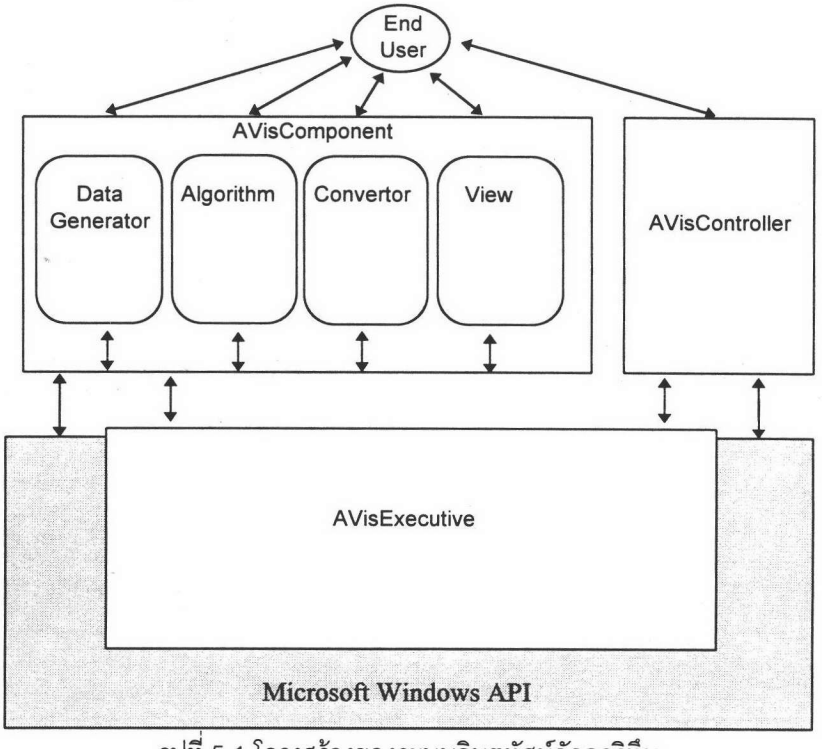
5. หน่วยบริหารการจินตทัศน์อัลกอริทึม

ในบทนี้จะกล่าวถึงโครงสร้างและการทำงานของหน่วยบริหารการจินตทัศน์และบริการต่างๆทั้งหมดของหน่วยบริหารการจินตทัศน์ซึ่งเป็นส่วนประกอบส่วนหนึ่งของระบบ AVis ที่ทำหน้าที่ควบคุมและบริหารให้เกิดการจินตทัศน์ขึ้นตามบทการจินตทัศน์ที่ผู้พัฒนาบทการจินตทัศน์ออกแบบไว้ หน้าที่อีกประการหนึ่งของหน่วยบริหารการจินตทัศน์ก็คือการให้บริการแก่ส่วนประกอบอื่นๆของระบบเช่นองค์ประกอบการจินตทัศน์และโปรแกรมควบคุมการจินตทัศน์ในรูปของการเรียกใช้ฟังก์ชัน (function call) เพื่อทำให้การพัฒนาส่วนประกอบเหล่านี้ทำได้ง่ายขึ้น

เนื้อหาในบทนี้กล่าวถึงเฉพาะโครงสร้างและขั้นตอนการทำงานต่างๆที่นำมาใช้ในการพัฒนาหน่วยบริหารการจินตทัศน์ แต่จะไม่กล่าวถึงการเปรียบเทียบประสิทธิภาพและข้อดีข้อเสียโครงสร้างและขั้นตอนการทำงานเหล่านี้กับโครงสร้างและขั้นตอนการทำงานแบบอื่นๆ ผู้ที่สนใจและต้องการศึกษาเปรียบเทียบประสิทธิภาพและข้อดีข้อเสียของโครงสร้างและวิธีการที่เลือกใช้ในหน่วยบริหารการจินตทัศน์กับโครงสร้างและขั้นตอนการทำงานอื่นๆ สามารถศึกษาเพิ่มเติมได้จากบทที่ 6

5.1 โครงสร้างของระบบจินตทัศน์อัลกอริทึม

จากที่กล่าวมาแล้วว่าระบบ AVis ประกอบไปด้วยส่วนประกอบหลักสามส่วนได้แก่ (1) หน่วยบริหารการจินตทัศน์, (2) โปรแกรมควบคุมการจินตทัศน์ และ (3) องค์ประกอบการจินตทัศน์



รูปที่ 5.1 โครงสร้างของระบบจินตทัศน์อัลกอริทึม

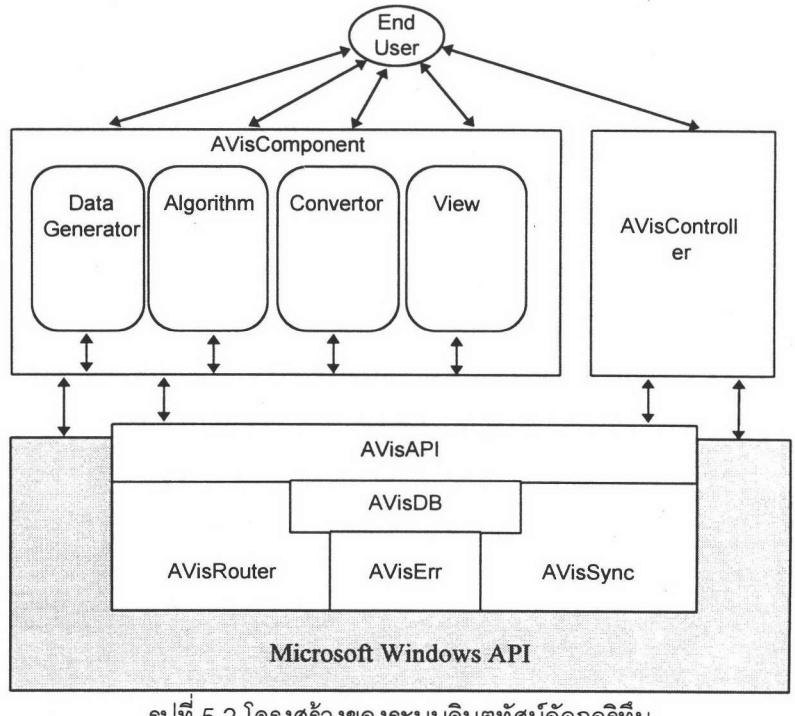
ส่วนประกอบทั้งสามของระบบ AVis จะมีการติดต่อประสานงานกันด้วยวิธีการสองวิธีคือ (1) การส่งข้อความคำสั่งระหว่างส่วนประกอบต่างๆ และ (2) การขอใช้บริการของหน่วยบริหารการจินตทัศน์ จากรูปที่ 5.1 จะพบว่าหน่วยบริหารการจินตทัศน์จะไม่ได้ทำการติดต่อประสานงานกับผู้ใช้ปลายทาง (End User) โดย

ตรง แต่ผู้ใช้ก็สามารถควบคุมและปรับแต่งการทำงานและการแสดงผลของการจินตทัศน์ผ่านองค์ประกอบการจินตทัศน์และโปรแกรมควบคุมการจินตทัศน์ได้ โดยส่วนประกอบเหล่านี้จะส่งผ่านคำสั่งต่างๆ ของผู้ใช้ไปให้หน่วยบริหารการจินตทัศน์ผ่านการเรียกใช้ฟังก์ชันต่างๆของหน่วยบริหารการจินตทัศน์ จากนั้นหน่วยบริหารการจินตทัศน์ก็จะใช้การส่งข้อความคำสั่งทำการควบคุมการทำงานของการจินตทัศน์ต่อไป

5.2 *หน่วยบริหารการจินตทัศน์ (AVisExecutive)*

หน่วยบริหารการจินตทัศน์ถือได้ว่าเป็นส่วนที่สำคัญมากและเป็นแกนหลักในการทำงานของระบบจินตทัศน์ เนื่องจากเป็นส่วนที่บริหารการทำงาน ข้อมูล และสภาพแวดล้อมต่างๆระหว่างการจินตทัศน์ หน่วยบริหารการจินตทัศน์ประกอบด้วยส่วนย่อยห้าส่วนที่คอยให้บริการต่างๆแก่ส่วนประกอบอื่นๆของระบบ (ดูรูปที่ 5.2) ได้แก่

1. *AVisSync* ให้บริการและควบคุมการประสานจังหวะการงานขององค์ประกอบอัลกอริทึมต่างๆในระบบ
2. *AVisRouter* ให้การส่งผ่านข้อความคำสั่งต่างๆระหว่างส่วนประกอบในระบบ
3. *AVisDB* ให้บริการการสอบถาม/ตรวจสอบข้อมูลขององค์ประกอบการจินตทัศน์ และ โปรแกรมควบคุมการจินตทัศน์
4. *AVisErr* ตรวจสอบและจัดการความผิดพลาดที่อาจเกิดขึ้นในระบบ
5. *AVisAPI* ตรวจสอบสิทธิ์และความถูกต้องของการขอใช้บริการ ก่อนที่จะได้รับบริการจากส่วนอื่นของหน่วยบริหารการจินตทัศน์



รูปที่ 5.2 โครงสร้างของระบบจินตทัศน์อัลกอริทึม

จากที่กล่าวมาแล้วว่าหน่วยบริหารการจินตทัศน์จะคอยให้บริการต่างๆต่อองค์ประกอบการจินตทัศน์และโปรแกรมควบคุมการจินตทัศน์ ซึ่งทั้งหมดนี้จะถูกพัฒนาเป็นโปรแกรมบนระบบวินโดว์ ดังนั้นจึงได้มีการพัฒนาหน่วยบริหารการจินตทัศน์ของระบบ AVis ให้อยู่ในรูปของคลังคำสั่งร่วมแบบพลวัตของระบบวินโดว์

ที่มีชื่อว่า AVISDLL.DLL เพื่อทำให้นำหน่วยบริหารการจินตทัศน์ไปใช้งานทำได้สะดวกและง่ายขึ้น นอกจากนี้ การพัฒนาหน่วยบริหารการจินตทัศน์ให้อยู่ในรูปของคลังคำสั่งร่วมแบบพลวัตซึ่งเป็นรูปแบบมาตรฐานของระบบวินโดว์ทำให้การเรียกใช้บริการของหน่วยบริหารการจินตทัศน์สามารถทำได้จากเกือบทุกภาษาที่ใช้ในการพัฒนาโปรแกรมบนระบบไมโครซอฟต์วินโดว์อีกด้วย

5.3 AVisAPI

AVisAPI (Algorithm Visualization Application Programming Interface) เป็นส่วนประกอบเพียงส่วนเดียวของหน่วยบริหารการจินตทัศน์ที่ส่วนประกอบอื่นๆของระบบ จะเรียกเพื่อขอใช้บริการของหน่วยบริหารการจินตทัศน์ได้ AVisAPI มีหน้าที่หลักอยู่สี่ประการคือ

1. ตรวจสอบสิทธิ์ของผู้เรียกใช้ฟังก์ชัน ทั้งนี้บางฟังก์ชันจะถูกเรียกใช้ได้จากองค์ประกอบบางประเภทเท่านั้น ตัวอย่างเช่น โปรแกรมควบคุมการจินตทัศน์เท่านั้นที่มีสิทธิ์กำหนดลักษณะการทำงานของการจินตทัศน์เป็นต้น
2. ตรวจสอบความถูกต้องของข้อมูลซึ่งผู้เรียกใช้ส่งมาประกอบการขอบริการ (parameter validation) ถ้าข้อมูลที่ส่งมาถูกต้องก็จะเรียกโปรแกรมย่อยของ AVisDB, AVisRouter หรือ AVisSync ที่เหมาะสมมาทำงานต่อไป แต่หากพบข้อผิดผลาดจะให้ค่าความผิดผลาดกลับไปยังผู้ขอใช้บริการโดยไม่เรียกโปรแกรมย่อยที่เกี่ยวข้องนั้นๆมาทำงาน
3. เป็นตัวกั้นระหว่างฟังก์ชันซึ่งผู้ขอใช้บริการมองเห็นจากฟังก์ชันและข้อมูลภายในของระบบ (encapsulation) การทำเช่นนี้มีประโยชน์คือ หากเมื่อใดเกิดการเปลี่ยนแปลงโครงสร้างข้อมูลและการทำงานภายในของระบบ ก็จะไม่มีผลต่อผู้ขอใช้บริการ
4. ให้บริการฟังก์ชันบางอย่างซึ่งเป็นฟังก์ชันที่ไม่เกียวกับระบบจินตทัศน์อัลกอริทึมโดยตรง แต่จะทำให้การพัฒนาองค์ประกอบการจินตทัศน์ทำได้ง่ายขึ้น เช่นการบริการการส่งผ่านข้อมูลที่เป็นแถวลำดับ เพื่อหลีกเลี่ยงการส่งข้อมูลทีละตัว การตั้งข้อความประจำวินโดว์ขององค์ประกอบที่ทำงานอยู่ เพื่อให้ผู้ใช้รับรู้ เป็นต้น

ฟังก์ชันต่างๆที่หน่วยบริหารการจินตทัศน์มีไว้บริการผ่านทาง AVisAPI นั้น สามารถแบ่งออกเป็นกลุ่มย่อยๆตามหน้าที่การทำงานได้ดังนี้

5.3.1 **บริการการจัดการองค์ประกอบที่ทำงานในระบบ**

ชื่อฟังก์ชัน	หน้าที่
ฟังก์ชันที่ใช้เปิดและปิดฐานข้อมูลที่จัดเก็บโดย AVisDB	
AVisOpenSession	จัดเตรียมระบบก่อนเริ่มสั่งการให้องค์ประกอบต่างๆในระบบเริ่มทำงาน
AVisCloseSession	บอกเลิกการจินตทัศน์ขององค์ประกอบต่างๆในระบบ
ฟังก์ชันที่ใช้เพิ่มข้อมูล	
AVisRegisterComponent	ลงทะเบียนองค์ประกอบเข้าสู่ระบบ

ตารางที่ 5.1 ฟังก์ชันซึ่งใช้จัดการองค์ประกอบที่ทำงานในระบบ

ชื่อฟังก์ชัน	หน้าที่
ฟังก์ชันที่ใช้ลบข้อมูล	
AVisUnregisterComponent	ถอนทะเบียนองค์ประกอบที่กำหนดออกจากระบบ
AVisUnregisterController	ยกเลิกหน้าที่โปรแกรมควบคุมการจินตทัศน์
ฟังก์ชันที่ใช้สืบค้นข้อมูล	
AVisEnumComponent	แจงองค์ประกอบทุกตัวที่มีในระบบ
AVisIsValidID	ตรวจสอบว่าองค์ประกอบว่าอยู่ในระบบหรือไม่
AVisGetComponentWindow	ให้ค่าหมายเลขประจำวินโดว์ขององค์ประกอบ
AVisGetComponentID	ให้ค่าหมายเลขประจำองค์ประกอบจากหมายเลขวินโดว์ (HWND) และหมายเลขโปรแกรม (HINSTANCE)
AVisGetComponentType	ให้บริการสอบถามประเภทขององค์ประกอบ
AVisGetSyncCount	ให้ค่าจำนวนครั้งของขั้นตอนการทำงานที่ถูกประสานจังหวะ

ตารางที่ 5.1 (ต่อ) ฟังก์ชันซึ่งใช้จัดการองค์ประกอบที่ทำงานในระบบ

การให้บริการของ AVisDB นั้นนอกจากจะให้บริการแก่ส่วนประกอบอื่นๆเช่นองค์ประกอบการจินตทัศน์แล้ว AVisDB ยังคอยให้บริการข้อมูลแก่ส่วนประกอบย่อยของหน่วยบริหารการจินตทัศน์อื่นๆเช่น AVisRouter , AVisSync และ AVisErr อีกด้วย

5.3.2 บริการการจัดการการรับส่งข้อความคำสั่ง

ชื่อฟังก์ชัน	หน้าที่
ฟังก์ชันที่ใช้สร้างและลบ และสืบค้นข้อมูลการรับส่งข้อความคำสั่ง	
AVisBind	เชื่อมความสัมพันธ์ในการรับส่งข้อมูลระหว่างองค์ประกอบผู้ส่งข้อมูล และองค์ประกอบผู้รับข้อมูล
AVisUnbind	ลบความสัมพันธ์ในการรับส่งข้อมูลระหว่างองค์ประกอบผู้ส่งข้อมูล และองค์ประกอบผู้รับข้อมูล ที่เคยถูกเชื่อมไว้ด้วยฟังก์ชัน AVisBind
AVisEnumLink	แจงองค์ประกอบทุกตัวที่เชื่อมโยงกับองค์ประกอบที่กำหนด
ฟังก์ชันที่ใช้ส่งข้อความคำสั่งไปยังองค์ประกอบอื่น	
AVisInputRequest	ส่งข้อความคำสั่งไปยังองค์ประกอบตัวแรกที่เป็นตัวส่งข้อมูล ขององค์ประกอบที่กำหนด
AVisInputRequestEx	ส่งข้อความคำสั่งจากองค์ประกอบหนึ่งไปยังอีกองค์ประกอบหนึ่งที่เป็นตัวส่งข้อมูล

ตารางที่ 5.2 ฟังก์ชันซึ่งให้บริการการจัดการการรับส่งข้อความคำสั่งของ AVisRouter

ชื่อฟังก์ชัน	หน้าที่
AVisOutputNotifyEx	ส่งข้อความคำสั่งจากองค์ประกอบหนึ่งไปยังอีกองค์ประกอบหนึ่งที่เป็นตัวรับข้อมูล
ฟังก์ชันที่ใช้เพื่อสร้างกลไกการส่งข้อความแบบขนาน AVisReplyMessage	แจ้งเตือนผู้ส่งข้อความว่าองค์ประกอบที่ได้รับคำสั่งทำงานเสร็จแล้ว
ฟังก์ชันที่ใช้เพื่อป้องกันการเกิดข้อความคำสั่งซ้อน AVisSuspendMessage AVisResumeMessage	แจ้งเตือนAVisว่าองค์ประกอบไม่พร้อมที่จะรับข้อความคำสั่งใดๆ ยกเลิกสภาวะไม่รับข้อความคำสั่งที่เกิดจากการเรียกฟังก์ชัน AVisSuspendMessage

ตารางที่ 5.2 (ต่อ) ฟังก์ชันซึ่งให้บริการการจัดการการรับส่งข้อความคำสั่งของ AVisRouter

5.3.3 บริการการประสานจังหวะการทำงานขององค์ประกอบอัลกอริทึม

ชื่อฟังก์ชัน	หน้าที่
ฟังก์ชันประสานจังหวะการทำงาน AVisOpenSync	แจ้งเตือนส่วนประสานจังหวะให้เตรียมโครงสร้างข้อมูลสำหรับเก็บข้อมูลการประสานจังหวะขององค์ประกอบที่เรียกใช้ฟังก์ชันนี้
AVisCloseSync	แจ้งเตือนส่วนประสานจังหวะให้ลบโครงสร้างข้อมูลที่เตรียมไว้ด้วยฟังก์ชัน AVisOpenSync เนื่องจากองค์ประกอบไม่ต้องการประสานจังหวะอีกต่อไป
AVisSync	แจ้งเตือนให้ส่วนประสานจังหวะทราบว่าองค์ประกอบอัลกอริทึมได้ทำงานมาแล้ว เพื่อให้ส่วนประสานจังหวะอนุญาติให้องค์ประกอบอัลกอริทึมอื่นทำงานได้
ฟังก์ชันที่ใช้สำหรับตั้งและอ่านค่าสถานะการทำงานของ AVisSync AVisGetSyncStatus	อ่านค่าสถานะส่วนประสานจังหวะ
AVisGetSyncMode	อานค่าสภาวะการทำงานของส่วนประสานจังหวะ
AVisSetSyncMode	ตั้งสภาวะการทำงานของส่วนประสานจังหวะให้อยู่ในสภาวะที่ต้องการ
AVisStepNextSync	สั่งเริ่มการทำงานรอบต่อไป หลังจากหยุดรอการทำงานทุกๆ รอบเมื่อทำงานในสภาวะทำทีละคำสั่ง
AVisEnumSync	แจงองค์ประกอบทุกๆตัวที่ทำงานแบบประสานจังหวะกันในระบบ

ตารางที่ 5.3 ฟังก์ชันซึ่งให้บริการการประสานจังหวะการทำงานของ AVisSync

5.3.4 บริการพิเศษอื่น ๆแก่องค์ประกอบต่าง ๆที่ทำงานในระบบ

ชื่อฟังก์ชัน	หน้าที่
AVisVBCreateArrayParam	การสร้างหมายเลขค่าขนาด long เพื่อแทนอาเรย์ในภาษาวิชวลเบสิก
AVisVBGetArrayParam	ทำสำเนาข้อมูลในแถวลำดับ (array) ซึ่งแทนด้วยค่าที่ได้จากฟังก์ชัน AVisVBCreateArrayParam มาเก็บไว้ในแถวลำดับของวิชวลเบสิก
AVisVBGetMainWindow	ให้ค่าหมายเลขกำกับวินโดว์หลักของโปรแกรมที่พัฒนาด้วยวิชวลเบสิก
AVisVBSetMainWindowCaption	ตั้งข้อความหัวเรื่อง (caption) ของโปรแกรมที่พัฒนาจากวิชวลเบสิก
AVisLoadComponent	เรียกองค์ประกอบการจินตทัศน์มาทำงาน
AVisUnloadComponent	ยุติการทำงานขององค์ประกอบการจินตทัศน์
AVisDispatchEvent	ตอบรับข้อความคำสั่งของ AVis และเรียกฟังก์ชันของผู้ใช้ที่ตอบสนองต่อข้อความคำสั่ง

ตารางที่ 5.4 ฟังก์ชันซึ่งให้บริการพิเศษอื่นๆแก่องค์ประกอบต่างๆที่ทำงานในระบบ

5.4 AVisDB

AVisDB (Algorithm Visualization Data Base) เป็นส่วนประกอบของหน่วยบริหารการจินตทัศน์ที่จัดเก็บข้อมูลทั้งหมดที่ต้องร่วมกันใช้ระหว่างส่วนประกอบย่อยต่างๆ นอกจากเป็นที่จัดเก็บข้อมูลแล้ว AVisDB จะให้บริการการสืบค้น แก้ไข เพิ่ม และลบข้อมูลเหล่านี้แก่ส่วนประกอบส่วนอื่นด้วย การแยกส่วนข้อมูลออกเป็นหน่วยย่อยต่างหากจากส่วนประกอบอื่นๆ มีข้อดีคือเราสามารถแยกการพัฒนาวิธีการจัดเก็บข้อมูลออกจากการพัฒนาส่วนประกอบอื่นๆ ทำให้สามารถปรับปรุงแก้ไขโครงสร้างข้อมูลให้มีประสิทธิภาพมากขึ้นโดยไม่กระทบกับส่วนประกอบอื่นในระบบ

ข้อมูลที่เก็บไว้ใน AVisDB ประกอบด้วยรายละเอียดต่างๆของตัวองค์ประกอบที่ถูกนำเข้าสู่ระบบ อันได้แก่

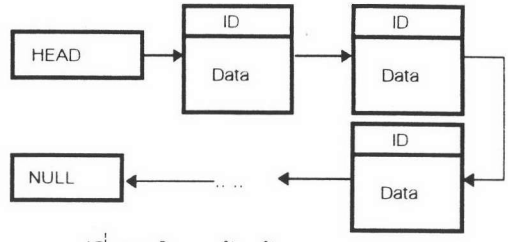
1. หมายเลขประจำองค์ประกอบ (ID) ซึ่งเป็นค่าเฉพาะตัวองค์ประกอบที่ไม่ซ้ำกัน องค์ประกอบที่ขอใช้บริการใดๆผ่านทาง AVisAPI ต้องแสดงค่า ID ของตัวเองเสมอ
2. หมายเลขประจำวินโดว์ขององค์ประกอบ (Window Handle) ซึ่งเป็นหมายเลขที่ส่วนประกอบอื่นๆของ AVisExec ต้องใช้ เมื่อต้องการขอใช้บริการของ Microsoft Windows API
3. ค่าสถานะการทำงานขององค์ประกอบ ค่านี้ถูกเปลี่ยนแปลงโดยส่วน AVisRouter AVisSync และ AVisErr

เนื่องจากโครงสร้างที่ใช้เก็บข้อมูลของ AVisDB สามารถถูกเพิ่ม ลบ และแก้ไขได้ในขณะทำงาน การจัดเก็บข้อมูลดังกล่าวจะอยู่รูปแบบของโครงสร้างข้อมูลแบบรายการโยง (linked list) ดังแสดงในรูปที่ 5.3 ข้อมูลประจำองค์ประกอบต่างๆจะถูกเก็บอยู่ในรายการโยง การค้นหาข้อมูลที่ต้องการจะใช้ ID เป็นตัวค้นหา*

การเพิ่มและลบข้อมูลใน AVisDB นั้นเกิดขึ้นจากการลงและการถอนทะเบียนของตัวองค์ประกอบที่ถูกนำเข้าสู่หรือนำออกจากระบบ โดยใช้ฟังก์ชัน AvisRegisterComponent และ AVisUnregisterComponent ของ AVisAPI ตามลำดับ

---

* เพื่อการเข้าถึง node ต่างๆในรายการโยงได้ทันทีเมื่อต้องการอ้างอิงข้อมูล ค่าของ ID ประจำ node นั้นก็คือ ที่อยู่เริ่มต้นของหน่วยความจำของ node นั้น



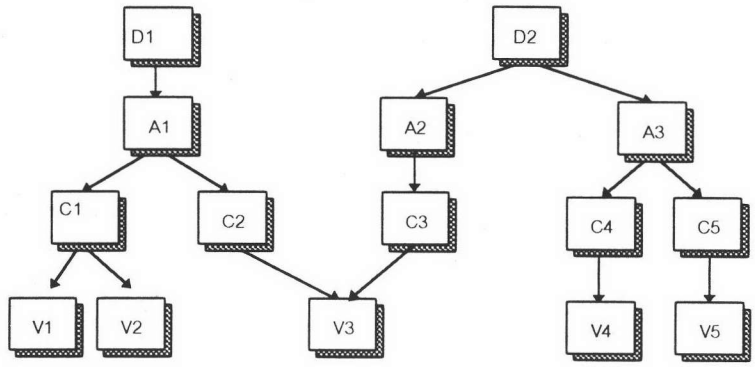
รูปที่ 5.3 โครงสร้างข้อมูลของ AVisDB

5.5 AVisRouter

องค์ประกอบต่างๆที่ถูกนำเข้าสู่ระบบ เพื่อประกอบกันเป็นบทการจินตทัศน์นั้น จะมีการรับส่งข้อความคำสั่งและข้อมูลซึ่งกันและกัน เพื่อแยกองค์ประกอบให้เป็นอิสระจากกัน องค์ประกอบเหล่านี้จะไม่ส่งข้อมูลถึงกันและกันโดยตรง แต่จะใช้บริการของ AVisRouter ในการรับส่งข้อความ แล้วผลักภาระให้ระบบเป็นผู้จัดการความสัมพันธ์ของการรับส่งข้อความดังกล่าว AVisRouter มีหน้าที่ในการส่งผ่านข้อความไปยังจุดหมายที่มีความสัมพันธ์กับองค์ประกอบที่ใช้บริการ ควบคุมไม่ให้เกิดการซ้อนทับกันของข้อความที่จัดส่ง และรวมถึงการรับประกันว่าในกรณีที่ส่งให้ผู้รับหลายราย ผู้รับทุกๆรายจะได้รับข้อความที่ส่งพร้อมๆกัน

5.5.1 กราฟทางเดินของข้อความ

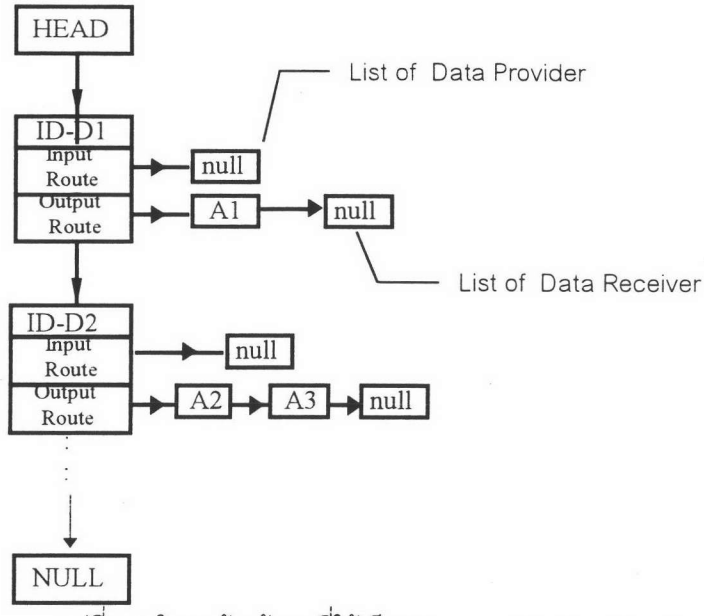
ความสัมพันธ์ของการรับส่งข้อความดังกล่าวนี้ จะบรรยายด้วยกราฟทางเดินของข้อความ (Message Routing Graph) ดังตัวอย่างในรูปที่ 5.4 ระบบจินตทัศน์อัลกอริทึมในงานวิจัยนี้แบ่งการเชื่อมต่อองค์ประกอบต่างๆเหล่านี้ออกเป็นสองประเภทคือ การเชื่อมต่อไปองค์ประกอบอื่นเพื่อขอข้อมูลด้วยข้อความคำสั่งขอข้อมูล และ การเชื่อมต่อไปองค์ประกอบอื่นเพื่อส่งข้อมูลหรือแสดงผลการทำงาน ในรูปที่ 5.4ลูกศรแสดงถึงทิศการไหลของข้อมูล



D : Data generator, A : Algorithm, C : Converter, V : View

รูปที่ 5.4 ตัวอย่าง Message Routing Graph ของบทการจินตทัศน์หนึ่ง

กราฟทางเดินของข้อความนี้ถูกเก็บในระบบโดยใช้รายการโยงแบบข้างเคียง (adjacency linked list) ซึ่งสามารถเพิ่มและลดได้ในขณะทำงานอย่างมีประสิทธิภาพ โดยแต่ละองค์ประกอบนั้นจะเก็บรายการโยงของผู้ส่งข้อความ และอีกหนึ่งรายการโยงสำหรับผู้รับข้อความขององค์ประกอบนั้นๆ ดังแสดงในรูปที่ 5.5 ซึ่งจะแสดงการจัดเก็บข้อมูลบางส่วนของกราฟทางเดินของข้อความในรูปที่ 5.4 ข้อมูลของแต่ละ node ในรายการนี้จะเก็บเฉพาะ ID ขององค์ประกอบเท่านั้น เมื่อต้องการรายละเอียดขององค์ประกอบนั้นๆก็สามารถสอบถามได้จากบริการของ AVisDB



รูปที่ 5.5 โครงสร้างข้อมูลที่ใช้เก็บ Message Routing Graph

กราฟทางเดินของข้อความจะถูกสร้างและเปลี่ยนแปลงภายใต้การควบคุมของโปรแกรมควบคุมการจินตทัศน์ เมื่อนำองค์ประกอบเข้าหรือออกจากระบบ เพื่อเชื่อมความสัมพันธ์ และเพื่อลบความสัมพันธ์ระหว่างองค์ประกอบต่างๆ โดยใช้ฟังก์ชัน AVisBind และ AVisUnbind ของ AVisAPI ตามลำดับ

5.5.2 บริการส่งข้อความของ AVisRouter

การเรียกใช้บริการส่งผ่านข้อความของ AVisRouter นั้นกระทำได้โดยใช้ฟังก์ชัน AVisInputRequest และ ฟังก์ชัน AVisOutputNotify การทำงานของทั้งสองฟังก์ชันเหมือนกันในแง่ที่ว่าฟังก์ชันทั้งคู่มีไว้เพื่อส่งข้อความคำสั่งไปยังองค์ประกอบอื่นๆ จะต่างกันก็ตรงที่ฟังก์ชัน AVisInputRequest นั้นจะส่งข้อความไปยังองค์ประกอบผู้ส่ง ในขณะที่ฟังก์ชัน AVisOutputNotify จะส่งข้อความไปยังทุกองค์ประกอบผู้รับ

```
AVisOutputNotify( compID, message )
{
    AVisSetWaitCount( # of data receivers of compID )
    for each data receiver i of compID {
        AVisSendMessage( compID,i, message )
    }
    AVisWaitforReply()
}
AVisSendMessage( compID, message )
{
    wait until compID is ready to receive
    /* Windows API */
    SendMessage( compID, message )
}
AVisWaitforReply ( )
{
  while ( nWaitCount > 0 ) {
    YieldControl()
  }
}
```

รูปที่ 5.6 กลไกการส่งข้อความคำสั่งของ AVisRouter

ในกรณีของการเรียกใช้ฟังก์ชัน AVisOutputNotify หากมีองค์ประกอบผู้รับหลายรายที่มีช่องทางต่อจากองค์ประกอบที่ส่งข้อความ AVisRouter จะต้องส่งข้อความไปยังองค์ประกอบผู้รับต่างๆเหล่านั้นพร้อมๆกัน ทั้งนี้เพื่อให้ผู้รับทุกๆตัวตอบสนองการทำงานตามข้อความที่ส่งไปพร้อมๆกัน (หรือเกือบพร้อมกัน) จึงเสมือนกับ

การที่องค์ประกอบผู้ส่งสั่งให้องค์ประกอบผู้รับทำงานแบบขนาน (Parallel Call) โดยไม่ต้องรอให้ผู้รับทำงานเสร็จทีละรายๆ โดยมีขั้นตอนการจัดการการสั่งงานแบบขนานดังรูปที่ 5.6

5.5.3 กลไกส่งข้อความของ AVisRouter

จากฟังก์ชันที่แสดงในรูปที่ 5.6 มีการใช้ semaphore ชื่อว่า nWaitCount เพื่อเก็บจำนวนองค์ประกอบที่ได้รับข้อความคำสั่งแล้วยังทำงานไม่เสร็จ นั่นคือ nWaitCount จะถูกตั้งค่าเท่ากับจำนวนผู้รับข้อความคำสั่งเสียก่อน แล้วจึงทยอยส่งข้อความ จากนั้นเข้าสู่วงวนรอจนกว่า nWaitCount จะมีค่าเป็นศูนย์ (อนึ่งระหว่างการรอนั้นจะเรียกฟังก์ชัน YieldControl เพื่อเปิดโอกาสให้โปรแกรมอื่นๆในระบบได้ทำงาน) องค์ประกอบผู้รับตัวใดที่ทำงานเสร็จก็จะลดค่าของ nWaitCount ลงหนึ่งโดยใช้ฟังก์ชัน AVisReplyMessage เมื่อ nWaitCount มีค่าเป็นศูนย์ก็แสดงว่าการสั่งการผ่านการเรียกใช้ฟังก์ชัน AVisOutputNotify ได้ทำงานเสร็จเรียบร้อย นี้จึงเปรียบเสมือนกับการเรียกใช้โปรแกรมย่อยในองค์ประกอบผู้รับข้อความต่างๆแบบขนาน

นอกจากนี้การส่งข้อความต่างๆใน AVisRouter ยังมีสิ่งซึ่งต้องคำนึงถึงอีกสองประการก็ได้แก่

1.จะต้องรับประกันได้ว่าองค์ประกอบที่รับข้อความจะต้องพร้อมที่รับข้อความ นั่นคือไม่ได้อยู่ในขั้นตอนการทำงานเพื่อตอบสนองข้อความอื่นๆอยู่ หรืออีกนัยหนึ่งคือการป้องกันการส่งข้อความซ้อนทับกัน

2.จะต้องป้องกันการเกิดการติดตาย(dead lock)ของระบบวินโดว์อันเนื่องมาจากการใช้ฟังก์ชัน SendMessage ของวินโดว์

การควบคุมและป้องกันเงื่อนไขดังกล่าวนี้ กระทำได้โดยการสร้างฟังก์ชัน AVisSendMessage ขึ้นใช้แทนการใช้ฟังก์ชันการส่งข้อความของ Windows โดยตรง โดยใน AVisSendMessage จะมีการตรวจสอบความพร้อมขององค์ประกอบที่จะรับข้อความก่อนที่จะส่งและมีกลไกป้องกันการเกิดการติดตาย โดยจะทำงานประสานกับองค์ประกอบการจินตทัศน์ที่จะได้รับข้อความคำสั่ง เพื่อตรวจสอบและป้องกันสภาวะดังกล่าว

5.5.4 การตรวจสอบความพร้อมของผู้รับ

การตรวจสอบความพร้อมขององค์ประกอบผู้รับก่อนส่งข้อความคำสั่งจะมีสองขั้นตอนคือ ขั้นแรก AVisRouter จะตรวจสอบค่าตัวบ่งชี้สถานะ (status flags) ขององค์ประกอบเพื่อตรวจดูว่าองค์ประกอบพร้อมที่จะรับข้อความคำสั่งที่จะส่งไปหรือไม่หากไม่พร้อมก็จะทำการหยุดรอจนกว่าองค์ประกอบที่เป็นผู้รับข้อความคำสั่งจะพร้อม ซึ่งองค์ประกอบที่เป็นผู้รับข้อความคำสั่งสามารถกำหนดและยกเลิกค่าของตัวบ่งชี้สถานะของการรับข้อความคำสั่งต่างๆได้ด้วยฟังก์ชัน AVisSuspendMessage และ AVisResumeMessage ของ AVis

หลังจากตรวจสอบจากค่าตัวบ่งชี้สถานะขององค์ประกอบแล้วว่าองค์ประกอบพร้อมที่จะรับข้อความคำสั่ง AVisRouter จะส่งข้อความคำสั่งด้วยฟังก์ชัน SendMessage ไปให้องค์ประกอบ แล้วตรวจสอบค่าผลการทำงานของฟังก์ชัน SendMessage โดยในการออกแบบระบบ AVis ได้กำหนดไว้ว่าหากองค์ประกอบได้รับข้อความคำสั่งและพร้อมที่จะทำงานในทันทีให้ตอบสนองต่อข้อความคำสั่งด้วยค่า AVIS_MSGRESULT_OK (0x0000) แต่หากไม่พร้อมและต้องการให้ AVisRounter ส่งข้อความคำสั่งให้ใหม่ในภายหลังให้ ตอบสนองต่อข้อความคำสั่งด้วยค่า AVIS_MSGRESULT_SUSPENDED (0xFFFF) ซึ่งหากองค์ประกอบตอบสนองต่อข้อความคำสั่งด้วยค่า AVIS_MSGRESULT_SUSPENDED AVisRouter ก็จะให้ผู้ส่งหยุดรอจนกว่าผู้รับจะพร้อมจึงทำการส่งข้อความคำสั่งใหม่

5.5.5 การป้องกันการติดตาย

ปัญหาประการหนึ่งของการส่งข้อความคำสั่งด้วยฟังก์ชัน SendMessage ก็คือการติดตายของระบบวินโดว์ ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อมีการส่งข้อความคำสั่งไปให้วินโดว์ของโปรแกรมอื่นและในการทำงานเพื่อตอบสนองข้อความคำสั่งนั้นมีการเรียกใช้ฟังก์ชันที่ทำให้เกิดการสลับการทำงานของโปรแกรมในระบบวินโดว์ขึ้น ทั้งนี้เนื่องจากในข้อกำหนดของฟังก์ชัน SendMessage นั้นผู้ส่งจะต้องหยุดรอจนกว่าผู้รับจะตอบสนองการทำงานของข้อความคำสั่ง แต่เมื่อเกิดการสลับการทำงานขึ้นในขณะที่องค์ประกอบผู้รับกำลังทำงานอยู่ ระบบวินโดว์จะต้องเลือกโปรแกรมอื่นมาทำงานแทน และหากโปรแกรมที่เลือกได้เป็นโปรแกรมซึ่งเป็นผู้ส่งซึ่งไม่สามารถทำงานต่อได้เพราะหยุดรอการทำงานของฟังก์ชัน SendMessage อยู่ก็จะทำให้ระบบวินโดว์เกิดการติดตายขึ้น

ในระบบวินโดว์การป้องกันการติดตายสามารถกระทำโดยการเรียกใช้ฟังก์ชัน ReplyMessage ก่อนที่จะเรียกใช้ฟังก์ชันที่ทำให้เกิดการสลับการทำงานของโปรแกรม เมื่อมีการ เรียกใช้ฟังก์ชัน ReplyMessage ระบบวินโดว์จะยินยอมให้ผู้ส่งข้อความคำสั่งทำงานต่อไปได้เสมือนว่าข้อความคำสั่งได้รับการตอบสนองแล้วทำให้ไม่เกิดการติดตายเมื่อเกิดการสลับการทำงาน ฟังก์ชัน ReplyMessage จะรับพารามิเตอร์หนึ่งค่าซึ่งก็คือค่าผลการทำงานที่ผู้ส่งจะได้รับจากฟังก์ชัน SendMessage

รูปที่ 5.7 แสดงโปรแกรมจำลองเพื่อแสดงขั้นตอนการทำงานของ AVisDispatchEvent ที่องค์ประกอบผู้รับข้อความจะต้องเรียกเพื่อรับข้อความคำสั่งของ AVis ฟังก์ชันนี้จะทำงานร่วมกับฟังก์ชัน AVisSendMessage เพื่อทำให้การรับและส่งข้อความคำสั่งของระบบ AVis เป็นไปด้วยความเรียบร้อย

```
AVisDispatchEvent(hwnd, msg,UserMsgResponseRoutine)
{
    if msg is not a AVIS message
        return FALSE;
    if not ready to receive message {
        ReplyMessage(AVIS_MSGRESULT_SUSPEND)
        return TRUE
    }
    AVisSuspendMessage(msg) /* prevent message overrun */

    /* use Windows API to prevent dead lock from SendMessage*/
    ReplyMessage(AVIS_MSGRESULT_OK)

    /* Call user routine to response AVIS message */
    retval = UserMsgResponseRoutine(msg)

    /*  reset flags set in AVisSuspendMessage */
    AVisResumeMessage(msg);
    AVisReplyMessage(result); /* Reply to sender */
}
```

รูปที่ 5.7 การทำงานของฟังก์ชัน AVisDispatchEvent

5.6 AVisSync

ในระบบจินตทัศน์อัลกอริทึมนั้น ผู้ใช้งานอาจต้องการสังเกตการทำงานของหลายๆอัลกอริทึมในเวลาเดียวกัน ทั้งนี้เพื่อเปรียบเทียบพฤติกรรมของอัลกอริทึม อันรวมถึงเวลาการทำงานเชิงเปรียบเทียบของอัลกอริทึม ดังนั้นระบบจะต้องมีส่วนให้บริการการประสานจังหวะการทำงานเพื่อให้แต่ละอัลกอริทึมมีโอกาสได้ทำงานอย่างยุติธรรม เนื่องจากสภาพปฏิบัติการไมโครซอฟต์วินโดวส์ทำงานแบบหลายภารกิจในลักษณะร่วมกันทำงาน (coorporative multitasking หรือที่เรียกว่า non pre-emptive) นั่นคือโปรแกรมต่างๆในระบบจะต้องคอยแบ่งเวลาการทำงานให้กันและกัน(หากผู้ใดไม่ยอมปล่อยให้ผู้อื่นทำงานผู้อื่นก็ทำงานไม่ได้) ส่วนที่ให้บริการและคอย

ควบคุมการประสานจังหวะการทำงานขององค์ประกอบการจินตทัศน์ต่างๆเพื่อรับประกันว่าเวลาการทำงานสัมพัทธ์ของอัลกอริทึมต่างๆเป็นไปอย่างถูกต้องก็คือ AVisSync นั้นเอง

โครงสร้างข้อมูลที่ใช้ประกอบการประสานจังหวะการทำงานของอัลกอริทึมคือ Synchronization list ซึ่งสร้างแบบรายการโยงแบบวน (circular linked list) ทั้งนี้เพื่อให้สอดคล้องกับวิธีการจัดลำดับการทำงานขององค์ประกอบอัลกอริทึม ที่ใช้วิธี round robin (นั่นคือวิธีจัดลำดับการทำงานแบบผลัดกันทำ) เพื่อให้โอกาสทุกอัลกอริทึมได้ทำงานโดยเท่าเทียมกัน องค์ประกอบอัลกอริทึมจะถูกเพิ่มเข้าหรือลบออกจาก synchronization list เมื่อองค์ประกอบนั้นมีการเรียกใช้บริการ AVisOpenSync หรือ AVisCloseSync ของ AVisAPI ตามลำดับ

ในการประสานจังหวะการทำงานนี้ AVisSync จะต้องได้รับความร่วมมือจากผู้พัฒนาองค์ประกอบอัลกอริทึม ทั้งนี้เพราะ AVisSync จะสลับการทำงานของอัลกอริทึมก็ต่อเมื่อองค์ประกอบอัลกอริทึมเรียกใช้ฟังก์ชัน AVisSync ของ AVisAPI เพื่อแจ้งว่าทำงานผ่านคำสั่งการทำงานพื้นฐานแล้วเท่านั้น ซึ่งการทำงานของฟังก์ชัน AVisSync จะมีการทำงานดังในรูปที่ 5.8

```
AVisSync(component)
{
   do {  /* allow a task switching to another window application
         even there is only one algorithm */
     YieldControl()
   }
   /* wait until our turn */
   while ( Next_Runable_Component !=  component );
   if component is the last component in a list {
        while (synchronization mode == Pause ) {
             YieldControl()  /* Wait until not pause */
        }
        if (synchronization mode == step ) {
             Notify AVisController /* Handle step mode */
        }
        while AVisController dose not allow AVisSync to continue {
             /* Wait until controller allow AVisSync to continue */
             YieldControl()
        }
   }
   Next_Runable_Component = next component in a synchronization list
}
```

รูปที่ 5.8 การทำงานของฟังก์ชัน AVisSync

นอกจาก AVisSync จะมีหน้าที่ในการประสานจังหวะการทำงานแล้วนั้น AVisSync ยังมีหน้าที่ในการควบคุมการทำงานและความเร็วของการจินตทัศน์ โดยผู้ใช้สามารถสั่งให้ AVisSync เริ่มหรือหยุดการจินตทัศน์ผ่านโปรแกรมควบคุมการจินตทัศน์ได้ โดยการจินตทัศน์จะมีสถานะการทำงาน(visualization status) อยู่สองสถานะคือ (1) หยุดรอ(stop) และ (2)ทำงาน(run)

เมื่อการจินตทัศน์อยู่ในสถานะทำงาน ผู้ใช้สามารถควบคุมสั่งให้ AVisSync ทำการควบคุมความเร็วของการทำงานของอัลกอริทึมได้ ตั้งแต่การให้ทำงานทีละหนึ่งคำสั่งพื้นฐาน (single step) หยุดการทำงานชั่วคราว(pause) และทำงานตามปกติ(run) ลักษณะการทำงานเหล่านี้เรียกว่าภาวะการทำงานของการประสานจังหวะ (synchronization mode) การควบคุมภาวะการทำงานของการประสานจังหวะของ AvisSync สามารถทำได้โดยการใช้คำสั่ง AVisSetSyncMode ของ AVisAPI

5.7 AVisErr

เนื่องจากระบบจินตทัศน์อัลกอริทึมที่ออกแบบและพัฒนาขึ้นนี้ทำงานบน Microsoft Windows ซึ่งมีลักษณะการทำงานของโปรแกรมหลายโปรแกรมแบบร่วมกันทำงาน ดังนั้นหากเกิดความผิดพลาดใดๆกับองค์ประกอบ อาจเป็นผลให้การทำงานของทั้งระบบหยุดชะงักไปได้ AVisErr เป็นหน่วยพิเศษภายในหน่วยบริหารการจินตทัศน์ที่มีหน้าที่คอยตรวจสอบว่าองค์ประกอบการจินตทัศน์ต่างๆรวมทั้งโปรแกรมควบคุมการจินตทัศน์ยังทำงานอยู่ในระบบอย่างปกติหรือไม่ โดยจะตรวจสอบว่าโปรแกรมเหล่านี้เลิกการทำงานไปโดยไม่ได้ถอนทะเบียนออกจากระบบหรือไม่ หากเกิดเหตุการณ์ดังกล่าว

5.7.1 ผลของข้อผิดผลาดต่อระบบ

การที่องค์ประกอบการจินตทัศน์และโปรแกรมควบคุมการจินตทัศน์หยุดทำงานโดยไม่ทำการถอนทะเบียนจากระบบก่อน จะมีผลต่อการทำงานของระบบ ซึ่งผลกระทบเหล่านี้ได้แก่

1. ทำให้โครงสร้างข้อมูลภายใน AVisDB, AVisRouter, AVisSync ไม่ตรงกับสภาพความเป็นจริง
2. หากข้อผิดผลาดเหล่านี้เกิดขึ้นกับผู้รับข้อความขณะทำงานตามข้อความที่ได้รับ จะทำให้ผู้ส่งข้อความต้องรออย่างไม่มีที่สิ้นสุด เพราะกลไกของการสั่งการแบบขนานป้องกันไม่ให้ผู้ส่งข้อความทำงานอื่นจนกว่าผู้รับข้อความจะตอบรับข้อความกลับมาหมดทุกราย
3. หากองค์ประกอบอัลกอริทึมที่ขอใช้บริการการประสานจังหวะหยุดทำงานโดยไม่แจ้งให้ AVisSync ทราบ ทำให้ AvisSync ไม่สามารถส่งต่อการทำงานให้กับองค์ประกอบอัลกอริทึมอื่นๆในระบบได้ ยังผลให้เกิดการหยุดรออย่างไม่สิ้นสุดขององค์ประกอบอัลกอริทึมอื่นๆเหล่านั้น

เหตุการณ์เช่นนี้จะเกิดขึ้นบ่อยมากในช่วงของการพัฒนาองค์ประกอบต่างๆระหว่างการทดสอบ หากผู้พัฒนาเขียนโปรแกรมผิดพลาด อาจเกิด abnormal termination โดยที่โปรแกรมควบคุมการจินตทัศน์ไม่รับรู้ได้

5.7.2 สาเหตุของการหยุดทำงานของโปรแกรมในระบบวินโดว์

การหยุดการทำงานอย่างผิดปกติขององค์ประกอบในระบบซึ่งพัฒนาขึ้นในรูปแบบของโปรแกรมบนระบบวินโดว์มีได้หลายสาเหตุ ซึ่งนอกจากความผิดพลาดที่เกิดขึ้นกับตัวองค์ประกอบเองแล้ว โดยทั่วไปมักเกิดขึ้นจากการเรียกใช้บริการของวินโดวส์ที่ไม่ถูกต้องหรือตัววินโดวส์เองที่ทำงานผิดพลาด ซึ่งจะทำให้องค์ประกอบนั้นหยุดการทำงานทันที ซึ่งสาเหตุของการเกิดข้อผิดผลาดเหล่านี้ตามที่ระบุไว้ในเอกสาร ของไมโครซอฟต์ [1] ได้แก่

1. ความผิดพลาดของ stack ภายในระบบ โดยทั่วไปมีสาเหตุมาจากการพยายามอ้างอิงที่อยู่ในหน่วยความจำที่เกินช่วงของ stack segment
2. การทำงานคำสั่งที่ไม่ถูกต้อง โดยทั่วไปเกิดจากการพยายามไปเริ่มทำคำสั่งในช่วงที่เป็นข้อมูล หรือการเปลี่ยนแปลงคำสั่งในหน่วยความจำจนทำให้คำสั่งผิด
3. การคำนวณที่ได้ผลผิดพลาด โดยเฉพาะการหาร เช่นการหารด้วยศูนย์ เป็นต้น
4. ข้อผิดพลาดในการอ้างอิงหน่วยความจำ เช่นการอ้างอิงเกินขอบเขต (เนื่องมาจากการคำนวณตำแหน่งของหน่วยความจำผิดพลาด) หรือ การอ้างอิงหน่วยความจำตำแหน่ง 0 (NULL pointer) หรือ การเขียนในหน่วยความจำที่อนุญาตให้อ่านอย่างเดียว เป็นต้น

---

[1] Microsoft, Causes of General Protection Faults, *Microsoft Developers' Network : Window 3.x Knowledge Base, version 3.0*, Microsoft Inc., April, 1995.

5.7.3 การตรวจสอบข้อผิดพลาด

หน้าที่หลักของ AVisErr คือการคอยตรวจสอบว่ามีองค์ประกอบการจินตทัศน์ใดบ้างในระบบที่หยุดการทำงานอย่างผิดปกติ เพื่อจะได้แก้ไขข้อผิดพลาดดังกล่าว AVisErr จะคอยตรวจสอบข้อผิดผลาดเหล่านี้โดยใช้วิธีการตรวจสอบเป็นระยะๆ วิธีนี้กระทำโดยการกำหนดให้ฟังก์ชันตรวจสอบและจัดการข้อผิดพลาด ถูกเรียกใช้ตรวจสอบระบบเป็นระยะๆเช่นทุก 1 วินาที เป็นต้น การทำงานนี้ก็เพียงแต่ตรวจสอบว่าองค์ประกอบต่างๆที่เก็บไว้ใน AVisDB นั้นยังคงทำงานอยู่ในระบบหรือไม่ ในกรณีที่องค์ประกอบใดที่มีทะเบียนในระบบ แต่ไม่ปรากฏว่ากำลังทำงานอยู่ ก็แสดงว่าเกิดข้อผิดพลาดขึ้น ข้อผิดพลาดที่ตรวจพบก็จะถูกจัดการให้ระบบมีการทำงานตามปกติ

การตรวจสอบข้อผิดผลาดโดยใช้วิธีตรวจสอบระบบเป็นระยะ จะมีข้อดีคือเป็นวิธีที่ทำได้ง่าย ไม่ซับซ้อน อีกทั้งแนวคิดนี้สามารถนำไปใช้ได้กับระบบปฏิบัติการโดยทั่วไป แต่ก็มีข้อเสียคือการตรวจสอบจะต้องตรวจสอบทุกๆองค์ประกอบที่มีทะเบียนในระบบ เพื่อหาเฉพาะองค์ประกอบที่ทำงานผิดพลาดซึ่งไม่เกิดขึ้นบ่อยนัก ทำให้เสียเวลาและลดประสิทธิภาพการทำงานของระบบลงหากตั้งช่วงการตรวจสอบถี่เกินไป ดังนั้น AVisErr จะทำการตรวจสอบระบบทุกๆ 1 วินาที ซึ่งในทางปฏิบัติการแล้วจะไม่กระทบประสิทธิภาพการทำงานของระบบ ทั้งนี้เนื่องจากเวลาส่วนใหญ่ที่เสียไปในการจินตทัศน์จะอยู่ที่การแสดงผลการทำงานแบบกราฟิกขององค์ประกอบแสดงผล

5.7.4 การแก้ไขข้อผิดพลาด

หลังจากที่เรารับรู้ว่าองค์ประกอบหยุดทำงานอย่างผิดปกติ หน้าที่ต่อไปของ AVisErr ก็คือการแจ้งเตือนไปให้ส่วนอื่นๆ ของระบบจินตทัศน์อันได้แก่ AVisRouter AVisSync โปรแกรมควบคุมการจินตทัศน์ และ AVisDB ให้ทราบตามลำดับ เพื่อทำการแก้ไขข้อมูลในส่วนที่ตัวเองดูแลอยู่ ดังนี้

1. AVisRouter จะทำการลบหมายเลขประจำองค์ประกอบที่เกิดปัญหาออกจากโครงสร้างข้อมูลที่ใช้เก็บหมายเลขประจำองค์ประกอบทั้งหมดที่องค์ประกอบผู้ส่งข้อความคำสั่งจะต้องหยุดรอ เพื่อป้องกันไม่ให้องค์ประกอบผู้ส่งหยุดรอองค์ประกอบที่เกิดปัญหาแบบไม่สิ้นสุด จากนั้นจะให้ค่าแสดงความผิดพลาดดังกล่าวให้กับองค์ประกอบผู้ขอใช้บริการส่งข้อความคำสั่งทราบผ่านทางค่าผลการทำงานของฟังก์ชัน AVisOutputNotify หรือ AVisInputRequest ต่อไป
2. AVisSync จะลบข้อมูลเกี่ยวกับองค์ประกอบที่เกิดปัญหาออกจาก Synchronization list และหากองค์ประกอบที่เกิดปัญหาเป็นองค์ประกอบที่ได้รับสิทธ์จาก AVisSync ให้ทำงาน AVisSync ก็จะยกเลิกสิทธ์นั้นและนำสิทธิ์การทำงานไปให้องค์ประกอบอัลกอริทึมองค์ประกอบถัดไป เพื่อป้องกันไม่ให้เกิดการรออย่างไม่สิ้นสุด เนื่องจากองค์ประกอบที่ได้รับสิทธ์การทำงานไม่ยอมมอบสิทธ์การทำงานให้กับองค์ประกอบถัดไปทำงานด้วยฟังก์ชัน AVisSync
3. โปรแกรมควบคุมการจินตทัศน์ หลังจาก AVisErr แจ้งเตือนให้ AVisRouter และ AVisSync ทราบแล้ว ในช่วงนี้ระบบจะอยู่ในสภาวะที่มีเสถียรภาพพอที่จะแจ้งให้ โปรแกรมควบคุมการจินตทัศน์ซึ่งเป็นส่วนประกอบภายนอกของหน่วยบริหารการจินตทัศน์ทราบเพื่อแจ้งข้อผิดพลาดที่เกิดขึ้นกับผู้ใช้ และทำการปรับเปลี่ยนสภาพการทำงานของระบบจินตทัศน์อัลกอริทึมให้เหมาะสมกับเหตุการณ์ที่เกิดขึ้นต่อไป

4. AVisDB จะเป็นส่วนสุดท้ายของหน่วยบริหารการจินตทัศน์ที่จะได้รับการแจ้งเตือน ทั้งนี้เพราะการแก้ไขข้อผิดพลาดของ AVisDB จะทำโดยการลบข้อมูล หรือทำการถอนทะเบียนขององค์ประกอบที่หยุดทำงานออกไปจากระบบ ซึ่งข้อมูลเหล่านี้อาจมีความจำเป็นต่อการแก้ไขข้อผิดพลาดของส่วนประกอบสามส่วนแรก จึงต้องแจ้งเตือนให้ทั้งสามส่วนนั้นทำการแก้ไขก่อน แล้วจึงแจ้งเตือนให้ AVisDB เป็นขั้นตอนสุดท้าย

หลังจากที่ส่วนประกอบเหล่านี้ทำการแก้ไขข้อผิดผลาดเรียบร้อยแล้ว ระบบ AVis ก็จะกลับเข้าสู่การทำงานตามปรกติอีกครั้งหนึ่ง พร้อมที่จะรับคำสั่งต่างๆจากผู้ใช้และทำงานต่อไปอย่างต่อเนื่อง โดยมีผลกระทบต่อผู้ใช้น้อยที่สุด

5.8 สรุป

ในบทนี้ได้นำเสนอโครงสร้างและการทำงานของหน่วยบริหารการจินตทัศน์ (AVisExecutive) อันเป็นแก่นกลางของระบบจินตทัศน์อัลกอริทึม ซึ่งทำงานบนสภาพปฏิบัติการไมโครซอฟต์วินโดวส์ หน่วยบริหารนี้จะคอยให้บริการและควบคุมการทำงานของส่วนประกอบอื่นๆของระบบ AVis โดยการให้บริการและการควบคุมการทำงานเหล่านี้สามารถแบ่งเป็นกลุ่มๆได้ห้ากลุ่มได้แก่ (1)ให้บริการทางด้านการส่งผ่านข้อความ (AVisRouter) (2)การประสานจังหวะการทำงานของอัลกอริทึมต่างๆ(AVisSync) (3)การสอบถามข้อมูลต่างๆของสภาพการจินตทัศน์(AVisDB) (4)การจัดการความผิดพลาดในระบบ (AVisErr) และ(5)ชุดเชื่อมประสานกับโปรแกรมประยุกต์ (AVisAPI) หน่วยบริหารการจินตทัศน์ถูกพัฒนาขึ้นในรูปแบบของคลังโปรแกรมเชื่อมโยงแบบพลวัตที่มีชื่อว่า AVISDLL.DLL ซึ่งทำให้การนำไปใช้งานทำได้ง่ายและไม่ขึ้นกับภาษาการเขียนโปรแกรมที่ใช้พัฒนาองค์ประกอบการจินตทัศน์