การพัฒนาโปรแกรมประยุกต์โดยใช้ TAPI และ SAPI

Application Development with TAPI and SAPI

นาย สุเมธ จิตตประวัติ
นาย สุรชัย ล้อเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

การพัฒนาโปรแกรมประยุกต์โดยใช้ TAPI และ SAPI

Application Development with TAPI and SAPI

โดย

นาย สุเมธ จิตตประวัติ

นาย สุรชัย ล้อเจริญ

อาจารย์ที่ปรึกษา

อ. ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมโดยใช้ TAPI และ SAPI

Application using TAPI and SAPI

ผู้จัดทำ

1. นาย สุเมธ จิตตประวัติ รหัสประจำตัว 41014484
2. นาย สุรชัย ล้อเจริญ รหัสประจำตัว 41014485

ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา

(ดร. ชุติเมษฎ์ ศรีนิลทา)

การพัฒนาโปรแกรมโดยใช้ SAPI และ TAPI

นาย สุเมธ จิตตประวัติ 41014484
นาย สุรชัย ล้อเจริญ 41014485
ดร. ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

โครงงานนี้มุ่งเน้นที่จะพัฒนาระบบเพื่ออำนวยความสะดวกในการใช้งานอีเมล์ ผู้ใช้ไม่มีความจำเป็นต้องยึดติดกับวิธีการตรวจสอบอีเมล์แบบเดิมที่ต้องนั่งอยู่หน้าคอมพิวเตอร์แต่สามารถตรวจสอบอีเมล์ผ่านทางโทรศัพท์ และจะมีบริการแจ้งการมีอีเมล์ใหม่ผ่านโทรศัพท์มือถือ โดยนำเทคโนโลยี Text to Speech (TTS), Computer Telephony Integration (CTI), Java Server Page (JSP), Post Office Protocol version 3 (POP3) และ Short Message Service (SMS) มารวมกัน โครงงานนี้ได้สร้างเอนจินสำหรับแปลงข้อความภาษาไทยเป็นเสียงโดยใช้อินเตอร์เฟซของ Speech Application Programming Interface (SAPI) ผนวกเข้ากับ Telephony Application Programming Interface (TAPI) มาอิมพลีเมนต์ในส่วนของการเชื่อมต่อระหว่างคอมพิวเตอร์กับโทรศัพท์

และเพื่อให้ระบบมีความสมบูรณ์มากขึ้นจึงได้สร้างเว็บไซท์ขึ้นเพื่อเพิ่มช่องทางในการติดต่อเพื่อใช้บริการ โดยใช้ Servlet และ JSP ซึ่งมีประสิทธิภาพสูงในการสร้างเว็บแอพพลิเคชั่น และยังมีการอำนวยความสะดวกในการแจ้งเตือนอีเมล์ที่มาใหม่ผ่านบริการส่งข้อความเข้าโทรศัพท์มือถือ ทำให้ระบบสามารถตอบสนองความต้องการของผู้ใช้ได้อย่างเหมาะสม

Application Development with TAPI and SAPI

Mr Sumeth Chittapawat
Mr Surachai Locharoen
Dr. Chutimet Srinilta Advisor

ABSTRACT

The purpose of this project is to build an EmailPhone System that facilitates user to access email on mail server. There is no need to connect to the internet directly for checking email as old fashion. Users can only dial to certain telephone number and then listen to there emails. It has integrated Text to speech (TTS), Computer Telephony Integration (CTI), Post office protocol version 3 (POP3) and Short Message Service (SMS) in to one system. System implements Thai TTS Engine to support in this system and use TAPI to enable computer to communicate with telephone network.

Additionally, system offers the way to interact with user by web application, which uses JSP and Servlet. System can alert email incoming to the user by short message service. This system is another solution to access email.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้ สำเร็จลงด้วยดีได้ เพราะความกรุณาจากอาจารย์ต่างๆ ที่ให้การอบรมสั่งสอนวิชาความรู้ตั้งแต่ต้น จนถึงปัจจุบัน โดยเฉพาะอย่างยิ่ง อาจารย์ที่ปรึกษา อาจารย์ชุติเมษฎ์ ศรีนิลทา ที่คอยให้คำปรึกษา พร้อมทั้งแนวคิด ข้อเสนอแนะ ตลอดจนการแก้ไขบกพร่องต่างๆ โดยตลอด ผู้จัดทำรู้สึกซาบซึ้ง และขอกราบขอบพระคุณอย่างสูง มา ณ โอกาสนี้

ขอกราบขอบพระคุณ บิดา มารดา ที่คอยอบรมสั่งสอน รับฟังปัญหาต่างๆ และทุ่มเทกำลังกาย กำลังใจในการให้การศึกษาแก่ผู้จัดทำ และให้กำลังใจมาเป็นอย่างดีโดยตลอด ผู้จัดทำขอกราบขอบพระคุณมา ณ โอกาสนี้ด้วย

นาย สุเมธ จิตตประวัติ

นาย สุรชัย ล้อเจริญ

สารบัญ

สารบัญตาราง

หน้าที่

สารบัญรูปภาพ

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญ และที่มา

ปัจจุบันมีการนำเอาเสียงพูด (Human Voice) มาใช้ให้เกิดประโยชน์ในคอมพิวเตอร์มากขึ้น สาเหตุที่มีการพัฒนาเทคโนโลยีด้านนี้ก็เพื่อความสะดวกสบายโดยเฉพาะ เพราะแทนที่เราจะต้องมาเสียเวลาใช้เมาส์หรือแป้นพิมพ์ในการควบคุมคอมพิวเตอร์เราก็ใช้เสียงของเราแทนในการสั่งงานให้คอมพิวเตอร์ทำงานตามที่เราต้องการได้ หรือแทนที่เราจะต้องอ่านข้อความที่มีอยู่เราก็สามารถให้เครื่องคอมพิวเตอร์อ่านออกเสียงแทนได้ โดยเทคโนโลยีที่เกี่ยวข้องกับเสียงพูดนี้แบ่งการศึกษาออกเป็น 2 เรื่องใหญ่ ๆ คือ การจดจำเสียง (Speech Recognition หรือ SR) และการแปลงข้อความให้เป็นเสียง (Text to Speech หรือ TTS) โดยในแต่ละเรื่องสามารถสร้างแอพพลิเคชั่นได้มากมายแยกย่อยได้อีกหลายอย่าง ตัวอย่างแอพพลิเคชั่นสำหรับการจดจำเสียง ก็ได้แก่ การสั่งงานคอมพิวเตอร์ด้วยเสียง (Voice Command), การเขียนตามคำบอก (Dictation) เป็นต้น ส่วนการแปลงข้อความให้เป็นเสียงนั้นเหมาะกับแอพพลิเคชั่นที่มีข้อความอยู่แล้วและต้องการให้อ่านออกเสียงออกมา เช่น แอพพลิเคชั่นสำหรับคนพิการพูดไม่ได้แต่สามารถกดแป้นพิมพ์ได้ การส่งข้อความผ่านอินเตอร์เน็ทแล้วให้แปลงเป็นเสียง หรือแม้แต่แอพพลิเคชั่นสำหรับช่วยในการอ่านออกเสียง เป็นต้น

สำหรับประเทศไทยสาเหตุที่ไม่ค่อยมีผู้ให้ความสนใจเทคโนโลยีด้านนี้เป็นเพราะส่วนใหญ่แอพพลิเคชั่นที่มี สร้างขึ้นมาเพื่อใช้เฉพาะกับบางภาษาเท่านั้นส่วนใหญ่มีเฉพาะภาษาอังกฤษ ซึ่งไม่เหมาะกับคนไทยเป็นอย่างมากเพราะผู้ใช้ส่วนใหญ่มีข้อจำกัดทางด้านภาษาเนื่องจากไม่ได้เป็นเจ้าของภาษาอีกทั้งยังไม่ได้ใช้ในชีวิตประจำวัน ทำให้เมื่อใช้งานแล้วไม่รู้สึกสะดวกสบายแถมยังรู้สึกลำบากมากขึ้นเพราะติดขัดเรื่องของภาษา ด้วยเหตุนี้จึงทำให้มีผู้พัฒนาเทคโนโลยีด้านนี้ในประเทศค่อนข้างจำกัดเพราะเทคโนโลยีด้านนี้ไม่ได้รับความสนใจเท่าที่ควร เพราะฉะนั้นการที่จะทำให้เทคโนโลยีนี้เกิดประโยชน์สำหรับคนไทยได้นั้นจึงควรที่จะต้องมีการพัฒนาให้ใช้กับภาษาไทยได้

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาเทคโนโลยีที่เกี่ยวข้องกับการทำ Text to Speech สำหรับภาษาไทยและทฤษฎีที่เกี่ยวข้อง โดยใช้ SAPI 5.0 ของไมโครซอฟท์เป็นหลัก ศึกษาว่า SAPI มีสถาปัตยกรรมอย่างไร มีอินเทอร์เฟซอะไรบ้างที่ใช้สำหรับการทำ Text to Speech และสามารถนำไปประยุกต์ใช้งานได้อย่างไรบ้าง

1.2.2 ศึกษาการเขียนโปรแกรมโดยใช้ API (Application Programming Interface) ว่าแตกต่างจากการเขียนโปรแกรมแบบทั่วไปอย่างไร และช่วยให้สามารถพัฒนาโปรแกรมให้มีประสิทธิภาพได้อย่างไร รวมถึงมีข้อดีข้อเสียอย่างไร

1.2.3 ศึกษาเทคโนโลยีอื่น ๆ ที่ใช้ในการพัฒนาแอพลิเคชันเช่น การส่ง SMS (Short Message Service) และ JavaMail เป็นต้น

1.2.3 นำเอ็นจินที่สร้างขึ้นมาสร้างแอพลิเคชันสำหรับอำนวยความสะดวกในการตรวจเช็คอีเมล์

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะพัฒนาแอพพลิเคชั่นส่วนที่เป็น Text to Speech Engine ภาษาไทย โดยใช้คอมไพล์เลอร์ Microsoft Visual C++, Microsoft Speech SDK ของ SAPI 5.0 และ Microsoft Platform SDK ในการพัฒนาการโปรแกรม

นำเอ็นจินที่ได้มาใช้งานร่วมกับ TAPI เพื่อสร้างแอพลิเคชันสำหรับเช็คอีเมล์ผ่านทางโทรศัพท์ ทั้งนี้ได้เพิ่มบริการส่ง SMS โดยเลือกส่ง SMS ผ่านทาง ICQ และสร้างเวปแอพลิเคชันสำหรับเช็คอีเมล์อีกทางหนึ่ง

1.4 วิธีการดำเนินงาน

โครงงานนี้เริ่มจากการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องในการสร้างแอพพลิเคชั่น Text to Speech เพื่อนำมาใช้กับ SAPI ในส่วนของเอ็นจินภาษาไทย จากนั้นนำความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบสถาปัตยกรรมของระบบ Text to Speech System จากนั้นเป็นขั้นตอนของการพัฒนาโปรแกรม ซึ่งกล่าวถึงองค์ประกอบโดยรวมของระบบที่พัฒนาขึ้นมาทั้งหมด และยังอธิบายไปถึงรูปแบบการติดต่อกับผู้ใช้ และการประมวลผลต่าง ๆ ซึ่งจะอธิบายแต่ละฟังก์ชันการทำงาน

ท้ายสุดจะเป็นการทำการทดสอบระบบโดยรวมที่สร้างขึ้น และเป็นการสรุปการทำงาน ผลที่ได้รับจากโครงงานนี้ และแนวทางในการพัฒนาโครงงานนี้เพิ่มเติม รวมถึงแนวทางในการนำแอพพลิเคชั่นที่สร้างขึ้นไปประยุกต์ใช้งานจริง

บทที่ 2

The Component Object Model (COM)

2.1 บทนำ

COM เป็นรูปแบบการเขียนโปรแกรมแบบหนึ่ง ซึ่งเป็นพื้นฐานสำหรับ Microsoft ActiveX Technology, Automation และ OLE ในอนาคตการเขียนโปรแกรมบน Microsoft Windows จะใช้ COM เราจึงควรทำความเข้าใจหลักการทำงานของ COMให้ดีก่อนจึงจะสามารถประยุกต์ใช้งานได้

COM เป็นมาตรฐานในทางอุตสาหกรรมของซอฟท์แวร์ที่ซัพพอร์ตโดยบริษัทชั้นนำ เช่น Microsoft, Digital Equipment Corporation และบริษัทอื่นมากมาย แต่ COMไม่ได้เป็นมาตรฐานเดียวที่มีอยู่ในปัจจุบัน ยังมี Corba ของ Open Software Foundation (OSF) อีกด้วย

2.2 COM ช่วยแก้ปัญหา

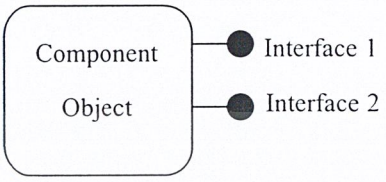
ลองนึกถึงการเขียนโปรแกรมแบบเก่า เช่น Windows API ที่มีฟังก์ชันให้เลือกใช้มากกว่า 350 อัน จะเรียกใช้งานแต่ละครั้งต้องส่งค่าพารามิเตอร์กันมากมาย อาจทำให้เกิดความสับสนได้ แต่ปัญหาเหล่านี้สามารถแก้ไขได้โดยการเขียนโปรแกรมแบบ COM

COM ทำให้การเขียนโปรแกรมบน Windows ง่ายขึ้น เพราะจะจัดการทำให้โปรแกรมมีลักษณะเป็นโมดูล ช่วยลดการส่งค่าพารามิเตอร์ ทำให้สามารถพัฒนาซอพท์แวร์ขนาดใหญ่ได้อย่างต่อเนื่อง COM สามารถเขียนได้หลายภาษา เช่น C/C++, JAVA, Visual Basic, Delphi ในการแก้ไขโปรแกรมส่วนใดส่วนหนึ่งก็ไม่มีผลกระทบต่ออีกส่วนหนึ่ง เพราะมันมีการแยกส่วนการอิมพลีเมนต์และส่วนอินเทอร์เฟซออกจากกันอย่างชัดเจน

2.3 หลักการพื้นฐานของ COM

แนวคิดของ COM มีลักษณะเหมือนกับ Object Oriented Programming (OOP) คือมองปัญหาเป็นวัตถุ ซึ่งมีคุณสมบัติ และ ความสามารถประจำตัว นอกจากนี้ COM ยังมีความสามารถทางเทคนิคอีกหลายประการ เช่น สามารถประมวลผลโปรแกรมที่แยกส่วนกันได้ (Integrate software at runtime), จัดการเรื่องเธรด (Thread), สามารถประมวลผลโปรแกรมข้ามเครื่องกัน (Remote Procedure Call) เป็นต้น

COM แบ่งส่วนของโปรแกรมออกเป็นคอมโพเน็นต์ออบเจ็กต์ (Component Object) ซึ่งคอมโพเน็นต์ออบเจ็กต์แต่ละตัวสามารถติดต่อถึงกันผ่านทางอินเทอร์เฟซ โดยที่ในคอมโพเน็นต์ออบเจ็กต์หนึ่งตัวสามารถมีได้หลายอินเทอร์เฟซ



รูปที่ 2-1 Componet Object Model (COM) พื้นฐาน

2.4 อินเทอร์เฟซ (Interface)

อินเทอร์เฟซ คือ กลุ่มของเมธอดที่ไม่มีการอิมพลีเมนต์ (เป็น Virtual Function) เป็นส่วนที่ใช้ติดต่อกับคอมโพเน็นต์ออบเจ็กต์อื่นๆ เพื่อที่จะทำความเข้าใจความหมายของอินเทอร์เฟซ ลองนึกถึงเกมยานอวกาศ เราสร้างคอมโพเน็นต์ออบเจ็กต์ CSpaceship ขึ้นมา ซึ่งมีอินเตอร์เฟซอยู่ 2 อัน คือ IMotion และ IVisual ซึ่งเมื่อต้องการนำคอมโพเน็นต์นี้ไปใช้ก็เรียกใช้อินเตอร์เฟซที่ต้องการได้ และในภายหลังถ้าต้องการจะเพิ่มอินเตอร์เฟซก็ทำได้โดยไม่มีผลกระทบต่ออินเตอร์เฟซที่มีอยู่ก่อนหน้านี้ เพื่อให้ง่ายต่อการเข้าใจเราจึงเขียนในลักษณะที่เลี่ยงการใช้มาโคร โดยให้มี 2 อินเทอร์เฟซ คือ IMotion และ IVisual ดังนี้

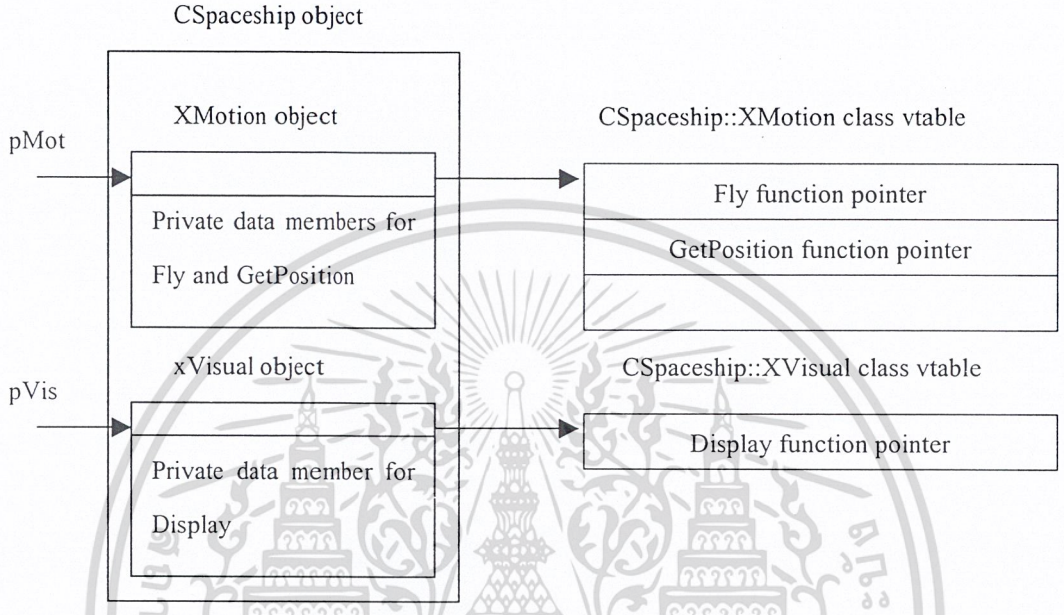
```
    Struct IMotion
    {
    virtual void Fly() = 0;
        virtual int& GetPosition() = 0;
    };
Struct IVisual
{
        virtual void Display() = 0;
};
Clas Cspaceship
{
        protected:
```

```
        int m_nPosition;
        int m_nAcceleration;
        int m_nColor;
        public:
        Cspaceship()
        {
                M_nPosition = m_nAcceleration = m_nColor = 0;
        }
        class Xmotion : public IMotion
        {
                public:
                XMotion() {}
                virtual void Fly();
                virtual int& GetPosition();
        } m_xMotion;
        class Xvisual : public Ivisual
        {
                public
                XVisual() {}
                virtual void Display
        } m_xVisual;
        friend class XVisual;
        friend class XMotion;
};
```

สามารถเรียกใช้เขียนดังนี้

```
IMotion* pMot;
IVisual* pVis;
// some code for get interface pointer
pMot->Fly();
    pVis->Display();
```

จากตัวอย่าง pMot และ pVis จะมีเมทธอดที่ไม่เหมือนกัน pMot จะเรียกใช้ ได้เฉพาะ Fly() และ GetPosition() ส่วน pVis จะเรียกใช้ได้แต่ Display() แต่ทั้งสองชี้ไปยังคอมโพเนนต์เดียวกันแต่คนละอินเตอร์เฟซกันเท่านั้นเอง



รูปที่ 2-2 Memory Layout ของ COM

ที่แสดงให้ดูในตัวอย่างนี้เป็นวิธีการสร้างคอมโพเนนต์แบบ Nested Class คือมี class อยู่ใน class อีกต่อหนึ่ง ยังมีวิธีการสร้างคอมโพเนนต์แบบหนึ่งคือ Multiple Inheritance ที่ใช้วิธีสืบถอด class มาจากหลาย ๆ อินเตอร์เฟซ

รูปแบบการเขียน COM แบบ Nested Class มีข้อดีกว่า Multiple Inheritance ดังนี้

1. Nested class สามารถ acces data member ของ parent ได้
2. บ่งบอกอย่างชัดเจนว่าอินเตอร์เฟซใดบ้างที่คอมโพเนนต์มี

แต่ที่แสดงให้เห็นยังไม่ใช่คอมโพเน็นต์ออบเจ็กต์ที่สมบูรณ์แบบเพราะยังขาดกลไกที่จัดการในการ Query Interface และช่วงชีวิตของคอมโพเน็นต์ออบเจ็กต์นั้นว่าควรอยู่นานเท่าไร

2.5 IUnknown Interface

จากตัวอย่างที่ผ่านมายังมีปัญหาอยู่ว่าจะรู้ค่าพอยน์เตอร์ของ pMot และ pVis ได้อย่างไร ปัญหานี้แก้ได้ด้วยอินเทอร์เฟซ IUnknown ซึ่งเป็นอินเทอร์เฟซพื้นฐาน ที่ถูก Inherit มาเฉพาะอินเทอร์เฟซ ทำให้ทุก ๆ อินเทอร์เฟซจะมีเมธอด ดังนี้

```
Struct IUnknown
{
        virtual BOOL QueryInterface(int nIid, void** ppvObj)=0;
        virtual DWORD AddRef() = 0;
        virtual DWORD Release() = 0;
};
```

Method	Description
QueryInterface(int nIid, void** ppvObj)	เป็น method ที่จะให้ค่าพอยน์เตอร์ที่ชี้ไปยังอินเทอร์เฟซ ที่มี Interface ID ตรงกับ nIid
AddRef()	เพิ่มค่า reference count
Release()	ลดค่า reference count

ตารางที่ 2-1 แสดงเมธอดของ IUnknown

ในการเขียนโปรแกรมแบบ COM ทั้ง Class และ อินเทอร์เฟซถูกอ้างถึงด้วย globally unique identifier (GUID) เป็นตัวเลข 128 บิต ซึ่งมีโปรแกรมสร้างให้ และจะถูกเก็บไว้ใน Registry

Reference count คือค่าที่แสดงว่าคอมโพเน็นต์ออบเจ็กต์ถูกอ้างจาก Interface Pointer เป็นจำนวนเท่าไร ถ้าค่านี้มีค่าเป็น 0 คอมโพเน็นต์ออบเจ็กต์ตัวนั้นก็จะถูกทำลาย

กลับมาที่ตัวอย่างอีกครั้งเพื่อให้ IMotion และ IVisual ได้ interface inherit มาจาก IUnknow จะได้โค้ดดังนี้ข้อแตกต่าง ระหว่าง interface inherit และ implement inherit คือ interface inherit จะต้อง implement ใหม่ทุกครั้งที่มีการ inherit

```
Struct IMotion : public IUnknown
{
        virtual void Fly() = 0;
        virtual int& GetPosition() = 0;
};
struct IVisual : public IUnknown
{
        virtual void Display() = 0;
    };
```

ภายใน class CSpaceship จะต้องอิมพลีเมนต์ Query Interface, Addref, Release ใหม่ทั้งหมด ตัวอย่างการอิมพลีเมนต์

```
BOOL CSpaceship::XMotion::QueryInterface(int nIid, void** ppvObj)
{
        // pThis เป็นตัวแปรที่ชี้ไปที่ parent class
        METHOD_PROLOGUE(CSpaceship, Motion)
        Switch (nIid)
        {
                Case IID_IUnknown:
                Case IID_IMotion:
                        *ppvObj = &pThis->m_xMotion;
                        break;
                Case IID_IVisual:
                        *ppvObj = &pThis->m_xVisual;
                        break;
                Default:
                        *ppvobj = null
                        return FALSE;
        }
        return TRUE;
}
```

ในการอิมพลีเมนต์ Addref และ Release เราต้องเพิ่ม m_dwRef ไว้ใน Class CSpaceship

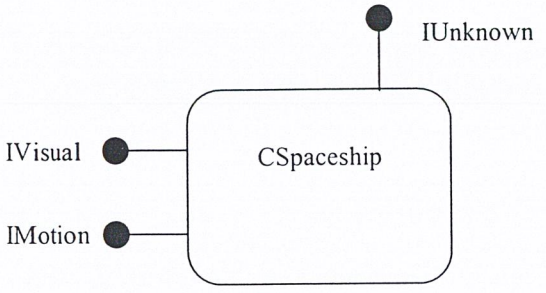
```
DWORD CSpaceship::XMotion::Release()
{
        METHOD_PROLOGUE(CSpaceship, Motion) // make pThis
        If (pThis->m_dwRef == 0)
                Return 0;
        If (--pThis->m_dwRef == 0)
        {
                Delete pThis;
                Return 0;
        }
        return pThis->m_dwRef;
}
```

เมื่อไคล์เอ็นต์ต้องการเรียกใช้คอมก็สามารถเรียกใช้ได้ดังนี้

```
IMotion* pMot;
IVisual* pVis;
GetClassObject(CLSID_CSpaceship, IID_IMotion, (void**) &pMot);
    pMot->Fly();
pMot->QueryInterface(IID_IVisual, (void**) &pVis);
pVis->Display();
```

รายละเอียดของ GetClassObject มีดังนี้

```
GetClassObject(int& nClsid, int& nIid, void** ppvObj)
{
        ASSERT(nClsid == CLSID_CSpaceship);
        CSpaceship* pObj = new CSpaceship();
        IUnknown* punk = &pObj->m_xMotion;
        Return punk->QueryInterface(nIid, ppvObj);
}
```



รูปที่ 2-3 รูปทั่วไปของคอมโพเนนต์อย่างง่าย

2.6 Class Factories

Class Factories หรืออีกชื่อหนึ่ง Class Object เป็น class ที่อิมพลีเมนต์มาจาก IclassFactory มีหน้าที่สร้างคอมโพเน็นต์ออบเจ็กต์ ซึ่งเรียกใช้ผ่านเมธอด coCreateInstance เปรียบได้กับ GetClassObject ในตัวอย่างก่อนหน้านี้

2.7 COM and the Windows Registry

ในการใช้งานจริง COM ถูกแบ่งออกเป็น 2 ประเภท คือ In-Process Component สร้างเป็น DLL และ Out-of-Process Component สร้างเป็น EXE เมื่อโปรแกรมทางฝั่งไคล์เอ็นต์ต้องการสร้าง คอมโพเน็นต์ออบเจ็กต์มันก็จะไปอ่านไฟล์ที่บรรจุคอมโพเน็นต์ที่ต้องการขึ้นมาจากดิสก์ โดยหาจาก Windows Registry โดยใช้ Class ID 128 bit ก็จะได้ที่อยู่ของไฟล์ จากนั้น COM ก็เรียกมันขึ้นมาพร้อมกับเชื่อมโยงเข้ากับโปรแกรมทางฝั่งของไคล์เอ็นต์

2.8 COM Client เรียกใช้ In-Process Component ได้อย่างไร

In-Process Component หรือ DLL Component มีการติดต่อกับไคล์เอ็นต์ง่ายกว่า EXE Component ดังแสดงได้ด้วย Pseudocode (ในชีวิตจริงใช้ MFC Library จะช่วยซ่อนความซับซ้อนในการอิมพลีเมนต์ได้มากทีเดียว) ต่อไปนี้

Client

```
CLSID clsid;
IClassFactory* pClf;
```

Iunknown* pUnk;

CoInitialize(NULL) // Initialize COM

CLSIDFromProgID("componentname",&clsid);

COM

COM uses the Registry to look up the class ID from "componentname"

Client

CoGetClassObject(clsid, CLSCTX_INPROC_SERVER, NULL, IID_IclassFactory, (void**)&pClf);

COM

COM uses the class ID to look for a component in memory

If (component DLL is not loaded already)

{

COM gets DLL filename from Registry

COM load the component DLL into process memory

}

DLL Component

If (component just loaded)

{

Global factory objects are constructed

DLL's InitInstance called (MFC only)

}

COM

COM calls DLL'S global exported DllGetClassObject with the CLSID value that was passed to CoGetClassObject

Dll Component

DllGetClassObject returns ICassFactory*

COM

COM return IclassFactory* to client

Client

PClf->CreateInstance (NULL, IID_Iunknown, (void**) &pUnk);

DLL Component

Class factory 's CreateInstance function called (called directly-through component's vtable)

Constructs object of "componentname" class

Return requested interface pointer

Client

PClf->Release();

PUnk->Release();

DLL Component

"componentname" Release is called through vtable

if (refcount==0)

{

Object destroys itself

}

Client

CoFreeUnusedLibraries();

COM

COM calls DLL's global exported DllCanUnloadNow

DLL Component

DllCanUnloadNow called if (all DLL's object destroyed)

{

Return TRUE

}

Client

CoUnintialize(); // COM free the DLL if DllCanUnloadNow return TRUE just prior to exit

COM

COM release resources

Client

Client exits

DLL Component

Windows unloads the DLL if it is still loaded and no other program are using it

2.9 COM Client เรียกใช้ Out-of-Process ได้อย่างไร

วิธีติดต่อกับ Exe component ที่ประมวลผลอยู่กันคนละโพรเซส หรืออยู่กันคนละเครื่องวิธีการเรียกใช้ยังคงเหมือนเดิม COM จะจัดการติดเครื่องคอมพิวเตอร์ที่อยู่ห่างไกลให้ โดยใช้ Remote Procedure Calls (RPC)

ด้วยวิธี RPC ไคล์เอ็นต์จะติดต่อไปยัง DLL ชนิดพิเศษ ที่เรียกว่า Proxy แล้วตัว Proxy จะส่งสตรีมของข้อมูลไปให้กับ stub ซึ่งอยู่ในอีกเครื่องหนึ่ง เมื่อไคล์เอ็นต์เรียกคอมโพเน็นต์ฟังก์ชัน (Component Function) ตัว proxy จะส่งสตรีมข้อมูลให้กับ stub แล้ว stub ก็จะไปเรียกคอมโพเน็นต์ฟังก์ชัน และส่งผลลัพธ์กลับ กลไกในการเปลี่ยนพารามิเตอร์กับสตรีมข้อมูลที่ใช้ส่ง เรียกว่า Marshaling

ถ้าเราใช้อินเทอร์เฟซมาตรฐาน (อินเทอร์เฟซที่กำหนดโดย Microsoft) stub และ proxy จะอยู่ใน OLE32.dll แต่ถ้าเราสร้างอินเทอร์เฟซขึ้นมาเอง เช่น IMotion และ IVisual เราต้องสร้าง stub และ proxy เอง โดยกำหนดส่วนอินเทอร์เฟซด้วย Interface Definition Language (IDL) และคอมไพล์ด้วย MIDL Compiler

การเรียกใช้ Out-of-Process จะแสดงด้วย pseudocode ข้างล่าง ซึ่งถ้าเปรียบเทียบกับ DLL Component จะเห็นว่าในส่วนของไคล์เอ็นต์มีการเรียกใช้ที่ไม่แตกต่างกัน

Client

```
CLSID clsid;
IClassFactory* pClf;
IUnknown* pUnk;
CoInitialize(NULL); // Initialize Com
CLSIDFromProgID("componentname",&clsid);
```

COM

```
COM uses the Registry to look up the class ID from "componentname"
```

Client

```
CoGetClassObject(clsid, CLSCTX_LOCAL_SERVER, NULL, IDD_IClassFactory, (void**)pClf);
```

COM

```
COM uses the class ID to look for a component in memory
If (component EXE is not loaded already, or If we need another instance)
{
    COM get EXE filename from the Resgistry
    COM loads the compoent EXE
}
```

EXE Component

```
If (just loaded)
{
    Global factory objects are constructecd
    InitInstance called (MFC only)
    CoInitialize(NULL);
    For each factory object
    {
        CoRegisterClassObject(…);
        Return IClassFactory* to COM
    }
}
```

COM

COM return the requested interface pointer to the client
(client's pointer is not the same as the cmpoent's interface pointer)

Client

```
PClf->CreateInstance(NULL,IID_Iunknown,(void**)&pUnk);
```

EXE Component

Class factory's CreateInstance function called
(called indirectly through marshaling)
Cnstructs object of "componentname" class
Return reuested interface pointer indiredtly

Client

```
PClf->Release();
PUnk->Release();
```

EXE Component

"component" Release is called indirectly

```
if (refcount==0)
{
    Object destroys itself
}
```

if (all objects released)

{

Component exits gracefully

}

Client

CoUninitialize(); // just prior to exit

COM

COM calls Release for any objects this client has failed to release

EXE Component

Component exits

COM

COM releases resources

Client

Client exits

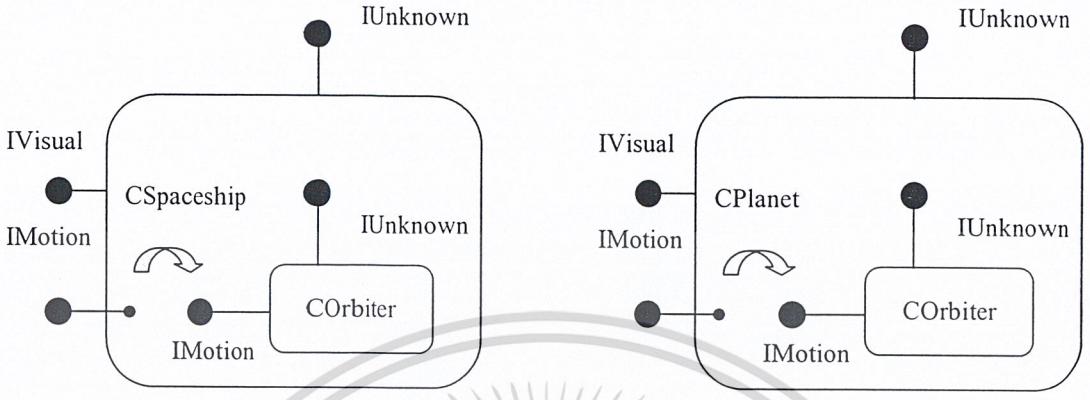
จากที่ผ่านมา COM มีบทบาทสำคัญยิ่งในการติดต่อระหว่างไคล์เอ็นต์ และ คอมโพเน็นต์ โดยแต่ละคอมโพเน็นต์ จะบริหาร Reference count ของตัวเอง ด้วยกลไกของ AddRef/Release และทำลายตัวเองเมื่อไม่มีใครอ้างถึงมัน

2.10 Containment และ Aggregation

การเขียนโปรแกรมแบบ COM จะไม่มีการอิมพลีเมนต์ inherit แต่จะใช้วิธี Containment และ Aggregation แทน เพื่อสอดคล้องแนวคิดของการ Reuse ในแบบ OOP

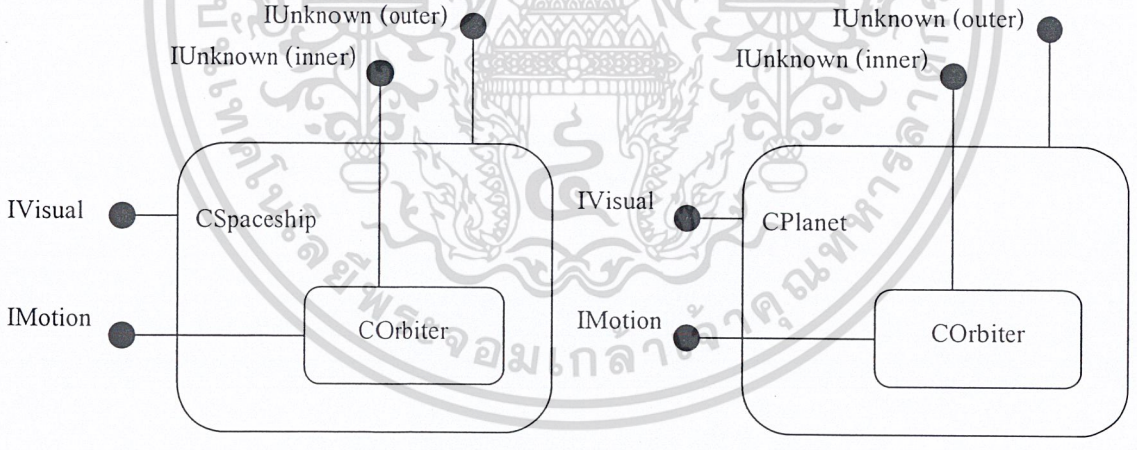
Containment อนุญาตให้คอมโพเน็นต์ออบเจ็กต์ (inner class) สามารถอยู่ภายในคอมโพเน็นต์ออบเจ็กต์ (outer class) อีกตัวหนึ่งได้ ในการอิมพลีเมนต์ outer class ต้องอิมพลีเมนต์ทุก ๆ อินเทอร์เฟซฟังก์ชันยกเว้นอินเทอร์เฟซฟังก์ชันที่สามารถเรียกใช้อินเทอร์เฟซฟังก์ชันของ inner class ได้

โดยอินเทอร์เฟซภายนอกจะติดต่อกับอินเทอร์เฟซภายใน ดังรูป



รูปที่ 2-4 แสดงวิธีการ Containment

Aggregation คือ การทำ Containment ที่มีความ ซับซ้อนมากขึ้น โดยไคลเอนต์สามารถเข้าถึงอินเทอร์เฟซของ inner class ได้โดยตรง



รูปที่ 2-5 แสดงวิธีการ Aggregation

จากรูป IVisual จะเห็นแค่เพียงอินเทอร์เฟซ IUnknown (outer) เท่านั้น ไม่สามารถ QueryInterface IMotoin ได้ ถ้าต้องการอินเทอร์เฟซ IMotion จะต้อง QueryInterface จาก Iunknown (inner) แต่ถ้ามี IMotion อยู่ ถ้าต้องการ IVisual จะต้องสร้าง Corbiter ให้รู้จัก IUnknown (outer) โดยเมธอด CreatInstance แล้วผ่านค่า Iunknown (outer) ที่สร้างไว้แล้วให้มันรู้จัก เพียงเท่านี้ก็จะสามารถ QueryInterface IVisual จาก IMotion ได้

2.11 คำสำคัญ

ถ้าต้องการทดสอบว่าเข้าใจหลักการการทำงานของ COM แล้วหรือยัง ลองถามตัวเองว่าเข้าใจในคำเหล่านี้แล้วหรือยัง

- Interfaces >> กลไกที่ทำให้ออบเจ็กต์แสดงว่าตัวมันทำอะไรได้บ้าง
- IUnknown >> Basic interface ที่ทุก interface ต้อง Inherit เฉพาะอินเทอร์เฟซ
- Reference counting >> ค่า counter ที่ใช้นับว่ามีพอยน์เตอร์อินเทอร์เฟซอ้างถึงคอมโพเน็นต์ออบเจ็กต์อยู่ทั้งหมดกี่ตัว
- QueryInterface >> เป็นเมธอดที่ใช้เพื่อเข้าถึงอินเทอร์เฟซ
- Marshaling >> กลไกในการเปลี่ยนพารามิเตอร์กับสตรีมข้อมูลที่ใช้ส่ง
- Containment >> การสร้างคอมโพเน็นต์ออบเจ็กต์หนึ่ง ไว้ในอีกคอมโพเน็นต์ออบเจ็กต์หนึ่ง ที่มีการเข้าถึงของอินเทอร์เฟซภายในจากอินเทอร์เฟซภายนอก
- Aggreagation >>การสร้าง com object หนึ่ง ไว้ในนอีก com object หนึ่ง ที่มีการเข้าถึงของ inner interface โดยตรง

บทที่ 3

Thai Text to Speech System

3.1 บทนำ

ในระบบ Text to Speech จะประกอบด้วยส่วนสำคัญ 2 ส่วนคือ การวิเคราะห์ข้อความ (Text Analysis) และการสังเคราะห์เสียงพูดจากข้อความที่ได้วิเคราะห์ (Text Synthesis)

เนื่องจากอินพุทเป็นข้อความซึ่งประกอบด้วยบริบท วลี ต่าง ๆ มากมาย ตัวบริบทนั้นอาจประกอบด้วยคำไทย คำทับศัพท์ภาษาต่างประเทศ (Transliteration) คำที่เป็นชื่อเฉพาะ (Proper Noun) คำย่อ หรือแม้แต่สัญลักษณ์พิเศษ ด้วยเหตุนี้จึงต้องมีการแยกข้อความเหล่านั้นออกเป็นคำที่มีความหมายเพื่อจะทำการสังเคราะห์เสียงต่อไป สำหรับการสังเคราะห์เสียงเพื่อให้ได้เสียงที่เป็นธรรมชาติต้องคำนึงถึงการนำข้อมูลเสียงมาต่อกัน (Concatenate) ซึ่งสามารถเลือกหน่วยเสียงสำหรับนำมาต่อได้หลายแบบ เช่นอัดเสียงเป็นคำ (Word) พยางค์ (Syllable) หรือใช้วิธีอัดเสียงครึ่งพยางค์ (Demisyllable) แต่สำหรับวิธีอัดเสียงครึ่งพยางค์นี้ถึงแม้จะใช้หน่วยเสียงน้อยแต่จะทำให้ได้คุณภาพเสียงที่ไม่เป็นธรรมชาติ ซึ่งเทียบกับวิธีเก็บเสียงเป็นพยางค์แล้ว คุณภาพเสียงจะดีกว่าถึงแม้จะใช้ข้อมูลเสียงมากกว่าก็ตาม

3.2 สถาปัตยกรรมของระบบ Text to Speech สำหรับภาษาไทย

สถาปัตยกรรมของระบบการสังเคราะห์เสียงพูดด้วยภาษาไทยประกอบด้วยส่วนต่าง ๆ ดังนี้

- การตัดคำ (Thai Word Segmentation Algorithm) ทำหน้าที่แบ่งประโยคที่เขียนติดกันให้แยกออกเป็นคำ ๆ โดยอาศัยโครงสร้างข้อมูลพจนานุกรมแบบทรัย (Trie Data Structure)
- วจีวิภาค (Parsing Algorithm) ทำหน้าที่แบ่งคำที่ไม่รู้จักออกเป็นพยางค์ย่อย ๆ โดยอาศัยหลักภาษาไทย 16 ข้อ
- พจนานุกรมการอ่านออกเสียง (Dictionary Based Pronounciation)

3.3 การตัดคำ

ในการตัดคำ (Thai Word Segmentation Algorithm) ใช้โครงสร้างข้อมูลพจนานุกรมแบบทรัย (Trie Data Structure) เนื่องจากให้ประสิทธิภาพสูงทั้งในแง่ของความรวดเร็วในการทำงานและประหยัดเนื้อที่ในการจัดเก็บข้อมูลพจนานุกรม ทรัยคือโครงสร้างข้อมูล แบบต้นไม้ที่มีกิ่งก้าน (Edges) แทนตัวอักษรที่เข้ารหัสเป็นข้อมูลใด ๆ ที่ต้องการจัดเก็บ ในกรณีที่ใช้ทรัยจัดเก็บข้อมูลพจนานุกรม คำหนึ่งคำก็จะได้จากเส้นทาง (Path) จากราก (Root) ไล่ไปจนถึงใบ (Leaf) ดังตัวอย่างประโยค

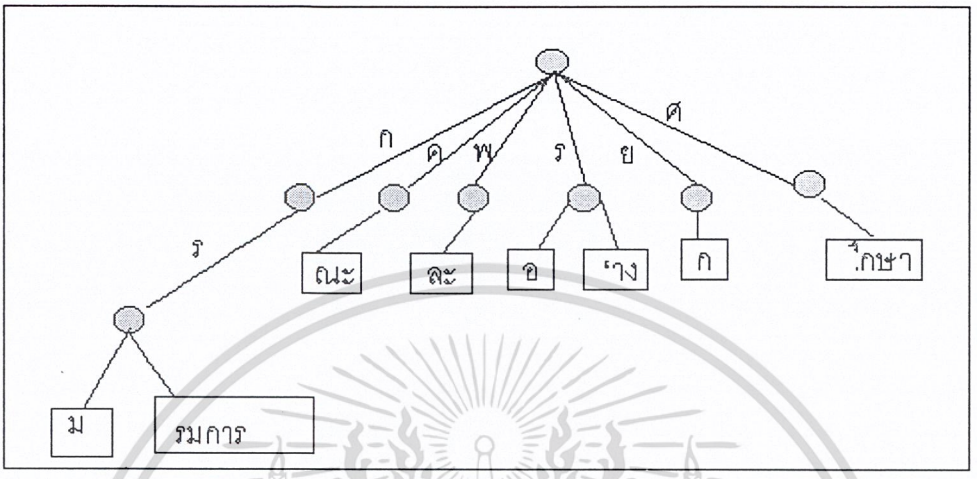
คณะกรรมการกรมพละศึกษารอยกร่าง

จากตัวอย่างข้างบนสามารถแยกคำซึ่งประกอบด้วยคำต่าง ๆ ดังต่อไปนี้ ซึ่งจะสังเกตเห็นว่าไม่มีการจัดเก็บอักษรที่มีส่วนต้นซ้ำกัน เมื่อทำการจัดเก็บด้วยโครงสร้างข้อมูลแบบทรัย

คำ	ทรัย
กรม	กรม
กรรมการ	รมการ
คณะ	คณะ
พละ	พละ
รอ	รอ
ร่าง	่าง
ยก	ยก
ศึกษา	ึกษา

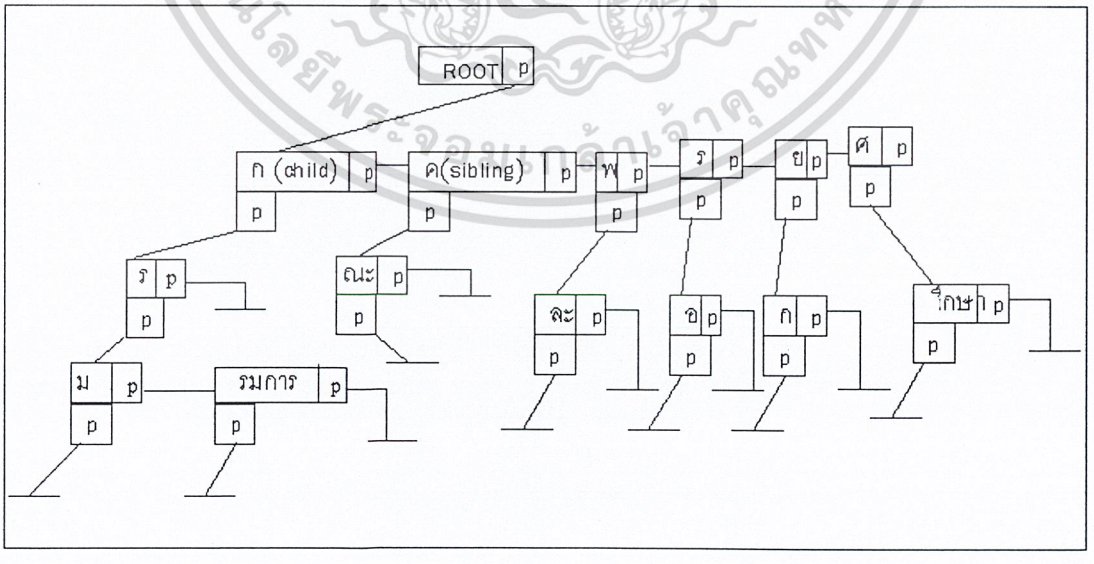
รูปที่ 3-1 แสดงคำที่ได้จากการแยกคำเพื่อทำการจัดเก็บข้อมูลแบบทรัย

หลังจากนำคำที่แยกออกจากประโยคตัวอย่างไปจัดเก็บในโครงสร้างข้อมูลแบบทรัยจะได้โครงสร้างข้อมูลแบบทรัย ดังรูปข้างล่าง



รูปที่ 3-2 แสดงโครงสร้างข้อมูลทรัย

การสร้างโครงสร้างข้อมูลแบบทรัยอาจทำได้โดยใช้อาร์เรย์ (Array) หรือใช้พอยน์เตอร์ (Pointer) ก็ได้ ในที่นี้จะแสดงการสร้างด้วยการใช้พอยน์เตอร์แบบหลายทางโดยให้โหนดแรกเป็นลูกและ โหนดถัดไปเป็นลูกในระดับเดียวกัน (First Child Next Sibling Multi way Tree) รูปข้างล่างคือ การสร้างทรัยแบบพอยน์เตอร์ดังกล่าวของตัวอย่างจากทรัยในรูปข้างบน



รูปที่ 3-3 แสดงการสร้างทรัยแบบพอยน์เตอร์

เนื่องจากทรัยคือโครงสร้างข้อมูลแบบต้นไม้ดังนั้นความรวดเร็วในการทำงานโดยเฉลี่ย เป็น O(log n) หรือความเร็วในการทำงาน มีผลแปรผัน น้อยมากกับขนาดของข้อมูลจึงมี ความเหมาะสมกับข้อมูลขนาดใหญ่มากกว่าโครงสร้างข้อมูลแบบลิงค์ลีสต์ที่แปรผัน โดยตรงกับขนาดข้อมูลที่ป้อนให้ O(n) ในแต่ละโหนดของทรัยเมื่อใช้พอยน์เตอร์เป็นตัวสร้างจะ ประกอบไปด้วยสามส่วนคือ ข้อมูล พอยน์เตอร์ชี้ไปยังลูกตัวแรก (first child) และพอยน์เตอร์ชี้ไป ยังลูกข้างเคียง (next sibling) โหนดของทรัยดังกล่าวนี้สามารถแสดงในลักษณะการนิยามตามแบบภาษาซีได้ดังต่อไปนี้

```
Typedef struct trie_node *trie_ptr;
struct trie_node
{
    element_type element;
    trie_ptr first_child;
    trie_ptr next_sibling;
}
```

3.4 วจีวิภาค

วจีวิภาค (Parsing Algorithm) หรือขั้นตอนและวิธีการแบ่งพยางค์จากคำใด ๆ ทั้งที่รู้จักและไม่รู้จักโดยอาศัยกฎ 16 ข้อ จากหลักภาษาไทยของอาจารย์กำชัย ทองหล่อ เป็นหลักโดยกฎดังกล่าวสามารถอธิบายได้ดังนี้

สัญลักษณ์	คำนิยาม
พ	พยัญชนะไทย
(พ)	พยัญชนะตัวควบกล้ำ ซึ่งอาจจะมีหรือไม่มีก็ได้
รร	ตัว ร หัน
ส นำ	สระนำในภาษาไทย
ส ตาม	สระตามในภาษาไทย
ด	ตัวสะกดที่ใช้ในภาษาไทย
[ย, อ]	พยัญชนะที่ใช้เป็นสระ ซึ่งจะขาดไม่ได้ เป็น ตัว ย หรือตัว อ เท่านั้น

ตารางที่ 3-1 ตารางสัญลักษณ์และคำนิยามที่ใช้ใน Parsing Algorithm

กฎข้อที่ 1 การใช้อักษร อ นำ ย ซึ่งกรณีนี้เป็นกรณีพิเศษ ได้แก่คำว่า อย่า อยู่ อย่าง

กฎข้อที่ 2 การใช้อักษร อ นำ ย ซึ่งกรณีนี้เป็นกรณีพิเศษ ได้แก่คำว่า อยาก

กฎข้อที่ 3 การใช้ตัว ร หัน ในกรณีที่มีตัวสะกดตามหลัง สามารถแทนด้วยสัญลักษณ์ พ รร ด

ตัวอย่างเช่น พรรค วรรณ

กฎข้อที่ 4 การใช้ตัว ร หัน ในกรณีที่ไม่มีตัวสะกดตามหลัง สามารถแทนด้วยสัญลักษณ์ พ รร

ตัวอย่างเช่น พรรษา วรรณา

กฎข้อที่ 5 การใช้ สระ เอา เอาะ สามารถแทนด้วยสัญลักษณ์ เ พ พ า (ะ)

ตัวอย่างเช่น เพราะ เขลา

กฎข้อที่ 6 การใช้ คำที่อ่านออกเสียงเป็น เอิง สามารถแทนด้วยสัญลักษณ์ เ พ (พิ) ด

ตัวอย่างเช่น เพลิง เทิง

กฎข้อที่ 7 การใช้ สระ เออะ เออ สามารถแทนได้ด้วยสัญลักษณ์ เ พ อ (ะ)

ตัวอย่างเช่น เลอะเทอะ เผลอ เจอ

กฎข้อที่ 8 การใช้ สระ เอียะ เอีย เอือะ เอือ ตามด้วยตัวสะกด สามารถแทนได้ด้วยสัญลักษณ์ เ พ (พี, พี) [ย, อ] ด

ตัวอย่างเช่น เพื่อน เรียน เกลียด

กฎข้อที่ 9 กฎของการใช้ สระ เอียะ เอีย เอือะ เอือ สามารถแทนได้ด้วยสัญลักษณ์ เ พ (พี, พี) [ย, อ]

ตัวอย่างเช่น เสีย เผียะ เพื่อ เกือะ

กฎข้อที่ 10 การใช้ สระนำ โอะ โอ ไอ ใอ เอะ เอ สามารถแทนด้วยสัญลักษณ์ ส นำ พ พ (ะ)

ตัวอย่างเช่น ไกล โกน โต๊ะ ใจ เกะ เห

กฎข้อที่ 11 การใช้ สระ โอ ไอ ใอ เอ ที่มีตัวสะกดสามารถแทนด้วยสัญลักษณ์ ส นำ พ (พ) ด

ตัวอย่างเช่น โอ่ง โล่ง โกน เพลง

กฎข้อที่ 12 การใช้ พยัญชนะแล้วตามด้วยสระ ตัวสะกดสามารถแทนด้วยสัญลักษณ์ พ (พ) ส ตาม ด

ตัวอย่างเช่น กราม พราหม กัด

กฎข้อที่ 13 การใช้ พยัญชนะ สระ สามารถแทนด้วยสัญลักษณ์ พ (พ) ส ตาม

ตัวอย่างเช่น พระ กะทิ

กฎข้อที่ 14 การใช้ พยัญชนะ ตัวสะกด สามรถแทนด้วยสัญลักษณ์ พ (พ) ด

ตัวอย่างเช่น กด ห่ม

กฎข้อที่ 15 การใช้ พยัญชนะ ที่ตามด้วยตัวสะกด ร ล สามารถแทนด้วยสัญลักษณ์ พ [ร, ล]

ตัวอย่างเช่น พร พล กร กล

กฎข้อที่ 16 การใช้ พยัญชนะ ที่ออกเสียง อะ กึ่งเสียง สามารถแทนด้วยสัญลักษณ์ พ
ตัวอย่างเช่น ขจี อนัตตา สนิท

3.5 การติดตามหาคำที่ไม่รู้จัก

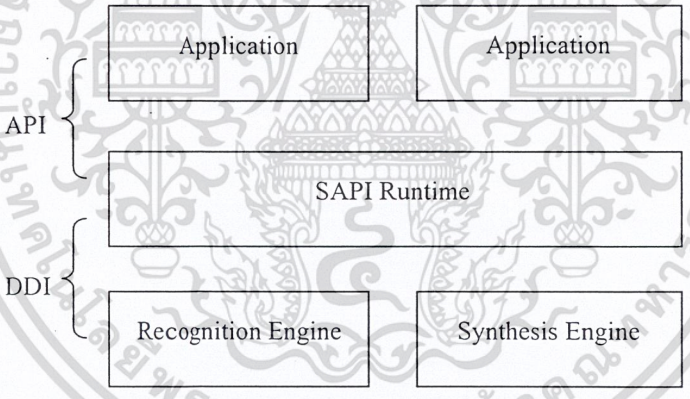
กระบวนการติดตามหาร่องรอยของคำที่ไม่รู้จักเพื่อแยกออกมาจากคำที่มีรู้จักใดๆ ทำโดยเริ่มจากเมื่อเกิดการตัดคำผิดพลาดเกิดขึ้นให้นำเศษของคำไปรวมกับคำที่เพิ่งตัดแบ่งแยกมาได้ก่อนหน้านี้แล้วผ่านกระบวนการวจีวิภาค ถ้าทำได้สำเร็จให้ถือว่าผลของการรวมกันคือคำที่ไม่รู้จักหนึ่งคำ แต่ถ้าทำไม่สำเร็จคือเหลือเศษของพยางค์ให้นำเศษที่เหลือไปรวมกับคำที่สามารถตัดได้หลังจากเศษของพยางค์นี้แล้วลองทำวจีวิภาคดู ถ้าทำสำเร็จให้ถือว่าผลของการรวมทั้งสองได้คำที่ไม่รู้จักสองคำ ถ้าทำไม่สำเร็จให้นำทั้งสามส่วนคือคำที่ตัดได้ก่อนหน้า เศษของคำ และ คำที่ตัดได้หลังเศษคำนำมารวมกันเป็นคำที่ไม่รู้จักหนึ่งคำ หลังจากนี้ทำกระบวนการวจีวิภาคกับคำที่ไม่รู้จักทั้งหมดเพื่อแยกพยางค์และหาหน่วยเสียงต่อไป

บทที่ 4

Speech API (SAPI)

4.1 บทนำ

SAPI (Speech API) เป็น API (Application Programming Interface) ที่จัดการเกี่ยวกับเสียงพูด ซึ่ง SAPI แบ่งการทำงานเกี่ยวกับเสียงพูดออกเป็น 2 ประเภท คือ Speech Recognition (SR) คือการจดจำเสียงพูด และ Speech Synthesis (TTS หรือ Text To Speech) เป็นการแปลงข้อความให้เป็นเสียงพูด ทั้ง Speech Recognition และ Speech Synthesis นั้นจะมี SAPI ทำหน้าที่เป็นตัวกลางในการทำให้ แอพพลิเคชั่นเสียงกับ เอ็นจินเสียง สามารถติดต่อถึงกันผ่านทางอินเทอร์เฟซ 2 ตัว คือ Application Programming Interface (API) ซึ่งเป็นอินเทอร์เฟซสำหรับทางฝั่งแอพพลิเคชั่น และ Device Driver Interface (DDI) เป็นอินเทอร์เฟซสำหรับทางฝั่งเอ็นจิน



รูปที่ 4-1 โครงสร้างของ SAPI

การที่ SAPI มี API และ DDI ทำให้โปรแกรมทางฝั่งของแอพพลิเคชั่นไม่มีความจำเป็นต้องรู้ฟังก์ชันการทำงานของตัวเอ็นจิน เปรียบได้กับว่าแอพพลิเคชั่นมี SAPI ทำหน้าที่เป็นล่ามคอยสื่อสารกับเอ็นจิน ด้วยเหตุนี้จึงช่วยลดขั้นตอนในการเขียนโปรแกรมจัดการกับเสียงบางส่วน เช่น การทำมัลติเธรดดิง (Multi-threading) และการจัดการเกี่ยวกับออดิโอดีไวซ์ (Audio Device) ทั้งในส่วนของ SR และ TTS โปรแกรมเมอร์จึงสามารถพัฒนาโปรแกรมที่เกี่ยวข้องกับเสียงพูดได้สะดวกมากยิ่งขึ้น

4.2 Application Programming Interface (API)

สำหรับอินเทอร์เฟซในส่วนที่ให้แอพพลิเคชั่นใช้ติดต่อกับเอ็นจิน จะแบ่งอินเทอร์เฟซออกสำหรับการทำงานอีก 2 ส่วน คือ อินเทอร์เฟซสำหรับการทำ Speech Recognition (SR) และ อินเทอร์เฟซสำหรับการทำ Speech Synthesis (TTS หรือ Text to Speech) ซึ่งรายละเอียดของอินเทอร์เฟซแต่ละส่วนมีดังนี้

4.2.1 API สำหรับ Speech Synthesis

อินเทอร์เฟซที่ใช้ในการทำ Speech Synthesis มีเพียงอินเทอร์เฟซเดียวคือ ISpVoice อินเทอร์เฟซ ISpVoice นอกจากจะสามารถจัดการในการทำ TTS หรือ Text to Speech แล้วยังสามารถเปลี่ยนแปลงค่าคุณสมบัติเฉพาะของแต่ละเสียงได้อีกด้วย โดยสามารถเปลี่ยนแปลงคุณสมบัติต่าง ๆ ได้ทันทีขณะโปรแกรมกำลังทำงาน

ในการสังเคราะห์เสียงจากข้อความนั้น นอกจากจะส่งข้อความที่ต้องการเป็นค่าพารามิเตอร์สำหรับการสังเคราะห์เสียง ไปให้เมธอด ISpVoice::Speak แล้ว ยังสามารถเลือกใช้ภาษาในการสังเคราะห์เสียงที่ใช้ในรูปแบบของภาษา XML ผ่านเมธอด ISpVoice::Speak ได้อีกด้วยและยังสามารถแสดงผลข้อมูลเสียงที่เป็นสตรีมได้โดยเมธอด ISpVoice::SpeakStream

จะเห็นได้ว่า ISpVoice เพียงอินเทอร์เฟซเดียวแต่สามารถจัดการกับแอพพลิเคชั่นได้หลายรูปแบบทำให้สามารถอิมพลีเมนต์ไปใช้งานได้อย่างหลากหลาย

4.2.2 API สำหรับ Speech Recognition

อินเทอร์เฟซสำหรับ Speech Recognition มีอินเทอร์เฟซที่สำคัญอยู่ 3 ตัว คือ ISpRecoContext, ISpRecognizer และ ISpRecoGrammar

- ISpRecoContext เป็นอินเทอร์เฟซหลักที่ใช้ในการทำ Speech Recognition เช่นเดียวกับอินเทอร์เฟซ ISpVoice ใน Speech Synthesis ซึ่ง ISpRecoContext จะเป็นตัวเชื่อมต่อสำหรับแอพพลิเคชั่นสำหรับการรับอีเวนต์ของการทำ Speech Recognition รวมถึงฟังก์ชันและโครงสร้างของเอ็นจิน
- ISpRecognizer เป็นอินเทอร์เฟซที่แทนแต่ละเอ็นจิน แอพพลิเคชั่นสามารถติดต่อ Recognizer ออบเจ็กต์ได้มากกว่าหนึ่งตัวขึ้นอยู่กับการอิมพลีเมนต์
- ISpRecoGrammar เป็นอินเทอร์เฟซสำหรับให้แอพพลิเคชั่นจัดการกับคำหรือข้อความที่ต้องการให้เอ็นจินทำการ Recognize สำหรับแอพพลิเคชั่นนั้น ๆ

4.3 Device Driver Interface (DDI)

อินเทอร์เฟซสำหรับเอ็นจิน แบ่งออกเป็น 2 ส่วนเช่นเดียวกับ Application Programming Interface (API) คือ อินเทอร์เฟซสำหรับการทำ Speech Recognition (SR) และ อินเทอร์เฟซสำหรับการทำ Speech Synthesis (TTS หรือ Text to Speech) ซึ่งรายละเอียดของอินเทอร์เฟซแต่ละส่วนมีดังนี้

4.3.1 DDI สำหรับ Speech Synthesis

ในการทำ Speech Synthesis ของเอ็นจิน ทำโดยการอิมพลีเมนต์อินเทอร์เฟซ ISpTTSEngine โดยมีเมธอดหลักที่เรียกโดย SAPI ในการใช้งานคือ ISpTTSEngine::Speak SAPI จะเป็นคนจัดการกับ XML Parsing เพื่อแปลงค่าให้ได้อินพุทที่เป็นสตรีมของข้อความแทนที่จะหน้าที่ของเอ็นจิน ส่วนเมธอด Speak จะถูกสั่งให้จัดการกับข้อความตามสถานะของ XML และด้วยเมธอด Speak นี้จะได้รับพอยน์เตอร์ของ SpVoice ผ่านอินเทอร์เฟซ ISpTTSEngineSite ซึ่ง TTS เอ็นจินจะใช้อินเทอร์เฟซนี้ในการจัดลำดับอีเวนต์และเขียนข้อมูลกลับ สำหรับรายละเอียดจะกล่าวถึงในบทต่อไป

4.3.2 DDI สำหรับ Speech Recognition

Speech Recognition DDI จะจัดการกับฟังก์ชันการทำงานในการรับข้อมูลเสียงจาก SAPI แล้วคืนค่าของการ Recognition กลับไป อินเทอร์เฟซที่ใช้มีสองตัว คือ ISpSREngine และ ISpSREngineSite โดยที่ ISpSREngine จะถูกอิมพลีเมนต์ทางฝั่งเอ็นจินและ ISpSREngineSite จะถูกอิมพลีเมนต์โดย SAPI ซึ่ง SAPI จะติดต่อกับเอ็นจินและเป็นคนคอยจัดสรรข้อมูลที่ใช้ในการทำ Speech Recognition ให้กับเอ็นจิน

เอ็นจินจะให้บริการแก่ SAPI ผ่านอินเทอร์เฟซ ISpSREngine ฟังก์ชันที่ถูกสร้างขึ้นจากการ Recognition คือ ISpSREngine::RecognizerStream เมื่อ SAPI เรียกเมธอด ISpSREngine::SetSite มันจะส่งพอยน์เตอร์ของอินเทอร์เฟซ ISpSREngineSite เข้ามายังเอ็นจินที่ติดต่อกับ SAPI ขณะกำลังประมวลผลฟังก์ชัน ISpSREngine::RecognizeStream SAPI จะยอมให้เธรดแก่ออบเจ็กต์ของ ISpSREngine และเอ็นจินจะหยุดส่งค่ากลับเมื่อเกิดความผิดพลาด หรือ SAPI หยุดอ่านซึ่งหมายถึงข้อมูลหมดแล้ว

SAPI ช่วยให้นักพัฒนาโปรแกรมไม่จำเป็นต้องจัดการกับรายละเอียดในส่วนของออดีโอดีไวซ์ โดย SAPI จะสร้าง Logical Stream จากเสียงพูด แล้วส่งไปให้กับเอ็นจิน เอ็นจินก็จะสามารถประมวลผลการ Recognition ได้

บทที่ 5

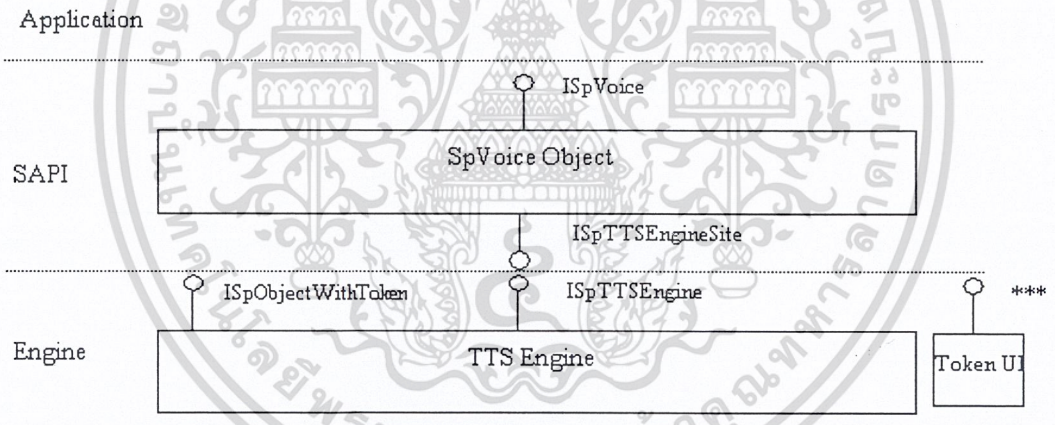
SAPI Text To Speech Engine

5.1 ภาพรวมของ SAPI 5.0

Microsoft Speech API (SAPI) เป็นชั้นระหว่างแอพพลิเคชั่นกับสปีชเอนจิน จุดมุ่งหมายหลักคือเพื่อเป็นมาตรฐานที่ใช้เชื่อมต่อว่างกันและลดความซับซ้อนที่ไม่จำเป็น เช่น การติดต่อ กับอุปกรณ์ออดิโอ, จัดการ theads เป็นต้น ทำให้นักพัฒนาโปรแกรมสามารถพัฒนาโปรแกรมเกี่ยวกับเสียงได้อย่างเต็มที่

5.2 ออบเจ็กต์และอินเทอร์เฟซของ SAPI

ใน SAPI 5.0 มีออบเจ็กต์หลักที่น่าสนในอยู่ 2 ตัว คือ SpVoice object (SAPI) และ TTS Engine object ดังรูป



รูปที่ 5-1 โครงสร้างของ SAPI

SpVoice Object อิมพลีเมนต์ 2 อินเทอร์เฟซ คือ

- ISpVoice เป็น อินเทอร์เฟซที่แอพพลิเคชั่นใช้เพื่อเข้าถึง ฟังก์ชั่นการทำงานของ TTS
- ISpTTSEngineSite เป็น อินเทอร์เฟซที่ Engine ใช้เพื่อส่งผ่านข้อมูลออดิโอ และ อีเวนต์ มายัง SAPI

TTS Engine ต้องอิมพลีเมนต์ 2 อินเทอร์เฟซ คือ

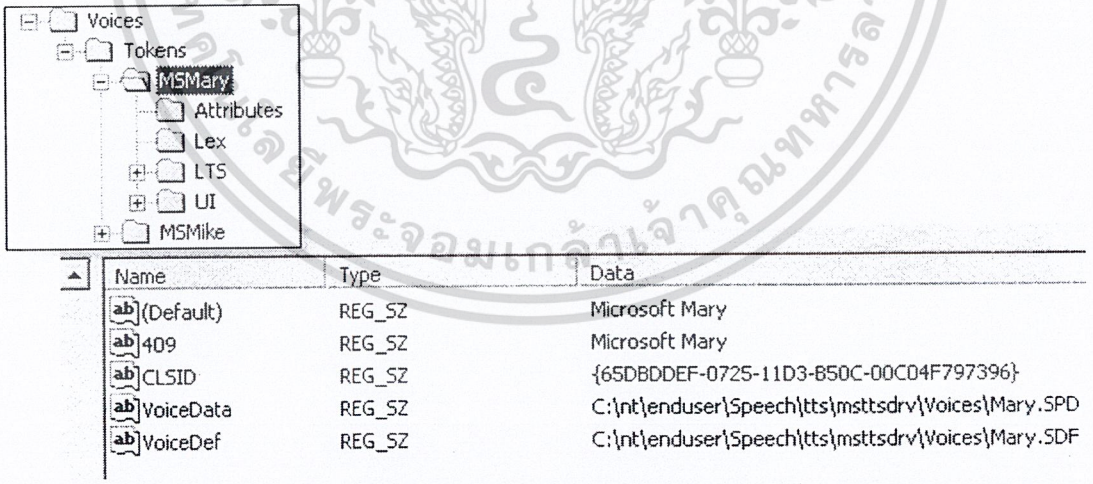
- ISpTTSEngine เป็น อินเทอร์เฟซที่ SAPI เรียกใช้เพื่อสั่งให้ TTS Engine สร้างเสียงจากข้อความ
- IspObjectWithToken เป็น อินเทอร์เฟซที่ SAPI เรียกใช้ เพื่อตั้งค่าเริ่มต้นให้กับ TTS Engine

5.3 การสร้าง และ กำหนดค่าเริ่มต้นให้กับเอ็นจิน

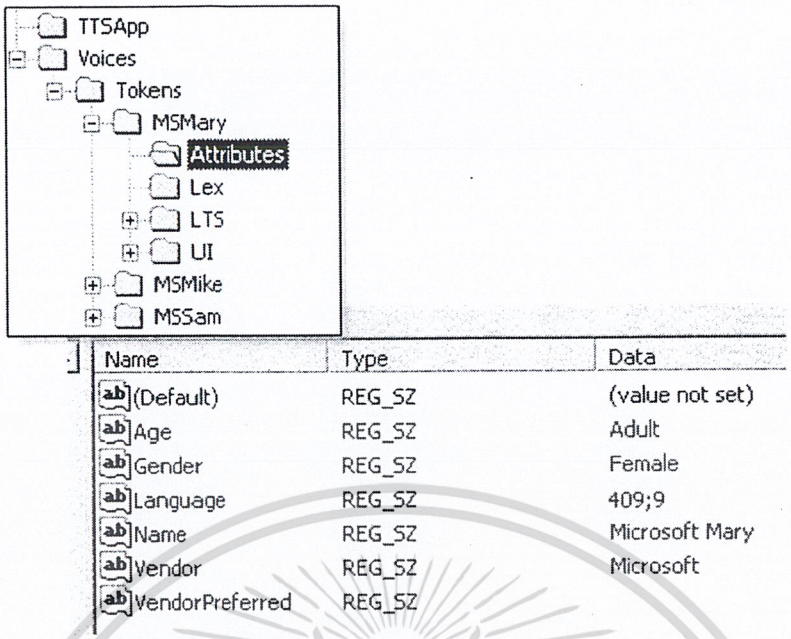
ในสถาปัตยกรรมของ SAPI 5.0 ตัวแอพพลิเคชั่นจะรู้จักแต่ Voice ส่วนตัว SAPI จะรู้จักเฉพาะเอ็นจินเท่านั้น เอ็นจินสามารถสร้างเสียงได้หลายเสียงตามค่าพารามิเตอร์ที่กำหนดให้มัน โดยเมื่อแอพพลิเคชั่นต้องการ Voice SAPI จะสร้างเอ็นจินแล้วตั้งค่าพารามิเตอร์ให้ด้วยเมธอด SetObjectToken ของอินเทอร์เฟซ ISpObjectWithToken

SAPI จะรู้ได้ว่า Voice มีคุณสมบัติอย่างไรและจำเป็นต้องสร้าง Engine ตัวใดมารองรับ SAPI 5.0 ใช้ Token เพื่อแก้ปัญหานี้ Token ใช้แทน TTS Voices ที่มีอยู่ในเครื่องนั้นๆ ภายใน Token จะประกอบด้วย CLSID ของออบเจค และคุณลักษณะอื่นที่มีผลต่อเสียง เมื่อแอพพลิเคชั่นต้องการใช้ TTS Voice SAPI ก็จะดึง Token ขึ้นมาจากรีจิสตรี้ จากนั้น TTS Engine ก็จะถูกสร้างขึ้น จากนั้น SpVoice ออบเจ็กต์ก็จะ Queryinterface ISpObjectWithToken ขึ้นมา เมื่อได้ ISpObjectWithToken มาแล้ว SpVoice ออบเจ็กต์ก็จะเรียกเมธอด SetObjectToken จาก ISpObjectWithToken เพื่อให้เอ็นจินเพื่อเซตค่าพารามิเตอร์ที่ใช้สังเคราะห์เสียงตาม Token ที่อยู่ในรีจิสตรี้

ต่อไปเป็นการแสดงตัวอย่าง Token ของ TTS Voice ใน รีจิสตรี้



รูปที่ 5-2 แสดง TTS Voice



รูปที่ 5-3 แสดง Token Voice Attribute

Token ที่เห็นอยู่ในรูปนี้อยู่ใน HKEY_LOCAL_MACHINE\SOFTWAREMICROSOFT\Speech \Voices\Tokens เป็นเสียงของ MS Mary ซึ่งเป็น ผู้หญิง วัยผู้ใหญ่

5.4 การรับเรียกจาก SAPI (Receiving Call from SAPI-ISpTTSEngine)

เมื่อ TTS Engine ถูกสร้างขึ้นมา SAPI ก็จะเรียกตัว TTS Engine ผ่านทาง ISpTTSEngine. ซึ่ง ISpTTSEngine มี 2 เมธอดคือ –GetOutputFormat และ Speak

GetOutputFormat ใช้เพื่อถามเอ็นจินว่าให้ออดิโอเอาท์พุทแบบใด แล้วเอ็นจินจะให้ค่าใกล้เคียงที่สุดที่ SAPI ต้องการ เมธอดนี้จะถูกเรียกเมื่อไรก็ได้ในช่วงชีวิตของเอ็นจิน

```
HRESULT GetOutputFormat(
        [in] const GUID * pTargetFmtId,
        [in] const WAVEFORMATEX * pTargetWaveFormatEx,
        [out] GUID * pOutputFormatId,
        [out] WAVEFORMATEX ** ppCoMemOutputWaveFormatEx
);
```

pTargetFmtId คือรูปแบบออดิโอเอาท์พุทที่ SAPI ต้องการ ส่วน pOutputFormatId คือรูปแบบออดิโอเอาท์พุทที่เอ็นจินสนับสนุน โดยปกติจะมีค่าเป็น SPDFID_WaveFormatEx สำหรับ pTargetWaveFormatEx ซึ่งมีโครงสร้างเป็น WAVEFORMATEX เพื่อใช้อธิบายออดิโอเอาท์พุทที่ SAPI

ต้องการ TTS Engine ควรตอบกลับด้วยรูปแบบของออดิโอเอาท์พุทที่ใกล้เคียงที่สุดที่จะเป็นไปได้ ผ่านทาง ppComemOutputWaveFormatEx

ถ้า pTargetFmtId = NULL ตัว TTS Engine ควรจะให้ค่าโดยปริยายออกมา

ถ้า pTargetFmdId เป็น SPDFID_TEXT Engine จะให้ค่าอะไรออกมาก็ได้ มีไว้เพื่อดีบักโปรแกรม

Speak เป็นฟังก์ชันที่สำคัญของ ISpTTSEngine ข้อความที่ถูกส่งเข้าไปในฟังก์ชันนี้จะถูกแปลงเป็นเสียงและส่งอีเวนต์กลับมายัง SAPI แล้ว SAPI จะจัดการกับออดิโอให้เอง พร้อมกับส่งอีเวนต์ไปยังแอพพลิเคชั่นโปรแกรม

```
HRESULT Speak(
        [in]DWORD dwSpeakFlags,
        [in]REFGUID rguidFormatId,
        [in]const WAVEFORMATEX * pWaveFormatEx,
        [in]const SPVTEXTFRAG* pTextFragList,
        [in]ISpTTSEngineSite* pOutputSite,
);
```

พารามิเตอร์ตัวแรก dwSpeakFlags มีค่าที่เป็นได้คือ 0 กับ SPF_NLP_SPEAK_PUNC ถ้าเป็น 0 เอ็นจินจะสังเคราะห์เสียงแบบปกติไม่อ่านเครื่องหมาย แต่ถ้าเป็น SPF_NLP_SPEAK_PUNC จะอ่านเครื่องหมายด้วย เช่น . อ่านว่า พีเรียด

พารามิเตอร์ตัวที่สองและสามเป็นตัวกำหนดรูปแบบเอาท์พุทที่เอ็นจินใช้ ซึ่งต้องรับประกันว่าเป็นรูปแบบที่เอ็นจินสนับสนุน

พารามิเตอร์ตัวที่สี่เป็นโครงสร้างของข้อความที่จะใช้สังเคราะห์เสียง เป็นโครงสร้างข้อมูลแบบลิงค์ลิสของส่วนย่อยข้อความ (Text Fragment) มีโครงสร้างดังนี้

```
typedef struct SPVTEXTFRAG
{
        struct SPVTEXTFRAG *pNext;
        SPVSTATE State;
        LPCWSTR pTextStart;
        ULONG ulTextLen;
        ULONG ulTextSrcOffset;
} SPVTEXTFRAG;
```

pTextStart เป็นพอยน์เตอร์ชี้ไปที่จุดเริ่มต้นของส่วนย่อยข้อความ ulTextLen คือความยาวของส่วนย่อยข้อความ ulTextSrcOffset เป็นตำแหน่งของส่วนย่อยข้อความตัวแรกอ้างอิงกับข้อความทั้งหมด State คือ SAPI 5.0 XML ที่เกี่ยวข้องกับส่วนย่อยข้อความนั้น

```
typedef [restricted] struct SPVSTATE
{
        SPVACTIONS eAction;
        LANGID LangID;
        WORD wReserved;
        long EmphAdj;
        long RateAdj;
        ULONG Volume;
        SPVPITCH PitchAdj;
        ULONG SilenceMSecs;
        SPPHONEID *pPhoneIds;
        SPPARTOFSPEECH ePartOfSpeech;
        SPVCONTEXT Context;
} SPVSTATE
```

eActions เป็น enumerate value ที่ใช้บอกว่าเอ็นจิน ควรทำอย่างไรกับส่วนย่อยข้อความนี้ มีค่าที่เป็นไปได้ดังนี้

LANGID จะมีค่าเป็น 0 ยกเว้นแต่ว่า SAPI จะพบ <Lang> tag ใน ข้อความที่ผ่านเข้ามา

EmphAdj จะมีค่าเป็น 0 ยกเว้นแต่ว่า SAPI จะพบ <Emph> tag ใน ข้อความที่ผ่านเข้ามา

Volume จะมีค่าเป็น 100 ยกเว้นแต่ว่า SAPI จะพบ <Volume> tag ใน ข้อความที่ผ่านเข้ามา

RateAdj จะมีค่าเป็น 0 ยกเว้นแต่ว่า SAPI จะพบ <Rate> tag ใน ข้อความที่ผ่านเข้ามา

PitchAdj ประกอบด้วย MiddleAdj และ RangeAdj สามารถเซตได้ด้วย <Pitch> tag

```
typedef enum SPVACTIONS
{
        SPVA_Speak = 0,
        SPVA_Silence,
        SPVA_Pronounce,
        SPVA_Bookmark,
        SPVA_SpellOut,
```

```
        SPVA_Section,
        SPVA_ParseUnknownTag
} SPVACTIONS;
```

SPVA_Speak(ค่าโดยปริยาย) มีความหมายให้เอ็นจิน จัดการแปลงส่วนย่อยข้อความนั้นให้เป็นรูปแบบเอาท์พุทที่ถูกต้อง

SPVA_Silience ส่วนย่อยข้อความนั้นอยู่ใน <Silence> tag เอ็นจินควรจะเงียบตามจำนวนเวลาที่กำหนดไว้ใน SilenceMSecs(ดูที่ SPVSTATE)

SPVA_Pronounce มีความหมายว่าส่วนย่อยข้อความนั้นอยู่ใน <Pron> tag และ เอ็นจินควรจะใช้ pPhonIds(ดูที่ SPVSTATE)

SPV_Bookmark มีความหมายว่าส่วนย่อยข้อความนั้นอยู่ใน <Bookmark> tag เอ็นจินควรจะส่งค่าอีเวนต์ Bookmark ไปให้ SAPI

SPVA_SpellOut มีความหมายว่าเอ็นจินควรจะสังเคราะห์เสียงโดยสะกดทีละตัว

SPVA_Section ปัจจุบันนี้ยังไม่ใช้งาน

SPVA_ParseUnknownTag มีความหมายว่าเมื่อเจอ tag ที่ไม่รู้จัก ให้เอ็นจินตีความหมายเอง

```
typedef struct SPVPITCH
{
        long MiddleAdj;
        long RangeAdj;
} SPVPITCH;
```

ePartOfSpeech จะเป็น SPPS_Unknow เว้นแต่ว่า SAPI จะพบ <PartOfSp> tag ในข้อความที่ผ่านเข้ามา ePartOfSpeech ใช้กับคำที่มีความหมายกำกวมออกเสียงได้หลายแบบ เพื่อจะได้รู้ว่าจะออกเสียงอย่างไร

```
typedef [restricted] struct SPVCONTEXT
{
        LPCWSTR pCategory;
        LPCWSTR pBefore;
        LPCWSTR pAfter;
} SPVCONTEXT;
```

Context ใช้บ่งบอกว่าส่วนย่อยข้อความที่ผ่านเข้ามาเป็นข้อความแบบพิเศษ (เช่น วันที่ เวลา เบอร์โทรศัพท์)

ตัวอย่าง ข้อความที่ส่งไปยัง SAPI

"This is a <PITCH MIDDLE = '6'> sample piece of <PARTOFSP PART = 'Noun'> text </PARTOFSP> which will <BOOKMARK MARK = '1'/> demonstrate <VOLUME LEVEL = '30'> what a <VOLUME LEVEL = '90'> fragment </VOLUME> list </VOLUME> looks like </PITCH> conceptually."

จากตัวอย่าง ลิงค์ลิสต์ของ SPVTEXTFRAG ที่ SAPI ผ่านเข้ามายังฟังก์ชัน Speak สามารถแสดงได้ดังตารางต่อไปนี้

SPVTEXTFRAGs			Element 1	Element 2
pNext			Element 2	Element 3
State	eAction		SPVA_Speak	SPVA_Speak
	LangId		0	0
	EmphAdj		0	0
	RateAdj		0	0
	Volume		100	100
	PitchAdj	MiddleAdj	0	6
		RangeAdj	0	0
	SilenceMSecs		0	0
	pPhoneIds		NULL	NULL
	ePartOfSpeech		SPPS_Unknown	SPPS_Unknown
	Context	pCategory	NULL	NULL
		pBefore	NULL	NULL
		pAfter	NULL	NULL
pTextStart			“This is a <PITCH …“	“sample piece of <PART…”
ulTextLen			10	16
ulTextSrcOffset			0	31

ตารางที่ 5-1 แสดง ลิ้งลิสต์ของ SPVTEXTFRAG

SPVTEXTFRAGs			Element 3	Element 4	Element 5
pNext			Element 4	Element 5	Element 6
State	eAction		SPVA_Speak	SPVA_Speak	SPVA_Bookmark
	LangId		0	0	0
	EmphAdj		0	0	0
	RateAdj		0	0	0
	Volume		100	100	100
	PitchAdj	MiddleAdj	6	6	6
		RangeAdj	0	0	0
	SilenceMSecs		0	0	0
	pPhoneIds		NULL	NULL	NULL
	ePartOfSpeech		SPPS_Noun	SPPS_Unknown	SPPS_Unknown
	Context	pCategory	NULL	NULL	NULL
		pBefore	NULL	NULL	NULL
		pAfter	NULL	NULL	NULL
pTextStart			"text </PART…"	"which will <B…"	"1'/> demonstrate…"
ulTextLen			5	11	1
ulTextSrcOffset			72	89	100

ตารางที่ 5-1 แสดง สิ่งลิสต์ของ SPVTEXTFRAG (ต่อ)

SPVTEXTFRAGs			Element 6	Element 7	Element 8
pNext			Element 7	Element 8	Element 9
State	eAction		SPVA_Speak	SPVA_Speak	SPVA_Speak
	LangId		0	0	0
	EmphAdj		0	0	0
	RateAdj		0	0	0
	Volume		100	30	90
	PitchAdj	MiddleAdj	6	6	6
		RangeAdj	0	0	0
	SilenceMSecs		0	0	0
	pPhoneIds		NULL	NULL	NULL
	ePartOfSpeech		SPPS_Unknown	SPPS_Unknown	SPPS_Unknown
	Context	pCategory	NULL	NULL	NULL
		pBefore	NULL	NULL	NULL
		pAfter	NULL	NULL	NULL
pTextStart			“demonstrate <V…”	“what a <VOL…”	“fragment </VOL…”
ulTextLen			12	7	9
ulTextSrcOffset			123	157	186

ตารางที่ 5-1 แสดง สิ่งสิสต์ของ SPVTEXTFRAG (ต่อ)

SPVTEXTFRAGs			Element 9	Element 10	Element 11
pNext			Element 10	Element 11	Element 12
State	eAction		SPVA_Speak	SPVA_Speak	SPVA_Speak
	LangId		0	0	0
	EmphAdj		0	0	0
	RateAdj		0	0	0
	Volume		30	100	100
	PitchAdj	MiddleAdj	6	6	0
		RangeAdj	0	0	0
	SilenceMSecs		0	0	0
	pPhoneIds		NULL	NULL	NULL
	ePartOfSpeech		SPPS_Unknown	SPPS_Unknown	SPPS_Unknown
	Context	pCategory	NULL	NULL	NULL
		pBefore	NULL	NULL	NULL
		pAfter	NULL	NULL	NULL
pTextStart			"list </VOL…"	"looks like </PIT…"	"conceptually."
ulTextLen			5	11	14
ulTextSrcOffset			205	220	240

ตารางที่ 5-1 แสดง สิ่งสิสต์ของ SPVTEXTFRAG (ต่อ)

พารามิเตอร์ตัวสุดท้ายของฟังก์ชัน Speak เป็นพอยน์เตอร์ที่ชี้ไปที่ ISpTTSEngineSite pOutputSite ถูกใช้เพื่อส่งข้อมูลออดิโอและอีเวนต์กลับไปยัง SpVoice ออบเจ็กต์

5.5 การส่งข้อมูลกลับไปยัง SAPI (Write Data Back to SAPI –ISpTTSEngineSite)

5.5.1 รับการร้องขอแบบทันเวลา (Getting Real-Time Action Requests)

ภายในฟังก์ชัน Speak ตัวเอ็นจินควรจะเรียกเมธอด ISpTTSEngine::GetActions บ่อยที่สุดเท่าที่จะทำได้เพื่อให้รับแอคชั่นจาก SAPI ให้ทันเวลาฟังก์ชันนี้จะให้ค่า DWORD ซึ่งเป็นค่าอีนัมเมอร์เรชันของ SPVESACTION

```
DWORD  GetActions( void );
```

```
typedef enum SPVESACTIONS
{
        SPVES_CONTINUE = 0,
        SPVES_ABORT = ( 1L << 0 ),
        SPVES_SKIP = ( 1L << 1 ),
        SPVES_RATE = ( 1L << 2 ),
        SPVES_VOLUME = ( 1L << 3 )
} SPVESACTIONS;
```

SPVES_CONTINUE เป็นค่าโดยปริยาย หมายถึงให้เอ็นจินทำงานต่อไป

SPVES_ABORT หมายถึงให้เอ็นจินยกเลิกเมธอด Speak

SPVES_VOLUME หมายถึง Volume มีการเปลี่ยนแปลงให้เอ็นจินมารับค่า Volume ที่ถูกต้องไปด้วยฟังก์ชัน GetVolume

```
HRESULT GetVolume(
        [out] USHORT *pusVolume
);
```

SPVES_RATE หมายถึง ค่า Rate มีการเปลี่ยนแปลงให้เรียกฟังก์ชันต่อไปนี้

```
HRESULT GetRate(
        [out] long *pRateAdjust
);
```

SPVES_SKIP หมายถึง SAPI ต้องการให้เอ็นจินข้ามการสังเคราะห์เสียง ต้องเรียกฟังก์ชันต่อไปนี้มารับข้อมูลว่าจะข้ามไปเท่าไร

```
HRESULT GetSkipInfo(
        [out] SPVSKIPTYPE *peType,
        [out] long *plNumItems
);
```

และตอบรับจำนวน item ที่ข้ามไปด้วยฟังก์ชัน CompleteSkip

```
HRESULT CompleteSkip(
        [in] long ulNumSkipped
);
```

5.5.2 การจัดลำดับอีเวนต์ (Queing Events)

อีเวนต์คือข้อมูลที่เอ็นจิน ส่งกลับไปให้ แอพพลิเคชั่น โดยส่งไปให้ SAPI รับผิดชอบว่าจะส่งอีเวนต์ไปแอพพลิเคชั่นในเวลาที่เหมาะสม ด้วยฟังก์ชันต่อไปนี้

```
HRESULT AddEvents(
        [in] const SPEVENT* pEventArray,
        [in] ULONG ulCount
);
```

เอ็นจินสามารถรับรู้อีเวนต์ที่ SAPI สนใจได้ด้วยฟังก์ชัน GetEvenInterest

```
HRESULT GetEventInterest(
        [out] ULONGLONG * pullEventInterest
);
```

ฟังก์ชันนี้จะให้ค่ากลับมาทาง pullEventInterest ซึ่งเป็นค่าที่อยู่ใน SPEVENTENUM อีนัมเมอเรชั่นดังนี้

- SPEI_TTS_BOOKMARK
- SPEI_WORD_BOUNDARY
- SPEI_SENTENCE_BOUNDARY
- SPEI_PHONEME
- SPEI_VISEME

เอ็นจินจะสร้างอีเวนต์ที่มีโครงสร้างดังต่อไปนี้

```
typedef [restricted] struct SPEVENT
{
        WORD eEventId;
        WORD elParamType;
        ULONG ulStreamNum;
        ULONGLONG ullAudioStreamOffset;
        WPARAM wParam;
        LPARAM lParam;
} SPEVENT;
```

สำหรับ ulStreamNum นั้นไม่จำเป็นต้องจัดการเพราะ SAPI จะเป็นคนดูแลให้เอง

ต่อไปนี้จะเป็นการแสดงค่าของ SPEVENT ตามชนิดของอีเวนต์ต่างๆ

SPEI_TTS_BOOKMARK เพื่อบ่งบอกว่า TTS engine พบกับ <Bookmark> tag แล้ว ซึ่งมีฟิลด์ข้อมูลดังนี้

SPEVENT Field	**Bookmark event**
eEventId	SPEI_TTS_BOOKMARK
elParamType	SPET_LPARAM_IS_STRING
wParam	ค่าของบุคส์มาร์คสตริงเมื่อเปลี่ยนเป็นค่าตัวเลข (_wtol(...) ใช้ได้)
lParam	บุคส์มาร์คสตริงที่ลงท้ายด้วย null

ตารางที่ 5-2 ตารางอีเวนต์ของ SPEI_TTS_BOOKMARK

SPEI_TTS_WORD_BOUNDARY บ่งบอกว่า TTS engine กำลังสังเคราะห์เสียงระดับคำอยู่

SPEVENT Field	**Word Boundary event**
eEventId	SPEI_TTS_WORD_BOUNDARY
elParamType	SPET_LPARAM_IS_UNKNOWN
wParam	ตำแหน่งตัวแรกที่เริ่มสังเคราะห์เสียง
lParam	ความยาวของคำ

ตารางที่ 5-3 ตารางอีเวนต์ของ SPEI_TTS_WORD_BOUNDARY

SPEI_TTS__SENTENCE_BOUNDARY บ่งบอกว่า TTS engine กำลังสังเคราะห์เสียงระดับประโยคอยู่

SPEVENT Field	Sentence Boundary event
eEventId	SPEI_TTS_SENTENCE_BOUNDARY
elParamType	SPET_LPARAM_IS_UNKNOWN
wParam	ตำแหน่งเริ่มต้นในการ สังเคราะห์เสียง
lParam	ความยาวของประโยค

ตารางที่ 5-4 ตารางอีเวนต์ของ SPEI_TTS_SENTENCE_BOUNDARY

SPEI_TTS_PHONEME บ่งบอกว่า TTS engine เริ่มสังเคราะห์เสียงโฟนีม

SPEVENT Field	Phoneme event
eEventId	SPEI_TTS_PHONEME
elParamType	SPET_LPARAM_IS_UNKNOWN
wParam	ไฮท์เวิร์ด (Hight word)เป็น ระยะเวลา ในหน่วยมิลลิวินาที. โลว์เวิร์ด (Low word) เป็น PhoneID ของตัวถัดไป
lParam	ไฮท์เวิร์ด เป็น SPVFEATURE ของโฟนีมปัจจุบัน โลว์เวิร์ดเป็น PhoneID ของโฟนีมปัจจุบัน

ตารางที่ 5-5 ตารางอีเวนต์ของ SPEI_TTS_PHONEME

SPVFEATURE สามารถเป็นได้ 2 ค่า คือ SPVFEATURE_STRESSED ซึ่งหมายถึงเน้นเฉพาะสระ และ SPVFEATURE_EMPHASIS ซึ่งหมายถึงเน้นทั้งคำ

SPEI_TTS_VISEME บ่งบอกว่า TTS engine สังเคราะห์ viseme

SPEVENT Field	Viseme event
eEventId	SPEI_TTS_VISEME
elParamType	SPET_LPARAM_IS_UNKNOWN
wParam	ไฮท์เวิร์ด เป็น ระยะเวลา ในหน่วยมิลลิวินาที. โลว์เวิร์ดเป็น code ของตัวถัดไป
lParam	ไฮท์เวิร์ด เป็น SPVFEATURE ของโฟนีมปัจจุบัน โลว์เวิร์ดเป็น code ของโฟนีมปัจจุบัน

ตารางที่ 5-6 ตารางอีเวนต์ของ SPEI_TTS_VISEME

5.5.3 การจัดลำดับให้กับข้อมูลออดิโอ (Queing Audio Data)

หลังจากจัดลำดับของอีเวนต์แล้ว ขั้นต่อไปก็ส่งข้อมูลกลับไปยัง SAPI ด้วย

```
HRESULT Write(
        const void* pBuff,
        ULONG cb,
        ULONG *pcbWritten
);
```

pBuff เป็นพอยน์เตอร์ชี้ไปที่ข้อมูลออดิโอที่จะส่งไปให้ SAPI

cb เป็นจำนวนไบต์ที่ส่งไป และ

pcbWritten คือค่าที่แท้จริงที่ส่งไปได้สำเร็จ

บทที่ 6

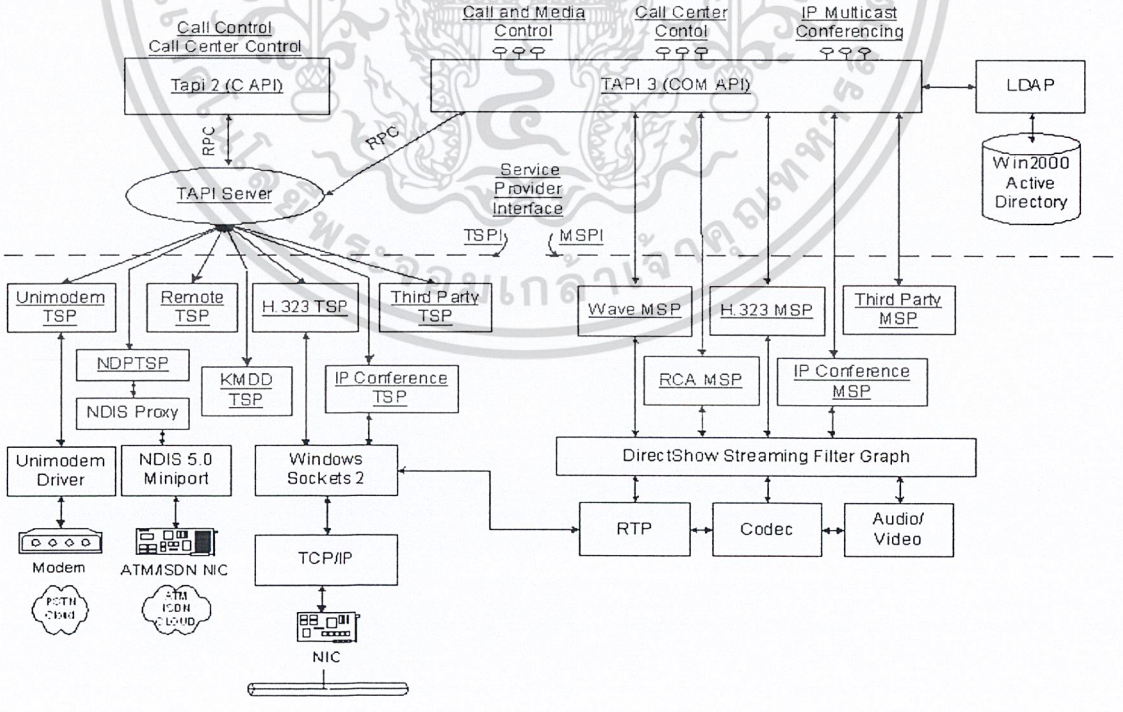
Telephony API (TAPI)

6.1 บทนำ

TAPI เป็น API ที่เกี่ยวข้องกับการโทรศัพท์ซึ่งไม่จำเป็นจะต้องผ่านเครือข่าย PSTN เท่านั้น แต่สามารถที่จะใช้กับเครือข่ายอินเทอร์เนตหรือเครือข่ายไอพีก็ได้ การใช้ TAPI ทำให้กำจัดความยุ่งยากในการพัฒนาระดับรายละเอียดในแอพพลิเคชั่นต่าง ๆ ลงได้

6.2 ภาพรวมเทเลโฟนนีของไมโครซอฟท์

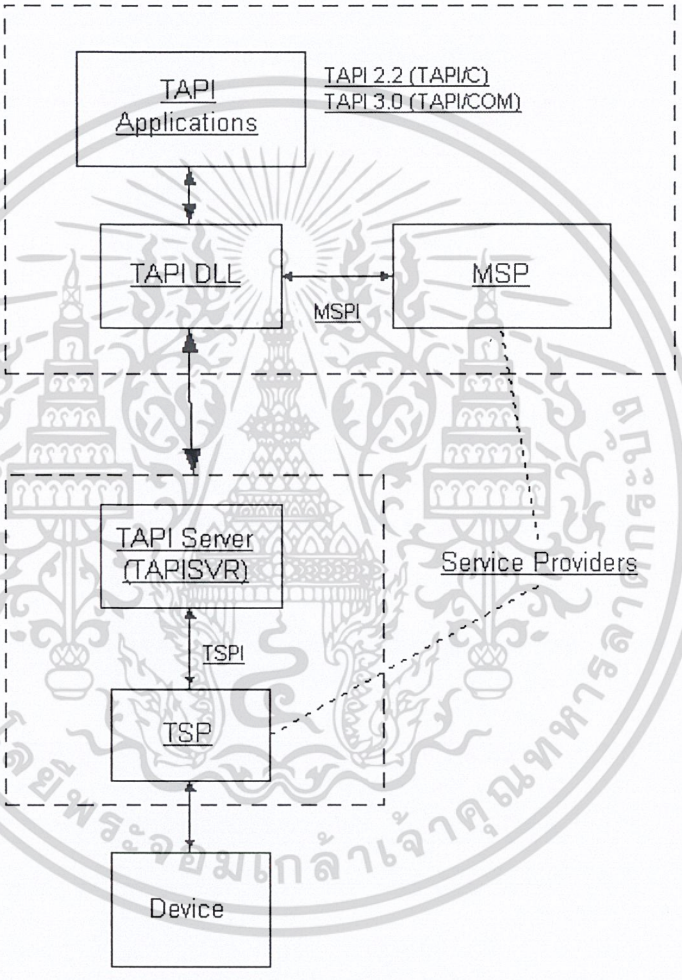
Telephony Application Programming Interface (TAPI), Service Provider Interface (TSPI) และ Media Service Provider (MAPI) ของไมโครซอฟท์สนับสนุนการพัฒนาแอพพลิเคชั่นของการติดต่อสื่อสารที่ทำงานบนระบบปฏิบัติการที่สนับสนุน Microsoft Win32 API เช่น Microsoft Windows 98 และ Microsoft Windows 2000 จากรูปที่ 6.1 แสดงถึงสถาปัตยกรรมของเทเลโฟนนีของไมโครซอฟท์ ซึ่ง TAPI สามารถนำมาใช้กับระบบ PATN, ISDN และการติดต่อแบบ TCP/IP อีกทั้งยังสนับสนุนมาตรฐาน H.323 ด้วย



รูปที่ 6-1 สถาปัตยกรรมของไมโครซอฟท์เทเลโฟนนี

6.3 โมเดลการเขียนโปรแกรมเทเลโฟนนีของไมโครซอฟท์

รูปบบโมเดลการเขียนโปรแกรมโดยใช้เทคโนโลยีของไมโครซอฟท์ที่เรียกว่า Microsoft Telephony Programming Model ดังรูปที่ 6.2 เป็นการจำลองการควบคุมการติดต่อจากส่วนที่ควบคุมอุปกรณ์และแอพพลิเคชั่นทางฝั่งของผู้ใช้ จากการใช้โมเดลนี้ทำให้แอพพลิเคชั่นไม่จำเป็นจะต้องรู้ถึงข้อมูลในระดับรายละเอียดในการควบคุมอุปกรณ์ และอุปกรณ์ไม่จำเป็นต้องถูกจัดการโดยแอพพลิเคชั่นโดยตรง ทำให้สามารถเปลี่ยนแปลงตัวแอพพลิเคชั่นหรืออุปกรณ์สื่อสารได้โดยไม่ต้องเปลี่ยนแปลงอีกส่วนหนึ่ง



รูปที่ 6-2 โครงสร้างของ Microsoft Telephony Programming Model

โดยจากรูปที่ 6.2 สามารถแบ่งส่วนประกอบต่าง ๆ ได้เป็น TAPI Application, TAPI DLL, TAPI Server, TSP และ MSP ออกมาเป็นรายละเอียดดังนี้

6.3.1 TAPI Application

TAPI Application เป็นแอพพลิเคชั่นที่จะคำนึงถึงเพียงความต้องการของผู้ใช้ ขณะเดียวกันส่วนของ TAPI DLL และ TAPI Server (TAPISVR) จะจัดการในการติดต่อระหว่างไคล์เอนต์และเซิร์ฟเวอร์ และส่วนของตัวให้บริการ (Service Providers) จะทราบถึงรายละเอียดในการควบคุมอุปกรณ์ต่าง ๆ

6.3.2 TAPI DLL

TAPI DLL ใช้ร่วมกับ TAPI Server (Tapisrv.exe) เป็นส่วนสำคัญที่แบ่งแอพพลิเคชั่นของผู้ใช้ปลายทางและเซิร์ฟเวอร์ออกจากส่วนของ Service Provider ซึ่งแอพพลิเคชั่น TAPI จะโหลด DLL ที่จำเป็นไว้ในหน่วยความจำของโพรเซสระหว่างการทำ initialization TAPI จะสร้างการติดต่อแบบ RPC กับ Tapisrv.exe ซึ่งมีไฟล์ DLLs ทั้งหมด 3 ไฟล์ที่เกี่ยวข้องกับ TAPI ได้แก่ Tapi.dll, Tapi32.dll และ Tapi3.dll

6.3.3 TAPI Server

TAPI Server (Tapisrv.exe) เป็นศูนย์กลางที่เก็บของข้อมูลเกี่ยวกับเทเลโฟนนีของเครื่องผู้ใช้ โพรเซสของบริการนี้จะจัดการกับทรัพยากรของเทเลโฟนนีทั้งแบบโลคอลและรีโมต และจัดการให้แอพพลิเคชั่นลงทะเบียนเพื่อที่จะรับรองการร้องขอและระหว่างการรอคอยฟังก์ชันแบบอะซิงโครนัส รวมทั้งสร้างอินเทอร์เฟซกับ TSP ตามรูปที่ 4.2 ของ Microsoft Telephony Programming Model

6.3.4 Service Providers

Service Providers จะอิมพลีเมนต์การควบคุมอุปกรณ์ในส่วนของรายละเอียดของเทเลโฟนนี โดยแบ่งเป็น Telephony Service Provider (TSP) และ Media Service Provider (MSP) ซึ่ง TSP จะทำหน้าที่ในการบริการของการควบคุมในการ Call และ MSP จะเป็นตัวจัดการกับข้อมูลของมีเดีย โดยทุก TSP จะถูกประมวลผลในโพรเซสของ Tapisrv.exe ซึ่งสามารถสร้างเธรดใน TAPISRV เพื่อใช้ในการทำงานตามที่ต้องการ และมั่นในได้ว่าไม่มีทรัพยากรที่ถูกสร้างขึ้นจะสามารถถูกทำลายโดยการออกของแอพพลิเคชั่นใดๆ TAPI Server จะแปลคำสั่งของแอพพลิเคชั่นเป็นคำสั่งที่รู้จักในนามของ Telephony Service Provider Interface (TSPI) สำหรับ MSP จะทำงานในพื้นที่โพรเซสของแอพพลิเคชั่น ซึ่งทำให้มีการตอบรับที่รวดเร็วซึ่งบางครั้งจำเป็นในการควบคุมตัวมีเดียด้วย ตัว TAPI DLL จะทำงานร่วมกับ Media Service Provider Interface (MSPI)

6.4 การควบคุมมีเดียและการ Call ใน TAPI3.0

มีเดียและการ Call ของ TAPI3.0 เป็นกลุ่มของ COM ออบเจ็กต์ อินเทอร์เฟซ และเมธอดสำหรับทำการ Call ระหว่างเครื่องตั้งแต่ 2 เครื่องขึ้นไป การ call ของ TAPI 3.0 ไม่ได้หมายถึงการส่งเสียงผ่านเครือข่ายโทรศัพท์เท่านั้น แต่หมายถึงตัวกลางและวิธีในการส่งทุกๆ ชนิดที่ผู้ให้บริการสามารถให้การสนับสนุนได้

ออบเจ็กต์หลัก 5 ตัว ในสถาปัตยกรรมของการควบคุมมีเดียและการ Call คือ TAPI, Address, Terminal, Call และ CallHubในรูปจะแสดงออบเจ็กต์เหล่านี้และอินเทอร์เฟซต่างๆ ที่สัมพันธ์กัน

6.4.1 ออบเจ็กต์ TAPI

ออบเจ็กต์ TAPI แสดงถึงทรัพยากรของเทเลโฟนนี่ทั้งหมดที่คอมพิวเตอร์สามารถเข้าถึงได้ ทำให้แอพพลิเคชั่นสามารถระบุแอเดรสโลคอลและรีโมตได้ทั้งหมด แอพพลิเคชั่น TAPI 3.0 ต้องสร้างอินแสตนท์ของออบเจ็กต์ TAPI และทำการ Initialize มันด้วย

6.4.2 ออบเจ็กต์ Address

ออบเจ็กต์ Address หมายถึงสิ่งที่แสดงว่าสามารถที่จะทำและรับการ Call ได้ ออเจ็กต์นี้มีอินเทอร์เฟซและเมธอดที่อนุญาต ให้แอพพลิเคชั่นกระทำสิ่งต่างๆ ได้เช่น

- สามารถแสดงว่าแอดเดรสที่กำหนดให้สามารถสนับสนุนชนิดของมีเดียที่ต้องการได้หรือไม่
- ระบุการ Call ปัจจุบันที่สัมพันธ์กับแอดเดรส
- สร้างหรือทำการส่งต่อการ Call ได้
- แสดงชื่อของผู้ให้บริการที่เกียวข้องได้
- ถ้ามี MSP อยู่ จะรับพอยน์เตอร์ของอินเทอร์เฟซที่อนุญาตกระทำกับเทอร์มินอลในระดับรายระเอียดได้
- รับและเซตรายละเอียดอื่นๆ เช่น มีข้อความรออยู่หรือไม่

6.4.3 ออบเจ็กต์ Terminal

ออบเจ็กต์ Terminal แสดงถึงแหล่งกำเนิด (source) หรือตัวแปลสัญญาณ (Render) เช่น ไมโครโฟนหรือลำโพง เป็นต้น แอพพลิเคชั่นจะเลือกเทอร์มินอลที่มีขึ้นอยู่กับทิศทางของมีเดียและชนิดของเซสชั่นในการติดต่อสื่อสาร ข้อมูลแต่ละมีเดียที่เกี่ยวข้องก็จะถูกเลือกให้เข้ากับเทอร์มินอลที่เหมาะสมเพื่อที่จะเริ่มทำสตรีมมิ่ง

6.4.4 ออบเจ็กต์ Call

ออบเจ็กต์ Call แสดงถึงการติดต่อของแอดเดรสระหว่างแอดเดรสโลคอลกับแอดเดรสอื่น ๆ การควบคุมการ Call ทั้งหมดจะถูกทำผ่านออบเจ็กต์ Call ซึ่ง ITBasicCallCntrol และ ITCallInfo เป็นอินเทอร์เฟซที่ใช้บ่อยที่สุดของออบเจ็กต์ Call อินเทอร์เฟซเหล่านี้อิมพลีเมนต์การทำงานและการร้องขอต่าง ๆ เช่น ขอพอยน์เตอร์ของอินเทอร์เฟซสำหรับข้อมูลของมีเดีย

6.4.5 ออบเจ็กต์ CallHub

ออบเจ็กต์ CallHub แสดงถึงลักษณะการมองของเธิร์ดพาร์ตี้ของการ Call จากผู้ผลิตหลายราย อินเทอร์เฟซและเมธอดที่เกี่ยวข้องด้วยจะรับและเซตข้อมูลที่เกี่ยวข้องกับศูนย์กลาง (hub) เช่นมันกำลังทำงานอยู่หรือไม่ การใช้ออบเจ็กต์ CallHub ผู้ใช้ที่ต้องการความปลอดภัยของข้อมูลสามารถพบและควบคุมผู้สนทนาคนอื่น ๆ ในการ Call ได้ ออบเจ็กต์ CallHub ไม่สามารถถูกสร้างขึ้นโดยตรงได้โดยแอพพลิเคชั่น แต่จะถูกสร้างทางอ้อมเมื่อมีการ Call เข้ามาแล้วถูกรับผ่าน TAPI 3.0 ซึ่ง TAPI 3.0 อาศัยผู้ให้บริการ TAPI เพื่อจัดหาข้อมูลที่จำเป็นเกี่ยวกับ Call ต่าง ๆ เพื่อที่จะนำมาอิมพลีเมนต์โดยใช้ออบเจ็กต์ CallHub เนื่องจากไม่ใช่ผู้ให้บริการทั้งหมดที่จะจัดหาข้อมูลเหล่านี้ให้และไม่ใช่ฮาร์ดแวร์ทั้งหมดที่จะสามารถติดตาม CallHub ได้ข้อมูลที่เกี่ยวกับผู้ใช้คนอื่น ๆ ในการติดต่ออาจจะถูกจำกัดหรือไม่มีอยู่เลย

6.4.6 ออบเจ็กต์ Stream

ออบเจ็กต์ Stream เป็นแอ็บสแทรกต์ของข้อมูลมีเดียหรือข้อมูลอื่น ๆ ที่เกี่ยวข้องกับเซสชันการ Call อินเทอร์เฟซและเมธอดต่าง ๆ ที่ถูกเปิดใช้ให้กับออบเจ็กต์สตรีมและสตรีมย่อย (substream) อนุญาตให้แอพพลิเคชั่นสามารถเข้าถึงการควบคุมในรายละเอียดได้ เช่น การหยุดสตรีมชั่วคราว เพิ่มชนิดของมีเดียให้กับเซสชันในการติดต่อสื่อสาร หรือปรับระดับเสียงของผู้ร่วมสนทนา

6.4.7 อีเวนต์

อีเวนต์เป็นส่วนที่สำคัญการควบคุมการ Call ภายใต้ TAPI 3.0 ซึ่งประกอบด้วย 4 ขั้นตอน โดยในการลงทะเบียนและรองรับการทำงานของอีเวนต์จะต้องทำดังนี้

1. ทำการอิมพลีเมนต์เมธอด ITTAPIEventNotification::Event (TAPI จะ call เมธอดนี้เมื่อมีอีเวนต์เกิดขึ้น) โดยปกติแล้วการอิมพลีเมนต์นี้ไม่ได้ทำไปมากกว่าการทำ AddRef กับพอยน์เตอร์อินเทอร์เฟซของ IDispatch แล้วจึงส่งให้กับตัวสร้างเมสเซจของแอพพลิเคชั่น
2. ลงทะเบียนอินเทอร์เฟซ ITTAPIEventNotification ที่ออกไปโดยใช้มาตรฐาน COM ของอินเทอร์เฟซ IConnectionPointContainer และ IConnectionPoint แล้วทำการส่งพอยน์เตอร์ของเมธอด IConnectionPointContainer ::Advise ไปยัง ITTAPIEventNotification::Event
3. ทำการเรียกเมธอด ITTAPI::put_EventFilter เพื่อที่จะบอก TAPI ว่าอีเวนต์ไหนบ้างที่แอพพลิเคชั่นจะรองรับ ตัวกรองอีเวนต์ (event filter) จะประกอบไปด้วยสมาชิก ORed ของตัวระบุ TAPI_EVENT ซึ่งจะต้องทำการเรียกเมธอด ITTAPI::put_EventFilter เพื่อที่จะเซตตัวกรองอีเวนต์และรองรับการทำงานของอีเวนต์ ถ้าหากไม่ทำการเรียก ITTAPI::put_EventFilter แล้ว ตัวแอพพลิเคชั่นจะไม่สามารถรับอีเวนต์ใด ๆ ได้เลย

6.4.8 ออบเจ็กต์ Request

ออบเจ็กต์ Request ถูกสร้างโดยใช้ CoCreateInstance ของ COM และมีอินเทอร์เฟซตัวเดียวคือ ITRequest อินเทอร์เฟซนี้จะมีเมธอด MakeCall ซึ่งอนุญาตให้แอพพลิเคชั่น TAPI 3.0 สามารถใช้ตัวช่วยเทเลโฟนนี่ (Assisted Telephony) ได้ ตัวช่วยเทเลโฟลนี่จะจัดหาแอพพลิเคชั่นที่มีความสามารถของเทเลโฟนนี่โดยใช้กลไกที่ไม่สลับซับซ้อนนการโทรศัพท์ ทำให้ผู้พัฒนาไม่จำเป็นต้องรู้เทเลโฟลนี่มากนัก ตัวอย่างเช่น แอพพลิเคชั่นจำพวกสเปรดชีตสามารถเพิ่มปุ่ม Make Call โดยใช้ตัวช่วยเทเลโฟนนี่ได้โดยง่าย ซึ่งไม่จำเป็นต้องทำการอิมพลีเมนต์ตัวควบคุมที่มีในแอพพลิเคชั่น TAPI ทั้งหมด

6.4.9 Dispatch Mapper

Dispatch Mapper ถูกสร้างใช้ CoCreateInstance ของ COM และมีอินเทอร์เฟซตัวหนึ่งมาด้วยคือ ITDispatchMapper อินเทอร์เฟซนี้อนุญาตให้แอพพลิเคชั่นรับดิสแพตช์พอยน์เตอร์ (dispatch pointer) ของอินเทอร์เฟซอีกตัวบนออบเจ็กต์หนึ่งมาให้กับอินเทอร์เฟซนั้นและ GUID ของอีกอินเทอร์เฟซหนึ่ง อินเทอร์เฟซนี้ถูกสร้างขึ้นเพื่อช่วยโปรแกรมเมอร์ใช้ในการเขียนแอพพลิเคชั่น ซึ่งไม่ได้ช่วยในการดึงอินเทอร์เฟซของออบเจ็กต์มาใช้ Dispatch Mapper นี้ใช้อินเทอร์เฟซ IobjectSafety ของออบเจ็กต์เพื่อให้แน่ใจว่าออบเจ็กต์นั้นมีความปลอดภัยสำหรับการเขียนอินเทอร์เฟซที่ต้องการ ถ้าออบเจ็กต์ไม่ทำการอิมพลีเมนต์ IobjectSafety หรือถ้าออบเจ็กต์นั้นไม่ปลอดภัยกับอินเทอร์เฟซนี้ การ Call ก็จะล้มเหลว

บทที่ 7

Post Office Protocol – Version 3

7.1. บทนำ

ในระบบเครือข่ายอินเทอร์เน็ตโหนด (Node) ซึ่งเป็นส่วนหนึ่งของเครือข่ายมักจะมีปัญหาในการจัดการกับระบบการรับส่งข้อมูล (Message Transport System; MTS) ตัวอย่างเช่น เครื่องเวิร์กสเตชันส่วนมากจะมีทรัพยากรไม่เพียงพอสำหรับกระบวนการขออนุญาตเครื่องเซิร์ฟเวอร์ (SMTP server) และเชื่อมต่อกับระบบรับส่งเมล์เพื่อที่ทำให้การเชื่อมต่อให้ดำเนินต่อไป เช่นเดียวกับเครื่องคอมพิวเตอร์ส่วนบุคคลการเชื่อมต่อแบบไอพีที่ต้องเชื่อมต่อเป็นเวลานานก็มีปัญหาเหมือนกัน (ปัญหาการขาดแคลนทรัพยากรสำหรับการการเชื่อมต่อเรียกว่า "Connectivity")

ถึงอย่างไรก็ตามถ้าโหนดสามารถจัดการเมล์ได้นั้นจะเป็นประโยชน์อย่างยิ่ง และโหนดเหล่านี้ต้องรองรับยูสเซอร์เอเจนต์ (User Agent; UA) ในการจัดการกับงานที่เกี่ยวกับเมล์ เพื่อแก้ไขปัญหาในจุดนี้โหนดจะต้องมี MTS เอนติตีคอยให้บริการเมล์ดรอป (Maildrop) โพรโตคอล POP3 มีจุดมุ่งหมายเพื่อที่จะอนุญาตให้เครื่องเวิร์กสเตชันเข้าถึงบริการเมล์ดรอปบนเครื่องเซิร์ฟเวอร์ได้อย่างไดนามิก นั้นก็แสดงว่าโพรโตคอล POP3 อนุญาตให้เครื่องเวิร์กสเตชันสามารถดูข้อมูลของเมล์จากเครื่องเซิร์ฟเวอร์ที่คอยให้บริการอยู่ได้

โพรโตคอล POP3 ไม่ได้ดูแลเฉพาะเมล์ที่อยู่บนเครื่องเซิร์ฟเวอร์เท่านั้น แต่ยังรวมถึงเมล์ที่ถูกดาวน์โหลดและเมล์ที่ถูกลบทิ้งด้วย

7.2. การทำงานพื้นฐาน

เริ่มต้นฝั่งเซิร์ฟเวอร์จะเริ่มให้บริการ POP3 โดยการคอยฟัง (Listening) ด้วยโพรโตคอล TCP พอร์ต 110 เมื่อไคล์เอ็นต์ต้องการขอใช้บริการก็จะเริ่มสร้างคอนเน็กชันกับฝั่งเซิร์ฟเวอร์ด้วยโพรโตคอล TCP เมื่อสร้างคอนเน็กชันสำเร็จฝั่งเซิร์ฟเวอร์จะส่งสัญญาณเริ่มต้นมายังฝั่งไคล์เอ็นต์ หลังจากนั้นฝั่งเซิร์ฟเวอร์กับฝั่งไคลเอ็นต์จะสามารถแลกเปลี่ยนคำสั่ง (Command) และผลตอบรับ (Response) ซึ่งกันและกันตราบจนกระทั่งคอนเน็กชันที่สร้างขึ้นนั้นจะถูกปิดหรือถูกยกเลิก

คำสั่งต่าง ๆ ในโพรโตคอล POP3 จะประกอบด้วยคีย์เวิร์ดซึ่งบางทีอาจจะมีอาร์กิวเมนต์ตามหลังหนึ่งตัวหรืออาจมากกว่าหนึ่งตัว คำสั่งทุกคำสั่งจะลงท้ายด้วย CRLF คีย์เวิร์ดและอาร์กิวเมนต์จะประกอบด้วยตัวอักษรที่เป็น ASCII ที่สามารถพิมพ์ได้ คีย์เวิร์ดและอาร์กิวเมนต์จะถูกแยกจากกันด้วยช่องว่าง (SPACE) คีย์เวิร์ดจะประกอบด้วยตัวอักษรที่มีความยาวตั้งแต่ 3 - 4 ตัวอักษร ส่วนอาร์กิวเมนต์อาจมีตัวอักษรได้ถึง 40 ตัวอักษร

ผลตอบรับใน POP3 จะประกอบด้วยสถานะและคีย์เวิร์ดและบางทีอาจตามด้วยข้อมูลเพิ่มเติม ในทุก ๆ ผลตอบรับจะลงท้ายด้วย CRLF ผลตอบรับอาจมีตัวอักษรได้ถึง 512-ตัวอักษรรวม CRLF แล้ว สำหรับสถานะแล้วจะมีด้วยกัน 2 สถานะคือ ยืนยัน (+OK) และ ปฏิเสธ (-ERR) เซิร์ฟเวอร์จะต้องส่งสถานะเป็นตัวพิมพ์ใหญ่เสมอ

ในกรณีที่ผลตอบรับมีหลายบรรทัด ในแต่ละบรรทัดจะปิดท้ายด้วย CRLF หลังจากที่ทุกบรรทัดของผลตอบรับถูกส่งไปจนหมดในบรรทัดสุดท้ายจะต้องประกอบด้วยตัวปิดสตริง "." และตามด้วย CRLF และเรียกบรรทัดนั้นว่า "byte-stuffed" ไคล์เอ็นต์จะต้องตรวจสอบว่าบรรทัดใดที่เริ่มต้นด้วยตัวปิดสตริงและตัวที่ตามมาเป็น CRLF แสดงว่าเป็นบรรทัดสุดท้ายของผลตอบรับ

เซสชันของ POP3 นั้นขึ้นอยู่กับสเตท (State) เมื่อเริ่มต้นสร้างคอนเน็กชันและ POP3 เซิร์ฟเวอร์ส่งสัญญาณเริ่มต้นแล้วเซสชันจะอยู่ที่สเตทพิสูจน์สิทธิ์ (AUTHORIZATION) ในสเตทนี้ไคล์เอ็นต์จะต้องพิสูจน์สิทธิ์กับเซิร์ฟเวอร์ หลังจากพิสูจน์สิทธิ์เรียบร้อยแล้ว เซิร์ฟเวอร์จะยอมให้ไคล์เอ็นต์ได้เข้ามาใช้ทรัพยากรในการใช้บริการเมล์ดรอปได้เรียกสเตทนี้ว่าทรานส์แซ็กชัน (TRANSACTION) ในสเตทนี้ไคล์เอ็นต์จะสามารถทำกิจกรรมบนเซิร์ฟเวอร์ได้ตามที่ร้องขอไป เมื่อไคล์เอ็นต์ส่งคำสั่ง QUIT เซสชันก็จะเข้าสู่สเตทอัพเดต (UPDATE) ในสเตทนี้เซิร์ฟเวอร์จะปลดปล่อยทรัพยากรที่ได้ให้กับไคล์เอ็นต์ระหว่างสเตททรานส์แซ็กชันและส่งสัญญาณยกเลิกการติดต่อ คอนเน็กชันก็จะถูกปิด

เซิร์ฟเวอร์จะต้องสามารถตอบรับสำหรับคำสั่งที่ไม่รู้จัก ไม่ได้ทำการอิมพลีเมนต์หรือคำสั่งที่ผิดโครงสร้าง ซึ่งจะต้องตอบรับด้วยสถานะปฏิเสธ เซิร์ฟเวอร์จะต้องตอบรับคำสั่งที่อยู่ผิดสเตทด้วยสถานะปฏิเสธเช่นเดียวกัน ไคล์เอ็นต์จะไม่สามารถแยกแยะความแตกต่างระหว่างคำสั่งที่ไม่ได้ทำการอิมพลีเมนต์และคำสั่งที่ไม่สามารถทำงานได้

POP3 เซิร์ฟเวอร์อาจจะต้องมีออโตล็อกเอาท์ไทม์เมอร์ (Autologout Timer) ซึ่งไทม์เมอร์จะต้องมีช่วงเวลาอย่างน้อย 10นาที ถ้ามีการตอบสนองคำสั่งในช่วงเวลาที่กำหนดก็จะมีการรีเซต (Reset) ไทม์เมอร์ใหม่ แต่ถ้าถึงกำหนดไทม์เมอร์แล้วยังไม่เข้าสเตทอัพเดตเซิร์ฟเวอร์ควรจะปิดคอนเน็กชันโดยไม่มีการเคลื่อนย้ายข้อมูลหรือส่งผลตอบรับไปยังไคล์เอ็นต์

7.3. สเตทพิสูจน์สิทธิ์

เมื่อเริ่มมีกาสร้างคอนเน็กชันเซิร์ฟเวอร์จะส่งสัญญาณเริ่มต้นหนึ่งบรรทัดมายังฝั่งไคล์เอ็นต์ ตัวอย่างเช่น

S: +OK เซิร์ฟเวอร์ POP3 พร้อมสำหรับให้บริการเมล์ดรอป

เซสชันในตอนนี้จะเป็นสเตทพิสูจน์สิทธิ์ ไคล์เอ็นต์จะต้องยืนยันสิทธิ์ของตนเองกับเซิร์ฟเวอร์ มีกลไกที่เป็นไปได้ในการพิสูจน์สิทธิ์ 2 วิธีคือใช้คำสั่ง USER รวมกับ PASS และคำสั่ง APOP เซิร์ฟเวอร์จะต้องรองรับอย่างน้อยหนึ่งวิธีในการพิสูจน์สิทธิ์จากทั้งสองวิธี

เซิร์ฟเวอร์จะต้องพิจารณาคำสั่งในการพิสูจน์สิทธิ์ว่าควรจะให้ไคล์เอ็นต์เข้าถึงบริการเมล์ดรอปหรือไม่ หลังจากนั้นเซิร์ฟเวอร์จะทำการล็อกเมล์ดรอปเพื่อป้องกันการแก้ไขข้อมูลหรือเคลื่อนย้ายข้อมูลก่อนที่เซสชันจะเข้าสู่สเตทอัพเดต ถ้าสามารถทำการล็อกเมล์ดรอปได้สำเร็จเซิร์ฟเวอร์จะขานรับด้วยสถานะยืนยัน แล้วเซสชันก็จะเข้าสู่สเตททรานส์แซ็กชัน แต่ถ้าไม่สามารถใช้บริการเมล์ดรอปได้ด้วยสาเหตุใด ๆ ก็ตาม เช่น ไม่สามารถล็อกเมล์ดรอปได้ ไคล์เอ็นต์ถูกปฏิเสธในการเข้าถึงเมล์ดรอป หรือแม้แต่เมล์ดรอปไม่สามารถแสดงผลได้ เซิร์ฟเวอร์จะต้องขานรับด้วยสถานะปฏิเสธ ถ้าเกิดว่าสามารถล็อกเมล์ดรอปได้แต่เซิร์ฟเวอร์แสดงสถานะปฏิเสธ เซิร์ฟเวอร์จะทำการปลดล็อกก่อนแล้วจึงค่อยปฏิเสธคำสั่งนั้น หลังจากที่แสดงสถานะปฏิเสธเซิร์ฟเวอร์ก็จะทำการปิดคอนเน็กชันเพราะถ้าไม่ปิดคอนเน็กชันไคล์เอ็นต์อาจส่งคำสั่งในการพิสูจน์สิทธิ์มาใหม่หรืออาจจะส่งคำสั่ง QUIT มาก็ได้

หลังจากที่เซิร์ฟเวอร์เปิดให้ใช้บริการเมล์ดรอปแล้วมันจะกำหนดหมายเลขของจดหมาย (Messagenumber) ให้กับจดหมายแต่ละฉบับและบอกขนาดของจดหมาย จดหมายฉบับแรกในเมล์ดรอปจะเริ่มที่ “1” คำสั่งทั้งหมดและผลตอบรับที่เกี่ยวกับหมายเลขของจดหมายและขนาดของจดหมายจะอยู่ในรูปของเลขฐานสิบ

ต่อไปนี้เป็นข้อกำหนดของคำสั่ง QUIT ที่ใช้ในสเตทพิสูจน์สิทธิ์

QUIT

Arguments:	ไม่มี
Restrictions:	ไม่มี
Discussion:	+OK

ตัวอย่างเช่น

C: QUIT

S: +OK dewey POP3 server signing off

7.4. สเตททรานส์แซ็กชัน

หลังจากที่ไคล์เอ็นต์พิสูจน์สิทธิ์กับเซิร์ฟเวอร์แล้ว เซิร์ฟเวอร์จะล็อกและเปิดให้บริการเมล์ดรอป เซสชันจะมาอยู่ที่สเตททรานส์แซ็กชัน ไคล์เอ็นต์สามารถส่งคำสั่งใด ๆ ก็ได้มายังเซิร์ฟเวอร์อาจส่งคำสั่งมาทำซ้ำก็ได้ หลังจากที่ไคล์เอ็นต์ส่งคำสั่งมาเซิร์ฟเวอร์ก็จะส่งผลตอบรับกลับไป ท้ายที่สุดเมื่อไคล์เอ็นต์ส่งคำสั่ง QUIT เซสชันจะเข้าสู่สเตทอัพเดต

ต่อไปนี้เป็นข้อกำหนดของคำสั่งที่อยู่ในสเตททรานแซ็กชัน

STAT

Arguments:	ไม่มี
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	เซิร์ฟเวอร์ส่งสถานะยืนยันมาพร้อมข้อมูลสำหรับบริการเมล์ดรอป บรรทัดนี้จะถูกเรียกว่า “Drop listing” เพื่อให้ใช้ได้กับทุก ๆ เซิร์ฟเวอร์จำเป็นต้องมีรูปแบบที่แน่นอนสำหรับ Drop listing โดยผลตอบรับจะประกอบด้วย “+OK” ตามด้วยช่องว่าง จำนวนของจดหมายในเมล์ดรอป ช่องว่าง และขนาดของเมล์ดรอปและปิดท้ายด้วย CRLF อาจมีการอิมพลีเมนต์โดยการเพิ่มข้อมูลเพิ่มเติมอื่น ๆ ได้
Possible Responses:	+OK จำนวนของจดหมาย ขนาดของเมล์ดรอป

ตัวอย่างเช่น

C: STAT

S: +OK 2 320

LIST [msg]

Arguments:	หมายเลขของจดหมาย (มีหรือไม่มีก็ได้)
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	ถ้ามีอาร์กิวเมนต์และเซิร์ฟเวอร์ส่งสถานะยืนยันมาพร้อมข้อมูลของจดหมาย บรรทัดนี้จะถูกเรียกว่า “Scan listing” ถ้าไม่มีอาร์กิวเมนต์และเซิร์ฟเวอร์ส่งสถานะยืนยันผลตอบรับจะมีหลายบรรทัด เริ่มต้นด้วย +OK สำหรับจดหมายแต่ละฉบับในเมล์ดรอปและตามด้วยข้อมูลของจดหมายแต่ละฉบับ บรรทัดนี้จะถูกเรียกว่า “Scan listing” แต่ถ้าไม่มีจดหมายในเมล์ดรอปเซิร์ฟเวอร์จะส่งสถานะยืนยันตามด้วยตัวปิดสตริงและ CRLF เพื่อให้ใช้ได้กับทุก ๆ เซิร์ฟเวอร์จำเป็นต้องมีรูปแบบที่แน่นอนสำหรับ Scan listing โดย Scan listing จะประกอบด้วยหมายเลขของจดหมาย ตามด้วยช่องว่าง และขนาดที่แท้จริงของจดหมาย ปิดท้ายด้วย CRLF อาจมีการอิมพลีเมนต์โดยการเพิ่มข้อมูลเพิ่มเติมอื่น ๆ ได้
Possible Responses:	+OK scan listing -ERR ไม่มีจดหมาย

ตัวอย่างเช่น

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200

S:

...

C: LIST 2

S: +OK 2 200

...

C: LIST 3

S: -ERR มีจดหมายเพียง 2 ฉบับในเมล์ดรอป

RETR msg

Arguments:	หมายเลขของจดหมาย
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	ถ้าเซิร์ฟเวอร์ส่งสถานะยืนยันจะได้ผลตอบรับหลายบรรทัด เริ่มต้นด้วย +OK และตามด้วยจดหมายที่ตรงกับหมายเลขของจดหมายที่ส่งมา
Possible Responses:	+OK จดหมาย -ERR ไม่มีจดหมาย

ตัวอย่างเช่น

C: RETR 1

S: +OK 120 octets

S: <เซิร์ฟเวอร์ส่งจดหมายทั้งหมดตรงนี้>

S:

DELE msg

Arguments:	หมายเลขของจดหมาย
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	เซิร์ฟเวอร์จะมาร์ค (Mark) จดหมายว่าถูกลบ ถ้ามีการอ้างถึงหมายเลขของจดหมายนี้คำสั่งจะเกิดข้อผิดพลาด เซิร์ฟเวอร์จะยังไม่ลบจดหมายในทันทีจนกว่าเซสชันจะเข้าสู่สเตทอัพเดต
Possible Responses:	+OK ลบจดหมาย -ERR ไม่มีจดหมาย

ตัวอย่างเช่น

C: DELE 1

S: +OK ลบจดหมายฉบับที่ 1

...

C: DELE 2

S: -ERR จดหมายฉบับที่ 2 ถูกลบไปแล้ว

NOOP

Arguments:	ไม่มี
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	เซิร์ฟเวอร์ไม่ได้ทำอะไรเพียงแต่ส่งสถานะยืนยันกลับมาเท่านั้น
Possible Responses:	+OK

ตัวอย่างเช่น

C: NOOP

S: +OK

RSET

Arguments:	ไม่มี
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	ถ้าจดหมายฉบับใดถูกมาร์คไว้ก็จะทำการอันมาร์ค (Unmark) เซิร์ฟเวอร์จะส่งสถานะยืนยันกลับมา
Possible Responses:	+OK

ตัวอย่างเช่น

C: RSET

S: +OK มีจดหมาย 2 ฉบับในเมล์ดรอป

7.5. สเตทอัพเดต

เมื่อไคล์เอ็นต์ส่งคำสั่ง QUIT จากสเตททรานส์แซ็กชัน เซสชันจะเข้าสู่สเตทอัพเดต (ถ้าไคล์เอ็นต์ส่งคำสั่ง QUIT ในสเตทพิสูจน์สิทธิ์เซสชันจะหยุดแต่ไม่เข้าสู่สเตทอัพเดต)

ถ้าเซสชันหยุดโดยที่ไคล์เอ็นต์ไม่ได้ส่งคำสั่ง QUIT มาเซสชันจะไม่เข้าสู่สเตทอัพเดตและก็จะไม่เคลื่อนย้ายจดหมายออกจากเมล์ดรอป

QUIT

Arguments:	ไม่มี
Restrictions:	ไม่มี
Discussion:	เซิร์ฟเวอร์จะลบจดหมายทุกฉบับที่ถูกมาร์คออกจากเมล์ดรอปและส่งสถานะของการทำงาน ถ้าเกิดข้อผิดพลาดขณะทำการลบจดหมาย เช่น ทรัพยากรไม่เพียงพอ เมล์ดรอป จะไม่มีการลบจดหมายที่ถูกมาร์คไว้ ไม่ว่าจะลบได้หรือไม่เซิร์ฟเวอร์จะปลดล็อกเมล์ดรอปและปิดคอนเน็กชัน
Possible Responses:	+OK -ERR ไม่ได้ลบจดหมาย

ตัวอย่างเช่น

C: QUIT

S: +OK dewey POP3 server signing off (เมล์ดรอปว่างเปล่า)

...

C: QUIT

S: +OK dewey POP3 server signing off (มีจดหมาย 2 ฉบับในเมล์ดรอป)

...

7.6. คำสั่งอื่น ๆ

คำสั่งที่กล่าวมาทั้งหมดจำเป็นต้องมีสำหรับทุก ๆ เซิร์ฟเวอร์ แต่สำหรับคำสั่งอื่น ๆ ที่จะพูดถึงต่อจากนี้จะทำให้ไคล์เอ็นต์สามารถจัดการกับจดหมายได้อย่างอิสระในขณะที่ไม่สร้างความยุ่งยากให้กับเซิร์ฟเวอร์ด้วยเช่นเดียวกัน

TOP msg n

Arguments:	หมายเลขของจดหมาย ตามด้วยจำนวนบรรทัดที่ต้องไม่ติดลบ
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	ถ้าเซิร์ฟเวอร์ส่งสถานะยืนยันดังนั้นผลตอบรับจะมีหลายบรรทัด เริ่มด้วย +OK ส่วนหัวของจดหมาย (Message Header) บรรทัดว่างเพื่อแยกส่วนหัวกับส่วนข้อมูลของจดหมายและตามด้วยตัวจดหมายตามจำนวนบรรทัด **หมายเหตุ** ถ้าจำนวนบรรทัดที่ไคล์เอ็นต์ส่งมามีค่ามากกว่าจำนวนบรรทัดของตัวจดหมาย เซิร์ฟเวอร์จะส่งเต็มฉบับมาให้ไคล์เอ็นต์
Possible Responses:	+OK ส่วนบนของจดหมาย -ERR ไม่มีจดหมาย

ตัวอย่างเช่น

C: TOP 1 10

S: +OK

S: <เซิร์ฟเวอร์ส่งหัวของจดหมายและตามด้วย 10 บรรทัดแรกของจดหมาย>

S:

...

C: TOP 100 3

S: -ERR ไม่มีจดหมาย

UIDL [msg]

Arguments:	หมายเลขของจดหมาย (มีหรือไม่มีก็ได้)
Restrictions:	ใช้เฉพาะสเตททรานส์แซ็กชันเท่านั้น
Discussion:	ถ้ามีอาร์กิวเมนต์และเซิร์ฟเวอร์ส่งสถานะยืนยันมาพร้อมข้อมูลของจดหมาย บรรทัดนี้จะถูกเรียกว่า “Unique-id listing” ถ้าไม่มีอาร์กิวเมนต์และเซิร์ฟเวอร์ส่งสถานะยืนยันผลตอบรับจะมีหลายบรรทัด เริ่มต้นด้วย +OK สำหรับจดหมายแต่ละฉบับในเมล์ดรอปและตาม

	ด้วยข้อมูลของจดหมายแต่ละฉบับ บรรทัดนี้จะถูกเรียกว่า “Unique-id listing” เพื่อให้ง่ายสำหรับการส่งทุก ๆ เซิร์ฟเวอร์จำเป็นต้องมีรูปแบบที่แน่นอนสำหรับ Unique-id listing โดย Unique-id listing จะประกอบด้วยหมายเลขของจดหมาย ตามด้วยช่องว่าง และ Unique-id ของจดหมาย Unique-id เป็นสตริงตามแต่เซิร์ฟเวอร์จะกำหนด ประกอบด้วยตัวอักษรตั้งแต่ 1-70 ตัวอักษรอยู่ในช่วง 0x21 ถึง 0x7E ซึ่งจะสามารถแยกแยะจดหมายแต่ละฉบับในเมล์ดรอป ถึงเซิร์ฟเวอร์จะสามารถกำหนด Unique-id ได้ตามอำเภอใจแต่ก็มีข้อกำหนดที่ว่าต้องสามารถคำนวณ Unique-id ได้จากตัวจดหมาย ไคล์เอ็นต์ควรจะสามารถจัดการกับเหตุการณ์ที่สำเนาของจดหมายมีหมายเลขหมายเลข Unique-id เดียวกันกับจดหมายต้นฉบับ
Possible Responses:	+OK ตามด้วย unique-id -ERR ไม่มีจดหมาย

ตัวอย่างเช่น

C: UIDL

S: +OK

S: 1 whqtswO00WBw418f9t5JxYwZ

S: 2 QhdPYR:00WBw1Ph7x7

S: .

...

C: UIDL 2

S: +OK 2 QhdPYR:00WBw1Ph7x7

...

C: UIDL 3

S: -ERR no such message, only 2 messages in maildrop

USER name

Arguments:	สตริงที่ระบุเมล์บ็อกซ์
Restrictions:	ใช้เฉพาะสเตทพิสูจน์สิทธิ์หลังจากที่ POP3 ส่งสัญญาณเริ่มต้นหรือตอนที่คำสั่ง USER หรือ PASS ทำไม่สำเร็จ (พิสูจน์สิทธิ์ใหม่นั่นเอง)
Discussion:	ในการพิสูจน์สิทธิ์จะใช้ USER และ PASS รวมกัน โดยหลังจากที่ไคล์เอ็นต์ส่ง USER ถ้าเซิร์ฟเวอร์ส่งสถานะยืนยัน "+OK" ไคล์เอ็นต์จะส่ง PASS เพื่อเสร็จสิ้นกระบวนการพิสูจน์สิทธิ์ หรือส่ง QUIT เพื่อหยุดเซสชัน ถ้าเซิร์ฟเวอร์ส่งสถานะปฏิเสธให้กับคำสั่ง USER ไคล์เอ็นต์อาจจะเริ่มการพิสูจน์สิทธิ์ใหม่หรืออาจจะส่ง QUIT เพื่อยุติการพิสูจน์สิทธิ์ เซิร์ฟเวอร์จะส่งสถานะปฏิเสธถ้ารหัสผ่านไม่ถูกต้อง
Possible Responses:	+OK ชื่อของเมล์บ็อกซ์ -ERR ไม่มีเมล์บ็อกซ์

ตัวอย่างเช่น

C: USER frated

S: -ERR ไม่มีเมล์บ็อกซ์ของ frated

...

C: USER mrose

S: +OK mrose is a real hoopy frood

PASS string

Arguments:	รหัสผ่าน
Restrictions:	ใช้เฉพาะสเตทพิสูจน์สิทธิ์ทันทีหลังจากที่ทำคำสั่ง USER สำเร็จ
Discussion:	เมื่อไคล์เอ็นต์ส่งคำสั่ง PASS เซิร์ฟเวอร์จะใช้อาร์กิวเมนต์จาก USER และ PASS เพื่อตรวจสอบดูว่าสมควรให้บริการเมล์ดรอปแก่ไคล์เอ็นต์หรือไม่
Possible Responses:	+OK ล็อกเมล์ดรอปและพร้อมให้บริการ -ERR รหัสผ่านไม่ถูกต้อง -ERR ไม่สามารถล็อกเมล์ดรอปได้

ตัวอย่างเช่น

C: USER mrose

S: +OK mrose is a real hoopy frood

C: PASS secret

S: -ERR maildrop already locked

...

C: USER mrose

S: +OK mrose is a real hoopy frood

C: PASS secret

S: +OK mrose's maildrop has 2 messages (320 octets)

APOP name digest

Arguments:	สตริงที่ระบุเมล์บ็อกซ์ และไดเจสสตริง (MD5 digest string)
Restrictions:	ใช้เฉพาะสเตทพิสูจน์สิทธิ์หลังจากที่ POP3 ส่งสัญญาณเริ่มต้นหรือตอนที่คำสั่ง USER หรือ PASS ทำไม่สำเร็จ (พิสูจน์สิทธิ์ใหม่นั่นเอง)
Discussion:	โดยปกติแล้วเซสชันจะเริ่มต้นด้วยการแลกเปลี่ยน USER/PASS ซึ่งในกรณีที่ใช้งานเซิร์ฟเวอร์แบบไม่ต่อเนื่องอาจสิ้นเปลืองทรัพยากร ดังนั้นจึงมีการคอนเน็กกับเซิร์ฟเวอร์เป็นช่วง ๆ ผลที่ได้ก็คืออาจมีการลอบดักจับรหัสผ่านได้ คำสั่ง APOP จะป้องกันการทำ Replay attack และยังสามารถใช้ในการพิสูจน์สิทธิ์ได้อีกด้วย เซิร์ฟเวอร์ที่อิมพลีเมนต์คำสั่ง APOP จะส่งไทม์แสตมป์ (Timestamp) ไปในสัญญาณเริ่มต้น รูปแบบของไทม์แสตมป์จะเหมือนกับหมายเลขของจดหมายและจะต้องแตกต่างกันในแต่ละเวลาที่เซิร์ฟเวอร์ส่งสัญญาณเริ่มต้น ตัวอย่างเช่น ในระบบ UNIX จะมีหมายเลขของโพรเซสสำหรับแต่ละโพรเซสในเซิร์ฟเวอร์รูปแบบของไทม์แสตมป์จึงเป็นดังนี้ <Process-ID.clock@hostname> โดยที่ Process-ID คือ PID (หมายเลขโพรเซส) clock เป็นเวลาของระบบ และ hostname เป็นชื่อเต็มของ domain-name ที่อ้างถึงโฮสต์ที่อยู่บนเซิร์ฟเวอร์ ไคล์เอ็นต์จะต้องบันทึกไทม์แสตมป์และส่งคำสั่ง APOP พารามิเตอร์ name

	เหมือนกับ name ใน USER ส่วน digest เป็นค่าที่คำนวณได้จากอัลกอริทึม MD5 โดยส่งสตริงที่ประกอบด้วยไทม์แสตมป์และ Share secret เข้าไปคำนวณ ซึ่ง Share secret เป็นค่าที่รู้กันระหว่างไคล์เอ็นต์กับเซิร์ฟเวอร์ ซึ่ง digest ที่ส่งจะเป็นเลขฐานสิบหกและเป็นตัวพิมพ์เล็กทั้งหมด เมื่อเซิร์ฟเวอร์ได้รับคำสั่ง APOP มันก็จะทำการตรวจสอบ digest ถ้า digest ถูกต้องก็จะส่งสถานะยืนยันพร้อมกับเข้าสู่สเตททรานส์แซ็กชัน แต่ถ้าไม่ถูกต้องก็จะส่งสถานะปฏิเสธและยังคงอยู่ที่สเตทพิสูจน์สิทธิ์ต่อไป
Possible Responses:	+OK ล็อกเมล์ดรอปและพร้อมให้บริการ -ERR ไม่มีสิทธิ์

ตัวอย่างเช่น

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb

S: +OK maildrop has 1 message (369 octets)

ในตัวอย่างนี้ Share secret คือ 'tanstaaf' ดังนั้น อัลกอริทึม MD5 ก็จะเป็นดังนี้

<1896.697170952@dbc.mtview.ca.us>tanstaaf

และ digest ที่ได้ออกมาคือ c4c9334bac560ecc979e58001b3e22fb

7.7. สรุปคำสั่งต่าง ๆ ของ POP3

สเตท	คำสั่งและอาร์กิวเมนต์
สเตทพิสูจน์สิทธิ์	USER name PASS string QUIT
สเตททรานส์แซ็กชัน	STAT LIST [msg] RETR msg DELE msg NOOP RSET QUIT

ตารางที่ 7-1 แสดงคำสั่งพื้นฐานที่จำเป็นต้องมีสำหรับทุก ๆ เซิร์ฟเวอร์

สเตท	คำสั่งและอาร์กิวเมนต์
สเตทพิสูจน์สิทธิ์	APOP name digest
สเตททรานส์แซ็กชัน	TOP msg n UIDL [msg]

ตารางที่ 7-2 แสดงคำสั่งอื่น ๆที่จะช่วยเพิ่มประสิทธิภาพในการใช้งาน POP3

ผลที่เซิร์ฟเวอร์ส่งกลับ	ผลของคำสั่ง
+OK	ยืนยัน
-ERR	ปฏิเสธ

ตารางที่ 7-3ผลตอบรับของเซิร์ฟเวอร์

7.8. ตัวอย่างเซสชันของ

S: <wait for connection on TCP port 110>

C: <open connection>

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb

S: +OK mrose's maildrop has 2 messages (320 octets)

C: STAT

S: +OK 2 320

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200

S:

C: RETR 1

S: +OK 120 octets

S: <the POP3 server sends message 1>

S:

C: DELE 1

S: +OK message 1 deleted

C: RETR 2

S: +OK 200 octets

S: <the POP3 server sends message 2>

S:

C: DELE 2

S: +OK message 2 deleted

C: QUIT

S: +OK dewey POP3 server signing off (maildrop empty)

C: <close connection>

S: <wait for next connection>

บทที่ 8

การอิมพลีเมนต์ POP3 โพรโตคอล

8.1 บทนำ

โพรโตคอล POP3 เป็นมาตรฐานสากลสำหรับการเข้าถึงเมล์ที่อยู่บนเครื่องเซิร์ฟเวอร์ ซึ่งสามารถนำไปอิมพลีเมนต์ได้โดยไม่ขึ้นอยู่กับภาษาในการเขียนโปรแกรมและแพลตฟอร์ม

สำหรับภาษาจาวามี JavaMail API เวอร์ชัน 1.2 ที่มีการอิมพลีเมนต์ POP3 โพรโตคอลไว้ให้แก่ผู้สนใจจะพัฒนาโปรแกรมนำไปใช้ ส่วนภาษาซีมี Cpop3Connection เวอร์ชัน 1.25 เขียนโดย PJ Naughter ที่เป็น MFC คลาสซึ่งผู้เขียนอนุญาตให้ผู้พัฒนาโปรแกรมสามารถนำไปใช้ได้เลย

8.2 JavaMail

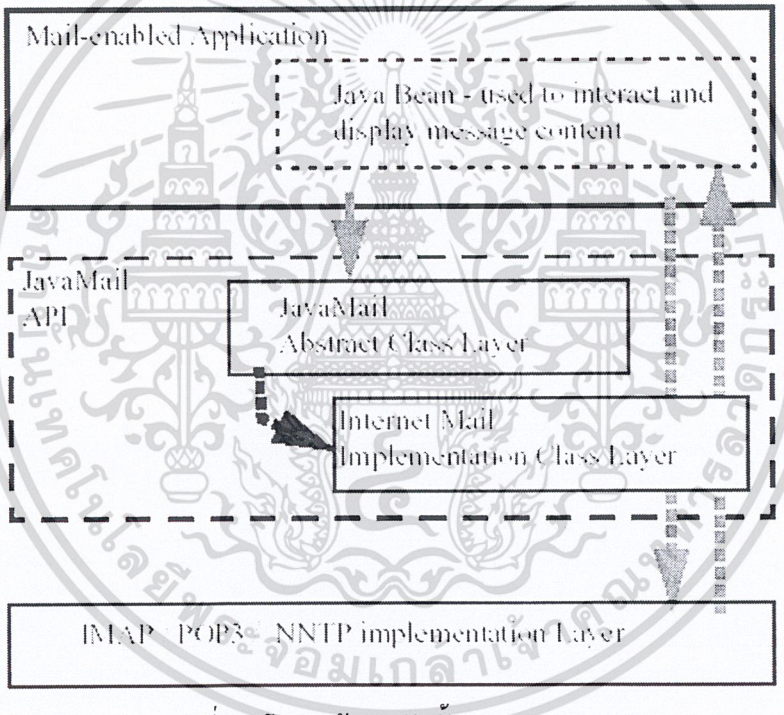
หลังจากใช้เวลาไม่นานตั้งแต่เริ่มเผยแพร่ โปรแกรมภาษาจาวาก็ได้รับการพัฒนาจนกลายเป็นแพลตฟอร์มที่ใช้กันอยู่ในปัจจุบัน ซึ่งแพลตฟอร์มนี้ได้เพิ่มฟังก์ชันการทำงานในส่วนที่เป็น Distributed Object เช่น RMI (Java Remote Method Invocation) และ COBRA (Common Object Request Broker Architecture) และในส่วนที่เป็นคอมโพเน็นต์ เช่น JavaBeans ส่วนที่เป็นโปรแกรมประยุกต์อื่นๆ ก็ถูกพัฒนาเพื่อผนวกเข้าเป็นแพลตฟอร์มด้วยเช่นกัน สำหรับระบบจดหมายอิเล็กทรอนิกส์และโครงสร้างทั้งการรับและส่งข้อความนั้นก็มี JavaMail API เป็นตัวจัดการ

JavaMail API ได้จัดสรรแอ็บสแทรกต์คลาสต่างๆ สำหรับระบบจดหมายอิเล็กทรอนิกส์ เช่น คลาสเมสเซจ (Message) คลาสสตอร์ (Store) และคลาสทรานสปอร์ต (Transport) เป็นต้น ซึ่งคลาสเหล่านี้สามารถนำไปใช้ได้เลยหรืออาจทำการสับคลาส (Subclassed) สำหรับโพรโตคอลใหม่และยังสามารถเพิ่มฟังก์ชันที่จำเป็นได้

8.2.1 โครงสร้างลำดับชั้นของ JavaMail

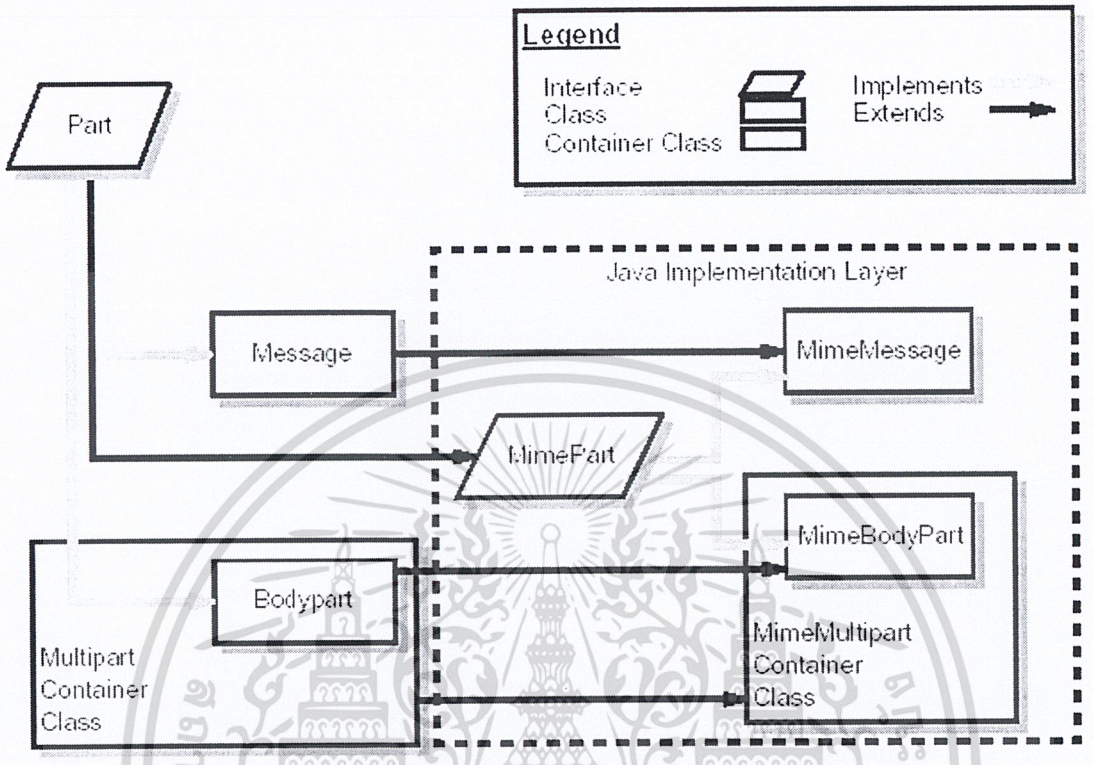
โครงสร้างของ JavaMail จะแบ่งออกได้ดังนี้

- แอ็บสแทรกต์เลเยอร์ (Abstracted Layer) คลาส อินเทอร์เฟซและแอ็บสแทรกต์เมธอด (Abstract methods) ต้องสามารถใช้งานได้กับระบบจดหมายอิเล็กทรอนิกส์ทุกระบบ
- อินเทอร์เน็ตอิมพลีเมนต์เตชันเลเยอร์ (Internet Implementation Layer) จะสนับสนุนมาตรฐานอินเทอร์เน็ตได้แก่ RFC822 และ MIME (Multipurpose Internet Mail Extensions)
- JavaMail ใช้ JAF (JavaBeans Activation Framework) ในการปกป้องข้อมูลของจดหมายและเพื่อจัดการกับคำสั่งต่างๆ ที่ต้องข้องเกี่ยวกับจดหมายนั้น



รูปที่ 8-1 โครงสร้างลำดับชั้นของ JavaMail

8.2.2 ความสัมพันธ์ของแต่ละคลาสใน JavaMail



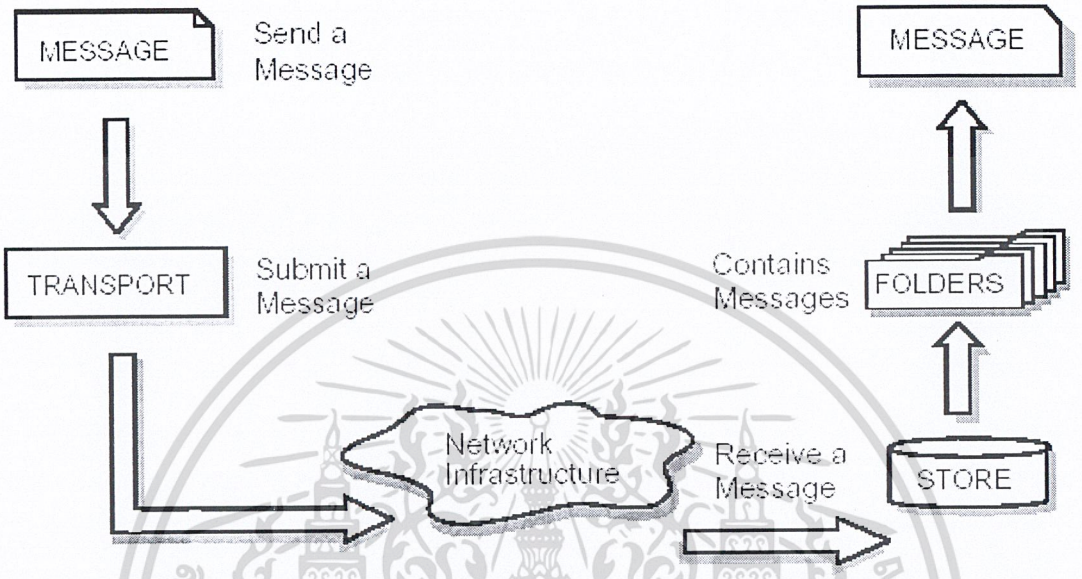
รูปที่8- 2 แสดงคลาสหลักๆ และอินเทอร์เฟซที่ประกอบกันเป็น JavaMail

8.2.3 ขอบข่ายการทำงานของ JavaMail

JavaMail ต้องสามารถจัดการกับการประมวลผลมาตรฐานของระบบจดหมายอิเล็กทรอนิกส์

- สร้างจดหมายที่ประกอบด้วยรายระเอียดของหัวจดหมาย (Message Header) เนื้อความของจดหมายซึ่งต้องเป็นข้อมูลประเภทเดียวกับ Content-Type ที่ระบุไว้ในส่วนหัวของจดหมาย JavaMail ใช้อินเทอร์เฟซของพาร์ต (Part) กับคลาสเมสเซจ ในการนิยามตัวจดหมาย และใช้ DataHandler ออบเจ็กต์ที่นิยามโดย JAF เก็บข้อมูลของจดหมายนั้น
- สร้างออบเจ็กต์ของเซสชัน (Session) ขึ้นมาเพื่อทำการพิสูจน์สิทธิ์สำหรับผู้ใช้ และใช้ควบคุมการเข้าถึงข้อมูลรวมทั้งการส่งข้อมูล
- ส่งจดหมายไปยังผู้รับได้
- ดึงข้อมูลของจดหมายขึ้นมาอ่านได้
- ทำการประมวลผลคำสั่งในระดับสูงของการดึงข้อมูลของจดหมายขึ้นมา เช่น ทำการดูตัวอย่างก่อนการพิมพ์ และทำการพิมพ์ ซึ่งจะต้องอาศัย JAF

หมายเหตุ – ขอบข่ายการทำงานของ JavaMail นั้นไม่ได้พูดถึงกระบวนในการนำส่งจดหมาย ระบบความปลอดภัย การยกเลิกการติดต่อ ฟังก์ชันในการกรองจดหมาย และ Directory Service



รูปที่ 8-3 แสดงการจัดการกับการประมวลผลของระบบจดหมายอิเล็กทรอนิกส์

8.2.4 คอมโพเน็นต์หลักของ JavaMail

- คลาสเมสเซจ

คลาสนี้จะนิยามรายละเอียดข้อมูลที่จะประกอบเป็นตัวจดหมาย ซึ่งข้อมูลนี้จะระบุรายละเอียดของข้อมูล โครงสร้างของตัวข้อมูล รวมถึง Content-Type ซึ่งตัวข้อมูลนี้จะถูกใช้งานเป็นออบเจ็กต์ของ DataHandler ซึ่งจะห่อข้อมูลจริงเอาไว้

- คลาสโฟล์เดอร์ (Folder) และคลาสสตอร์

จดหมายจะถูกเก็บไว้ในโฟล์เดอร์ออบเจ็กต์ ซึ่งออบเจ็กต์นี้สามารถมีสับโฟล์เดอร์ (subfolder) ได้ คลาสโฟล์เดอร์จะมีเมธอดหลัก ๆ ดังนี้ เฟตช์ (Fetch) จดหมายขึ้นมา เพิ่มจดหมาย (Append) คัดลอก และลบจดหมาย

คลาสสตอร์จะอธิบายฐานข้อมูลที่จัดการกับโฟล์เดอร์ที่เก็บจดหมายไว้ คลาสสตอร์จะระบุโพรโตคอลในการเข้าถึงโฟล์เดอร์และการดึงข้อมูลของจดหมายที่เก็บอยู่ในโฟล์เดอร์ ผู้ให้บริการจะทำการอิมพลีเมนต์โพรโตคอลในการเข้าถึงจดหมาย (เช่น IMAP, POP3 เป็นต้น) จะทำการสับคลาสคลาสสตอร์ ผู้ใช้จะเริ่มสร้างเซสชันเพื่อติดต่อกับระบบจดหมายอิเล็คทรอนิกส์โดยการอิมพลีเมนต์คลาสสตอร์

- คลาสทรานสปอร์ต

เมื่อไคล์เอ็นต์สร้างจดหมายขึ้นมาใหม่แล้วต้องการส่งไปให้ผู้รับ ขั้นตอนในการส่งจะเรียกเมธอด Send ของ คลาสทรานสปอร์ต

คลาสทรานสปอร์ตจะมีทรานสปอร์ตเอเจนต์ (Transport Agent) ในการหาเส้นทางสำหรับส่งจดหมายไปยังที่อยู่ของผู้รับ

- คลาสเซสชัน

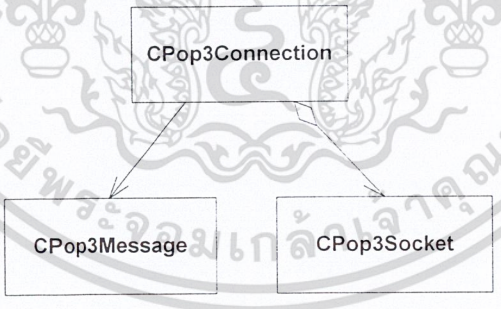
คลาสเซสชันจะกำหนดเป็นโกลบอลของผู้ใช้แต่ละคนซึ่งจะเป็นอินเทอร์เฟซระหว่างไคล์เอ็นต์กับเครือข่าย คลาสเซสชันไม่สามารถทำการสับคลาสได้

คลาสเซสชันจะทำหน้าที่ผลิตคลาสสตอร์และคลาสทรานสปอร์ต เมื่อเรียกใช้เมธอดของออบเจ็กต์เซสชัน ไคล์เอ็นต์สามารถเรียกใช้คลาสสตอร์และคลาสทรานสปอร์ตได้

8.3 CPop3Connection

สำหรับ CPop3Connection จะมีคลาสหลัก ๆ อยู่ 3 คลาส คือ CPop3Connection CPop3Message และ CPop3Socket โดยทั้งสามคลาสมีความสัมพันธ์กันดังนี้

- CPop3Connection มีความสัมพันธ์แบบ has a กับ CPop3Socket
- CPop3Connection มีความสัมพันธ์แบบ associate กับ CPop3Message



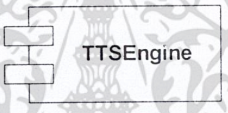
รูปที่ 8-4 แสดงคลาสและความสัมพันธ์ระหว่างคลาสของ CPop3Connection

บทที่ 9

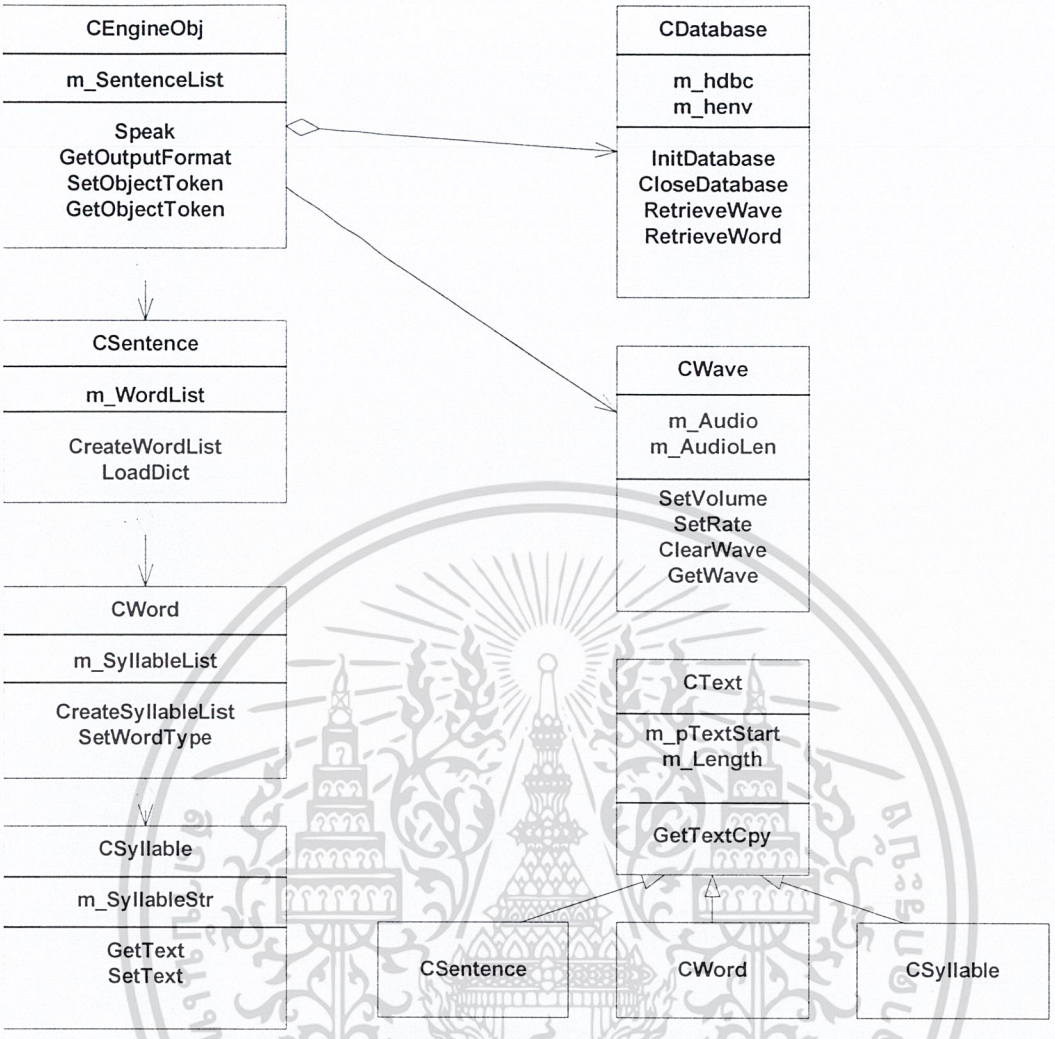
ระบบอีเมล์ครบวงจร

9.1 โครงสร้าง TTS Engine

จากสถาปัตยกรรมของ SAPI โครงงานนี้ได้ทำการอิมพลีเมนต์เอ็นจินภาษาไทยขึ้นมา โดยใช้วิธีเก็บเสียงของแต่ละพยางค์ไว้ในฐานข้อมูล เวลาจะใช้ก็นำแต่ละพยางค์มาต่อกันเป็นคำแล้วนำคำต่อเป็นประโยค ซึ่งวิธีนี้จะใช้หน่วยความจำสำหรับจัดเก็บเสียงมาก แต่ให้คุณภาพเสียงที่ดี ซึ่งเสียงที่ใช้จะถูกจัดเก็บให้อยู่ในรูปแบบ PCM มีคุณภาพเสียง 16 บิต แซมปลิ้ง 11 กิโลเฮิรตซ์ โมโน ซึ่งมีความชัดเจนเพียงพอสำหรับโทรศัพท์



รูปที่ 9-1 ภาพรวมของเอ็นจินเปลี่ยนข้อความเป็นเสียงภาษาไทย



รูปที่ 9-2 องค์ประกอบของเอ็นจิน เปลี่ยนข้อความเป็นเสียงภาษาไทย

9.2 รายละเอียดของคลาส

CEngineObj เป็นคลาสที่อิมพลีเมนต์อินเทอร์เฟซ ISpTTSEngine และ ISpObjectWithToken โดยมีเมธอดที่สำคัญดังนี้

SetObjectToken เป็นเมธอดที่ถูกเรียกโดย SAPI เพื่อกำหนดค่าพารามิเตอร์ให้กับเสียงที่เอ็นจินผลิตออกมา เช่น อายุ เพศ เสียงสูง-ต่ำ เป็นต้น ฟังก์ชันนี้จะทำการโหลดพจนานุกรมมาสร้างเป็นโครงสร้างข้อมูลแบบทรัย สำหรับที่อยู่ของไฟล์พจนานุกรมจะถูกจัดเก็บไว้ในรีจิสตรี้ของวินโดวส์แล้ว

GetObjectToken เป็นเมธอดที่ถูกเรียกโดย SAPI เพื่ออ่านค่าว่าขณะนี้เอ็นจินต์สร้างเสียงด้วยพารามิเตอร์อะไรบ้าง

GetOutputFormat เป็นเมธอดที่ถูกเรียกโดย SAPI เพื่อถามเอ็นจินว่าคุณภาพของเสียงที่ผลิตเป็นอย่างไร เพื่อ SAPI จะได้สร้างออดิโอดีไวซ์ที่เหมาะสมกับเอ็นจิน ในการอิมพลีเมนต์ครั้งนี้เลือกใช้คุณภาพเสียง 16 บิต แซมปลิ้ง 11 กิโลเฮิรตซ์ โมโน

Speak เป็นเมธอดที่ถูกเรียกโดย SAPI ถือเป็นฟังก์ชันสำคัญที่ใช้ติดต่อระหว่างเอ็นจินกับ SAPI ข้อความที่ส่งผ่านเข้ามาทางฟังก์ชัน Speak จะถูกเปลี่ยนเป็นเสียงและส่งกลับไปให้ SAPI ผ่านทางอินเทอร์เฟซ ISpTTSEngineSite

CSentence, CWord, CSyllable เป็นคลาสที่จัดการกับข้อความในระดับต่าง ๆ ในขั้นตอนแรกฟังก์ชัน Speak จะสร้าง m_SetenceList ขึ้นมาโดยใช้ช่องว่าง (Space) เป็นตัวแบ่งประโยค หลังจากนั้นจะเรียกใช้เมธอดต่าง ๆ ดังนี้

CSentence::CreateWordList แบ่งประโยคออกเป็นคำโดยใช้พจนานุกรมและทำวจีวิภาคในกรณีที่ไม่รู้จักคำ ๆ นั้น

CWord::CreateSyllableList แบ่งคำที่ได้จาก Csentence::CreateWordList ออกเป็นพยางค์โดยใช้ข้อมูลจากฐานข้อมูลคำอ่าน

CWave เป็นคลาสที่จัดการกับเสียงมีเมธอดใช้เปลี่ยนแปลงเสียงตามความดัง และความเร็วในการพูด

CDatabase เป็นคลาสที่ติดต่อกับฐานข้อมูลทั้งที่เป็นไบนารีและข้อความด้วย SQL สเตทเมนต์

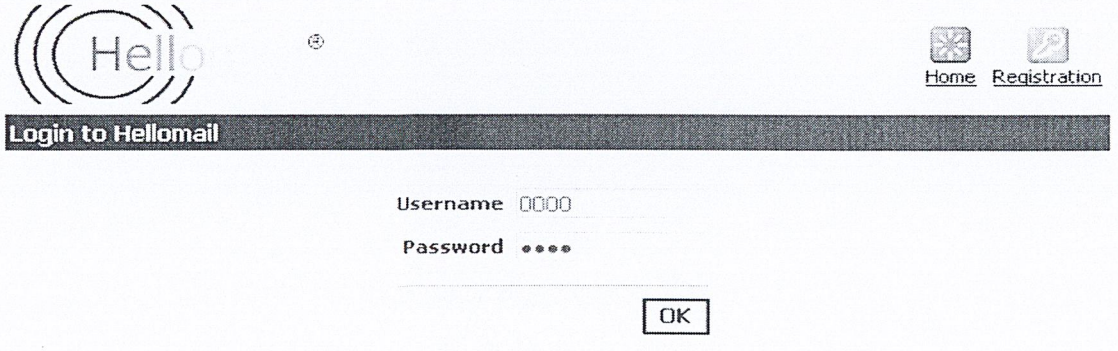
9.3 EmailPhone System

เป็นระบบที่อำนวยความสะดวกในการตรวจสอบอีเมล์ผ่านโทรศัพท์ เว็บไซท์ และโทรศัพท์มือถือ ประกอบด้วย 3 ส่วนหลัก ๆ ดังนี้

1. เว็บไซท์สำหรับสมัครสมาชิก ระบุความต้องการต่าง ๆ เพื่อให้บริการที่เหมาะสมกับผู้ใช้
2. โปรแกรมตรวจสอบอีเมล์ที่เข้ามาใหม่ เมื่อมีอีเมล์ใหม่ก็จะส่งข้อความขนาดสั้นแจ้งให้ผู้ใช้ทราบ
3. โปรแกรมสำหรับตรวจสอบอีเมล์ผ่านทางโทรศัพท์ เมื่อผู้ใช้โทรศัพท์มายังเครื่องที่เปิดให้บริการ EmailPhone โปรแกรมจะทำการติดต่อไปยังเมล์เซิร์ฟเวอร์เพื่ออ่านอีเมล์ที่ผู้ใช้ต้องการ

9.3.1 เว็บไซท์

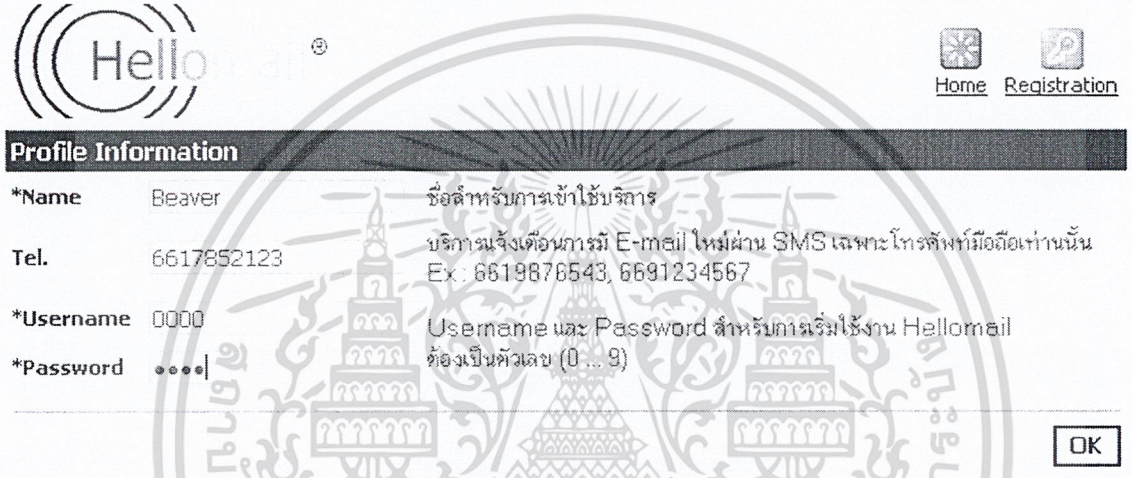
เว็บเพจแรกจะเป็นหน้าจอสำหรับทำการล็อกอินเพื่อเข้าไปใช้บริการ สำหรับผู้ใช้ที่ต้องการสมัครสมาชิกใหม่สามารถเลือกที่ Registration เพื่อขอเข้าใช้บริการได้



รูปที่ 9-3 แสดงหน้าจอล็อกอิน

ในขั้นตอนการสมัครสมาชิกสามารถทำได้โดยใส่ชื่อของผู้สมัคร หมายเลขโทรศัพท์ Username และ Password โดยที่ Username กับ Password จะต้องเป็นตัวเลขเท่านั้น ซึ่งจะใช้สำหรับล็อกอินเข้ามาในเว็บไซท์และใช้สำหรับล็อกอินในการโทรศัพท์มาฟังอีเมล์ด้วย

หมายเหตุ หมายเลขโทรศัพท์ต้องเป็นโทรศัพท์มือถือเท่านั้นเนื่องจากจะใช้ในการให้บริการส่ง SMS แจ้งเตือนว่ามีเมล์ใหม่เข้ามา ในการกรอกหมายเลขโทรศัพท์มีข้อกำหนดดังนี้ เริ่มจากรหัสประเทศตามด้วยหมายเลขโทรศัพท์ เช่น ประเทศไทยมีรหัสประเทศเป็น “66” และหมายเลขโทรศัพท์เป็น “017539515” หมายเลขที่จะต้องกรอกก็จะเป็น “6617539515” จะทำการตัด “0” ที่นำหน้าหมายเลขโทรศัพท์ทิ้ง

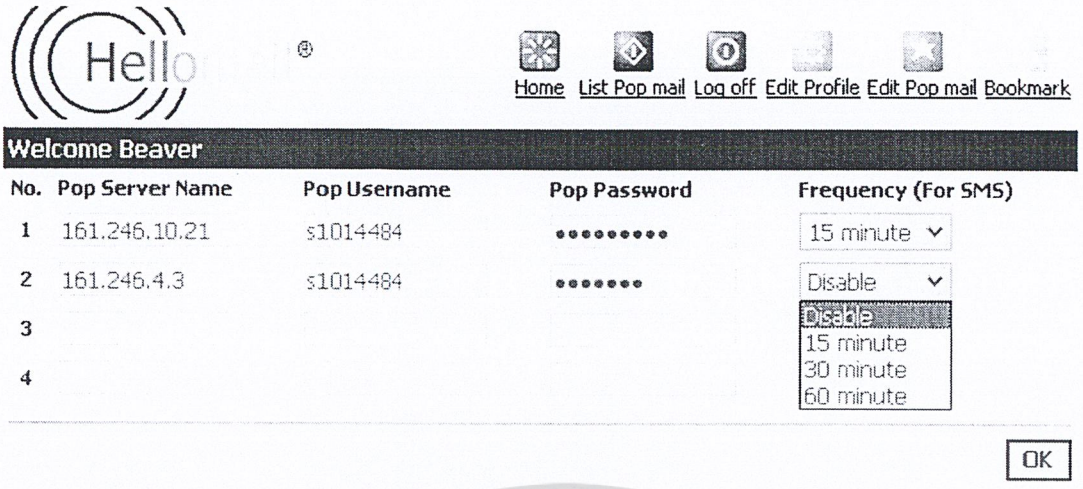


รูปที่ 9-4 แสดงหน้าจอสมัครสมาชิก

ถ้าผู้ใช้เพิ่งจะเคยเข้ามาใช้บริการครั้งแรกจะยังไม่มี Account ต้องทำการแก้ไขข้อมูลของ Account โดยเลือกที่ Edit Pop mail

รูปที่ 9-5 แสดงหน้าจอสำหรับผู้ใช้ที่เข้ามาใช้บริการครั้งแรก

หลังจากที่ได้ทำการเพิ่มข้อมูลของ Account เสร็จเรียบร้อยแล้วก็จะสามารถใช้บริการเช็คเมล์ได้ทั้งบนเว็บและทางโทรศัพท์ ผู้ใช้สามารถเลือกที่จะเช็คเมล์ได้ทั้งสิ้น 4 Account

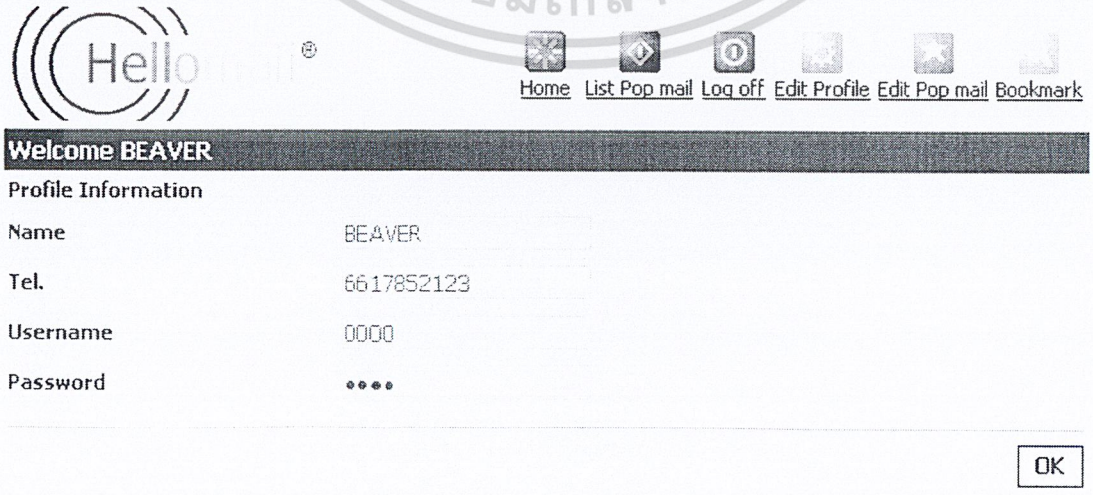


รูปที่ 9-6 แสดงหน้าจอแก้ไขข้อมูลของ Account

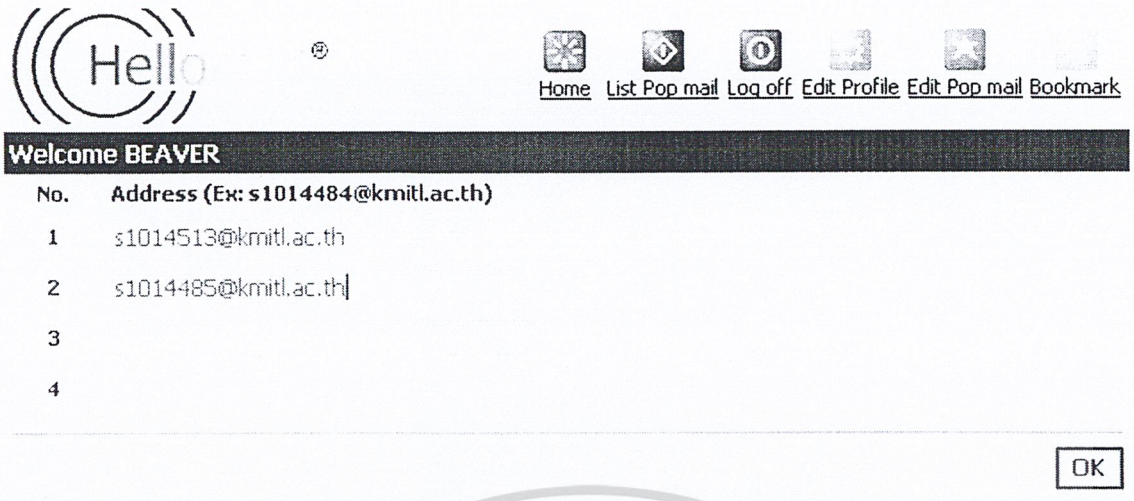
แสดงรายการ Account ทั้งหมดของยูสเซอร์ โดยที่ยูสเซอร์สามารถเลือก List Pop mail เพื่อให้แสดงรายการของเมล์ทั้งหมดที่ได้ฝาก Account ไว้

รูปที่ 9-7 แสดงหน้าจอหลังจากทำการล็อกอินเข้ามาโดยที่มี Account อยู่แล้ว

ถ้าต้องการแก้ไขข้อมูลส่วนตัว เช่น เปลี่ยนเบอร์โทรศัพท์ ก็สามารถทำได้โดยเลือก Edit Profile



รูปที่ 9-8 แสดงหน้าจอแก้ไขข้อมูลส่วนตัว



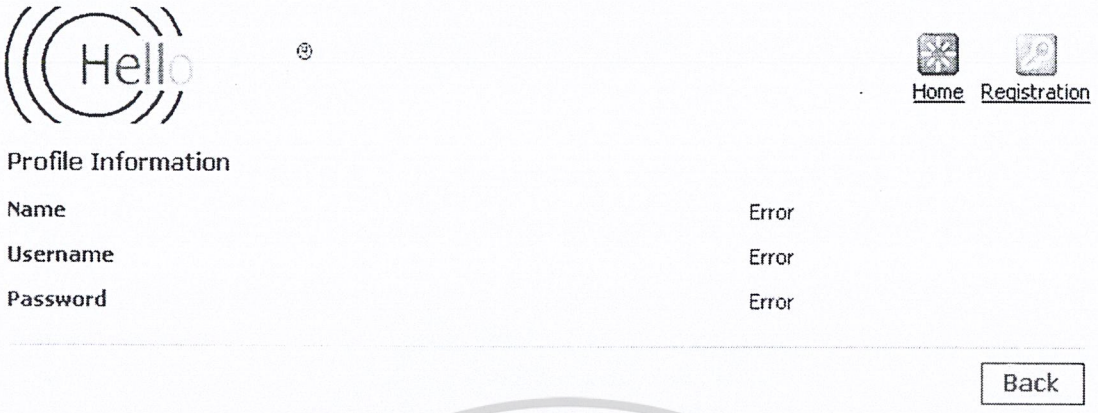
รูปที่ 9-9 แสดงหน้าจอสำหรับระบุอีเมล์ที่สนใจเป็นพิเศษ

ต่อไปเป็นหน้าจอแสดงผลเมื่อเกิดความผิดพลาดในการล็อกอิน

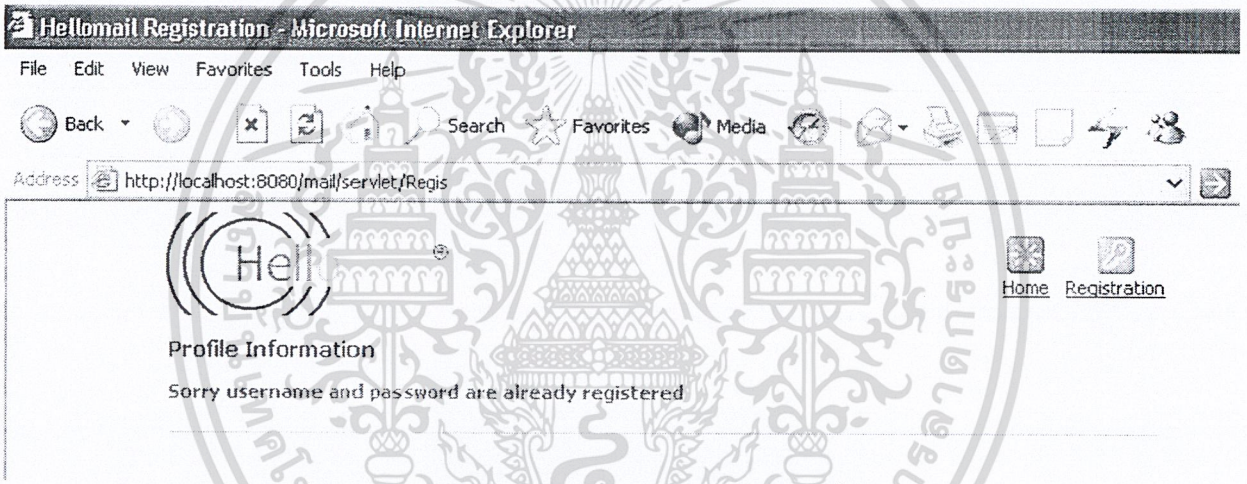
รูปที่ 9-10 ไม่สามารถล็อกอินได้ เนื่องจาก Username และ Password ไม่ตรงกัน

รูปที่ 9-11 กรอก Username กับ Password ผิดรูปแบบ

ต่อไปเป็นหน้าจอแสดงผลเมื่อเกิดความผิดพลาดในการสมัครสมาชิก

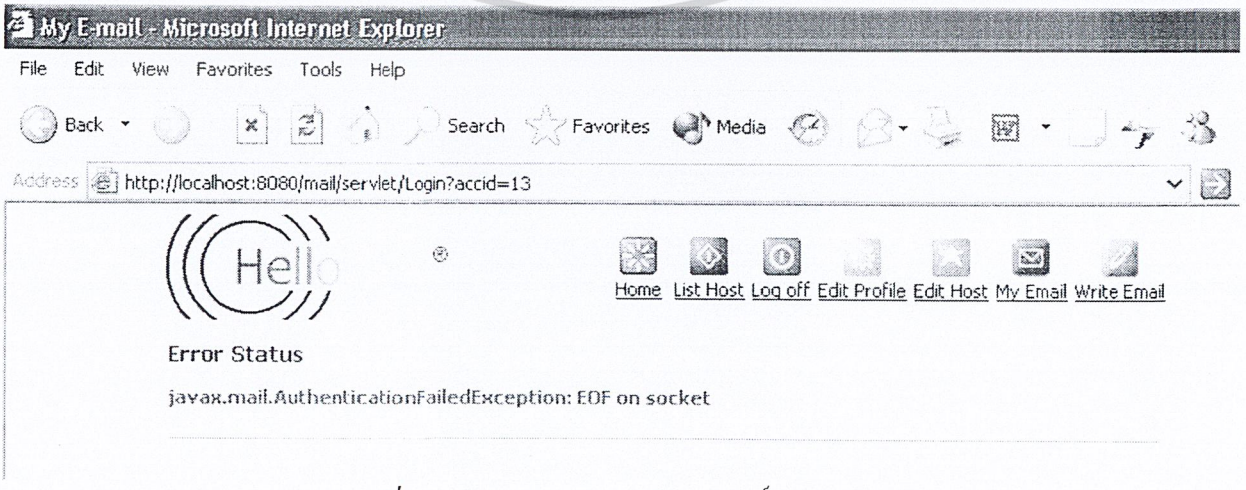


รูปที่ 9-12 กรอกข้อมูลผิดรูปแบบที่กำหนด

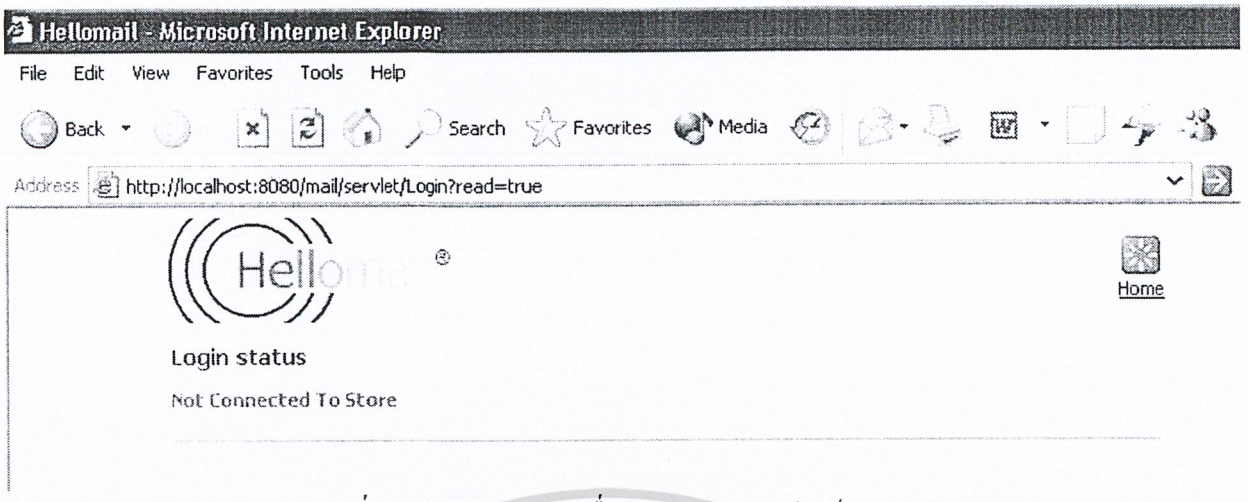


รูปที่ 9-13 Username และ Password มีคนใช้ก่อนหน้านี้แล้ว

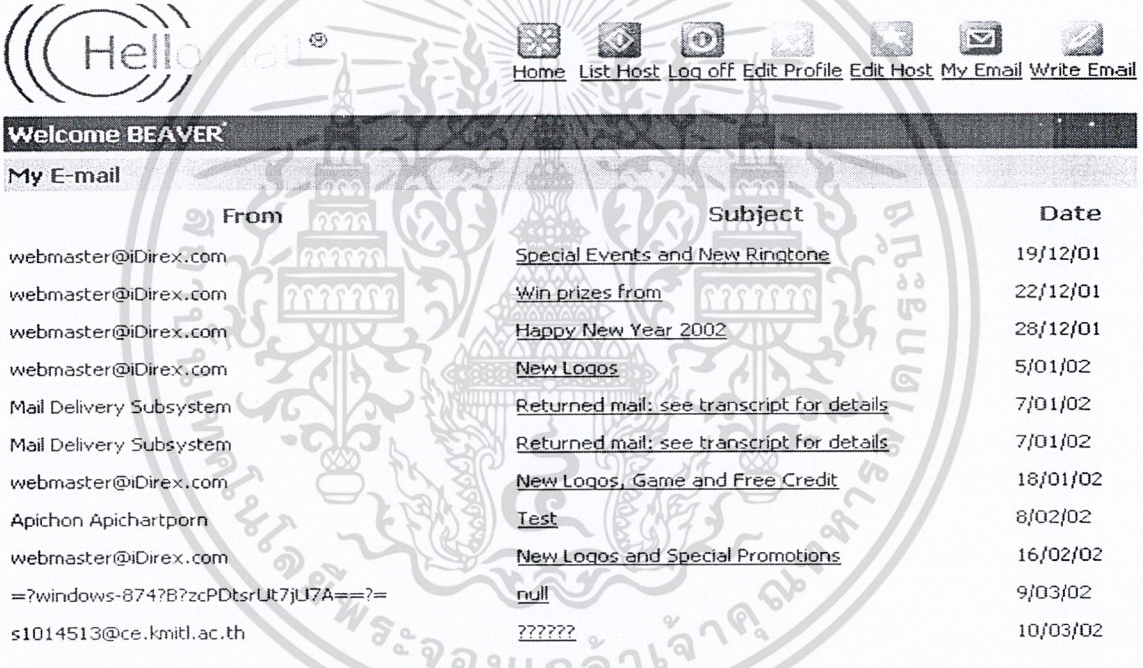
ต่อไปเป็นหน้าจอแสดงผลเมื่อเกิดความผิดพลาดขณะทำการเช็คอีเมล์



รูปที่ 9-14 ไม่สามารถทำการพิสูจน์สิทธิ์ได้



รูปที่ 9-15 ไม่ได้สร้างทางเชื่อมต่อสำหรับเช็คอีเมล์



รูปที่ 9-16 แสดงรายละเอียดอีเมล์ที่อยู่ใน Account

Welcome BEAVER

My E-mail

From : webmaster@iDirex.com 16/02/02

To : s1014484@ce.kmitl.ac.th

Subject : New Logos and Special Promotions

สวัสดีครับ สมาชิกทุกท่าน Logo ใหม่ประจำสัปดาห์นี้ Come on Baby, Couple, Heart Beat, I Love U, Love, Boy and Girl, Love ya, Women and Men, Friend 4 Ever และ ดึ๋กะหมวย ในช่วงนี้ iDirex.com ร่วมกับ Silkspan มีโปรโมชั่นต่างๆไว้บริการคุณ โปรดอย่ารอช้าครับ ที่ http://www.idirex.com/iSilk_promo.html ข่าวดี ! สำหรับผู้ที่ต้องการรับผลการแข่งขันฟุตบอลโลก2002 ทาง SMS ฟรี เพียงคุณช่วยกันเข้าไปเลือกรับได้แล้วครับที่ Event Reminder ในส่วน Special ->World cup ครับ หากมีจำนวนคนมากพอที่ทางสปอนเซอร์ต้องการ เราก็สามารถจัดกิจกรรมดี ๆ อย่างนี้เพื่อสมาชิกได้ http://www.idirex.com/even.asp สำหรับท่านที่แต้มหมดก็สามารถซื้อแต้มเพิ่มอีกได้ที่ My iDirex - Buy Points (http://www.idirex.com/buyp.asp) อีกนิดสำหรับท่านที่โอนเงินเข้ามาแล้ว อย่าลืมแฟกซ์ใบ Pay In เข้ามานะครับ ถ้าท่านใดยังไม่ได้รับการเพิ่มแต้มติดต่อกลับมาที่ iDirex ได้เลยครับ ขอบคุณครับ ชัฏชัยยศ webmaster@idirex.com Tel: +662-257-0334-5 Fax: +662-257-0478 iDirex.com - Thailand's first wireless advertising service. Dear iDirex members This week new Logo - Come on Baby, Couple, Heart Beat, I Love U, Love, Boy and Girl, Love ya, Women and Men, Friend 4 Ever and ดึ๋กะหมวย IDirex joint with Silkspan provide special services and promotions for you at http://www.idirex.com/iSilk_promo.html Get free SMS Worldcup 2002 daily report just click Event Reminder World cup icon. If there is enough number of people as required by sponsor we will organize this activity for you. At http://www.idirex.com/even.asp Members who want to buy more points, please go to My iDirex - Buy Points http://www.idirex.com/buyp.asp If you have transferred your money, don't forget to fax your Pay In krub. Thank You Chatchaiyos webmaster@idirex.com Tel: +662-257-0334-5 Fax: +662-257-0478 iDirex.com - Thailand's first wireless advertising service.

รูปที่ 9-17 แสดงรายละเอียดเมล์

Welcome BEAVER

Write E-mail

From : s1014484@kmitl.ac.th Ex. s1014484@kmitl.ac.th

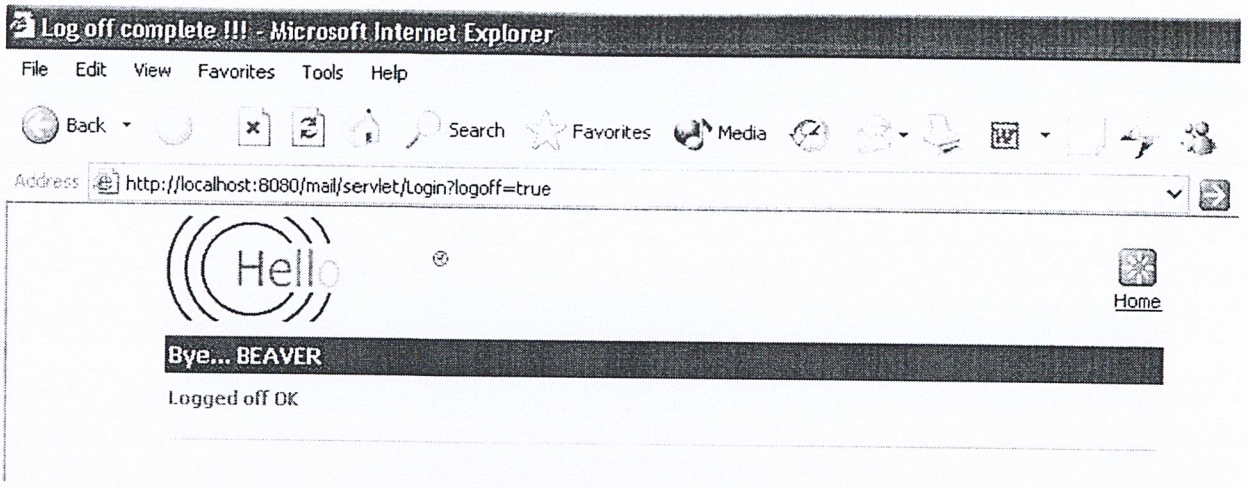
To : s1014485@kmitl.ac.th Ex. s1014484@kmitl.ac.th

Subject : Hello

Hello Kan how r you?

Send Reset

รูปที่ 9-18 แสดงหน้าจอสำหรับส่งอีเมล์



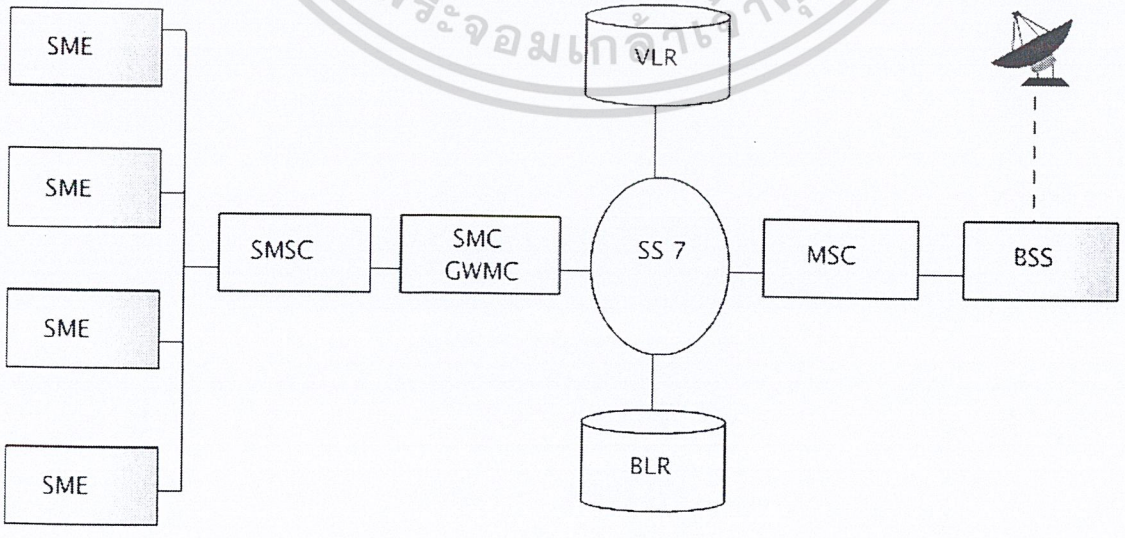
รูปที่ 9-19 แสดงหน้าจอ Logoff

9.3.2 โปรแกรมตรวจสอบอีเมล์ที่มาใหม่

Short Message Service (SMS) เป็นบริการที่ใช้กันอย่างกว้างขวาง ใช้ในการส่งข้อมูลที่เป็นตัวเลขตัวอักษร (Alphanumeric) ระหว่างโทรศัพท์มือถือกับระบบภายนอก เช่น จดหมายอิเล็กทรอนิกส์ ระบบเพจ (Paging System) และระบบวอยซ์เมล์ (Voice Mail System)

SMS เป็นสื่อที่มีประโยชน์อย่างมหาศาล ข้อดีของ SMS ได้แก่

- เพิ่มประสิทธิภาพให้แก่การสื่อสารแบบไร้สาย (Wireless)
- เป็นอีกทางเลือกสำหรับการส่งข้อมูลที่เป็นตัวเลขและตัวอักษร
- ให้ข้อมูลกับผู้ใช้ได้ง่าย
- เพิ่มประสิทธิภาพให้กับบริการต่าง ๆ เช่น จดหมายอิเล็กทรอนิกส์ วอยซ์เมล์ และยังสามารถผสมผสานกับระบบแฟกซ์ได้ แจ้งบริการตารางสายการบิน เป็นต้น



รูปที่ 9-20 องค์ประกอบของ SMS

จากรูปที่ 9-20 แสดงโครงสร้างเครือข่ายของ SMS อย่างง่ายซึ่งประกอบด้วย

- **Short Messaging Entity** (SME) SME จะคอยรับและส่งเมสเซจ โดยจะระบุเครือข่ายที่แน่นอน เช่น เครือข่ายโทรศัพท์มือถือหรือศูนย์ให้บริการอื่น ๆ
- **Short Message Service Center** (SMSC) รับผิดชอบในการเก็บและส่งต่อเมสเซจระหว่าง SME กับเครือข่ายโทรศัพท์มือถือ
- **SMS-Gateway MSC** ใช้ความสามารถของ MSC ในการรับเมสเซจจาก SMSC โดยจะถามข้อมูลจาก HLR สำหรับข้อมูลในการหาเส้นทางและส่งข้อมูลไปให้ยัง MSC ของเครือข่ายของผู้รับ
- **Home Location Register** (HLR) เป็นฐานข้อมูลสำหรับเก็บข้อมูลและจัดการกับผู้ที่เป็นสมาชิกและให้บริการข้อมูลของผู้ใช้ ซึ่งจะให้ข้อมูลของเส้นที่ชี้ไปยังผู้ใช้แก่ SMSC
- **Mobile Switching Center** (MSC) ทำหน้าที่สับเปลี่ยนฟังก์ชันของระบบและควบคุมการใช้งานของโทรศัพท์ว่าโทรไปหาใครและใครโทรเข้ามา ควบคุมการใช้งานฐานข้อมูล
- **Visitor Location Register** (VLR) เป็นฐานข้อมูลซึ่งประกอบด้วยข้อมูลชั่วคราวเกี่ยวกับผู้ใช้ ซึ่งข้อมูลนี้จำเป็นสำหรับ MSC ในการให้บริการแก่ผู้ใช้

โพรเซสที่จำเป็นใน SMS

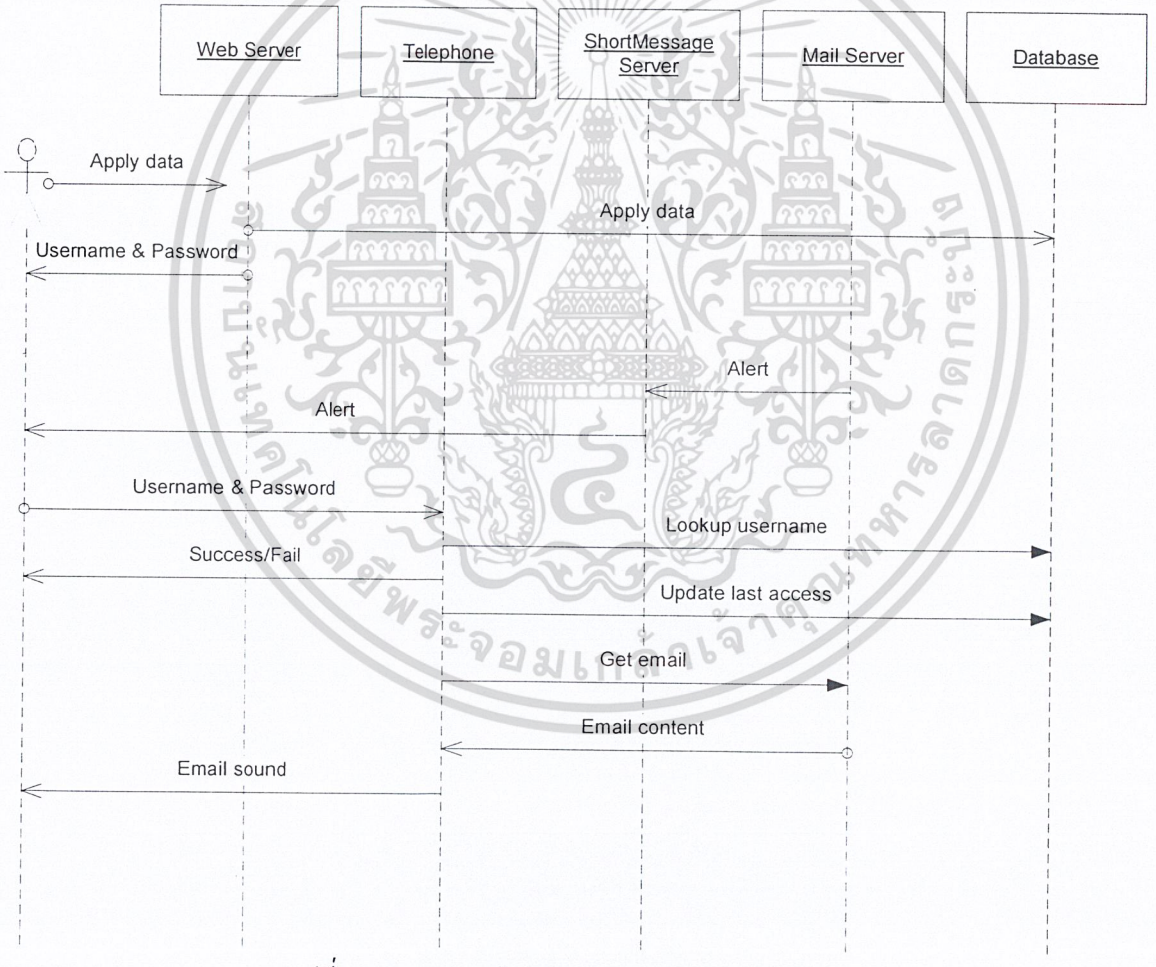
- **Routing Information Request** SMSC จำเป็นต้องทราบข้อมูลของเส้นทางที่ชี้ไปยังผู้ใช้จาก HLR ในการระบุที่อยู่ของเครือข่ายของผู้ใช้ โพรเซสนี้จะเกิดก่อนที่จะมีการส่งเมสเซจ
- **Point to Point Short Message Delivery** เป็นกลไกที่ SMSC ใช้ในการส่งเมสเซจไปยัง MSC ที่ระบุโดยที่อยู่เครือข่ายของผู้ใช้
- **Shortmessage waiting indication** โพรเซสนี้จะถูกทำเมื่อ เมสเซจที่ส่งมายัง SMSC ผิดพลาด โพรเซสนี้จะสั่งให้ HLR เพิ่มที่อยู่ของ SMSC ไปในรายชื่อของ SMSC เพื่อให้ง่ายในการระบุเครือข่าย
- **Service Center Alert** โพรเซสนี้จะให้ข้อมูลของ SMSC ที่เคยส่งเมสเซจไม่สำเร็จมาก่อน ซึ่งทำให้จำที่อยู่ที่เคยส่งไปในครั้งก่อนได้

โปรแกรมตรวจสอบอีเมล์ใหม่เป็นโปรแกรมที่ทำงานตลอดเวลาโดยจะเข้าไปอ่านเมล์ทุกช่วงเวลาที่กำหนด เมื่ออ่านอีเมล์มาแล้วก็จะทำการเปรียบเทียบกับเวลาครั้งหลังสุดที่เคยทำการเช็คเมล์ไปแล้วว่ามีอีเมล์ใหม่เข้ามาบ้างหรือไม่ถ้ามีก็จะส่ง SMS ไปบอกผู้ใช้ว่ามีอีเมล์ใหม่เข้ามา ผู้ใช้ก็จะสามารถรู้ล่วงหน้าว่ามีเมล์เข้ามาใหม่โดยที่ไม่ต้องเข้ามาเช็คเองเลย และถ้าต้องการตรวจสอบอีเมล์ด้วยตนเองก็สามารถโทรศัพท์เข้ามาฟังได้หรือจะเข้ามาเช็คทางเว็บไซท์ก็ได้

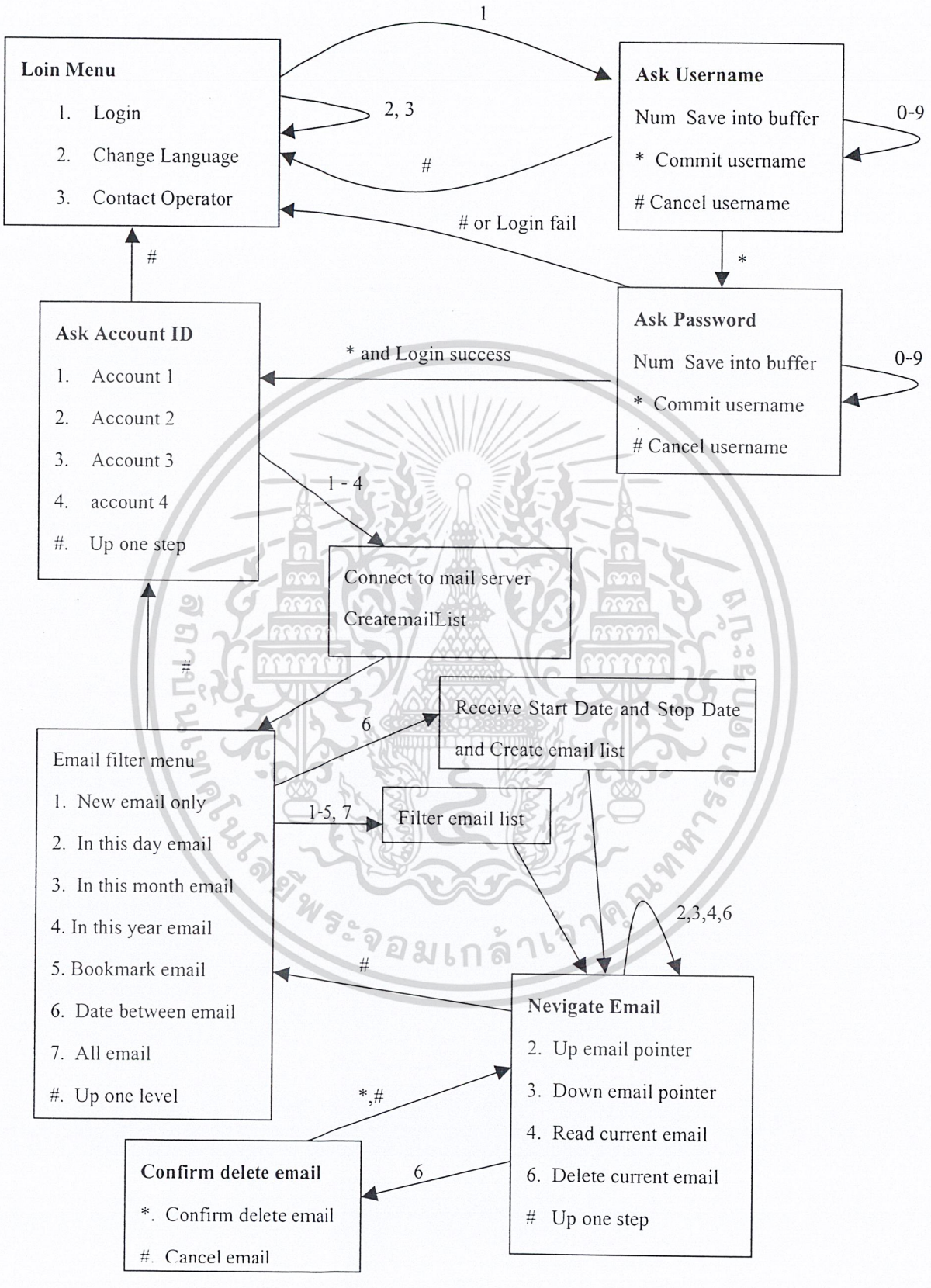
9.3.3 โปรแกรมบริการรับโทรศัพท์อัตโนมัติและอ่านอีเมล์ตามที่ผู้ใช้ต้องการ

เป็นแอพพลิเคชั่นสำหรับฟังอีเมล์ผ่านทางโทรศัพท์ มีการเรียกใช้งาน TAPI, SAPI และติดต่อไปยัง Mail Server ด้วย Winsock โดยใช้โพรโตคอล POP3 ผู้ใช้จะต้องลงทะเบียนเป็นสมาชิกในเว็บไซท์ก่อนจึงจะสามารถล็อกอินผ่านทางโทรศัพท์ได้ เมื่อผู้ใช้ล็อกอินผ่านทางโทรศัพท์ได้แล้ว โปรแกรมก็จะให้เลือกว่าจะติดต่อไปยังแอคเคาน์ใด เมื่อผู้ใช้เลือกโปรแกรมก็จะโหลดอีเมล์ทั้งหมดมาเก็บไว้ (เฉพาะ plain/text) และ พูดสรุปว่ามีอีเมล์เข้ามาเท่าไร มีอีเมล์ใหม่กี่เมล์ เป็นต้น ผู้ใช้สามารถเลือกได้ว่าจะดูเฉพาะ อีเมล์ที่มาใหม่ อีเมล์ที่เข้ามาในวันนี้ อีเมล์ที่เข้ามาในเดือนนี้ อีเมล์ที่เข้ามาในปีนี้ อีเมล์ที่สนใจ อีเมล์ที่ทั้งหมด หรือ อีเมล์ที่อยู่ในช่วงวันที่กำหนด หลังจากนั้นก็สามารถเลือกอีเมล์เพื่ออ่านหรือลบ

เราสามารถเขียนซีเควนไดอะแกรมและสเตทไดอะแกรมของระบบ EmailPhone ได้ดังนี้

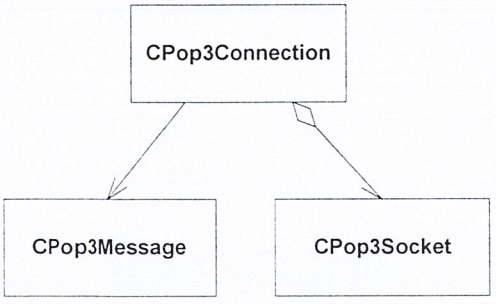


รูปที่ 9-21 Sequence Diagramของระบบ EmailPhone

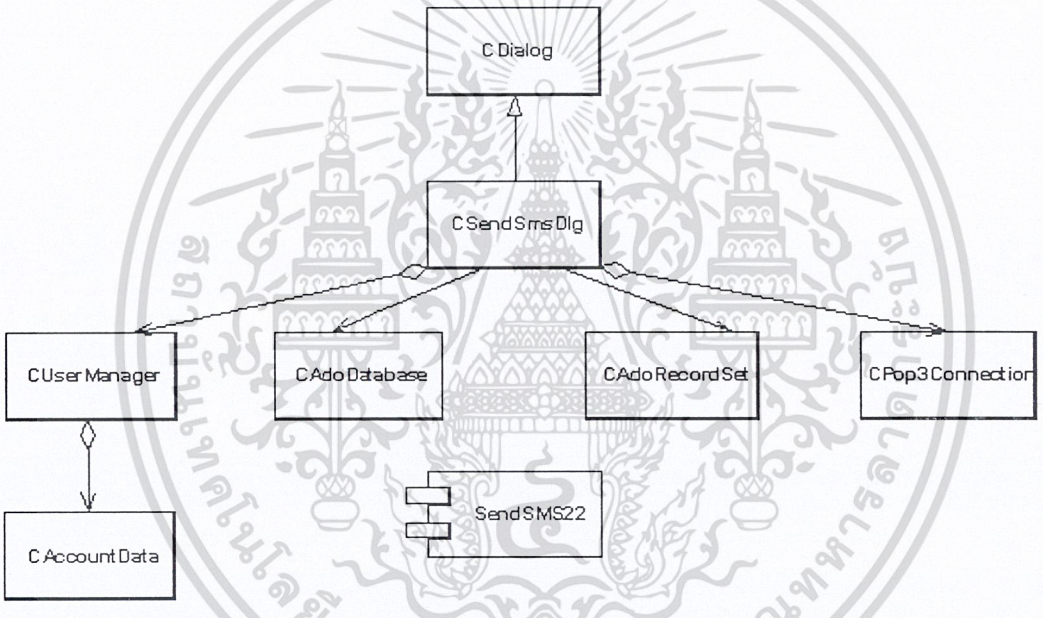


รูปที่ 9-22 State Diagram ของโปรแกรมรับโทรศัพท์

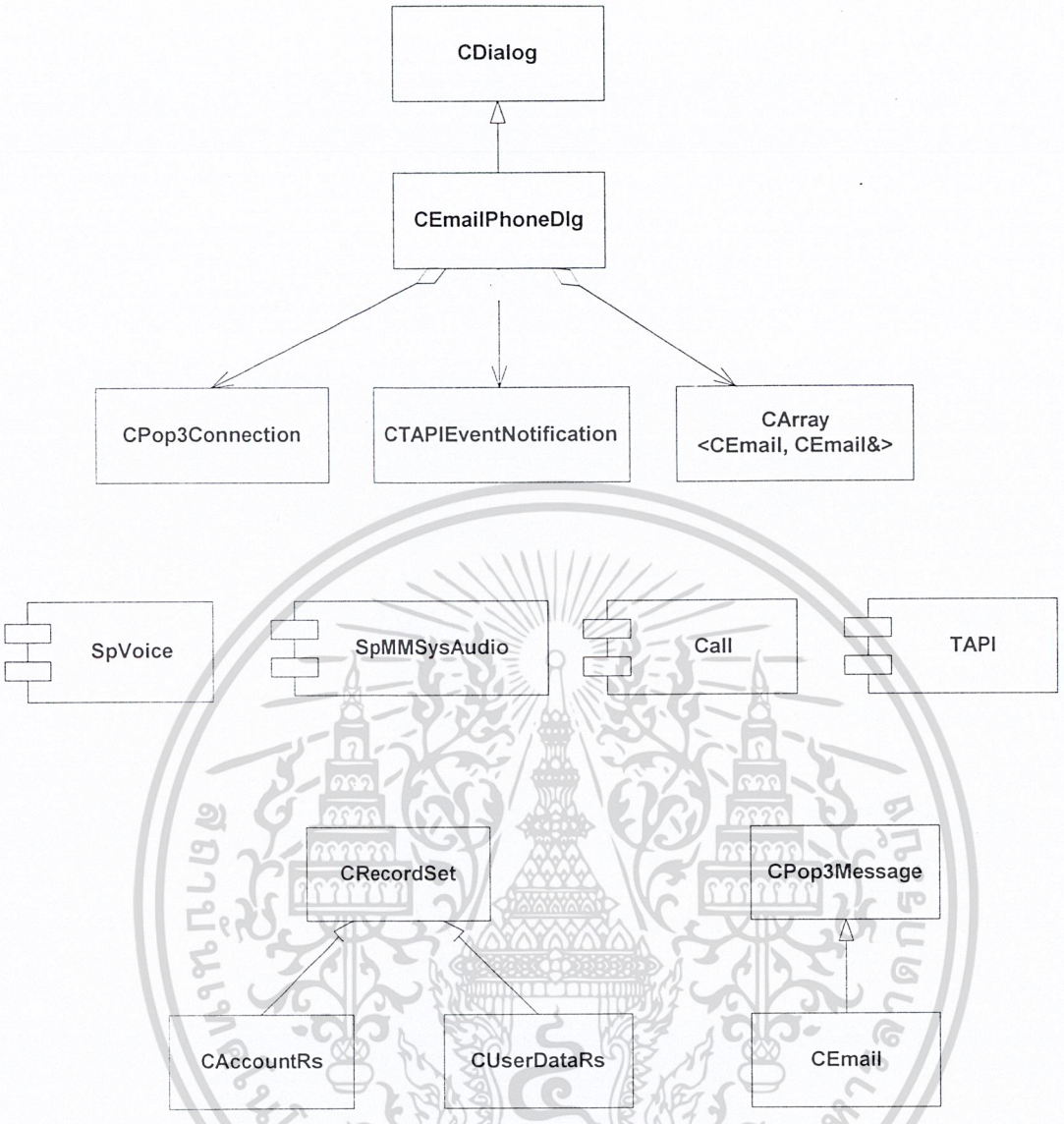
9.4 Class Diagram ของระบบทั้งหมด



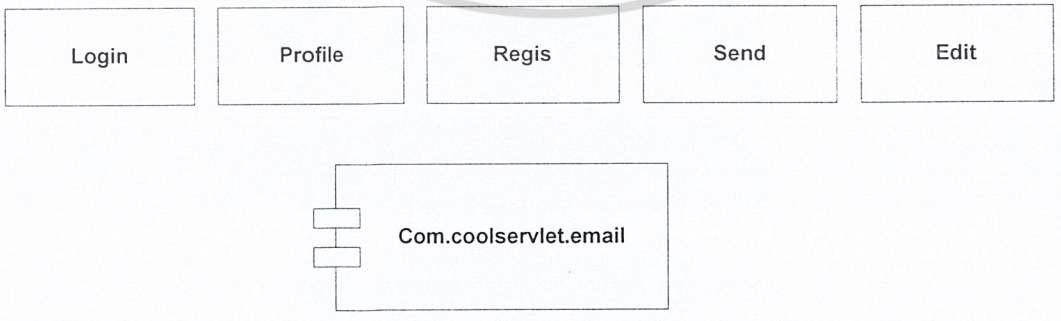
รูปที่ 9-23 Class Diagram ของ CPop3Connection



รูปที่ 9-24 Class Diagram ของโปรแกรมตรวจสอบอีเมล์ที่เข้ามาใหม่



รูปที่ 9-25 Class Diagram ของโปรแกรมอ่านเมล์ผ่านโทรศัพท์



รูปที่ 9-26 Class Diagram ของเว็บแอพพลิเคชั่น

บทที่ 10

ผลการทดลอง บทสรุปและวิจารณ์

10.1 ผลการทดลองใช้งาน

เอ็นจินภาษาไทยที่สร้างขึ้นสามารถใช้งานได้ดีกับโปรแกรมที่พัฒนาด้วย SAPI มีความชัดเจนและสามารถใช้ควบคู่กับเอ็นจินภาษาอังกฤษได้เป็นอย่างดี อัลกอลิทึมที่ใช้ตัดคำมีความรวดเร็วและถูกต้องในการตัดคำภาษาไทยทั่ว ๆ ไปซึ่งมีความถูกต้องเฉลี่ย 99.59% และในกรณีของข้อความที่ประกอบด้วยคำที่ไม่รู้จักทั้งหมดอัลกอริทึมมีความถูกต้องเฉลี่ย 96.69% คุณภาพเสียงคุณภาพเสียง 16 บิต แซมปลิ้ง 11 กิโลเฮิรตซ์ จำนวน 1 ช่องสัญญาณ (mono) ซึ่งเป็นคุณภาพที่ดีพอสมควร พื้นที่ในการจัดเก็บเสียง 65 MB จำนวนคำที่มีอยู่ในพจนานุกรม 19000 คำ มีเสียงที่แตกต่างกัน 5000 เสียงโดยประมาณ

สำหรับส่วนที่ติดต่อกับอีเมล์ใช้ POP3 สามารถอ่านอีเมล์ได้ดีสำหรับเซิร์ฟเวอร์เครื่องเจ้าคุณและไดมอนด์แต่ถ้าเป็นเมล์เซิร์ฟเวอร์อื่นจะใช้เวลามากน้อยขึ้นอยู่กับระยะทาง จำนวนอีเมล์และความหนาแน่นของข้อมูลโดยเฉลี่ยแล้วประมาณ 1-3 นาที และอีเมล์ที่อ่านมาได้หลายรูปแบบทั้ง html, binary, text ฯลฯ ซึ่งโปรแกรมจะแปลงอีเมล์ให้เป็นเสียงได้เฉพาะรูปแบบ plain/text เท่านั้น

คอมพิวเตอร์ติดต่อกับเครือข่ายโทรศัพท์โดยใช้วอยซ์โมเด็ม ผลปรากฏว่าใช้ได้ดีกับโทรศัพท์สาธารณะ PSTN และ โทรศัพท์ภายใน PBX เสียงที่ส่งไปได้มีความชัดเจน

10.2 สรุปและวิจารณ์

ระบบที่ได้ขึ้นมานี้เกิดจากหลายเทคโนโลยีโดยพยายามประยุกต์ใช้รวมกันให้ได้มากที่สุด ทำให้โปรแกรมต้องการเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูง รวมทั้งการพัฒนาโปรแกรมในปัจจุบันมีลักษณะเชิงวัตถุทำให้การเขียนโปรแกรมง่ายขึ้นแต่จะมีปัญหามากถ้าออกแบบไม่ดีเพราะจะทำให้ดูแลและแก้ไขได้ยาก ดังนั้นการพัฒนาโปรแกรมจึงจำเป็นต้องออกแบบอย่างรอบคอบก่อน นอกจากนี้สิ่งที่จะขาดเสียไม่ได้เลยในการพัฒนาโปรแกรมที่ใช้ API ใดๆ ก็ตาม ก็คือการศึกษา API นั้นให้ถ่องแท้เสียก่อนเพราะขั้นตอนการพัฒนาจะกระทำต่อกันไปเรื่อย ๆ ถ้าผิดที่ขั้นตอนใดขั้นตอนหนึ่งหมายถึงต้องเริ่มใหม่ทั้งหมด ส่วนเทคโนโลยีที่เกี่ยวกับเสียงของภาษาไทยที่พัฒนาขึ้นเป็นเอ็นจินตัวแรกที่ใช้อินเตอร์เฟซของ SAPI ซึ่งในอนาคตระบบการติดต่อระหว่างคอมพิวเตอร์กับคน มีแนวโน้มที่จะใช้เสียงเป็นสื่อมากขึ้น เอ็นจินสังเคราะห์เสียงจะเป็นเหมือนส่วนหนึ่งบนเครื่องคอมพิวเตอร์

10.3 ข้อเสนอแนะ

1. พัฒนาเอ็นจินให้เป็นแบบไดโฟน (diphone) เพื่อลดเนื้อที่ของหน่วยความจำ
2. พัฒนาเอ็นจินให้ออกเสียงภาษาไทยได้อย่างถูกฉันทลักษณ์ (posody)
3. ใช้ Digital Signal Processing แก้ไขเสียงตามพารามิเตอร์ที่ต้องการเช่น เสียงสูงต่ำ พิทช์
4. นำ TAPI และ SAPI ไปพัฒนาระบบอื่น ๆ เช่น การจองตั๋วหนัง ตรวจสอบคะแนนการสอบ การจองโรงแรม
5. อาจใช้ TAPI ส่งอีเมล์เป็นแฟกซ์แทนที่จะเป็นข้อความเสียง

บรรณานุกรม

[1] Microsoft Corporation (January 2001): *"MSDN Library:Platform SDK"* http://msdn.microsoft.com/library/default.asp, January 2001

[2] Microsoft Corporation: *"Micorsoft Speech SDK 5.0"* http://microsoft.com/speech, 2002

[3] Pisit Promchan, WittayaWongvachirapanich and Saanti Chinnakan: *" Morphological Derivative for Unknown Words in Thai text-to-speechSynthesis"* PsTNLP laboratory, http://geocities.com/pisitp 1999

[4] Rational Resource:*"OMG Unified Modeling Language Specification"* http://www.rational.com/media/uml/post.pdf June 1999

[5] Pradit Mittrapiyanuruk, Chatchawarn Hansakunbuntheung, Virongrong Tesprasit and Virach Sornlertlamvanich: *" Issues in Thai Text-to-Speech Synthesis The NECTEC Approach"*, National Electronics and Computer Technology Center (NECTEC) การประชุมวิชาการ ปีงบประมาณ 2543

[6] Virach Sornlertlamvanich, Tanapong Potipiti, Chai Wutiwiwatchai and Pradit Mittrapiyanuruk: *"The State of the Art in Thai Language Processing"*, National Electronics and Computer Technology Center (NECTEC), 1999

[7] [RFC821] Postel, J., *"Simple Mail Transfer Protocol"*, STD 10, RFC 821, USC/Information Sciences Institute, August 1982.

[8] [RFC822] Crocker, D., *"Standard for the Format of ARPA-Internet Text Messages"*, STD 11, RFC 822, University of Delaware, August 1982.

[9] [RFC1321] Rivest, R., *"The MD5 Message-Digest Algorithm"*, RFC 1321, MIT Laboratory for Computer Science, April 1992.

[10] [RFC1730] Crispin, M., *"Internet Message Access Protocol – Version 4"*, RFC 1730, University of Washington, December 1994.

[11] [RFC1734] Myers, J., *"POP3 AUTHentication command"*, RFC 1734, Carnegie Mellon, December 1994.