บทที่ 4

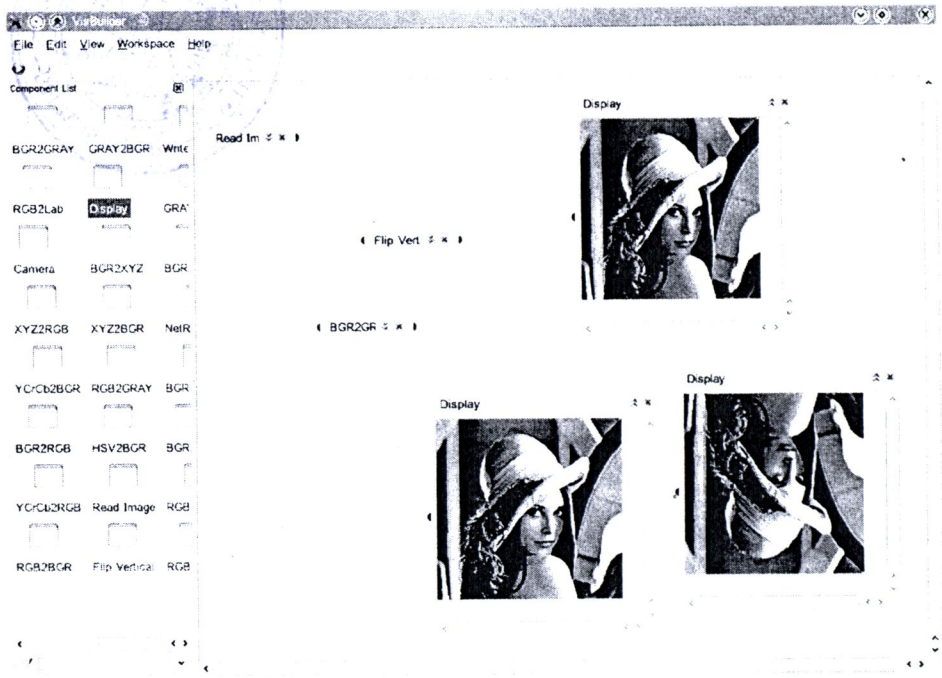
VisBuilder

4.1 ภาพรวมและสถาปัตยกรรมระบบ

ซอฟต์แวร์ที่ออกแบบและสร้างขึ้นนี้ชื่อว่า VisBuilder ซึ่งได้ออกแบบตามความต้องการของระบบดังที่ได้กล่าวไว้ในบทที่ 3 สามารถรองรับงานพัฒนาระบบ การมองเห็นของคอมพิวเตอร์ได้ดังจะได้แสดงในบทที่ 5 VisBuilder สร้างขึ้นด้วยภาษา C/C++ สามารถใช้งานร่วมกับชุดพัฒนาอื่น ๆ ที่สามารถใช้งานผ่านการโปรแกรมด้วยภาษา C/C++ ได้การติดต่อกับผู้ใช้ในแบบวิชวลโดยเน้นการที่ไม่ต้องเขียนโปรแกรมใหม่

ในขณะนี้ VisBuilder สามารถทำงานได้บนระบบปฏิบัติการวินโดวส์และลินุกซ์สามารถต่อกับอุปกรณ์รับภาพที่สามารถทำงานบนระบบปฏิบัติการทั้งสองนี้ได้การทำงานร่วมกันในระบบเครือข่ายคอมพิวเตอร์สามาถทำได้โดยมีองค์ประกอบสำหรับการรับส่งข้อมูลผ่านระบบเครือข่ายที่ได้จัดเตรียมไว้ให้

ส่วนหลักของส่วนต่อประสานกราฟิกกับผู้ใช้แสดงดังรูปที่ 4.1 จะประกอบไปด้วยสามส่วนหลัก คือ ส่วนทางด้านซ้ายสำหรับแสดงองค์ประกอบที่สามารถใช้ได้รวมถึงองค์ประกอบที่ผู้ใช้สร้างขึ้นเองด้วย ส่วนทางด้านขวาเป็นพื้นที่สำหรับให้ผู้ใช้ลากและวาง องค์ประกอบจากด้านซ้ายและสร้างการเชื่อมต่อเพื่อพัฒนาอัลกอริทึมของระบบการมองเห็นของคอมพิวเตอร์ การกำหนดคุณสมบัติของแต่ละองค์ประกอบทำได้โดยดับเบิ้ลคลิกที่องค์ประกอบนั้น ๆ ซึ่งจะแสดงหน้าต่างคุณสมบัติขึ้นมาดังรูปที่ 4.2 และส่วนสุดท้ายซึ่งอยู่ด้านล่างเป็นที่แสดงข้อความของผลการทำงานหรือแสดงข้อความบ่งบอกความผิดพลาดในการทำงานช่วยให้ผู้ใช้หาข้อผิดพลาดในการโปรแกรมได้ง่ายขึ้น



รูปที่ 4.1 ส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder

Properties	Value
File Input	/home/dev/test.jpg
Name	Read Image1

รูปที่ 4.2 ตัวอย่างหน้าต่างคุณสมบัติขององค์ประกอบย่อย

การออกแบบ VisBuilder โดยการพิจารณาระบบเป็นเลเยอร์ดังรูปที่ 4.3 เลเยอร์ที่สูงกว่าจะสร้างจากเลเยอร์ที่ต่ำกว่าเลเยอร์ที่ต่ำกว่าจะไม่สามารถเข้าถึงคลาสหรือ ฟังก์ชันของเลเยอร์ที่สูงกว่าได้ เลเยอร์ทั้งหมดประกอบด้วย แอพพลิเคชันเลเยอร์ (application layer) เฟรมเวิร์คเลเยอร์ (framework layer) เลเยอร์องค์ประกอบย่อย (component layer) เลเยอร์ข้อมูล (data layer) และเลเยอร์การไหลของข้อมูล (dataflow layer)

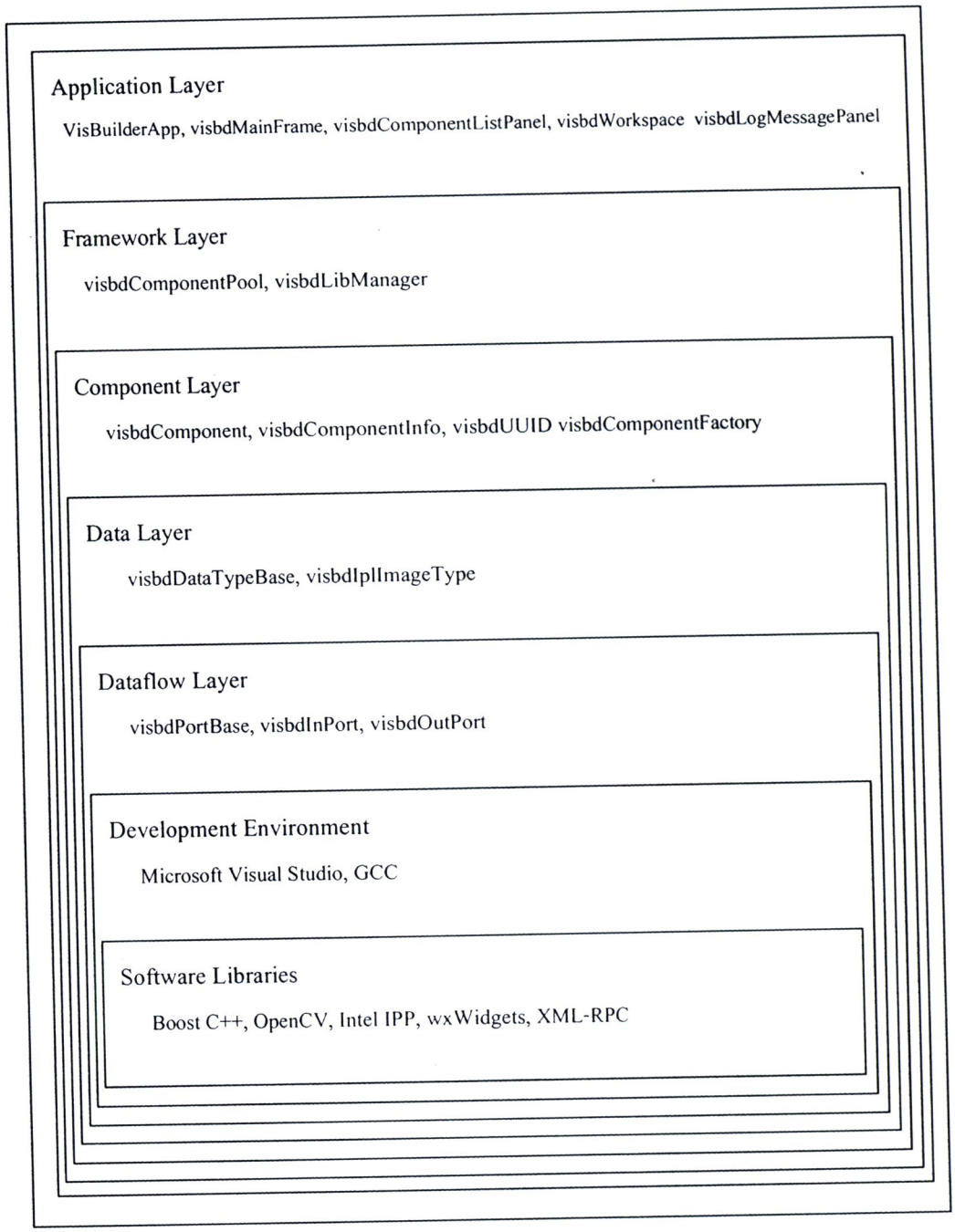
เลเยอร์ที่อยู่ชั้นบนสุด คือ แอพพลิเคชันเลเยอร์ประกอบด้วยคลาสที่รวมคลาสจากเฟรมเวิร์คเข้าด้วยกัน เป็นส่วนที่เชื่อมผู้ใช้กับเฟรมเวิร์คเข้าด้วยกันโดยการใช้การติดต่อกับผู้ใช้ผ่านส่วนต่อประสานกราฟิก คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) VisBuilderApp (2) visbdMainFrame (3) visbdComponentListPanel (4) visbdWorkspace และ (5) visbdLogMessagePanel

เฟรมเวิร์คเลเยอร์ ประกอบด้วยคลาสสำหรับจัดการองค์ประกอบย่อย การเพิ่มหรือลบองค์ประกอบย่อยออกจากระบบการจัดตารางการทำงานขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ visbdComponentPool และ visbdLibManager

เลเยอร์องค์ประกอบย่อย ประกอบด้วยคลาสต้นแบบของการสร้างองค์ประกอบย่อย และคลาสที่เป็นคุณสมบัติขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) visbdComponent (2) visbdComponentInfo (3) visbdUUID และ visbdComponentFactory

เลเยอร์ข้อมูล เป็นส่วนที่ประกอบด้วยคลาสต้นแบบสำหรับการสร้างชนิดข้อมูลที่ใช้ในซอฟต์แวร์และสำหรับจัดการข้อมูลต่าง ๆ คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่คลาส visbdDataTypeBase และคลาสอนุพันธ์จาก visbdDataTypeBase เช่น visbdIplImageType

เลเยอร์การไหลของข้อมูล เป็นเลเยอร์ที่มีคลาสสำหรับจัดการการส่งข้อมูลภายในระหว่างองค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ คลาส visbdCircularBuffe และvisbdPortBaser โดยที่คลาสที่อนุพันธ์จากคลาส visbdPortBase คือ คลาส visbdInPort และคลาส visbdOutPort

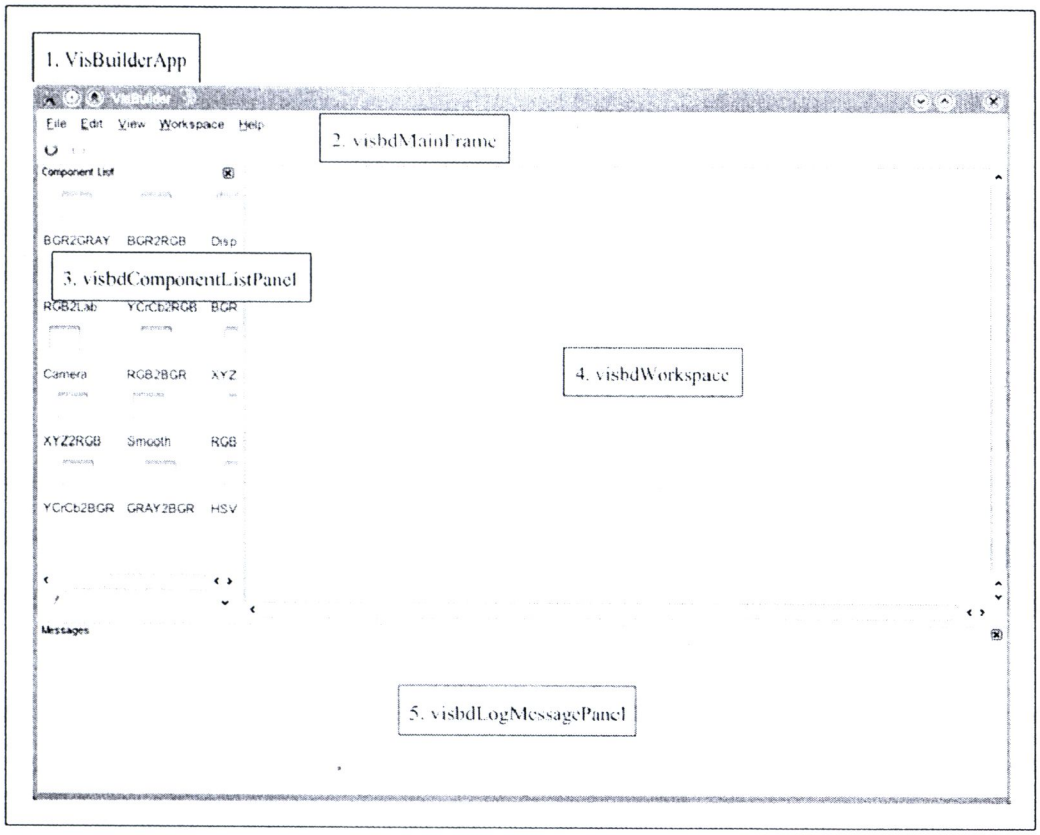


รูปที่ 4.3 การออกแบบสถาปัตยกรรมระบบในลักษณะเลเยอร์

4.2 การออกแบบระบบ

4.2.1 แอพพลิเคชันเลเยอร์ (Application Layer)

แอพพลิเคชันเลเยอร์เป็นเลเยอร์บนสุดของ VisBuilder ทำหน้าที่เชื่อมผู้ใช้กับเฟรมเวิร์คเลเยอร์เข้าด้วยกัน โดยการใช้การติดต่อกับผู้ใช้ผ่านส่วนต่อประสานกราฟิก และยังทำหน้าที่ค้นหาองค์ประกอบที่มีอยู่ในระบบประกอบด้วยห้าคลาส คือ (1) VisBuilderApp (2) visbdMainFrame (3) visbdComponentListPanel (4) visbdWorkspace และ (5) visbdLogMessagePanel การจัดวางของคลาสที่สัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้แสดงดังรูปที่ 4.4



รูปที่ 4.4 คลาสที่สัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ของ VisBuilder

4.2.1.1 คลาส VisBuilderApp

คลาส VisBuilderApp คือ ตัว VisBuilder เป็นจุดเริ่มต้นของระบบมีความสัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ดังรูปที่ 4.4 ซึ่งคือ ตำแหน่งหมายเลข 1 ใช้สำหรับการกำหนดค่าของซอฟต์แวร์นี้และทำหน้าที่จัดการกับ event loop ของวินโดว์คลาสนี้ อนุพันธ์มาจากคลาส wxApp

4.2.1.2 คลาส visbdMainFrame

คลาส visbdMainFrame ทำหน้าที่สร้างเฟรมวินโดว์ซึ่งเป็นส่วนหลักของส่วนต่อประสานกราฟิก คลาสนี้อนุพันธ์มาจากคลาส wxFrame สำหรับจัดวางองค์ประกอบต่าง ๆ ของส่วนต่อประสานกราฟิก เช่น เมนูบาร์ (menubar) ทูลบาร์ (toolbar) สเตตัสบาร์ (statusbar) เป็นต้น ซึ่งเป็นการกำหนดหน้าตาของ VisBuilder เฟรมวินโดว์ที่สร้างขึ้นนี้สามารถปรับเปลี่ยนขนาดได้และสามารถบรรจุเฟรมวินโดว์ลงไปได้ตำแหน่งของคลาสนี้อยู่ที่หมายเลข 2 ของรูปที่ 4.4

4.2.1.3 คลาส visbdComponentListPanel

คลาส visbdComponentListPanel อนุพันธ์มาจากคลาส wxPanel ทำหน้าที่แสดงองค์ประกอบย่อยที่มีอยู่ในระบบซึ่งสามารถนำไปใช้ได้และยังมีความสามารถค้นหาองค์ประกอบย่อยจากชื่อได้มีความสัมพันธ์กับส่วนต่อประสานกราฟิกกับผู้ใช้ดังรูปที่ 4.4 ในตำแหน่งหมายเลข 3

4.2.1.4 คลาส visbdWorkspace

คลาส visbdWorkspace พัฒนาหรืออนุพันธ์มาจากคลาส wxScrolled-Window ตำแหน่งของคลาสนี้อยู่ที่หมายเลข 4 ของรูปที่ 4.4 ทำหน้าที่แสดงองค์ประกอบย่อยและการเชื่อมต่อในรูปแบบกราฟิกเป็นจุดสั่งให้ระบบการมองเห็นที่ออกแบบไว้หยุดหรือทำงาน

คลาสนี้สั่งให้ระบบทำงานโดยเริ่มจากการจัดตารางการทำงานขององค์ประกอบย่อยโดยใช้ข้อมูลจากคลาส ComponentPool ที่อยู่ในเฟรมเวิร์คเลเยอร์เป็นตัวตัดสินใจว่าองค์ประกอบย่อยใดควรทำงานก่อนหลัง

การพัฒนาระบบการมองเห็นของคอมพิวเตอร์จะกระทำที่ตำแหน่งนี้ผู้ใช้สามารถลากองค์ประกอบย่อยจากตำแหน่งที่ 3 ในรูปที่ 4.4 มาวางลงเพื่อเริ่มสร้างระบบการมองเห็นและจัดวางการเชื่อมต่อเข้าด้วยกัน

4.2.1.5 คลาส visbdLogMessagePanel

คลาส visbdLogMessagePanel สำหรับแสดงข้อมูลของความผิดพลาดต่าง ๆ ของการทำงานในรูปแบบข้อความ และยังสามารถใช้แสดงข้อมูลอื่นเพิ่มเติมได้หากผู้พัฒนาต้องการคลาสนี้อนุพันธ์มาจากคลาส wxPanel ซึ่งอยู่ในตำแหน่งที่ 5 ของรูปที่ 4.4

4.2.2 เฟรมเวิร์คเลเยอร์ (Framework Layer)

โครงสร้างของระบบ VisBuilder แสดงในรูปที่ 4.5 โดยในส่วนหลักหรือ VisComponent แบ่งออกเป็นองค์ประกอบย่อย 3 ประเภทหลัก องค์ประกอบทั้งสามแบบถูกจัดให้เป็นองค์ประกอบพื้นฐาน (basic component) ของระบบ และคลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ visbdComponentPool และ visbdLibManager

4.2.2.1 องค์ประกอบแหล่งจ่าย (Source Component)

เป็นองค์ประกอบย่อยในการรับภาพจากอุปกรณ์รับภาพไฟล์ภาพหรือข้อมูลอื่น ๆ จากภายนอกเข้าสู่ตัวซอฟต์แวร์ได้แก่ องค์ประกอบย่อยการรับภาพจากกล้อง (viscomCamera) องค์ประกอบย่อยการอ่านไฟล์ภาพ (viscomImageReader) และองค์ประกอบย่อยการรับข้อมูลจากระบบเครือข่ายคอมพิวเตอร์ (visbdNetInput)

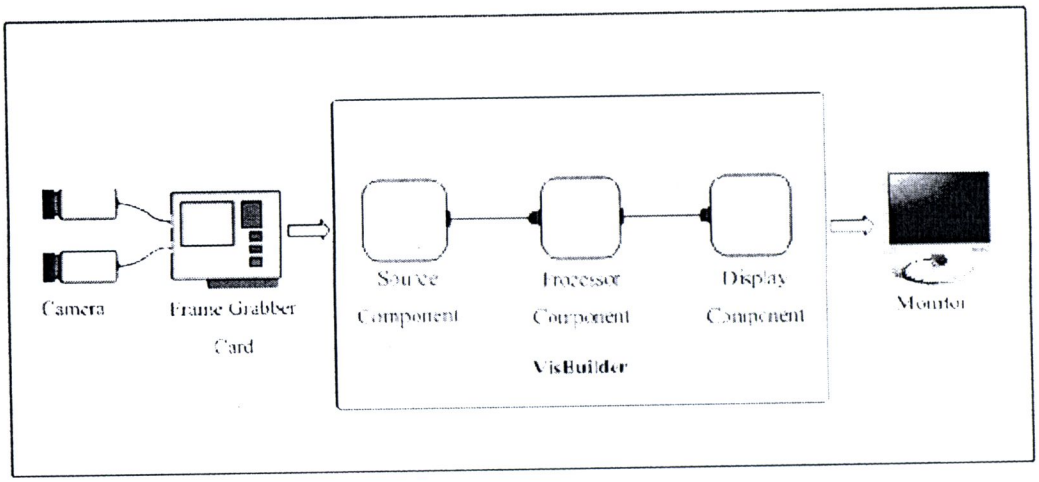
4.2.2.2 องค์ประกอบการประมวลผล (Processing Component)

เป็นองค์ประกอบย่อยในการประมวลผลสัญญาณภาพหรือดำเนินการต่าง ๆ กับข้อมูลภายในและข้อมูลที่รับเข้ามา องค์ประกอบย่อยในส่วนนี้มีมากมายหลายชนิดและสามารถพัฒนาเพิ่มเติมได้ซึ่งในขณะนี้ได้พัฒนาขึ้นบางส่วน ตัวอย่างเช่น

องค์ประกอบย่อยสำหรับการแปลงระบบสีของไฟล์รูปภาพ เช่น (1) viscomRgb2Gray (2) viscomGray2Rgb (3) viscomRgb2Hsv เป็นต้น องค์ประกอบย่อยการหมุนภาพ เช่น (1) viscomFlipVertical (2) viscomFlipHorizontal (3) viscomRotate เป็นต้น และองค์ประกอบย่อยชนิดอื่น ๆ ที่ยังอยู่ในระหว่างการพัฒนา

4.2.2.3 องค์ประกอบการแสดงผล (Display Component)

เป็นองค์ประกอบย่อยในการแสดงผลภาพหรือส่งข้อมูลไปยังอุปกรณ์ภายนอกที่ใช้แสดงผลได้แก่ องค์ประกอบย่อยการแสดงภาพบนหน้าจอ (viscomMonitor) องค์ประกอบย่อยการบันทึกไฟล์ภาพ (viscomImageWriter) และองค์ประกอบย่อยการส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ (visbdNetOutput)



รูปที่ 4.5 โครงสร้างการทำงานของ VisBuilder

4.2.2.4 คลาส visbdComponentPool

คลาส visbdComponentPool เป็นคลาสสำหรับจัดเก็บองค์ประกอบย่อยที่ผู้ใช้เพิ่มเข้าไปในระบบการเพิ่มหรือลบองค์ประกอบย่อยทั้งสามข้างต้นจะดำเนินการภายในคลาสนี้

4.2.2.5 คลาส visbdLibManager

คลาส visbdLibManager เป็นคลาสสำหรับค้นหาไฟล์ที่บรรจุองค์ประกอบย่อย และเพิ่มองค์ประกอบย่อยนั้นเข้าไปในระบบของ VisBuilder เพื่อให้ผู้ใช้ใช้นำมาใช้งานได้ไฟล์ที่บรรจุองค์ประกอบย่อยสำหรับระบบปฏิบัติการวินโดวส์ คือ ไฟล์ DynamicLink Library (DLL) ซึ่งจะมีนามสกุลเป็น *.dll และสำหรับในกรณีของระบบปฏิบัติการลินุกซ์ คือ ไฟล์Shared Object มีนามสกุลเป็น *.so

เมื่อทำการโหลดไฟล์ที่บรรจุองค์ประกอบย่อย ข้อมูลทั้งหมดขององค์ประกอบย่อยจะถูกนำไปเก็บไว้ในคลาส visbdComponentFactoryTable ซึ่งจะจัดเก็บอินสแตน (instance) ของคลาส visbdComponentFactory ไว้ภายใน ดังจะได้กล่าวในหัวข้อ 4.2.3.4

4.2.3 เลเยอร์องค์ประกอบย่อย (Component Layer)

เลเยอร์องค์ประกอบย่อย ประกอบด้วยคลาสต้นแบบของการสร้างองค์ประกอบย่อยและคลาส ที่เป็นคุณสมบัติขององค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ได้แก่ (1) visbdComponent (2) visbdComponentInfo (3) visbdUUID (4) visbdComponentFactory

4.2.3.1 คลาส visbdComponent

คลาส visbdComponent เป็นคลาสฐานของการสร้างองค์ประกอบย่อย ทุกองค์ประกอบย่อยที่พัฒนาขึ้นจะต้องอนุพันธ์จากคลาสนี้ทั้งหมดโดยไม่มีข้อยกเว้น

คลาสนี้ประกอบด้วย4ฟังก์ชันหลักซึ่งเป็นฟังก์ชันที่องค์ประกอบย่อยที่พัฒนาขึ้นจะต้องเขียนรหัสโปรแกรมเพิ่มเติมลงไปได้แก่ ฟังก์ชัน on_create() ฟังก์ชัน on_delete() ฟังก์ชัน process() และฟังก์ชัน on_create_properties_page()

ฟังก์ชัน on_create() จะทำงานในขณะที่มีการเพิ่มองค์ประกอบย่อยเข้าไปในระบบ จึงเป็นส่วนของการกำหนดค่าเริ่มต้นต่าง ๆ ขององค์ประกอบย่อย

ฟังก์ชัน on_delete() ทำงานเมื่อมีการลบองค์ประกอบย่อยออกจากระบบ องค์ประกอบย่อยที่ต้องการให้มีการดำเนินการใด ๆ ก่อนถูกลบจากระบบเช่นบันทึกไฟล์ เป็นต้น สามารถทำได้ในฟังก์ชันนี้

ฟังก์ชัน process() ทำงานเมื่อมีการสั่งระบบให้เริ่มการประมวลผลการดำเนินการคำนวณใด ๆ ขององค์ประกอบย่อยต้องเขียนขึ้นภายในฟังก์ชันนี้

ฟังก์ชัน on_create_properties_page() จะทำงานเมื่อผู้ใช้ต้องการกำหนดค่าคุณสมบัติขององค์ประกอบย่อย ซึ่งจะแสดงหน้าต่างคุณสมบัติขององค์ประกอบนั้น ๆ ขึ้นมาให้ผู้ใช้ทำการปรับแต่ง ฟังก์ชันนี้ช่วยให้มีความยืดหยุ่นในการเพิ่มส่วนต่อประสานกราฟิกกับผู้ใช้สามารถกำหนดหน้าตา ข้อความ หรือปุ่มกดได้ตามความต้องการ

สำหรับฟังก์ชันทั้งสี่นั้นมีเพียงสองฟังก์ชันที่บังคับให้เขียนรหัสโปรแกรมลงไปคือ ฟังก์ชัน on_create() และฟังก์ชัน process() ไม่เช่นนั้นองค์ประกอบย่อยที่พัฒนาขึ้นนี้จะไม่สามารถใช้งานได้

4.2.3.2 คลาส visbdComponentInfo

คลาส visbdComponentInfo เป็นคลาสที่จัดเก็บข้อมูลขององค์ประกอบย่อยอาทิเช่น ชื่อคำอธิบาย รูปไอคอนและตัวระบุที่ไม่ซ้ำกัน (GUID) ข้อมูลทั้งหมดจะถูกนำไปแสดงผลเพื่อให้ผู้ใช้สามารถเลือกใช้องค์ประกอบย่อยได้อย่างถูกต้อง

4.2.3.3 คลาส visbdUUID

คลาส visbdUUID ใช้เก็บค่าตัวระบุที่ไม่ซ้ำกัน (GUID) เป็นค่าสำหรับจำแนกชนิดขององค์ประกอบย่อยต่าง ๆ ซึ่งแต่ละชนิดจะต้องมีค่าไม่ซ้ำกัน เพื่อให้การสร้างองค์ประกอบย่อยเข้าไปในระบบทำได้ง่าย เมื่อผู้ใช้เลือกองค์ประกอบย่อยเพื่อเพิ่มเข้าไปในระบบค่าข้อมูลนี้จะส่งไปให้ระบบเพื่อค้นหาชนิดขององค์ประกอบย่อยที่ระบุนี้แล้วทำการสร้างขึ้น

การสร้างองค์ประกอบย่อยเพิ่มเติมต้องกำหนดค่าตัวระบุที่ไม่ซ้ำกันด้วยตัวเอง ซึ่งสามารถสร้างได้จากคำสั่ง uuidgen ของระบบปฏิบัติการลินุกซ์หรือใช้โปรแกรม GUID Generator ของไมโครซอฟต์ Visual Studio สำหรับระบบปฏิบัติการวินโดวส์ ตัวอย่าง เช่น "21852B6DFD01-4eda-817F-2B8B67C66C19"

4.2.3.4 คลาส visbdComponentFactory

คลาส visbdComponentFactory เป็นคลาสสำหรับการสร้างองค์ประกอบย่อยหรือเป็นโรงงานผลิดองค์ประกอบย่อยนั่นเองทุกองค์ประกอบย่อยจะมีโรงงานผลิดเป็นของตัวเองซึ่งจะถูกโหลดมาด้วยคลาส visbdLibManager ชื่อโรงงานนี้จะกำหนดด้วยค่าตัวระบุที่ไม่ซ้ำกันของแต่ละองค์ประกอบย่อย เมื่อผู้ใช้เลือกองค์ประกอบย่อยที่จะเพิ่มเข้าไปในระบบค่าตัวระบุที่ไม่ซ้ำกันขององค์ประกอบย่อยที่เลือกจะส่งไปให้กับระบบเพื่อค้นหาโรงงานที่ผลิตองค์ประกอบย่อยดังกล่าวแล้วผลิดออกมา การใช้วิธีการเช่นนี้ช่วยให้การพัฒนาเพิ่มเติมองค์ประกอบย่อยใหม่ทำได้โดยไม่ต้องคอมไพล์ VisBuilder ใหม่

4.2.4 เลเยอร์ข้อมูล (Data Layer)

เลเยอร์ข้อมูลเป็นส่วนของการนิยามชนิดข้อมูลที่ใช้ในการประมวลผลของ VisBuilder ทั้งหมดคลาสที่จัดอยู่ในเลเยอร์นี้ ได้แก่ คลาส visbdDataTypeBase เป็นคลาสฐานสำหรับชนิดข้อมูลทั้งหมดและเป็นคลาสแบบนามธรรม (abstract class) การสร้างชนิดข้อมูลใหม่ต้องทำการอนุพันธ์จากคลาสนี้ตัวอย่างคลาสที่อนุพันธ์จากคลาสนี้ คือ คลาส visbdIplImageType เป็นชนิดข้อมูลภาพที่สร้างมาจาก IplImage ซึ่งเป็นชนิดข้อมูลภาพของ OpenCV (OpenCV,2009) เนื่องจาก VisBuilder ใช้ชุดพัฒนานี้เป็นแกนในการประมวล

4.2.5 เลเยอร์การไหลของข้อมูล (Dataflow Layer)

เลเยอร์การไหลของข้อมูลเป็นส่วนที่นิยามคลาสสำหรับการส่งผ่านข้อมูลระหว่างองค์ประกอบย่อย คลาสที่จัดอยู่ในเลเยอร์นี้ ได้แก่ คลาส visbdPortBase คลาส visbdCircularBuffer และคลาสที่อนุพันธ์จากคลาส visbdPortBase คือ คลาส visbdInPort และคลาส visbdOutPort

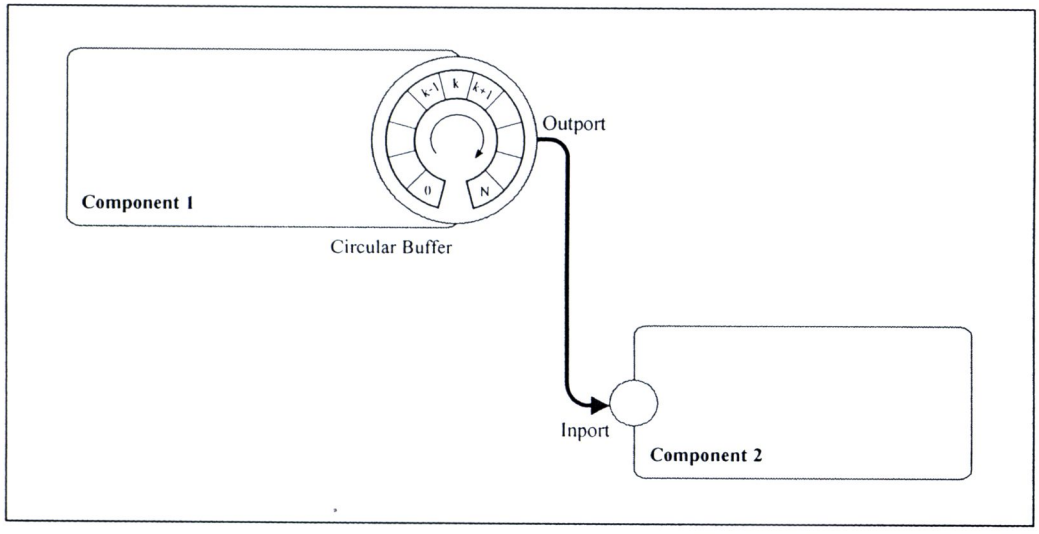
4.2.5.1 คลาส visbdPortBase

การเชื่อมต่อระหว่างองค์ประกอบย่อยจะใช้การสร้างพอร์ทสำหรับการเชื่อมต่อขึ้นคลาสสำหรับการสร้างพอร์ทดังกล่าว คือ คลาส visbdPortBase ซึ่งเป็นคลาสฐานของการเชื่อมต่อทั้งหมดในการเชื่อมต่อระหว่างองค์ประกอบย่อยจะแบ่งเป็นสองประเภทได้แก่ การเชื่อมต่อขาเข้าใช้คลาส visbdInPort และการเชื่อมต่อขาออกใช้คลาส visbdOutPort

คลาส visbdInPort สำหรับการเชื่อมต่อข้อมูลขาเข้า ทำหน้าที่บันทึกการเชื่อมต่อขาเข้าว่ามาจากองค์ประกอบย่อยใด ซึ่งก็คือเป็นข้อมูลขาออกขององค์ประกอบย่อยอื่น เมื่อสั่งให้ VisBuilder ทำการประมวลผลคลาสนี้จะร้องขอข้อมูลจากปลายอีกด้านที่เป็นการเชื่อมต่อขาออกขององค์ประกอบย่อยที่เชื่อมต่ออยู่

คลาส visbdOutPort สำหรับการเชื่อมต่อข้อมูลขาออกมีการเก็บข้อมูลจากการประมวลผลไว้ภายในโดยใช้คลาส visbdCircularBuffer ซึ่งเป็นบัฟเฟอร์รูปแบบหนึ่งที่ใช้ในการประมวลผลโดยอาศัยหลักการของการไหลของข้อมูล

การทำงานระหว่างคลาส visbdInPort และคลาส visbdOutPort แสดงดังรูปที่ 4.6 จะเห็นว่าคลาส visbdOutPort เท่านั้นที่มีการเก็บข้อมูลด้วยบัฟเฟอร์แบบหมุน ข้อมูลที่คลาส visbdInPort ต้องการใช้งานจะเข้าถึงด้วยการอ้างอิงไม่มีการคัดลอกไปไว้ที่ตัวคลาส visbdInPort จึงไม่จำเป็นต้องมีการใช้บัฟเฟอร์แบบหมุน การเข้าถึงข้อมูลด้วยการอ้างอิงช่วยลดเวลาการประมวลผลที่ต้องสูญเสียไปกับการคัดลอกข้อมูลได้มาก



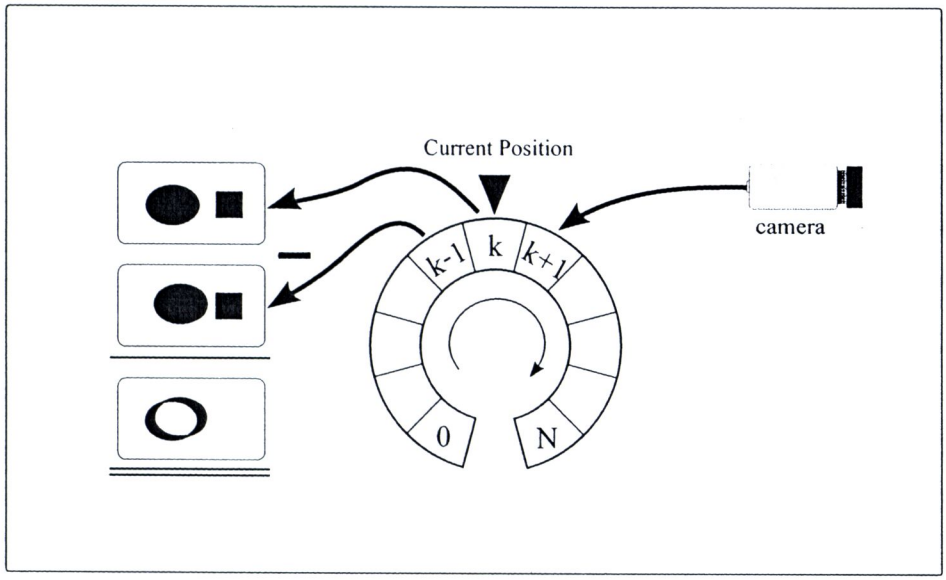
รูปที่ 4.6 ความสัมพันธ์ระหว่างคลาส visbdInPort และคลาส visbdOutPort

4.2.5.2 คลาส visbdCircularBuffer

ในการประมวลผลสัญญาณแบบทันเวลา (real-time processing) จำเป็นที่จะต้องเข้าถึงข้อมูลในขณะปัจจุบัน รวมถึงความสามารถที่จะเข้าถึงข้อมูลก่อนหน้าจำนวนหนึ่งขึ้นกับลักษณะการประมวลผล และการทำงานดังกล่าวจำเป็นต้องมีการปรับเปลี่ยนข้อมูลใหม่อยู่ตลอดเวลา ข้อมูลใหม่เข้ามาแทนที่ข้อมูลชุดเก่าแต่ยังคงต้องรักษาข้อมูลจำนวนหนึ่งที่ผ่านมาไว้ การทำงานลักษณะนี้จะเกิดขึ้นในเวลาเดียวกัน ตัวอย่างเช่น การตรวจจับความเคลื่อนไหวจากไฟล์วีดีโอหรือกล้องรับภาพ วิธีการที่ง่ายที่สุด คือ การหาความแตกต่างระหว่างภาพ (Shuigen et al.,2009) ยิ่งมีความแตกต่างมากก็ยิ่งมีความเคลื่อนไหวมาก การดำเนินการดังกล่าวต้องอาศัยข้อมูลภาพปัจจุบันกับภาพก่อนหน้าซึ่งมีการปรับเปลี่ยนอยู่ตลอดเวลา ภาพใหม่ถูกนำมาแทนที่ภาพเก่า ความจำเป็นดังกล่าวต้องการจำนวนพื้นที่หน่วยความจำจำนวนหนึ่ง หากกำหนดหน่วยความจำมากไปก็จะสิ้นเปลืองน้อยไปก็จะไม่เพียงพอต่อการประมวลผล รวมถึงการจองหน่วยความจำต้องเสียเวลา วิธีการหนึ่งในการจัดการปัญหาดังกล่าวคือการใช้บัฟเฟอร์แบบหมุน (Smith, 1997)

รูปที่ 4.7 แสดงการทำงานของบัฟเฟอร์แบบหมุน เมื่อพิจารณาการตรวจจับความเคลื่อนไหวจากภาพที่รับเข้ามาด้วยกล้องรับภาพ ค่า k เป็นตำแหน่งหน่วยความจำปัจจุบันหรือเป็นภาพปัจจุบัน เมื่อภาพใหม่จากกล้องเข้ามาจะจัดเก็บไว้ที่ตำแหน่ง k+1 ภาพก่อนหน้าอยู่ที่ตำแหน่ง k-1 จะเห็นว่าการประมวลผลสามารถเข้าถึงภาพที่อยู่ก่อนหน้าได้และยังสามารถรับภาพใหม่เข้ามาได้ในขณะเดียวกัน

การจองพื้นที่หน่วยความจำของบัฟเฟอร์แบบหมุนนี้จะเริ่มจองเมื่อสร้างบัฟเฟอร์ชนิดนี้ขึ้นมาในระบบตามขนาดที่ระบุและจะคืนหน่วยความจำเมื่อลบบัฟเฟอร์นี้ออกจากระบบเป็นการทำงานเพียงครั้งเดียว จึงช่วยลดเวลาการทำงานไปได้มาก



รูปที่ 4.7 การทำงานของบัฟเฟอร์แบบหมุน (circular buffer)

4.2.6 สภาวะแวดล้อมการพัฒนา

สภาวะแวดล้อมของการพัฒนาที่ใช้กับ VisBuilder นี้มีสองชนิด คือ Microsoft Visual Studio ในการพัฒนาบนระบบปฏิบัติการวินโดวส์ และ GNU Makefile สำหรับการพัฒนาบนระบบปฏิบัติการลินุกซ์

4.2.7 ชุดเครื่องมือพัฒนาซอฟต์แวร์

ชุดเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ VisBuilder มีดังต่อไปนี้

4.2.7.1 Boost C++

Boost C++ (Karlsson, 2005) เป็นชุดพัฒนาที่ช่วยการเขียนโปรแกรมยืดหยุ่นขึ้น การพัฒนาซอฟต์แวร์จะทำได้รวดเร็วขึ้น ชุดพัฒนานี้มีลักษณะการโปรแกรมเช่นเดียวกับ STL ของภาษา C++ (standard template library) สำหรับซอฟต์แวร์ VisBuilder นี้จะใช้ Boost C++ ในการจัดการหน่วยความจำและจัดการข้อมูลชนิดอาร์เรย์ การนำชุดพัฒนานี้มาใช้ช่วยลดปัญหาในเรื่องการทำงานระหว่างเธรด เนื่องจาก Boost C++ ได้ผ่านการทดสอบการใช้งานในลักษณะนี้มาแล้ว

4.2.7.2 wxWidgets

การออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้ถือเป็นส่วนที่ต้องพิจารณาให้ความสำคัญ เพื่อให้การใช้งาน ซอฟต์แวร์เป็นไปอย่างสะดวกตามวัตถุประสงค์ที่ตั้งไว้ นอกเหนือไปจากเพื่อให้มีความยืดหยุ่น ในการนำไปใช้บนระบบปฏิบัติการต่าง ๆ ได้ด้วย การพัฒนาซอฟต์แวร์ VisBuilder นี้จึงเลือกใช้ชุดพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ที่ชื่อว่า wxWidgets (Smart et al., 2005) เป็นชุดพัฒนาที่ใช้กันอย่างแพร่หลายมีเสถียรภาพสูง เปิดต้นรหัส และทำงานได้หลายแพลทฟอร์ม ทั้งวินโดวส์ แมค และลินุกซ์

4.2.7.3 OpenCV

OpenCV เป็นชุดพัฒนาที่มีประโยชน์มากสำหรับงานพัฒนาระบบการมองเห็นของคอมพิวเตอร์งานวิจัยนี้ได้นำชุดพัฒนานี้มาใช้เป็นส่วนหลักในการพัฒนาโดยใช้สำหรับจัดการโครงสร้างข้อมูลภายในและฟังก์ชันการประมวลผลภาพต่าง ๆ ซึ่งในจะนำมาใช้เป็นแกนหลักในการทำงานของ VisBuilder ทั้งในส่วนที่เป็นโครงสร้างข้อมูล เช่น IplImageType และส่วนการประมวลผลอื่น ๆ องค์ประกอบที่พัฒนาในขณะนี้ใช้การฟังก์ชันของ OpenCV ในการประมวลผล เช่น การแปลงสีภาพ การหมุนภาพ การรับภาพจากกล้อง เป็นต้น

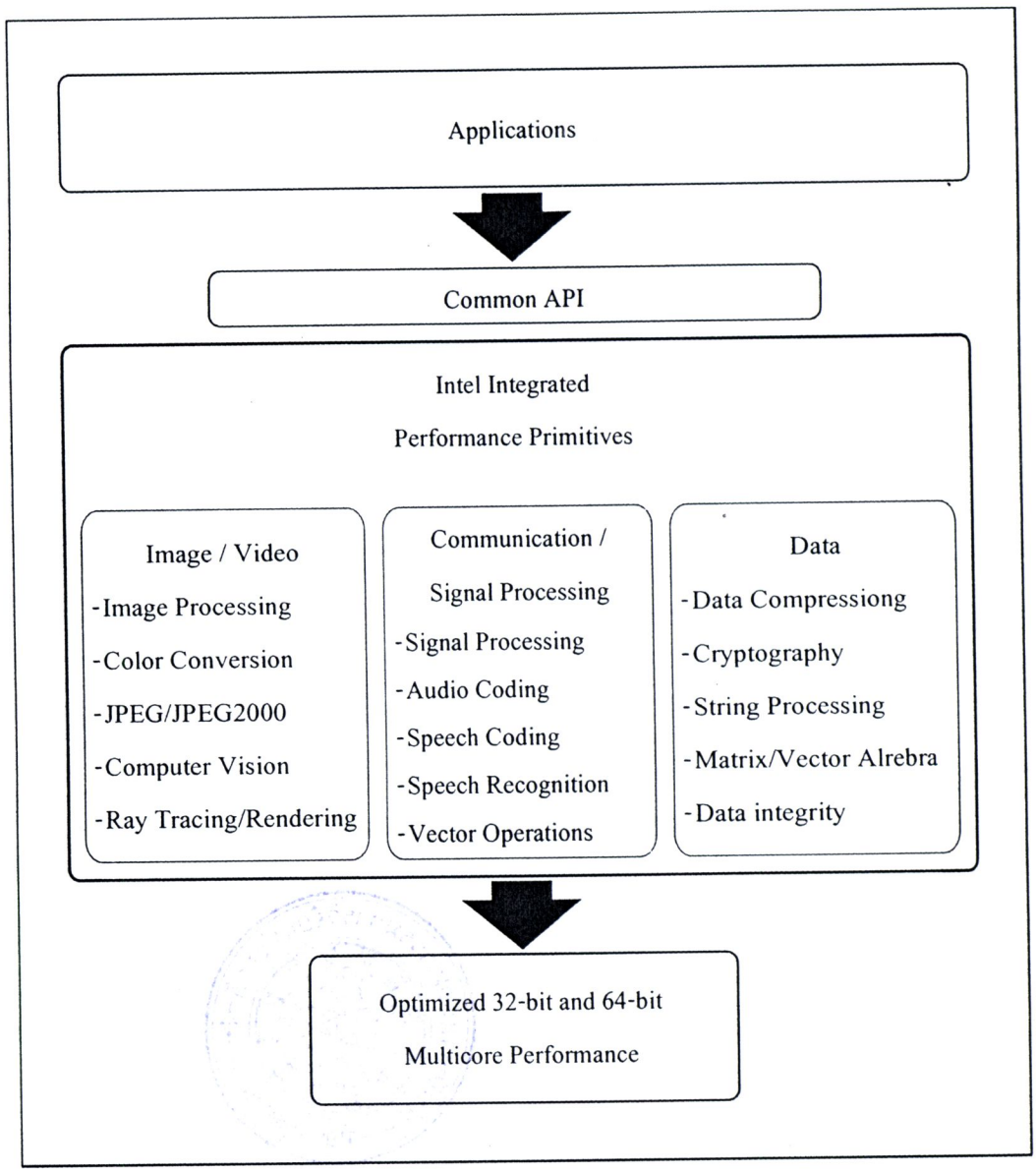
4.2.7.4 Intel IPP

ชุดพัฒนาซอฟต์แวร์สำหรับการประมวลผลที่ใช้ซีพียูของอินเทล (intel integrated performance primitives หรือ IPP) ดังแสดงในรูปที่ 4.8 เป็นชุดพัฒนาซอฟต์แวร์ทางบริษัทอินเทลได้พัฒนาเพื่อใช้งานการประมวลผลที่ต้องการประสิทธิภาพการทำงานสูงสุดบนซีพียูของอินเทลเอง (Intel, 2003) โดยทำเป็นชุดพัฒนาซอฟต์แวร์ในระดับต่ำ (low level software layer) สามารถใช้งานได้ทั้งบนระบบปฏิบัติการ Windows Linux และ Mac มีความยืดหยุ่นสูง สามารถนำต้นรหัสที่เขียนขึ้นไปคอมไพล์ได้ทั้งสามระบบปฏิบัติการโดยไม่ต้องแก้ไขต้นรหัสแต่อย่างใด นอกเหนือไปจากการประมวลผลภาพและสัญญาณ 2 มิติและการมองเห็นของคอมพิวเตอร์แล้ว ในเวอร์ชันล่าสุดของ IPP นั้นมีฟังก์ชันการใช้งานที่หลากหลายครอบคลุมการพัฒนาซอฟต์แวร์ด้านต่าง ๆ ได้ทั่วถึง เช่น การประมวลผลสัญญาณเสียง การคำนวณทางคณิตศาสตร์การบีบอัดข้อมูล เป็นต้น

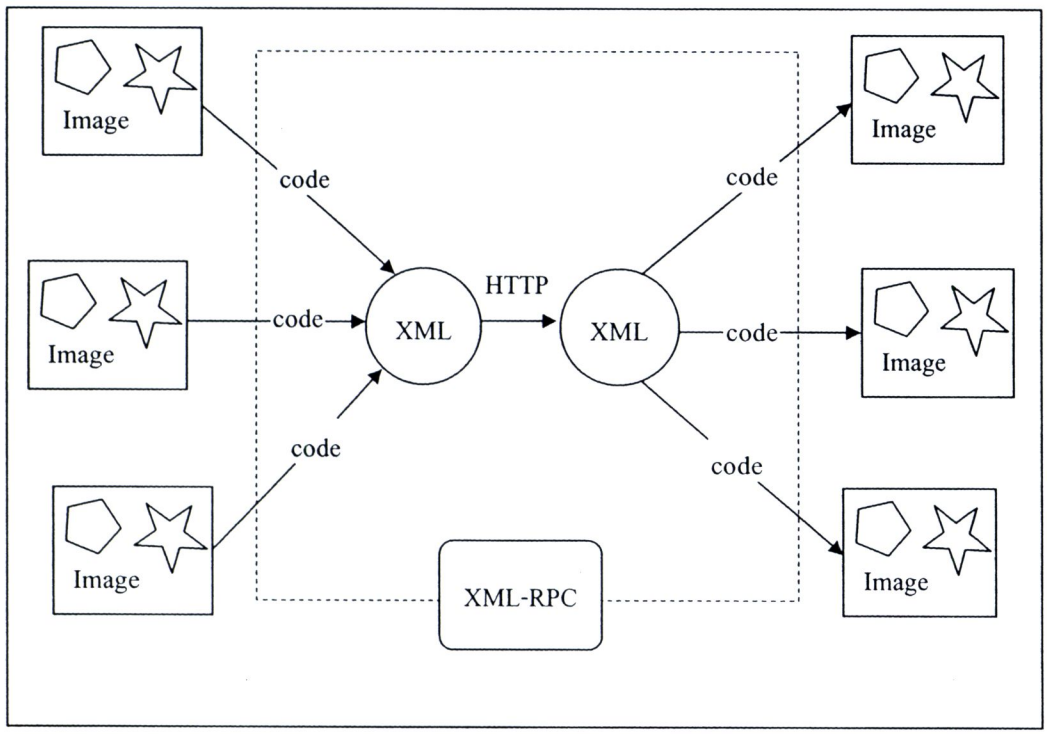
4.2.7.5 XML-RPC

การเรียกกระบวนงานระยะไกล (remote procedure call หรือ RPC) คือ เทคโนโลยีการเรียกกระบวนงานระยะไกล เป็นการอนุญาตให้โปรแกรมบนคอมพิวเตอร์เครื่องหนึ่งสามารถเรียกใช้โปรแกรมที่อยู่บนอีกเครื่องหนึ่งผ่านระบบเครือข่ายคอมพิวเตอร์ได้ (Allman, 2003) เสมือนหนึ่งว่าเป็นการเรียกใช้โปรแกรมบนเครื่องเดียวกัน กล่าวคือ มีการซ่อนการสื่อสารผ่าน ระบบเครือข่ายคอมพิวเตอร์ไว้ภายในในระบบแม่ข่ายลูกข่ายมักจะติดต่อกับแม่ข่ายโดยผ่านทางการเรียกกระบวนงานระยะไกลซึ่งช่วยให้การติดต่อกันระหว่างระบบย่อยเป็นไปได้ด้วยดีและทันเวลา ขั้นตอนการทำงานจะเริ่มที่ลูกข่ายเรียกใช้โปรแกรมบนเครื่องแม่ข่ายผ่านทางการเรียกกระบวนงานระยะไกล แม่ข่ายจะดำเนินการตามโปรแกรมที่ลูกข่ายร้องขอจากนั้นผลลัพธ์ที่ได้จึงถูกส่งกลับไปให้ลูกข่าย การเลือกใช้เทคโนโลยี RPC ดังกล่าวทำให้การใช้ทรัพยากรร่วมกันในระบบการมองเห็นของคอมพิวเตอร์เป็นไปอย่างสะดวกและมีประสิทธิภาพยกตัวอย่าง เช่น อุปกรณ์จับสัญญาณภาพเข้าสู่คอมพิวเตอร์อาจจะไม่จำเป็นจะต้องอยู่ที่เดียวกับระบบประมวลผลสัญญาณภาพก็ได้

รูปที่ 4.9 แสดงการทำงานของ XML-RPC ในการส่งข้อมูลภาพผ่านโปรโตคอล HTTP โดยจะต้องแปลงภาพให้อยู่ในรูปแบบข้อมูล XML แล้วส่งผ่านเครื่องข่ายคอมพิวเตอร์ด้วยโปรโตคอล HTTP เมื่ออีกด้านได้รับข้อมูลจึงแปลงกลับเป็นข้อมูลภาพตามเดิม

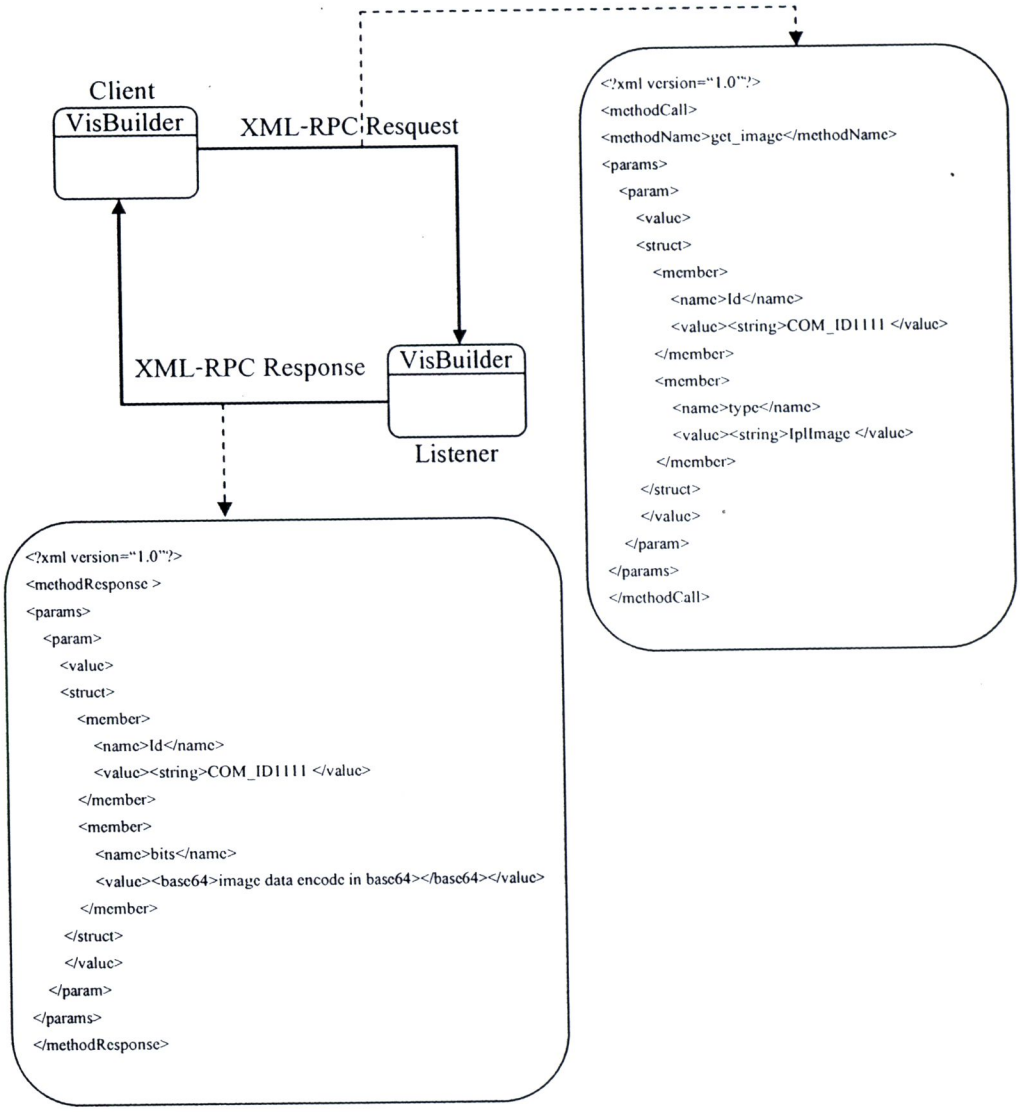


รูปที่ 4.8 โครงสร้างของ Intel IPP



รูปที่ 4.9 การทำงานของ XML-RPC

รูปที่ 4.10 แสดงการทำงานของ VisBuilder ในการส่งข้อมูลภาพด้วย XMLRPC ด้าน VisBuilder ที่เป็น client คือ ผู้ร้องขอข้อมูลจะส่งรูปแบบข้อมูลในรูปแบบของ XML ด้วยข้อกำหนดการใช้งานของ XML-RPC โดยใช้คำสั่งที่ชื่อ methodCall โดยผ่านชื่อฟังก์ชันที่จะเรียกใช้ที่เครื่องปลายทาง ซึ่งในรูป คือ get_image ฝ่าย VisBuilder ที่เป็น listener จะส่งข้อมูลกลับมาในรูปแบบ XML เช่นเดียวกัน โดยจะส่งมาด้วยคำสั่ง methodResponse เพื่อระบุการส่งกลับข้อมูลซึ่งจะมีข้อมูลภาพกลับมาด้วยข้อมูลภาพจะต้องแปลงด้วยวิธี BASE64 เป็นการแปลงข้อมูลไบนารีเป็นข้อมูลที่เป็นข้อความเพื่อให้ใช้กับรูปแบบ XML ได้



รูปที่ 4.10 การส่งข้อมูลของ XML-RPC

ในการทำงานร่วมกันของซอฟต์แวร์นี้ผ่านทางระบบเครือข่ายคอมพิวเตอร์จะใช้การเชื่อมต่อผ่านทางการเรียกกระบวนงานระยะไกล (Sim and Park, 1996) โดยในงานวิจัยนี้ได้เลือกใช้ไลบรารีที่ชื่อว่า Xmlrpc c (Henderson, 2004) สำหรับการโปรแกรมภาษาซีซึ่งจะส่งข้อมูลในรูปแบบ XML (Bray and Maler, 2000) ผ่านทางโปรโตคอล HTTP การพัฒนาตรงส่วนนี้จึงต้องทำการแปลงข้อมูลขององค์ประกอบเป็นข้อมูลในรูปแบบ XML ซึ่งรองรับชนิดข้อมูลตามที่ได้กล่าวไว้ในหัวข้อที่ 4.2.4

4.3 วิธีการออกแบบที่ใช้กับ VisBuilder

กระบวนการออกแบบ VisBuilder ใช้แนวคิดการออกแบบหลายวิธี เช่น การออกแบบเชิงวัตถุ (object oriented design) เพื่อให้ฟังก์ชันอินเตอร์เฟซต่าง ๆ มีความง่ายและชัดเจนในการใช้งานและการซ่อนข้อมูลไว้ภายใน ฉะนั้นจึงทำให้ปลอดภัยจากการเปลี่ยนแปลงอย่างบังเอิญ อาจกล่าวได้ว่าข้อมูลและฟังก์ชันของข้อมูลถูก encapsulate ไปเป็นก้อนเดียวกันรวมถึงความสามารถในการนำกับมาใช้ใหม่ได้ (reusability) กล่าวคือ เมื่อได้เขียนสร้างและแก้ไขข้อบกพร่องของคลาสแล้ว สามารถนำคลาสนี้ไปแทนในโปรแกรมเดิมได้โดยไม่ส่ง ผลกระทบกับการทำงานร่วมกันของแต่ละคลาส

การตรวจสอบการส่งกลับข้อมูลความผิดพลาดของซอฟต์แวร์ที่พัฒนาด้วยภาษา C/C++ เป็นสิ่งที่สำคัญมาก ภาษา C/C++ มีวิธีจัดการกับการส่งกลับข้อมูลความผิดพลาด คือ exception แต่วิธีการนี้ไม่ได้นำมาใช้ในการสร้าง VisBuilder เนื่องจากจะส่งผลต่อประสิทธิภาพในการทำงานของซอฟต์แวร์

VisBuilder ไม่ใช้ตัวแปลส่วนกลาง (globalvariable) เนื่องจากปัญหาหลายอย่างในการพัฒนาซอฟต์แวร์เกิดจากการใช้ตัวแปลชนิดนี้

4.3.1 การโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

การสร้าง VisBuilder ด้วยการพิจารณาแบบเลเยอร์และการใช้การออกแบบเชิงวัตถุช่วยให้ระบบง่ายต่อการใช้และการพัฒนา การโปรแกรมได้ใช้ภาษา C++ ซึ่งมีความสามารถที่จำเป็นของการโปรแกรมเชิงวัตถุ เช่น การ encapsulate การสร้างคลาสแบบ abstract การอนุพันธ์คลาส (inheritance) เป็นต้น

คลาสฐานของทุกองค์ประกอบย่อยจะเป็นแบบ abstract ซึ่งจะทำให้ได้ฟังก์ชันการทำงานพื้นฐานที่ไม่เปลี่ยนแปลงสามารถนำไปใช้ได้กับทุกระบบปฏิบัติการ ความสามารถในการอนุพันธ์คลาสสามารถดูได้จากการสร้างองค์ประกอบใหม่ซึ่งสามารถทำได้โดยการอนุพันธ์จากคลาสฐานนี้จึงเป็นไปได้ที่จะเพื่อความสามารถการประมวลผลที่ยังขาดเข้าไปในระบบหรือสามารถสร้างองค์ประกอบย่อยที่ครอบชุดพัฒนาอื่น ๆ ไว้ภายในได้การเพิ่มองค์ประกอบย่อยเข้าไปสามารถทำได้อย่างไม่จำกัดยิ่งกว่านั้น ความสามารถในการ encapsulate ช่วยให้เลเยอร์ที่อยู่สูงกว่าไม่ได้รับผลกระทบในกรณีที่มีการเปลี่ยนแปลงการทำงานภายในทำให้การปรับปรุงประสิทธิภาพของระบบสามารถทำได้ง่าย

4.3.2 การจัดการ error code

สิ่งหนึ่งที่สำคัญมากในการออกแบบซอฟต์แวร์คือ การตรวจสอบสถานะของค่าที่ส่งกลับ (return code) จากฟังก์ชันหรือสถานะการทำงานของส่วนใด ๆ ในต้นรหัส VisBuilder ไม่ใช้ exception ในการการจัดการสถานะของค่าที่ส่งกลับ เพราะการใช้ exception จะส่งผลต่อความเร็วในการประมวลผล การจัดการสถานะของค่าที่ส่งกลับด้วยตัวเองจะให้การทำงานที่รวดเร็วกว่า

ค่าที่ส่งกลับจากฟังก์ชันจะมีเพียงสองสถานะเท่านั้นเพื่อให้สามารถจัดการได้ง่าย และมีประสิทธิภาพสถานะที่ใช้ในที่นี้ คือ VISBD_SUCCESS สำหรับการทำงานของฟังก์ชันที่เสร็จสมบูรณ์ซึ่งมีค่าเท่ากับศูนย์และ VISBD_FAIL สำหรับการทำงานที่เกิดข้อผิดพลาดใด ๆ ซึ่งมีค่าเท่ากับลบหนึ่งทุกฟังก์ชันที่มีการส่งกลับค่าสถานะการทำงานจะต้องส่งค่าใดค่าหนึ่งจากสองค่านี้พร้อมด้วยข้อมูลเพิ่มเติมของสถานะการทำงาน

ข้อมูลเพิ่มเติมสำหรับอธิบายสถานะการทำงานจะใช้การจัดการด้วยแมโครซึ่งจะส่งสถานะข้อมูลเพิ่มเติมหรือคำอธิบายและค่าประจำตัวขององค์ประกอบกลับไปให้ระบบ ข้อมูลทั้งหมดจะถูกนำมาแสดงที่ส่วนสำหรับแสดงข้อผิดพลาดของส่วนต่อประสานกราฟิกกับผู้ใช้ที่อยู่ด้านล่างของหน้าต่างหลัก สิ่งเหล่านี้จะช่วยให้ผู้พัฒนาซอฟต์แวร์สามารถตรวจสอบหาความผิดพลาดที่เกิดขึ้นได้สามารถรู้ส่วนของต้นรหัสที่ทำให้เกิดความผิดพลาดนี้ทำให้การแก้ไขปรับปรุงทำได้ง่าย VisBuilder มีความสามารถในการจัดการกับความผิดพลาดได้โดยไม่ต้องปิดโปรแกรม การจัดการความผิดพลาดด้วยวิธีนี้ช่วยให้คาดคะเนความเร็วได้ต่างจากการใช้ exception ซึ่งจะขึ้นอยู่กับชุดพัฒนาที่ใช้คอมไพล์ทำให้ไม่สามารถคาดคะเนได้โดยตรง

4.3.3 การทำงานแบบระบบงานพร้อมกัน

หลักการที่สำคัญในการออกแบบ VisBuilder คือ การทำให้องค์ประกอบย่อยทำงานไปพร้อมกัน โดยการใช้ความสามารถของเธรด (thread) ซึ่งเป็นการแบ่งงานออกเป็นส่วนย่อยและให้ทำงานได้พร้อมกัน การใช้เธรดไม่ได้เป็นข้อกำหนดตายตัวของการทำงานของ VisBuilder อาจจะใช้การแบ่งงานออกด้วยการใช้โปรเซส (process) แทนได้โปรเซสนั้นจะแตกต่างจากเธรดตรงที่ว่าเธรดเป็นระบบงานย่อยที่ทำงานอยู่ในโปรเซส โปรเซสให้ประสิทธิภาพการทำงานที่ดีกว่า แต่วิธีการใช้ทรัพยากรร่วมกันระหว่างโปรเซสจะแตกต่างจากเธรด ซึ่งจะมีความซับซ้อนมากกว่าการเลือกใช้วิธีการใดไม่ว่าจะเป็นเธรดหรือโปรเซสหรือวิธีการอื่น ๆ ไม่ใช่สิ่งที่ต้องนำมาพิจารณามากนักเพราะสามารถแทนกันได้แต่สิ่งที่สำคัญในที่นี้คือ VisBuilder มีความสามารถแบบระบบงานพร้อมกัน

สำหรับ VisBuilder ในขณะนี้ได้เลือกใช้เธรดด้วยเหตุผลที่ว่า การใช้วิธีการแยกงานเป็นโปรเซส นั้นเกินความจำเป็น เนื่องจากสามารถสร้างองค์ประกอบย่อยที่ส่งรับข้อมูลระหว่าง

โปรเซสแทนได้การใช้งานโดยการเปิด VisBuilder หลาย ๆ ตัวให้สื่อสารกันผ่านองค์ประกอบย่อยดังกล่าว ซึ่งก็ไม่ต่างจากการแบ่งงานเป็นโปรเซส

การพัฒนาซอฟต์แวร์สำหรับระบบการมองเห็นโดยการใช้ประโยชน์จากคุณสมบัติของการทำงานแบบระบบงานพร้อมกันจะช่วยให้ได้ประสิทธิภาพการทำงานของระบบที่ดีอีกทั้งยังช่วยให้การออกแบบระบบซอฟต์แวร์ทำได้ง่าย แต่ละเธรดจะทำงานอย่างอิสระภายใต้เงื่อนไขการทำงานของระบบการมองเห็นที่ได้รับมอบหมายให้ทำ ตัวอย่างเช่น องค์ประกอบย่อยสำหรับการรับภาพจากอุปกรณ์รับภาพสามารถรับภาพและรอรับภาพถัดไป เมื่อพิจารณาจะเห็นว่าการทำงานมีเพียงการรับภาพและรอภาพถัดไปเท่านั้น การพิจารณาในลักษณะนี้สามารถใช้กับการออกแบบองค์ประกอบย่อยชนิดอื่น ๆ ได้เช่นเดียวกัน เมื่อมองในแง่ของความเร็ว ในการทำงานความเร็วของเธรดจะขึ้นอยู่กับกรรมวิธีทางระบบการมองเห็นที่เธรดได้รับทำให้สามารถประเมินความเร็วของแต่ละองค์ประกอบย่อยได้ซึ่งช่วยในการปรับปรุงประสิทธิภาพการทำงาน

4.3.4 การไม่ใช้ตัวแปลส่วนกลาง (global variable)

VisBuilder ได้รับการออกแบบให้ไม่มีการใช้ตัวแปลส่วนกลางในทุกส่วนของโปรแกรม ปัญหาหลายอย่างในการพัฒนาซอฟต์แวร์เกิดจากการใช้ตัวแปลชนิดนี้ คำว่าส่วนกลาง (global) หมายถึงตัวแปรที่บรรจุข้อมูลไว้นั้นจะถูกประกาศ (declare) เอาไว้ภายนอกทุก ๆ ฟังก์ชัน ซึ่งทำให้ฟังก์ชันทุก ๆ ฟังก์ชันสามารถที่จะเข้าถึงตัวแปรเหล่านี้ได้ฟังก์ชันเหล่านี้จะดำเนินการต่อข้อมูลด้วยวิธีการต่าง ๆ ไม่ว่าจะเป็นการอ่านข้อมูล การวิเคราะห์ข้อมูล การปรับข้อมูล การจัดข้อมูลใหม่การแสดงข้อมูล การเขียนข้อมูลกลับไปยังดิสก์เป็นต้น

อย่างไรก็ตามอาจมีการใช้ตัวแปลส่วนกลางในชุดพัฒนาซอฟต์แวร์ที่นำมาใช้เช่นจากการใช้ไฟล์ส่วนหัว (header file) ของระบบที่ใช้พัฒนาซอฟต์แวร์หรืออาจมาจากการนำชุดพัฒนาอื่น ๆ มาใช้การไม่มีตัวแปลส่วนกลางช่วยให้ไม่ต้องกังวลปัญหาเนื่องการทำงานขององค์ประกอบย่อยแต่ละตัวมีการซ้อนทับกันในการเข้าถึงตัวแปล องค์ประกอบย่อยแต่ละตัวจะมีตัวแปลเป็นของตัวเองสำหรับ การบรรจุข้อมูล การเก็บสถานะการทำงาน ซึ่งตัวแปลดังกล่าวจะถูกซ่อนไว้ภายใน ป้องการการเข้าถึงได้โดยตรงจากองค์ประกอบย่อยอื่น ๆ

4.3.5 การจัดเก็บไฟล์

การจัดเก็บอัลกอริทึมของระบบการมองเห็นของคอมพิวเตอร์ที่ผู้ใช้ออกแบบจะใช้การบันทึกในรูปแบบ XML การจัดการกับไฟล์ลักษณะนี้จะใช้หลักการ XML Parser เป็น API (application programming interface) ที่ช่วยในการอ่านหรือเขียนข้อมูลโดย API ที่นิยมกันมากคือ DOM (Marini, 2002) และ SAX (Brownell, 2004) ทั้งสองวิธีจะใช้การเข้าถึงข้อมูลที่แตกต่างกัน

การเข้าถึงข้อมูลแบบ DOM จะมองเอกสาร XML ในลักษณะของโครงสร้างต้นไม้ โดยมีหลักการในการอ่านเอกสาร XML มาวางในหน่วยความจำในลักษณะต้นไม้ซึ่งประกอบด้วย element และ attribute ต่าง ๆ การเข้าถึงข้อมูลจะใช้การเดินไปตามกิ่งก้านของต้นไม้แบบต่อเนื่องไปเรื่อย ๆ หรือจะใช้การอ้างอิงกิ่งก้านแบบเฉพาะเจาะจงลงไป หรือจะเข้าถึงแบบสุ่ม (random access) ก็ได้ข้อจำกัดของ DOM คือต้องการหน่วยความจำที่เพียงพอกับปริมาณข้อมูลที่ใช้ในการประมวลผลเนื่องจาก DOM จะอ่านข้อมูลทั้งหมดมาเก็บไว้ในหน่วยความจำในครั้งเดียว แต่ก็มีข้อดี คือ การเขียนโปรแกรมสามารถทำได้ง่ายกว่า

การเข้าถึงเอกสาร XML แบบ SAX จะจัดการกับข้อมูลด้วยแนวทาง event driven การทำงานของ SAX จะไม่อ่านข้อมูลทั้งหมดมาไว้ในหน่วยความจำแต่จะอ่านข้อมูลจากไฟล์ตั้งแต่เริ่มต้นไล่ลงไปเรื่อย ๆ แล้วสร้าง event ต่าง ๆ ออกมา เช่น การเปิด element การปิด element การพบ attribute เป็นต้น ดังนั้นผู้เขียนโปรแกรมจึงมีหน้าที่ดักจับ event เหล่านั้นเพื่อจัดการกับข้อมูล

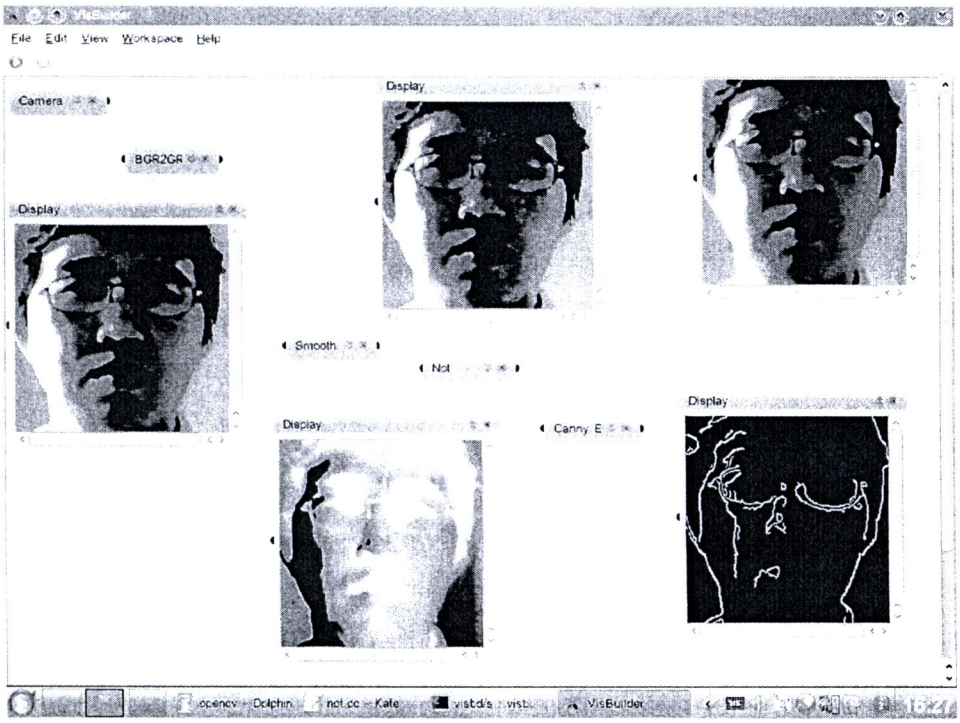
การจัดเก็บไฟล์ของ VisBuilder จะใช้ทั้งสองวิธีตามความเหมาะสมในการใช้งาน กล่าวคือ ในการบันทึกไฟล์จะใช้ API แบบ DOM เพราะบันทึกไฟล์สามารถทำให้เสร็จในครั้งเดียว ได้ส่วนการอ่านไฟล์จะใช้ SAX เนื่องจากการสร้างระบบการมองเห็นจากไฟล์ขึ้นใหม่จะทำได้ง่ายในแง่ของการโปรแกรม เช่น เมื่อเกิด event จากการเปิด element ที่ชื่อว่า COMPONENT ก็จะทำการสร้างองค์ประกอบย่อยขึ้นและเพิ่มเข้าไปในระบบ เมื่ออ่านถึง element ชื่อ LINK ก็จะสร้างการเชื่อมต่อระหว่างองค์ประกอบย่อยตามที่ระบุไว้เป็นต้น

4.3.6 การจองหน่วยความจำ

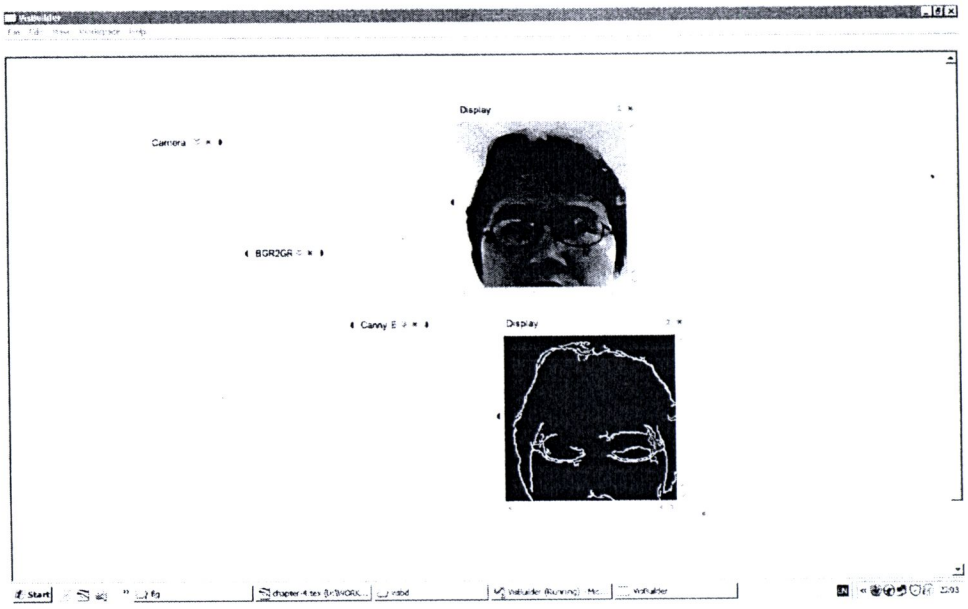
VisBuilder ได้รับการออกแบบให้ทำงานกับข้อมูลภาพอย่างต่อเนื่อง ในหลายส่วนของซอฟต์แวร์นี้มีการจองหน่วยความจำไว้ล่วงหน้าหรือการสร้างบัฟเฟอร์เพื่อบรรจุข้อมูลสำหรับการประมวลผล หน่วยความจำหรือบัฟเฟอร์ดังกล่าวจะถูกใช้งานซ้ำแล้วซ้ำอีกตลอดเวลาที่มีการเรียกใช้ฟังก์ชัน ที่มีความเกี่ยวข้องกับบัฟเฟอร์นี้การจองหน่วยความจำขนาดใหญ่สำหรับการใช้งานในช่วงเวลาสั้น ๆ ในระหว่างการทำงานแล้วคืนกลับให้ระบบ การทำเช่นนี้หลาย ๆ ครั้งจะส่งผลกระทบต่อความเร็ว นการทำงานโดยรวมของซอฟต์แวร์ดังนั้นการจองหน่วยความจำไว้ล่วงหน้าเพื่อการประมวลผล แล้วคืนให้ระบบเมื่อองค์ประกอบนั้น ๆ ถูกลบออกจากระบบจะช่วยเพิ่มประสิทธิภาพการทำงานให้ดียิ่งขึ้น

4.4 สรุป

ซอฟต์แวร์ VisBuilder ที่สร้างขึ้นเป็นไปตามความต้องการของระบบตามที่ได้กล่าวไว้ในบทที่ 3 ซึ่งรองรับการทำงานแบบระบบงานพร้อมกันโดยการใช้เธรดพัฒนาในลักษณะขององค์ประกอบย่อย สามารถขยายความสามารถของระบบโดยการเพิ่มองค์ประกอบย่อยเข้าไป มีความยืดหยุ่นในการใช้งานรองรับการทำงานบนระบบปฏิบัติการต่าง ๆ ดังแสดงในรูปที่ 4.11 และรูปที่ 4.12 คลาสในเฟรมเวิร์คสำหรับการรับภาพ การประมวลผลภาพ และการแสดงผลภาพสร้างจากการอนุพันธ์จากคลาสฐานที่ชื่อว่า Component ซึ่งมีความสามารถในการทำงานแบบระบบงานพร้อมกันทำให้การรับภาพ การประมวลผลภาพต่าง ๆ และการแสดงผลทำงานได้พร้อมกัน ผู้พัฒนาระบบการมองเห็นของคอมพิวเตอร์สามารถเพิ่มคการประมวลผลหรือการเชื่อมต่อกับอุปกรณ์ต่าง ๆ ได้โดยใช้ต้นแบบที่ได้เตรียมไว้ให้



รูปที่ 4.11 การทำงานของ VisBuilder บนระบบปฏิบัติการลินุกซ์



รูปที่ 4.12 การทำงานของ VisBuilder บนระบบปฏิบัติการวินโดวส์

ความสามารถในการทำงานในลักษณะระบบงานพร้อมกันช่วยลดความล่าช้าในการประมวลผลภาพและลดความซับซ้อนของระบบความล่าช้าเป็นผลมาจากความจำเป็นในการรอผลการประมวลผลภาพและการแสดงภาพ ก่อนที่จะรับภาพใหม่เข้ามา องค์ประกอบย่อยการประมวลผลบางชนิดอาจทำงานได้เร็วกว่าองค์ประกอบย่อยอื่น ๆ ดังนั้นการทำงานได้พร้อมกันขององค์ประกอบย่อยจะช่วยลดความล่าช้าในการทำงานโดยรวมลงไปได้มาก

VisBuilder พัฒนาในลักษณะแยกการทำงานออกเป็นส่วนย่อยหรือองค์ประกอบย่อย แต่ละส่วนมีความอิสระจากกัน ซึ่งช่วยให้การจัดการองค์ประกอบย่อยและการพัฒนาร่วมกับชุดพัฒนาอื่น ๆ โดยการเรียกใช้ภายในองค์ประกอบย่อยจะทำได้ง่าย การเปลี่ยนแปลงแก้ไขต้นรหัสใด ๆ ที่เกิดขึ้นกับองค์ประกอบย่อยจะไม่ส่งผลกระทบกับส่วนอื่น ๆ ของซอฟท์แวร์ องค์ประกอบย่อยดังกล่าวสามารถนำไปใช้ร่วมกับ VisBuilder เดิมโดยไม่ต้องคอมไพล์ใหม่ในกรณีขององค์ประกอบย่อยที่สร้างขึ้นใหม่สามารถนำมาใช้ได้ทันทีซึ่งมีการทำงานคล้ายกับการต่อปลั๊กของอุปกรณ์ใหม่เข้ากับระบบเดิมที่มีอยู่

การออกแบบ VisBuilder ใช้การแบ่งออกเป็นเลเยอร์ซึ่งมี 5 เลเยอร์โดยเลเยอร์ที่อยู่ด้านบนสร้างจากเลเยอร์ที่อยู่ด้านล่างและไม่มีการใช้ตัวแปลส่วนกลางเพื่อให้เหมาะสมกับการพัฒนาแบบแยกเป็นส่วนย่อยขององค์ประกอบย่อยต่าง ๆ สำหรับรับภาพประมวลผลภาพ และแสดงผลมีการทำงานที่อิสระจากกันและมีต้นรหัสที่ง่ายต่อการทำความเข้าใจ

การคอมไพล์ต้นรหัสขององค์ประกอบย่อยต่าง ๆ สามารถเลือกคอมไพล์เฉพาะส่วนที่เหมาะสมกับระบบปฏิบัติการนั้น ๆ หรือให้สอดคล้องกับชุดพัฒนาที่เกี่ยวข้องได้โดยทั่วไปการคอมไพล์สามารถทำได้สองวิธี คือ การใช้ Makefile หรือการใช้ Project file ซึ่ง VisBuilder รองรับทั้งสองวิธี

VisBuilde มีความสามารถในการขยายระบบรองรับการพัฒนาบนสภาวะแวดล้อมของการพัฒนาได้หลายแบบทั้งการพัฒนาบนระบบปฏิบัติการวินโดวส์และระบบปฏิบัติการลินุกซ์และรองรับการสร้างองค์ประกอบย่อยการรับภาพ การประมวลผลภาพ และการแสดงผลภาพเพิ่มเติมเข้าไปในระบบ

การสื่อสารระหว่างองค์ประกอบย่อยใช้หลักการไหลของข้อมูลซึ่งได้สร้างไว้ภายในผู้ใช้ไม่มีความจำเป็นต้องเขียนโปรแกรมสำหรับการส่งข้อมูลระหว่างองค์ประกอบด้วยตนเองการสื่อสารดังกล่าวจะเกิดขึ้น ภายในโดยไหลไปตามจุดเชื่อมต่อระหว่างองค์ประกอบย่อยการเพิ่มชนิดข้อมูลใหม่สามารถทำได้ผ่านทาง การอนุพันธ์จากคลาสที่เตรียมไว้ให้