บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ภาษาโปรแกรมมิ่งแบบรูปภาพ (Graphical Programming)

Mark และคณะ ได้ทำการศึกษาวิจัยการเปรียบเทียบการใช้งานระหว่างแลปวิว ซึ่งเป็นภาษาโปรแกรมแบบรูปภาพ กับโปรแกรมแมทแลป (Matlab) ซึ่งเป็นภาษาโปรแกรมแบบเชิงโครงสร้างในการสอนวิชาการประมวลผลสัญญาณดิจิตอล (Digital Signal Processing, DSP) การศึกษานี้ได้ใช้ภาษาโปรแกรมทั้งสองรูปแบบนี้สอนในชั้นเรียนที่มีนักศึกษาชั้นปีที่ 2 สาขาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ โดยให้นักศึกษาทำโครงงานเล็กๆ เป็นการบ้านโดยใช้โปรแกรมแมทแลปก่อน หลังจากนั้นทั้งสองท่านได้ให้นักศึกษากลุ่มเดิมทำโครงงานนี้ซ้ำอีกครั้งหนึ่ง แต่ให้ทำใหม่โดยใช้โปรแกรมแลปวิว และยังทดลองในทำนองเดียวกันนี้อีกกับนักศึกษาอีกหลายกลุ่มทุกภาคเรียนการศึกษาและนำผลคะแนนที่ได้จากการทำโครงงานมาวิเคราะห์ผล และได้ผลการวิจัยดังตาราง 2.1 (3)

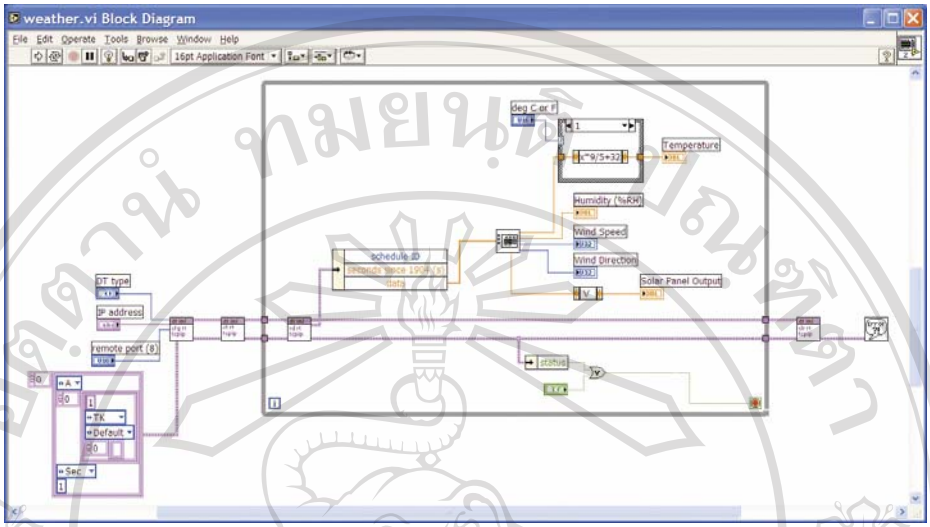
ตาราง 2.1 แสดงผลการวิจัยของ Mark และ Bruce

	Win 02-03	Spr 02-03	Win 03-04	Spr 03-04	Win 04-05	Spr 04-05	Win 05-06	Spr 05-06
Language	MATLAB	MATLAB	MATLAB	MATLAB	MATLAB	MATLAB	LABVIEW	LABVIEW
Number of Students	38	40	56	29	61	42	64	34
Normalized Gain	37	36	32	29	41	41	46	37

สรุปผลได้ว่า ผลคะแนนเฉลี่ยของนักศึกษาที่ใช้โปรแกรมแลปวิวทำโครงงานจะสูงกว่าประมาณ 5 คะแนนเสมอ แต่อย่างไรก็ตาม การวิจัยนี้ยังคงดำเนินต่อไป เพื่อยืนยันผลการทดลองดังกล่าวให้ชัดเจนยิ่งขึ้น และทั้งสองจะทำการสัมภาษณ์นักศึกษาเพื่อขอความคิดเห็นเกี่ยวกับการใช้งานทั้ง 2 ภาษาโปรแกรมอีกด้วย เพื่อที่จะเป็นข้อมูลเพิ่มเติมในการวิเคราะห์ผลการวิจัยต่อไป

ภาษาโปรแกรมแบบรูปภาพ เป็นภาษาโปรแกรมที่ใช้รูปภาพหรือสัญลักษณ์แทนการเขียนด้วยตัวอักษรเหมือนภาษาโปรแกรมปกติทั่วไป คือจะใช้บล็อกฟังก์ชันซึ่งแทนด้วยรูปไอคอน (Icon) แทนการเขียนโปรแกรมย่อย (Subroutine) และใช้เส้นเชื่อมต่อระหว่างบล็อกฟังก์ชันแทนการไหลของข้อมูลระหว่างโปรแกรมย่อยนั้นๆ คล้ายกับการเขียนโฟลว์ชาร์ต (Flow Chart) หรือบล็อกไดอาแกรม (Block Diagram) ของโปรแกรม โดยไม่ต้องจดจำรูปแบบคำสั่งที่ยุ่งยาก ปัจจุบันมีเครื่องมือที่ใช้เขียนภาษาโปรแกรมแบบรูปภาพหลายชนิด เช่น MATLab-Simulink, VEE,

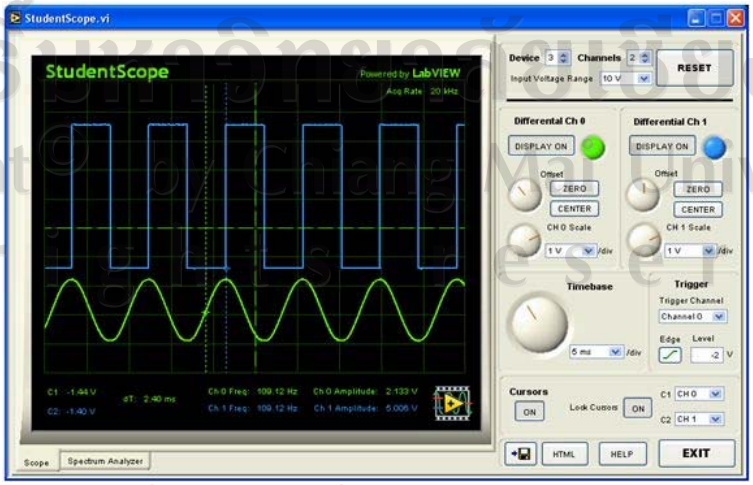
Microsoft Visual Programming Language, LabView เป็นต้น แต่ในการวิจัยนี้จะใช้เครื่องมือ LabVIEW ของ บริษัท National Instruments เป็นหลักในการพัฒนาซอฟต์แวร์ ดังแสดงในรูปที่ 2.1



รูป 2.1 แสดงการพัฒนาซอฟต์แวร์โดยใช้ภาษาโปรแกรมแบบรูปภาพ (4)

แลปวิว (LabVIEW) เป็นเครื่องมือพัฒนาซอฟต์แวร์เพื่อนำมาใช้ในด้านการวัดและเครื่องมือวัดสำหรับงานทางวิศวกรรม (5) คำว่า LabVIEW ย่อมาจาก Laboratory Virtual Instrument Engineering Workbench ซึ่งหมายความว่าเป็นโปรแกรมที่สร้าง เครื่องมือวัดเสมือนจริงในห้องปฏิบัติการทางวิศวกรรม ดังนั้นจุดประสงค์หลักของการทำงานของแลปวิว ก็คือการจัดการในด้านการวัดและเครื่องมือวัดอย่างมีประสิทธิภาพนั่นเอง

โปรแกรมที่พัฒนาขึ้นโดยใช้แลปวิวจะเรียกว่า เวอร์ชวลอินสทรูเมนท์ (Virtual Instrument) หรือเรียกย่อ ๆ ว่า วีไอ (VI) ซึ่งหมายถึงเครื่องมือวัดเสมือน ดังตัวอย่างจากรูป 2.2 ซึ่งเป็นกราฟแสดงผลรูปคลื่นสัญญาณจากเครื่องมือตรวจจับสัญญาณไฟฟ้าที่เรียกว่า ออสซิลโลสโคป (Oscilloscope) ดังรูป 2.2 ดังนี้



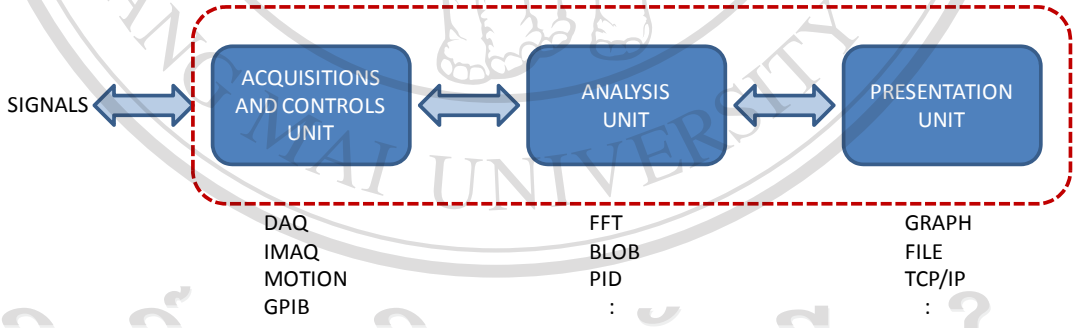
รูปที่ 2.2 แสดงรูปคลื่นสัญญาณจาก ออสซิลโลสโคป (6)

แลปวิวจะอาศัยหลักการทำงานของเครื่องมือวัดหรือการวัดคุมทำให้ผู้ใช้สามารถออกแบบรูปแบบโปรแกรมตามที่ผู้ใช้ต้องการ หลักการดังกล่าวแบ่งออกเป็น 3 ส่วนใหญ่ ๆ ดังรูปที่ 2.3 ดังนี้

(1) ส่วนรับข้อมูลและควบคุมอินพุต (Acquisitions And Controls Unit) ส่วนนี้จะเป็นส่วนที่นำข้อมูลจากสิ่งแวดล้อมภายนอกเข้าสู่ระบบคอมพิวเตอร์ โดยที่ข้อมูลที่เข้าสู่ระบบนี้อาจมาจากการ์ดดีเอคิว (DAQ Card) สำหรับรับข้อมูลที่เป็นสัญญาณทางไฟฟ้า, การ์ดไอเอ็มเอคิว (IMAQ Card) สำหรับข้อมูลประเภทรูป หรือจากเครื่องมือวัดทั่วไปทางจีพีไอบีบัส (GPIB Bus) ซึ่งจะได้อธิบายในหัวข้อที่ 4.5) ต่อไป

(2) ส่วนวิเคราะห์ข้อมูล (Analysis Unit) เมื่อได้รับข้อมูลมาแล้ว จะต้องผ่านส่วนนี้เพื่อที่จะทำให้ข้อมูลไปแสดงผลในรูปที่สื่อความหมายในสิ่งที่ผู้ใช้งานสามารถนำไปแสดงแทนสื่อที่วัดและใช้งานได้

(3) ส่วนแสดงผล (Presentation) เป็นการแสดงผลในรูปแบบที่เข้าใจง่ายและเป็นประโยชน์ต่อผู้ใช้งาน เช่น ดิจิตอลมัลติมิเตอร์ แสดงผลเฉพาะสัญญาณที่วัดได้โดยไม่จำเป็นต้องรู้ความสัมพันธ์กับเวลาที่ได้มากจากส่วนวิเคราะห์ข้อมูล เป็นต้น



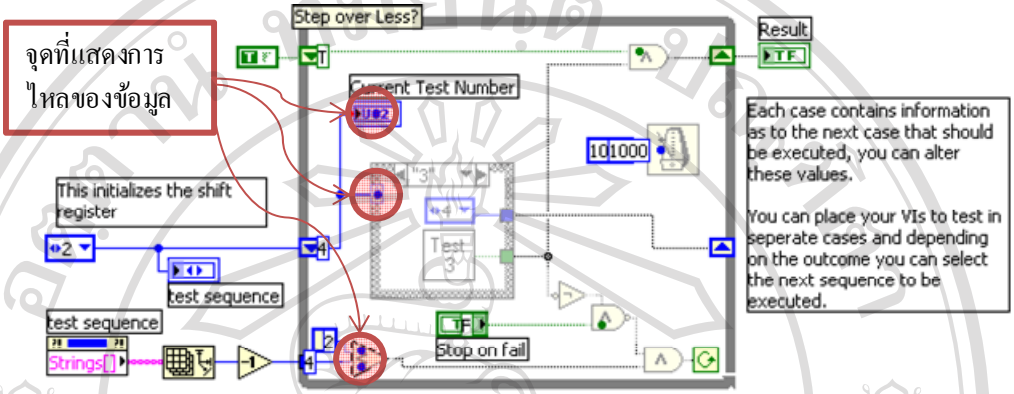
รูป 2.3 แสดงหลักการทำงานของเครื่องมือวัดหรือการวัดคุม

2.2 หลักการการไหลของข้อมูล (Dataflow Programming Fundamental)

หลักการการไหลของข้อมูล เป็นรูปแบบการทำงานที่ขึ้นอยู่กับการไหลของข้อมูลระหว่างบล็อกฟังก์ชั่นต่างๆ ในลักษณะที่ทำงานพร้อมกัน ทิศทางของการไหลนั้นจะถูกกำหนดด้วยเส้นเชื่อมต่าง ๆ ว่าจะไปที่ส่วนใด ผ่านการประเมินผลและคำนวณในส่วนใดบ้าง และจะให้แสดงผลอย่างไร ซึ่งลักษณะของการไหลของข้อมูลของภาษาโปรแกรมของรูปภาพนี้จะมีลักษณะ

เหมือนกับการเขียนบล็อกไดอาแกรม ซึ่งทำให้ผู้พัฒนาซอฟต์แวร์ให้ความสนใจกับการเคลื่อนที่และเปลี่ยนแปลงข้อมูลได้อย่างชัดเจนเสมือนมองเห็นจากข้างในของเครื่องมือวัดจริง ๆ (7)

รูปแบบการทำงานที่ขึ้นอยู่กับการไหลของข้อมูล นั้นสามารถแสดงดังรูป 2.4 ดังนี้



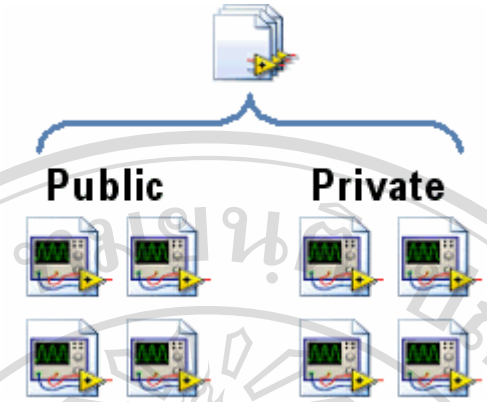
รูป 2.4 แสดงรูปแบบการทำงานของซอฟต์แวร์ที่ใช้หลักการการไหลของข้อมูล

2.3 ไลบรารี่ (Library)

ไลบรารี่โดยทั่วไปแล้วไลบรารี่เป็นการบริหารการจัดเก็บฟังก์ชันย่อย ๆ แล้วจัดเก็บรวมไว้เป็นโมดูลใหญ่ ๆ เพื่อความสะดวกในการเลือกใช้และง่ายในการใช้งาน เคลื่อนย้าย และสามารถใช้งานร่วมกันได้ ในส่วนของเครื่องมือวัดอุตสาหกรรม การสร้างไลบรารี่จะช่วยในการจัดการกับการติดต่อสื่อสารหรือส่งผ่านข้อมูล รวมถึงการจัดการกับหน่วยความจำของเครื่องมือวัดเพื่อจะสามารถรวบรวมข้อมูลมาใช้ในการคำนวณและเก็บข้อมูลให้ได้ประโยชน์สูงสุดนั้น โดยอาศัยความสามารถของการไหลของข้อมูลที่เรียกว่า การสร้างสัญญาณ (Signal Generator), ขบวนการจัดการสัญญาณ (Signal Processing) มาช่วยในการสร้างไลบรารี่ให้ง่ายขึ้น

โดยทั่วไปไลบรารี่ของแลปวิวจะเป็นแบบเชิงโครงการ (Project-Style Library) (8) ซึ่งภายในไลบรารี่จะมีวีไอรวมอยู่ด้วยกันเป็น 2 ส่วนคือ แบบสาธารณะ (Public) และแบบส่วนตัว (Private) โดยวีไอแบบสาธารณะ (Public VI) สามารถเรียกใช้งานได้จากวีไอตัวอื่นโดยตรงได้ แต่วีไปแบบส่วนตัวไม่สามารถเรียกได้จากวีไปตัวอื่น สามารถเรียกใช้งานได้แค่เพียงวีไอที่อยู่ในไลบรารี่เดียวกันเท่านั้น โดยทั่วไปจะถูกสร้างขึ้นมาเพื่อสนับสนุนการทำงานของวีไอแบบสาธารณะมากกว่า

ไลบรารี่ของแลปวิวจะเป็นแบบเชิงโครงการ (Project-Style Library) แสดงดังรูป 2.5 ดังนี้



รูป 2.5 แสดงโครงสร้างไลบรารี่ของแลปวิวแบบเชิงโครงการ (Project-Style Library)

2.4 โครงสร้างไดรฟ์เวอร์ของเครื่องมือวัด (Instrument Driver Organization)

โครงสร้างของไดรฟ์เวอร์ของเครื่องมือวัด (9) โดยปรกติแล้ว การสร้างไลบรารี่เพื่อทำงานเป็นไดรฟ์เวอร์ของเครื่องมือวัดนั้น จะเป็นแบบโครงการ (Project-Style Instrument Driver) ซึ่งตามหลักการออกแบบแบบนี้ จะต้องสร้างส่วนติดต่อกับผู้ใช้งาน (User Interface) ที่เป็นเมนูหลักดังต่อไปนี้

(1) วีไอเตรียมการเริ่มต้น (The Initialize VI) เป็นวีไอตัวแรกสุดที่จะต้องใช้ วีไอตัวนี้จะประกอบไปด้วยฟังก์ชั่นที่ประสานงานติดต่อกับตัวเครื่องมือวัด และเตรียมพร้อมการปฏิบัติงานที่สำคัญในการเตรียมเครื่องมือวัดให้อยู่ในสภาพเริ่มต้นหรือสภาพใด ๆ ที่พร้อมที่จะปฏิบัติตามคำสั่งอื่นๆ ต่อไป โดยปรกติแล้ววีไปตัวนี้จะต้องเรียกขึ้นมาทำงานในขณะเริ่มต้นของโปแกรมที่พัฒนาทุกครั้ง

(2) วีไอจัดการโครงแบบ (The Configuration VI) เป็นกลุ่มของโปรแกรมย่อย ๆ ที่จัดการกับโครงแบบของเครื่องมือวัดสำหรับเตรียมการปฏิบัติงานตามที่ผู้ใช้ต้องการ วีไอจัดการโครงแบบจะมีมากหรือน้อยจะขึ้นอยู่กับฟังก์ชั่นของเครื่องมือวัดนั้น ๆ หลังจากที่โปรแกรมที่สร้างได้เรียกใช้วีไปตัวนี้แล้ว เครื่องมือวัดนั้น ๆ จะพร้อมที่จะวัดสัญญาณหรือจำลองการทำงานของระบบการวัดได้ทันที

(3) วีไอปฏิบัติการ (The Action VI) เป็นการสั่งเครื่องมือวัดปฏิบัติการวัดสัญญาณหรือจำลองการทำงานของระบบการวัด สามารถสั่งให้เริ่มหรือหยุดการปฏิบัติงานได้ สิ่งที่วีไอปฏิบัติการแตกต่างจากวีไอจัดการโครงแบบก็คือมันจะไม่สามารถเปลี่ยนการปรับตั้ง (Setting) เพียงแต่สามารถสั่งเครื่องมือวัดให้กระทำการใดๆ บนโครงแบบปัจจุบันที่ถูกปรับมาจากวีไอจัดการโครงแบบได้เท่านั้น

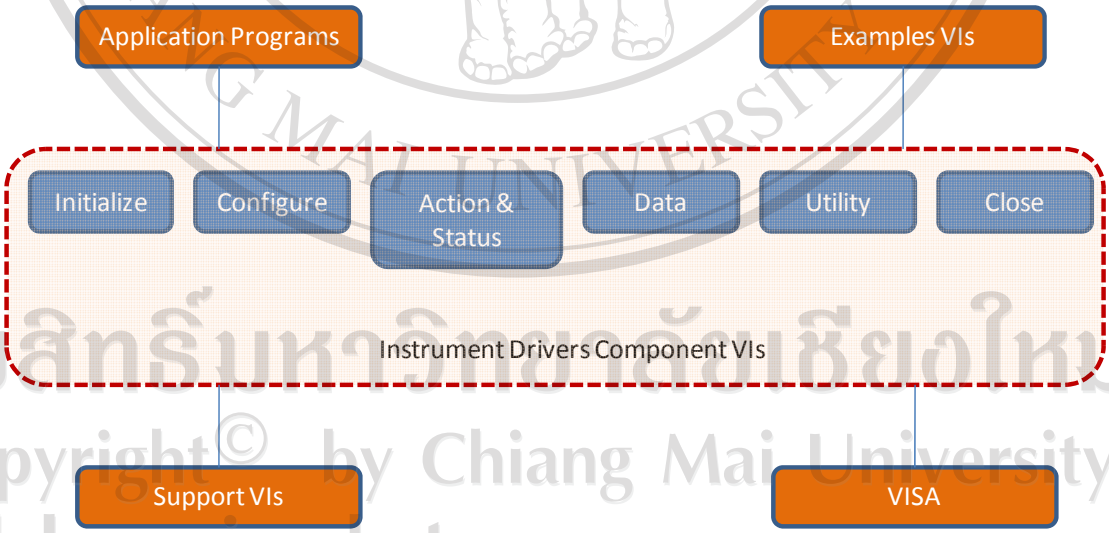
(4) วีไอสถานะ (The Status VI) จะประกอบไปด้วยสถานะปัจจุบันของเครื่องมือวัดหรือสถานะของการปฏิบัติการที่กำลังทำอยู่เป็นต้น

(5) วีไอข้อมูล (The Data VI) สำหรับโอนย้ายข้อมูลระหว่างเครื่องมือวัดกับโปรแกรมที่พัฒนา ตัวอย่างเช่น วีไอย่อยที่อ่านข้อมูลจากค่า/รูปคลื่นที่กำลังวัด หรือวีไปย่อยที่บรรจุข้อมูลจากค่า/รูปคลื่นลงสู่เครื่องมือวัด เป็นต้น

(6) วีไออรรถประโยชน์ (The Utility VI) สำหรับเก็บฟังก์ชั่นต่าง ๆ ที่ใช้บ่อย ๆ ในไดรฟ์เวอร์ เช่น การตั้งใหม่ (Reset), การทดสอบตัวเอง (Self Test), การตรวจสอบรุ่น (Revision), การสอบถามความผิดพลาด (Error Query), การตรวจสอบข้อความแสดงความผิดพลาด (Error Message) เป็นต้น วีไออรรถประโยชน์ นี้อาจจะรวมทั้งการเตรียมการปฏิบัติต่างๆ เหล่านี้ เช่น การสอบเทียบมาตรฐาน (Calibration), การจัดเก็บข้อมูลต่าง (Storage) รวมถึงการเรียกคืนการปรับตั้ง (Recall of Setups) อีกด้วย

(7) วีไอปิด (The Close VI) จะเป็นการหยุดการติดต่อระหว่างตัวซอฟต์แวร์กับตัวเครื่องมือวัดและคืนค่าหน่วยความจำให้แก่ทรัพยากรระบบ โดยปรกติแล้ว จะใช้วีไอปิดนี้เมื่อต้องการจะปิดโปรแกรมที่กำลังพัฒนา หรือในขณะที่เสร็จสิ้นการติดต่อกับเครื่องมือวัดแล้ว และต้องแน่ใจว่าทุกครั้งที่เรียกวีไอเตรียมการเริ่มต้น จะต้องเรียกวีไอปิดควบคู่กันไปด้วยเพื่อรักษาหน่วยความจำของทรัพยากรระบบไม่ให้เสียไปโดยเปล่าประโยชน์

โครงสร้างของไดรฟ์เวอร์ของเครื่องมือวัด ดังกล่าวสามารถแสดงได้ดังรูป 2.6 ดังนี้



รูป 2.6 แสดงโครงสร้างไดรฟ์เวอร์ของเครื่องมือวัด

Dany ได้ทำการศึกษาถึงการพัฒนาและใช้ไดรฟ์เวอร์ของเครื่องมือวัดเสมือนเพื่อที่จะเพิ่มประสิทธิภาพระบบเครื่องมือวัด (10) และได้สรุปว่าเทคโนโลยีไดรฟ์เวอร์ของเครื่องมือได้พัฒนามาอย่างต่อเนื่องในรอบทศวรรษที่ผ่านมา โดยเริ่มจากเทคโนโลยี เอสซีพีไอ (SCPI) และวีเอ็กซ์ไอ

ปลั๊กแอนด์เพลย์ (VXI Plug& Play) ซึ่งสามารถสร้างชุดคำสั่งได้ง่ายด้วยคำสั่งมาตรฐานและด้วยฟังก์ชั่นอรรถประโยชน์ระดับสูง และที่เกิดขึ้นหลังสุดก็คือพื้นฐานการแลกเปลี่ยนข้อมูลระหว่างเครื่องมือวัดที่เรียกว่า ไอวีไอ(Interchangeable Virtual Instrument, IVI) ไอวีไอได้กำหนดมาตรฐานสำหรับการพัฒนาไดรฟ์เวอร์ของเครื่องมือวัดที่สร้างข้อดีมากมายในการปรับปรุงประสิทธิภาพให้สูงขึ้น ข้อดีเหล่านี้ประกอบไปด้วย การแลกเปลี่ยนข้อมูลระหว่างเครื่องมือวัด, การจำลองการวัดทดสอบ และการเก็บสถานะของเครื่องมือวัด และเนื่องจากไดรฟ์เวอร์ของเครื่องมือวัดเป็นส่วนหนึ่งของระบบการวัดสมัยใหม่ ดังนั้นไอวีไอจึงได้สร้างเครื่องมือมาเพื่อช่วยให้นักพัฒนาไดรฟ์เวอร์ได้สะดวกยิ่งขึ้น

งานวิจัยของ Dany นี้จึงมุ่งเน้นการศึกษาความเป็นไปได้ทางเทคนิคว่าโครงสร้างสถาปัตยกรรมของไอวี-ไอมีนัยสำคัญในการช่วยให้นักพัฒนาระบบการวัดปรับปรุงประสิทธิภาพและช่วยลดค่าใช้จ่ายที่เสียไปในการบำรุงรักษาซอฟต์แวร์ในระยะยาวได้หรือไม่

2.5 โพรโทคอลของจีพีไอบี/ไอทริปเปิลอี488 บัส (GPIB/ IEEE488 Bus Protocol)

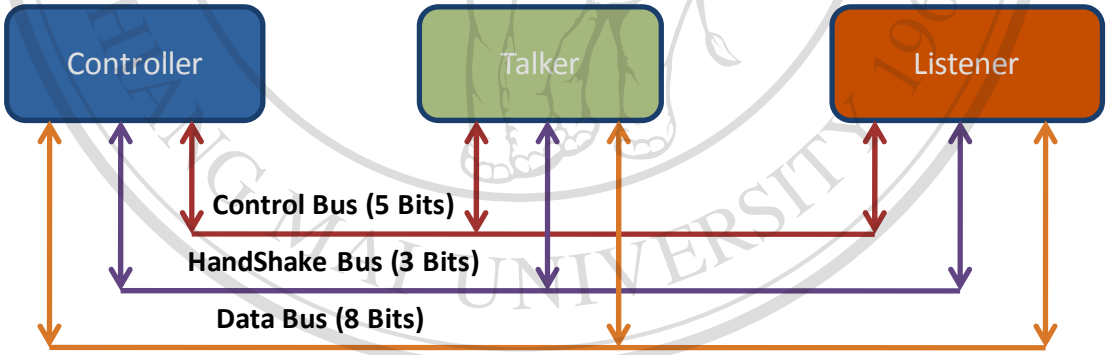
จีพีไอบีบัส (GPIB Bus) (11) ย่อมาจากคำว่า General Purpose Interface Bus เป็นมาตรฐานการสื่อสารเพื่อการขนถ่ายข้อมูลและสื่อสารควบคุมกับเครื่องจักรและเครื่องมือมือวัดต่าง ๆ โดยมาตรฐานนี้ถูกบริษัท Hewlett-Packard พัฒนามาตั้งแต่ปลายทศวรรษ 1960 จึงมีชื่อเรียกอีกอย่างหนึ่งว่า เอชพีไอบีบัส (HPIB Bus) ซึ่งย่อมาจากคำว่า Hewlett-Packard Interface Bus นั่นเอง และได้พัฒนาจนกระทั่งได้รับมาตรฐานจาก Institute of Electrical and Electronic Engineer (IEEE) ในปี 1975 ซึ่งต่อมารู้จักกันในชื่อ IEEE 488 standard

จีพีไอบีเป็นการติดต่อแบบใหม่ที่สามารถเชื่อมต่ออุปกรณ์หลายชิ้นเข้ากับจีพีไอบีพอร์ตตัวเดียวได้ โดยสามารถต่ออุปกรณ์ได้สูงถึง 15 ชั้น โดยใช้บัสเพียงตัวเดียว ทำให้ประหยัดและขจัดปัญหาในการปรับตั้งอุปกรณ์เครื่องมือวัดที่ต่ออยู่ทั้งหมดได้ ลักษณะโดยทั่วไปของการติดต่อแบบจีพีไอบีมีดังนี้

- ส่งผ่านข้อมูลด้วยวิธีแบบขนาน ครั้งละ 1 ไบต์ (8-บิต)
- ฮาร์ดแวร์จะเป็นผู้จัดการเรื่องการติดต่อแลกเปลี่ยนสัญญาณควบคุม (Handshaking) , การกำหนดเวลา (Timing) และอื่นๆ
- ระยะทางความยาวของสายในการส่งสัญญาณประมาณ 2 เมตร - 20 เมตร
- อัตราการส่งผ่านข้อมูล 20,000 กิโลไบต์ต่อวินาที หรือมากกว่า ซึ่งนับว่าเร็วมากเมื่อเทียบกับพอร์ตแบบเก่า
- ใช้รหัสคำสั่งมาตรฐานแอสกี้ (ASCII) ในการติดต่อกับเครื่องมือวัด

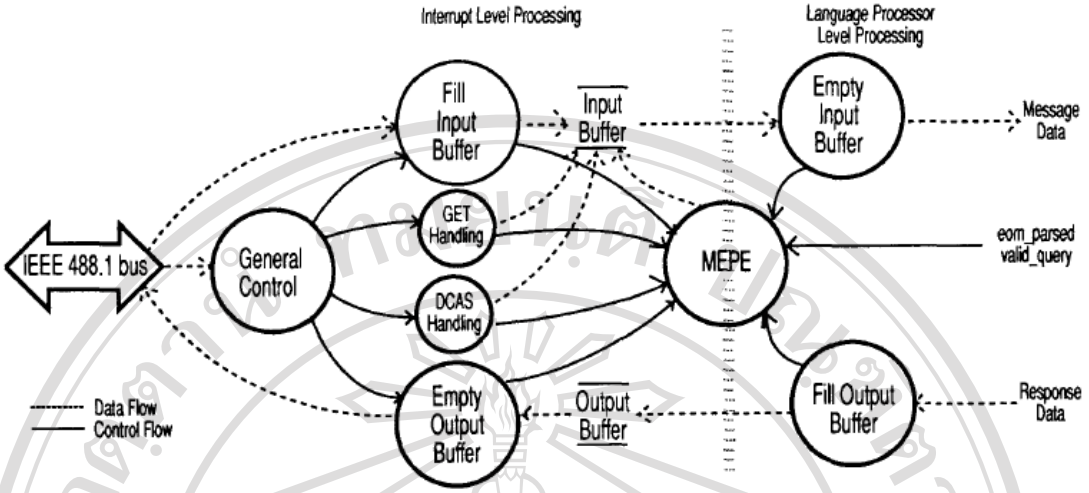
รูป 2.7 แสดงการต่อเครื่องมือวัดกับระบบคอมพิวเตอร์ด้วยจีพีไอบีบัส (12)

โพรโทคอลของจีพีไอบีบัส ใช้เส้นทางสัญญาณ 16 เส้นสำหรับการเชื่อมต่อ ซึ่งได้จำกัดความเร็วโดยเครื่องมือวัดที่เชื่อมต่อและคุณสมบัติทางไฟฟ้า โดย เส้น 16 เส้นถูกแบ่ง คือบัสข้อมูล (Data Bus) มีจำนวน 8 เส้นสำหรับติดต่อข้อมูล และบัสจัดการการสื่อสาร การถ่ายโอนข้อมูลที่เป็นจังหวะเดียวกัน (Hand Check Bus) จำนวน 3 เส้น และบัสควบคุม (Control Bus) 5 เส้นสำหรับการจัดการหน่วยเพื่อควบคุมการใช้เส้นทางดังแสดงในรูป 2.8



รูป 2.8 แสดงโพรโทคอลของจีพีไอบีบัส

Joseph ได้ศึกษาวิจัยประสิทธิผลของการออกแบบเครื่องมือวัดโดยใช้มาตรฐาน IEEE 488.2 (13) ซึ่งประกอบไปด้วยรูปแบบ โพรโตคอล และคำสั่งมาตรฐานกับเครื่องมือวัดที่โปรแกรมได้ มาสร้างเป็นซอฟต์แวร์ไดรฟ์เวอร์ และเขียนเป็นไดอาแกรมได้ดังรูป 2.9 ดังนี้



รูป 2.9 แสดงไดอาแกรมของไดรฟ์เวอร์จากงานวิจัยของ Joseph

งานวิจัยของ Joseph ได้แสดงให้เห็นว่า ประสิทธิผลของการพัฒนาไดรฟ์เวอร์ของเครื่องมือวัดจากมาตรฐาน IEEE488.2 มีประโยชน์ในการแยกส่วนประสานของส่วนนำเข้ากับส่วนนำออก (I/O Interface) ออกจากส่วนควบคุมการปฏิบัติงาน (Execution Control) ซึ่งทำให้การออกแบบง่ายขึ้นเป็นอย่างมาก

2.6 กระบวนการผลิตซอฟต์แวร์แบบคู่ขนาน (Parallel Development)

กระบวนการผลิตซอฟต์แวร์แบบคู่ขนาน เป็นวิธีปฏิบัติที่ประกอบไปด้วยขั้นตอนการดำเนินงานที่เรียงต่อเนื่องกันเป็นลำดับคล้ายกับกระบวนการผลิตซอฟต์แวร์แบบน้ำตก (Waterfall Model) ใช้แบบจำลองกระบวนการผลิตซอฟต์แวร์ที่ใช้แนวทางเชิงวัตถุเป็นหลัก (14) โดยขั้นตอนพื้นฐานในการดำเนินงานผลิตซอฟต์แวร์แบบนี้ มี 5 ขั้นตอนเช่นเดียวกัน ได้แก่ การวางแผน การวิเคราะห์ การออกแบบ การพัฒนา และการประกอบ ดังรูป 2.10

จากงานวิจัยนี้ ซึ่งกำหนดให้มีการพัฒนาไลบรารี่ที่ทำงานเป็นไดรฟ์เวอร์แยกกันเป็นอิสระ มีลักษณะแยกเป็นโมดูล แต่ละโมดูลสามารถแยกกันพัฒนาแบบขนานได้ไปพร้อมกันไม่มีผลกระทบต่อกัน และในขั้นตอนสุดท้ายค่อยนำมาประกอบเข้าด้วยกันเพื่อทดสอบในระดับประกอบ (Integration Test) จึงเห็นว่ากระบวนการผลิตซอฟต์แวร์แบบคู่ขนานมีความเหมาะสมที่สุดต่องานวิจัยนี้

(1) การวางแผน (Planning)

การวางแผนเพื่อพัฒนาระบบจะประกอบไปด้วยการกำหนดมูลค่าทางธุรกิจ ซึ่งจะทำให้เราทราบถึงความต้องการของระบบ หลังจากนั้นจะต้องวิเคราะห์ความเป็นไปได้ในการพัฒนา

ระบบในด้านองค์กร ด้านเศรษฐกิจ และทางด้านเทคนิค จากนั้นเราจึงเอามาพัฒนาแผนการดำเนินงาน ซึ่งจะระบุระยะเวลาที่ใช้ทำงานของแต่ละเฟสหรือการแสดงระยะเวลาเริ่มต้น/สิ้นสุดโครงการในแต่ละช่วง และทราบจำนวนคนที่ต้องการใช้ในแต่ละขั้นตอน ทำให้ควบคุมและกำหนดทิศทางของโครงการได้

(2) การวิเคราะห์ระบบ (Analysis)

เป็นการวิเคราะห์ระบบงานปัจจุบันเพื่อค้นหาปัญหาและนำไปกำหนดความต้องการของระบบงานใหม่ ผลลัพธ์ที่จะได้จากขั้นตอนนี้ ก็คือแบบจำลองที่จะใช้ในการพัฒนาระบบ และแนวทางกระบวนการวิเคราะห์ข้อมูลของแบบจำลองนั้น

(3) การออกแบบระบบ (Design)

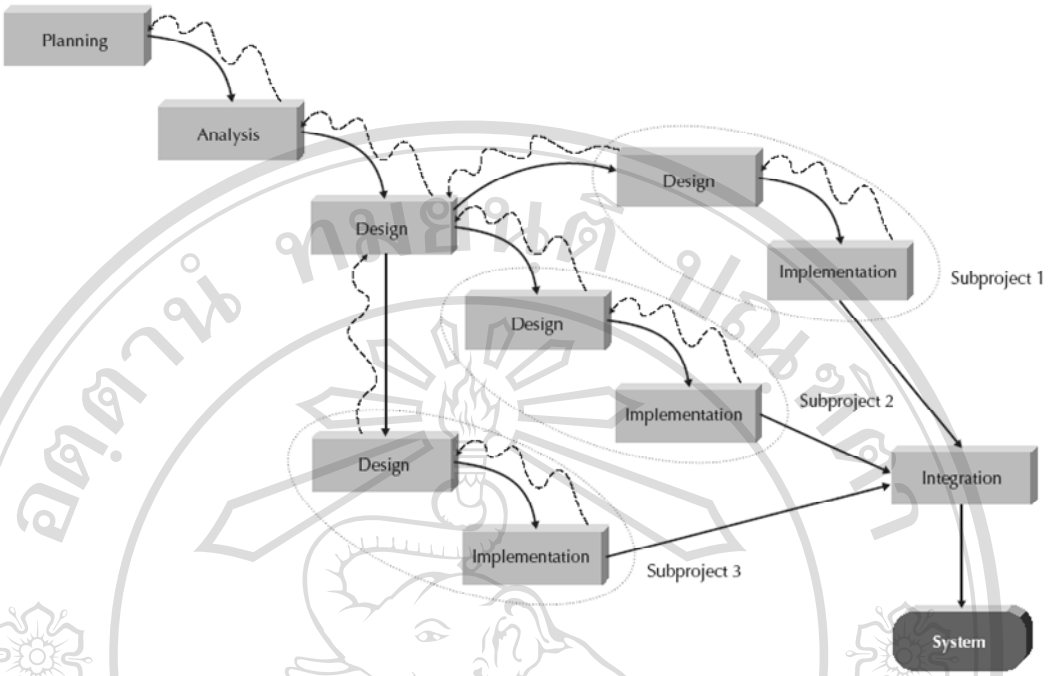
เป็นการออกแบบระบบงานใหม่ให้เป็นไปตามการวิเคราะห์ข้อมูลที่เคยกระทำไว้ในขั้นตอนการวิเคราะห์ระบบ จะประกอบไปด้วยการวางแผนในการออกแบบ และมีการออกแบบโครงสร้างด้านสถาปัตยกรรม (Architecture design) การออกแบบส่วนติดต่อกับผู้ใช้งาน (User Interface design) รวมถึงออกแบบฐานข้อมูลและแฟ้มข้อมูล (Design database and files) อีกด้วย ในขั้นตอนนี้จะมีการออกแบบแต่ละโมดูลแยกกันออกไป และสามารถออกแบบขนานกันไปได้ เนื่องจากการออกแบบเป็นอิสระต่อกัน แต่ละโมดูลไม่ขึ้นอยู่ด้วยกัน

(4) การพัฒนา (Implementing)

เป็นการพัฒนาโปรแกรมให้ได้ตามข้อกำหนดที่ได้ออกแบบเอาไว้ ติดตั้งซอฟต์แวร์จนสามารถใช้งานได้ ขั้นตอนนี้จะประกอบไปด้วยขั้นตอนการเขียนชุดคำสั่ง ซึ่งจะได้แผนการทดสอบโปรแกรม (Test Plan) และเอกสารประกอบการใช้งาน (Manual Documentation) ในขั้นตอนนี้จะมีการพัฒนาแยกกันออกไปเช่นเดียวกับขั้นตอนการออกแบบ และสามารถพัฒนาขนานกันไปได้ เนื่องจากการพัฒนาเป็นอิสระต่อกัน แต่ละโมดูลไม่ขึ้นอยู่ด้วยกัน

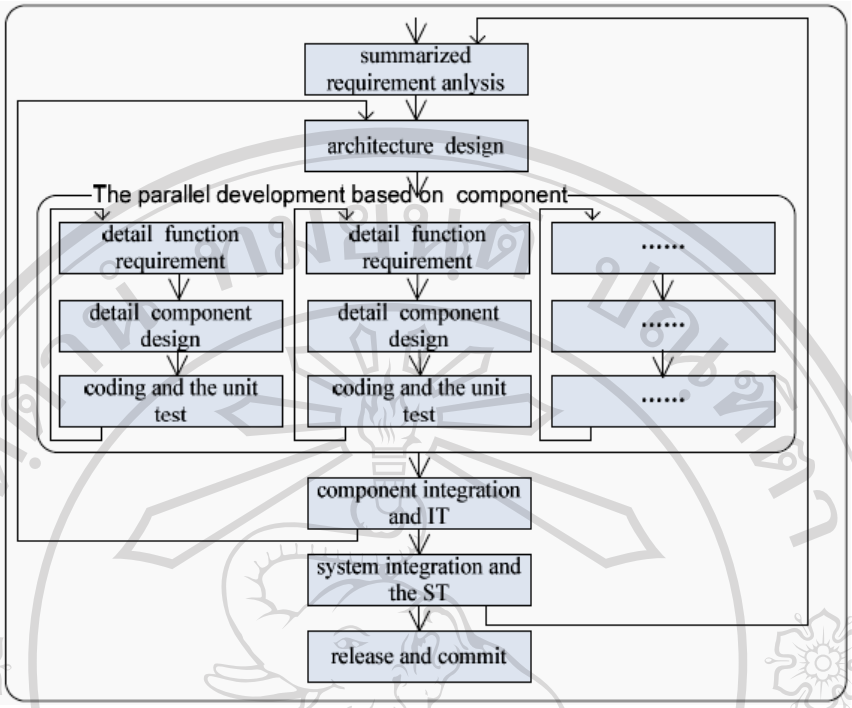
(5) การประกอบ (Integration)

เป็นการนำเอาแต่ละโมดูลที่ได้จากขั้นตอนการออกแบบและพัฒนามารวมเข้าด้วยกันและทดสอบร่วมกันด้วยแผนการทดสอบระบบ (System Test Plan)



รูป 2.10 แบบจำลองกระบวนการผลิตซอฟต์แวร์แบบคู่ขนาน (15)

Hai-tao Li and others (2008) ได้ศึกษาเรื่องแบบจำลองการพัฒนาแบบคู่ขนานและการประกอบแบบอิสระสำหรับงานวิจัยทางวิทยาศาสตร์ในสาขาวิศวกรรมซอฟต์แวร์ (16) และได้สรุปว่า แบบจำลองนี้เหมาะสมอย่างยิ่งต่อความต้องการทางด้านการพัฒนาซอฟต์แวร์สำหรับงานวิจัยทางวิทยาศาสตร์ โดยได้แบ่งการพัฒนาออกเป็น 6 ระยะใหญ่ ๆ ได้แก่ การรวบรวมวิเคราะห์ความต้องการ (Summarized Requirement Analysis), การออกแบบเชิงสถาปัตยกรรม (Architecture Design), การพัฒนาแบบคู่ขนานโดยใช้ส่วนประกอบต่างๆ เป็นพื้นฐาน(Parallel Development Based-On Component), การประกอบส่วนประกอบต่าง ๆ และการทดสอบระดับการประกอบ (Component Integration and Integration Test), การประกอบระดับระบบและการทดสอบระดับระบบ (System Integration and System Test) และการปล่อยและเสนอซอฟต์แวร์ (Release and Submission) ดังแสดงในรูป 2.11



รูป 2.11 แสดงแบบจำลองการพัฒนาซอฟต์แวร์แบบคู่ขนานในงานวิจัยของ Hai-tao Li และคณะ

โดยในระยะที่ 3 ซึ่งจะเป็นระยะที่มีการพัฒนาแบบคู่ขนานนั้น จะแยกองค์ประกอบของซอฟต์แวร์ไปเป็นส่วนย่อยๆ ตามส่วนประกอบต่างๆ ที่เรียกว่า คอมโพเน้นท์ (Component) ในขั้นตอนนี้ ในอนาคตจะเน้นและแบ่งออกอีกเป็นส่วนย่อย ๆ ดังนี้ การวิเคราะห์ฟังก์ชั่นแบบละเอียด (Detail Function Requirement Analysis), การออกแบบส่วนประกอบแบบละเอียด (Detail Component Design), การเขียนรหัสคำสั่งโปรแกรมและการทดสอบระดับหน่วยในแต่ละส่วนประกอบ (Coding and Unit Test)

2.7 มาตรฐานไอเอสโอ 12207 (ISO 12207)

มาตรฐานสำหรับกระบวนการผลิตและพัฒนาซอฟต์แวร์ มาตรฐานไอเอสโอ 12207 เป็นเกณฑ์คุณภาพของการผลิตซอฟต์แวร์ที่มีลักษณะเป็นโปรเซสโมเดลลิ่ง (Process Modeling) คือ เน้นในส่วนของการกำหนดขั้นตอนทีละขั้นตอน (Process) ในการผลิตซอฟต์แวร์ที่มีคุณภาพ ตั้งแต่เริ่มต้นจนกระทั่งจบขั้นตอนของการผลิตซอฟต์แวร์ เพื่อให้การผลิตซอฟต์แวร์นั้นมีคุณภาพ โดยจะมีการกำหนดว่าจะมีผลลัพธ์จากขั้นตอนของการผลิตซอฟต์แวร์ตามที่กำหนดในมาตรฐานไอเอสโอ 12207