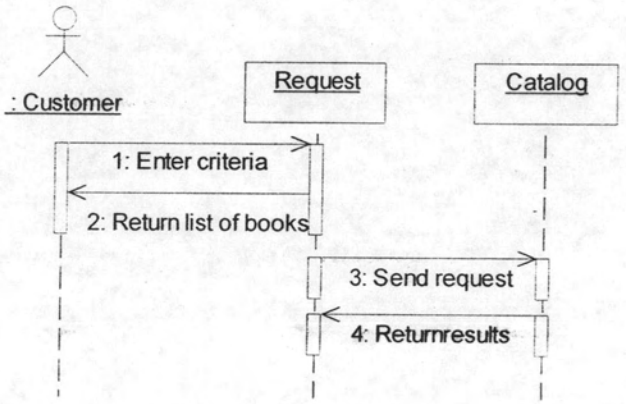
รูปที่ 2-8: แผนภาพดีพลอยเมนท์

มุมมองพลวัต (Dynamic View)

คือแผนภาพ (Diagram) ที่แสดงคุณสมบัติในเชิงพลวัต (Dynamic) ของระบบ โดยเป็นการแสดงถึงสิ่งที่เกิดขึ้นจากกิจกรรมของคลาส (Class) ต่างๆที่มีในระบบ ซึ่งมุมมองพลวัตประกอบด้วย แผนภาพซีเควน (Sequence Diagram) แผนภาพคอลแลบบอเรชัน (Collaboration Diagram) แผนภาพเสตทชาร์ท (State Chart Diagram) และแผนภาพแอ็กทีวิตี (Activity Diagram)

1.) แผนภาพซีเควน (Sequence Diagram)

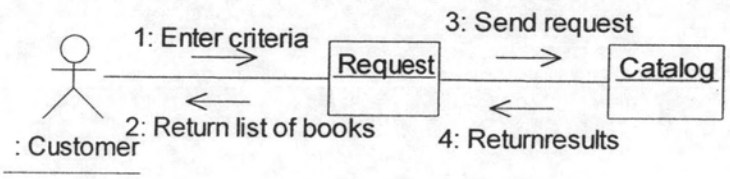
เป็นแผนภาพที่แสดงพฤติกรรมของระบบและการทำงานของระบบตามลำดับของเวลาที่เกิด โดยในรูปที่ 2-9 แสดงตัวอย่างของแผนภาพซีเควน



รูปที่ 2-9: แผนภาพซีเควน

2.) แผนภาพคอลแลบอเรชัน (Collaboration Diagram)

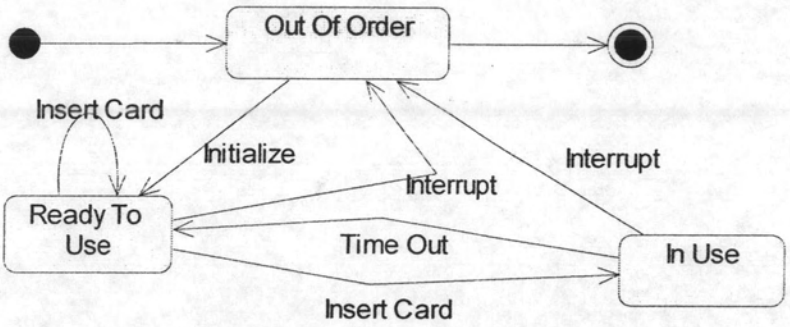
เป็นแผนภาพที่สร้างขึ้นเพื่อแสดงพฤติกรรมของระบบ แผนภาพนี้ต่างกับแผนแบบซีเควนตรงที่แผนภาพคอลแลบอเรชันจะนำเสนอการส่งผ่านข้อความของระบบ ส่วนแผนภาพซีเควนจะเน้นที่ลำดับและเวลาของการทำงาน โดยในรูปที่ 2-10 แสดงตัวอย่างของแผนภาพคอลแลบอเรชัน



รูปที่ 2-10: แผนภาพคอลแลบอเรชัน

3.) แผนภาพเสตทชาร์ท (State Chart Diagram)

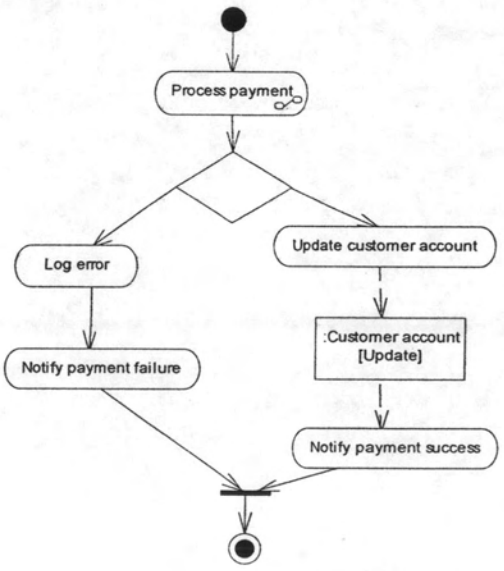
เป็นแผนภาพที่แสดงสถานะทั้งหมดของแต่ละคลาสหรือแต่ละวัตถุที่สถานะ (State) ต่างๆ ที่คลาสหนึ่งคลาสจะเป็นได้ในระหว่างช่วงชีวิตในการตอบสนองต่อเหตุการณ์ (Event) ที่เกิดขึ้น โดยในรูปที่ 2-11 แสดงตัวอย่างแผนภาพเสตทชาร์ท



รูปที่ 2-11: แผนภาพเสตทชาร์ท

4.) แผนภาพแอ็กทีวิตี้ (Activity Diagram)

เป็นแผนภาพที่ใช้แสดงผังงานของระบบโดยสามารถแสดงการทำงานร่วมกันของวัตถุในรูปแบบที่คล้ายกับแผนภาพการทำงาน (Flow Chart) โดยในรูปที่ 2-12 แสดงแสดงตัวอย่างแผนภาพแอ็กทีวิตี้



รูปที่ 2-12: แผนภาพแอ็กทีวิตี้

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวกับการสำรวจและนำเสนอแนวทางในการพัฒนาเคสทูลนั้นมีหลายเรื่องที่น่าสนใจ โดยจะกล่าวถึงงานวิจัยและบทความที่น่าสนใจ 9 เรื่อง ดังนี้

2.2.1 งานวิจัยของ Lending และ Norman (1998) ซึ่งได้สำรวจการใช้งานเคสทูลจากคำถามวิจัยจำนวนหนึ่ง ได้แก่ องค์กรควรมีการใช้งานเคสทูลหรือไม่ ถ้าใช้ คุณสมบัติ (Features) ใดในเคสทูลที่ควรใช้ และได้สำรวจเหตุผลว่าเคสทูลนั้นเปลี่ยนงานของนักพัฒนาระบบ (Systems Developer) มากน้อยเพียงไร รวมถึงนักพัฒนาระบบมีทัศนคติในทางลบหรือไม่กับการใช้งานเคสทูล โดยผลการสำรวจนักพัฒนาระบบ 233 คนจาก 7 องค์กร พบว่ามี 4 องค์กรที่ใช้เคสทูลและในองค์กรที่มีการใช้นั้นใช้โดยนักพัฒนาระบบเพียงบางคนเท่านั้น และพบว่านักพัฒนาระบบที่ใช้เคสทูลใช้ระเบียบวิธีที่เป็นมาตรฐานมากกว่านักพัฒนาระบบที่ไม่ใช้เคสทูล โดยนักพัฒนาระบบมีเหตุผลที่แตกต่างกันไปในการใช้และไม่ใช้เคสทูล และมีการสอบถามถึงคุณสมบัติของเคสทูลที่ใช้งาน ซึ่งพบว่ามีการใช้เพียงบางคุณสมบัติของเคสทูลเท่านั้น คุณสมบัติของเคสทูลที่ใช้ในแบบสอบถามของการวิจัยนี้เป็นคุณสมบัติตามเอฟซีทีเอ็ม (FCTM - Functional CASE Technology Model) นำเสนอโดย Henderson และ Cooprider (1990) โดยเอฟซีทีเอ็มได้แบ่งคุณสมบัติออกเป็น 5 กลุ่ม คือ ฟังก์ชันการนำเสนอ (Representation Functionality) ฟังก์ชันการวิเคราะห์ (Analysis Functionality) ฟังก์ชันการแปลง (Transformation Functionality) ฟังก์ชันการควบคุม (Control Functionality) และฟังก์ชันการร่วมมือ (Cooperative Functionality) ซึ่งมีจำนวนรายการคุณสมบัติทั้งหมด 58 รายการ (ภาคผนวก ค) โดยงานวิจัยนี้ใช้เพียง 2 กลุ่มในการสอบถามคือ ฟังก์ชันการวิเคราะห์ และฟังก์ชันการแปลง ซึ่งมี 28 รายการ โดยสรุปผลการสำรวจการใช้งานในรูปของสถิติเชิงพรรณนา (Descriptive Statistics) ซึ่งจากการสำรวจพบว่านักพัฒนาระบบที่ใช้งานเคสทูลมีความพึงพอใจและรู้สึกว่าเคสทูลมีประโยชน์อย่างมากในการพัฒนาซอฟต์แวร์ และพบว่าองค์กรที่สำรวจเกินครึ่งมีการใช้งานเคสทูล

งานวิจัยนี้ยังมีข้อจำกัดคืองานวิจัยดังกล่าวสอบถามจากนักพัฒนาระบบจาก 7 องค์กรซึ่งไม่ใช่องค์กรที่รับจ้างพัฒนาซอฟต์แวร์ (Software House) โดยองค์กรที่สำรวจนั้นเป็นองค์กรทางด้านการเงิน (Financial) อุตสาหกรรมการผลิต (Manufacturing) และธุรกิจขายปลีก (Retail) ที่มีการพัฒนาซอฟต์แวร์ใช้เอง และในงานวิจัยนี้ใช้รายการคุณสมบัติของเอฟซีทีเอ็มในการศึกษาเพียง 2 กลุ่มเท่านั้น ซึ่งทำให้เห็นการใช้งานคุณสมบัติเพียงบางส่วน

2.2.2 งานวิจัยของ McMurtrey และคณะ (2000) ซึ่งเป็นงานวิจัยที่เน้นการศึกษาถึงการใช้งานคุณสมบัติของเคสทูล โดยในงานวิจัยนี้เป็นอีกงานวิจัยที่ศึกษาถึงคุณสมบัติของเคสทูลที่ใช้รายการคุณสมบัติในการศึกษาตามเอฟซีทีเอ็ม (FCTM - Functional CASE Technology Model) (1990) โดยในงานนี้ได้ศึกษาการใช้งานโดยการใช้แบบสอบถาม (Questionnaire) ซึ่งสำรวจจากนักพัฒนาระบบ 226 คน จากหลากหลายองค์กร โดยนักพัฒนาระบบส่วนใหญ่มาจากองค์กรทางด้านประกันภัย (Insurance) และอุตสาหกรรมการผลิต (Manufacturing) ซึ่งงานวิจัยนี้สนใจถึงการสำรวจการใช้งานคุณสมบัติของเคสทูล โดยสรุปผลการสำรวจเป็นสถิติเชิงพรรณนา และได้นำเสนอคุณสมบัติ 10 รายการที่สำคัญและคุณสมบัติที่ยังต้องการปรับปรุงจากรายการของเอฟซีทีเอ็ม โดยงานวิจัยนี้พยามทำความเข้าใจถึงช่องว่าง (Gap) ที่เกิดขึ้นระหว่างคุณสมบัติของเคสทูลที่มีอยู่ (Features Possessed) กับคุณสมบัติของเคสทูลที่ต้องการ (Features Needed) ที่จะส่งผลให้การพัฒนาระบบมีคุณภาพ โดยผู้วิจัยพบว่ายังคงมีปัจจัยที่เปลี่ยนแปลงมากมายที่ทำให้เกิดช่องว่าง

งานวิจัยนี้ยังมีข้อจำกัดคือการศึกษาถึงการใช้งานคุณสมบัติต่างๆของเคสทูลนั้นใช้รายการของเอฟซีทีเอ็มซึ่งรายการคุณสมบัติส่วนใหญ่ยังอิงกับการพัฒนาแบบซอฟต์แวร์เชิงโครงสร้าง ทำให้ขาดการศึกษาคุณสมบัติในเชิงวัตถุ และยังขาดการนำเสนอถึงคุณสมบัติที่ยังไม่เพียงพอกับความต้องการว่าควรเป็นอย่างไร ที่ผู้วิจัยงานดังกล่าวเรียกว่า "ช่องว่าง (Gap)" ที่เกิดขึ้น ซึ่งเป็นส่วนที่น่าสนใจที่น่าจะนำมาศึกษาถึงช่องว่างดังกล่าวว่าเป็นอย่างไร เพื่อนำเสนอคุณสมบัติที่เหมาะสมซึ่งจะช่วยลดช่องว่างดังกล่าวต่อไป

2.2.3 งานวิจัยของ Maccari และคณะ (2002) ซึ่งศึกษาถึงการใช้งานคุณสมบัติต่างๆของเคสทูลที่มีการใช้งาน (Used CASE Tool Features) รวมถึงคุณสมบัติในอุดมคติ (Ideal CASE Tool Features) ที่ควรจะมีจากกลุ่มผู้พัฒนาซอฟต์แวร์ในบริษัทโนเกีย (Nokia) โดยกำหนดรายการในการศึกษาขึ้นทั้งหมด 33 รายการ (รายละเอียดในภาคผนวก ค) และใช้แบบสอบถาม(Questionnaire) โดยส่งทางจดหมายอิเล็กทรอนิคส์ (E-mail) ไปยังผู้จัดการ (Managers) และนักพัฒนาซอฟต์แวร์ (Software Developers) ที่มีประสบการณ์ในการใช้งานเคสทูลในบริษัทโนเกีย โดยงานวิจัยดังกล่าวสรุปผลการสำรวจเป็นสถิติเชิงพรรณนาถึงคุณสมบัติที่มีการใช้งาน (Used CASE Tool Features) และคุณสมบัติในอุดมคติ (Ideal CASE Tool Features) โดยเป็นการศึกษาถึงการใช้งานคุณสมบัติต่างๆของเคสทูล

ในงานวิจัยนี้รายการคุณสมบัติที่วิจัยรองรับการพัฒนาซอฟต์แวร์เชิงวัตถุมากว่าเอฟซีทีเอ็ม เช่น การระบุในรายการว่าเคสทูลควรจะสนับสนุนมาตรฐานสัญลักษณ์ของภาษายูเอ็มแอล เป็นต้น แต่

จากการศึกษางานวิจัยนี้พบว่ายังมีข้อจำกัด คือ งานวิจัยนี้เน้นศึกษาในอุตสาหกรรมด้านเดียวคือด้านมือถือ (Mobile) และใช้บริษัทโนเกีย (Nokia) เพียงบริษัทเดียวในการส่งแบบสอบถาม ซึ่งผู้วิจัยคือ Maccari และคณะเองก็ได้นำเสนอว่าควรขยายไปยังบริษัทอื่นๆและอุตสาหกรรมอื่นๆ ที่มีการใช้งานเคสทูลในการพัฒนาซอฟต์แวร์ เพื่อเพิ่มความน่าเชื่อถือของผลการวิจัย และควรจะมีการส่งผลการสำรวจเชื่อมโยงกับบริษัทที่เป็นผู้พัฒนาเคสทูลเพื่อปรับปรุงให้ตรงกับความต้องการใช้ต่อไป

2.2.4 งานวิจัยของ Hubert และ Laura (2003) ซึ่งได้ทดลองใช้เคสทูลที่สนับสนุนการพัฒนาซอฟต์แวร์เชิงวัตถุในการพัฒนาซอฟต์แวร์ โดยทดลองใช้ในวิชาวิศวกรรมซอฟต์แวร์ (Software Engineering) ภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมองคแลร์สเตท (Computer Science Department at Montclair State University) เพื่อต้องการให้นักศึกษาสามารถพัฒนาระบบที่มีคุณภาพสูงจากการใช้แนวความคิดเชิงวัตถุและเครื่องมือ (Tool) โดยเคสทูลที่นำมาใช้ในการทดลองครั้งนี้คือเรชั่นนัลโรส และระบบที่ใช้ในการทดลองพัฒนาคือระบบเครื่องรับจ่ายเงินอัตโนมัติสำหรับธนาคารมหาวิทยาลัยมองคแลร์สเตท (Automated Teller Machine (ATM) for the Montclair State University Bank) และในการทดลองใช้นักศึกษาเป็นตัวแทนนักพัฒนาระบบในการพัฒนาระบบ

ซึ่งจากงานวิจัยนี้พบว่าการใช้เคสทูลช่วยให้นักศึกษาสามารถคิดและมองระบบเชิงวัตถุได้อย่างชัดเจนและเปลี่ยนทัศนคติของนักศึกษาที่ว่าการพัฒนาซอฟต์แวร์เป็นเพียงการลงรหัส (Code) โปรแกรมเท่านั้น โดยการศึกษาพบว่าการใช้เคสทูลในการวิเคราะห์และออกแบบซอฟต์แวร์นั้นเป็นประโยชน์อย่างมาก กล่าวคือ

1. เคสทูลช่วยในการสร้างโมเดล (Modeling) ตลอดช่วงการออกแบบระบบ (Design Phase) นอกจากนั้นการใช้เคสทูลยังช่วยให้มีความเข้าใจในระบบงานที่จะพัฒนาได้ดียิ่งขึ้น
2. ช่วยสนับสนุนการพัฒนาแบบวนซ้ำ (Iterative) โดยการเพิ่มเติมหรือเปลี่ยนแปลงระบบนั้นทำได้ง่ายขึ้น
3. การใช้งานเคสทูลช่วยในการพัฒนาระบบที่เป็นทีม จากการสังเกต (Observation) และการสัมภาษณ์ (Interviewing) สมาชิกในทีมที่พัฒนาระบบ ผลปรากฏอย่างชัดเจนว่าเครื่องมือ (Tool) นั้นช่วยส่งเสริมนักพัฒนาให้เห็นภาพของระบบ

โดยข้อจำกัดของการทดลองครั้งนี้คือไม่มีการจัดการเรื่องของการวางแผนก่อนโครงการ (Pre-Planned) แผนการจัดการความเสี่ยงของโครงการ (Project Risk Planning) และใช้ซอฟต์แวร์เรชั่นนัลโรสเพียงตัวเดียวในการพัฒนาระบบ

งานวิจัยนี้แสดงให้เห็นถึงประโยชน์ในการนำเคสทูลเข้ามาใช้ในการพัฒนาซอฟต์แวร์และทำให้สามารถเข้าใจซอฟต์แวร์ที่จะพัฒนา แต่ข้อจำกัดของงานวิจัยนี้ก็คือการใช้เคสทูลเพียงตัวเดียวในศึกษาและทดลองใช้ ทำให้ไม่เห็นข้อแตกต่างของเคสทูลตัวอื่น ว่ามีคุณสมบัติแตกต่างกันอย่างไร รวมถึงข้อดีและข้อเสียของเคสทูลแต่ละตัว ความเหมาะสมในการใช้ และคุณสมบัติที่มีอยู่นั้นเพียงพอหรือไม่ในการใช้พัฒนาซอฟต์แวร์เชิงวัตถุ และไม่มีการแสดงถึงความต้องการคุณสมบัติของเคสทูลหลังจากที่ผู้พัฒนาใช้เคสทูลในการพัฒนาซอฟต์แวร์ ว่ามีส่วนใดบ้างที่เคสทูลไม่รองรับขณะที่ผู้ใช้ทดลองใช้งาน

2.2.5 งานวิจัยของ Peter Ordén และ Tom Boive (2001) ได้เปรียบเทียบยูเอ็มแอลเคสทูลโดยได้สำรวจเคสทูลที่สนับสนุนการพัฒนาซอฟต์แวร์เชิงวัตถุ ด้วยการค้นหาข้อมูลบนอินเตอร์เน็ต และได้พบเว็บไซต์ http://www.objectfaq.com/oofaq2/body/case.htm ซึ่งเป็นเว็บไซต์ที่เป็นฐานข้อมูลเก็บรวบรวมลิงค์ของเคสทูลที่สนับสนุนการพัฒนาซอฟต์แวร์เชิงวัตถุและได้เลือกเคสทูลที่จะนำมาเปรียบเทียบจากลิงค์ของเว็บไซต์ดังกล่าว ซึ่งการเลือกเคสทูลที่จะนำมาเปรียบเทียบนั้นได้พิจารณาจากความสามารถที่ใกล้เคียงกันและมีให้ดาวโหลดฟรี (Free Download) เป็นเวอร์ชั่นทดสอบ (Demo Version) โดยงานวิจัยนี้ได้เลือกเคสทูล 3 ตัว มาทำวิจัยเปรียบเทียบ คือ เรชั่นนัลโรส 2000, เมจิกดรอยูเอ็มแอลรุ่นมาตรฐาน (MagicDraw UML Standard Edition) และวิชวลยูเอ็มแอลรุ่นมาตรฐาน (Visual UML Standard Edition) และได้กำหนดเกณฑ์ (Criteria) ในการเปรียบเทียบโดยใช้แผนภาพจากหนังสือยูนิไฟด์โมเดลลิ่งแลงกวิจยูสเซอไกด์ (Unified Modeling Language User Guide) (Grady Booch James Rumbaugh and Ivar Jacobson,1999) และเดอะยูนิไฟด์โมเดลลิ่งแลงกวิจเรฟเฟอร์เร็นซแมนนวล (The Unified Modeling Language Reference Manual) (Grady Booch James Rumbaugh and Ivar Jacobson, 1999) เป็นหลักในการกำหนดเกณฑ์ และทดลองใช้เคสทูลในแต่ละเครื่องมือทำตามเกณฑ์ที่กำหนดขึ้นว่าสามารถทำได้หรือไม่ในแต่ละเกณฑ์ โดยสรุปผลของงานวิจัยเป็นข้อดี – ข้อด้อยของเคสทูลแต่ละตัวและสรุปภาพรวมของเคสทูลที่นำมาเปรียบเทียบด้วย

งานวิจัยนี้แสดงให้เห็นถึงการเปรียบเทียบเคสทูลสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุและการกำหนดเกณฑ์ (Criteria) ในการเปรียบเทียบเคสทูล แต่อย่างไรก็ตามงานวิจัยนี้ยังมีข้อจำกัด คือ การกำหนดเกณฑ์กำหนดโดยไม่ได้มีการจัดกลุ่มคุณสมบัติของเคสทูล แต่เป็นการกำหนดตามแผนภาพทั้ง 9 แผนภาพของภาษายูเอ็มแอล และมีส่วนที่เป็นเกณฑ์อื่นๆ (Other Criteria) ซึ่งทำให้เกณฑ์ที่กำหนดไม่ครอบคลุมตลอดการพัฒนาซอฟต์แวร์ เช่น ยังไม่ครอบคลุมการทดสอบซอฟต์แวร์

(Software Testing) เป็นต้น และเนื่องจากผู้วิจัยงานนี้ใช้เคสทูลที่เป็นเวอร์ชั่นทดสอบ (Demo Version) ทำให้เกิดข้อจำกัดในการเปรียบเทียบ เนื่องจากผู้วิจัยไม่สามารถเปรียบเทียบบางเกณฑ์ได้ทันก่อนเคสทูลหมดอายุ ซึ่งทำให้ผลการเปรียบเทียบออกมาไม่ครบถ้วนและไม่ชัดเจน และข้อจำกัดอีกข้อของงานวิจัยนี้คือการเปรียบเทียบนั้นเป็นเพียงการเปรียบเทียบแยกเป็นแผนภาพและเปรียบเทียบตามเกณฑ์อื่นๆ โดยใช้แผนภาพที่ไม่มีความสัมพันธ์เป็นระบบงาน และไม่ได้มีการจำลองระบบต้นแบบเพื่อใช้งานเคสทูลในการพัฒนาซอฟต์แวร์อย่างเป็นระบบ ซึ่งจะช่วยทำให้เห็นถึงคุณสมบัติในแง่ของการบูรณาการฟังก์ชันการทำงานทั้งหมดเข้ากับวงจรการพัฒนาซอฟต์แวร์ อย่างที่นำเสนอในงานวิจัยของ Hubert และ Laura (2003) และไม่ได้มีการนำเสนอคุณสมบัติของเคสทูล

2.2.6 บทความของ Jie Zhao และ Jeremy Meyer (2005) ได้เปรียบเทียบยูเอ็มแอลเคสทูล (UML CASE Tool) 2 ตัว คือ เอ็นเตอร์ไพส์อาร์คีเทคเจอร์ 5.0 (Enterprise Architect 5.0) และเรชั่นนัลโรส 2003 (Rational Rose 2003) โดยกำหนดเกณฑ์ (Criteria) ในการเปรียบเทียบเคสทูลตามหัวข้อต่างๆที่กำหนด เช่น คุณสมบัติยูเอ็มแอล (UML Features) ราวทิปเอ็นจีเนียริ่ง (Roundtrip Engineering) และการสนับสนุนวงจรของโครงการ (Project Life Cycle Support) เป็นต้น โดยมีการสรุปในรูปแบบของตารางในแต่ละหัวข้อและสรุปผลการเปรียบเทียบเป็นคำอธิบาย นอกจากนั้นยังมีการเปรียบเทียบการใช้งานจริง (Practical Considerations) โดยมีการสรุปถึงการใช้งาน (Usability) และทางด้านค่าใช้จ่าย (Cost) ในการใช้งาน

บทความนี้แสดงให้เห็นถึงการเปรียบเทียบเคสทูลสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุและการกำหนดเกณฑ์ (Criteria) ในการเปรียบเทียบเคสทูล แต่อย่างไรก็ตามบทความนี้ยังมีข้อจำกัด คือ การกำหนดเกณฑ์ในการเปรียบเทียบเคสทูลโดยผู้เขียนเอง ซึ่งแม้ว่าหัวข้อต่างๆที่กำหนดขึ้นนั้นจะครบคลุมการพัฒนาซอฟต์แวร์เชิงวัตถุมากกว่างานของ Peter Ordén และ Tom Boive ที่เน้นการเปรียบเทียบในการวาดแผนภาพต่างๆ แต่ก็ไม่การจัดกลุ่มคุณสมบัติของเคสทูลอย่างชัดเจนเพื่อกำหนดเกณฑ์ในการเปรียบเทียบ ประกอบกับไม่ได้มีการจำลองระบบต้นแบบเพื่อเปรียบเทียบการใช้งานเคสทูลในการพัฒนาซอฟต์แวร์อย่างเป็นระบบ ซึ่งจะทำให้เห็นถึงคุณสมบัติในแง่ของการบูรณาการฟังก์ชันการทำงานทั้งหมด แต่ในบทความนี้มีการสรุปทางด้านการใช้งานจริงและค่าใช้จ่ายของเครื่องมือ

2.2.7 บทความของ Eriksson และ Penker (1998) ซึ่งนำเสนอคุณสมบัติของเคสทูลที่ควรมีในการสร้างโมเดล (Model) ของซอฟต์แวร์ตามหลักการเชิงวัตถุ ซึ่งได้นำเสนอคุณสมบัติที่ควรมีดังนี้

- วาดแผนภาพ (Draw Diagrams)

เครื่องมือที่สามารถวาดแผนภาพต่างๆในการสร้างโมเดล (Model) ได้ง่าย โดยเครื่องมือจะต้องมีความฉลาดเพียงพอกล่าวคือเข้าใจจุดประสงค์ในการวาดแผนภาพ เข้าใจถึงสัญลักษณ์และกฎพื้นฐานต่างๆ ของแต่ละแผนภาพ โดยจะต้องป้องกันไม่ให้ผู้ใช้ทำผิดพลาดในการสร้างแผนภาพ

- มีการใช้งานรีโพสิโทรี (Act as a Repository)

เครื่องมือต้องสนับสนุนรีโพสิโทรีในการเก็บข้อมูลเกี่ยวกับการสร้างโมเดล (Model) ต่างๆ เมื่อมีการเปลี่ยนแปลงใดๆในองค์ประกอบของแผนภาพเกิดขึ้น จะต้องส่งผลให้เกิดการเปลี่ยนแปลงในแผนภาพอื่นๆที่ใช้งานองค์ประกอบนั้น เช่น หากมีการเปลี่ยนชื่อคลาส (Class) โดยเปลี่ยนในแผนภาพคลาส จะต้องเปลี่ยนในแผนภาพอื่นๆที่นำคลาสนั้นไปใช้ด้วย

- สนับสนุนการนำทาง (Support Navigation)

เครื่องมือต้องมีระบบนำทางที่ง่ายและให้ผู้ใช้สามารถทำตามหลักการโมเดล (Model) ซอฟต์แวร์ได้ เช่นสามารถนำข้อมูลหรือองค์ประกอบจากแผนภาพหนึ่งไปยังอีกแผนภาพหนึ่งได้ หรือสามารถที่จะอธิบายรายละเอียดของแต่ละองค์ประกอบลงในแผนภาพ

- รองรับการทำงานที่มีผู้ใช้งานหลายคน (Provide Multi-user Support)

เครื่องมือต้องสนับสนุนการทำงานที่มีผู้ใช้งานหลายคนได้ โดยต้องมีการป้องกันการทำงานซ้ำซ้อนในการสร้างโมเดลเดียวกัน คือสามารถล็อกแผนภาพในขณะทำงาน รวมถึงการควบคุมผู้ใช้อื่นในการเปลี่ยนแปลงองค์ประกอบอื่นๆที่มีผลกระทบต่องานที่ทำอยู่ด้วย

- การสร้างรหัส (Generate Code)

เครื่องมือสร้างรหัส โดยสามารถนำเอาข้อมูลจากการสร้างโมเดล (Model) ทั้งหมดเปลี่ยนเป็นโครงร่างของรหัส ซึ่งจะมีประโยชน์และใช้เป็นพื้นฐานในขั้นตอนการพัฒนาระบบ (Implementation Phase)

- การทำวิศวกรรมย้อนกลับ (Reverse Engineering)

การทำวิศวกรรมย้อนกลับเป็นการทำงานที่ตรงข้ามกับการสร้างรหัส โดยเครื่องมือจะอ่านรหัส (Code) และสร้างโมเดล (Model) จากรหัสดังกล่าว โดยการสร้างรหัส (Generate Code) และการทำวิศวกรรมย้อนกลับ (Reverse Engineering) นั้นเรียกรวมกันว่า Round-trip Engineering ดังนั้นนักพัฒนาสามารถพัฒนาเพิ่มเติมจากระบบงานเดิมที่มีโปรแกรมอยู่แล้ว โดยสามารถแปลงกลับมาเป็นแผนภาพเพื่อออกแบบและพัฒนาเพิ่มเติม โดยหลังจากออกแบบเสร็จสามารถใช้การสร้างรหัสในการสร้างรหัสของระบบใหม่ได้

- การเชื่อมโยงกับเครื่องมืออื่นๆ (Integrate with Other Tools)

เครื่องมือต้องสามารถเชื่อมโยงกับเครื่องมืออื่นๆได้ ไม่ว่าจะเป็นเครื่องมือทางด้านการพัฒนา (Development Environment) เช่น เครื่องมือการเขียนโปรแกรม (Editor) ตัวแปลโปรแกรม (Compiler) โปรแกรมตรวจและแก้ที่ผิด (Debugger) และเครื่องมือในการทดสอบโปรแกรม (Test Tool) ต่างๆ รวมถึงการเชื่อมโยงกับเครื่องมือด้านอื่นๆ เช่น เครื่องมือการจัดการโครงแบบซอฟต์แวร์และควบคุมรุ่น (Configuration Management and Version Control) เป็นต้น

- การแลกเปลี่ยนโมเดล (Interchange Models)

เครื่องมือต้องสามารถวาดแผนภาพจากเครื่องมือหนึ่งและนำข้อมูลออกมาเพื่อนำเข้าในเครื่องมืออื่นที่สัมพันธ์กันได้ โดยข้อมูลที่นำออกมาต้องมีรูปแบบที่เป็นมาตรฐาน โดยแผนภาพยูเอ็มแอลส่วนใหญ่ใช้มาตรฐานของโอเอ็มจี (OMG) ซึ่งผู้พัฒนาเครื่องมือควรเอื้อให้มีการเปลี่ยนแปลงระหว่างเครื่องมือได้

บทความนี้แสดงให้เห็นถึงคุณสมบัติของเคสทูลที่ควรมีในการสร้างโมเดล (Model) ของซอฟต์แวร์ตามหลักการเชิงวัตถุ แต่บทความนี้ยังมีข้อจำกัด คือ คุณสมบัติที่นำเสนอนั้นเป็นการนำเสนอในภาพรวมและไม่ได้นำมาเปรียบเทียบกับเคสทูลที่มีการใช้งานอยู่ในปัจจุบันว่ามีการรองรับคุณสมบัติที่นำเสนออย่างไร

2.2.8 บทความของ Frank Haubenschild (2001)ได้นำเสนอการเปรียบเทียบเคสทูลที่มีใช้งานในปัจจุบัน ที่รองรับการพัฒนาซอฟต์แวร์เชิงวัตถุและสนับสนุนการใช้ภาษายูเอ็มแอลในการ

วิเคราะห์และออกแบบที่มีการใช้งานบนระบบปฏิบัติการลีนุกซ์ โดยเลือกเคสทูลจำนวน 5 ตัว คือ ไดอะ 0.86 (Dia 0.86) อาร์โกยูเอ็มแอล (ArgoUML) ทูเกทเทอร์โซโล 4.2 (Together Solo 4.2) ทูเกทเทอร์คอนโทนเซ็นเตอร์ 4.2 (Together Control Center 4.2) และเรชั่นนัลโรส 2001 (Rational Rose 2001) โดยนำเสนอผลในรูปของการเปรียบเทียบฟังก์ชันการใช้งานแยกเป็นแต่ละซอฟต์แวร์ว่ามีความสามารถในการทำฟังก์ชันใดได้บ้าง ดังตารางที่ 2-1

ตารางที่ 2-1: ผลการเปรียบเทียบเคสทูลจากบทความของ Frank Haubenschild

CASE tools in overview					
Product	**Dia 0.86**	**ArgoUML 0.8.1**	**Together Solo 4.2**	**Together Control Center 4.2**	**Rational Rose 2001**
Manufacturer	Alexander Larsson	University of California	Together Soft	Together Soft	Rational
Internet	www.lysator.liu.se/~alla/dia	www.ArgoUML.org	www.togethersoft.com	www.togethersoft.com	www.rational.com
Price (approx.)	GPL	free	£1600	£2800	£5500
Diagram types					
Use case	+	+	+	+	+
Class	+	+	+	+	+
State	-	+	+	+	+
Deployment	-	+	+	+	+
Activity	-	+	+	+	+
Collaboration	-	+	+	+	+
Sequence	+	-	+	+	+
Entity-Relationship	-	-	-	+	+
Code generation	only via Dia2code (C++, Java)	Java	Java, C++, IDL	Java, C++, IDL	C++, IDL, Java, Ada 83, Ada 95
Team support	-	-	+	+	+
Reverse engineering	-	-	+	+	+
Roundtrip engineering	-	-	+	+	+
Other	expandable by XML	open source, XMI support, needs JRE 1.2	Version 5.0 just out	Java debugger, Version 5.0 just out	Look & Feel matches Windows version

นอกจากนั้นในบทความนี้ยังสรุปถึงข้อดีและข้อเสียของเคสทูลที่เลือกมานำเสนอ โดยเคสทูลอย่างเช่น อาร์โกยูเอ็มแอล และทูเกทเทอร์โซโล มีคุณสมบัติทางด้านการใช้งานส่วนติดต่อผู้ใช้งาน (User Interface) ที่ใช้งานยากและมีข้อจำกัด ส่วนทูเกทเทอร์โซโลและเรชั่นนัลโรสถูกจำกัดด้วยราคาของค่าลิขสิทธิ์ (License) ที่สูง โดยในบทความนี้ยังได้นำเสนอคุณสมบัติของเคสทูลโดยแบ่งเป็น 3 ระดับคือ คุณสมบัติหลัก (Main CASE Tool Features) ความต้องการของผู้ทำงานระดับกึ่งมืออาชีพ (Semi Professional Requirement) และความต้องการของผู้ทำงานระดับมืออาชีพ (Professional Requirement) ไว้ดังนี้

- คุณสมบัติหลัก (Main CASE Tool Features)
  - ความสามารถในการสนับสนุนการสร้างแผนภาพยูเอ็มแอล
    (Support for common UML diagram types)
  - ใช้งานง่ายและมีการแนะนำ
    (Simple, intuitive user guidance)
  - ความง่ายในการวาดแผนภาพ
    (Easy-to-use diagram-lay outer)

- ความต้องการของผู้ทำงานระดับกึ่งมืออาชีพ (Semi Professional Requirement)
  - ความสามารถในการทำวิศวกรรมย้อนกลับ
    (Reverse Engineering)
  - ความสามารถในการสนับสนุนการพัฒนาซอฟต์แวร์ร่วมกันเป็นทีม
    (Support Software Development Team)
  - ความยืดหยุ่นของเครื่องมือสร้างเอกสารในรูปแบบต่างๆ เช่น เว็บเพ็จ
    (Flexible documentation creation)
  - เครื่องมือสร้างรหัสสามารถรองรับได้หลายภาษา
    (Code Generation (Support for Several Target Languages))

- ความต้องการของผู้ทำงานระดับมืออาชีพ (Professional Requirement)
  - การรองรับการทำงานกับฐานข้อมูล
    (Database Support)
  - การรองรับการทำงานเป็นทีมในโครงการขนาดใหญ่
    (Team Support for Larger Projects)
  - เป็นสถาปัตยกรรมเปิดที่สามารถรองรับการขยายงาน
    (Open architecture for any potential expansions)
  - สามารถรองรับความแตกต่างในสภาพแวดล้อมต่างๆได้ เช่นรองรับหลายแพร์ทฟอร์ม
    (In Heterogeneous Environments, Support for as Many Platforms as Possible)

บทความนี้ยังมีข้อจำกัด คือ ศึกษาเพียงเคสทูลที่ทำงานบนระบบปฏิบัติการลีนุกซ์เท่านั้น และศึกษาเคสทูลในแง่ของฟังก์ชันการทำงานเป็นฟังก์ชันๆ ไม่ได้มีการจำลองระบบต้นแบบเพื่อใช้งานเคสทูลในการพัฒนาซอฟต์แวร์อย่างเป็นระบบ ซึ่งจะช่วยทำให้เห็นถึงคุณสมบัติในแง่ของการบูรณาการของฟังก์ชันการทำงานทั้งหมดเข้ากับวงจรการพัฒนาซอฟต์แวร์ อย่างที่นำเสนอในงานวิจัยของ Hubert และ Laura (2003) และไม่ได้มีการนำเสนอคุณสมบัติของเคสทูล

2.2.9 บทความของ Hoffer และคณะ (2002) ได้กล่าวถึงรูปแบบของเคสทูลในอนาคตว่า เคสทูลในอนาคตนั้นจะมีการพัฒนาอย่างรวดเร็ว โดยจะเน้นการจัดทำโปรแกรมให้มีความรวดเร็วและง่ายขึ้น และกล่าวเพิ่มเติมอีกว่าเคสทูลมีแนวโน้มที่จะเป็นเครื่องมือที่สนับสนุนในการพัฒนาซอฟต์แวร์เชิงวัตถุมากขึ้น โดยมองว่าการพัฒนาซอฟต์แวร์เชิงวัตถุเป็นแนวทางที่กำลังเป็นที่นิยม ซึ่งเคสทูลจะช่วยให้ปัญหาสำหรับการบำรุงรักษาระบบลดลงและช่วยในการรองรับคุณสมบัติของการพัฒนาซอฟต์แวร์เชิงวัตถุ เช่น เรื่องแนวคิดการนำกลับมาใช้ใหม่ (Reuse) และช่วยในเรื่องของการสร้างคลาส (Class) รวมถึงการรองรับการรวมข้อมูลและคำสั่งไว้ในวัตถุหนึ่งๆ ซึ่งคือการห่อหุ้ม (Encapsulation) และมีการเน้นการนำรหัสกลับมาใช้ใหม่ (Reusable Code) ซึ่งในส่วนนี้จะช่วยลดเวลาและค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ใหม่ อย่างไรก็ตาม การนำเสนอแนวทางในการพัฒนาเคสทูลในอนาคตของ Hoffer และคณะนั้น ยังขาดการนำเสนอในด้านของคุณสมบัติ (Features) ของเคสทูลว่าควรเป็นอย่างไร และขาดการนำเสนอในรายละเอียดของคุณสมบัติที่นำเสนอ

2.3 ข้อสรุป

จากงานวิจัยและการศึกษาในอดีตพบว่ายังไม่มีการศึกษาใดที่สำรวจและนำเสนอคุณสมบัติของเคสทูลที่สนับสนุนการพัฒนาซอฟต์แวร์เชิงวัตถุในประเทศไทย ซึ่งปัจจุบันประเทศไทยมีความต้องการส่งเสริมศักยภาพในการพัฒนาซอฟต์แวร์ในระดับมหภาค และการใช้งานเคสทูลจะส่งผลให้การพัฒนาเป็นไปได้อย่างรวดเร็วและถูกต้องมากยิ่งขึ้น ถึงแม้จะมีงานวิจัยต่างๆที่ศึกษาและเปรียบเทียบการใช้เคสทูลที่สนับสนุนการพัฒนาซอฟต์แวร์เชิงวัตถุตามที่ได้นำเสนอในข้างต้น แต่ยังไม่พบว่ามีงานวิจัยใดที่ได้กำหนดหรือระบุรายละเอียดของเกณฑ์หรือเงื่อนไขที่จะนำมาใช้วัดคุณสมบัติของเคสทูลไว้อย่างชัดเจน ผู้วิจัยจึงเล็งเห็นประโยชน์ของการศึกษาและทำวิจัยเพื่อเป็นแนวทางในการพัฒนาเคสทูลสำหรับการพัฒนาซอฟต์แวร์เชิงวัตถุ โดยศึกษาและเปรียบเทียบคุณสมบัติของเคสทูลที่มีการใช้อยู่ในปัจจุบัน และนำเสนอคุณสมบัติของเคสทูลสำหรับการพัฒนา

ซอฟต์แวร์เชิงวัตถุในประเทศไทย ซึ่งเชื่อว่าผลที่ได้จากการศึกษาน่าจะเป็นประโยชน์ต่อการพัฒนาซอฟต์แวร์ และต่อคุณภาพของซอฟต์แวร์ที่ผลิตในประเทศไทย