การจัดการข้อผิดผลาดในระบบจินตทัศน์อัลกอริทึม

Error Handling in Algorithm Visualization System

ชัชวาล วงศ์ศิริประเสริฐ
นิสิตปริญญาโท
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย กรุงเทพฯ

สมชาย ประสิทธิ์จูตระกูล
ผู้ช่วยศาสตราจารย์
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย กรุงเทพฯ

บทคัดย่อ

บทความนี้นำเสนอกลไลการทำงานของส่วนจัดการข้อผิดพลาดของระบบจินตทัศน์อัลกอริทึม ที่ทำงานบนระบบปฏิบัติการไมโครซอฟต์วินโดวส์ ข้อผิดพลาดที่ได้รับตรวจสอบประกอบด้วยข้อผิดพลาดระหว่างการขอใช้บริการการจินตทัศน์ ข้อผิดพลาดนี้ถูกจัดการโดยตัวควบคุมเฉพาะงานการประสานการจินตทัศน์กับการใช้คำสั่ง **`On Error Goto`** ของระบบการพัฒนาโปรแกรมวิชวลเบสิก ข้อผิดพลาดอีกประการหนึ่งคือการหยุดการทำงานอย่างผิดปกติขององค์ประกอบที่ทำงานในระบบ ข้อผิดพลาดแบบนี้ถูกจัดการโดยการตรวจสอบข้อผิดพลาดดังกล่าวในระบบเป็นระยะๆ เพื่อจะได้ปรับข้อมูลภายในระบบให้ตรงกับสภาพความเป็นจริงขององค์ประกอบที่เป็นอยู่ และแจ้งให้ส่วนเกี่ยวข้องอื่นๆในระบบ ด้วยกลไกการจัดการข้อผิดพลาดดังกล่าวทำให้ระบบสามารถแก้ไขความผิดผลาดเหล่านี้ได้ด้วยตัวเอง พร้อมที่จะทำงานได้อย่างต่อเนื่อง เพื่อลดภาระการจัดการข้อผิดพลาดของผู้พัฒนาการจินตทัศน์

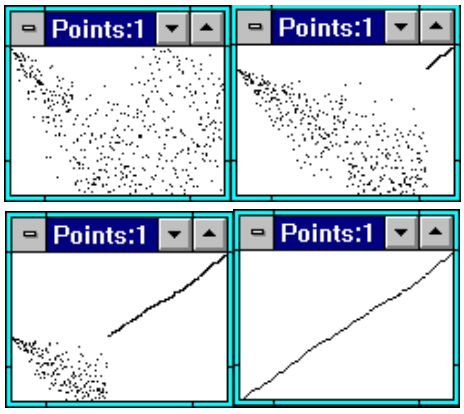
Abstract

This paper presents the mechanisms of error handler of the algorithm visualization system running on Microsoft Windows operating environment. The errors can be caused by errors during visualization services provided by the Executive. This is handled by a visualization custom control and the use of "On Error Goto" statement in Visual Basic Programming Development System. In addtion, errors can be caused by abnormal terminations of system components. The abnormal termination error can be detected by periodically checking the system components for data consistency. The proposed error handling mechanisms make the system self-recovery and ready for continuous operation. Moreover it lessens the error handling task of algorithm visualization developers.

1. บทนำ

การจินตทัศน์อัลกอริทึม (ALGORITHM VISUALIZATION) เป็นกรรมวิธีหนึ่งในการศึกษาทำความเข้าใจการทำงานของอัลกอริทึม ด้วยการใช้ภาพ และการเปลี่ยนแปลงของภาพ เป็นสื่อในการแสดงถึงขั้นตอนการทำงานของอัลกอริทึม ในบางครั้งการใช้กรรมวิธีนี้จะเป็นเปิดโลกของผู้ทำการศึกษาเข้าสู่พฤติกรรมของการทำงานของอัลกอริธึมที่ไม่เคยจินตนาการและคาดหวังมาก่อน ตัวอย่างระบบจินตทัศน์อัลกอริทึมและงานวิจัยที่เกี่ยวข้องสามารถศึกษาเพิ่มเติมได้ใน [1-16]

รูปที่ 1 แสดงตัวอย่างการจินตทัศน์วิธีการเรียงลำดับข้อมูลแบบฮีป (HEAP SORT) โดยแสดงภาพบางภาพระหว่างการแสดงขั้นตอนการเรียงลำดับข้อมูลบนข้อมูลที่มีค่าเริ่มต้นแบบสุ่ม โดยแสดงข้อมูลทั้งหมดในระนาบสองมิติที่ประกอบด้วยจุดต่างๆ จุดหนึ่งจุดแทนข้อมูลหนึ่งตัว มีพิกัด x ของจุดแทนตำแหน่งของข้อมูลนั้นในรายการ และพิกัด y ของจุดแทนค่าของข้อมูลนั้น ดังนั้นเมื่อเรียงลำดับเสร็จเรียบร้อยจากค่าน้อยไปค่ามาก จะได้ผลลัพธ์ดังรูปขวาสุดล่างของรูปที่ 1



รูปที่ 1 ภาพการจินตทัศน์การเรียงลำดับข้อมูลแบบฮีป

เพื่ออำนวยความสะดวกแก่ผู้สร้างการจินตทัศน์สำหรับอัลกอริทึมอื่นๆ ระบบจินตทัศน์อัลกอริทึมจะต้องมีความสามารถในการจัดการข้อผิดพลาด (ERROR HANDLING) ได้อย่างดี ทั้งนี้เนื่องจากการจินตทัศน์อัลกอริทึมหนึ่งๆ จะประกอบไปด้วยการทำงานร่วมกันของหลายๆองค์ประกอบ (โปรแกรม) โดยทั่วไป 6 องค์ประกอบขึ้นไป ทำงานประสานงานกันในระบบพร้อมๆ กันไป หากองค์ประกอบที่ผู้สร้างการจินตทัศน์พัฒนาขึ้นเกิดข้อผิดพลาด มักมีผลกระทบถึงองค์ประกอบอื่นๆในระบบ ดังนั้นการจัดการข้อผิดพลาดจึงเป็นสิ่งที่หลีกเลี่ยงไม่ได้ จะต้องถูกจัดการในเวลาอันสั้น ใช้ทรัพยากรน้อย เพื่อลดภาระของผู้เขียนโปรแกรมเพื่อสร้างการจินตทัศน์ให้มากสุด

บทความนี้นำเสนอโครงสร้างและการทำงานของส่วนจัดการข้อผิดผลาดของระบบการจินตทัศน์อัลกอริทึม ซึ่งทำงานบนระบบปฏิบัติการไมโครซอฟต์วินโดวส์ โดยมีลำดับการนำเสนอดังนี้ หัวข้อที่ 2 บรรยายสรุปถึงลักษณะโครงสร้างของระบบการจินตทัศน์อัลกอริทึม โดยการทำงานของส่วนจัดการข้อผิดพลาดจะถูกนำเสนออย่างละเอียดในหัวข้อที่ 3 จากนั้นสรุปสาระของบทความนี้ในหัวข้อที่ 4

2. โครงสร้างของระบบจินตทัศน์อัลกอริทึม

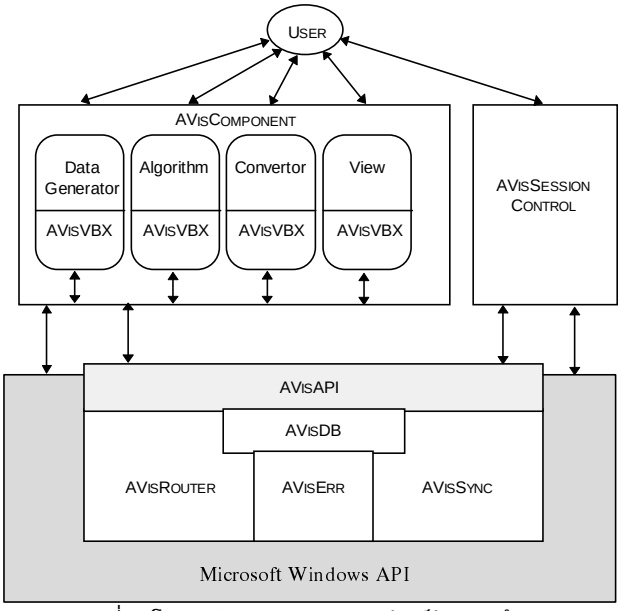
ระบบจินตทัศน์อัลกอริทึมประกอบส่วนต่างดังนี้ (ดูรูปที่ 2สำหรับรายละเอียดของส่วนต่างๆสามารถศึกษาได้จาก [1])

1 *หน่วยบริหารการจินตทัศน์* (AVISEXEC) เป็นแกนกลางสำหรับการจินตทัศน์ ส่วนนี้ให้บริการงานต่างๆดังนี้ การรับบริการจากองค์ประกอบอื่นๆ (AVISAPI) การประสานจังหวะการทำงานขององค์ประกอบต่างๆ (AVISSYNC) การส่งผ่านข้อความคำสั่งต่างๆระหว่างองค์ประกอบต่างๆ (AVISROUTER) การให้บริการการสอบถามข้อมูลต่างๆของสภาพการทำงาน (AVISDB) และการจัดการความผิดพลาดที่อาจเกิดขึ้นในระบบ (AVISERR)

2 *หน่วยควบคุมบทจินตทัศน์* (AVISSESSIONCONTROL) คือส่วนควบคุม ออกแบบ ตั้งค่า และสั่งงานการจินตทัศน์ ส่วนนี้จะเป็นส่วนที่ติดต่อประสานงานกับผู้ใช้โดยตรง เพื่อให้เกิดความคล่องตัวระหว่างการจินตทัศน์

3 *องค์ประกอบการจินตทัศน์* (AVISCOMPONENT) อันประกอบด้วยองค์ประกอบการผลิตข้อมูล องค์ประกอบอัลกอริทึม องค์ประกอบการแปลงคำสั่ง และองค์ประกอบมุมมองเพื่อการแสดงผล องค์ประกอบต่างๆเหล่านี้ใช้บริการการจินตทัศน์ต่างๆจาก AVISEXEC

การจินตทัศน์หนึ่งๆประกอบด้วยการสั่งการองค์ประกอบการจินตทัศน์ที่เกี่ยวข้อง เพื่อให้การพัฒนาองค์ประกอบการจินตทัศน์เหล่านี้กระทำได้ง่ายและรวดเร็ว ผู้วิจัยเลือกระบบการพัฒนาโปรแกรมวิชวลเบสิก [17] โดยแต่ละองค์ประกอบจะมี AVISVBX ซึ่งเป็นตัวควบคุมเฉพาะงานของวิชวลเบสิก (Visual Basic Custom Control) เพื่อประสานการทำงานระหว่างองค์ประกอบการจินตทัศน์กับ AVISEXEC ตัวควบคุม AVISVBX นี้เองจะเป็นผู้ซ่อนความซับซ้อนของกลไกการประสานงานเหล่านี้จากผู้พัฒนาองค์ประกอบ [1]



รูปที่ 2 โครงสร้างของระบบจินตทัศน์อัลกอริทึม

3. การจัดการข้อผิดผลาด

เนื่องจากระบบจินตทัศน์อัลกอริทึมเป็นระบบซึ่งประกอบด้วยส่วนบริหารการจินตทัศน์ (AVISEXEC) และองค์ประกอบการจินตทัศน์ต่างๆ (AVISCOMPONENT) องค์ประกอบการจินตทัศน์นี้คือส่วนที่จะถูกพัฒนาขึ้นโดยผู้สร้างการจินตทัศน์โดยใช้ระบบการพัฒนาวิชวลเบสิก เพื่อเอื้ออำนวยการพัฒนาองค์ประกอบเหล่านี้ AVISEXEC จำต้องคำนึงถึงความผิดพลาดระหว่างการทำงานขององค์ประกอบการจินต์ทัศน์ที่อาจเกิดได้โดยเฉพาะอย่างยิ่งในช่วงที่กำลังพัฒนาและทดสอบองค์ประกอบ จากแนวคิดที่ว่า AVISCOMPONENT คือผู้ขอใช้บริการจาก AVISEXEC ที่เป็นผู้ให้บริการ ผู้ให้บริการจะต้องมีความสามารถในการตรวจสอบ แก้ไข และแจ้งข้อผิดพลาดที่เกิดขึ้นในขณะที่ผู้ใช้บริการจะเป็นผู้ที่จัดการกับข้อผิดพลาดเหล่านั้นเมื่อได้รับแจ้ง ทั้งนี้เพื่อให้ระบบสามารถทำงานต่อไปได้ด้วยความเรียบร้อยถึงแม้จะเกิดข้อผิดผลาดขึ้นในขณะทำงานก็ตาม การจัดการข้อผิดพลาดตามที่ได้กล่าวมานี้เป็นหน้าที่ในการประสานการทำงานระหว่าง AVISERR ในส่วนบริหารการจินต

ทัศน์ กับ AVISVBX ในองค์ประกอบการจินตทัศน์ ที่จะได้กล่าวในรายละเอียดต่อไป

3.1 ประภทของขษอผิดผลาด

ข้อผิดผลาดต่างๆ ที่สามารถเกิดขึ้นได้ในขณะทำการจินตทัศน์ มีดังต่อไปนี้

1 การเรียกใช้บริการ AVISEXEC ในลักษณะที่ไม่ถูกต้องมีดังนี้

1.1 *ข้อผิดพลาดของหมายเลขประจำองค์ประกอบที่ขอใช้บริการ* การขอใช้บริการเกือบทุกอย่างของ AVISEXEC จะต้องให้หมายเลขประจำองค์ประกอบมาด้วยเพื่อแจ้งว่าใครเป็นผู้ขอใช้บริการ AVISAPI จะทำการตรวจสอบว่าหมายเลขที่ให้มานี้มีอยู่จริงหรือไม่ หากเป็นหมายเลขที่ไม่ถูกต้องก็จะแจ้งผลการทำงานให้กลับผู้ใช้บริการว่าผิดผลาด

1.2 *ข้อผิดพลาดเกี่ยวกับสิทธิ์ของผู้ขอใช้บริการ* หากผู้ขอใช้บริการไม่มีสิทธ์ขอใช้บริการก็จะแจ้งผลการทำงานว่าผิดผลาดและเหตุผลที่ผิดผลาดกลับไป ตัวอย่างเช่น เฉพาะ AVISSESSIONCONTROL เท่านั้นที่มีสทธิ์ในการสั่งให้ระบบหยุดการจินตทัศน์ชั่วครู่ หรือการถอนทะเบียนองค์ประกอบใดกระทำเฉพาะจากองค์ประกอบซึ่งเป็นผู้ลงทะเบียนเท่านั้น เป็นต้น

1.3 *ที่อยู่ของหน่วยความจำที่อ้างอิงถึงไม่ถูกต้อง* ที่อยู่ของหน่วยความจำที่ส่งมาให้ จะต้องเป็นที่ที่ระบบปฏิบัติการวินโดวส์ยินยอมให้ AVISEXEC อ้างอิงถึง เพื่ออ่านหรือแก้ไขข้อมูล ณ ตำแหน่งนั้นๆ

2 ข้อผิดพลาดระหว่างการขอใช้บริการจาก AVISEXEC ตัวอย่างเช่นหน่วยความจำไม่พอ ทรัพยากรของระบบเหลือน้อยเกินไป เกิดการบอกเลิกการจินตทัศน์กลางครัน เป็นต้น

3 การหยุดการทำงานอย่างผิดปกติขององค์ประกอบในระบบ โดยองค์ประกอบนั้นยังมิได้ถอนทะเบียนออกจากระบบอย่างถูกต้อง ส่งผลให้ข้อมูลภายในของ AVISEXEC ผิดผลาดและไม่สอดคล้องกับความเป็นจริง จนอาจเป็นเหตุให้ระบบทำงานผิดผลาดได้หากไม่มีการแก้ไข ข้อผิดพลาดการหยุดการทำงานอย่างผิดปกตินี้ จะเกิดขึ้นบ่อยมาก โดยเฉพาะในช่วงการพัฒนาองค์ประกอบการจินตทัศน์ นอกจากความผิดพลาดที่เกิดขึ้นกับตัวองค์ประกอบเองแล้ว โดยทั่วไปมักเกิดขึ้นจากการเรียกใช้บริการของวินโดวส์ที่ไม่ถูกต้อง หรือตัววินโดวส์เองที่ทำงานผิดพลาด ซึ่งจะทำให้องค์ประกอบนั้นหยุดการทำงานทันที อันมีสาเหตุมาจาก [18]

3.1 ความผิดพลาดของ stack ภายในระบบ โดยทั่วไปมีสาเหตุมาจากการพยายามอ้างอิงที่อยู่ในหน่วยความจำที่เกินช่วงของ stack segment

3.2 การทำงานคำสั่งที่ไม่ถูกต้อง โดยทั่วไปเกิดจากการพยายามไปเริ่มทำคำสั่งในช่วงที่เป็นข้อมูล หรือการเปลี่ยนแปลงคำสั่งในหน่วยความจำจนทำให้คำสั่งผิด

3.3 การคำนวณที่ได้ผลผิดพลาด โดยเฉพาะการหาร เช่น การหารด้วยศูนย์ เป็นต้น

3.4 ข้อผิดพลาดในการอ้างอิงหน่วยความจำ เช่นการอ้างอิงเกินขอบเขต (เนื่องมาจากการคำนวณตำแหน่งของหน่วยความจำผิดพลาด) หรือ การอ้างอิงหน่วยความจำตำแหน่ง 0 (NULL pointer) หรือ การเขียนในหน่วยความจำที่อนุญาตให้อ่านอย่างเดียว เป็นต้น

ข้อผิดพลาดแบบที่หนึ่งและแบบที่สอง เป็นข้อผิดพลาดจากการขอใช้บริการการจินตทัศน์ที่ถูกตรวจสอบระหว่างการทำงานในหน่วยบริหารจินตทัศน์ ซึ่งจะต้องแจ้งกลับไปยังองค์ประกอบผู้ขอใช้บริการ การแจ้งเหตุการณ์ผิดพลาดดังกล่าวนี้กระทำผ่าน AVISVBX ที่เป็นตัวควบคุมการทำงานที่มีกำกับทุกๆองค์ประกอบการจินตทัศน์

สำหรับข้อผิดพลาดแบบที่สามนั้น เป็นการผิดพลาดที่มิได้คาดมาก่อน เป็นผลให้ข้อมูลที่เก็บภายใน AVISSYNC AVISROUTER และ AVISDB มีค่าไม่ตรงกับสภาพความเป็นจริง เป็นผลให้อาจเกิดการหยุดรอการทำงานขององค์ประกอบที่มีปัญหา เนื่องจากเข้อผิดพลาดแบบนี้ผูกพันกับระบบมาก จึงเป็นหน้าที่ของหน่วยบริหารการจินตท์ทัศน์ในส่วนของ AVISERR ที่ต้องตรวจสอบและจัดการข้อผิดพลาดเป็นการภายใน

3.2 การจัดการข้อผิดผลาดของ AVISVBX

หน่วยบริหารการจินตทัศน์มีหน้าที่ให้บริการแก่ องค์ประกอบการจินตทัศน์ต่างๆ แต่ละองค์ประกอบจะมีตัวควบคุมเฉพาะงานที่ใช้กับวิชวลเบสิก (VISUAL BASIC CUSTOM CONTROL) AVISVBX ทำหน้าที่เชื่อมต่อกับหน่วยบริหารการจินตทัศน์ หน้าที่หนึ่งของ AVISVBX คือการจัดการข้อผิดพลาดที่อาจเกิดขึ้นระหว่างการขอรับบริการการจินตทัศน์ โดย AVISVBX จะเป็นผู้รับแจ้งข้อผิดพลาดจากระบบและส่งทอดต่อไปยังส่วนการแก้ไขหรือจัดการข้อผิดพลาดขององค์ประกอบที่ขอใช้บริการต่อไป

ตามที่ได้กล่าวไว้ข้างต้นแล้วว่า วิชวลเบสิกถูกเลือกเป็นเครื่องมือในการพัฒนาองค์ประกอบการจินตทัศน์ เนื่องจากความง่ายและไม่ซับซ้อนของระบบ ในด้านการจัดการข้อผิด

พลาดนั้นภาษาเบสิกมีกลไกรองรับข้อผิดพลาดที่เกิดขึ้นขณะทำงานได้อย่างดีนั่นคือคำสั่ง **On Error Goto** [17] คำสั่งนี้มีไว้บอกระบบการทำงานว่า หากเมื่อใดเกิดข้อผิดพลาดเกิดขึ้น จะให้ไปทำงานที่ใดในโปรแกรมย่อยที่คำสั่งนี้อยู่ ตัวอย่างเช่นในรูปที่ 3 แสดงโปรแกรมย่อยที่ระบุว่าหากเกิดข้อผิดพลาด จะย้ายการทำงานไปทำที่คำสั่งที่มีชื่อว่า **ErrorSection** (บรรทัดที่ 6) โดยเหตุที่ก่อให้เกิดข้อผิดพลาดจะถูกเก็บไว้เป็นรหัสตัวเลขในตัวแปรระบบชื่อว่า **Err** และข้อความอธิบายความหมายของข้อผิดพลาด จะเก็บไว้ที่ตัวแปรชื่อ **Error$**

```
1 Sub Foo
2   Dim Fname$,I%
3   On Error Goto ErrorSection
4     ...
5   Exit Sub
6 ErrorSection:
7   ... Error Handling Section ...
8 Exit Sub
```

รูปที่ 3 ตัวอย่างการใช้คำสั่ง **On Error Goto**

เมื่อเกิดข้อผิดพลาดระหว่างการขอใช้บริการการจินตทัศน์ใน AVISEXEC ระบบเองจะแจ้งข้อผิดพลาดมายัง AVISVBX จากนั้นตัว AVISVBX จึงแจ้งข้อผิดพลาดนี้ต่อไปให้ระบบการทำงานวิชวลเบสิก (ใช้ฟังก์ชัน **VBRuntimeError** [19]) เป็นผลให้เกิดการย้ายการทำงาน ไปยังส่วนของโปรแกรมที่ประกาศไว้โดยคำสั่ง **On Error Goto** รูปที่ 4 แสดงตัวอย่างองค์ประกอบอัลกอริทึมที่ทำการเรียงลำดับข้อมูลแบบเลือก ระหว่างการเรียงลำดับจะมีการขอใช้บริการจาก AVISEXEC (ในบรรทัดที่ 7 และ 9) ที่อาจก่อให้เกิดข้อผิดพลาดได้ จึงได้ประกาศไว้ที่บรรทัดที่ 3 ว่าหากเกิดข้อผิดพลาดจะย้ายการทำงานไปที่บรรทัดที่ 13 เป็นต้นไป

```
1. Function SelectionSort As Integer
2.   Dim i%, j%
3.   On Error Goto ErrorHandling
4.   For i = DataCount to 2 step -1
5.     j = GetMaxPosition( DataI, i )
6.     SwapData( DataI, i, j )
7.     Call AVisVBOutputNotify(id,SWAP,i,j,"")
8.   next i
9.   Call AVisVBOutputNotify(id,FINISH,0,0,"")
10.  SelectionSort = OK
11.  Exit Function
12.
13.ErrorHandling:
14.  SelectionSort = Err
15.Exit Function
```

รูปที่ 4 ตัวอย่างการควบคุมการทำงานเมื่อเกิดข้อผิดพลาดหลังการใช้บริการของ AVISEXEC

หากเราไม่ใช้คุณสมบัติการจัดการข้อผิดพลาด ดังที่แสดงไว้ข้างต้นนี้ จะทำให้การเขียนโปรแกรมดูลุ่มล่าม เนื่องจากเมื่อใดมีการขอใช้บริการของ AVisExec ก็ต้องคอยตรวจสอบค่าที่คืนจากฟังก์ชันว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องจะได้จัดการกับข้อผิดพลาด นั่นหมายความว่าผู้เขียนโปรแกรมต้องเป็นผู้รับผิดชอบการควบคุมลำดับการทำงานทั้งหมด ดังตัวอย่างในรูปที่ 5 โปรแกรมต้องตรวจสอบข้อผิดพลาดเองทุกๆครั้ง ที่เรียกใช้บริการ (บรรทัดที่ 7 และ 9) หากใช้บริการมากก็ย่อมต้องทดสอบมากตามไปด้วย

```
1.  Function SelectionSort As Integer
2.    Dim i%, j%
3.
4.    For i = DataCount to 2 step -1
5.     j = GetMaxPosition( DataI, i )
6.     SwapData( DataI, i, j )
7.      If  AVisOutputNotify(  id,SWAP,i,j,"")
            <> AVIS_ERR_OK Then Goto Err
8.    next i
9.    If  AVisOutputNotify(  id,FINISH,0,0,"")
            <> AVIS_ERR_OK Then Goto Err
10.   SelectionSort = OK
11.   Exit Function
12.
13. Err:
14.   SelectionSort = ERROR
15. Exit Function
```

รูปที่ 5 ตัวอย่างการเขียนโปรแกรมจัดการข้อผิดพลาดโดยไม่ใช้คำสั่ง **On Error Goto**

จะเห็นได้ว่าการใช้ AVISVBX ประกอบกับคำ**สั่ง On Error Goto** ทำให้โปรแกรมมีความกระชับและชัดเจนกว่าการใช้บริการของ AVISEXEC และตรวจสอบข้อผิดพลาดโดยตรง

3.3 การจัดการข้อผิดผลาดของ AVISERR

หน้าที่ประการหนึ่งของ AVISERR คือการคอยตรวจสอบว่ามีองค์ประกอบการจินตทัศน์ใดบ้างในระบบที่หยุดการทำงานอย่างผิดปกติ เพื่อจะได้แก้ไขข้อผิดพลาดดังกล่าว เทคนิคการตรวจสอบข้อผิดพลาดชนิดนี้มีอยู่สามวิธีคือ

1 *การดักการขัดจังหวะหลังเกิดข้อผิดพลาด* โดยทั่วไปเมื่อเกิดข้อผิดผลาดขึ้นจนทำให้โปรแกรมใดๆ หยุดทำงานแบบผิดปกติ จะทำให้เกิดการส่งสัญญาณขัดจังหวะขึ้นภายในระบบปฏิบัติการ ในระบบปฏิบัติการบางระบบนั้น จะยินยอมให้ผู้เขียนโปรแกรมระบบ เพิ่ม (hook) ฟังก์ชันพิเศษเข้าสู่ระบบปฏิบัติการ เพื่อที่ระบบปฏิบัติการจะเรียกฟังก์ชันนี้ทุกๆครั้งที่เกิดการขัดจังหวะแบบนี้ขึ้น ฟังก์ชันนี้ก็คือฟังก์ชันการจัดการข้อผิดพลาดหลังจากตรวจพบข้อผิดพลาด

2 *การตรวจสอบข้อผิดพลาดของระบบเป็นระยะๆ* วิธีนี้กระทำโดยการกำหนดให้ฟังก์ชันตรวจสอบและจัดการข้อผิดพลาดถูกเรียกใช้ตรวจสอบระบบเป็นระยะๆ เช่นทุกๆ 1 วินาที เป็นต้น การทำงานนี้ก็เพียงแต่ตรวจสอบว่าแต่ละองค์ประกอบที่เก็บไว้ใน AVISDB นั้นยังคงทำงานอยู่ในระบบ

หรือไม่ ในกรณีที่องค์ประกอบใดที่มีทะเบียนในระบบ แต่ไม่ปรากฏว่ากำลังทำงานอยู่ ก็แสดงว่าเกิดข้อผิดพลาดขึ้น ข้อผิดพลาดที่ตรวจพบก็จะถูกจัดการให้ระบบมีการทำงานตามปกติ

3 การผสมการดักการขัดจังหวะและการตรวจสอบเป็นระยะๆ วิธีนี้เราจะนำการทำงานของวิธีที่หนึ่งและสองมารวมกันเพื่อขจัดข้อด้อยของทั้งสองวิธี กล่าวคือองค์ประกอบที่ทำงานผิดพลาดจะถูกดักพบด้วยวิธีแรกและบันทึกข้อมูลไว้ในระบบ โดยจะยังไม่จัดการกับข้อผิดพลาดที่พบนั้น แล้วจึงใช้วิธีที่สองมาตรวจดูผลการตรวจสอบเป็นระยะๆ เมื่อพบข้อมูลที่บันทึกไว้ว่าเกิดข้อผิดพลาด จึงจะจัดการกับข้อผิดพลาดเหล่านั้น

การตรวจสอบข้อผิดพลาดแบบแรกนั้น จะไม่มีผลต่อความเร็วในการทำงานตามปกติของระบบเลย เพราะได้ผลักภาระการตรวจสอบไปให้ระบบปฏิบัติการ เมื่อมีข้อผิดพลาดจึงแจ้งให้ส่วนจัดการข้อผิดพลาดทราบ แต่การเพิ่มเสริมฟังก์ชันใหม่ให้กับระบบปฏิบัติการนั้นเป็นเรื่องยุ่งยาก และบั่นทอนความมั่นคงของระบบ อีกทั้งในทางปฏิบัติการสร้างกลไกในลักษณะนี้จะขึ้นกับระบบปฏิบัติการมาก การจัดการกับข้อผิดพลาดในช่วงที่ตรวจสอบพบข้อผิดพลาดทันทีนั้น เป็นช่วงที่ระบบไม่ค่อยมีเสถียรภาพมาก จึงอาจจำกัดการขอใช้บริการของระบบปฏิบัติการในการดำเนินการจัดการข้อผิดพลาดในระบบได้

สำหรับวิธีการตรวจสอบและจัดการแบบที่สองนั้น จะมีการทำงานที่ง่าย ไม่ซับซ้อน อีกทั้งแนวคิดนี้สามารถนำไปใช้ได้กับระบบปฏิบัติการโดยทั่วไป แต่ก็มีข้อเสียคือการตรวจสอบจะต้องตรวจสอบทุกๆองค์ประกอบที่มีทะเบียนในระบบ เพื่อหาเฉพาะองค์ประกอบที่ทำงานผิดพลาด ทำให้เสียเวลาและอาจลดประสิทธิภาพการทำงานของระบบลงหากตั้งช่วงการตรวจสอบถี่เกินไป

สำหรับการตรวจสอบและจัดการข้อผิดพลาดในแบบที่สามนั้น ได้นำข้อดีของสองแบบเสริมกัน แต่ระบบนี้ก็ยังมีข้อเสียเหมือนวิธีแรกก็คือความยุ่งยากในการเพิ่มฟังก์ชันพิเศษเข้าไปเชื่อมกับระบบปฏิบัติการ

ส่วนจัดการข้อผิดพลาด AVISERR ในงานวิจัยนี้เลือกใช้การตรวจสอบข้อผิดพลาดแบบที่สอง โดยทำการตรวจสอบระบบทุกๆ 1 วินาที ซึ่งในทางปฏิบัติการแล้วจะไม่กระทบประสิทธิภาพการทำงานของระบบ ทั้งนี้เนื่องจากเวลาส่วนใหญ่ที่เสียไปในการจินตทัศน์ส่วนใหญ่จะอยู่ที่ส่วนแสดงผลในองค์ประกอบมุมมอง (View)

3.3.1 การแก้ไขข้อผิดผลาดของ AVISERR

หลังจากที่เรารับรู้ว่าองค์ประกอบหยุดทำงานอย่างผิดปกติ หน้าที่ต่อไปของ AVISERR ก็คือการแจ้งเตือนไปให้ส่วนอื่นๆ ของระบบจินตทัศน์อันได้แก่ AVISROUTER AVISSYNC AVISSESSIONCONTROL และ AVISDB ทราบตามลำดับ ภพืศอ ทำการมกษฤขษอมูลรนสศวนทีศตัวภองดูมลอยูศ ดังนีษ

1 AVISROUTER การส่งข้อความคำสั่งระหว่างองค์ประกอบที่ใช้ในระบบจินตทัศน์ใช้กลไกการสัศงการมบบขนาน [3] ยดยผูษสศงจะตษองรอจนกวศาผูษรับทุกรายตอบกลับมาวศาทำงานภสรวจมลษว จึงจะสามารถทำงานต่อไปได้ แต่ถ้าหากผู้รับข้อความหยุดการทำงานอย่างกระทันหันโดยไม่ตอบรับข้อความกลับ ผู้ส่งก็จะต้องหยุดรออย่างไม่มีที่สิ้นสุด ดังนั้น เมื่อ AVISROUTER ฤดษรับมจษงมาจาก AVISERR ถึงภหตุการณพดังกลศาว จะตัดขษอมูลภายรนทีศภกีศยวขษองกับองคพประกอบปัญหาดังกลศาว ภพืศอจะฤดษฤมศตษองรอองคพประกอบนัษน จากนัษนจะคืนคศามสดงความผิดพลาดดังกลศาวรหษกับองคพประกอบผูษขอรชษบริการสศงขษอความคำสัศงทราบตศอฤป

2 AVISSYNC ระบบจินตทัศน์อัลกอริทึมนี้อนุญาตให้มีการจินตทัศน์ได้มากกว่าหนึ่งอัลกอริทึม แต่ละอัลกอริทึมจะใช้บริการการประสานจังหวะที่ AVISSYNC โดยมีการแบ่งเวลาการทำงานแบบ round robin เมื่อองค์ประกอบอัลกอริทึมหนึ่งมีปัญหาและหยุดการทำงานอย่างผิดปกติ ก่อนที่จะแจ้งให้ AVISSYNC ทราบว่าตัวเองยินยอมให้องค์ประกอบถัดไป ทำงานต่อได้ เป็นผลให้ AVISSYNC รออย่างไม่สิ้นสุด ทำให้องค์ประกอบอัลกอริทึมอื่นไม่มีโอกาสทำงาน ดูภสมือนวศาระบบหยุดการทำงานฤปทัษงหมด ดังนั้น เมื่อ AVISSYNC ฤดษรับมจษงมาจาก AVISERR ถึงภหตุการณพดังกลศาว จะตัดขษอมูลภายรนทีศภกีศยวขษองกับองคพประกอบปัญหาดังกลศาว ภพืศอจะฤดษฤมศตษองรออัลกอริทึมนัษน มลษวปลศอยรหษอัลกอริทึมอืศนๆทำงานตศอฤป

3 AVISSESSIONCONTROL หลังจาก AVISERR แจ้งเตือนไปให้AVISROUTER และ AVISSYNC แล้ว ในช่วงนี้ระบบจะอยู่ในสภาวะที่มีเสถียรภาพพอที่จะแจ้งให้ AVISSESSIONCONTROL ซืศงภปานสศวนประกอบภายนอกของ AVISEXEC ทราบ ภพืศอมจษงขษอผิดผลาดทีศภกิดขึษนกับผูษรชษ มละทำการปรับภปลีศยนสภาพการทำงานของระบบจินตทัศน์อัลกอริทึมให้เหมาะสมกับเหตุการที่เกิดขึ้นต่อไป

4 AVISDB จะเป็นส่วนสุดท้ายของ AVISEXEC ที่จะได้รับการแจ้งเตือน ทั้งนี้เพราะการแก้ไขข้อผิดผลาดของ AVISDB จะทำโดยการลบข้อมูล หรือทำการถอนทะเบียนขององค์ประ

กอบที่หยุดทำงานออกไปจากระบบ ซึ่งข้อมูลเหล่านี้อาจมีความจำเป็นต่อการแก้ไขข้อผิดพลาดของส่วนประกอบสามส่วนแรก จึงต้องแจ้งเตือนให้ทั้งสามส่วนนั้นทำการแก้ไขก่อนแล้วจึงแจ้งเตือนให้ AVISDB เป็นขั้นตอนสุดท้าย

4. สรุป

บทความนี้ได้นำเสนอวิธีการตรวจสอบ แก้ไข และจัดการข้อผิดพลาด ซึ่งอาจเกิดขึ้นได้ในขณะที่ระบบจินตทัศน์อัลกอริทึมกำลังทำงาน ข้อผิดพลาดเหล่านี้ประกอบด้วยข้อผิดพลาดระหว่างการขอใช้บริการการจินตทัศน์ ซึ่งเป็นหน้าที่ของ AVISVBX (อันเป็นส่วนเชื่อมโยงส่งทอดข้อผิดพลาดที่เกิดขึ้นไปยังระบบการทำงานวิชวลเบสิก) กับการใช้คำสั่ง **ON ERROR GOTO** เพื่อควบคุมลำดับการทำงาน ข้อผิดพลาดอีกประการหนึ่งคือเกิดจากการหยุดการทำงานอย่างผิดปกติขององค์ประกอบที่ทำงานในระบบ ข้อผิดพลาดนี้ถูกจัดการได้โดยการตรวจสอบเหตุการณ์ดังกล่าวในระบบเป็นระยะๆทุกๆ 1 วินาที เพื่อจะได้ปรับข้อมูลภายในระบบให้ตรงกับความเป็นจริงและแจ้งให้ผู้เกี่ยวข้องกับเหตุการณ์ดังกล่าวทราบ ด้วยกลไกการจัดการข้อผิดพลาดดังกล่าวทำให้ระบบสามารถแก้ไขความผิดผลาดเหล่านี้ได้ด้วยตัวเอง พร้อมที่จะทำงานได้อย่างต่อเนื่อง โดยมีผลกระทบต่อผู้ใช้น้อยที่สุด

เอกสารอ้างอิง

[1] สมชาย ประสิทธิ์จูตระกูล และ ชัชวาล วงศ์ศิริประเสริฐ, "โครงสร้างของหน่วยบริหารการจินตทัศน์อัลกอริทึม," จะตีพิมพ์เผยแพร่ในการประชุมทางวิชาการ Electro Technology'95 วิศวกรรมสถานแห่งประเทศไทย, สิงหาคม 2538

[2] S. Prasitjutrakul and W. Watcharawittayakul, "Algorithm Visualization : a Revisit to Sorting Algorithm," *Thai Journal of Development Administration*, Vol. 33, No. 2., pp. 193-201., April-June 1993.

[3] S. Prasitjutrakul and W. Watcharawittayakul, "Algorithm Visualization System : An Overview of Internal Structure," *Proc. of Third ASEAN Regional Seminar on Microelectronics and Information Technology*, Bangkok, August 1994.

[4] สมชาย ประสิทธิ์จูตระกูล และ วิทยา วัชระวิทยากุล, "ระบบจินตทัศน์อัลกอริธึมภายใต้สภาพปฏิบัติการไมโครซอฟต์วินโดวส์," การประชุมทางวิชาการคอมพิวเตอร์ 2537

[5] Kenneth C. Knowlton. L6: "Bell Telephone Laboratories Low-Level Linked List Language," two 16mm black and white sound film,1966

[6] Kellog S. Booth, "PQ Trees," 16mm color silent film,12 minutes,1975

[7] Ronal M. Baecker and David Sherman, "Sorting Out Sorting," 16mm color sound film, 30 minutes,1981

[8] Brad A. Myers, "Incense: A System for Displaying Data Structures," *Computer Graphics*, Vol 17, No.3,115-125, July1983

[9] David B. Baskerville, "Graphic Presentation of Data Structure in the DBX Debugger," Report No.UCB/CSD 86/260, University of California at Berkeley, Berkeley, CA, October 1985

[10] Thomas G. Moher, "PROVIDE: a Process Visualization and Debugging Environment," Technical Report, University of Illinois at Chicago, Chicago, IL July 1985

[11] Edward Yarwood, "Toward Program Illustration," M.Sc. Thesis, Dept. of Computer Science, University of Toronto, Toronto, ON, 1974.

[12] James M. De Boer, "A System for the Animation of Micro-PL/I Programs," M.Sc. Thesis, Dept. of Computer Science, University of Torinto, Toronto, ON, 1974.

[13] Marc H. Brown, *Algorithm Animation*, The MIT Press, Cambridge, MA, 1988

[14] Marc H. Brown, "Zeus: A System for Algorithm Animation and Multi View Editing," *Proc. IEEE Workshop Visual Language*, IEEE CS Press, CA, pp. 27-39, 1991.

[15] Stasko, "Tango: A Framwork and System for Algorithm Animation," *Computer*,Vol. 23,No. ,pp.27-399, Sept. 1990.

[16] Roman et al., "Pavene: A System for Declarative Visualization of Concurrent Computations," *J. Visual Language and Computing*, Vol. 3, No. 2,pp. 161-163,June 1992.

[17] Microsoft, *Microsoft Visual Basic Programming System for Windows Programmer's Guide version 3.0*, Microsoft Inc., 1993.

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

[19] Microsoft, *Microsoft Visual Basic Programming System for Windows Professional Feature Book I*, Microsoft Inc., 1993.