โปรแกรมจำลองแบบและวิเคราะห์ระบบคิว

QUEUEING SYSTEM SIMULATION AND ANALYSIS PROGRAM

บุญยงค์ แก้วบุดดี

BOONYONG KAEWBUDDEE

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2542

ISBN 974-622-433-6

โปรแกรมจำลองแบบและวิเคราะห์ระบบคิว

QUEUEING SYSTEM SIMULATION AND ANALYSIS PROGRAM

บุญยงค์ แก้วบุดดี

BOONYONG KAEWBUDDEE

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2542

ISBN 974-622-433-6

QUEUEING SYSTEM SIMULATION AND ANALYSIS PROGRAM

BOONYONG KAEWBUDDEE

THIS THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1999

ISBN 974-622-433-6

หัวข้อวิทยานิพนธ์	โปรแกรมจำลองแบบและวิเคราะห์ระบบคิว
	QUEUEING SYSTEM SIMULATION AND ANALYSIS PROGRAM
ชื่อนักศึกษา	นายบุญยงค์ แก้วบุดดี
รหัสประจำตัว	35620015
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2542
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ. ดร. บุญธีร์ เครือตราชู

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอโครงสร้างข้อมูลและอัลกอริธึมในการพัฒนาโปรแกรมจำลองแบบและวิเคราะห์ระบบคิว โดยใช้หลักการจำลองแบบด้วยคอมพิวเตอร์ (Computer Simulation) มาทำการจำลองคุณสมบัติและพฤติกรรมของคิวและโครงข่ายคิวประเภทต่าง ๆ และพัฒนารูปแบบให้ใช้งานได้สะดวกและเห็นการทำงานได้ชัดเจนโดยใช้การติดต่อแบบวินโดว์ (Windows Interface) ผู้ใช้สามารถออกแบบระบบคิวและโครงข่ายคิวได้จากสัญลักษณ์ที่เข้าใจง่ายบนหน้าจอ แก้ไขตัวแปรต่าง ๆแล้วสั่งโปรแกรมจำลองแบบ ได้ออกแบบส่วนวิเคราะห์และสรุปผลการจำลองแบบสำหรับระบบอย่างสมบูรณ์ตามวิธีการทางสถิติ การพัฒนาโปรแกรมใช้หลักการโปรแกรมแบบสนใจวัตถุ (Object Oriented) ซึ่งช่วยให้การสร้างโปรแกรมจำลองแบบที่ซับซ้อนทำได้ง่ายขึ้นและกระทัดรัด โปรแกรมนี้อยู่ในรูปแบบจาวาแอ็บเพล็ต(Java Applet) ซึ่งสามารถแสดงผลผ่านโปรแกรมดูเอกสารอินเตอร์เน็ต(WEB Browser) และสามารถนำโปรแกรมนี้ไปใช้เป็นเครื่องมือในการเรียนรู้ ออกแบบและพัฒนาระบบที่มีคิวเป็นส่วนประกอบได้

Thesis title	Queueing System Simulation and Analysis Program
Student	Mr. Boonyong Kaewbuddee
Student ID.	35620015
Degree	Master of Engineering
Program	Electrical Engineering
Year	1999
Thesis advisor	Asst. Prof. Dr. Boontee Kruatrachue

ABSTRACT

The aim of this thesis is to present data structure and algorithm of queueing system simulation and analysis program which employs the computer simulation technique to study the characteristic of each specific kind of queue and networked queue. User interface is based on windows system's GUI, user-friendly, it makes the program easier to use and the user can easily visualize and understand how it processes each of its functions. The user can design queueing system or networked queue with the significance symbol through the screen, simpler to tailor system parameters. The simulation analysis portion and the conclusion are designed completely under statistical techniques. The program development uses the object oriented programming method that provides powerful and practical features for simulation program. This program is designed under Java Applet which able to display through the Web Browser. It can be used as a tool for learning, designing, and analyzing about the queuing systems.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาเกี่ยวกับการสร้างโปรแกรมจำลองแบบระบบคิว จาก ผศ. ดร. บุญธีร์ เครือตราชู ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกทราบซึ้งในความอนุเคราะห์ด้วยความจริงใจยิ่งจากท่านอาจารย์และขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณพระคุณสำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติที่ให้ทุนการศึกษาระดับปริญญาโทฯ เป็นเวลา 1 ปี

สุดท้ายขอขอบคุณบัณฑิตวิทยาลัย ที่ได้ให้ทุนสนับสนุนการทำวิทยานิพนธ์ครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

บุญยงค์ แก้วบุดดี

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของการศึกษา

การจำลองแบบโดยคอมพิวเตอร์เป็นแนวทางในการออกแบบโมเดลของระบบจริงหรือระบบตามทฤษฎีให้สามารถทำงานได้บนคอมพิวเตอร์และมีการวิเคราะห์ผลลัพธ์จากการวิ่งโปรแกรมด้วย การจำลองแบบนำเสนอหลักการการเรียนรู้จากการกระทำ นั่นคือการเรียนรู้เกี่ยวกับระบบ ต้องสร้างโมเดลขึ้นมาแล้วทำให้โมเดลทำงานได้จริง การใช้วิธีจำลองแบบเป็นกิจกรรมที่เป็นธรรมชาติเช่นเดียวกันกับเด็กเล่นของเด็กเล่นแล้วเรียนรู้บทบาทชีวิต เพื่อความเข้าใจความเป็นจริงและความซับซ้อนต่างๆ จำเป็นต้องสร้างวัตถุเทียมและสามารถทำงานได้เหมือนธรรมชาติ การจำลองแบบโดยคอมพิวเตอร์ถือว่าเป็นบทละครอิเลคทรอนิกส์ เป็นหลักการแบบผสมผสานเพื่อประยุกต์ใช้ในหลากหลายรูปแบบหลายแขนง

ระบบตามธรรมชาติหรือระบบคอมพิวเตอร์ เมื่อวิเคราะห์ถึงองค์ประกอบสำคัญๆ มักจะมีคิวเป็นองค์ประกอบ ดังนั้นการศึกษาระบบจำเป็นต้องศึกษาพฤติกรรมของคิวในอันดับแรก ซึ่งวิธีการที่นิยมใช้ในการหาค่าคำตอบของระบบคือการเขียนโปรแกรมให้มีพฤติกรรมเหมือนกับโมเดลคิวแล้วสังเกตพฤติกรรมของโมเดล ข้อดีของการจำลองแบบคือประหยัด ปลอดภัย และสามารถประยุกต์ใช้งานได้ทั่วไป ประเด็นสำคัญในการพิจารณาสำหรับการสร้างจำลองแบบคือ ค่าใช้จ่ายในการสร้างโปรแกรม ค่าใช้จ่ายในการประมวลผลของซีพียูและการวิ่งโปรแกรม และค่าใช้จ่ายในการวิเคราะห์ข้อมูลสถิติของพฤติกรรมระบบ

ดังนั้น ในการศึกษานี้จึงได้พัฒนาโปรแกรมเลียนแบบโครงสร้างการทำงานของระบบคิวและโครงข่ายคิวเพื่อให้ผู้ใช้เข้าใจองค์ประกอบและการทำงานของคิวได้ง่ายขึ้น โดยถ่ายทอดโครงสร้างออกมาเป็นวัตถุบนระบบวินโดว์ด้วยสัญลักษณ์ที่เข้าใจง่าย ผู้ใช้สามารถออกระบบที่สมบูรณ์ได้ด้วยตัวเองภายใต้ระบบกราฟฟิกแบบวินโดว์และทำการวิ่งโปรแกรมจำลองแบบหาค่าคำตอบของปัญหาได้โดยในโปรแกรมได้ออกแบบส่วนการวิเคราะห์ข้อมูลไว้ด้วย

โครงสร้างข้อมูลของโปรแกรมนี้ได้เน้นออกแบบให้เป็นธรรมชาติและยืดหยุ่นต่อการพัฒนาในอนาคต โดยการประยุกต์ใช้โครงสร้างข้อมูลพื้นฐานเช่น อาเรย์(Array), คิว(Queue) และลิงค์ลิส(Linked List) ซึ่งช่วยให้การโปรแกรมทำได้ง่าย และไม่มีปัญหาเกี่ยวกับหน่วยความจำโปรแกรมมีขนาดเล็ก เนื่องจากโปรแกรมนี้ได้เน้นให้เป็นเครื่องมือในการศึกษาการทำงานของระบบคิวและโครงข่ายคิว เพื่อการกระจายการใช้งานของโปรแกรมสามารถทำได้กว้างขวางมากขึ้นจึงได้ออกแบบให้ทำงานเป็นจาวาแอ็บเพล็ต ซึ่งสามารถใช้โปรแกรมนี้ไปแสดงผลบนเอกสารอินเตอร์เน็ตได้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1. เพื่อศึกษาโครงสร้างของระบบคิวและพัฒนาโปรแกรมสำหรับจำลองแบบการทำงานของระบบคิวประเภทต่างๆ ด้วยโครงสร้างข้อมูลและอัลกอริธึมที่เหมาะสม สามารถจำลองแบบการทำงานได้ทั้งแบบคิวพื้นฐานและแบบโครงข่ายคิว
2. เพื่อให้เป็นเครื่องมือในการเรียนรู้ทฤษฎีคิวได้สะดวกด้วยระบบการติดต่อแบบวินโดว์และผู้ใช้สามารถใช้งานผ่านทางอินเตอร์เน็ตได้
3. เพื่อพัฒนาโปรแกรมใช้เองและสามารถพึ่งตนเองได้

1.3 แนวคิดที่ใช้ในการวิจัย

ในชีวิตประจำวันมักจะได้สัมผัสกับระบบที่มีคิวเป็นส่วนประกอบเสมอเช่น การเข้าแถวรับบริการจากจุดบริการของธนาคาร การจราจรบนท้องถนน หรือแม้แต่ในระบบคอมพิวเตอร์และระบบสื่อสารก็มีคิวประกอบอยู่ด้วย ขณะที่มีผู้เข้ารับบริการจำนวนมากจนเกินอัตราการให้บริการของผู้ให้บริการ จึงเกิดคิวขึ้นโดยอัตโนมัติ

มีทฤษฎีมากมายที่ศึกษาพฤติกรรมของคิวแต่ละประเภทและหาแนวทางในการจัดการเวลารอในคิวให้มีน้อยที่สุดเท่าที่จะเป็นไปได้ นั่นหมายถึงประสิทธิภาพสูงสุดของระบบ โดยทั่วไปมักจะเป็นวิธีทางคณิตศาสตร์เช่น**การประมาณค่า**(Approximation)[2] เป็นต้น ผู้ศึกษาต้องมีพื้นฐานเกี่ยวกับคณิตศาสตร์ระดับสูงจึงจะหาคำตอบให้กับระบบได้ แต่**วิธีการจำลองแบบ**[2]เป็นอีกแนวทางหนึ่งที่มีประสิทธิภาพในการหาค่าคำตอบ ในที่นี้เป็นการจำลองแบบด้วยระบบโปรแกรมคอมพิวเตอร์ซึ่งจะให้ความยืดหยุ่นสูงในการทำงานและหาคำตอบได้อย่างถูกต้องและรวดเร็วแม้ว่าระบบจะมีความซับซ้อนมากก็ตาม ภายใต้ระบบที่ซับซ้อน คิวที่เป็นส่วนประกอบมักจะมีการทำงานเป็นโครงข่ายคิว เพื่อศึกษาการทำงานของโครงข่ายคิวที่สะดวกมากขึ้น จึงพัฒนาเครื่องมือศึกษาคิวให้สามารถจำลองแบบการทำงานโครงข่ายคิวที่ซับซ้อนได้ การพัฒนาโปรแกรมชุดนี้ผู้พัฒนาได้นำระบบวินโดว์มาเป็นส่วนติดต่อกับผู้ใช้ ซึ่งผู้ใช้โดยทั่วไปคุ้นเคยกับวินโดว์ดีอยู่แล้ว การศึกษาองค์ประกอบและพฤติกรรมของคิวผ่านวัตถุบนระบบวินโดว์น่าจะเป็นไปได้ง่ายและช่วยให้เรียนรู้เรื่องคิวได้อย่างรวดเร็ว โปรแกรมนี้ออกแบบด้วยวิธีสนใจวัตถุและแปลงรูปแบบเป็นจาวาแอ็บเพล็ตเพื่อให้สามารถนำเสนอโปรแกรมผ่านทางอินเตอร์เน็ตได้อีกทางหนึ่ง

1.4 ขอบเขตการวิจัย

 วิทยานิพนธ์นี้ได้ศึกษาพฤติกรรมและองค์ประกอบของคิวแต่ละประเภทแล้วนำเสนอทฤษฎีคิวพื้นฐาน(Basic Queueing Theory)ในรูปของกราฟฟิกแบบวินโดว์ ภายใต้โครงสร้างข้อมูล

และอัลกอริธึมที่เหมาะสม โดยโปรแกรมสามารถจำลองแบบการทำงานในโครงข่ายคิวได้ นอกจากนี้ได้เน้นให้เป็นเครื่องมือช่วยศึกษาระบบคิวผ่านทางอินเตอร์เน็ตด้วย

1.5 วิธีที่ใช้ในการดำเนินการวิจัย

เป็นการพัฒนาโปรแกรมจำลองแบบระบบคิวบนระบบวินโดว์โดยวิธีการโปรแกรมแบบสนใจวัตถุ เครื่องมือที่นำมาใช้พัฒนาคือโปรแกรมภาษาจาวา(Java Language) โปรแกรมนี้ทำงานบนระบบวินโดว์ 95 (Windows 95) และอยู่ในรูปจาวาแอ็บเพล็ต

บทนี้กล่าวถึงแนวคิดการออกแบบโมเดลการจำลองแบบและการพัฒนาโปรแกรมตามแบบจำลองนั้น โดยเน้นศึกษาองค์ประกอบและการทำงานของคิวหลายๆประเภทแล้วถ่ายทอดองค์ประกอบออกมาเป็นอ็อปเจคบนวินโดว์ด้วยสัญลักษณ์ที่เข้าใจง่าย ได้เลือกใช้การโปรแกรมแบบสนใจวัตถุโดยภาษาจาวา และแปลงรูปแบบให้เป็นแอ็บเพล็ตเพื่อให้สามารถทำงานบนเอกสารอินเตอร์เน็ตได้ ในบทต่อไปจะกล่าวถึงทฤษฎีคิวที่นำมาเป็นพื้นฐานในการออกแบบโปรแกรม

บทที่ 2

ทฤษฎีคิว(Queueing Theory)

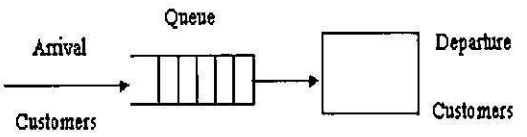
2.1 ทฤษฎีคิว

ทฤษฎีคิวเป็นสาขาหนึ่งของทฤษฎีความน่าจะเป็นที่เกี่ยวข้องกับคณิตศาสตร์ที่ศึกษาคิว ในชีวิตประจำวันมักเห็นเหตุการณ์รอในคิวเสมอเช่นรถที่จอดรอสัญญาณไฟจราจร แถวลูกค้าที่รอรับบริการในธนาคาร เป็นต้น

คิวเกิดจากเหตุการณ์ที่อัตราการเข้ารับบริการมีค่าสูงกว่าอัตราการให้บริการ ในการศึกษาการจำลองแบบมักเริ่มต้นจาการศึกษาระบบคิวเพราะ 1) ระบบที่ซับซ้อนจำนวนมากมีระบบย่อยเป็นส่วนประกอบคือคิว 2) คิวเป็นระบบง่าย ๆ ซึ่งการจำลองแบบสามารถเห็นพฤติกรรมได้ชัดเจน 3) ระบบคิวบางอันมีแบบแผนการวิเคราะห์ค่าคำตอบซึ่งสามารถนำมาประเมินความแม่นยำของการจำลองแบบทางคอมพิวเตอร์ได้ 4) เมื่อพิจารณาโหลดของผู้ให้บริการ ถ้าผู้ให้บริการว่างมากเกินไปการให้บริการนั้นจะไม่มีค่าทางเศรษฐกิจแต่อย่างใดและอาจเกิดการซ้ำซ้อนในระบบ และ 5) ค่าเฉลี่ยความยาวของคิวสามารถนำมาออกแบบสถานที่รองรับลูกค้าได้อย่างเพียงพอ และต้องพิจารณาว่าคุ้มค่าทางเศรษฐกิจหรือไม่ การมีคิวยาวเกินไปก็ไม่เป็นเป็นผลดี การที่ลูกค้าออกไปจากคิวแสดงให้เห็นว่าคุณค่าการบริการไม่ควรค่าแก่การรอ

2.2 โครงสร้างพื้นฐานของระบบคิว

โครงสร้างอย่างง่ายของคิวแสดงด้วยรูปที่ 2.1 หน่วยอินพุทของระบบคิวเรียกว่าลูกค้า (Customer) หรืองาน (Job) ในที่นี่จะใช้คำว่า**ลูกค้า**สำหรับอธิบาย ลูกค้าต้องเรียงกันเข้ามารับบริการ ณ จุดให้บริการ(Server) เมื่อลูกค้าเดินทางมาถึงจุดบริการ ถ้าขณะนั้นผู้ให้บริการว่าง ลูกค้าก็จะเข้ารับบริการและออกจากคิวเมื่อรับบริการเสร็จแล้ว ถ้าผู้ให้บริการไม่ว่าง ลูกค้าต้องเข้ารอคิวเพื่อรอรับบริการในเวลาถัดไปตามกฎของการให้บริการ(Service discipline) พฤติกรรมของระบบคิวมีแตกต่างกันออกไปขึ้นอยู่กับข้อสมมติขององค์ประกอบหลักคือ รูปแบบการมาถึงของลูกค้า(Arrival pattern) , กลไกการให้บริการ(Service mechanism), และกฎของคิว(Queue discipline) ดังที่จะอธิบายต่อไป



รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของระบบคิว

2.2.1 รูปแบบการมาถึง

ลูกค้ามาจากประชากรหรือแหล่งอินพุทของระบบ คุณลักษณะอย่างหนึ่งของลูกค้าคือขนาด(Size) อาจเป็นแบบขนาดจำกัด(Finite)และไม่จำกัด(Infinite) คุณลักษณะอีกอย่างหนึ่งที่สำคัญคือแบบแผนการมาถึงของลูกค้าเมื่อเทียบกับเวลา การมาถึงแบบพื้นฐานคือการมาแบบปกติ (Regular arrivals) หมายถึงลูกค้ามาในระยะเวลาห่างเท่าๆ กัน กำหนดสัญลักษณ์เป็น τ ในที่นี้อัตราการมาถึงคือ $\lambda = 1/\tau$ ต่อหน่วยเวลา แต่ในแง่ความเป็นจริงแล้วการมาถึงแบบนี้เป็นไปได้น้อยแต่ข้อสมมุติทางคณิตศาสตร์ทำได้ง่าย โดยทั่วไป การมาถึงที่นิยมพิจารณาคือแบบสุ่มโดยสมบูรณ์(Completely random arrival process) โดยการศึกษามักอ้างถึงการมาถึงแบบพัวซอง(Poisson arrival process)[5] หรือกล่าวย่อๆว่า พัวซองโพรเซส(Poisson Process) รายละเอียดในหัวข้อ 2.3

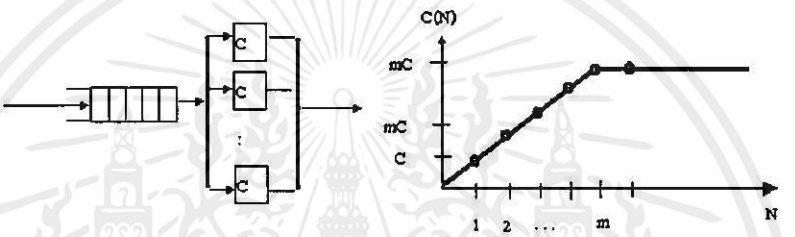
2.2.2 การกระจายการให้บริการ(Service Distribution) และความสามารถของการบริการ (Service Capacity)

องค์ประกอบอย่างที่สองของระบบคิวคือลักษณะการให้บริการแก่ลูกค้าแต่ละคน มักเรียกว่าความต้องการบริการ(Service Demand) หรืองาน(Work) หน่วยของการบริการขึ้นอยู่กับลักษณะของผู้ให้บริการและลูกค้า เช่นถ้าผู้ให้บริการคือซีพียูและลูกค้าคือโปรแกรม หน่วยที่เหมาะสมคือคำสั่ง(Instruction) หรือถ้าผู้ให้บริการคือ Transaction Line และผู้รับบริการคือเมสเซสหรือข้อมูล หน่วยคือบิทหรือไบท์ เป็นต้น ในการคิดโดยทั่วไปจะกำหนดให้ลูกค้าเป็นประเภทเดียวกัน นั่นคือความต้องการในการรับบริการเป็นการกระจายแบบปกติเรียกว่า Service Distribution แต่กรณีระบบที่ซับซ้อนลูกค้าที่เข้ามาจะมีหลายแบบ แล้วแต่ละแบบก็ต้องการการกระจายการบริการที่แตกต่างกัน

ลักษณะการมาถึงและการให้บริการ สองประเด็นนี้ยังไม่เพียงพอในการกำหนดคุณลักษณะของระบบคิว ต้องมีการกำหนดความสามารถของผู้ให้บริการด้วย(Processing Rate) ว่ามีศักยภาพให้บริการเร็วเพียงใดในงานที่กำหนดให้ สมมติกำหนดให้ C (Service Capacity) คือความสามารถของการบริการ กำหนดให้ S (Service Unit)คือจำนวนความต้องการงานของลูกค้า ดังนั้นอัตราส่วนของ S/C เรียกว่าเวลาให้บริการ(Service Time) ค่าคาดหวังของ $\overline{S}/C$ เรียกว่าเวลาการให้บริการเฉลี่ย(Average service time)

เมื่อกลับเศษส่วน จะได้ว่าอัตราการบริการ(Service rate) , $\mu = C / \overline{S}$ โดย C คือค่าคงที่

ในบางกรณี ค่าความสามารถในการบริการเป็นค่าไม่คงที่ ขึ้นอยู่กับภาวะความคับคั่งของแต่ละสถานีบริการ โดยทั่วไปพบในระบบหลายผู้ให้บริการแบบขนาน เช่นมีทั้งหมดคือ m จุดและมีคิวร่วมกัน ในรูปที่ 2.2 (ซ้ายมือ) สมมติว่าเป็นระบบหลายผู้ให้บริการแบบสมดุล(Symmetric multi-server) นั่นคือความสามารถในการให้บริการแต่จุดมีค่าเท่ากัน เท่ากับ C เมื่อกำหนดให้ N เป็นจำนวนลูกค้าที่เข้ามารับบริการในแต่ละสถานี(ทั้งที่รับบริการอยู่และรอในคิว) ดังนั้นสรุปได้ว่า ความสามารถในการให้บริการของระบบรวม (Total Capacity) เป็นสถานะไม่อิสระ(State-dependent) กำหนดโดย $C(N) = \min\{N,m\}C$ ดังแสดงในรูปที่ 2.2 (ขวามือ) แสดงกราฟความสัมพันธ์ระหว่าง $C(N)$ กับ N



รูปที่ 2.2 แสดงความสัมพันธ์ของ $C(N)$ และ N

2.2.3 กฎการสับหลีก(Scheduling)

พิจารณาการที่ลูกค้ามารับบริการ การสับหลีกที่เห็นได้บ่อยๆคือ มาก่อนได้ก่อน FCFS (First-come, first-served) แต่ในสถานการณ์บางอย่างมักมีระดับความสำคัญมาร่วมด้วย จึงต้องจัดการสับหลีกแบบมีลำดับความสำคัญ(Priority scheduling) เรียกคิวประเภทนี้ว่าคิวแบบมีลำดับความสำคัญ(Priority queue) โดยทั่วไปแล้วจะแบ่งคิวแบบมีลำดับความสำคัญเป็นสองประเภทคือ แบบแซงคิวได้ทันที(Preemptive) เมื่อคนที่สำคัญกว่ามาถึงจุดบริการ และแบบแซงคิวไม่ได้ทันที(Non-preemptive) จนกว่าลูกค้าที่รับบริการอยู่เสร็จธุระแล้ว แต่ในบทความนี้สนใจการทำงานแบบมาก่อนได้รับบริการก่อน

2.2.4 การออกแบบส่วนโปรแกรมสุ่มตัวเลข(Random Number Generator)

ตัวแปรสุ่มที่สร้างขึ้นจะนำไปใช้ในการแสดงเวลาให้บริการของผู้ให้บริการ(Server) เวลาการมาถึงของลูกค้า(Arrival) แบบแผนการกระจายของตัวเลขสุ่มมีหลายชนิด แต่แบบแผนการกระจายที่กำหนดใช้ในการศึกษานี้คือเอ็กซ์โปเนนเชียล ส่วนแบบอื่นๆได้กำหนดไว้ในโปรแกรมด้วย

ชื่อฟังก์ชั่นการสุ่ม	ตัวอย่างอินพุทของฟังก์ชั่น
RandUniform	RNG1 (low, high)
RandExponential	RNG2 (mean)
RandNormal	RNG8 (mean, sd)

2.3 พัวซองโพรเซส(Poisson Process)

2.3.1 พัวซองโพรเซส

เมื่อพิจารณาช่วงเวลาจำกัด(Finite time interval) $(0,T)$ และหาการกระจายความน่าจะเป็นของจำนวนของลูกค้าที่เข้ามาถึงในช่วงเวลา ในการคำนวณค่านี้ จะแบ่งช่วงเวลา T เป็นช่วงย่อย ๆ จำนวน m ช่วง โดยความยาวที่ได้คือ $h=T/m$ ในรูปที่ 2.3 ถ้าให้ λ แทนค่าเฉลี่ยอัตราการมาถึงของลูกค้า สำหรับในแต่ละช่วงย่อย ความน่าจะเป็นที่ลูกค้าคนใดคนหนึ่งมาถึงคือ $\lambda h + o(h)$ นั่นคือจำนวนลูกค้าสองคนหรือมากกว่าสองคนมาถึงคือ $o(h)$ ดังนั้นจะได้ว่า เวลาที่ลูกค้าไม่ได้มาถึงคือ $1-\lambda h + o(h)$ โดยที่สัญลักษณ์ $o(h)$ แทนจำนวนใด ๆก็ได้ที่มีค่าเข้าใกล้ 0 ก่อนค่า h เมื่อ $h \to 0$ นั่นคือ $o(h)/h \to 0$ เมื่อ $h \to 0$ เมื่อกล่าวว่ากระบวนการมาถึงคือแบบสุ่มโดยสมบูรณ์หรือพัวซอง ความหมายเป็นดังต่อไปนี้คือ ถ้าสังเกตช่วงย่อยใด ๆ ให้เหตุการณ์เป็นอิสระต่อกันและไม่ทับซ้อนกัน ถ้ากำหนดให้การมาถึงเป็นแบบสำเร็จ(Success) ของการทดลองเบอร์โนลี(Bernoulli trial)[5] แล้วรูปแบบการมาถึงของช่วงเวลา T คือ $T=mh$ แล้วความน่าจะเป็นที่ลูกค้าคนที่ i มารถึงที่ช่วงย่อย m สามารถประมาณค่าได้จากการกระจายแบบไบโนเมียล (Binomial distribution) b(i: m, $\lambda h + o(h)$)

$$\binom{m}{i}[\lambda h + o(h)]^{i}[1-\lambda h + oh]^{m-i} \qquad \text{----------(1)}$$

เมื่อกล่าวถึงลิมิตโดย $h \to 0$ และ $m \to \infty$ เมื่อยังกำหนด $mh=T$ เป็นค่าคงที่ พบว่า $n(T)$ เป็นจำนวนลูกค้ามาถึงในช่วงวลา T มีความน่าจะเป็น คือ

$$P[m(T)=i] = \frac{(\lambda T)^{i}}{i!} \lim_{m\to 0} \frac{m!}{m^{i}(m-i)!} \lim_{m\to\alpha} (1-\frac{\lambda T}{m})^{m-i}]$$

$$= \frac{(\lambda T)^{i}}{i!} e^{-\lambda T} \qquad \text{----------(2)}$$

ซึ่งนี้คือการกระจายแบบพัวซอง (Poisson distribution) สังเกตได้ว่าสมการ (2) มี λT เป็นตัวแปรการกระจาย ค่าเฉลี่ยเลขคณิต(Mean) และค่าความแปรปรวน(Variance) ของตัวแปรสุ่ม $n(T)$ มีค่าเท่ากับ λT สามารถอนุมานได้ว่า $E[n(T)/T] \to \lambda$ โดยที่ $\mathrm{Var}[n(T)/T] = \lambda T \to 0$ เมื่อ $T \to 0$ ดังนั้นค่า $n(T)/T$ ลู่เข้าสู่ λ เมื่อ $T \to \infty$ และกำหนดให้ λ แทนอัตราการมาถึง(Arrival rate) ในพัวซองโพรเซส รูปที่ 2.3 แสดงกราฟการกระจายของแต่ละ λT

รูปที่ 2.3 แสดงการแบ่งช่วง *(0,T)* ออกเป็นช่วงย่อยทั้งหมด m ช่วง

คุณสมบัติที่สำคัญอีกประการหนึ่งของพัวซองโพรเซสคือการกระจายของช่วงที่อยู่ระหว่างเวลาการมาถึงแต่ละค่า ให้ X เป็นช่วงเวลาจากเวลาเริ่มต้นใดๆ ถึงช่วงเวลาอันแรกสุด จะได้ว่าค่าการกระจายของ X ที่ไม่มีค่าใดๆและไม่มีการมาถึงเกิดขึ้นในช่วง (0,x) ถ้ากำหนดให้ $X > x$ เท่านั้น

$$P[X>x] = P[n(x)=0] \quad \text{----------(3)}$$

โดยที่ n(x) แทนจำนวนของการมาถึงระหว่างเวลา x หน่วย จากสมการที่ (2) ทราบว่า $P[n(x)=0]=e^{-\lambda x}$

$$o(h) = 2h^2, o(h) = \frac{h^2}{2!} + \frac{h^3}{3!} + \ldots$$

----------(4)

และทราบว่า $o(h)+o(h) = o(h), o(h)-o(h) = o(h)$, เป็นต้น

กำหนดให้ $F_x(x)$ เป็นฟังก์ชันการกระจายของ X และ $f_x(x)$ เป็นฟังก์ชันความน่าจะเป็น จะได้ว่า

$$F_x(x) = 1 - e^{-\lambda x}, x \geq 0 \quad \text{----------(5)}$$

และโดย

$$f_x(x) = F'_x(x) = \lambda e^{-\lambda x}, x \geq 0 \quad \text{----------(6)}$$

ตามลำดับ ดังนั้นสำหรับกระบวนการมาถึงแบบพัวซอง ช่วง X ระหว่างค่าคงที่ใดๆและเวลาของช่วงแรกเป็นการกระจายแบบเอกซ์โปเนนเชียลด้วยค่าเฉลี่ยเท่ากับ $1/\lambda$

ฟังก์ชันการกระจายในสมการที่ (5) สามารถหาได้อีกวิธีโดยการกำหนดตัวแปรดังนี้ เลือกเวลาเริ่มต้น ณ ค่าที่ใด ๆ และกำหนดให้ $P_0(x)$ แทนความน่าจะเป็นที่ไม่มีลูกค้ามาถึงในช่วงเวลา (0,x)

$$P_0(x)=P[n(x)=0] \quad \text{----------(7)}$$

จะได้ว่า

$$P_0(x+dx)=P[\text{no customers arrive during}(0,x) \text{ nor during } (x, x+dx]$$
$$= P_0(x)\{1-dx+ o(dx)\} \quad \text{----------(8)}$$

จากสมการ (7) ถ้าใช้คุณสมบัติการเป็นอิสระทางสถิติของการมาถึงแบบพัวซองและอาศัยกฎการคูณของความน่าจะเป็นของเหตุการณ์ที่มีเงื่อนไข ได้ว่า

$$\{P_0(x+dx)-P_0(x)\}/dx = P_0(x)\{-\lambda+o(dx)\} \quad \text{----------(9)}$$

เมื่อลิมิต $dx \longrightarrow 0$ พบว่า

$$P'_0(x) = -\lambda P_0 x(x) \quad \text{----------(10)}$$

เมื่อนำสมการ (10) มาทำการ differential ได้ดังนี้

$$P'_0(x) = -\alpha e^{-\lambda x} \quad \text{----------(11)}$$

โดยที่ α คือค่าคงที่ เพราะว่าค่า $P_0(0) = 1$ โดยนิยาม เมื่อพิจารณาที่ค่า α=1 ได้ว่า

$$P_0(x) = e^{-\lambda x} \quad \text{----------(12)}$$

จากสมการ (3), (7), และ (12) ได้สมการฟังก์ชันการกระจาย $F_x(x)$ ตามสมการ (5) และคุณสมบัติที่สำคัญอื่น ๆของพัวซองโพรเซสมีดังต่อไปนี้

2.4 คุณสมบัติของพัวซองโพรเซส[5]

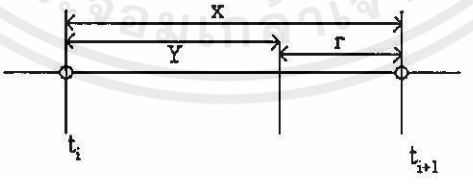
2.4.1 คุณสมบัติ Memoryless ของพัวซองโพรเซส

ให้ t_i แทนเวลาของการมาถึงครั้งที่ i_{th} และสมมติให้เวลา Y หน่วยแทนช่วงเวลาก่อนที่ลูกค้าคนถัดไปมาถึง ดังแสดงในรูปที่ 2.4 เป้าหมายเพื่อหาความน่าจะเป็นสำหรับลูกค้าคนถัดไปจะมาถึงภายในเวลา r หน่วย หรืออาจกล่าวอีกกรณีหนึ่งคือต้องการหาค่าความน่าจะเป็น $P[R \le r \mid X \ge Y]$ โดยที่ R=X-Y แทนเวลาที่ยังมีอยู่จนกระทั่งถึงเวลาการมาถึงครั้งถัดไป โดยการประยุกต์ใช้ความน่าจะเป็นแบบมีเงื่อนไข เขียนสมการได้ดังนี้

$$\frac{P[R \le r \mid X \ge Y]}{P[X \ge Y]} = \frac{P[Y \le X \le Y + r]}{P[X \ge Y]}$$

$$= \frac{e^{-\lambda Y} - e^{-\lambda (Y+r)}}{e^{-\lambda r}}$$

$$= 1 - e^{-\lambda r} \quad \text{----------(13)}$$



รูปที่ 2.4 คุณสมบัติ Memoryless ของพัวซองโพรเซส

ดังนั้นการกระจายแบบมีเงื่อนไขของ R เป็นอิสระต่อ Y และมีการการกระจายเป็นเหมือนกับเวลาการมาถึง นั่นคือเป็นไปตามสมการ (5) ที่กล่าวมาข้างต้น

นั่นคือพัวซองโพรเซสมีคุณสมบัติ Memoryless ในการคำนวณความน่าจะเป็นของเวลาที่ยังมีอยู่ก่อนการมาถึงครั้งถัดไป ซึ่งไม่จำเป็นต้องพิจารณากรณีการมาถึงครั้งสุดท้าย

2.4.2 คุณสมบัติ Superposition ของพัวซองโพรเซส

พิจารณา m คือแหล่งอินพุทอิสระที่สร้างสายลูกค้าให้กับระบบ และสมมติว่าลูกค้าแต่ละสายมีแบบแผนเป็นพัวซองด้วยอัตรา λ_K โดย K=1,2,3,...,m ถ้ารวมสายลูกค้าเหล่านี้เข้าด้วยกันเป็นสายเดียว ตามรูปที่ 2.5 แล้วจะได้พัวซองโพรเซสด้วยอัตราที่เป็นผลรวมของทุกๆองค์ประกอบ นั่นคือ $\lambda=\lambda_1+\lambda_2+\lambda_3+...+\lambda_m$ การรวมพัวซองโพรเซสสามารถแสดงได้ด้วยการใช้ความน่าจะเป็นของฟังก์ชันการกำเนิด(Generating Function Method) โดยพิจารณาช่วงความยาว T แล้วจำนวนของการมาถึงจาก K_{th} แหล่งในช่วงนี้คือการกระจายพัวซองด้วยตัวแปร $\lambda_K T$ และความน่าจะเป็นของฟังก์ชันกำเนิดของช่วง T จากสมการของฟังก์ชันกำเนิด

$$G_k(z) = e^{-\lambda k(T-1-z)} \qquad \text{----------(14)}$$

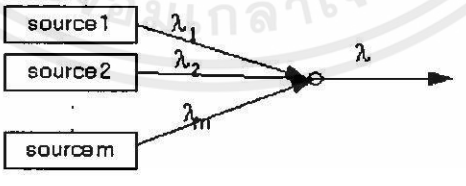
ดังนั้นจำนวนการมาถึงทั้งหมดจากทุกๆแหล่งมีความน่าจะเป็นของฟังก์ชันกำเนิดคือ

$$G_k(z) = \prod_{k=1}^{m} G_k(Z) = e^{-\lambda T(1-z)} \qquad \text{----------(15)}$$

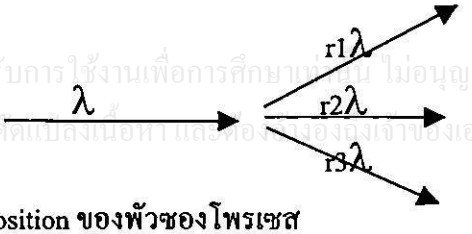
with

$$\lambda = \sum_{k=1}^{m} \lambda_k \qquad \text{----------(16)}$$

โดยรูปแบบผลคูณของสมการ (14) จะเป็นไปตามค่าสถิติของ m แหล่งที่เป็นอิสระต่อกัน ดังนั้นจำนวนการมาถึงในสายที่รวมกันเป็นการกระจายด้วยค่าพัวซอง λT



รูปที่ 2.5 แสดง Superposition ของพัวซองโพเซส



รูปที่ 2.6 Decomposition ของพัวซองโพรเซส

2.4.3 คุณสมบัติ Decomposition ของพัวซองโพรเซส

เมื่อพิจารณากรณีสายพัวซองแตกออกเป็น m แบบดังแสดงในรูปที่ 2.6 ถ้าอัตราอินพุทเป็น λ และแต่ละสายที่แตกออกไปมีแบบการมาถึงของแต่ละสายเป็นอิสระต่อกันด้วยความน่าจะเป็น r_K แล้วสามารถแสดงได้ว่า แต่ละ K_{th} สายเอาท์พุทเป็นพัวซองโพรเซสด้วยอัตรา $r_k\lambda$ โดย k= 1,2,3,...m ซึ่งค่าสาย k เหล่านี้เป็นอิสระทางสถิติด้วย

ให้ n(T) แทนจำนวนของอินพุทการมาถึงในช่วงเวลา T หน่วย และให้ $n_k(T)$ แทนจำนวนของอินพุทการมาถึงที่แตกออกไปเป็น K_{th}สาย แล้วการกระจายร่วมเงื่อนไขของ $n_k(T)$ (k=1,2,3,...,m) เมื่อให้ n(T) = n เป็นการกระจายชนิดมัลติโนเมียล(Multimomial)

$$P[n_1(T)=n_1, n_2(T)=n_2,..., n_m(T)=n_m \mid n(T)=n = \frac{n!}{n_1!n_2!..n_m!} r_1^{n_1} r_2^{n_2} ... r_m^{n_m}$$

----------(17)

โดยการคูณความน่าจะเป็นของตัวแปรสุ่ม n ซึ่งเป็นการกระจายพัวซองด้วยตัวแปรλ(T) จะได้ว่า

$$P[n_1,n_2,...,n_m] = \frac{n!}{n_1!n_2!..n_m!} r_1^{n_1} r_2^{n_2} .. r_m^{n_m} \frac{(\lambda T)^n}{n!} e^{\lambda T}$$

$$= \prod_{k=1}^{m} \frac{(r_k \lambda T)^{n_k}}{n_k!} e^{-n_k \lambda T}$$

----------(18)

เพราะว่าองค์ประกอบความน่าจะเป็นร่วมทั้งหมด m ค่าเป็นการกระจายพัวซอง ตัวแปรสุ่ม n_1,n_2,n_m เป็นอิสระกันสำหรับช่วง T ใดๆที่เลือกใช้ ดังนั้นสายเอาท์พุทจึงเป็นพัวซองโพรเซสอิสระ

2.5 การกระจายเวลาการบริการ(Service Distribution)

2.5.1 การกระจายแบบเอ็กซ์โปเนนเชียล(Exponential Distribution)[5]

การกระจายนี้เป็นวิธีพื้นฐานที่ทฤษฎีคิวที่นิยมใช้ในการศึกษา ค่าเฉลี่ยของการกระจายนี้ปกติแทนด้วยด้วย $1/\mu$

$$F_x(t) = P[S \le t] = 1-e^{-\mu t}$$ ----------(19)

ฟังก์ชันความน่าจะเป็นที่สอดคล้องกันคือ

$$f_s(t) = F_S(t)] = \mu e^{-\mu t}$$ ----------(20)

และค่าเบี่ยงเบนมาตราฐานของการกระจายเอ็กซ์โปเนนเชียล มีค่าเท่ากับค่าเฉลี่ยคือ

$$\bar{S} = E[S] = \frac{1}{\mu}$$

และ ----------(21)

$$\delta_s^2 = E[S - \bar{S}] = \frac{1}{\mu^2}$$

----------(22)

ผู้ให้บริการที่บริการด้วยการกระจายแบบเอ็กซ์โปเนนเชียลด้วยค่าเฉลี่ย $1/\mu$ เรียกว่า ผู้ให้บริการเอ็กซ์โปเนเนนเชียล(Exponential Server) ด้วยค่าอัตราการบริการ μ

คุณสมบัติที่สำคัญของการกระจายการให้บริการเอ็กซ์โปเนนเชียลคือ memoryless property [12] ไม่ว่าบริการ Y ที่ลูกค้าได้รับจะนานเพียงใด การกระจายของความน่าจะเป็นของ *residual life* = S-Y สามารถแทนได้ด้วยค่าการกระจายเอ็กซ์โปเนนเชียลเดียวกัน

$$P[R \leq r \mid S \geq Y] = 1 - e^{-\mu t}$$ ----------(23)

จากผลลัพธ์นี้เมื่อพิจารณาตามหลัก Uniformity ของการให้บริการแล้วเสร็จ ความน่าจะเป็นที่บริการให้ลูกค้าจะแล้วเสร็จในช่วงเวลา Δt โดยการให้บริการดำเนินการมาเป็นเวลา Y หน่วยเวลา คือ

$$P[0 \leq R \leq \Delta t \mid S \geq Y] = P[Y \leq S \leq Y + \Delta t]/P[S \geq Y$$ ----------(24)

$$= \{Fs(Y+\Delta t - Fs(Y)\}/\{1-Fs(Y)\}$$

$$= 1 - e^{-\mu \Delta t} = \mu \Delta t + o(\Delta t^2)$$

ซึ่งเป็นอิสระต่อ Y อาจกล่าวอีกมุมหนึ่งได้ว่า ความน่าจะเป็นของการให้บริการจะแล้วเสร็จในช่วงเวลาค่าเล็ก ๆเป็นค่าคงที่ เป็นอิสระต่อจำนวนการให้บริการที่ลูกค้าได้รับไป

2.6 ตัวอย่างระบบจำลองแบบคิวพื้นฐานที่ใช้ในการวิจัย

รูปแบบสัญลักษณ์สำหรับอธิบายระบบคิวมีรูปแบบคือ A/S/c/k/m [6] โดยที่

A แทนการกระจายเวลาการมาถึงของลูกค้า (Inter-arrival time distribution)

S แทนการกระจายของเวลาการบริการ (Service time distribution)

c แทนจำนวนผู้ให้บริการ (Servers)

k แทนจำนวนสูงสุดของลูกค้าที่ยอมให้มีในคิว (Total Customers)

m แทนจำนวนของลูกค้าที่สามารถมีได้ (Customer Available at Source)

ในการอธิบายระบบคิวนิยมใช้สัญลักษณ์ดังต่อไปนี้

D คือค่าคงที่ของการมาถึง (Constant inter-arrival) หรือ เวลาให้บริการ (Service time)

M คือ การกระจายแบบเอกซ์โปเนนเชียล(Exponential)

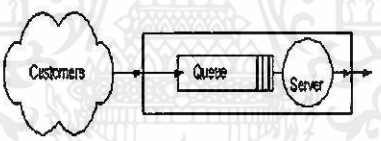
Ek คือ k-Erlang Distribution[1]

Hk คือ k-stage Hyperexponential distribution[2]

G คือการกระจายแบบปกติใดๆ (General distribution)

ถ้าขนาดลูกค้าเป็นแบบไม่จำกัด มักจะไม่เอาค่า m มาคำนวณในแบบจำลอง(Model) ในทำนองเดียวกันถ้าความจุ (Capacity) ของระบบเป็นแบบไม่จำกัดก็จะไม่เอาค่า k มาคำนวณเช่นกัน สมมุติให้กฎควบคุมคิวเป็นแบบมาก่อนได้รับบริการก่อน(First-in First-out) เมื่อพิจารณาตามสัญลักษณ์ที่กำหนดข้างต้นจะได้ว่า ระบบคิวที่มีรูปแบบการมาเป็นเอ็กซ์โปเนนเชียล เวลาการให้บริการแบบคงที่ และมีผู้ให้บริการคนเดียว สามารถเขียนแทนได้คือระบบคิว M/D/1 หรือ ระบบที่มีการมาถึงแบบเอ็กซ์โปเนนเชียล เวลาให้บริการแบบปกติ จำนวนผู้ให้บริการสองคน และสามารถจุลูกค้าในคิวได้ k คน สามารถเขียนแทนด้วย M/G/2/k เป็นต้น

สมมุติให้ระบบมีผู้ให้บริการคนเดียว ดังรูปที่ 2.7 งานของระบบจะมาจากประชากรที่ไม่จำกัด การมาถึงของประชากรเป็นการกระจายแบบเอกซ์โปเนนเชียล ด้วยค่าเฉลี่ย Ta เมื่อประชากรมาถึง ถ้าผู้ให้บริการว่างก็สามารถเข้ารับบริการได้ แต่ถ้าผู้ให้บริการไม่ว่างก็ต้องเข้าคิว ในที่นี้กำหนดให้ขนาดของคิวเป็นแบบไม่จำกัด มีกฎแบบมาก่อนได้ก่อนและเวลาการบริการเป็นแบบเอกซ์โปเนนเชียล ด้วยค่าเฉลี่ย Ts สามารถเขียนสัญลักษณ์แทนโมเดลได้คือ M/M/1



รูปที่ 2.7 ระบบคิวที่มีผู้ให้บริการคนเดียว

สมมุติกำหนดให้ Ta=200 หน่วยเวลา , Ts=100 หน่วยเวลา, เวลาในการวิ่งจำลองแบบ=200,000 หน่วยเวลา(คาดว่ามีการมาถึงของลูกค้าประมาณ 1000 ครั้ง) , Time แทนเวลาในการจำลองแบบ ,n แทนจำนวนลูกค้าในระบบ ณ จุดเวลาใด ๆ มีเหตุการณ์ (Event) 2 แบบคือที่เกิดขึ้น *แบบที่ 1* แทนภาวะการมาถึง *แบบที่ 2* แทนภาวะการรับบริการโดย เหตุการณ์จะสัมพันธ์กับเวลา t1 และ t2 ซึ่งจะเป็นเวลาบอกเหตุการณ์ถัดไปที่จะเกิดขึ้น กำหนดให้ t1 เป็นเวลาของการมาเวลาถัดไป (Next Arrival) และ t2 เป็นเวลาของรับบริการช่วงถัดไป (Next Completion) การพิจารณาว่าเหตุการณ์ถัดไปจะเป็นอะไรได้จากการนำเวลา t1 และ t2 มาเปรียบเทียบกัน ถ้า t1<t2 แล้วเหตุการณ์ถัดไปคือการมาถึงแล้วเวลา (Time) จะถูกเซ็ตให้เป็นเวลาของการมาถึง มีการเพิ่มจำนวนประชากร (n)ในระบบ

[1]k-Erlang การการจายที่เป็นขนาดผลรวมจาก k แบบของการกระจายแบบเอกซ์โปเนนเชียลที่เหมือนกัน

[2]k-stage การการจายที่เป็นขนาดผลรวมจาก k แบบของการกระจายแบบเอกซ์โปเนนเชียลที่แตกต่างกัน

จากนั้นจะคำนวณเวลาสำหรับเหตุการณ์ถัดไปที่จะเกิดขึ้น ถ้าผู้ให้บริการว่างขณะที่ลูกค้าเข้ามา เวลาที่เป็นของการบริการจะถูกคำนวณค่าใหม่ ถ้า t1>t2 เหตุการถัดไปคือการให้บริการ ตัวแปรเวลาจะถูกเพิ่มให้เป็นของเวลาให้บริการ จำนวนประชากรจะถูกลดลง ถ้ายังมีประชากรอยู่ในระบบหลังจากที่ให้บริการเสร็จแล้วผู้ให้บริการก็จะให้บริการต่อไป จึงต้องคำนวณเวลาให้เป็นของการให้บริการ ถ้าเวลาปัจจุบันของการให้บริการยังมีอยู่แต่ไม่มีลูกค้าเข้ามา เวลา t2 จะถูกเซ็ตใหม่เพื่อบังคับว่าเหตุการณ์ถัดไปต้องเป็นการมาถึงคล้ายกับเป็นเหตุการณ์ครั้งแรกสุด

```
//Java code MM1
import  java.lang.Math;
import  java.util.Random;
import  java.io.*;
public class mm1{
     public static void main( String args[] ){
          int ta=200,ts=100,te=20000,t1,t2,time;
          int n;   n=0;t1=0;t2=te;time=0;
          XRandom r = new XRandom();
          System.out.println("te   time   t1   t2    ta    ts    n");
          while(time<te){
           if(t1<t2){//event 1: arrival
            time=t1;n++;
            t1=time+(r.get(ta));
            if(n==1){
              t2=time+ t1+(r.get(ts));
            }
           }
           else{ //event 2: completion
            time=t2;n--;
            if(n>0){t2=time+(r.get(ts));}
            else{ t2=te;}
           }
            System.out.println(te+"  "+time+"  "+t1+"  "+t2+"  "+ta+"
            "+ts+"   "+n);
          }//while
     }//main
}//mm1
```

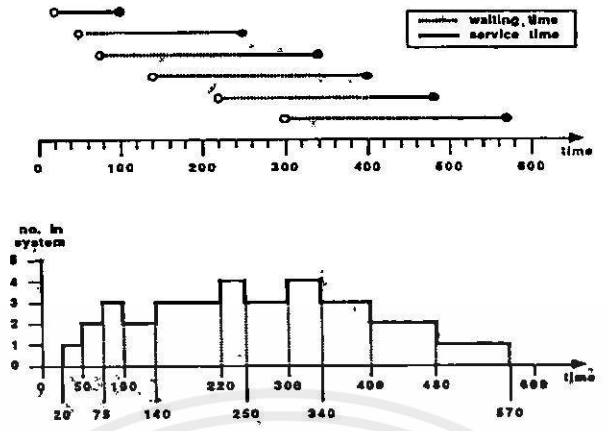
ตัวอย่างโปรแกรมจำลองระบบ M/M/1

กราฟแสดงพฤติกรรมของระบบ

สมมุติว่าเวลาในการจำลองแบบเท่ากับ 600 หน่วยเวลา ในโปรแกรมที่กล่าวมานี้และวาดกราฟการทำงานของโมเดล แสดงในรูปที่ 2.8

สุ่มเวลาการมาถึงได้เป็น 20 30 25 65 80 80 50 70 50 60 หน่วยเวลาตามลำดับ

สุ่มเวลาการให้บริการได้เป็น 80 150 90 60 80 90 80 60 70 100 หน่วยเวลาตามลำดับ



รูปที่ 2.8 กราฟแสดงพฤติกรรมของระบบคิวตัวอย่าง

2.6.1 การวัดค่าทางสถิติของโมเดล

สมมุติให้เวลาของระบบทั้งหมด คือ T จำนวนลูกค้าที่มาถึงทั้งหมดคือ A และจำนวนที่ได้รับบริการเสร็จแล้วคือ C จะได้ว่า อัตราการมาถึง (Arrival Rate) :

$$\lambda = A/T \quad \text{----------(25)}$$

และงานที่ทำได้ต่อเวลา(Throughput Rate) :

$$X = C/T \quad \text{----------(26)}$$

ในการพิจารณาระบบจริงอาจจะมีลูกค้ารับบริการอยู่ทั้งเวลาจุดเริ่มต้นและเวลาจุดสิ้นสุดของช่วงเวลาที่วัด อย่างไรก็ตามถ้าสมมุติว่าเวลานั้นยาวนานพอ จะพบว่าจำนวนลูกค้าที่เข้ามาควรใกล้เคียงกับจำนวนลูกค้าที่ได้รับบริการ ($C \approx A$) นั่นคือสมมุติให้ $\lambda = X$ ข้อสมมุตินี้เรียกว่า *Fow Balance* [6] จากข้อสมมุตินี้จำเป็นต้องนับจำนวนของการมาถึงหรือจำนวนที่ได้รับบริการอย่างใดอย่างหนึ่ง สมมุติว่า ในการวัดสนใจค่าเฉลี่ยของเวลาให้บริการ (Server Busy Time) B จะได้ว่า การใช้ประโยชน์ของผู้ให้บริการ(Server Utilization):

$$U = B/T \quad \text{----------(27)}$$

และค่าเฉลี่ยเวลาให้บริการลูกค้า(Mean Service Time Per Customer):

$$Ts = B/C \quad \text{----------(28)}$$

Utilization Law.[6] นำสมการ (26) (27) (28) มารวมกัน จะได้ Utilization Law ดังนี้

$$U = B/T,\ U = B/(C/X) = BX/C = XTs$$

$$U = XTs \quad \text{----------(29)}$$

ถ้าสมมุติตาม Flow Balance

$$U = \lambda Ts \quad \text{----------(30)}$$

สามารถอธิบายได้ว่า Utilization ของผู้ให้บริการคือผลคูณของ Throughput Rate (X) กับ เวลาการบริการเฉลี่ย (Ts) นั่นเอง

Little's Law.[5]ให้ L คือค่าเฉลี่ยของจำนวนลูกค้าในระบบในช่วงเวลาที่วัด และ W คือเวลาเฉลี่ยที่ลูกค้าใช้ในระบบ กำหนดให้ Wi เป็นเวลาที่ใช้ในระบบโดยลูกค้าคนที่ ith จะได้ว่า $W=\sum wi/C$ จากรูปที่ 2.8 จะได้ว่าจำนวนลูกค้าโดยเฉลี่ยในระบบเท่ากับค่าเฉลี่ยความสูงของกราฟ หรือค่าของพื้นที่ใต้กราฟหารด้วยช่วงเวลาทั้งหมด ลูกค้าแต่ละคนจะสร้างพื้นที่ขนาด 1xWi ผลรวมของพื้นที่คือ $\sum wi$ ของจำนวนลูกค้าทั้งหมดที่รับบริการในช่วงเวลาการจำลองแบบ ซึ่งมีค่าเท่ากับ WC จำนวนลูกค้าเฉลี่ยคือพื้นที่ใต้กราฟหารด้วยเวลาทั้งหมดคือ $L=WC/T$ ดังนั้นจากสมการที่ $C/T = X$ จะได้ว่า

$$L=XW \qquad \text{----------(31)}$$

ตามข้อสมมุติ Flow Balance ;

$$L=\lambda W \qquad \text{-- ----------(33)}$$

อธิบายได้ว่าจำนวนเฉลี่ยของลูกค้าในระบบคือผลคูณของ System's Throughput Rate และ เวลาเฉลี่ยที่ลูกค้าใช้ในระบบ ถ้ากำหนดให้ Lq ค่าเฉลี่ยของจำนวนลูกค้าที่เข้าคิวในระบบ(ที่ยังไม่ได้รับบริการ) และให้ Wq เวลาเฉลี่ยที่รอในคิว สามารถประยุกต์ใช้ Little' Law ได้ว่า

$$Lq = \lambda Wq \qquad \text{----------(34)}$$

เนื่องจาก $L=Lq+U$ และ $W=Wq+Ts$ ซึ่งสามารถพิจารณา Utilization Law จาก Little Law โดยให้ U ค่าเฉลี่ยจำนวนลูกค้าที่กำลังรับบริการอยู่

จากรูปที่ 2.8 มีจำนวน Completion = 6 , เวลาที่วัด = 600 หน่วย และค่าอื่น ๆ ตามรูปจะได้ว่า

Throughput : X=6/600=0.01

Total Busy Time: B=(100-20)+(250-100)+(340-250)+(400-340)+(480-400)+(570-480)=550

Mean Service Time : Ts = 550/6=91.7

Utilization : U=550/600=0.917 หรือ U=0.01*91.7 = 0.917

Residence Time Sum : $\sum wi$ = (100-20)+(250-50)+(340-75)+(400-140)+(480-220)+(570-300)=1335

Mean Residence Time : W=1335/6 =222.5

Mean Queuing Time : Wq = 222.5-91.7 = 130.8

Mean Number In System : L = 1335/600 = 2.225

Mean Number In Queue : Lq = 2.225-0.917=1.308

2.6.2 อัลกอริธึมสำหรับวัดค่าทางสถิติของโมเดล [6]

กำหนดให้ T คือเวลาทั้งหมดในการวัด , C คือ จำนวน Service Completion , B คือ Server Busy Time มีสองวิธีการในการวัดโมเดลเพื่อหาค่าจำนวนลูกค้าเฉลี่ยในคิว (L) และค่าเฉลี่ยของเวลาที่เข้ามาในระบบ (Mean Residence Time ,W) *วิธีการแรก*เก็บเวลาสะสมของลูกค้าที่เข้ามาในระบบจนรับบริการเสร็จ เมื่อเวลาจำลองแบบหมดลง นำค่าผลรวมนี้มาหารด้วย C จะได้ค่าเฉลี่ยของเวลาที่เข้ามาในระบบ แล้วใช้ Little's Law คำนวณหาค่า L ต่อไป สิ่งที่จำเป็นอีกอย่างหนึ่งของวิธีนี้คือต้องเก็บเวลาที่ลูกค้ามาถึงแต่ละจุดของผู้ให้บริการด้วย(ตามกราฟรูปที่ 2.8 ด้านบน) *วิธีการสอง*คือวัดค่า L โดยตรง จากรูปที่ 2.8(ล่าง) ค่าจำนวนลูกค้าเฉลี่ยในระบบคือพื้นที่ใต้กราฟหารด้วยเวลาทั้งหมด พื้นที่นี้หาได้จากการรวมเวลาที่เข้ามาในระบบของลูกค้าแต่ละคน อย่างไรก็ตามค่าพื้นที่นี้หามาจากพื้นที่สี่เหลี่ยม ความสูงของสี่เหลี่ยมคือจำนวนลูกค้าในระบบ ส่วนอีกแกนหนึ่งคือช่วงเวลาที่เปลี่ยนแปลงจำนวนลูกค้า ดังนั้นอัลกอริธึมที่นำมาวัดค่านี้ มีการกำหนด s และ tn มาควบคุม สมมุติว่า n คือจำนวนลูกค้าในระบบทั้งหมด จุดเริ่มต้นของการวัดให้ $s=0$ และเซ็ตเวลา tn เป็นเวลาปัจจุบันของการจำลองแบบ(time) แต่ละเวลาที่ลูกค้าเข้ามาในระบบ ต้องเก็บค่าสถิติดังนี้

s +=n(time-tn); n++;tn=time*

และแต่ละเวลาที่ลูกค้าแสร็จจากการรับบริการ ต้องเก็บ

s +=n(time-tn); n--;tn=time*

หลังจากหมดช่วงเวลาการวิ่งแล้ว นำค่า s มาหารด้วยช่วงเวลาทั้งหมดก็จะได้ค่าเฉลี่ยจำนวนที่ลูกค้าเข้ามาในระบบ โปรแกรมต่อไปนี้เป็นตัวอย่างการทำงานตามอัลกอริธึมแบบที่สอง

```
//MM1 with performance measurement algorithm
import java.lang.Math;
import java.util.Random;
import java.io.*;
public class mm1x{
    int ta,ts,te;
    public static void main( String args[] ) {
    XRandom r = new XRandom();
    int ta=200,ts=100,te=20000;
    int t1,t2,time; double B,C,L,s,tb,tn,U,W,X;
    int n,totalcustomer; n=0;t1=0;t2=te;time=0;totalcustomer=0;
    B=s=0.0;
    L=C=tb=tn=W=X=0.0;tn=time;
     while(time<te){
         if(t1<t2){/*event 1: arrival */
         time=t1;s+=n*(time-tn);n++;
               tn=time;totalcustomer++;
         t1=time+(r.get(ta));
         if(n==1){ tb=time;t2=t1+time+(r.get(ts));}
         }
         else{/*event 2: completion */
           time=t2;s+=n*(time-tn);n--;tn=time;C++;
           if(n>0){t2=time+(r.get(ts));}
            else{ t2=te;B+=time-tb;}
         }System.out.println(C);
```

```
    }//while
   System.out.println("Simulation time   ="+time);
   System.out.println("Total customer    ="+totalcustomer);
   System.out.println("Total completion ,C="+C);
   System.out.println("busy time , B   ="+B);
   X=C/time;  System.out.println("throughput,X=C/time="+X);
   U=B/time;  System.out.println("utilization,U=B/time="+U);
   L=s/time; System.out.println("mean no.in system,L=s/time="+L);
   W=L/X;System.out.println("mean residence time,W=L/X="+W);
   }//main
}//mm1x
```

2.6.3 การวิเคราะห์ผลลัพธ์ของการจำลองแบบ

ประเด็นที่สนใจมีดังนี้

1. การวิ่งโปรแกรมควรวิ่งนานเท่าไรจึงจะเหมะสมและได้ค่าที่น่าเชื่อถือได้?
2. วัดความถูกต้องของผลลัพธ์ได้อย่างไร?
3. ปัญหาที่เกี่ยวข้องกับการวิ่งโปรแกรมเป็นเวลานานๆ คือค่าใช้จ่าย และ ความอดทน จะจัดการอย่างไร?
4. อินพุทของโปรแกรมต้องมาจากการสุ่มโดยมีการกระจายของความเป็นไปได้ตามข้อกำหนด
5. ข้อมูลที่ได้จากโปรแกรมเป็นตัวแปรของตัวเลขสุ่ม ดังนั้นจึงต้องวิเคราะห์ด้วยสถิติเช่นกัน

2.6.3.1 ชนิดของการวิ่งโปรแกรมจำลองแบบ

ถ้าพิจารณาจากผลของโปรแกรม ชนิดของการวิ่งระบบมีสองอย่าง คือวิ่งจนกระทั่งจบโปรแกรม(Terminating) และผลขณะวิ่งโปรแกรมหรือภาวะไม่เสถียร(Transient) ตัวอย่างของภาวะไม่เสถียรเช่นขณะที่เริ่มต้นระบบ

สิ่งที่ต้องพิจารณาคือต้องวิ่งโปรแกรมกี่รอบ(ด้วยค่าตัวเลขสุ่มที่แตกต่างกัน) จึงจะได้ผลลัพธ์ที่ถูกต้อง ในภาวะที่ระบบเสถียร การวัดค่ามักกำหนดเวลาไว้ที่จุดใดจุดหนึ่งหรือเป้าหมายใด ๆที่วัดได้ภายในเวลาที่วิ่งโปรแกรม สมมุติว่าสนใจค่าเฉลี่ยของเวลารอในคิว ถ้ากำหนดเวลาวิ่งโปรแกรมเป็นไม่จำกัด ค่าการกระจายของเวลารอในคิวไม่มีการเปลี่ยนแปลงและค่าเฉลี่ยของเวลารอในคิวลู่เข้าลิมิตค่าใดค่าหนึ่ง ผลลัพธ์ที่ได้น่าจะเป็นตัวแทนคำตอบที่ดีของระบบ ในทางปฏิบัติมักกำหนดให้เวลาการวิ่งเป็นแบบจำกัด และการวิ่งโปรแกรมจะได้ข้อมูลเป็น sample set จากการกระจายของอินพุทที่กำหนดให้ สิ่งที่นำมาวิเคราะห์คือค่าเฉลี่ยที่ได้จาก sample set มีค่าใกล้เคียงกับ true mean ของค่าการกระจายอินพุทหรือไม่เมื่อระบบเสถียร โดยนิยาม ค่าผลลัพธ์ต้องเป็นอิสระต่อเงื่อนไขเริ่มต้น

2.6.3.2 วิธีการวัดความถูกต้องของโมเดล [6]

โดยส่วนใหญ่ การวัดประสิทธิภาพที่สนใจคือค่าเฉลี่ยของผลลัพธ์การการวิ่งโปรแกรม ค่าตัวแปรอาจเป็นเวลาแบบต่อเนื่อง(Continuous time) หรือแบบเวลาไม่ต่อเนื่อง (Discrete time) ซึ่ง

ขึ้นอยู่กับลักษณะอินพุท สำหรับกรณีเวลาไม่ต่อเนื่อง การวิ่งโปรแกรมจะให้ผลลัพธ์ออกมาเป็นลำดับ n ค่าคือ $X_1, X_2, \ldots, X_n$ ซึ่งหาค่าเฉลี่ย $\bar{X}$ ได้คือ

$$\bar{X} = \sum_{i=1}^{n} Xi / n \qquad \text{----------(35)}$$

สำหรับชนิดเวลาต่อเนื่อง ตัวแปร X มี sample value เท่ากับ Xt ที่ค่าของเวลา t ใดๆ และค่าเฉลี่ยคือ

$$\bar{X} = \int_{0}^{T} Xtdt / T \qquad \text{----------(36)}$$

โดย T คือช่วงเวลาของการวิ่งโปรแกรม ค่า $\bar{X}$ เรียก Sample mean เพราะว่าฟังก์ชันของ T เป็นตัวแปรสุ่มดังนั้นค่าเฉลี่ยนี้จึงเป็นตัวแปรสุ่มด้วย ถ้าลำดับของผลลัพธ์และช่วงเวลาต่างกัน ค่าเฉลี่ยที่ได้จะต่างกันด้วย

อย่างไรก็ตาม ขณะที่ n มีค่าเข้าใกล้ infinity ในสมการ (35) หรือ T เข้าสู่ Infinity ในสมการ(36) ค่า $\bar{X}$ จะลู่เข้าสู่ค่าหนึ่งคือ $E[X]$ เรียกว่าค่าคาดหวังของ X โดยทั่วไป มักแทนค่าคาดหวังด้วย μ หรืออาจกล่าวได้อีกว่า μ คือค่าเฉลี่ยจริงของการกระจายของ X เรียกว่า Distribution mean

ค่าเฉลี่ยในคิวหรือค่าเฉลี่ยเวลาที่ลูกค้าอยู่ในระบบเป็นตัวอย่างของโพรเซส(Process)ชนิดไม่ต่อเนื่อง ขณะที่การกระจายเวลารอในคิวและเวลาที่ลูกค้าอยู่ในระบบเป็นโพรเซสแบบต่อเนื่อง การจำลองแบบจะให้ผลลัพธ์ออกมาเป็นค่าไม่ต่อเนื่อง และวัดค่าความน่าจะเป็นด้วยค่าเฉลี่ยของโพรเซสไม่ต่อเนื่อง ตัวอย่างเช่น ถ้าต้องการพิจารณาความน่าจะเป็นที่ลูกค้ามาถึงและรอในคิวของระบบในขณะที่ผู้ให้บริการไม่ว่าง กำหนดให้ผลลัพธ์ Xi เท่ากับ 1 ถ้าลูกค้าวิ่งเข้ามาและลูกค้าคนที่ i พบว่าผู้ให้บริการไม่ว่าง ในทางตรงกันข้ามให้เซ็ตเป็น 0 และการวิ่งโปรแกรมมีลูกค้าทั้งหมด n คน ค่าเฉลี่ยของ X หาได้จากสมการ (35) ซึ่งเป็นค่าโดยประมาณของค่าความน่าจะเป็นที่ต้องการ

ค่าเฉลี่ยความยาวของคิวและค่าอัตราการใช้ประโยชน์ (Facility utilization) เป็นอีกตัวอย่างหนึ่งที่นิยมนำมาพิจารณาในโพรเซสแบบต่อเนื่อง โดยเป็นไปตามสมการ (36)

ในบางครั้งการวัดเพียงค่าเฉลี่ยอาจไม่เพียงพอสำหรับบางระบบเช่นระบบโต้ตอบ ที่มักสนใจเวลาตอบสนอง ถ้าต้องการวัดค่าการกระจายของเวลาตอบสนอง ซึ่งหาได้จากค่าความแปรปรวนของการกระจาย ดังนั้นค่าความแปรปรวนของค่า $X_1, X_2, \ldots X_n$ คือ

$$S^2 = \sum_{i=1}^{n} (Xi - \bar{x})^2 / (n-1) \qquad \text{----------(37)}$$

S^2 เรียกว่าค่าแปรปรวน เมื่อ n เข้าสู่อินฟินิตี้ S^2 จะลู่เข้าสู่ค่าลิมิต $E[(X-\mu)^2]$ แทนด้วย σ^2 ดังนั้นค่า $\bar{X}$ และ S^2 เป็นค่าเฉลี่ยและค่าแปรปรวน และค่า μ และ σ^2 คือค่าการกระจายของค่าเฉลี่ยและแปรปรวน ในสมการที่ (37) ตัวหารคือ n-1 ดังนั้นค่า $E[S^2]$ จึงมีค่าเท่ากับ $E[(X-\mu)^2]$ ค่า

รากที่สองของความแปรปรวนคือค่าเบี่ยงเบนมาตรฐาน (Standard deviation) และค่าอัตราส่วนระหว่างค่าเบี่ยงเบนมาตราฐานและค่าเฉลี่ยเรียกว่าค่าสัมประสิทธิ์ของความแปรปรวน(Coefficient of variation)

2.6.3.3 ช่วงความมั่นใจ[6]

การวิ่งระบบหลาย ๆครั้งเป็นการประเมินค่า Sample mean ที่ได้จากการวิ่งโปรแกรมโดยไม่จำกัดเวลาว่ามีค่าเข้าใกล้ Distribution mean , μ เพียงใด หรือทำนองเดียวกัน อาจกล่าวว่าต้องวิ่งระบบนานเพียงใดจึงจะได้ค่า Sample mean ที่เข้าใกล้ค่า μ

พิจารณาการประเมินค่าเฉลี่ยเวลารอในคิวของระบบ M/M/1 การเก็บค่าเวลาในคิวหรือค่าเฉลี่ยเวลารอในคิวสามารถนำไปคำนวณหาค่าเฉลี่ยความยาวคิวได้ โดยสมมุติให้ค่าเฉลี่ยลูกค้าเข้ามาเป็น Ta=125 ค่าเฉลี่ยเวลาให้บริการเป็น Ts=100 ให้ Xi แทนเวลาที่ลูกค้าคนที่ i รอในคิว และให้ X แทนค่าเฉลี่ยเวลารอในคิวที่ได้จากการวิ่งโปรแกรมเมื่อลูกค้าครบ 5,000 คน ต้องตรวจสอบว่า ค่า X เข้าใกล้ μ หรือไม่

ต่อไปนี้เป็นตัวอย่างจากการวิ่งโปรแกรมที่ตัวเลขสุ่มแตกต่างกัน ได้ค่าเฉลี่ยของเวลารอในคิว 10 ค่าดังนี้

ตารางที่ 2.1 แสดงตัวอย่างผลลัพธ์ที่ได้จากการวิ่งโปรแกรมทดสอบ

ผลครั้งที่ 1. 331.993	ผลครั้งที่ 6. 447.532
ผลครั้งที่ 2. 366.052	ผลครั้งที่ 7. 420.858
ผลครั้งที่ 3. 403.524	ผลครั้งที่ 8. 355.959
ผลครั้งที่ 4. 464.856	ผลครั้งที่ 9. 492.144
ผลครั้งที่ 5. 393.393	ผลครั้งที่ 10. 389.200

จะเห็นได้ว่าค่าเฉลี่ยอยู่ในช่วง 332 - 492 ซึ่งค่าเฉลี่ยของสิบตัวอย่างนี้คือ 406.554 ต้องตรวจสอบว่าค่านี้ใกล้เคียงกับค่า μ เพียงใด

สามารถตอบคำถามได้โดยวัดเรียกว่าช่วงความมั่นใจ โดยสมมุติว่ามีชุดข้อมูล N ชุด มีค่า Sample value คือ $Y_1, Y_2, ..., Y_N$ ด้วยค่าเฉลี่ยการกระจาย μ (ยังไม่ทราบค่า) ดังนั้นค่า Sample mean คือ

$$\bar{Y} = \sum_{i=1}^{N} Yi / N \qquad \text{----------(38)}$$

กำหนดให้ 1-α คือความน่าจะเป็นที่ค่าสัมบูรณ์ของผลต่างระหว่าง Sample mean และ μ มีค่าน้อยกว่าหรือเท่ากับ H

$$P[|\bar{Y}-\mu| \le H] = 1-\alpha$$

----------(39)

ดังนั้นช่วงความมั่นใจของค่าเฉลี่ย คือ

$$P[\bar{Y}-H \le \bar{Y}+H] = 1-\alpha$$

----------(40)

ค่า $\bar{Y}-H$ ถึง $\bar{Y}+H$ เรียกว่าช่วงความมั่นใจ ค่า H เรียกครึ่งช่วงของช่วงความมั่นใจ(half-width) และ 1-α เรียกว่าระดับความมั่นใจ (Confidence level) หรือค่าสัมประสิทธิ์ความมั่นใจ (Confidence Coefficient) โดยปรกติค่าอยู่ในระดับ 0.90-0.95 ค่าช่วงความมั่นใจกำหนดโดยผู้วิเคราะห์ระบบ ดังนั้นค่า H จึงขึ้นอยู่กับ Sample value, จำนวนตัวอย่าง, และค่าของ α

H เป็นฟังก์ชันของตัวแปรสุ่ม ดังนั้นจึงเป็นตัวแปรสุ่มด้วย เช่นเดียวกันช่วง $\bar{Y}\pm H$ จึงเป็นช่วงสุ่มด้วย การทดลองที่แตกต่างกันจะให้ค่าของ Y ต่างกัน ส่งผลให้ได้ช่วงความมั่นใจต่างกัน ถ้าทำการทดลองซ้ำหลายๆหนแล้วคำนวณค่าช่วงความมั่นในสำหรับแต่ละการทดลอง สัดส่วนของช่วงที่มีค่า μ ตกอยู่คือ 1-α ช่วงความมั่นใจที่มีค่าเฉลี่ยการกระจาย μ ตกอยู่ซึ่งกล่าวได้ว่าครอบคลุมค่าเฉลี่ยได้ และค่า 1-α เรียกว่าความน่าจะเป็นส่วนน้อย

เมื่อ $Y_1, Y_2, \ldots, Yn$ เป็นตัวแปรสุ่มที่เป็นอิสระต่อกันจากการกระจายปกติด้วยค่าเฉลี่ย μ หาค่า H ได้โดย

$$H = t_{\alpha/2}; N-1 s / N^{1/2}$$

----------(41)

โดย $t_{\alpha/2}; N-1$ คือช่วงบนควอไทล์ที่ $\alpha/2$ ของการกระจาย t ด้วยค่าองศาอิสระ $N-1$ และ S^2 คือค่าความแปรปรวน

$$S^2 = \sum_{i=1}^{N} (Yi-\bar{Y})^2 / (N-1)$$

----------(42)

S^2 เรียกว่าค่าความแปรปรวนของการกระจายของ Sample value Y_i ค่า S^2/N คือค่าแปรปรวนของการกระจายของ Sample mean , $\bar{Y}$ และ $S/N^{1/2}$ คือค่าเบี่ยงเบนมาตรฐานของการกระจายของ Sample mean หรือบางครั้งเรียกว่า Standard error ของค่าเฉลี่ย

ภายใต้ข้อสมมติของความเป็นอิสระและความเป็นปกติ ค่า Sample mean เป็นการกระจายแบบ t โดยค่าการกระจายแบบ t มาตรฐานตามนิยามมีค่าเฉลี่ยเท่ากับศูนย์ และค่าเบี่ยงเบนมาตรฐานเท่ากับหนึ่ง นั่นคือจำนวนเปอร์เซ็นต์ 100(α/2) ของ Sample value จากการกระจายนี้มีค่ามากกว่าค่าเบี่ยงเบนมาตรฐาน + $t_{\alpha/2}; N-1$ จากค่าเฉลี่ย และในเปอร์เซ็นต์ที่เท่ากันมากกว่าค่าเบี่ยงเบนมาตรฐาน - $t_{\alpha/2}; N-1$ จากค่าเฉลี่ย ดังนั้นเปอเซนต์ 100(α/2) จึงตกอยู่ภายในหน่วยค่าเบี่ยงเบนมาตร

ฐาน $\pm t_{\alpha/2}; N-1$ ของค่าเฉลี่ยตามสมการ (41) แล้วค่า H ซึ่งหาจากการคูณจำนวนหน่วยค่าเบี่ยงเบนมาตรฐานที่กำหนดโดย α และ N-1 โดยค่าเบี่ยงเบนมาตรฐานของค่าเฉลี่ย ค่า N-1 เรียกว่า Number of degree of freedom รูปร่างการกระจาย t และควอไทล์ที่สัมพันธ์กับค่าเบี่ยงเบนมาตรฐานที่กำหนดให้เปลี่ยนแปลงเมื่อ N-1 เปลี่ยนแปลง

ตารางที่ 2.2 แสดงค่าของ $t_{\alpha/2}; N-1$

	$\alpha/2$ =0.05	$\alpha/2$=0.025		$\alpha/2$ =0.05	$\alpha/2$=0.025
N-1	t	t	N-1	t	t
4	2.13	2.78	15	1.75	2.13
5	2.02	2.57	16	1.75	2.12
6	1.94	2.45	17	1.74	2.11
7	1.90	2.45	18	1.73	2.10
8	1.86	2.31	19	1.73	2.09
9	1.83	2.26	20	1.73	2.09
10	1.81	2.23	21	1.72	2.08
11	1.80	2.20	22	1.72	2.07
12	1.78	2.18	23	1.71	2.07
13	1.77	2.16	24	1.71	2.06
14	1.76	2.15	α	1.65	1.96

จากค่าความมั่นใจ 95 เปอร์เซ็นต์ สำหรับค่าเฉลี่ยเวลารอในคิวระบบ M/M/1 ในตัวอย่างข้างต้น คำนวณค่าต่าง ๆ ได้ดังนี้

$1-\alpha = 0.95$, ดังนั้น α =0.05

Sample mean $\bar{y}$ =406.55

Sample variance S^2 = 2540.08

Standard deviation of the sample mean $s/N^{1/2}$ = 15.94

t.05/2;9 = 2.26 (จากตารางที่ 2.2)

Confidence interval half-width H = 2.26x15.94 = 36.02

Confidence interval = [406.55+36.02, 406.55-36.02]

ดังนั้นจะเห็นว่าที่ช่วงความมั่นใจ 95 เปอร์เซ็นต์ ค่าเฉลี่ยเวลารอในคิวอยู่ระหว่าง 370.53 และ 442.47 (ตามทฤษฎีค่าเฉลี่ยคือ 400) ถ้าเพิ่มจำนวนในการวิ่งโปรแกรมหลายๆครั้งและหาค่าช่วง

ความมั่นใจของแต่ละครั้ง กล่าวได้ว่า 95 เปอร์เซ็นต์ของช่วงความมั่นใจเหล่านี้มีค่า True mean ตกอยู่ ซึ่งเห็นได้ว่ามีค่าเฉลี่ย 4 ค่าในจำนวนทั้งหมด 10 ค่าตกอยู่ในช่วงความมั่นใจนี้

บทนี้กล่าวถึงทฤษฎีคิวและศึกษาโครงสร้างพื้นฐานของคิวพร้อมทั้งโพรเซสพัวซองที่ศึกษาถึงการกระจายการมาถึงและการกระจายการบริการซึ่งเป็นองค์ประกอบสำคัญของระบบคิว นอกจากนี้ได้นำเสนออัลกอริธึมสำหรับการวัดประสิทธิภาพของระบบ พร้อมทั้งตัวอย่างในการวิเคราะห์ข้อมูลจากโปรแกรมจำลองแบบชุดนี้ บทที่ 3 จะกล่าวถึงรายละเอียดในการพัฒนาโมเดลและโปรแกรมตามวิธีสนใจวัตถุ

บทที่ 3

การออกแบบโปรแกรมจำลองแบบ

3.1 การพัฒนาโมเดลและการทดสอบ [6]

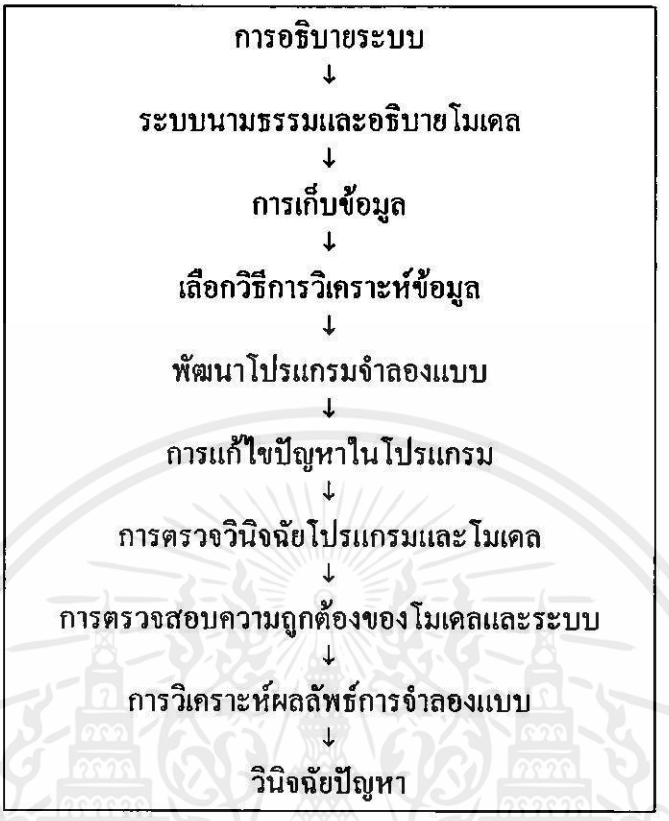
3.1.1 กระบวนการวิเคราะห์และสร้างโมเดล

การพัฒนาโมเดลโดยการใช้โมเดลการจำลองแบบ หลักสำคัญคือการออบแบบระบบนามธรรม(Abstracting system design) แล้วเชื่อมโยงเข้าสู่การออกแบบโมเดล(Model design) การออกแบบมักเริ่มจากระบบที่ง่าย ๆ การศึกษาข้อมูลจากระบบที่มีอยู่แล้วก็มีส่วนช่วยให้การออกแบบดีขึ้นซึ่งได้ศึกษาข้อมูลจริงเป็นตัวอย่าง เช่นลักษณะภาระระบบ(workload) และมีการตรวจสอบความถูกต้องของโมเดลได้ด้วย รูปที่ 3.1 แสดงถึงกระบวนการออกแบบและวิเคราะห์โมเดล กระบวนการแบ่งออกได้ 3 เฟส คือ การพัฒนา การทดสอบ และ การวิเคราะห์ข้อมูล

ขั้นแรกในการพัฒนาคือการอธิบายการทำงานของระบบจากมุมมองด้านประสิทธิภาพ ข้อมูลต้องเชื่อมโยงกับจุดประสงค์ในการวิเคราะห์ด้วย กำหนดสิ่งที่นำเสนอในโมเดล การเชื่อมต่อ คุณสมบัติต่าง ๆและมีการเก็บข้อมูลอย่างละเอียด นอกจากนี้มีการเลือกวิธีในการวิเคราะห์ที่เหมาะสมด้วย

ขั้นการทดสอบประกอบด้วย 3 ขั้นตอนย่อยคือ การแก้ไขปัญหา การตรวจวินิจฉัยและการตรวจสอบความถูกต้องของข้อมูล

การวิเคราะห์ข้อมูล กล่าวถึงวิธีการวัดค่าตัวแปรที่สนใจในระบบ การประเมินค่าและการควบคุมความถูกต้องของผลลัพธ์การจำลองแบบ



รูปที่ 3.1 แสดงการวิเคราะห์และสร้างโมเดล

3.2 การพัฒนาโปรแกรมจำลองแบบ

สิ่งที่ต้องพิจารณามีดังนี้

- การออกแบบโมเดลจำลองแบบ
- การออกแบบโปรแกรม
- การจัดการตัวแปร
- การแก้ไขปัญหาโปรแกรม
- การวัดค่าตัวแปร

3.2.1 การออกแบบโมเดลจำลองแบบ

เป็นขั้นตอนการแปลงโมเดลรายละเอียดไปสู่การออกแบบโมเดลจำลองแบบ ต้องกำหนดภาษาที่สอดคล้องกับความซับซ้อนของโมเดล เช่นกรณีโมเดลไม่ซับซ้อน มักเลือกภาษาที่สนับสนุน Event-oriented ส่วนโมเดลที่ซับซ้อนภาษาที่ใช้จะเป็น Process-oriented นอกจากนี้ต้องออกแบบการเชื่อมต่อโมดูล การบำรุงรักษา ข้อมูลของโมเดล ซึ่งในขั้นแรกควรทดลองทำการจำลองแบบด้วยมือก่อนเช่นการนำสเปรดชีตมาช่วยจำลองแบบ

3.2.2 การจัดการโปรแกรม

ขั้นนี้เป็นการนำโมเดลที่ออกแบบไปสู่การออกแบบโปรแกรม เช่นการออกแบบคลาส อ็อปเจ็ค ฟังก์ชัน โปรซีเดอร์ รูทีนต่างๆ

3.2.3 การจัดการตัวแปร

ในโปรแกรมที่ซับซ้อน ต้องจัดระเบียบตัวแปรให้อยู่ในเส้นทางที่ต้องการและง่ายต่อการควบคุม เพื่อประโยชน์ในการใช้งานและการตรวจสอบข้อผิดพลาดในโปรแกรม

3.2.4 การแก้ข้อผิดพลาดในโปรแกรม

เป็นขั้นการตรวจหาข้อผิดพลาดของในโปรแกรมเพื่อให้ระบบวิ่งได้ตามที่ต้องการ ซึ่งมีกระบวนการหลายรูปแบบ เช่น การวินิจฉัยข้อผิดพลาด, การวิ่ง Trace, โปรแกรม Dump และดูรายงานจากโปรแกรม

3.2.5 การตรวจวินิจฉัย

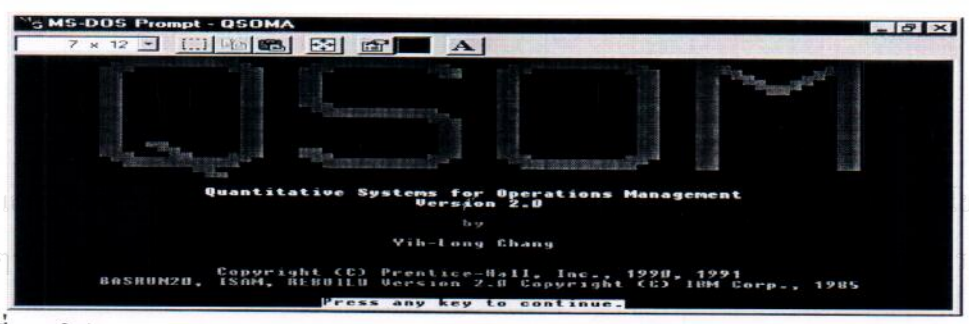
โมเดลต้องผ่านการตรวจสอบความถูกต้อง โดยทั่วไปมักมีการทดลองเปรียบเทียบกับระบบเก่าหรือระบบที่ทำงานคล้ายๆที่มีอยู่แล้ว มีการกำหนดตัวแปรทดสอบหลายๆ แบบ การออกแบบข้อสมมติเบื้องต้นของการกระจายที่เหมือนกัน กำหนดระดับความละเอียดของการทดสอบ กฎของคิวที่ใช้ นอกจากนี้แล้วยังนิยมเปรียบเทียบกับคำตอบที่ได้จากการวิเคราะห์แบบเดิม(Close-form)[2]

3.2.6 การตรวจสอบความถูกต้อง

เพื่อตรวจสอบว่าระบบจำลองสามารถแสดงการทำงานแทนระบบจริงได้สมเหตุสมผลหรือไม่ ต้องประเมินตัวแปรต่างๆในกรอบที่กำหนดเพื่อดูว่าระบบทำงานได้อย่างถูกต้อง โดยทั่วไปแล้วโมเดลได้ออกแบบวิธีวิเคราะห์ผลลัพธ์ด้วยแล้ว

3.3 โปรแกรมที่นำมาศึกษาเป็นตัวอย่าง โปรแกรมที่ 1 [1]

โปรแกรมนี้คือ Quantitative System for Operations Management Version 2.0 หรือ QSOM [2,3] ซึ่งสร้างออกมาใช้สำหรับงาน Operations Management โดยเฉพาะ



รูปที่ 3.2 โปรแกรม QSOM Version 2.0

3.3.1 ความสามารถของโปรแกรม

- ประกอบด้วย 18 โปรแกรมย่อยที่ช่วยการตัดสินใจในการจัดการการดำเนินงานโครงการสนับสนุนการตัดสินใจด้วยคอมพิวเตอร์
- การเชื่อมต่อระหว่างทฤษฎีและการประยุกต์ใช้อยู่ในเกณฑ์ดี มีการอธิบายทฤษฎีในรูปสมการคณิตศาสตร์และตัวแปรของสมการอย่างละเอียด
- เป็นแบบฝึกหัดที่นำเข้าสู่การความเข้าใจเทคนิคการจัดการดำเนินงาน ในภาวะแวดล้อมที่กำหนด นำไปประกอบการทำงานได้จริงในหลายรูปแบบ เช่นการวางแผนการผลิต บริหารโครงงาน เป็นต้น
- วิ่งได้บนเครื่องพีซีรุ่นซีพียู 80386 เป็นอย่างต่ำ ภายใต้ระบบปฏิบัติการดอส(DOS Operating System)

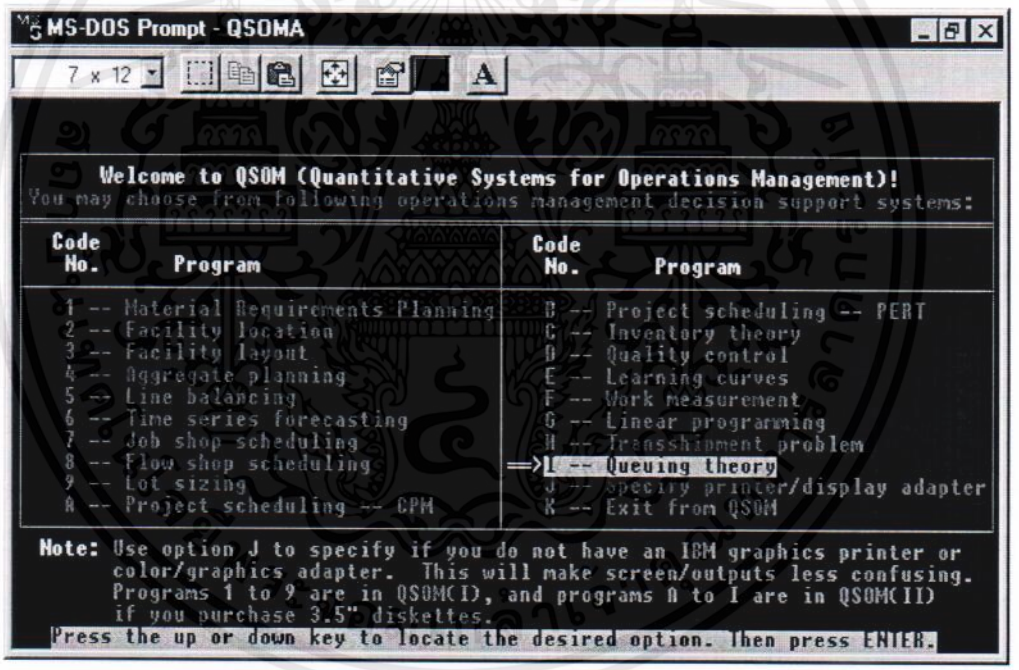
3.3.2 ฟังก์ชันต่าง ๆใน QSOM

- การวางแผนการใช้วัสดุ - Material requirements planing(MRP)
- ระบบสาธารณูปโภค - Facility location
- การวางแผนการผลิตสาธารณูปโภค - Facility production planning
- การวางแผนการผลิตแบบรวม - Aggregate production planning
- การปรับสมดุลสายการผลิต - Production line balancing
- การคาดการณ์ด้านเวลา - Time series forecasting
- การสับหลีกตามงาน - Job shop scheduling
- การสับหลีกตามการไหลของงาน - Flow shop scheduling
- การจัดการขนาดที่ไม่สามารถคาดการณ์ได้ - Uncapacitated lot sizing
- วิธีเลือกเส้นทางวิกฤติ - Critical path method(CPM)
- การประเมินผลโครงการ - Program evaluation and review technique(PERT)
- ทฤษฎีพัสดุคงคลัง - Inventory theory
- การควบคุมคุณภาพ - Quality control
- กราฟเรียนรู้ - Learning curve
- การวัดผลงาน - Work measurement
- การโปรแกรมแบบเส้นตรง - Linear programming
- ปัญหาการขนส่งและการส่งสินค้าออก - Transportation and transshipment problems
- ทฤษฎีคิว - Queuing theory

3.3.3 ข้อดีของโปรแกรมที่นำมาศึกษา

1. ส่วนการติดต่อผู้ใช้ (User Interface) มีเฉพาะเมนูแบบ Pull Down ทำงานภายใต้โหมดตัวอักษร(Text Mode)
2. การรายงานผลเป็นแบบตารางสรุปค่าและรายงานได้ภายหลังจากที่ระบบวิ่งเสร็จแล้วเท่านั้น โดยแสดงผลในประเด็นที่คลอบคลุมใจความสำคัญของเรื่องนั้น ๆ
3. รายงานผลได้ไม่เกิน 200 รายการ
4. รับข้อมูลจากอินพุทจากคีย์บอร์ด จากไฟล์ และสามารถเก็บข้อมูลลงไฟล์ได้
5. จำนวนอินพุทของระบบจัดกัดเพียง 5,000 ค่า

หัวข้อโดยส่วนใหญ่เป็นการประยุกต์ใช้คิวกับงานเฉพาะด้าน ในหัวข้อสุดท้ายที่อธิบายทฤษฎีคิว มีการปรับพื้นฐานทฤษฎีคิวพอสมควรและให้รายละเอียดขั้นตอนในการจำลองแบบ ผู้ใช้ต้องมีความรู้เรื่องคณิตศาสตร์คิวมาก่อนจึงจะเข้าใจได้ง่าย เนื่องจากไม่มีการกำหนดมุมมองเป็นกราฟฟิค



รูปที่ 3.3 แสดงเมนูหลักของโปรแกรม

3.3.4 ตัวอย่างศึกษาจากโปรแกรมเรื่อง Queueing Theory

ตัวอย่างระบบที่โปรแกรมสนับสนุนมีดังนี้

M/M/1, M/G/1, D/M/1, M/M/S, M/M/s/N, M/M/s//K, M/M/s,N/K, M/G/∞, M(b)/M/1, และ M/M/s/s

โครงสร้างเมนู

Overview of the QUEUE decision support system

Enter new problem

Read existing problem form disk(ette)

Show input data

- Display the input data
- Print the input data
- Save the input data in an ASCII file
- Return to the function menu

Solve problem

Save problem on disk(ette)

Modify problem

Show final solution

- Display the final solution
- Print the final solution
- Save the final solution in an ASCII file
- Return to the function menu

Return to the program menu

Exit form QSOM

รูปที่ 3.4 แสดงรายการเมนูย่อยระบบคิว

ตัวอย่างการแก้ปัญหาระบบคิวตามวิธีโปรแกรม QSOM

ขั้นตอน :

1. วิเคราะห์ปัญหา และกำหนดตัวแปรใช้งาน
2. ใส่ตัวแปรเข้าไปในระบบ หรืออ่านข้อมูลเก่าได้จากแผ่นดิสก์ที่เคยเก็บไว้
3. ดูข้อมูลเก่าเพื่อตรวจสอบข้อมูล
4. แก้ไขตัวแปรให้ถูกต้อง
5. วิ่งระบบ และแสดงผลลัพธ์
6. เลือกแสดงผลทางหน้าจอ เครื่องพิมพ์ หรือเก็บในดิสก์

3.3.5 ตัวอย่างระบบคิว M/M/1

บริษัทขนส่งแห่งหนึ่งมีทีมงานขนส่งสินค้า ซึ่งให้บริการส่งของด้วยการกระจายแบบเอ็กซ์โปเนนเชียลด้วยค่าเฉลี่ย 20 นาทีต่อคัน รถขนส่งออกจากสถานีด้วยค่าการกระจายพัวซองชั่ว

โมงละ 2 คัน ผู้จัดการต้องการประเมินผลการทำงานของทีมขนส่ง โดยทีมขนส่งแบ่งออกเป็น 2 ทีมเท่ากันและให้บริการการด้วยค่าการกระจายแบบเอ็กซ์โปเนนเชียลด้วยค่าเฉลี่ย 40 นาที ให้หาผลกระทบของการเปลี่ยนแปลงเงื่อนไขนี้

รูปที่ 3.5 แสดงชื่อของระบบคิว

ขั้นที่ 1. วิเคราะห์ปัญหาและกำหนดตัวแปร เช่น อัตราการให้บริการ อัตราการมาถึง จำนวนผู้ให้บริการ ระบบนี้เป็นชนิด M/M/1

อัตราการมาถึง(λ) = 2 ต่อชั่วโมง

อัตราการบริการ(μ) = 3 ต่อชั่วโมง

จำนวนผู้ให้บริการ (S) = 1

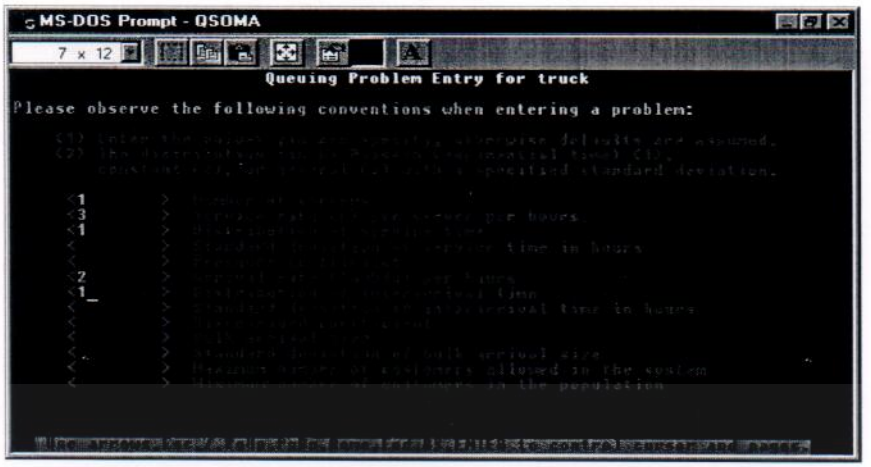
ขั้นที่ 2. ใส่ตัวแปรในโปรแกรม โดยทำผ่านเมนู และแสดงค่าที่ป้อนเข้าไปออกมาดู

ขั้นที่ 3. เลือกแสดงข้อมูลที่ป้อนเข้าไปในระบบ อาจเลือกพิมพ์หรือเก็บในแผ่นดิสก์

ขั้นที่ 4. สามารถเข้าไปแก้ไขข้อมูลได้ก่อนวิ่งโปรแกรม

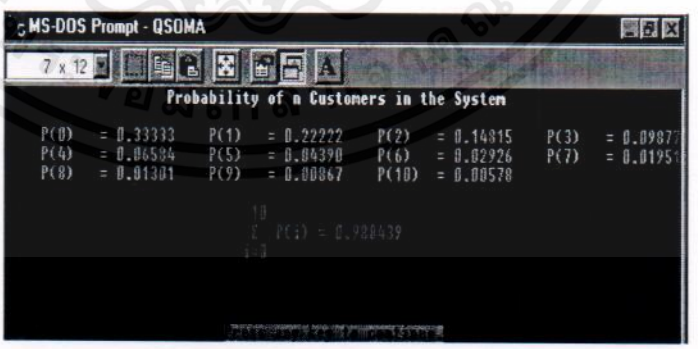
ขั้นที่ 5. วิ่งโปรแกรม ผลลัพธ์แสดงให้เห็นภายหลังจากการวิ่งโปรแกรมเสร็จ

ขั้นที่ 6. หลังจากวิ่งโปรแกรมเรียบร้อยแล้ว สามารถเลือกแสดงผลที่หน้าจอ ส่งพิมพ์ หรือเก็บในแผ่นดิสก์

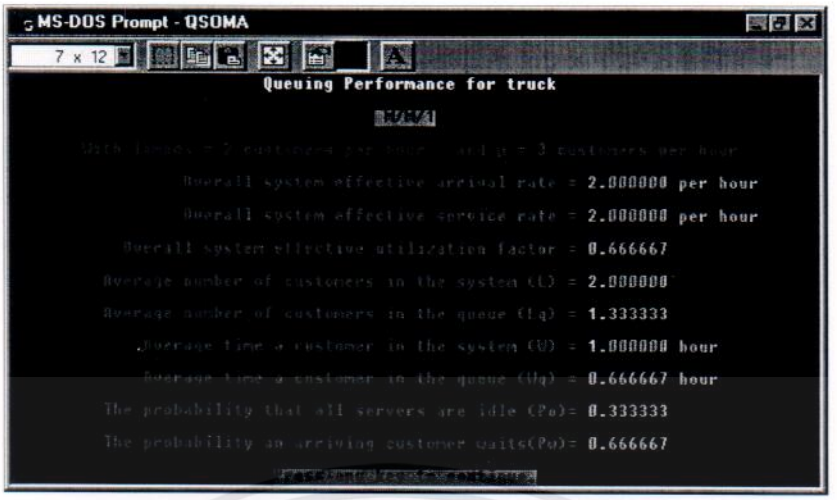


รูปที่ 3.6 การนำข้อมูลเข้าระบบ

รูปที่ 3.7 แสดงเมนูแสดงข้อมูลอินพุท



รูปที่ 3.8 แสดงจำนวนความน่าจะเป็นสำหรับรถบรรทุก



รูปที่ 3.9 แสดงประสิทธิภาพการวัดระบบ

รูปที่ 3.10 แสดงความน่าจะเป็นสำหรับรถบรรทุก

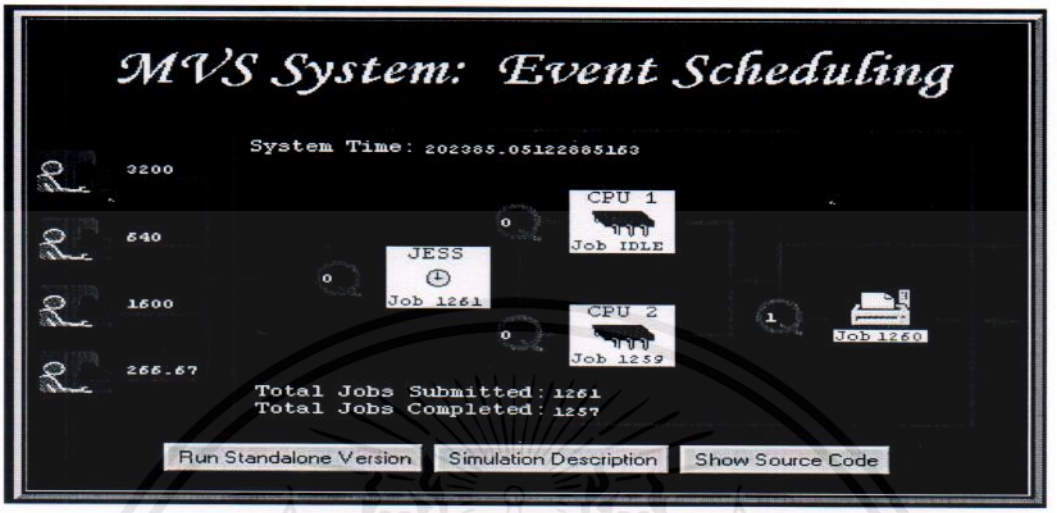
3.3.6 สรุปผลการศึกษาวิเคราะห์และเปรียบเทียบ

ตารางที่ 3.1 แสดงการเปรียบเทียบโปรแกรมตัวอย่าง กับ โปแกรมที่พัฒนา

	QSOM	Queuing Simulation Program
ส่วนเชื่อมต่อผู้ใช้	DOS, ไม่สนับสนุนการใช้เมาส์	กราฟฟิกบนวินโดว์, สนับสนุนการใช้เมาส์
เครื่องพิมพ์	สามารถพิมพ์รายงานได้	พิมพ์รายงานโดยฟังก์ชันวินโดว์
การรายงานผลโดยกราฟ	สนับสนุนการรายงานผลด้วยกราฟ	สนับสนุนการรายงานผลด้วยกราฟ
ฟังก์ชัน	มีครอบคลุมหลายแขนง	มีเฉพาะการจำลองแบบคิว
การหาคำตอบ	จำลองแบบ	จำลองแบบ
วิเคราะห์ผล	ใช้วิธีพื้นฐาน	ใช้วิธีพื้นฐานและช่วงความมั่นใจ
โครงข่ายคิว	ไม่สนับสนุน	จำลองแบบโครงข่ายคิวได้
M/M/1	λ= 2 ,μ=3 U=0.6666 L=2.0 Lq=1.3333 W=1.0000 Wq=0.6666 (ไม่ทราบจำนวนหน่วยเวลาในการวิ่งโปรแกรม)	λ= 2 ,μ=3 U=0.71 L=7.48 Lq=6.77 W=26.2456 Wq=23.7543 (วิ่งโปรแกรม 200 หน่วยเวลา)

3.4 โปแกรมที่ศึกษาเป็นตัวอย่าง โปรแกรมที่ 2

โปรแกรมจำลองแบบการทำงานระบบปฏิบัติการ MVS



รูปที่ 3.11 แสดงระบบจำลองแบบ MVS

3.4.1 อินพุทสำหรับโปแกรม

ตัวอย่างอินพุทของระบบ MVS

Number of Replications: 6 **(Must be > 0)**

Number of Jobs per Replication: 5000 **(Must be > 0)**

Steady State Start Time: 500 **(Must be > 0**

Produce Confidence Intervals? ○ **Yes or** ○ **No**

(To run Confidence Intervals, Number of Replications must be >= 3)

Submit Simulation Arguments submit

Valid Entries for Number of Replications, Number of Jobs per Replication and Steady State Start Time include:

- **Numeric Characters ONLY**
- **Do not enter Alpha Characters**
- **Do not enter Commas, Decimal Points, or any other extraneous Character**

รูปที่ 3.12 แสดงอินพุทของโปรแกรม

3.4.2 ระบบปฏิบัติการ MVS

จำลองแบบระบบปฏิบัติการ MVS วิ่งงาน 18,000 งานและเริ่มเก็บข้อมูลสถิติเมื่อได้งาน 3,000 งานจากการประมวลผล กำหนดให้ทำงานในรูปแบบจาวาแอบเพล็ต

คำอธิบายการจำลองแบบ

ระบบ MVS ทำงานด้วยซีพียู 2 ตัว ผู้ใช้ส่งงานแบช(Batch programs) เข้าไปประมวลผลใน MVS โดยคำสั่ง { submit} ของระบบ TSO (Time Sharing Option) ในรูปที่ 3. 10 แสดงผู้ใช้ 4 กลุ่ม ดังนี้

- ผู้ใช้หมุนโมเด็มเข้ามาใช้งาน ด้วยความเร็ว 300 บิทต่อวินาที
- ผู้ใช้หมุนโมเด็มเข้ามาใช้งาน ด้วยความเร็ว 1,200 บิทต่อวินาที
- ผู้ใช้หมุนโมเด็มเข้ามาใช้งาน ด้วยความเร็ว 2,400 บิทต่อวินาที
- ผู้ใช้เข้าสู่ระบบผ่านเครือข่าย LAN ด้วยความเร็ว 9,600 บิทต่อวินาที

ผู้ใช้แต่ละคนเขียนโปรแกรมแล้วส่งเข้ามาวิ่งในระบบ MVS จากการเก็บรวบรวมข้อมูล สมมติว่าผู้ใช้ส่งงานเข้ามาในระบบด้วยการกระจายเอ็กซ์โปเนนเชียลดังตาราง

ตารางที่ 3.2 แสดงข้อมูลที่เก็บรวบรวม

ชนิดผู้ใช้	เวลาเข้ามาถึง	ค่าเฉลี่ย
โมเด็มชนิด 300	เอ็กซ์โปเนนเชียล	3200 วินาที
โมเด็มชนิด 1200	เอ็กซ์โปเนนเชียล	640 วินาที
โมเด็มชนิด 2400	เอ็กซ์โปเนนเชียล	1600 วินาที
โมเด็มชนิด 9600	เอ็กซ์โปเนนเชียล	266.67 วินาที

อันดับโปรแกรมแบชต้องวิ่งเข้าสู่ระบบจัดการงานเรียก JES (job entry subsystem) ของ MVS ตัวจัดระบบงานของ JES คือ JES scheduler (JESS) เป็นตัวจ่ายงานเข้าไปให้กับซีพียูที่ 1 ด้วยความน่าจะเป็นเท่ากับ 0.6 หรือซีพียูที่สองด้วยความน่าจะเป็นเท่ากับ 0.4 ขณะที่โปรแกรมประมวลผลเสร็จ จะถูกส่งไปยังตัวจัดการเอาท์พุทด้วยความน่าจะเป็น 0.2 หรือเครื่องพิมพ์(Printer ,PRT) ด้วยความน่าจะเป็นเท่ากับ 0.8 สมมติว่าทุกคิวในระบบมีกฎแบบมาก่อนได้รับบริการก่อน ความน่าจะเป็นของการกระจายของเวลาประมวลผลสำหรับโปรแกรมในแต่ละหน่วยบริการดังตารางข้างล่าง

ตารางที่ 3.3 แสดงการกระจายเวลาประมวลผล

หน่วยบริการ	เวลาประมวลผล	ค่าเฉลี่ย
Jess	เอ็กซ์โปเนนเชียล	112 วินาที
CPU1	เอ็กซ์โปเนนเชียล	226.67 วินาที
CPU2	เอ็กซ์โปเนนเชียล	300 วินาที
PRT	เอ็กซ์โปเนนเชียล	160 วินาที

สมมติว่าโมเดลจำลองแบบถึงภาวะเสถียรหลังจากที่วิ่งโปรแกรมผ่านไปแล้ว3,000 โปรแกรม และที่ 15,000 โปรแกรม และสร้างช่วงความเชื่อมั่นสำหรับการวัดประสิทธิภาพ

ตารางที่ 3.4 แสดงผลลัพธ์

อัตราการใช้ประโยชน์ของ Jess	0.70
อัตราการใช้ประโยชน์ของ CPU1	0.85
อัตราการใช้ประโยชน์ของ CPU2	0.75
อัตราการใช้ประโยชน์ของ PRT	0.80
เวลาที่ใช้โดยโปรแกรมแบชในระบบ MVS	2400 วินาที
จำนวนเฉลี่ยของโปรแกรมแบชใน MVS	15

3.4.3 ข้อมูลสถิติจากโปรแกรม

ตารางที่ 3.5 แสดงสถิติจากระบบ

JESS	CPU1	CPU2	PRT	J TIME	J NUM
0.7321	0.8604	0.7734	0.7807	2113.0206	15.0688
0.7067	0.8170	0.7049	0.8008	1881.4151	12.9941
0.6842	0.8475	0.7244	0.7934	1994.4875	13.6745
0.7027	0.8169	0.7192	0.7834	1709.2545	11.8771
0.6902	0.8403	0.7319	0.8119	1891.0292	13.1421
0.6988	0.8740	0.7075	0.8227	2166.9909	15.2465

คำอธิบายแต่ละหลักของตาราง:

JESS: เปอร์เซ็นต์จำนวนเวลาที่ JESS ถูกใช้งาน

CPU1: เปอร์เซ็นต์จำนวนเวลาที่ CPU1 ถูกใช้งาน

CPU2: เปอร์เซ็นต์จำนวนเวลาที่ CPU2 ถูกใช้งาน

PRT: เปอร์เซ็นต์จำนวนเวลาที่ Printer ถูกใช้งาน

J TIME: เวลาเฉลี่ยที่งานใช้ไปในระบบ MVS

J NUM: จำนวนเฉลี่ยของงานในระบบ MVS ต่อหน่วยเวลา

อัตราการใช้ประโยชน์ของ JESS :

ค่าที่กำหนดให้ : 0.732 0.707 0.684 0.703 0.690 0.699

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนของจุดข้อมูล = 6

Sample Mean = 0.702525

Sample Variance = 0.000279

ตารางที่ 3.6 แสดงช่วงความมั่นใจ (JESS)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	0.692	0.689	0.685	0.680	0.675
Upper Limit	0.713	0.716	0.720	0.725	0.730

อัตราการใช้ประโยชน์ของ CPU1 :

ค่าที่กำหนดให้ : 0.860 0.817 0.848 0.817 0.840 0.874

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนจุดของข้อมูล = 6

Sample Mean = 0.842726

Sample Variance = 0.000530

ตารางที่ 3.7 แสดงช่วงความมั่นใจ (CPU1)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	0.829	0.824	0.819	0.811	0.805
Upper Limit	0.857	0.862	0.867	0.874	0.881

อัตราการใช้ประโยชน์ของ CPU2 :

ค่าที่กำหนดให้ : 0.773 0.705 0.724 0.719 0.732 0.708

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนของจุดข้อมูล = 6

Sample Mean = 0.726929

Sample Variance = 0.000623

ตารางที่ 3.8 ช่วงความมั่นใจ (CPU2)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	0.712	0.706	0.701	0.693	0.686
Upper Limit	0.742	0.747	0.753	0.761	0.768

อัตราการใช้ประโยชน์ของ PRT :

ค่าที่กำหนดให้ : 0.781 0.801 0.793 0.783 0.812 0.823

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนของจุดข้อมูล = 6

Sample Mean = 0.798873

Sample Variance = 0.000267

ตารางที่ 3.9 ช่วงความมั่นใจ (PRT)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	0.789	0.785	0.782	0.776	0.772
Upper Limit	0.809	0.812	0.816	0.821	0.826

เวลาเฉลี่ยในระบบ :

ค่าที่กำหนดให้ : 2113.021 1881.415 1994.488 1709.255 1891.029 2166.991

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนของจุดข้อมูล = 6

Sample Mean = 1959.366333

Sample Variance = 28250.496094

ตารางที่ 3.10 แสดงช่วงความมั่นใจ(เวลาเฉลี่ย)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	1858.086	1821.101	1782.950	1728.467	1682.699
Upper Limit	2060.646	2097.631	2135.783	2190.266	2236.034

จำนวนเฉลี่ยในระบบ :

ค่าที่กำหนดให้ : 15.069 12.994 13.675 11.877 13.142 15.247

ช่วงความมั่นใจสำหรับข้อมูล

จำนวนของจุดข้อมูล = 6

Sample Mean = 13.667252

Sample Variance = 1.678438

ตารางที่ 3.11 แสดงช่วงความมั่นใจ(จำนวนเฉลี่ย)

Level	0.100	0.050	0.025	0.010	0.005
Lower Limit	12.887	12.602	12.307	11.887	11.535
Upper Limit	14.448	14.733	15.027	15.447	15.800

3.4.3 สรุป

การจำลองระบบ MVS เป็นตัวอย่างที่สำหรับเรียนรู้การทำงานระบบคิว มีการวิเคราะห์ข้อมูลสถิติที่ชัดเจน โปรแกรมนี้ยังไม่ยืดหยุ่นต่อการเปลี่ยนแปลงค่าตัวแปรต่าง ๆของระบบ ผู้ใช้ไม่สามารถออกแบบระบบกราฟฟิกใหม่ได้

3.5 การพัฒนาโปรแกรมจำลองแบบ

การจำลองแบบเป็นวิธีที่นิยมนำมาใช้หาคำตอบให้กับระบบต่าง ๆ ในปัจจุบัน โดยการเขียนโปรแกรมให้เป็นแบบจำลองของปัญหานั้น ๆ แล้วสังเกตพฤติกรรมของโปรแกรม ข้อได้เปรียบของการจำลองคือความเป็นระบบที่ชัดเจนและเข้าใจง่าย ปัญหาสำหรับการจำลองแบบมีสามประเด็นที่สำคัญคือ ค่าใช้จ่ายในการพัฒนาโปรแกรม ค่าใช้จ่ายในการวิ่งโปรแกรม และการวิเคราะห์ค่าสถิติของพฤติกรรมของโปรแกรม

3.5.1 ข้อกำหนดของโปรแกรมที่พัฒนา

- สามารถจำลองแบบการทำงานของคิวและโครงข่ายคิวได้
- ไม่จำกัดจำนวนอ็อปเจคในโปรแกรม
- โปรแกรมวิ่งบนระบบวินโดว์ สนับสนุนการใช้เมาส์และมัลติมีเดีย
- สามารถแสดงผลผ่านทางโปรแกรมเวบเบราเซอร์ได้
- ผู้ใช้สามารถวาดโครงข่ายคิวได้อย่างอิสระ มีฟังก์ชันการแก้ไขที่สะดวก
- มีฟังก์ชันการเก็บข้อมูลสถิติเพื่อออกรายงานประสิทธิภาพการทำงานระบบ
- สามารถแก้ไขตัวแปรสำหรับการวิ่งโปรแกรมได้
- สามารถเก็บข้อมูลในการจำลองแบบได้
- สามารถตั้งจุดจบการทำงานของโปรแกรมได้
- มีฟังก์ชัน Tracing เพื่อดูการทำงานของระบบในแต่ละเวลา
- สามารถออกรายงานผลด้วยกราฟได้
- มีรายงานสรุปผลการวิ่งโปแกรม เช่น Throughput , Utilization เป็นต้น และส่วนการวิเคราะห์ข้อมูลสถิติ
- โปรแกรมมีขนาดเล็ก

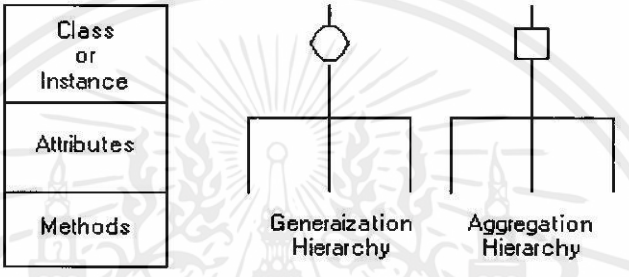
3.5.2 เทอมที่กำหนดใช้อธิบายระบบ

1. สถานะ(States) : แทนคุณลักษณะของระบบ เช่นจำนวนลูกค้าที่กำลังรอบริการจากผู้ให้บริการ
2. เหตุการณ์(Events) : เหตุการณ์ที่เกิดขึ้นในแต่ละจุดของเวลา ซึ่งอาจเปลี่ยนสถานะของระบบ เช่น การมาถึงของลูกค้า
3. วัตถุที่มีชื่อ(Entities) : วัตถุที่กำหนดชื่อได้ที่ส่งเข้าไปในระบบเช่นลูกค้าที่เข้าไปในธนาคาร Event (ตัวอย่างเช่น การมาถึงของลูกค้า)มีความสัมพันธ์กันกับวัตถุ(เช่นลูกค้า)
4. คิว(Queues) : วัตถุที่กำลังรอบริการบางอย่าง ซึ่งอาจเป็นทั้งคิวทางกายภาพและทางโปรแกรม
5. สับหลีก(Schedules) : กฎระเบียบที่นำมาใช้ควบคุมคิว

6. ตัวแปรสุ่ม(Random variates) : ตัวแปรสุ่มเพื่อสร้างอินพุทให้กับระบบ
7. การกระจาย(Distribution) : คุณลักษณะของตัวแปรสุ่มเชิงความน่าจะเป็น

3.5.3 การออกแบบโมเดลโดยแนวคิดแบบวัตถุ(Object Oriented Model Design)

การใช้แนวคิดแบบวัตถุในการพัฒนาโมเดล ต้องเริ่มจากการพิจารณาองค์ประกอบพื้นฐานของระบบ รวบรวมประเด็นสำคัญแล้วกำหนดวิธีการนำวัตถุในโลกความจริงไปสู่การโปรแกรม ซึ่งหลักการที่มีส่วนช่วยมากคือการนำโค้ดไปใช้ได้ใหม่(Code reuse) และการถ่ายทอดคุณสมบัติ(Inheritance) สำหรับการอธิบายแนวคิดในการออกแบบ กำหนดใช้สัญลักษณ์แทนลำดับชั้นของคลาส(Class) ดังนี้



รูปที่ 3.13 แสดงลำดับชั้นคลาส

ลำดับชั้นคลาส ประกอบด้วยคลาสเพียงอย่างเดียวเท่านั้น คลาสต่างๆถูกเชื่อมโยงกันในรูปแบบโครงสร้างลำดับชั้น อาจเป็นลำดับชั้นคลาสเดียว(Generalization) หรือลำดับชั้นของกลุ่มคลาส(Aggregation) อ็อปเจคเป็นสิ่งของที่กำหนดไว้ในคลาส ซึ่งอ็อปเจคแต่ละตัวจะมีคุณลักษณะ(Attribute)และวิธีทำงาน(Method) คุณลักษณะมักอ้างถึงคุณสมบัติของอ็อปเจค ขณะที่วิธีการทำงานจะอ้างถึงฟังก์ชันและชุดคำสั่งย่อย(Procedure)ที่ปฏิบัติกับลักษณะเหล่านั้น คุณลักษณะแบ่งออกเป็น 2 ชนิดคือแบบคงที่(static) และแบบเปลี่ยนได้(Dynamic) ลักษณะคงที่คือตัวแปรไม่มีการเปลี่ยนแปลงภายใต้กรอบเวลา ส่วนแบบเปลี่ยนได้ หมายถึงองค์ประกอบและตัวแปรสามารถเปลี่ยนค่าได้ การจำลองแบบส่วนใหญ่มีคุณลักษณะแบบเปลี่ยนค่าได้ จำเป็นต้องออกแบบฐานข้อมูลอ็อปเจคมารองรับ ซึ่งจะเก็บข้อสนเทศเกี่ยวกับระบบ การอธิบาย การวินิจฉัยและวิธีการเรียกดูข้อมูล ส่วนโมเดลถูกสร้างจากอ็อปเจคและเชื่อมต่อกันด้วยการส่งเมสเซส(Message) ระหว่างอ็อปเจค

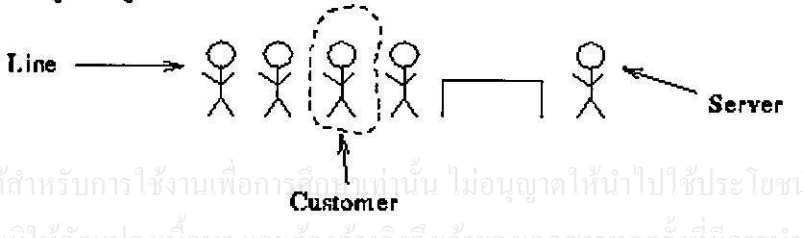
เมื่อกล่าวถึงอ็อปเจค ต้องกำหนดชื่อ เช่น engine1 สามารถระบุคลาสเพื่อจะทราบว่าต้องเชื่อมต่ออย่างไรเช่น engine(engine1) โดยทั่วไปมักกำหนดสัญลักษณ์ class(object).method() เมื่อต้องการแสดงแนวคิดของคลาส อ็อปเจคและวิธีทำงานตามลำดับ ถ้าไม่ต้องการระบุคลาสโดยตรง สัญลักษณ์เขียนแทนคือ object.method() อ็อปเจคต่างๆสื่อ

สารกันโดยส่งเมสเซสระหว่างกัน ซึ่งเป็นสัญญานแบบไม่ต่อเนื่อง สมมติว่าอ็อปเจค A ส่งเมสเซสไปให้อ็อปเจค B ข้อมูลเอาพุทจาก A จะถูกส่งตรงไปเป็นอินพุท B การส่งเมสเซสในภาษาจาวาจะถูกเข้ารหัสที่ B เพื่อเลือกวิธีการทำงานที่เหมาะสมตามอินพุทที่รับมาจาก A ภายใน B สามารถมีวิธีการทำงานเช่น solve() ซึ่งให้สถานะปัจจุบัน(คุณลักษณะของB) และอินพุทจาก A และสถานะใหม่ของ B กำหนดสมการเส้นตรงแทนได้คือ $\dot{X}=MX+NU$ แสดงทางที่เป็นไปได้ที่ solve() ทำงาน

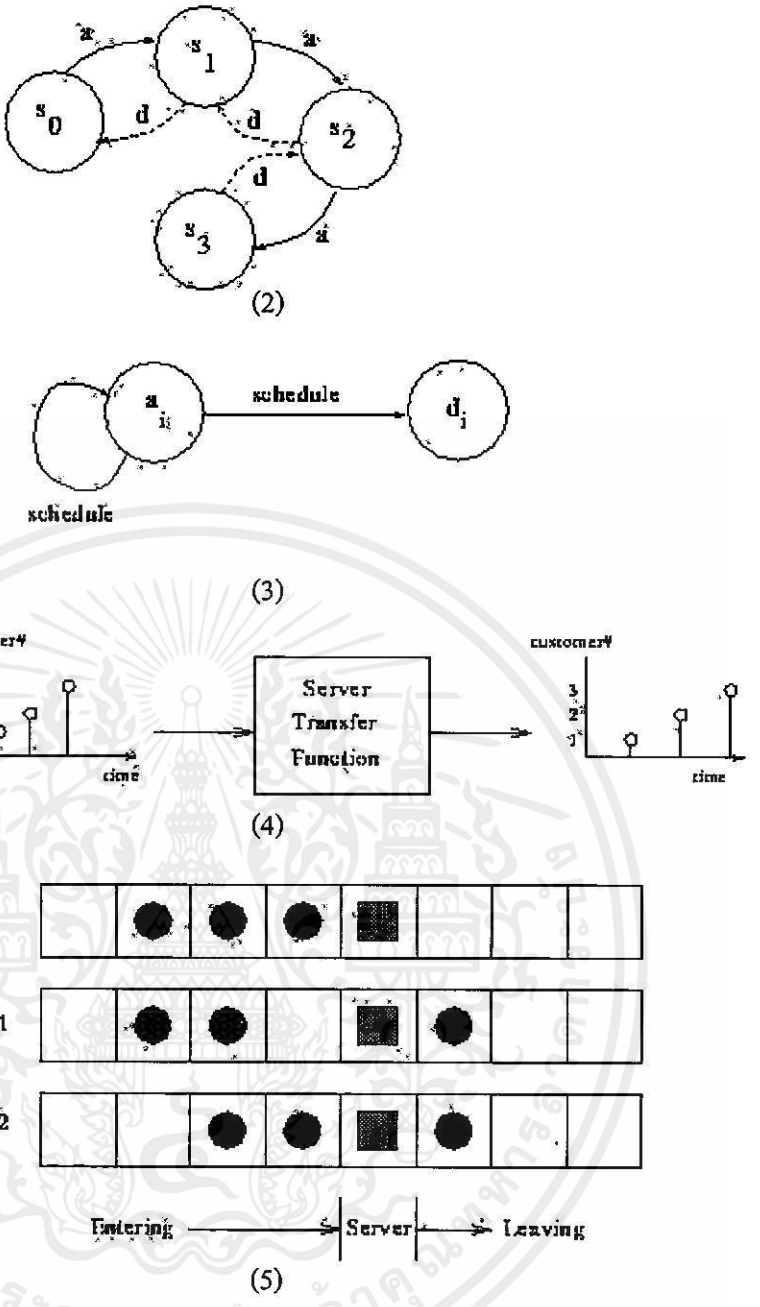
สถาปัตยกรรมของระบบซอฟต์แวร์เชิงวัตถุ(Object-oriented) ใช้คลาส (Class) เป็นตัวกำหนดการทำงานของโปรแกรม การนำคลาสมาใช้ก็โดยการใช้งานอ็อปเจค (Object)ของคลาสนั้น โดยจะส่งเมสเซส (Message) ให้อ็อปเจคเหล่านั้น ทำให้เกิดผลลัพธ์ขึ้น ระบบซอฟต์แวร์เชิงวัตถุมีความน่าเชื่อถือมากเพราะการประกอบส่วนต่างๆของซอฟต์แวร์ขึ้นเป็นระบบทำในขั้นสูงและเกิดขึ้นตั้งแต่เริ่มต้นการออกแบบด้วยการประกอบซอฟต์แวร์ในขั้นสูงนี้ยังสามารถตรวจสอบได้ก่อนที่รายละเอียดส่วนใหญ่ในขั้นต่ำจะถูกเขียนขึ้น

3.5.4 พิจารณาการสร้างโมเดล

เพื่อเข้าใจแนวคิดการสร้างโมเดล ได้พิจารณาระบบที่เกี่ยวข้องกับการรอคิวรับบริการ แนวคิดเบื้องต้นเกี่ยวกับระบบคิววิธีกำหนดภาวะ(Declarative perspective) วิธีนี้มีการระบุและการเปลี่ยนแปลงหรือกฎเพื่อบอกว่าในเวลาถัดไปสถานะต่อไปคืออะไร สถานะที่กำหนดให้แทนด้วยจำนวนวัตถุที่มีชื่อในระบบ(ทั้งที่รอในคิวและรับบริการ) การมาถึงและการจากไปเป็นเหตุการณ์ภายนอกและภายในที่ทำให้สถานะเปลี่ยนแปลง อีกแนวทางหนึ่งของวิธีกำหนดสถานะคือเน้นไปที่สถานะคู่(Dual) หรือสถานการณ์คู่ พิจารณาตามเหตุการณ์จะกำหนดเหตุการณ์มาถึงและการจากไปโดยการใช้เส้นเชื่อมโยงแทนการสับหลีก(Scheduling) เหตุการณ์ทั้งหมด(ทั้งการมาถึงและการจากไป)ถือว่าเป็นส่วนหนึ่งของเหตุการณ์รวม โดยที่การมาถึงจะอ้างถึงเซ็ตของการมาถึงทั้งหมดและการจากไปอ้างถึงเซ็ตของการจากไปทั้งหมดเช่นกัน ความสัมพันธ์ของเหตุการณ์ถูกกำหนดอย่างเคร่งครัดจากกลุ่มมาถึง $i(a_i)$ ไปสู่กลุ่มจากไป $i(d_i)$ รูปที่ 3.13(2) แสดงถึงคลาสของโมเดล 4 แบบสำหรับระบบคิวผู้แบบผู้ให้บริการคนเดียว



(1)



รูปที่ 3.14 แสดงชนิดโมเดล

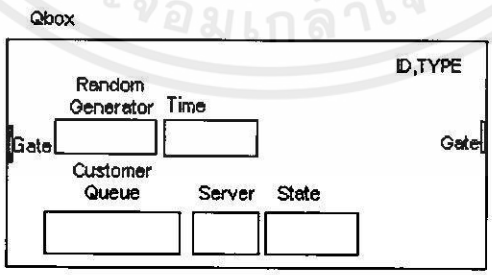
รูปที่ 3.14 (1) แสดงโมเดลเบื้องต้น เรียกว่าโมเดลระดับแนวคิด ใช้เป็นโมเดลตามธรรมชาติสำหรับสื่อสารถึงกระบวนการของคิวได้ โมเดลกำหนดภาวะในรูปที่ 3.14(2) ได้นิยามสถานะ 4 แบบที่เป็นไปได้สำหรับระบบ โดยสถานะถูกนิยามด้วยจำนวนคนที่รออยู่ในคิว ขณะเริ่มต้นจะไม่มีคนรอในคิว ดังนั้นสถานะยังอยู่ที่ S_0 เมื่อมีคนเข้าในระบบ สถานะจะเปลี่ยนเป็น S_1 ถ้ามีจนรับบริการเสร็จแล้วออกไป สถานะจะกลับมาที่ S_0 ดังนั้นเหตุการณ์ที่ทำให้สถานะของระบบเปลี่ยนแปลงคือการมาถึงและการจากไป การมาถึงเป็นอินพุทภายนอกเข้าสู่ระบบ(เหตุการณ์ภายนอก) และการจากไปเป็นอินพุทภายในเข้าไปยังระบบ(เหตุการณ์ภายใน) โมเดลชนิดนี้เป็นแบบสถานะจำกัดอัตโนมัติ สามารถเปลี่ยนสถานะได้โดยการกำหนดความน่าจะเป็นให้กับเส้นเชื่อมโยง อย่างไรก็ตามโมเดลนี้ไม่มีอินพุทเข้ามาเกี่ยว

ข้อง จุดที่สนใจในการจำลองแบบ ต้องกำหนดเวลาให้กับระบบในแต่ละสถานะ โมเดลกำหนดภาวะที่ปรับปรุงขึ้นอีก ได้ระบุเส้นเชื่อมระหว่างสถานะ i และ i+1 เมื่อ i เป็นสมาชิกของ {0,1,2} ส่วนเส้นเชื่อมโยงย้อนกลับ สามารถนิยามด้วยหลักการเดียวกัน สามารถเขียนสัญลักษณ์แสดงได้คือ state(A) :- state(B), 0 <= B < 3 , A เป็น B+1 โดยสัญลักษณ์ :- แทนการเปลี่ยนสถานะ โมเดลนี้อธิบายด้วยรูป 3.14 (2)

นอกจากการมองระบบจากมุมของสถานะหรือเหตุการณ์แล้ว สามารถแสดงระบบได้ด้วยกล่องดำ(แทนผู้ให้บริการ)(Black box) ตามรูปที่ 3.14(4) ซึ่งมีอินพุทเป็นสัญญาณไม่ต่อเนื่องที่กำหนดโดยลูกค้าที่วิ่งเข้าระบบในเวลานั้นๆ และให้เอาท์พุทเป็นสัญญาณหน่วงตามที่ลูกค้าวิ่งออกไป ความหน่วงจะเกิดขึ้นตลอดเวลาการจำลองแบบ

สำหรับฟังก์ชันโมเดล การเพิ่มเวลาหน่วงตามฟังก์ชันในกล่องดำเป็นความไม่ต่อเนื่อง(ตามสมการดิฟเฟอเรนเชียล) หรือแบบต่อเนื่อง(ตามสมการดิฟเฟอเรนเชียล) แบบใดแบบหนึ่ง ฟังก์ชันของโมเดลในรูป 3.14 ถูกนำมารวมกัน โดยใช้โมเดลกำหนดและกล่องดำ โมเดลในรูปที่ (4) (5) เป็นโมเดลทางเรขาคณิต วัตถุที่มีชื่อแทนด้วยวงกลมและผู้ให้บริการแทนด้วยสี่เหลี่ยม เมื่อมีคิวของผู้ให้บริการเกิดขึ้น สามารถอธิบายได้ 2 ระยะคือ ลูกค้ากำลังรับบริการ หรือลูกค้าที่เหลืออยู่มีการเปลี่ยนตำแหน่งในคิว

จากตัวอย่างแนวคิดการออกแบบโมเดลระบบคิวเมื่อนำมาออกแบบเชิงวัตถุตามข้อกำหนดของโปรแกรมที่พัฒนานี้ สามารถกำหนดวัตถุที่มีชื่อ(Entity)ได้ดังต่อไปนี้ 1) ลูกค้า(Customer) 2) คิว (Customer Queue) 3) เวลาหรือนาฬิกา (Clock) 4) ผู้ให้บริการ (Server) 5) เครื่องสร้างตัวเลขสุ่ม (Random Generator) 6) ตัวเชื่อมโครงข่าย(Gate) และ 7) ส่วนเก็บข้อมูลสถิติ(State) แต่ละ Entity จะแทนด้วยคลาสหนึ่งคลาส โดยแบบจำลองที่กำหนดขึ้นใช้นี้เรียกว่า**"ระบบคิว"** ดังรูปที่ 3.15



รูปที่ 3.15 โมเดลระบบคิวที่ใช้ในการศึกษา

อ็อปเจคแทนระบบคิวนี้กำหนดให้มี 1 คิวและมีผู้ให้บริการได้มากกว่า 1 คน คุณสมบัติของระบบคิวนี้คือจำลองพฤติกรรมของคิวได้ ระบบนี้สนใจเพียงเวลาที่ลูกค้าเข้ามาและเวลาที่ลูกค้าออกไปจากระบบแล้วทำการบันทึกข้อมูลสถิติของระบบ จากคุณสมบัตินี้

สามารถนำระบบคิวมาต่อเชื่อมเข้าด้วยกันด้วยฟังก์ชันตัวเชื่อมโครงข่ายเพื่อทำงานเป็นระบบโครงข่ายคิวได้ แต่ละอ็อปเจคสามารถวิ่งได้พร้อมพร้อมกันภายใต้นาฬิกาอันเดียวกัน การติดต่อกันแบบโครงข่ายสามารถทำได้สะดวกในระบบการโปรแกรมเชิงวัตถุมีการออกแบบจุดเชื่อมต่อและใช้เทคนิคการส่งเมสเซสไปมาระหว่างวัตถุแต่ละตัวในระบบ และต้องมีการควบคุมเงื่อนไขด้านเวลาอย่างสอดคล้องกันด้วย โครงสร้างข้อมูลที่ซับซ้อนสามารถแก้ปัญหาได้ด้วยการโปรแกรมแบบวัตถุเนื่องจากมีคุณสมบัติในการถ่ายทอดคุณสมบัติ (Inheritance) และการนำโค้ดมาใช้ใหม่(Code reuse)ได้ และสามารถสร้างฟังก์ชันที่ทำงานได้เป็นธรรมชาติมากขึ้นเพื่อแก้ปัญหาที่ซับซ้อน

3.5.5 การออกแบบโปรแกรมระบบคิวตามแนวคิดที่ใช้ในการวิจัย

3.5.5.1 การออกแบบการจำลองคิวในขั้นสูง (High-level)

วิธีการแก้ปัญหาโดยวิธีในใจวัตถุนั้น เริ่มด้วยการแตกปัญหาออกเป็นส่วน ๆ และแทนแต่ละส่วนด้วยวัตถุ ดังนั้นเราจึงต้องกำหนดส่วนหลักๆ ที่สำคัญเสียก่อน หลังจากนั้นค่อยกำหนดรายละเอียดย่อย ๆ ในภายหลัง สำหรับโปรแกรมการจำลองชุดนี้ จะแบ่งกำหนดวัตถุที่มีชื่อ(Entity) ได้ดังนี้

1. เซลล์พื้นฐาน(Cell)
2. ส่วนเก็บข้อมูลสถิติ(State)
3. ลูกค้า (Customer)
4. คิวลูกค้า (Customer Queue)
5. เวลาหรือนาฬิกา (Clock)
6. พนักงานบริการ (Server)
7. เครื่องสุ่มตัวเลข (Random Generator)
8. ตัวเชื่อมพื้นฐาน (Base Gate Router)
9. ตัวเชื่อมแบบสุ่ม (Random Gate Router)
10. ตัวเชื่อมแบบลำดับ (Sequence Gate Router)
11. เซลล์ที่สามารถวิ่งได้ (Runable Cell)
12. เซลล์ระบบคิว (Runable Queue System Cell)

โดยแต่ละวัตถุที่มีชื่อจะแทนด้วยคลาส (Class) หนึ่งคลาส ตามรายละเอียดดังต่อไปนี้

คลาส Cell เป็นคลาสพื้นฐานที่กำหนดขึ้นมาอธิบายอ็อปเจคในโปรแกรม ตัวแปรที่สำคัญมีดังนี้

ตัวแปร Private (คือตัวแปรประจำแต่ละอ็อปเจคที่อ็อปเจคอื่นไม่สามารถเข้าถึงได้)

- ไม่มี

ตัวแปร Public (คือตัวแปรที่ยอมให้อ็อปเจคอื่นเข้าถึงได้โดยผ่านฟังก์ชัน)

Public int id	อธิบายรหัสเซลล์ กำหนดด้วยข้อมูลชนิดอินทีเจอร์
Public String types	อธิบายชนิดของเซลล์ เช่น Supplier , Qbox, Terminator, หรือ Linking
Public int entry	อธิบายการรับข้อมูลสถิติของเซลล์เมื่อมีลูกค้าเข้ามาในระบบคิว
Public int depart	อธิบายการเก็บข้อมูลสถิติเมื่อมีการส่งลูกค้าออกจากระบบคิว
Public int count	เพื่อนับจำนวนลูกค้าในระบบคิว

ตัวแปร Protected (คือตัวแปรที่อนุญาตให้อ่านเฉพาะ friend หรือ อ็อปเจคลูกเท่านั้น)

- ไม่มี

อินพุท

Type เป็นตัวแปรชนิด String

Id เป็นตัวแปรชนิด Integer

เอาท์พุท

ไม่มีการคืนค่าใด ๆ

ฟังก์ชัน

1. Public Cell(String types, int id) เพื่อจองตำแหน่งเซลล์ในหน่วยความจำเมื่อสร้างอ็อปเจคขณะโปรแกรมทำงาน
2. Public void reset() เพื่อ reset ค่าเริ่มต้นใหม่
3. Public String name() เพื่อแสดงชื่อของเซลล์
4. Public abstract Boolean empty() เพื่อบอกสถานะว่าง แล้วรูทีน get() ไม่สามารถดึงข้อมูลได้
5. Public abstract int get() เพื่อดึงข้อมูลลูกค้าออกจากเซลล์
6. Public abstract boolean busy() เพื่อบอกสถานะว่าไม่ว่าง แล้วรูทีน put() ไม่สามารถเพิ่มลูกค้าเข้าไปในระบบได้ ต้องรอจนกว่าสถานะจะพร้อม
7. Public abstract void put(int x) เพื่อรับลูกค้าเข้ามาในเซลล์

คลาส State ถ่ายทอดคุณสมบัติมาจากคลาสเซลล์ มีหน้าที่เก็บข้อมูลของเซลล์ระบบคิว โดยแต่ละระบบคิวจะมีตัวเก็บข้อมูลแยกจากกัน ตัวแปรพื้นฐานมีดังนี้

ID	บอกรหัสของ State
LENGTH	บอกความยาวข้อมูลของ State

อินพุท

1. ID บอกว่าเป็นประเภทเซลล์เก็บข้อมูล
2. LENGTH, ความยาวเริ่มต้นคือ 1024 ไบต์ โดยขนาดสามารถเพิ่มขึ้นได้ทีละ 1024 ไบต์ ใช้โครงสร้างอาเรย์เป็นตัวจัดการ

เอาท์พุท

ไม่มีคืนค่าใดๆ ออกไป

ฟังก์ชัน

1. ฟังก์ชันพื้นฐานของคลาสเซลล์ โดยชื่อฟังก์ชันเหมือนกันแต่นิยามข้างในแตกต่างกันตามหลักของ polymorphism[10] เพื่อกำหนดพฤติกรรมการทำงานของ State
2. Public State(int id,int length) เพื่อจองตำแหน่ง State ในหน่วยความจำเมื่อสร้างอ็อปเจคใช้งาน
3. Public State dump() เพื่อดึงจำนวนหน่วยความจำออกมาเมื่อมีการย้ายตำแหน่งข้อมูลในหน่วยความจำ หรือการจัดการหน่วยความจำ
4. Public void put_start(int time) เพื่อเก็บข้อมูลเมื่อมีลูกค้าเข้ามาในระบบคิว
5. Public void put_depart(int time) เพื่อเก็บข้อมูลเมื่อลูกค้าออกไปจากระบบคิว
6. Public String toString() เพื่อพิมพ์ชื่อของ State

คลาส Customer Queue ถูกถ่ายทอดคุณสมบัติมาจากคลาสเซลล์ และมีการนิยามตัวแปรเพิ่มเติม ประกอบด้วยส่วนต่าง ๆดังต่อไปนี้

ตัวแปร Private

1. Constructor อธิบายโครงสร้างของคิวในหน่วยความจำ
2. ขนาดของคิว (Queue Length) โดยขนาดเริ่มต้นคือ 1024 ไบต์โดยขนาดสามารถเพิ่มขึ้นได้ทีละ 1024 ไบต์ ใช้โครงสร้างอาเรย์เป็นตัวจัดการ
3. ดัชนีบอกตำแหน่งหัวคิว (Queue head) และท้ายคิว (Queue tail) เพื่อใช้ควบคุมการแทรก(Insert)และลบ(Delete) ลูกค้าในคิว
4. Next คือตัวชี้บอกตำแหน่งในหน่วยความจำเมื่อมีการขยายขนาดและลดขนาด

ตัวแปร Public

ไม่มี

อินพุท

ขนาดความยาวของคิว (Queue Length)

เอาท์พุท

ไม่มีการคืนค่าใดๆ

ฟังก์ชัน

1. (ชุดฟังก์ชันพื้นฐานของคลาสเซลล์ทั้งหมด ชื่อฟังก์ชันเหมือนกันแต่การนิยามข้างในแตกต่างกันเพื่อกำหนดพฤติกรรมของคิว)
2. Public Queue() เพื่อจองตำแหน่ง Queue ในหน่วยความจำ

3. Public int count() เพื่อนับจำนวนขนาดความยาวของคิวและควบคุมขนาดของคิวให้เหมาะกับขนาดข้อมูลที่ปรับปรุงในขณะวิ่งโปรแกรม

คลาส Server ใช้นิยามผู้ให้บริการ โดยผู้ให้บริการมีได้มากกว่า 1 คน โดยแต่ละคนอาจมีเวลาการให้บริการที่แตกต่างกันได้

ตัวแปร Private

1. Private int tcount ใช้นับเวลาของ Server
2. Private int time ใช้กำหนดเวลาที่ลูกค้าเข้ามาที่ Server
3. Private int ltime ใช้ควบคุมเวลาของแต่ละ Server (เวลท้องถิ่น,local time)
4. Private State state เพื่อเก็บข้อมูลสถิติของ Server
5. Private boolean serve เพื่อบอกว่า Server เริ่มให้บริการ
6. Private boolean finish เพื่อบอกว่า Server เสร็จสิ้นการให้บริการ
7. Private Xrandom random เพื่อสร้างตัวเลขสุ่มสำหรับกำหนดค่าเวลาการให้บริการ

ตัวแปร Public

Public int busytime บอกสถานะ Server ไม่ว่าง

ฟังก์ชัน

1. ฟังก์ชันพื้นฐานที่ถ่ายทอดมาจากเซลล์พื้นฐาน
2. Public Server(int id, int time, Xrandom random) เพื่อจองตำแหน่ง Server ในหน่วยความจำ
3. Public State state() เพื่อเก็บสถิติของเซลล์ โดยจะมีการวัดเวลาที่ Serverให้บริการ (Busy time) ตำแหน่งเวลาที่ลูกค้าได้รับบริการ และตำแหน่งเวลาที่รับบริการแล้วเสร็จ
4. Public String toString() เพื่อแสดงชื่อของ Server

คลาส Random Generator สร้างตัวเลขสุ่มตามรูปแบบกระจายที่กำหนดโดยมีอินพุทคือค่าตัวเลขใด ๆ เช่น ในหนึ่งหน่วยเวลาสามารถส่งลูกค้าออกมาได้ n คนเป็นต้น

ตัวแปร Private

Private Random r

ตัวแปร Public

- ไม่มี

ฟังก์ชัน

1. Public Xrandom() เพื่อจองตำแหน่ง Random Generator ในหน่วยความจำ
2. Public int get(int service_time) เพื่อรับค่าเพื่อสุ่มตัวเลขเวลาให้บริการของ Server
3. Public static int rand(int max) เพื่อสร้างตัวเลขสุ่มใด ๆ ให้กับส่วนเชื่อมโยงโครงข่าย

คลาส Gate Router ทำหน้าที่เป็นตัวเชื่อมต่อเซลล์ ในโปรแกรมเข้าหากันเป็นโครงข่าย

ตัวแปร Private

- ไม่มี

ตัวแปร Public

Public int count เพื่อนับจำนวนเซลล์

ตัวแปร Protected

1. Protected Cell src[] เพื่อบอกจุดเริ่มต้นของการเชื่อมต่อ
2. Protected Cell dst[] เพื่อบอกจุดปลายทางของเชื่อมต่อ

ฟังก์ชัน

1. Public Gate(int id) เพื่อจองตำแหน่ง Gate ในหน่วยความจำ
2. Public void reset() เพื่อ Reset ค่าเริ่มต้นให้แก่ Gate
3. Public viod gate(Cell src[],Cell dst[]) เพื่อกำหนดการเชื่อมต่อเซลล์
4. Public abstract void route() เพื่อบอกให้มีการส่งสัญญาณไปยังเซลล์ถัดไป
5. Public String toString() เพื่อแสดงชื่อของ Gate
6. Public Gate dump() เพื่อขอดูตำแหน่งในหน่วยหน่วยความจำและจัดการหน่วยความจำ

คลาส Random Gate Router ถ่ายทอดคุณสมบัติมาจาก Gate Router ทำหน้าที่เป็นตัวเชื่อมต่อเซลล์ ในโปรแกรมเข้าหากันเป็นโครงข่าย โดยอัลกอริธึมจะเป็นชนิดการทำงานแบบสุ่ม

ตัวแปร Private

- ไม่มี

ตัวแปร Public

Public int count เพื่อนับจำนวนเซลล์

ตัวแปร Protected

Protected Cell src[] เพื่อบอกจุดเริ่มต้นของการเชื่อมต่อ

Protected Cell dst[] เพื่อบอกจุดปลายทางของการเชื่อมต่อ

ฟังก์ชัน

1. Public Gate(int id) เพื่อจองตำแหน่ง Gate ในหน่วยความจำ
2. Public void reset() เพื่อ Reset ค่าเริ่มต้นให้แก่ Gate
3. Public viod gate(Cell src[],Cell dst[]) เพื่อกำหนดการเชื่อมต่อเซลล์
4. Public abstract void route() เพื่อบอกให้มีการส่งสัญญาณไปยังเซลล์ถัดไป
5. Public String toString() เพื่อแสดงชื่อของ Gate
6. Public Gate dump() เพื่อขอดูตำแหน่งในหน่วยหน่วยความจำ

7. Public int route() เพื่อกำหนดการเชื่อมต่อเซลล์

คลาส Sequence Gate Router ถ่ายทอดคุณสมบัติมาจาก Gate Router ทำหน้าที่เป็นตัวเชื่อมต่อ Cell ในโปรแกรมเข้าหากันเป็นโครงข่าย โดยอัลอริธึมจะเป็นชนิดการทำงานแบบลำดับ

ตัวแปร Private

ไม่มี

ตัวแปร Public

Public int count เพื่อนับจำนวนเซลล์

ตัวแปร Protected

1. Protected Cell src[] เพื่อบอกจุดเริ่มต้นของการเชื่อมต่อ
2. Protected Cell dst[] เพื่อบอกจุดปลายทางของการเชื่อมต่อ

ฟังก์ชัน

1. Public Gate(int id) เพื่อจองตำแหน่ง Gate ในหน่วยความจำ
2. Public void reset() เพื่อ Reset ค่าเริ่มต้นให้แก่ Gate
3. Public viod gate(Cell src[],Cell dst[]) เพื่อกำหนดการเชื่อมต่อเซลล์
4. Public abstract void route() เพื่อบอกให้มีการส่งสัญญาณไปยังเซลล์ถัดไป
5. Public String toString() เพื่อแสดงชื่อของ Gate
6. Public Gate dump() เพื่อขอดูตำแหน่งในหน่วยหน่วยความจำ
8. Public int route() เพื่อกำหนดการเชื่อมต่อเซลล์

คลาส Runable Cell ถ่ายทอดคุณสมบัติมาจากคลาส Cell และมีการนิยามให้มีคุณสมบัติในการวิ่งได้เมื่อมีการสั่งการ

ตัวแปร Private

ถ่ายทอดมาจากคลาสเซลล์

ตัวแปร Public

ถ่ายทอดมาจากคลาสเซลล์

ตัวแปร Protected

ถ่ายทอดมาจากคลาสเซลล์

ฟังก์ชัน

Public Rcell(int id)

คลาส QBox ถ่ายทอดคุณสมบัติมาจากคลาส Runable Cell

ตัวแปร Private

ถ่ายทอดมาจากคลาส Runable Cell

ตัวแปร Public

ถ่ายทอดมาจากคลาส Runable Cell

ตัวแปร Protected

ถ่ายทอดมาจากคลาส Runable Cell

ฟังก์ชัน

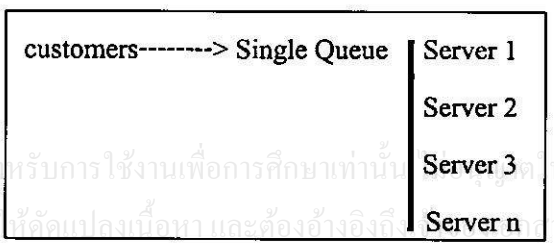
1. Public QBox(int id, int service_time[],Gate in, Gate out) เพื่อจองตำแหน่งในหน่วยความจำ
2. Public Qbox duupQBox()เพื่อขอดูตำแหน่งในหน่วยความจำ
3. Public boolean empty() เพื่อบอกสถานะว่าคิวว่าง
4. Public int get() เพื่อดึงรายการออกจากคิว
5. Public boolean busy() เพื่อบอกสถานะว่าคิวไม่ว่าง
6. Public void put(int I) เพื่อในรายการเข้าไปในคิว
7. Public void run(int g_time) เพื่อสั่งให้มีการวิ่งตามเงื่อนไขเวลาที่กำหนดให้
8. Public String toString() เพื่อแสดงชื่อ QBox

3.5.5.2 การออกแบบการจำลองคิวในขั้นต่ำ (low-level)

อัลกอริธึมที่จำเป็นในการใช้งานเบื้องต้น คือกำหนดคิวด้วยขนาดอาเรย์(array)และมีการควบคุมขนาดคิวเมื่อขยายตัวด้วย Linked List โดยมี Head กับ Tail ควบคุมการ insert และ delete หรืออาจกล่าวได้ว่าเป็นลิงค์ของอาเรย์นั่นเอง ขนาดเริ่มต้นของ array คือ 1024 ไบต์

ตารางที่ 3.12 แสดงฟังก์ชันของคิวขณะ Insert และ Delete

	First	Last
Start	0	First=Last
Insert	-	qh++,entry++
Delete	qt++,depart++	-



รูปที่ 3.16 แสดงโครงสร้างคิวตัวอย่าง

เริ่มจากเวลาที่ลูกค้ามาถึง ลูกค้าจะต้องผ่าน Router เพื่อหาตำแหน่งปลายทาง ซึ่งจะมีอัลกอริธึมแบบสุ่มหรือแบบลำดับหรือแบบมีลำดับความสำคัญ(Priority) หรือแบบกำหนดความน่าจะเป็น เมื่อมาถึง QBox ก็มาเข้ารอในคิวถ้าคิวไม่เต็มและมีการเก็บค่าสถิติของคิว ในขณะเดียวกันถ้าผู้ให้บริการ(Server) ว่างอยู่ ลูกค้าก็จะวิ่งเข้ารับบริการและมีการเก็บค่าสถิติ ในกรณีมีผู้ให้บริการหลายคน จะมีการเลือกวิธีเข้าไปรับบริการตามที่กำหนด ซึ่งมีอัลกอริธึมแบบสุ่มหรือแบบลำดับหรือแบบมีลำดับสำคัญ เนื่องจากทุกๆเซลล์ในระบบมีคุณสมบัติ Runable ที่เหมือนกัน ดังนั้นในโปรแกรมหลักสามารถสั่งให้ทุกเซลล์ทำงานพร้อมกันได้ในเงื่อนไขเวลาทำงานเดียวกัน การเก็บสถิติของเซลล์แต่ละเซลล์จะมีการเก็บข้อมูลแยกจากกันเพราะว่ามีพฤติกรรมต่างกันนั่นเอง ข้อมูลใน State จะมีเท่าที่จำเป็นในการออกรายงาน ทั้งนี้เพื่อประหยัดหน่วยความจำ

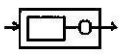
ในการวิ่งโปรแกรม ผู้ใช้ต้องกำหนดเวลาในการวิ่ง เมื่อเวลาเท่ากับค่าที่ป้อนให้การวิ่งจำลองแบบจะจบลงแล้วระบบจะให้รายงานออกมาเป็นค่าสถิติต่าง ๆ รายงานสรุปประสิทธิภาพการทำงาน หรือสถานะการทำงานตามลำดับเวลา(Trace) การทำงานที่เวลาต่าง ๆ และสามารถดูประสิทธิภาพการทำงานได้จากกราฟแสดงพฤติกรรมของระบบคิวหรือรายงานในประเด็นต่าง ๆ

การออกแบบในโปรแกรมนี้ ได้เน้นให้มีความยืดหยุ่นกับตัวแปรทุกตัว สามารถแก้ไขโครงสร้างได้ง่ายโดยไม่ทำให้โครงสร้างเดิมเสีย ดังนั้นโปรแกรมนี้จึงขยายการทำงานได้มาก ข้อจำกัดที่มีจะอยู่ที่ปริมาณหน่วยความจำของเครื่องคอมพิวเตอร์

3.6 ตัวอย่างผลลัพธ์ของการวิ่งโปรแกรมจำลองแบบคิวและการวิเคราะห์

ปัญหาที่น่าสนใจในการวิ่งระบบคือต้องวิ่งระบบนานแค่ไหนถึงจะได้รับคำตอบที่น่าเชื่อถือและต้องถูกต้องด้วย การวิ่งโปรแกรมมีผลกระทบโดยตรงต่อเวลาและค่าใช้จ่าย ดังนั้นการหาผลลัพธ์ที่ถูกต้องของระบบจึงต้องมีการควบคุมอย่างมีทิศทางและจำเป็นต้องใช้หลักการวิเคราะห์ทางสถิติเนื่องจากข้อมูลที่ป้อนให้กับระบบเป็นข้อมูลทางสถิติ ผลลัพธ์ที่ออกมาจากระบบก็จำเป็นต้องวิเคราะห์ด้วยหลักสถิติเช่นเดียวกัน การวัดผลที่ได้รับจากระบบอาจทำได้ทั้งภาวะที่ระบบเสถียรและไม่เสถียรทั้งนี้ขึ้นอยู่กับแต่ละปัญหา โดยทั่วไปจะวัดค่าเฉลี่ยด้านต่างๆของตัวแปรว่าใกล้เคียงกับ "True mean, μ" ของค่าการกระจายหรือไม่ ซึ่งวิธีการในการวัดที่นิยมนำมาใช้คือพิสูจน์ว่าค่าเฉลี่ยที่ได้อยู่ในช่วงความมั่นใจ(Confidence Interval) ที่กำหนดหรือไม่[14] ทางหลักสถิติจะสมมติว่าจำนวนตัวอย่างประชากรมากจะส่งผลให้ได้คำตอบมีความน่าเชื่อถือมากขึ้น ในการจำลองแบบก็สามารถทำได้ด้วยการวิ่งระบบหลายๆครั้งแล้วนำผลลัพธ์มาวิเคราะห์ค่าทางสถิติ

3.6.1 ระบบ 1 คิว 1 ผู้ให้บริการ - M/M/1



Time=500, Ts=2 ,Ta=1 (หน่วยเวลา)

วิ่งโปรแกรมทดสอบ 10 ครั้ง ได้ค่า

Mean queuing time : 73.51 109.80 94.04 88.67 105.06 94.85 65.38 96.13 98.06 85.11

95 % of CI for mean queuing time : 1-α =0.95 ,so α = 0.05

Sample mean = 91.06

(max,min) = (109.80,65.38)

Sample variance, s^2 =183.74

SD of sample mean, $s/N^{1/2}$ =18.37

t .05/2 ; 9 =2.26 (จากตาราง)

CI half width H=2.26x18.37 = 41.52

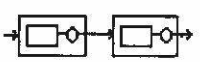
CI = 91.06 $\pm$41.52

พบว่าที่ 95 % ของ CI ค่า mean จะอยู่ระหว่าง [74.45-107.87] จากการทดลองมีค่า mean อยู่ในช่วงนี้ถึง 7 ค่า ในบางครั้งการสรุปผลอาจแสดงค่า Percentage of mean ในที่นี้ค่า percentage of mean = 65.38 % ถ้าต้องการให้ผลถูกต้องมากขึ้นเช่นต้องการให้ค่าเปอร์เซนต์ลดลงอีกครึ่งหนึ่งหมายถึงการลดค่า H ให้แคบลงอีก ทำได้ดังนี้ พิจารณาค่าครึ่งช่วงความมั่นใจ H H = t α/2 ; N-1 ,$s/N^{1/2}$

เทอมแรกเป็นฟังก์ชันของ N การเพิ่มค่า N เป็นอินฟินิตี้ ค่า H ลดลงได้เพียง 15 %

การแก้ไขค่า s ต้องเพิ่มค่า N เป็น 4 เท่าจึงจะลดลดค่า H ได้ 2 เท่า ในที่นี้คือต้องวิ่งโปรแกรม 40 ครั้ง หรือเพิ่มช่วงเวลาการวิ่งให้ยาวมากขึ้น โดยสรุปก็คือการเพิ่มความถูกต้องอีก 2 เท่า จะต้องวิ่งโปรแกรมยาวนานขึ้นเป็น 4 เท่า คือวิ่งนาน 2,000 หน่วยเวลา

3.6.2 ระบบ 2 คิว 2 ผู้ให้บริการแบบอนุกรม



Time=500, Ts=2 ,Ta=1 หน่วยเวลา

Mean queuing time : 84.01 60.76 60.46 45.26 61.70 61.92 52.14 65.02 57.46 55.26

95 % of CI for mean queuing time : 1-α =0.95 ,so α = 0.05

Sample mean = 60.40

(max,min) = (84.01,45.26)

Sample variance ,s^2 =101.69

SD of sample mean, $s/N^{1/2}$ =4.06

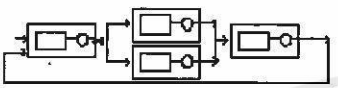
t .05/2 ; 9 =2.26

CI half width H=2.26x0.33 = 9.19

CI = 60.40 $\pm$9.19

พบว่าที่ 95 % ของ CI ค่า Mean จะอยู่ระหว่าง [51.20-69.59] จากการทดลองมีค่า mean อยู่ในช่วงนี้ถึง 8 ค่า ในบางครั้งการสรุปผลอาจแสดงค่า Percentage of mean ในที่นี้ ค่า Percentage of mean = 45.26 %

3.6.3 ระบบ 4 คิว 4 ผู้ให้บริการ (ระบบปิด)



Time=500, Ts=2 ,Ta=1

Mean queuing time : 33.40 32.09 33.94 33.01 29.20 31.66 36.12 37.96 29.89

95 % of CI for mean queuing time : 1-α =0.95 ,so α = 0.05

Sample mean = 32.48

(max,min) = (37.96,29.20)

Sample variance, s^2 =9.95

SD of sample mean ,$s/N^{1/2}$ =0.39

t .05/2 ; 9 =2.26

CI half width H=2.26x0.39 = 0.9

CI = 32.48 $\pm$0.9

พบว่าที่ 95 % ของ CI ค่า mean จะอยู่ระหว่าง [31.58-33.38] จากการทดลองมีค่า Mean อยู่ในช่วงนี้ถึง 3 ค่า

ตารางที่ 3.13 แสดงค่า $t\alpha/2;N-1$

	$\alpha/2=0.05$	$\alpha/2=0.025$
N-1	t	t
4	2.13	2.78
5	2.02	2.57
6	1.94	2.45
7	1.90	2.37
8	1.86	2.31
9	1.83	2.26
10	1.81	2.23
.		
∞ 1.65	1.96	

3.7 การปรับปรุงประสิทธิภาพการจำลองแบบ [6]

ประสิทธิภาพของของการจำลองแบบวัดจากจำนวน Sample values ที่ต้องการไปทำให้ได้ความถูกต้อง โดยจำนวนนี้ขึ้นอยู่กับ Sample Variance ตามสมการค่าครึ่งช่วงความเชื่อมั่น, H

$$k = [t_{\alpha/2}; k-1/H]^2 s^2 \qquad \text{----------(43)}$$

โดย k เป็นจำนวนของชุดการทำซ้ำ(Replications) หรืองานกลุ่ม(Batches) และแต่ละชุดประกอบด้วย Sample values เป็น m เพื่อหาความถูกต้องและช่วงความมั่นใจ (กรณีไม่สนใจความแปรปรวนของ t ด้วยค่า k)

$$n = mk \approx cs^2 \qquad \text{----------(44)}$$

โดยที่ค่า c เป็นค่าคงที่ ดังนั้นขนาดตัวอย่างที่จะส่งผลให้ได้ความถูกต้องจะขึ้นอยู่กับค่าความแปรปรวน สามารถปรับปรุงประสิทธิภาพของการจำลองแบบได้ถ้าสามารถหาค่าเฉลี่ยเพื่อไปลดค่าความแปรปรวนลงได้ มีเทคนิคหลายอย่างในการปรับค่านี้ แต่ที่นิยมปฏิบัติคือการวิ่งโปรแกรมเป็นเวลานาน ๆ

ในบทนี้ได้กล่าวถึงการศึกษาโปรแกรมตัวอย่างและวิธีการออกแบบโมเดลคิวและปรับโมเดลไปสู่การออกแบบในระดับโปรแกรมทั้งในระดับสูงและระดับต่ำ นอกจากนี้ได้กล่าวถึงวิธีการวัดประสิทธิภาพของโมเดลด้วย สำหรับบทต่อไปจะอธิบายถึงการออกแบบโครงสร้างข้อมูลในรายละเอียด

บทที่ 4
โครงสร้างข้อมูล

4.1 เปรียบเทียบโครงสร้างข้อมูลแต่ละแบบ

4.1.1 โครงสร้างข้อมูลแบบคงที่(Static Data Structure) [7]

โครงสร้างข้อมูลในภาษาจาวาที่เป็นแบบคงที่ที่สำคัญคืออาเรย์(Arrays) มีข้อดีคือประยุกต์ใช้งานง่าย ไม่ซับซ้อนแต่มีข้อเสียเรื่องการสิ้นเปลืองหน่วยความจำเมื่อเกิดการจองแล้วไม่ได้ใช้ ทำให้มีข้อจำกัดมากเมื่อโปรแกรมมีอ็อปเจคมากขึ้น

4.1.1.1 คุณสมบัติของอาร์เรย์

- ข้อมูลแต่ละชุดในอาเรย์เรียกอีลีเม้นท์(Elements)
- ทุกอีลีเม้นท์ต้องเป็นชนิดข้อมูลเดียวกัน
- ทุกอีลีเม้นท์ต้องอยู่เป็นที่ผืนเดียวกันในหน่วยความจำ และดัชนี(Index) ของอีลีเม้นต้องเริ่มจากศูนย์
- ชื่อของอาเรย์เป็นตัวแปรคงที่ซึ่งจะแทนตำแหน่งแรกของอีลีเม้นท์สำหรับอาเรย์

4.1.1.2 การประกาศอาเรย์

ตัวอย่าง :-

```
int intarray[12];    /* อาเรย์ของจำนวนเต็ม (integer)12 หน่วย */
char chararray[20]; /* อาเรย์ของตัวอักษร 12 หน่วย */
```

ตัวอย่างการกำหนดขนาดอาเรย์โดยอ้อม :-

```
#define iARRAY_MAX 20
#define fARRAY_MAX 15
int intarray[iARRAY_MAX];
float floatarray[fARRAY_MAX 15];
```

4.1.1.3 การเริ่มต้นการใช้งาน

- เมื่อเริ่มสร้างตัวแปร คุณสมบัติจะเป็นโดยรวม(Global) และแบบคงที่(Static)
- ต้องมีข้อมูลเริ่มต้น(ปกติคือ 0)เมื่ออาเรย์ถูกสร้างขึ้นในหน่วยความจำ
- การกำหนดข้อมูลเริ่มต้นสามารถทำได้โดยตรง

ตัวอย่าง :-

```
int intarray[3] = {-1,0,1};
static float floatpercent[4] = {1.141579,0.75,55e0,-.33e1};
```

```
static int intdecimal[3] = {0,1,2,3,4,5,6,7,8,9};
char charvowels[] = {'A' ,'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u'};
```

4.1.2 โครงสร้างข้อมูลแบบไดนามิก(Dynamic Data Structure) [17]

พอยเตอร์ (Pointers) เป็นโครงสร้างข้อมูลที่มีความยืดหยุ่นและมีประสิทธิภาพสูงมากในภาษาซี(C Language)หรือซีพลัสพลัส(C++ Language) หรือภาษาจาวา(Java Langluag) โปรแกรมขนาดใหญ่และซับซ้อนจำเป็นต้องใช้พอยเตอร์ทั้งนี้เพื่อความรวดเร็วในการทำงานของโปรแกรมพร้อมทั้งประสิทธิภาพในการใช้หน่วยความจำ

การนิยามพอยเตอร์

ตัวอย่าง:-

imemorycell_contents คือตัวแปรแบบอินทีเจอร์ (integer)

pimemorycell_address คือตัวแปรเก็บค่าตำแหน่ง(Address) ของตัวแปรในหน่วยความจำ การระบุตำแหน่งของตัวแปรทำได้โดยการใช้เครื่องหมาย "&" วางไว้หน้าตัวแปรนั้น ๆ

```
pimemorycell_address = &imemorycell_contents;
พอยเตอร์             = &ตัวแปร
```

การเข้าถึงตัวแปรแบบพอยเตอร์ โดยการใช้เครื่องหมาย "*" เช่น

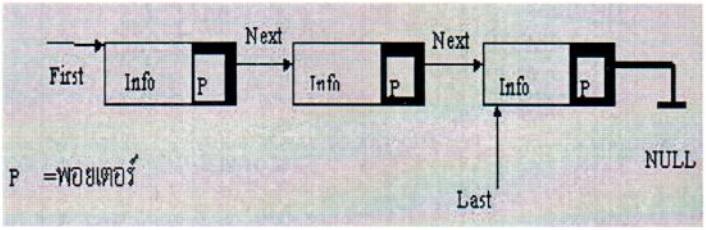
```
pimemorycell_address = &imemorycell_contents;
*pimemory_address = 20;
```

4.2 โครงสร้างข้อมูลที่เลือกใช้

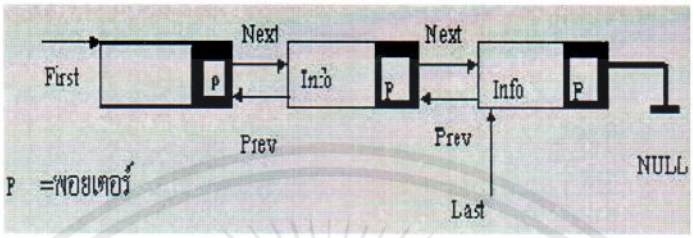
โครงสร้างข้อมูลหลักสำหรับระบบโปรแกรมนี้ใช้ลิงค์ลิสซึ่งประยุกต์การทำงานด้วยชนิดข้อมูลแบบพอยเตอร์ ซึ่งลักษณะที่นำมาประยุกต์มีดังต่อไปนี้

1. คิว(Queue) แทนกลไกการทำงานแบบคิว
2. Singly Linked List เป็นพอยเตอร์ทางเดียว ประยุกต์ใช้งานง่าย
3. Doubly Linked List เป็นพอยเตอร์สองทาง ต้องเสียที่มากกว่าแบบแรก ใช้จัดการส่วนติดต่อผู้ใช้เพราะให้ความเร็วในการค้นหาอ็อปเจค

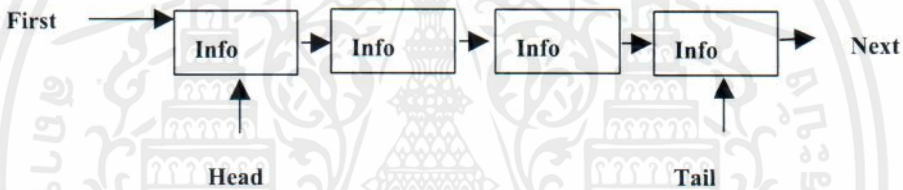
เมื่อพิจารณาพฤติกรรมของคิวลูกค้า พบว่ามีการเพิ่ม(Insert) และ ลบ(Delete) บ่อยมาก ดังการเลือก Singly Linked List มาประยุกต์ใช้จะได้ประโยชน์มากที่สุดเพราะทำงานได้รวดเร็วที่สุด



รูปที่ 4.1 แสดง Linked List ทางเดียว



รูปที่ 4.2 แสดง Linked List สองทาง



รูปที่ 4.3 แสดงโครงสร้างข้อมูลคิว

4.3 การออกแบบโครงสร้างข้อมูลสำหรับอ็อปเจค

ข้อมูลที่สนใจมีดังนี้

กรณีคิว

1. ID คือรหัสอ็อปเจค กำหนดเป็นอินทีเจอร์ขนาด 4 ไบท์
2. TYPE คือชนิดของอ็อปเจค กำหนดเป็นตัวอักษร
3. EVENT LIST คือส่วนเก็บเหตุการณ์ในระบบ โดยสนใจเวลาที่ลูกค้าเข้ามาในระบบ กำหนดการเก็บด้วยอาเรย์ของอินทิเจอร์

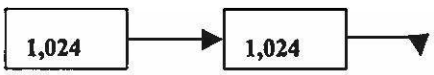
กรณีส่วนเก็บสถิติ

1. ID คือรหัสอ็อปเจค กำหนดเป็นอินทีเจอร์ขนาด 4 ไบท์
2. TYPE คือชนิดของอ็อปเจค กำหนดเป็นตัวอักษร
3. ENTRY คือค่าเวลาที่ลูกค้าเข้ามาในคิว
4. GET IN SERVICE คือเวลาที่ลูกค้าเข้ารับบริการ
5. DEPART คือเวลาที่ลูกค้าออกไปจากระบบ

การออกแบบโครงสร้างข้อมูล

กรณีคิว

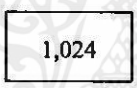
- กำหนดขนาดเริ่มต้นด้วยอาเรย์ของอินทีเจอร์ขนาด 1,024 หน่วย
  ขนาด = 4*1,024 ไบท์ = 4,096 ไบท์
- เพิ่มขนาดได้ด้วยอาเรย์ขนาด 1,024 หน่วย
  ขนาด = ขนาดปัจจุบัน + 1,024
- ประยุกต์การใช้งานด้วย Linked list ทางเดียว



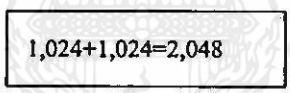
กรณีส่วนเก็บสถิติ

- กำหนดขนาดเริ่มต้นด้วยอาเรย์ของอินทีเจอร์ขนาด 1024 หน่วย
  ขนาด = 4*1,024 ไบท์ = 4,096 ไบท์
- เพิ่มขนาดได้ด้วยอาเรย์ขนาด 1,024 หน่วย
  ขนาด = ขนาดปัจจุบัน + 1,024
- ประยุกต์การใช้งานด้วยอาเรย์

ขนาดปัจจุบัน



ขนาดที่ขยาย



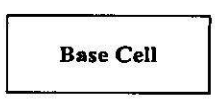
โดยการใช้ฟังก์ชัน Arraycopy ในภาษาจาวา

4.4 การออกแบบอ็อปเจค

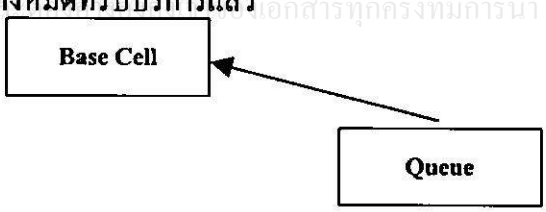
1. เครื่องสุ่มตัวเลข อธิบายด้วยคลาส Xrandom [ภาคผนวก ก. หน้า 1]
2. เซลล์พื้นฐาน อธิบายด้วยคลาส Cell

ข้อมูลสำคัญคือ

1. ID แทนรหัสเซลล์
2. TYPE แทนชนิดของเซลล์
3. ENTRY แทนจำนวนลูกค้าเข้ามาในระบบ
4. DEPART แทนจำนวนลูกค้าออกจากระบบ
5. COUNT แทนจำนวนลูกค้าทั้งหมดที่รับบริการแล้ว



3. คิว อธิบายด้วยคลาส Queue



ถ่ายทอดคุณสมบัติมาจาก Base Cell

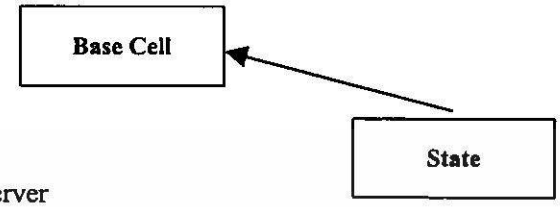
ID = 0 เพราะกำหนดให้มีคิวเดียวในเซลล์

TYPE = Queue

4. ส่วนเก็บข้อมูล อธิบายด้วยคลาส State

ID=n

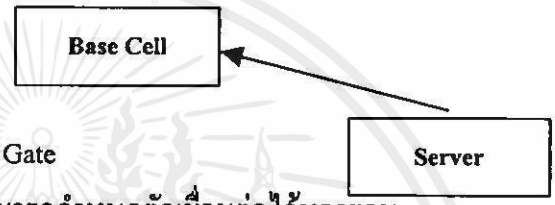
TYPE=State



5. ผู้ให้บริการ อธิบายด้วยคลาส Server

ID=n โดย n=1,2,3,...n สามารถกำหนดผู้ให้บริการได้หลายคน

TYPE=Server



6. คลาสเชื่อมต่อ อธิบายด้วยคลาส Gate

ID=n โดย n=1,2,3,...n สามารถกำหนดตัวเชื่อมต่อได้หลายคน

SRC = บอกตำแหน่งต้นทาง

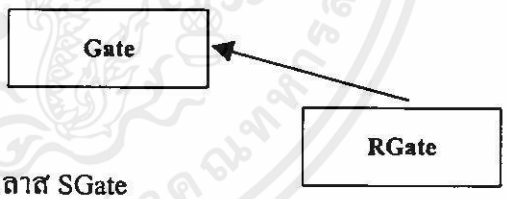
DST = บอกตำแหน่งปลายทาง

X = พิกัด x ของพื้นที่วาดกราฟบนจอภาพ

Y = พิกัด y ของพื้นที่วาดกราฟบนจอภาพ

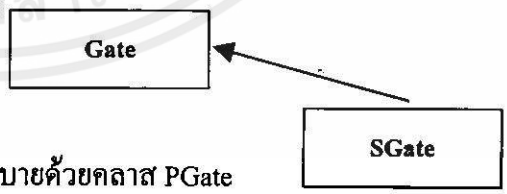
7. คลาสเชื่อมต่อแบบสุ่ม อธิบายด้วยคลาส RGate

ถ่ายทอดคุณสมบัติมาจากคลาส Gate และมีฟังก์ชันการทำงานแบบสุ่ม



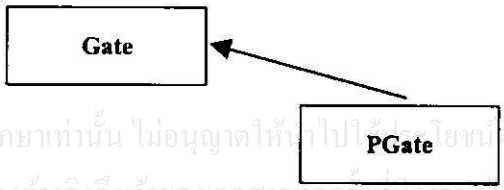
8. คลาสเชื่อมต่อแบบลำดับ อธิบายด้วยคลาส SGate

ถ่ายทอดคุณสมบัติมาจากคลาส Gate และมีฟังก์ชันเชื่อมต่อแบบลำดับ



9. คลาสเชื่อมต่อแบบความน่าจะเป็น อธิบายด้วยคลาส PGate

ถ่ายทอดคุณสมบัติมาจากคลาส Gate และมีฟังก์ชันเชื่อมต่อแบบลำดับ



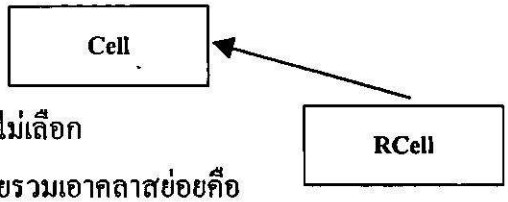
10. เซลล์ที่สามารถวิ่งได้ อธิบายด้วยคลาส RCell

X บอกพิกัด X

Y บอกพิกัด Y

INFO บอกข้อสนเทศของวัตถุ

PICK เพื่อระบุสถานะเลือกหรือไม่เลือก

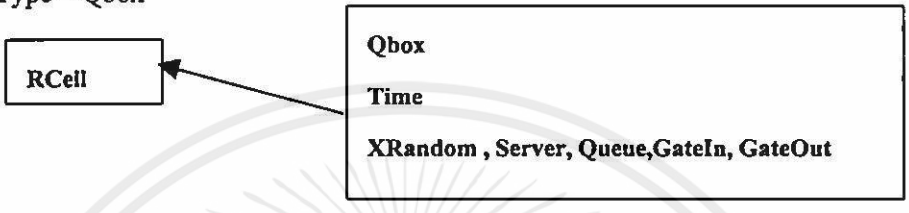


11. ระบบคิว อธิบายด้วยคลาส Qbox โดยรวมเอาคลาสย่อยคือ

Xrandom ,Server ,Queue,

Gate In ,Gate Out ,ID = n, โดย n=1,2,3,...n

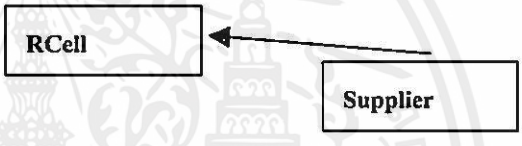
Type = Qbox



12. ส่วนผลิตลูกค้า อธิบายด้วยคลาส Supplier

ID =n โดย n=1,2,3,...,n สามารถกำหนดตำแหน่งจบได้หลายที่

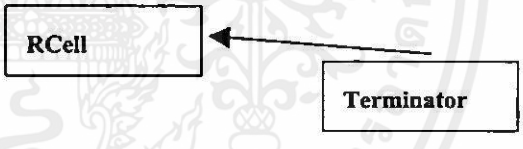
TYPE = supplier



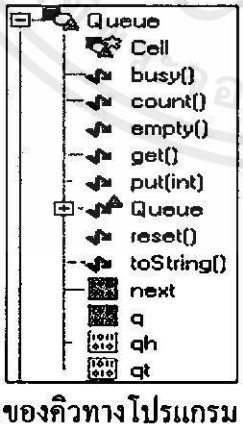
13. ตำแหน่งจบ อธิบายด้วยคลาส Terminator

ID =n

TYPE = terminator



4.5 การประยุกต์ Linked List สำหรับโครงสร้างข้อมูลของคิว

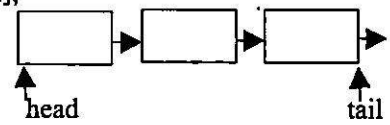


รูปที่ 4.4 แสดงโครงสร้างของคิวทางโปรแกรม

นิยามคิวด้วยขนาดที่กำหนดเป็นค่าเริ่มต้น

```
public Queue( int length ){
```

```
        super("Queue",0);   // 0 หมายถึงไม่มีการกำหนด IDให้
                                //แก่คิวเพราะกำหนดให้มีเพียงคิวเดียว
        q       = new int[length];
        reset(); }
private int     q[];
private int     qh,qt; /* queue head, queue tail */
private Queue   next; /* next pointer*/
```



ขนาดเริ่มต้นคืออาเรย์ของอินทเจอร์ขนาด 1,024 และการขยายขนาดทีละ 1024 หน่วย

ฟังก์ชัน Insert

1. อ่านข้อมูลเข้า
2. บวกเพิ่มจำนวน entry
3. ถ้าความยาวยังน้อยกว่า length ; ใส่ข้อมูลในคิว
4. ถ้าความยาวมากกว่าหรือเท่ากับ length ; สร้างคิวขึ้นใหม่โดยการขยายขนาดอาเรย์ แล้วปรับปรุงพอยเตอร์

ตัวอย่างฟังก์ชันการรับข้อมูลและมีการขยายขนาดเมื่อมีลูกค้าเข้ามากเกินความยาวของคิว

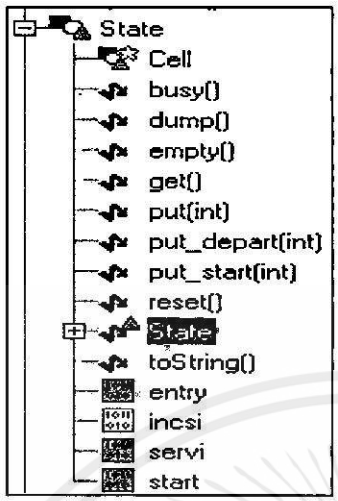
```
public void put( int x )
    {   entry++;
        if( qh<q.length ) //check length
            q[qh++]= x;
        else            // exceel length, extend queue length
        {   if( next==null )
                next    = new Queue( q.length );
            next.put( x );
        }
```

ฟังก์ชัน Delete

1. ตรวจสอบความยาวของ qt ว่าน้อยกว่า length หรือไม่ ; เพิ่มค่าให้ depart
2. กรณีที่ขนาด qt เท่ากับ length และค่า next ไม่เป็น Null ; ต้อง update link เพื่อถอยพอยเตอร์

```
    public int get() // practical chain
    {   int     r;
        if( qt<q.length )
        {   depart++; r= q[qt++];
            if( (qt==q.length)&&(next!=null) )
            {   q       = next.q;
                qh      = next.qh;
                qt      = next.qt;
                next    = next.next;
            }
            return( r );
        }
        return( -1 );       // Queue underflow
    }
```

4.6 การประยุกต์ Linked List สำหรับการเก็บข้อมูล



รูปที่ 4.5 แสดงโครงสร้างส่วนเก็บข้อมูลสถิติ

ส่วนอธิบายโครงสร้างของ State

การประยุกต์ใช้เหมือนกับกรณีของคิว

```
public State( int id, int length )
        {super("State",id);
        incsi     = length; //for increment size of state object
                  reset();
        }
```

นิยาม State ด้วย Array

```
public State( int id )
        {       this( id,1024 ); /* ขนาดเริ่มต้น 1,024 ไบต์*/
        }
private int       incsi; /* เพื่อขยายขนาด */
```

4.7 เทคนิคการขยายขนาดด้วยการคัดลอก ตามวิธีของจาวา [19] ด้วยฟังก์ชัน arraycopy

```
public void put( int entry_time )
{       if( count==entry.length )
        {       int       l = entry.length+incsi,
                          tmp[];
                tmp       = new int[l];
                System.arraycopy(tmp,0,entry,0,entry.length);
                entry     = tmp;
                tmp       = new int[l];
                System.arraycopy(tmp,0,start,0,start.length);
```

```
                start     = tmp;
                tmp       = new int[l];
                System.arraycopy(tmp,0,servi,0,servi.length);
                servi     = tmp;
            }
            entry[count]      = entry_time; //เวลาที่ลูกค้าเข้ามาในระบบ
        }
```

ฟังก์ชัน arraycopy จะคัดลอกข้อมูลจากจุดเริ่มต้นตำแหน่งใดๆถึงจุดสุดท้ายของอาเรย์ แล้วนำไปวางที่ตำแหน่งใหม่ใด ๆ

เห็นได้ว่าโครงสร้างข้อมูลในโปรแกรมได้เลือกโครงสร้างข้อมูลที่ประยุกต์ใช้งานได้ง่าย อย่างเช่นอาเรย์ คิว และลิงค์ลิสทางเดียว มาจัดการอ็อปเจคซึ่งให้ความเร็วสูงในการทำงาน และมีการออกแบบการเชื่อมต่อไว้ในแต่ละอ็อปเจคเพื่อการทำงานเป็นเครือข่ายอ็อปเจค ดังนั้นโครงสร้างข้อมูลแบบนี้จึงสามารถรองรับโครงข่ายซับซ้อนได้ และการใช้หน่วยความจำมีประสิทธิภาพอีกด้วย ประกอบกับความสามารถของภาษาจาวาในการจัดการหน่วยความจำ ทำให้การโปรแกรมทำได้สะดวก สั้น และเข้าใจง่าย ในบทต่อไปจะอธิบายถึงลักษณะการติดต่อกับอ็อปเจคเหล่านี้และอธิบายถึงการออกแบบโปรแกรมในส่วนการเชื่อมต่อผู้ใช้ซึ่งเป็นอีกกส่วนหนึ่งของโปรแกรม

บทที่ 5
ส่วนเชื่อมต่อผู้ใช้(User Interface)

5.1 การจำลองแบบผ่านเวบ(Web-based Simulation)

การจำลองแบบผ่านเวบได้รับความสนใจอย่างรวดเร็วจากการขยายตัวของเทคโนโลยี World-Wide Web และเทคโนโลยีที่เกี่ยวข้องเช่น HTML, HTTP, CGI เป็นต้น ประกอบกับความเชื่อถือในคอมพิวเตอร์ และการจำลองแบบโดยคอมพิวเตอร์ที่ช่วยแก้ปัญหาและเป็นเครื่องมือที่ดีช่วยตัดสินใจ

แอ็บเพล็ต(Applet)[8] เป็นโปรแกรมประเภทหนึ่งในภาษาจาวาที่สามารถแสดงผลผ่านเวบได้ ซึ่งผู้ใช้สามารถใช้งานได้ผ่านทางอินเตอร์เน็ต โดยภาษา HTML เป็นเครื่องมือกำหนดรายละเอียดของแอ็บเพล็ตในเวบเพจ

จากการปรากฏตัวของภาษาที่โปรแกรมผ่านเครือข่ายได้เช่น ภาษาจาวา(Java)[9] และเทคโนโลยีอ็อปเจ็คแบบกระจาย(Distributed Object Technologies) เช่นCommon Object Request Broker Architecture (CORBA) และ Object Linking and Embedding / Component Object Model (OLE/COM) ล้วนมีผลให้การจำลองแบบมีศักยภาพมากขึ้น

5.2 เครื่องมือที่นำมาช่วยในการออกแบบส่วนเชื่อมต่อผู้ใช้

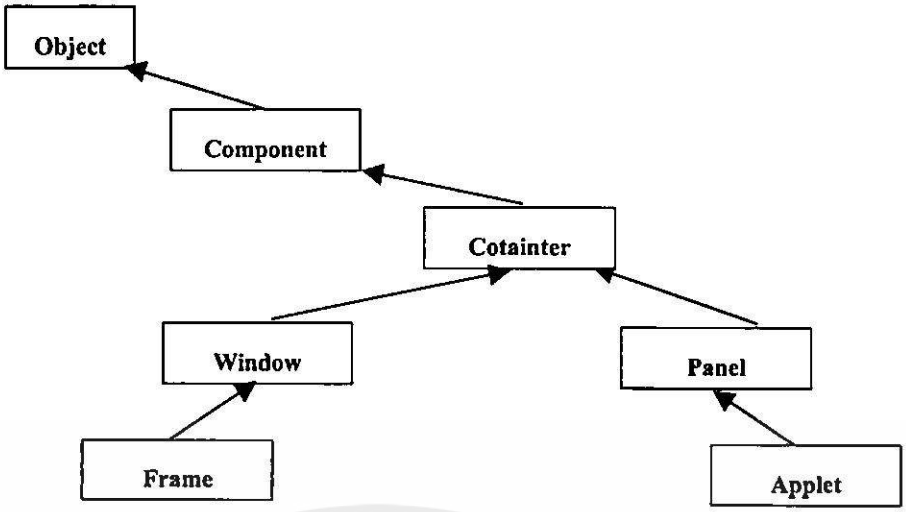
ได้นำโปรแกรมภาษาจาวามาพัฒนาระบบ แสดงผลแบบวินโดว์โดยโปรแกรมอยู่ในรูปแบบ Java Applet ทั้งนี้เพื่อให้สามารถแสดงผลผ่านเวบเบราเซอร์ได้

5.2.1 แอ็ปเพล็ต(Applet)

แอ็ปเพล็ตเป็นโปรแกรมขนาดเล็กที่สามารถเข้าถึงได้โดยผ่านทางอินเตอร์เน็ต สามารถส่งผ่านอินเตอร์เน็ตได้ การติดตั้งทำโดยอัตโนมัติและทำงานเป็นส่วนหนึ่งของเอกสารเวบ

5.2.2 โครงสร้างแอ็ปเพล็ตและตัวอย่างโปรแกรม[8][9]

แอ็บเพล็ตเป็นโปรแกรมจาวาที่ขยายการทำงานเป็นชนิดแอ็บเพล็ต ซึ่งเป็นส่วนหนึ่งของ java.applet.package โดยโครงสร้างการถ่ายทอดคุณสมบัติเป็นดังนี้



รูปที่ 5.1 แสดงผังการถ่ายทอดคุณสมบัติ

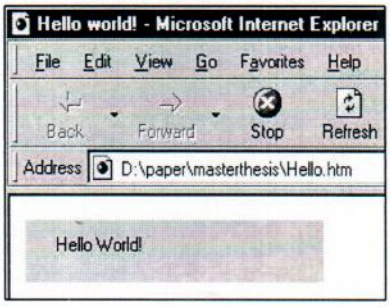
ตัวอย่างโปรแกรมแอ็บเพล็ต

```
import java.awt.*;
import java.applet.*;
public class HelloWorldApplet extends Applet{
        public void paint(graphic g){
        g.drawString("Hello World"),20,20);
}
}
```

รูปที่ 5.2 ตัวอย่างโปรแกรมแอ็ปเพล็ต

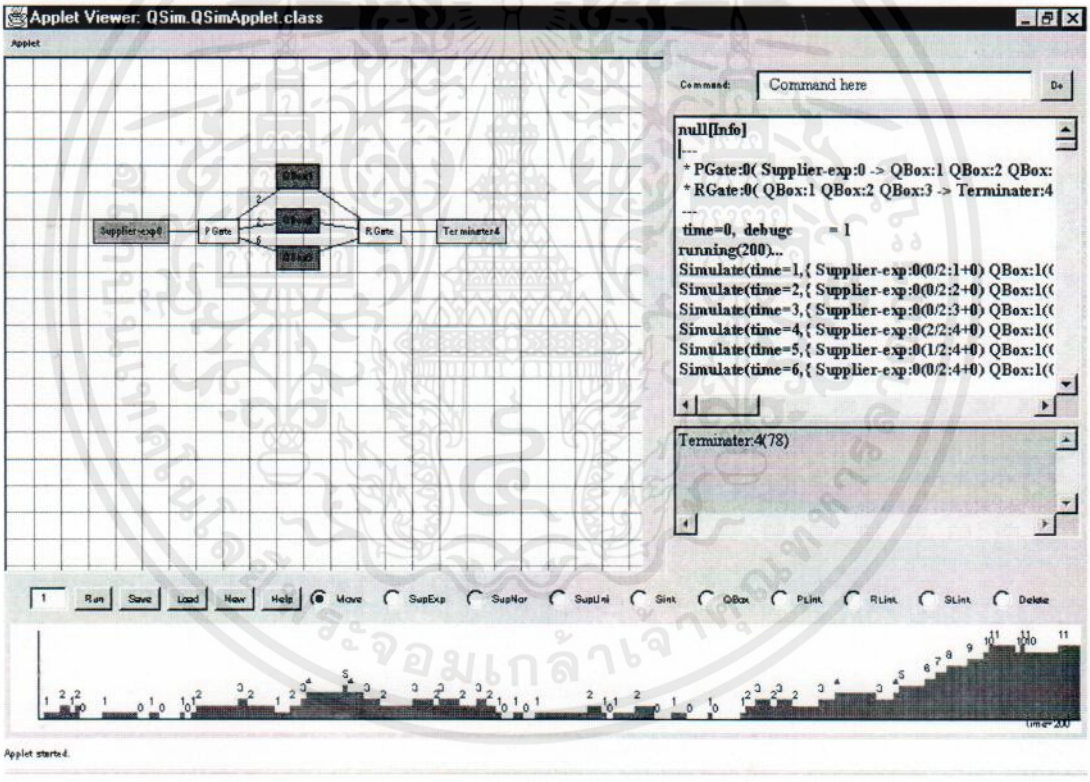
5.2.3 ขั้นตอนการสร้างแอ็ปเพล็ต

1. เขียนโปรแกรมจาวา และตั้งชื่อแฟ้มเป็น filename.java
2. คอมไพล์โปรแกรมด้วย javac filename.java
3. ดูการทำงานได้ด้วย appletviewer filename.java หรือโดยโปรแกรมเบราเซอร์ ในการเรียกใช้แอ็บเพล็ต ต้องกำหนดรายละเอียดอย่างน้อยดังนี้
   - ชื่อของคลาส
   - ตำแหน่งที่เก็บคลาส
   - ขนาดของแอ็บเพล็ตที่ต้องการให้ปรากฏบนเวบ บอกความกว้างและความสูงของแอ็บเพล็ต



รูปที่ 5.3 แสดงผลการวิ่งแอ็ปเพล็ต

5.3 การออกแบบหน้าจอแรกของโปรแกรม



รูปที่ 5.4 แสดงหน้าจอแรกของโปรแกรม

พื้นที่ในแอ็บเพล็ตประกอบด้วยส่วนสำหรับวาดภาพ ป้อนคำสั่งและส่วนแสดงผล บริเวณด้านล่างออกแบบเป็นส่วนควบคุมโดยมีบรรทัดเครื่องมือ ขนาดพื้นที่ที่กำหนดการแสดงผลคือกว้าง 640 พิกเซลและสูง 480 พิกเซล

กำหนดสัญลักษณ์สำหรับวาดภาพระบบ

Supplier-exp n	แทนอ็อปเจคที่ผลิตลูกค้าให้กับระบบด้วยค่าการกระจายเอ็กซ์โปเนนเชียล
Supplier-nor n	แทนอ็อปเจคที่ผลิตลูกค้าให้กับระบบด้วยค่าการกระจายปกติ
Supplier-uni n	แทนอ็อปเจคที่ผลิตลูกค้าให้กับระบบด้วยค่าการกระจายยูนิฟอร์ม
QBox n	แทนระบบคิว ซึ่งเป็นแบบคิวเดียวและผู้ให้บริการได้หลายคน
RGate	แทนการเชื่อมต่อแบบสุ่ม
SGate	แทนจุดการเชื่อมต่อแบบลำดับ
PGate	แทนจุดการเชื่อมต่อแบบกำหนดความน่าจะเป็น
Terminater n	แทนปลายทางของลูกค้า
n, s/t	แทนการเชื่อมโยง ซึ่งเป็นส่วนหนึ่งของการเชื่อมต่อ n แทนค่าความน่าจะเป็น , s แทนต้นทาง , t แทนปลายทาง

5.4 บรรทัดเครื่องมือ (Toolbar)

บรรทัดเครื่องมือเป็นส่วนที่ช่วยให้การทำงานรวดเร็วขึ้น โดยฟังก์ชันที่ใช้งานบ่อย ๆจะออกแบบให้ทำงานผ่านบรรทัดเครื่องมือ

ตารางที่ 5.1 แสดงรายการบรรทัดเครื่องมือ

ฟังก์ชัน	อธิบาย
Delete	ใช้ลบอ็อปเจคออกจากระบบ
Help	เพื่อขอดูข้อมูลช่วยเหลือในการใช้โปรแกรม
Load	ใช้โหลดแฟ้มข้อมูลเก่ามาวิ่งระบบ
Move	เพื่อย้ายตำแหน่งอ็อปเจคในพื้นที่แสดงผล
Save	ใช้เก็บตัวแปรของการจำลองแบบ, ไว้ในแฟ้มข้อมูล
Add Source Exp,Nor,Uni	ใช้สร้าง / เพิ่ม Source (ใช้สร้างลูกค้าให้กับระบบ)
Add Terminator	ใช้สร้าง / เพิ่ม Terminator(คือจุดสุดท้ายที่ลูกค้าวิ่งไป)

Add Qbox	ใช้สร้าง / เพิ่ม Queue Cell
Add Random Link	ใช้สร้าง / เพิ่ม Random Link
Add Sequence Link	ใช้สร้าง / เพิ่ม Sequence Link
Add Probability Link	ใช้สร้าง / เพิ่ม Probability Link

5.5 การเรียกใช้โปรแกรมโดยผ่านเวบเบราเซอร์

โปรแกรมนี้สามารถแสดงผลผ่านโปรแกรมเวบเบราเซอร์ที่สนับสนุน Java Language, Java Applet ต้องติดตั้งโปรแกรมนี้ไว้บนอินเตอร์เน็ตโฮส(Internet Host) ก่อนเรียกใช้ผ่านอินเตอร์เน็ต

ตัวอย่าง :

../qsim/qsim.class

เก็บโปรแกรมซึ่งอยู่ในรูป qsim.class ไว้ในโฟลเดอร์ชื่อ qsim

ตัวอย่างการเรียกใช้งานผ่านเบราเซอร์ :

(HTML TAG)

```
<APPLET
 CODEBASE = "."  CLASSPATH = "qsim"  CODE= qsim.qsimapplet.class"
 NAME  = "QApplet"  WIDTH  = 640  HEIGHT  = 480  HSPACE  = 0
 VSPACE  = 0  ALIGN  = middle >
<PARAM NAME = "initFile" VALUE = "test.sim">
<PARAM NAME = "editWidth" VALUE = 400>
<PARAM NAME = "editHeight" VALUE = 200>
</APPLET>
```

5.6 บรรทัดคำสั่ง(Command Line)

ตารางที่ 5.2 แสดงรายการคำสั่ง

Command Line	อธิบาย
Debugc <value> เช่น Debug 1	ใช้ระบุช่วงเวลาในการแสดงผล Trace value บอกช่วงของการ Trace
Info ไม่มีตัวแปรตาม	ใช้ดูรายการอ็อปเจคในระบบ
Locate x,y	บอกตำแหน่งการวางอ็อปเจคในบริเวณแสดง

	กราฟ x,y บอกพิกัดตำแหน่ง x,y บนหน้าจอ
Load <filename> เช่น Load test.sim	ใช้โหลดแฟ้มเพื่อวิ่งการจำลองแบบ filename แทนชื่อแฟ้มข้อมูลที่ต้องการโหลด
Prob Link < p1 p2..pn> to <t1 t2..tn > เช่น Link Probability 1 to 2	ใช้สร้างลิงค์เพื่อเชื่อต่อเซลล์เข้าเป็นโครงข่าย pn บอกตำแหน่งเซลล์ต้นทาง tn บอกตำแหน่งเซลล์ปลายทาง ทำงานแบบกำหนดความน่าจะเป็น
Qbox <s1 s2…sn> เช่น Qbox 1 2 มี 2 Server ตัวแรกมีอัตราการให้บริการเป็น 1 ตัวต่อมามีอัตราการให้บริการเป็น 2	ใช้สร้างเซลล์ระบบคิว โดย n บอกจำนวน Server ในระบบ sn บอกอัตราการให้บริการของ Server โดยเฉลี่ย
Random Link < p1 p2..pn> to <t1 t2..tn > เช่น Link Random 1 to 2 Link Random 1 to 2 3	ใช้สร้างลิงค์เพื่อเชื่อต่อเซลล์เข้าเป็นโครงข่าย pn บอกตำแหน่งเซลล์ต้นทาง tn บอกตำแหน่งเซลล์ปลายทางทำงานเป็นแบบสุ่ม
Reset	ใช้ reset ค่าเริ่มต้นให้อ็อปเจค
Run <value> เช่น Run 200	ใช้สั่งให้เริ่มการทำงาน , value แทนจำนวนหน่วยเวลาที่ต้องการวิ่งโปรแกรม
Save <filename> เช่น Save test.sim	ใช้เก็บตัวแปรของการจำลองแบบไว้ในแฟ้ม filename แทนชื่อแฟ้มที่ต้องการเก็บข้อมูล
Supplier <n> เช่น Supplier 2 หมายถึงส่งลูกค้าออกมา 2 คนโดยเฉลี่ย	ใช้สร้างเซลล์ supplier โดยในระบบสามารถมีได้หลาย supplier โดย n กำหนดจำนวนลูกค้าโดยเฉลี่ยที่สุ่มออกมาในหนึ่งหน่วยเวลา
Sequence Link < p1 p2..pn> to <t1 t2..tn > เช่น Link Sequence 1 to 2	ใช้สร้างลิงค์เพื่อเชื่อต่อเซลล์เข้าเป็นโครงข่าย pn บอกตำแหน่งเซลล์ต้นทาง tn บอกตำแหน่งเซลล์ปลายทาง ทำงานแบบมีลำดับ
Terminator ไม่มีตัวแปรตาม	ใช้สร้างเซลล์ประเภท Terminator ซึ่งบอกตำแหน่งปลายทางของลูกค้า

5.7 รูปแบบการรายงานผล

5.7.1 Tracing

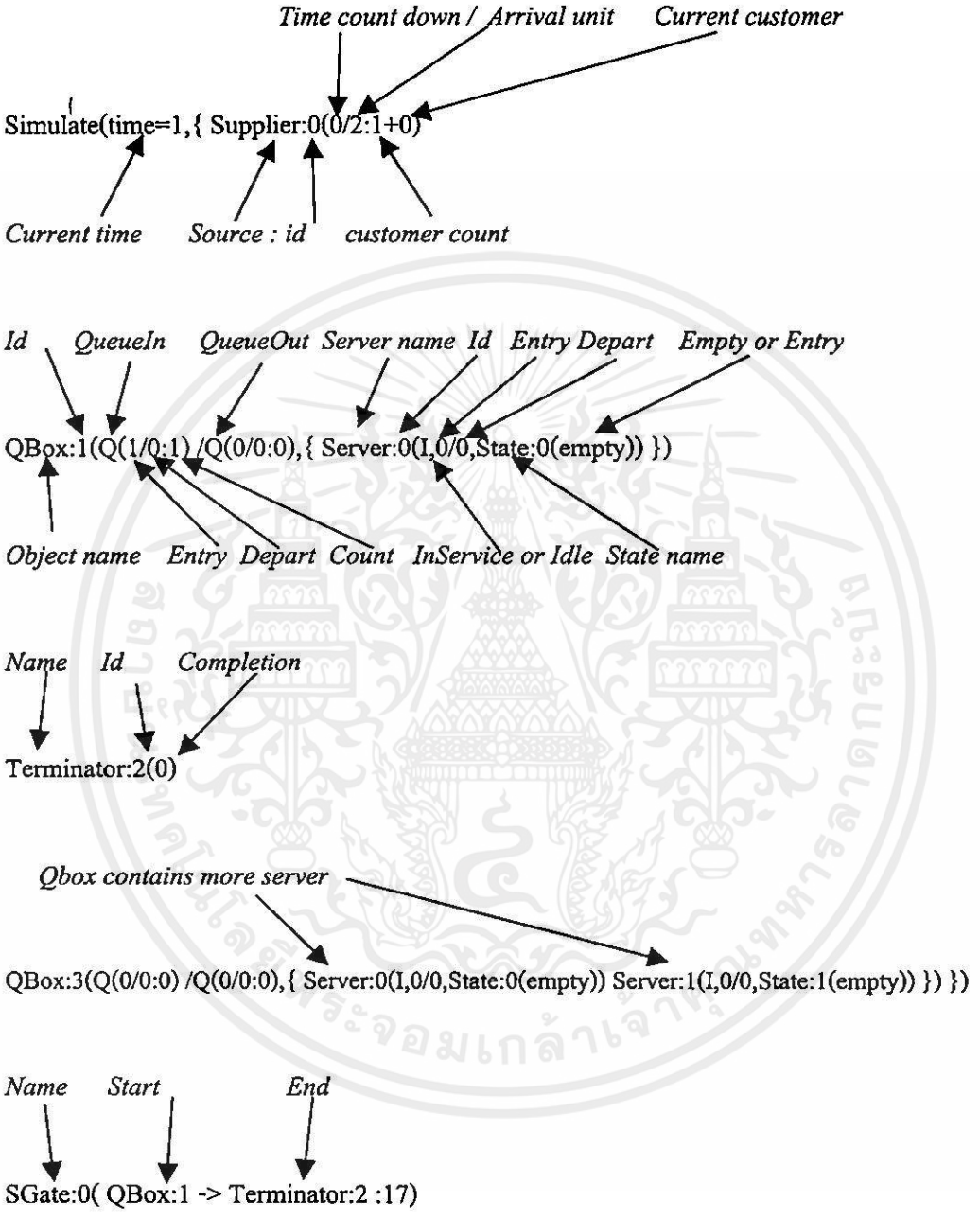
```
[Load parameters]
debugc 1
supplier 2
qbox 1
terminator
link sequence 0 to 1
link sequence 1 to 2
info
run 20
info
throughput
total-busy-time
mean-service-time
utilisation
resident-time-sum
mean-resident-time
mean-queue-time
mean-number-in-system
mean-number-in-queue

 * SGate:0( Supplier:0 -> QBox:1 :0)
 * SGate:0( QBox:1 -> Terminator:2 :0)

 time=0,  debugc       = 1
running(20)...
Simulate(time=1,{ Supplier:0(0/2:1+0) QBox:1(Q(1/0:1) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) Terminator:2(0) })
Simulate(time=2,{ Supplier:0(2/2:2+0) QBox:1(Q(2/1:1) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) Terminator:2(1) })
Simulate(time=3,{ Supplier:0(1/2:2+0) QBox:1(Q(2/2:0) /Q(2/2:0),{ Server:0(I,2/2,State:0(entried=2)) }) Terminator:2(2) })
Simulate(time=4,{ Supplier:0(0/2:2+0) QBox:1(Q(2/2:0) /Q(2/2:0),{ Server:0(I,2/2,State:0(entried=2)) }) Terminator:2(2) })
Simulate(time=17,{ Supplier:0(0/2:5+0) QBox:1(Q(5/5:0) /Q(5/5:0),{ Server:0(I,5/5,State:0(entried=5)) }) Terminator:2(5) })
...
```

รูปที่ 5.5 แสดงตัวอย่าง Trace จากโปรแกรม

เป็นการรายงานข้อมูลขณะที่โปรแกรมกำลังทำงานอยู่ โดยสามารถติดตามการทำงานในแต่ละเวลาได้ รูปแบบการรายงาน Trace เป็นดังนี้



5.7.2 รายงานทั่วไป

```
---
 * SGate:0( Supplier:0 -> QBox:1 :6)
 * SGate:0( QBox:1 -> Terminator:2 :6)
----
 time=20,  debugc    = 1
Throughput:
QBox:1=0.3
```

busytime:
QBox:1=3.0
Mean Service Time:
QBox:1=0.5
Server Utilisation:
QBox:1=0.15
Residence Time Sum:
QBox:1=11.0
Meam Residence Time:
QBox:1=1.8333334
Meam Queue Time:
QBox:1=1.3333334
Mean Number In System:
QBox:1=0.55
Mean Number In Queue:
QBox:1=0.4

รูปที่ 5.6 แสดงรายงานสรุป

เป็นรายงานสรุปผลเมื่อวิ่งโปรแกรมเสร็จแล้ว โดยโปรแกรมจะสรุปตัวแปรที่ใช้ในการวิเคราะห์คำตอบที่ต้องการอย่างสมบูรณ์

5.7.3 กราฟรายงาน

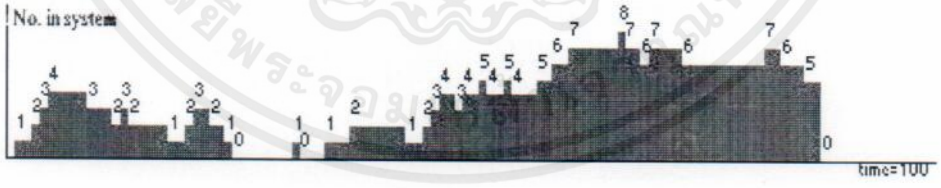
ได้ออกแบบโปรแกรมส่วนการสร้างกราฟด้วยเพื่อให้เห็นภาพการทำงานได้ง่ายและการสรุปจากกราฟสามารถคำนวณได้อย่างรวดเร็วด้วยอัลกอริธึมที่กล่าวไว้ในบทที่ 2

ตัวอย่างกราฟ

อัตราการมาถึงเฉลี่ย = 2

อัตราการบริการเฉลี่ย = 2

1 คิว 1 ผู้ให้บริการ



รูปที่ 5.7 แสดงกราฟพฤติกรรมของระบบ M/M/1

ส่วนติดต่อผู้ใช้เป็นส่วนสำคัญของโปรแกรมซึ่งให้ผู้ใช้สามารถเข้าใจการทำงานและสะดวกต่อการทำงาน โปรแกรมนี้ได้สื่อโครงสร้างคิวด้วยภาพ ผู้ใช้สามารถออกแบบระบบได้จากหน้าจอและแก้ไขตัวแปรได้ด้วยตนเอง นอกจากนี้ยังมีส่วนรายงานผลที่สามารถสรุปปัญหาได้พร้อมด้วยกราฟแสดงพฤติกรรมของระบบ ผู้ใช้สามารถบันทึกตัวแปรของระบบที่ออกแบบไว้ในแฟ้มข้อมูลได้ นอกจากนี้การใช้โปรแกรมได้กำหนดให้สามารถแสดงผลบนเอกสารอินเตอร์เน็ต ซึ่งเป็นการเผยแพร่ข้อมูลได้อีกทางหนึ่ง

บทที่ 6
ตัวอย่างการประยุกต์ใช้งาน

6.1 การจำลองแบบการให้บริการแก่ลูกค้าในธนาคาร

ตัวอย่างนี้เป็นการจำลองการให้บริการแก่ลูกค้าในธนาคารทั่วไป ลูกค้าจะเข้าคิวรับบริการของธนาคาร เมื่อสิ้นสุดเวลาการการให้บริการ โปรแกรมจะประเมินประสิทธิภาพออกมาเป็นผลลัพธ์

6.1.1 ข้อกำหนดของการจำลองแบบ

ผู้ใช้จะต้องป้อนจำนวนเวลาในการทำงานให้แก่โปรแกรม กำหนดให้หน่วยเป็นนาที

โดยสมมติว่า ลูกค้ามาถึงธนาคารตามฟังก์ชันความน่าจะเป็นแบบเอ็กซ์โปเนนเชียล (Exponential Arrival Process) โดยฟังก์ชันนี้จะมีตัวแปรที่ผู้ใช้จะต้องป้อนให้แก่อ็อปเจคสุ่มลูกค้า (Supplier) ได้แก่เวลาเฉลี่ยที่ลูกค้ามาห่างกัน (Average Inter-arrival time) โดยป้อนที่ตำแหน่งบรรทัดคำสั่ง เช่น "Supplier-exp 2" หมายถึงลูกค้าเดินทางมาด้วยอัตรานาทีละ 2 คนโดยเฉลี่ย

และสมมติว่าพนักงานบริการจะใช้เวลาในการให้บริการแก่ลูกค้าตามฟังก์ชันความน่าจะเป็นแบบเอ็กซ์โปเนนเชียลเช่นกัน ผู้ใช้ต้องป้อนเวลาให้บริการเฉลี่ยสำหรับพนักงานทุกคน โดยจำนวนพนักงานบริการได้กำหนดไว้อ็อปเจค Qbox ซึ่งการป้อนตัวแปรทำได้ดังนี้

คำสั่ง "Qbox 2" หมายถึง กำหนดให้มีผู้ให้บริการ 1 คน ให้บริการด้วยความเร็วเฉลี่ย 2 คนต่อนาที

ตัวอย่างนี้จะจำลองเป็นระบบหลายคิวและมีผู้ให้บริการหลายคน ถ้ามีพนักงานบริการว่างและลูกค้าเข้ามาใช้บริการ ลูกค้าจะได้รับบริการทันที โดยจะเลือกพนักงานบริการโดยการสุ่มจากพนักงานที่ว่างอยู่

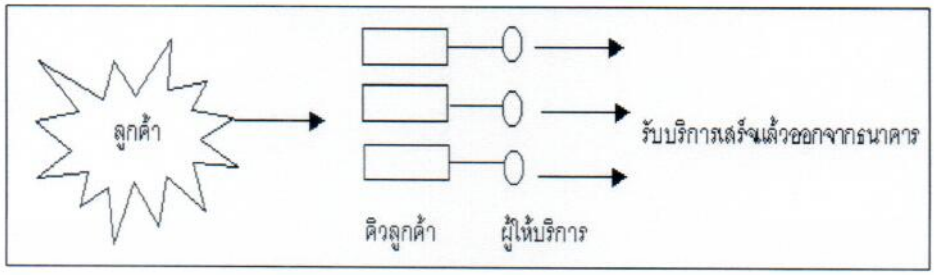
ในสายตาของธนาคาร ประสิทธิภาพของระบบนี้ขึ้นอยู่กับขนาดเวลาเฉลี่ยในคิว และอีกสิ่งหนึ่งที่ธนาคารสนใจก็คือ อัตราส่วนระหว่างเวลาที่พนักงานบริการให้บริการแก่ลูกค้าต่อเวลาที่ทำงานทั้งหมด ในแง่ของลูกค้า ประสิทธิภาพของระบบจะดีถ้าลูกค้าเสียเวลาในคิวน้อย

ผลลัพธ์ที่ต้องการมีดังนี้

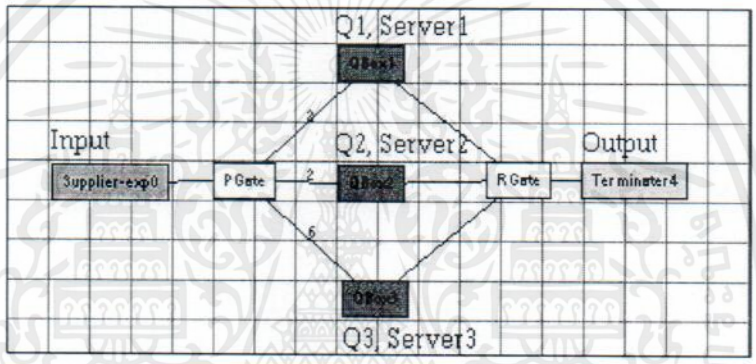
1. เปอร์เซ็นต์การทำงานของพนักงานบริการ
2. จำนวนลูกค้าที่พนักงานแต่ละคนให้บริการ
3. ขนาดเวลาเฉลี่ยของแต่ละคิว
4. ขนาดสูงสุดของแต่ละคิว
5. เวลาเฉลี่ยที่ลูกค้าต้องรอการให้บริการในแต่ละคิว

6. เวลาเฉลี่ยที่ลูกค้าต้องรอสำหรับลูกค้าทุกคน

6.1.2 ภาพอธิบายระบบ



รูปที่ 6.1 แสดงโมเดลธรรมชาติของธนาคาร



รูปที่ 6.2 แสดงภาพระบบคิวธนาคาร โดยโปแกรม

supplier-exp0 ส่งลูกค้าออกไปในระบบ ด้วยค่าเฉลี่ย 2 คนหน่วยนาที

-ส่งไปยังคิวที่ 1 ด้วยความน่าจะเป็น 0.2

-ส่งไปยังคิวที่ 2 ด้วยความน่าจะเป็น 0.2

-ส่งไปยังคิวที่ 3 ด้วยความน่าจะเป็น 0.6

Server 1 ให้บริการด้วยอัตราเฉลี่ย 1 คนต่อหน่วยนาที

Server 2 ให้บริการด้วยอัตราเฉลี่ย 2 คนต่อหน่วยนาที

Server 3 ให้บริการด้วยอัตราเฉลี่ย 3 คนต่อหน่วยนาที

เวลาจำลองแบบทั้งหมด 250 นาที

6.1.3 ตัวอย่างแฟ้มข้อมูลอินพุท

```
debugc 1
supplier-exp 2
qbox 1
qbox 2
qbox 3
terminator
link probability 0 to 1 2 3 with 2 2 6
link random 1 2 3 to 4
run 200
info
throughput
total-busy-time
arrival-rate
mean-service-time
utilisation
resident-time-sum
mean-resident-time
mean-queue-time
mean-number-in-system
mean-number-in-queue
```

6.1.4 ผลการวิ่งระบบ

```
|Info|
 * PGate:0( Supplier-exp:0 -> QBox:1 QBox:2 QBox:3  with  2 2 6 :0)
 * RGate:0( QBox:1 QBox:2 QBox:3 -> Terminator:4  with  0 :0)
 time=0,  debugc = 1
running(200)...
Simulate(time=1,{ Supplier-exp:0(0/2:1+0) QBox:1(Q(0/0:0) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) QBox:2(Q(0/0:0) /Q(0/0:0),
{ Server:0(I,0/0,State:0(empty)) }) QBox:3(Q(1/0:1) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) Terminator:4(9) })
Simulate(time=3,{ Supplier-exp:0(0/2:3+0) QBox:1(Q(0/0:0) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) QBox:2(Q(1/1:0) /Q(1/0:1),
{ Server:0(I,1/1,State:0(entried=1)) }) QBox:3(Q(2/1:1) /Q(1/0:1),{ Server:0(I,1/1,State:0(entried=1)) }) Terminator:4(0) })
Simulate(time=5,{ Supplier-exp:0(1/2:4+0) QBox:1(Q(0/0:0) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) QBox:2(Q(1/1:0) /Q(1/1:0),
{ Server:0(I,1/1,State:0(entried=1)) }) QBox:3(Q(3/2:1) /Q(1/1:0),{ Server:0(S,2/1,State:0(entried=1)) }) Terminator:4(2) })
Simulate(time=7,{ Supplier-exp:0(0/2:5+0) QBox:1(Q(0/0:0) /Q(0/0:0),{ Server:0(I,0/0,State:0(empty)) }) QBox:2(Q(1/1:0) /Q(1/1:0),
{ Server:0(I,1/1,State:0(entried=1)) }) QBox:3(Q(4/3:1) /Q(3/2:1),{ Server:0(I,3/3,State:0(entried=3)) }) Terminator:4(3) })
Simulate(time=9,{ Supplier-exp:0(2/2:7+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(2/1:1) /Q
(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:3(Q(4/4:0) /Q(4/4:0),{ Server:0(I,4/4,State:0(entried=4)) }) Terminator:4(6) })
Simulate(time=21,{ Supplier-exp:0(5/2:11+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(3/3:0) /Q
,(3/3:0),{ Server:0(I,3/3,State:0(entried=3)) }) QBox:3(Q(7/6:1) /Q(6/6:0),{ Server:0(I,6/6,State:0(entried=6)) }) Terminator:4(10) })
Simulate(time=23,{ Supplier-exp:0(3/2:11+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(3/3:0) /Q
(3/3:0),{ Server:0(I,3/3,State:0(entried=3)) }) QBox:3(Q(7/7:0) /Q(7/7:0),{ Server:0(I,7/7,State:0(entried=7)) }) Terminator:4(11) })
Simulate(time=25,{ Supplier-exp:0(1/2:11+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(3/3:0) /Q
(3/3:0),{ Server:0(I,3/3,State:0(entried=3)) }) QBox:3(Q(7/7:0) /Q(7/7:0),{ Server:0(I,7/7,State:0(entried=7)) }) Terminator:4(11) })
Simulate(time=27,{ Supplier-exp:0(1/2:12+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(3/3:0) /Q
(3/3:0),{ Server:0(I,3/3,State:0(entried=3)) }) QBox:3(Q(8/7:1) /Q(7/7:0),{ Server:0(I,7/7,State:0(entried=7)) }) Terminator:4(11) })
Simulate(time=29,{ Supplier-exp:0(0/2:13+0) QBox:1(Q(1/1:0) /Q(1/1:0),{ Server:0(I,1/1,State:0(entried=1)) }) QBox:2(Q(3/3:0) /Q
(3/3:0),{ Server:0(I,3/3,State:0(entried=3)) }) QBox:3(Q(9/8:1) /Q(8/8:0),{ Server:0(I,8/8,State:0(entried=8)) }) Terminator:4(12) })
Simulate(time=50,{ Supplier-exp:0(0/2:21+0) QBox:1(Q(2/2:0) /Q(2/2:0),{ Server:0(I,2/2,State:0(entried=2)) }) QBox:2(Q(4/4:0) /Q
(4/4:0),{ Server:0(I,4/4,State:0(entried=4)) }) QBox:3(Q(15/13:2) /Q(12/12:0),{ Server:0(S,13/12,State:0(entried=12)) }) Terminator:4
(18) })
Simulate(time=100,{ Supplier-exp:0(1/2:42+0) QBox:1(Q(7/7:0) /Q(7/7:0),{ Server:0(I,7/7,State:0(entried=7)) }) QBox:2(Q(11/10:1)
```

```
/Q(10/8:2),{ Server:0(I,10/10,State:0(entried=10)) }) QBox:3(Q(24/24:0) /Q(23/23:0),{ Server:0(S,24/23,State:0(entried=23)) })
Terminator:4(38) })
Simulate(time=150,{ Supplier-exp:0(0/2:71+0) QBox:1(Q(13/13:0) /Q(13/13:0),{ Server:0(I,13/13,State:0(entried=13)) }) QBox:2(Q
(22/21:1) /Q(21/21:0),{ Server:0(I,21/21,State:0(entried=21)) }) QBox:3(Q(36/34:2) /Q(33/33:0),{ Server:0(S,34/33,State:0
(entried=33)) }) Terminator:4(67) })
Simulate(time=200,{ Supplier-exp:0(0/2:89+0) QBox:1(Q(17/17:0) /Q(17/17:0),{ Server:0(I,17/17,State:0(entried=17)) }) QBox:2(Q
(23/23:0) /Q(23/23:0),{ Server:0(I,23/23,State:0(entried=23)) }) QBox:3(Q(49/40:9) /Q(39/38:1),{ Server:0(S,40/39,State:0
(entried=39)) }) Terminator:4(78) })

|Info|
 * PGate:0( Supplier-exp:0 -> QBox:1 QBox:2 QBox:3  with  2 2 6 :89)
 * RGate:0( QBox:1 QBox:2 QBox:3 -> Terminator:4  with  0 :78)
 time=200,  debugc          = 1
Throughput[X]:
server0completion:17.0 | X=0.085 , QBox:1=0.085
server0completion:23.0 | X=0.115 , QBox:2=0.115
server0completion:39.0 | X=0.195 , QBox:3=0.19

busytime [B]:
server0:10.0, QBox:1=10.0
server0:35.0, QBox:2=35.0
server0:128.0, QBox:3=128.0

Mean Service Time[Ts]:
Ts/server0:0.5882353, QBox:1=0.588
Ts/server0:1.5217391, QBox:2=1.521
Ts/server0:3.2820513, QBox:3=3.368

Server Utilisation[U]:
U/server0:0.05, QBox:1=0.05
U/server0:0.175, QBox:2=0.175
U/server0:0.64, QBox:3=0.64

Residence Time Sum[EWI]:
QBox:1=28.0
QBox:2=93.0
QBox:3=394.0

Meam Residence Time[W]:
QBox:1=1.647
QBox:2=4.043
QBox:3=10.368

Meam Queue Time[Wq]:
QBox:1=1.058
QBox:2=2.52
QBox:3=6.99

Mean Number In System[L]:
QBox:1=0.14
```

```
QBox:2=0.46
QBox:3=1.97

Mean Number In Queue[Lq]:
QBox:1=0.09
QBox:2=0.29
QBox:3=1.33
```

6.2 สรุป

คำตอบที่น่าสนใจจากระบบมีดังนี้

1. เปอร์เซ็นต์การทำงานของพนักงานบริการ

Server Utilisation[U]:

U/server0:0.05, QBox:1=0.05 หรือ 5 เปอร์เซ็นต์

U/server0:0.175, QBox:2=0.175 หรือ 17.5 เปอร์เซ็นต์

U/server0:0.64, QBox:3=0.64 หรือ 64 เปอร์เซ็นต์

2. จำนวนลูกค้าที่พนักงานแต่ละคนให้บริการ

Server 1 ให้บริการได้ 17 คน

Server 2 ให้บริการได้ 23 คน

Server 3 ให้บริการได้ 38 คน

รวมทั้งสิ้น 78 คน

3. ขนาดเวลาเฉลี่ยของแต่ละคิว

Mean Number In Queue[Lq]:

QBox:1=0.09

QBox:2=0.29

QBox:3=1.33

4. ขนาดสูงสุดของแต่ละคิว

[Queue Max]:

QBox:1 , Qin =17

QBox:2 , Qin =23

QBox:3, Qin =49

5. เวลาเฉลี่ยที่ลูกค้าต้องรอการให้บริการในแต่ละคิว

Meam Queue Time[Wq]:

QBox:1=1.058

QBox:2=2.52

QBox:3=6.99

6. เวลาเฉลี่ยที่ลูกค้าต้องรอสำหรับลูกค้าทุกคน

System Meam Queue Time[Wq]:

= 3.522

ระบบนี้ได้ออกแบบให้แต่ละ Qbox มีผู้ให้บริการ 1 คนและมี 1 คิว ซึ่งเป็นโมเดลที่ง่ายที่สุดของระบบคิว (M/M1)

ในการออกแบบระบบ ต้องมีการกำหนดโครงสร้างส่วนที่เป็นคิวและส่วนผู้ให้บริการออกมาให้ชัดเจน แล้วจึงออกแบบการเชื่อมต่อเป็นโครงข่าย

ในบทนี้ได้นำเสนอตัวอย่างการประยุกต์ใช้โปรแกรมเพื่อจำลองการทำงานของระบบคิวในธนาคาร ซึ่งเป็นระบบคิวที่เห็นได้อยู่บ่อย ๆ จะเห็นได้ว่าการปรับเปลี่ยนตัวแปรอินพุทให้กับโปรแกรม ระบบจะสะท้อนภาพให้เห็นสภาพการทำงาน ซึ่งข้อมูลนี้สามารถนำไปเป็นเครื่องมือช่วยในการจัดการระบบหรือออกแบบระบบใช้งานจริงได้

บทที่ 7
สรุปผลการวิจัยและข้อเสนอแนะ

โปรแกรมนี้สนับสนุนการศึกษาปัญหาในระบบโครงข่ายคิว จากการทดสอบโมเดลและโปรแกรมด้วยชุดตัวแปรต่าง ๆภายใต้การทำงานแบบโครงข่ายคิว โปรแกรมสามารถให้คำตอบได้อย่างถูกต้องซึ่งวัดตามวิธีคำนวณค่าเฉลี่ยตัวแปรที่สนใจว่าตกอยู่ภายในช่วงความมั่นใจที่กำหนดตามโมเดลการวิเคราะห์ผลลัพธ์

การพัฒนาโมเดลและโปรแกรมจำลองแบบนี้ มีลำดับขั้นตอนที่ง่ายและเป็นธรรมชาติโดยทำการศึกษาและออกแบบองค์ประกอบแต่ละส่วนอย่างละเอียดแล้วค่อยนำมาประกอบกันเป็นระบบรวมตามแบบวิธีการโปรแกรมแบบสนใจวัตถุ โดยนำเอาภาษาจาวามาเป็นเครื่องมือช่วยสร้างโปรแกรม ในส่วนสำคัญเช่นโครงสร้างข้อมูลได้นำเอาโครงสร้างแบบพื้นฐานมาใช้คือ โครงสร้างข้อมูลคิว อาเรย์และลิงค์ลิส ซึ่งการจัดการโครงสร้างข้อมูลสามแบบที่กล่าวมาทำได้ง่ายในภาษาจาวา ช่วยให้โปรแกรมสั้น เข้าใจง่ายและรองรับการออกแบบที่ซับซ้อนได้เป็นอย่างดี การบำรุงรักษาโปรแกรมทำได้ง่าย ด้วยโครงสร้างที่ดีนี้จึงช่วยให้การพัฒนาโปรแกรมนี้ในอนาคตทำได้สะดวกรวดเร็ว

โปรแกรมจำลองแบบนี้ผู้ใช้สามารถเรียนรู้ระบบคิวและโครงข่ายคิวได้จากการใช้กราฟฟิกแบบวินโดว์ที่สื่อความหมายและเข้าใจได้ง่าย ผู้ใช้สามารถใช้เมาส์สร้างระบบคิวหรือโครงข่ายคิวได้ด้วยตนเองจากหน้าจอภาพ สามารถหาผลลัพธ์จากการจำลองแบบและนำผลลัพธ์นี้ไปศึกษาและวิเคราะห์ระบบที่ซับซ้อนที่มีคิวเป็นองค์ประกอบได้ โปรแกรมนี้ได้ออกแบบส่วนรายงานผลไว้ทุกรูปแบบคือ ข้อมูลแสดงการทำงานของระบบตามลำดับเวลา ส่วนแสดงข้อมูลปัจจุบันของอ็อบเจค ส่วนรายงานสรุป และรายงานด้วยกราฟ ซึ่งทุกส่วนช่วยให้การสรุปคำตอบได้ดี อนึ่งโปรแกรมนี้ได้แปลงรูปแบบให้เป็นจาวาแอ็บเพล็ต โดยสามารถนำไปประกอบเอกสารอินเตอร์เน็ตได้ ช่วยให้การเผยแพร่โปรแกรมได้สะดวก ผู้สนใจสามารถเรียนรู้ได้ทางอินเตอร์เน็ต

โปรแกรมนี้รองรับการจำลองแบบระบบคิวและโครงข่ายคิวขนาดใหญ่ได้ดี มีการจัดการการใช้หน่วยความจำที่ดี โปรแกรมขนาดเล็ก ทำงานได้เร็ว การแก้ไขหรือดูแลรักษาโปรแกรมทำได้สะดวก ซึ่งทั้งหมดนี้จะช่วยลดค่าใช้จ่ายและช่วยตัดสินใจในการพัฒนาระบบได้อย่างมีประสิทธิภาพ สุดท้ายได้ช่วยลดผลกระทบจากลิขสิทธ์ซอฟต์แวร์ที่ทำให้ราคาซอฟต์แวร์ราคาแพง ช่วยให้คนไทยสามารถพึ่งตนเองทางซอฟต์แวร์ได้

ข้อเสนอแนะ

1. เนื่องจากโปรแกรมนี้ถูกแปลงให้อยู่ในรูปแบบจาวาแอ็ปเพล็ตและต้องการให้มีขนาดเล็ก ดังนั้น รูปแบบการใช้งานจึงต้องพัฒนาให้ง่ายที่สุด โดยจำเป็นต้องตัดบางส่วนออกไปเช่น เมนู บล็อกโต้ตอบและส่วนการพิมพ์ อย่างไรก็ตามถ้าแปลงโปรแกรมนี้ไปเป็นโปรแกรมประยุกต์บนระบบวินโดว์ ก็สามารถนำองค์ประกอบเหล่านั้นมาเพิ่มเข้าไปได้ ซึ่งจะทำให้โปรแกรมมีความสมบูรณ์
2. การวาดอ็อปเจคในวินโดว์ ผู้ใช้ต้องเลือกประเภทอ็อปเจคจากบรรทัดเครื่องมือก่อน แล้วจึงนำเมาส์ไปคลิ๊กตรงบริเวณที่ต้องการวางอ็อปเจคนั้น ๆ ขณะเดียวกันที่บริเวณบรรทัดคำสั่งจะปรากฏรายการตัวแปรที่ต้องป้อนให้กับอ็อปเจคนั้น ซึ่งชุดตัวแปรแสดงรูปแบบตัวอย่างเพื่อให้ป้อนตัวแปรได้อย่างถูกต้อง หลังจากนั้นให้ผู้ใช้คลิ๊กปุ่ม DO เพื่อวางอ็อปเจค
3. การวางอ็อปเจคบนพื้นที่ทำงานสามารถย้ายไปวางให้มีรูปแบบตามที่ต้องการได้ด้วยฟังก์ชัน Move ผู้ใช้ไม่สามารถปรับขนาดของอ็อปเจคได้ ทั้งนี้เพื่อให้เหมาะกับพื้นที่ทำงานของแอ็ป-เพล็ต ถ้าต้องการให้การวาดอ็อปเจคให้มีคุณภาพเหมือนโปรแกรมประเภทกราฟฟิคทั่วไป ต้องปรับปรุงรูทีนการวาดหน้าจอ (Paint)ให้สนับสนุนการวาดด้วยเมาส์ตามเทคนิค Drag-Drop และปรับให้อ็อปเจคเปลี่ยนขนาดได้
4. การลบอ็อปเจคจะต้องเลือกฟังก์ชันจากบรรทัดเครื่องมือ ต้องลบเส้นเชื่อมโยงต้องก่อนลบอ็อปเจคสี่เหลี่ยม
5. การวาดเส้นเชื่อมโยงจะต้องมีอ็อปเจคต้นทางและปลายทาง ถ้าโครงสร้างไม่ครบ โปรแกรมจะแจ้งข้อผิดพลาดให้ทราบและไม่มีการวาดอ็อปเจคนั้น
6. โปรแกรมได้กำหนดฟังก์ชันการกระจายให้กับผู้ให้บริการเพียงชนิดเดียวคือเอ็กซ์โปเนนเชียล
7. เพื่อให้การทำงานเป็นธรรมชาติมากขึ้นและการจำลองแบบที่ละเอียดยิ่งขึ้น ควรมีการออกแบบกราฟฟิครองรับเหตุการณ์ต่าง ๆบนหน้าจอ ผู้ใช้จะได้เห็นการทำงานที่ชัดเจนเหมือนจริง เกี่ยวกับเวลาของการจำลองแบบ ต้องออกแบบให้เป็นเวลาจริง(Real Time) ซึ่งจะช่วยให้ระบบสมเหตุสมผลมากขึ้น และสามารถพัฒนาไปสู่การออกแบบระบบใช้งานจริงได้

หนังสืออ้างอิง

[1] Chang, Yih-Long. 1991. Quantitative Systems for Operations Management Version 2.0. Englewood Cliffs, New Jersey 07632 : Prentice-Hall International, Inc.

[2] Charles, H. Sauer. and K. Mani, Chandy. Computer Systems Performance Modeling. Englewood Cliffs, New Jersey : Prentice-Hall International, Inc.

[3] Kleinrock, Leonard. Queueing Systems volumn I : Theory. : A wiley Interscience Publication.

[4] Kleinrock, Leonard. Queueing Systems volumn II : Applications. : A wiley Interscience Publication.

[5] Kobayashi, Hisashi. 1978. Modeling and Analysis An Introduction to System Performance Evaluation Methodology. : Addison-Wesley Publishing Company, Inc.

[6] MacDougall, M.H.. Simulating Computer Systems Techniques and Tools. Massachusetts : The MIT Press.

[7] Robert , L. Kruse et. al. 1991. Data structure and program design in C. Englewood cliffs : Printice Hall, Inc.

[8] S. Horstmann, Cay and Cornell, Gary. 1997. Core JAVA volumeI Andvanced Features. Mountain View : Sun Microsystem Press, A Prentice Hall Title.

[9] S. Horstmann, Cay and Cornell, Gary. 1997. Core JAVA volumeII Andvanced Features. Mountain View : Sun Microsystem Press, A Prentice Hall Title.

[10] Stroustrup, Bjarne. 1995. The C++ Programming Language. Second Edition. Murry Hill, New Jersey : Addison-Wesley Publishing Company, Inc.

ภาคผนวก

โปรแกรม

```
//Title:        Queueing Simulation Program
//Version:      1.0.0
//Copyright:    Copyright (c) 1999
//Author:       Boonyong Kaewbuddee
//Company:      KMITL
//Description: This program is a part of master thesis.
//File          QsimControl.java

package QSim;
import java.lang.Math;
import java.util.Random;
import java.util.StringTokenizer;
import java.io.*;
import java.awt.*;
//---------------------------------------------------------------------------------------------
// Random generator
class   XRandom
{       public XRandom()
        {       r       = new Random();
        }
 /*'exponential' returns a psedo-random variate from a negative exponential*/
 /* distribution with mean x */
        public int get_exp( int sv_time )
        {       Integer i       = new Integer(sv_time);
                double  d;
                while( (d=r.nextDouble())==0.0 );
                Double  dd      = new Double( (- i.doubleValue()) * Math.log( d ) );
                return  dd.intValue();
        }
 /*'normal' returns a psedo-random variate from a mormal distribution */
 /* with mean x and standard deviation s */
 public double get_nor( int x,int s)
        {
        double z1,z2=0.0;
        double  v1,v2,w;
        Double  i       = new Double(x);
                double  d;
          if(z2!=0.0){
            z1=z2;z2=0.0;
          }
          else{
          do{
            while( (d=r.nextDouble())==0.0 );
            v1=2*d-1;v2=2*d-1;w=v1*v1+v2*v2;
          }while(w>=1.0);
          w=Math.sqrt((-2.0*Math.log(w))/w); z1=v1*w;z2=v2*w;
          }
          return (x+z1*s);
        }
 /*'uniform' returns a psedo-random variate from a uniform distribution */
 /* with lower bound a and upper bound b*/
 public double get_uni( int a,int b)
        {
      double d;
      if (a>b)
        System.out.println("uniform Argument error: a>b");
      while( (d=r.nextDouble())==0.0 );
      return (a+(b-a)*d);
```

```
 }
        public static int rand( int max )
        {       int     i;
                Double  x       = new Double( max * Math.random() );
                i       = x.intValue();
                if( i>=max )
                        i       = max-1;
                return( i );
        }

        private Random  r;
}
//-----------------------------------------------------------------------------------------
// Base class for all queue cell
abstract class Cell
{       public Cell( String types, int id )
        {       this.id  = id;
                this.types      = types;
           entry        = 0;
                depart  = 0;
                count   = 0;
  n =0;
        }

        public void reset()
        {       entry   = 0;
                depart  = 0;
                count   = 0;
  n=0;
        }

        public String name()
        {       return  types+":"+id;
        }

        public abstract boolean empty();        // if !empty then can get()
        public abstract int get();
        public abstract boolean busy();                 // if !busy then can put()
 public abstract boolean get_cellbusy();
        public abstract void put( int x );

        public int      entry,  // entry (put) statistic
                        depart, // depart (get) statistic
                        count,  // number of data processed
    n, //number of customers
                        id;     // Cell id
        public String   types;
}
//-----------------------------------------------------------------------------------------
// Single FIFO Queue
class Queue extends Cell
{       public Queue( int length )
        {       super("Queue",0);       // 0 means no need id for queue
                q       = new int[length];
                reset();
        }

        public Queue()
```

```
        {       this( 1024 );
        }

        public void reset()
        {       super.reset();
                qh      = 0;
                qt      = 0;
                next    = null;
        }

        public boolean empty()
        {       return( (qh-qt)==0 );
        }

        public int get() // practical chain
        {       int     r;
                if( qt<q.length )
                {       depart++;
                        r       = q[qt++];
                        if( (qt==q.length)&&(next!=null) )
                        {       q       = next.q;
                                qh      = next.qh;
                                qt      = next.qt;
                                next    = next.next;
                        }
                        return( r );
                }
                return( -1 );   // Queue underflow
        }

        public boolean busy()
        {       return( false );
        }
public boolean get_cellbusy()
{
 return(false);
}
        public void put( int x )
        {       entry++;
                if( qh<q.length )
                        q[qh++]         = x;
                else
                {       if( next==null )
                                next    = new Queue( q.length );
                        next.put( x );
                }
        }

        public int count() //count through the chain
        {       int     c;
                if( next!=null )
                        c       = next.count();
                else
                        c       = 0;
                c       += qh-qt;
                return( c );
        }
```

```
        public String toString()//convert any value of this class to string for display.
        {       int     c       = count();
                return( "Q("+entry+"/"+depart+":"+c+")" );
        }

        private int     q[];
        private int     qh,qt;
        private Queue next;
}
//-----------------------------------------------------------------------------------------
// Statistical Collection
class State extends Cell
{       public  int     entry[];   // time point for each customer entry to queue
        public  int     start[];   // time point for each customer get in service
        public  int     servi[];   // time point for each used on service
 public  int sets[];  // s+=n*(time-tn);n++; tn=time;  --case put
 public  int setn[];    // s+=n*(time-tn);n--; tn=time;  --case get
        public State( int id, int length )
        {       super("State",id);
                incsi   = length; //increment interval for increment size of state object
                reset();
        }

        public State( int id )
        {       this( id,1024 );
        }

        public State dump()
        {       return  this;
        }

        public void reset()
        {       super.reset();
                entry   = new int[incsi];
                start   = new int[incsi];
                servi   = new int[incsi];
   sets = new int[incsi];
   setn = new int[incsi];
        }

        public boolean empty()
        {       return( count==0 );
        }

        public int get()
        {       return  count;
        }

        public boolean busy()
        {       return( false );
        }
        public boolean get_cellbusy()
        {       return( false );
        }

        public void put( int entry_time )
        {       if( count==entry.length )
                {       int     l = entry.length+incsi,
```

```
                        tmp[];
                    tmp     = new int[l];
                    System.arraycopy(tmp,0,entry,0,entry.length);
                    entry   = tmp;
                    tmp     = new int[l];
                    System.arraycopy(tmp,0,start,0,start.length);
                    start   = tmp;
                    tmp     = new int[l];
                    System.arraycopy(tmp,0,servi,0,servi.length);
                    servi   = tmp;
                    tmp     = new int[l];
                    System.arraycopy(tmp,0,sets,0,sets.length);
                    sets    = tmp;
                    tmp     = new int[l];
                    System.arraycopy(tmp,0,setn,0,setn.length);
                    setn    = tmp;

            }
            entry[count]    = entry_time;
    }

    public void put_start( int time )
    {       start[count]    = time;
    }

    public void put_depart( int time )
    {       servi[count++]  = time;

    }

    public String toString()
    {       String  s;
            if( count>0 )
                    s       = name()+"(entried="+count+")";
            else
                    s       = name()+"(empty)";
            return( s );
    }

    private int     incsi;
}
//-----------------------------------------------------------------------------------------
// Cell Server
class Server extends Cell
{       public Server( int id, int time, XRandom random )
        {       super("Server",id);
                this.time       = time;         // service time
                state           = new State(id);
                this.random     = random;       // Random generator
        }

        public void reset()
        {       super.reset();
                tcount          = 0;            // service time counter
                busytime        = 0;
                serve           = false;// serving flag
                finish          = false;
                state.reset();
```

```
    }

    public boolean empty()
    {       return( !finish );
    }

    public int get()
    {       finish  = false;
            return  ltime;
    }

    public boolean busy()
    {       return( serve );
    }
    public boolean get_cellbusy()
    {       return( serve );
    }

    public void put( int i )
    {       entry++;
            state.put( i );
            tcount  = random.get_exp( time );
            serve   = true;
            state.put_start( ltime );
    }

    public State state()
    {       return( this.state.dump() );
    }

    public void run( int g_time )
    {       ltime   = g_time;
            if( tcount>0 )
            {       tcount--; //count down service unit time.
                    busytime++;
            }
            else
            {       if( serve )
                    {       state.put_depart( g_time );
                            serve   = false;
                            finish  = true;
                            depart++;
                    }
            }
    }

    public String toString()
    {       return( name()+"("+(serve?"S,":"I,")+entry+"/"+depart+","+state+")" );
    }

    private int     tcount,ltime;
    private State   state;
    private boolean serve,finish;
    private XRandom random;
    public  int     busytime,time;
}
//-----------------------------------------------------------------------------
// Base Gate rounter
```

```
abstract class Gate
{       public Gate( int id )
    {    this.id = id;
         src    = null;
         dst    = null;
         prob   = null;  //control probability
              pick    = false;
              info    = false;
              count   = 0;
              x       = 0;
              y       = 0;
    }

       public void gate( Cell src[], Cell dst[],int prob[])
       {       this.src = src;
               this.dst = dst;
  this.prob = prob; //handle prob.
               reset();
       }

       public void reset()
       {       count   = 0;
       }

       public abstract void route();
       public String write()
       {       String  buf     = ""; //clear buffer string
               for( int i=0; i<src.length; i++ )
                       buf     += " " + src[i].id;
               buf     += " to";
               for( int i=0; i<dst.length; i++ )
                       buf     += " " + dst[i].id;
  buf  += " with";
               for( int i=0; i<prob.length; i++ )
                       buf     += " " + prob[i];
  return       buf;
       }

       public String toString()
       {       String  sf,st,sp;
               int     i;
               for( sf=" ",i=0; src!=null&&i<src.length; i++ )
                       sf      += src[i].name() + " ";
               for( st=" ",i=0; dst!=null&&i<dst.length; i++ )
                       st      += dst[i].name() + " ";
               for( sp=" ",i=0; prob!=null&&i<prob.length; i++ )
                       sp      += prob[i] + " ";
  if (prob!=null)
   {return( "Gate:"+id+"("+sf+"->"+st+" with "+sp+":"+count+")" );}
  else
               {return( "Gate:"+id+"("+sf+"->"+st+":"+count+")" );}
       }
       public String name()
       {       return  "Gate";
       }

       public Gate dump() //is used for copy pointer
```

```
        {       return( this );
        }

        public int[] get_source()
        {       int[]   s       = new int[src.length];
                for( int i=0; i<src.length; i++ )
                        s[i]    = src[i].id;
                return  s;
        }

        public int[] get_dest()
        {       int[]   s       = new int[dst.length];
                for( int i=0; i<dst.length; i++ )
                        s[i]    = dst[i].id;
                return  s;
        }
        public int[] get_prob()
        {       int[]   p       = new int[prob.length];
                for( int i=0; i<prob.length; i++ )
                        p[i]    = prob[i];
                return  p;
        }

        protected Cell src[];
        protected Cell dst[];
 public int prob[];
        public  int     count,id,x,y,w;
        public  boolean         pick,info;
}
//-----------------------------------------------------------------------------------
// Random gate
class RGate extends Gate
{       public RGate( int id )
    {   super(id);
    }

 public void route()
        {       int     rts[]   = new int[src.length];  // source route table
                int     rtd[]   = new int[dst.length];  // dest route table
                int     si,di;
                for( si=0; si<src.length; si++ ) rts[si] = si;
                for( di=0; di<dst.length; di++ ) rtd[di]        = di;
                while( (si>0)&&(di>0) )
                {       int     xs      = XRandom.rand( si );
                        if( !src[rts[xs]].empty() )
                        {       int     xd      = XRandom.rand( di );
                                if( !dst[rtd[xd]].busy() )
                                {       count++;
                                        dst[rtd[xd]].put( src[rts[xs]].get() );
                                }
                                else
                                        System.arraycopy(rtd,xd,rtd,xd+1,--di-xd);
                        }
                        else
                                System.arraycopy(rts,xs,rts,xs+1,--si-xs);
                }
        }
```

```
        public String write()
        {       return  "link random"+super.write();
        }

        public String toString()
        {       return( "R"+super.toString() );
        }

        public String name()
        {       return  "RGate";
        }
}
//-------------------------------------------------------------------------------------
// Sequence gate
class SGate extends Gate
{       public SGate( int id )
    {   super(id);
    }

        public void route()
        {       for( int i=0; i<src.length; i++)
                {       int     busy    = 0;
                        int     j       = 0;
                        while( !src[i].empty()&&(busy<dst.length) )
                        {       if( dst[j].busy() )
                                        busy++;
                                else
                                {       count++;
                                        dst[j].put( src[i].get() );
                                }
                                j++;
                                if( j>=dst.length )
                                {       j       = 0;
                                        if( busy<dst.length )
                                                busy    = 0;
                                }
                        }
                }
        }

        public String write()
        {       return  "link sequence"+super.write();
        }

        public String toString()
        {       return( "S"+super.toString() );
        }

        public String name()
        {       return  "SGate";
        }
}
//-------------------------------------------------------------------------------------
// Pop gate
class PGate extends Gate
{       public int      pop[];
 int sumpop;
```

```
        public PGate( int id, int pop[] )
    {       super(id);
                this.pop        = pop;
                sumpop                  = 0;
                for( int i=0; i<pop.length; i++)
                        sumpop          += pop[i];
    }
public void route()
        {       int     c       = 0,
                        i,xs;

                while( c<src.length )
                {       if( src[c].empty() )
                                c++;
                        else
                        {       xs      = XRandom.rand( sumpop );
                                for( i=0; (xs>0)&&(i<pop.length); xs-=pop[i++]);
                                if( i<dst.length ){
     count++;
                                        dst[i].put( src[c].get() );
    }
                        }
                }
        }

        public String write()
        {       return  "link propability"+super.write();
        }

        public String toString()
        {       return( "P"+super.toString() );
        }

        public String name()
        {       return  "PGate";
        }
}
//-----------------------------------------------------------------------------------
// Here is Simulation class
// Runable Cell
abstract class RCell extends Cell
{
        public RCell( int id, Color coln, Color colp )
        {       super("Box",id);
                info            = false;
                pick            = false;
                x               = 0;
                y               = 0;
                w               = 10;
                this.coln       = coln;
                this.colp       = colp;
        }
        public abstract void run( int time );
        public RCell dump()
        {       return( this );
        }
```

```
        public void paint(Graphics g, FontMetrics fm)
        {       String  lbl     = new String( this.types + this.id );
                g.setColor( pick ? colp : coln );
                        w = fm.stringWidth(lbl) + 10;
                int     h = fm.getHeight() + 4;
                g.fillRect(x - w/2, y - h / 2, w, h);
                g.setColor(Color.black);
                g.drawRect(x - w/2, y - h / 2, w-1, h-1);
                g.drawString(lbl, x - (w-10)/2, (y - (h-4)/2) + fm.getAscent());
                if( info )
                        info( g, fm );
        }

        public abstract void info(Graphics g, FontMetrics fm);
        public abstract String write();

        public  int     x,y,w;
        public  boolean         info,pick;
        protected       Color   coln,colp;
}
//-------------------------------------------------------------------------------------
// QueueBox Container
class QBox extends RCell
{ public  int gstate[];
  public  int gentry   = 0;
  public  float area   = (float)0.0;

  public QBox( int id, int sv_time[], Gate in, Gate out )
        {       super( id, Color.gray, Color.lightGray );
           types                = "QBox";
                XRandom r       = new XRandom();
                server  = new Server[sv_time.length];
                for( int i=0; i<sv_time.length; i++)
                     server[i]  = new Server( i,sv_time[i], r );
                qin     = new Queue[1];
                qin[0]  = new Queue();
                qout    = new Queue[1];
                qout[0]= new Queue();
                gin     = in.dump();
                gout    = out.dump();
                gin.gate(qin,server,null);
                gout.gate(server,qout,null);
    gstate  = new int[QSimPanel.timeSpace];
        }

        public QBox dumpQBox()
      {     return  this;
      }

  public void reset()
        {       super.reset();
                for( int i=0; i<server.length; i++ )
                        server[i].reset();
                qin[0].reset();
                qout[0].reset();
                gin.reset();
                gout.reset();
    area    = (float)0.0;
```

```
 gentry  = 0;
 for(int i=0; i<QSimPanel.timeSpace; i++)
  gstate[i] = 0;
      }

      public boolean empty()
      {       return( qout[0].empty() );
      }

      public int get()
      {       depart++;n--;
              return( qout[0].get() );
      }

      public boolean busy()
      {       return( qin[0].busy() );
      }

public void set_cellbusy()
{
 cellbusy=true;
 for(int i=0; i< server.length;i++)
 {
  cellbusy = server[i].busy() & cellbusy;
 }
}
public boolean get_cellbusy()
{
  return cellbusy;
}

      public void put( int i )
      {       entry++;n++;
              qin[0].put( i );
      }

      public void run( int g_time )
      { if( gentry < QSimPanel.timeSpace )
  gstate[gentry++]  = n;
 else
 { System.arraycopy(gstate,1,gstate,0,QSimPanel.timeSpace-1);
  gstate[QSimPanel.timeSpace-1]  = n;
 }
 area  += (float)n;
 gin.route();
              gout.route();
              for( int i=0; i<server.length; i++ ){
                      server[i].run( g_time );
 }
              gin.route();
              gout.route();
 set_cellbusy();
      }
      public String toString()
      {       String  s;
              s       = name()+"("+qin[0]+" /"+qout[0]+",{ ";
              for( int i=0; i<server.length; i++ )
```

```
                s       += server[i] + " ";
        s       += "})";
        return  s;
    }

    public void info(Graphics g, FontMetrics fm)
    {
    }

    public String write()
    {   String  buf     = "qbox";
        for( int i=0; i<server.length; i++ )
                buf     += " " + server[i].time;
        return  buf;
    }

    // Statistic Report ----------------------------------------------

    public int r_numserv()
    {   return  server.length;
    }

    public int r_serv_busytime( int i )
    {   return  server[i].busytime;
    }

    public int r_serv_busytimes()
    {   int     sum     = 0;
        for( int i=0; i<server.length; i++)
                sum     += server[i].busytime;
        return  sum;
    }

    public int r_numcust( int si )
    {   return  server[si].state().count;
    }

    public int r_cust_resident( int si, int i )
    {   State   s       = server[si].state();
        return  s.servi[i] - s.entry[i];
    }

public int r_cust_residents( int si )
{   int   sum       = 0;
    State  s      = server[si].state();
    for( int i=0; i<s.count; i++ )
    {   sum    += s.servi[i] - s.entry[i];
    }
    return  sum;
}

    public  Server  server[];
    private Queue qin[],qout[];
    private Gate   gin,gout;
    public int     entry,depart,count,n; //n for no. of customer
 public double s,tn; //s and tn are used for collect data to compute L
 public boolean cellbusy;
}
```

```
//-------------------------------------------------------------------------------------
// Customer supplier-expnential-distribution
class SupplyExp extends RCell
{       public SupplyExp( int id, int time, XRandom r )         // 1 customer/ time
        {       super(id, Color.cyan, Color.orange );
          types         = "Supplier-exp";
                this.time       = time;
                tc              = 0;    //time count down
                cc              = 0;
                this.r          = r;
        }

        public void reset()
        {       super.reset();
                tc      = 0;
                cc      = 0;
        }

        public boolean empty()
        {       return( cc==0 );
        }

        public int get()
        {       if( cc>0 )
                {       cc--;
                        count++;
                }
                return  gtime;
        }

        public boolean busy() // if always busy
        {       return  true;
        }
        public boolean get_cellbusy()// if always busy
        {       return  true;
        }

        public void put( int x )
        {
        }

        public void run( int gtime )
        {       this.gtime      = gtime;
                if( tc==0 )
                {       tc      = r.get_exp( time );
                        cc++;
                }
                else
                        tc--;
        }

        public String toString()
        {       return( name()+"("+tc+"/"+time+":"+count+"+"+cc+")" );
        }

        public void info(Graphics g, FontMetrics fm)
        {
        }
```

```
        public String write()
        {       return  "supplier-exp " + this.time;
        }

        private int     time,tc,cc,gtime;
        private XRandom r;
}
//-------------------------------------------------------------------------------------------
// Customer supplier-normal-distribution
class SupplyNor extends RCell
{       public SupplyNor( int id, int time, XRandom r )         // 1 customer/ time
        {       super(id, Color.cyan, Color.orange );
           types        = "Supplier-nor";
                this.time       = time;
                tc              = 0;    //time count down
                cc              = 0;
                this.r          = r;
        }

        public void reset()
        {       super.reset();
                tc      = 0;
                cc      = 0;
        }

        public boolean empty()
        {       return( cc==0 );
        }

        public int get()
        {       if( cc>0 )
                {       cc--;
                        count++;
                }
                return  gtime;
        }

        public boolean busy() // if always busy
        {       return  true;
        }
        public boolean get_cellbusy()// if always busy
        {       return  true;
        }

        public void put( int x )
        {
        }

        public void run( int gtime )
        {       this.gtime      = gtime;
                if( tc==0 )
                {       tc      = (int)r.get_nor(time,0 );
                        cc++;
                }
                else
                        tc--;
        }
```

```
        public String toString()
        {       return( name()+"("+tc+"/"+time+":"+count+"+"+cc+")" );
        }

        public void info(Graphics g, FontMetrics fm)
        {
        }

        public String write()
        {       return  "supplier-nor " + this.time;
        }

        private int     time,tc,cc,gtime;
        private XRandom r;
}
//-------------------------------------------------------------------------------------------
// Customer supplier-uniform-distribution
class SupplyUni extends RCell
{
 public SupplyUni( int id, int arri[],XRandom r )       // 1 customer/ time
        {       super(id, Color.cyan, Color.orange );
          types         = "Supplier-uni";
   this.arri=arri;
                tc              = 0;    //time count down
                cc              = 0;
                this.r          = r;
        }

        public void reset()
        {       super.reset();
                tc      = 0;
                cc      = 0;
        }

        public boolean empty()
        {       return( cc==0 );
        }

        public int get()
        {       if( cc>0 )
                {       cc--;
                        count++;
                }
                return  gtime;
        }

        public boolean busy() // if always busy
        {       return  true;
        }
        public boolean get_cellbusy()// if always busy
        {       return  true;
        }
        public void put( int x )
        {
        }
```

```
        public void run( int gtime )
        {       this.gtime      = gtime;

                if( tc==0 )
                {       tc      = (int)Math.round(r.get_uni(arri[0],arri[1]));
    System.out.println(tc);
                        cc++;
                }
                else
                        tc--;
        }

        public String toString()
        {       return( name()+"("+tc+"/"+arri[0]+","+arri[1]+":"+count+"+"+cc+")"
);
        }

        public void info(Graphics g, FontMetrics fm)
        {
        }

        public String write()
        {       return  "supplier-uni " + this.arri[0]+","+this.arri[1];
        }

        private int     arri[],tc,cc,gtime;
        private XRandom r;
}
//-----------------------------------------------------------------------------------------
// Customer terminator
class Terminate extends RCell
{       public  int     count;  // Count number of terminated customer

        public Terminate( int id )
        {       super( id, Color.pink, Color.red );
           types  = "Terminator";
                count   = 0;
        }
        public void reset()
        {       super.reset();
                count   = 0;
        }

        public boolean empty()  // always empty
        {       return  true;
        }

        public int get()
        {       return  0;
        }

        public boolean busy() // always free
        {       return  false;
        }
        public boolean get_cellbusy()// always free
        {       return  false;
        }
```

```
        public void put( int x )
        {       count++;
        }

        public void run( int time )
        {
        }

        public String toString()
        {       return( name()+"("+count+")" );
        }

        public void info(Graphics g, FontMetrics fm)
        {
        }

        public String write()
        {       return  "terminator";
        }
}
//---------------------------------------------------------------------------------------------
// Simulator class , this is the main program.
public class QSimPanel extends Canvas
{       public          int     time;
 public static final int timeSpace  = 200;
 QSimControl parent;

        public QSimPanel( int size, QSimControl parent )
        { newPanel(size);
  this.parent = parent;
        }

 public void newPanel( int size )
 {      cell            = new RCell[size];
                gate            = new Gate [size-1];
                cellc           = 0; //cell count
                time            = 0;
                gatec           = 0;//gate count
  lc_debug    = 100;
 }

 private  int loc_x,
          loc_y;
 private  boolean loc_c = false;

 public void locatei( int x, int y )
 { loc_x = x;
  loc_y = y;
  loc_c = true;
 }

        public int put( RCell c )
        {       int     r;
                if( cellc<cell.length )
                {       r       = cellc;
                        Dimension d   = getSize();
   if( loc_c )
   { c.x = loc_x;
```

```
  c.y = loc_y;
  loc_c = false;
 }
 else
                {       c.x     = (cellc+1)*90;
                        c.y     = d.height/2;
 }
                        cell[cellc++]   = c.dump();
                }
                else
                        r       = -1;
                return( r );
        }

        public int del( int nidx )
        {       int     r;
                if( (nidx<0)||(nidx>=cellc) )
                        return  -1;
                for( int i=0; i<gatec; i++ )
                {       int[]   list;
                        if( chkhitCell( nidx, gate[i].get_source() )||
                            chkhitCell( nidx, gate[i].get_dest() ) )
                                return  -1;
                }
                for( int i=nidx+1; i<cellc; i++ )
                        cell[i].id--;
                System.arraycopy( cell, nidx+1, cell, nidx, --cellc-nidx );
                return  cellc;
        }

        public void mlink( int from[], int to[], Gate g )
        {       if( gatec>=gate.length )
                        return;
                Cell    src[]   = new Cell[from.length];
                Cell    dst[]   = new Cell[to.length];
int  prob[] = new int[to.length];
                for( int i=0; i<from.length; i++ )
                        src[i]  = cell[from[i]].dump();
                for( int i=0; i<to.length; i++ )
                        dst[i]  = cell[to[i]].dump();
if ((g.name()).compareTo("PGate")==0){
  PGate pg=(PGate)g;  //point to PGate
  for(int i=0;i<to.length;i++)
  prob[i]=pg.pop[i]; //retrive pop value
}
g.gate(src,dst,prob);
                gate[gatec]     = g.dump();
if( loc_c )
{ gatec++;
  g.x = loc_x;
  g.y = loc_y;
  loc_c= false;
}
else
                adjGate( gatec++ );
        }

        public void rlink( int gidx )
```

```
        {       if( (gidx<0)||(gidx>=gatec) )
                        return;
                System.arraycopy( gate, gidx+1, gate, gidx, --gatec-gidx );
        }

        private boolean chkhitCell( int idx, int[] list )
        {       for( int i=0; i<list.length; i++ )
                        if( list[i]==idx )
                                return  true;
                return  false;
        }

        public  void sim_run()
        {       for( int i=0; i<gatec; i++ )
                        gate[i].route();
                for( int i=0; i<cellc; i++ )
                        cell[i].run( time );
                for( int i=0; i<gatec; i++ )
                        gate[i].route();
                time++;
        }

        public String toString()
        {       String  s;
                s       = "Simulate(time="+time+",{ ";
                for( int i=0; i<cell.length; i++ )
                if( cell[i]!=null )
                                s       += cell[i] + " ";
                s       += "})";
                return( s );
        }

        private RCell  cell[];
        private Gate   gate[];
        private int    cellc,gatec;

// File Input  functions
        private String  preerr,
            msginfo,
            msgstatus;

  public String getMsgInfo()
  { String  s = msginfo;
    msginfo   = "";
    return s;
  }

  public String getMsgStatus()
  { String  s = msgstatus;
    msgstatus = "";
    return s;
  }

        private void error( String s )
        {       System.out.print(preerr);
                System.out.println(s);
        }
```

```
private void info( String s )
{ //System.out.println( s );
  msginfo   = msginfo + s + "\n";
}
private void info_no_linefeed( String s )
{ //System.out.println( s );
  msginfo   = msginfo + s;
}

private void status( String s )
{ msgstatus = msgstatus + s + "\n";
}

        public boolean load_file( String filename )
        { boolean done          = false,
                  error         = false;
  loc_c = false;
  try
  { FileReader in = new FileReader( filename );
  StreamTokenizer ins           = new StreamTokenizer(in);
                  ins.commentChar(';');
                  ins.eolIsSignificant(true);
                  ins.slashSlashComments(true);
                  ins.lowerCaseMode(true);
                  int     cntw    = 0;
                  int     ttys;
                  String  buf     = "",
                  param   = "";
                  do
                  {       preerr  = "line:"+ins.lineno()+": ";
                          try
                          {       ttys      = ins.nextToken();
                        switch( ttys )
                          {  case StreamTokenizer.TT_EOF:
                             done    = true;
                                break;
                           case StreamTokenizer.TT_EOL:
                                        //System.out.println("'"+buf+"'("+param+")");
                                error= !cmdline(buf,param);
                                cntw    = 0;
                                buf     = "";
                                param   = "";
                                break;
                           case StreamTokenizer.TT_NUMBER:
                                if( cntw==0 )
                                {       error("need command");
                                        error   = true;
                                }
                                else
                                {       Double  d       = new Double(ins.nval);
                                        Integer i       = new Integer(d.intValue());
                                        if( cntw==1 )
                                                param   = i.toString();
                                        else
                                                param   +=" "+i;
                                }
                                cntw++;
                                break;
```

```
                        case StreamTokenizer.TT_WORD:
                            if( cntw==0 )
                                buf    = ins.sval;
                            else
                                if( cntw==1 )
                                    param   = ins.sval;
                                else
                                    param   += " "+ins.sval;
                            cntw++;
                            break;
                    }
                    }
                    catch( Exception e )
                    {       error("Input file error:"+e);
                            done    = true;
                    }
                }while(!done||error);
    }
    catch( Exception e )
    {     System.out.println("Can't load input file "+e);
                }
    return  !error;
        }

        public boolean cmdline( String s, String p1 )
        {
//          X;  represents throughput
//          B; represents totalbusytime;
//          T0; represents means service time
//          U; represents server utilisation
//          sumWi; represents residenc time sum
//          W; represents mean residence time
//          W0; represents mean queue time
//          L; represents mean number in system
//          L0; represents mean number in queue

            if( s.compareTo("info")==0 )  //display object information
                {       info("[Info]");
                        for( int i=0; i<cellc; i++ )
                                System.out.println(" * "+cell[i]);
                        info(" ---");
                        for( int i=0; i<gatec; i++ )
                                info(" * "+gate[i]);
                        info(" ---");
                        info(" time="+time+",  debugc\t= "+lc_debug);
                }
                else
                if( s.compareTo("run")==0 ) //simulation time
                {       int     loop;
                        try
                        {       loop    = Integer.parseInt(p1);
                        }
                        catch( Exception e )
                        {       error("Invalid 'run' parameter:"+e);
                                return  false;
                        }
                        loop    = Math.abs(loop);
                    info("running("+loop+")...");
```

```
                        while( loop>0 )
                        {       sim_run();
                                loop--;
                                if( (lc_debug>0)&&(loop%lc_debug)==0 )
                                        info(this.toString());
                        }
    this.cmdline("info","");
    this.cmdline("throughput","");
    this.cmdline("total-busy-time","");
    this.cmdline("arrival-rate","");
    this.cmdline("mean-service-time","");
    this.cmdline("utilisation","");
    this.cmdline("resident-time-sum","");
    this.cmdline("mean-resident-time","");
    this.cmdline("mean-queue-time","");
    this.cmdline("mean-number-in-system","");
    this.cmdline("mean-number-in-queue","");
                }
                else
                if( s.compareTo("debugc")==0 )  //debugging,trace
                {       try
                        {       lc_debug= Integer.parseInt(p1);
                        }
                        catch( Exception e )
                        {       error("Invalid 'debugc' parameter:"+e);
                                return  false;
                        }
                }
                else
    if( s.compareTo("locate")==0 )
    { int x,y;
     StringTokenizer st = new StringTokenizer( p1 );
     int  max    = st.countTokens();
     if( max!=2 )
                        {       error("Invalid 'locate' parameter: required x,y");
                                return  false;
                        }
     try
     { x= Integer.parseInt( st.nextToken() );
       y= Integer.parseInt( st.nextToken() );
     }
     catch( Exception e )
     { error("Invalid 'locate' parameter:"+e);
       return  false;
     }
     locatei(x,y);
    }
    else
                if( s.compareTo("supplier-exp")==0 )
                {       try
                        {       int i= Integer.parseInt(p1);
                                put( new SupplyExp(cellc,i,new XRandom()) );
                        }
                        catch( Exception e )
                        {       error("Invalid 'supplier-exponential-distribution'
parameter:"+e);
                                return  false;
                        }
```

```
                }
                else
                if( s.compareTo("supplier-nor")==0 )
                {       try
                        {       int i= Integer.parseInt(p1);
                                put( new SupplyNor(cellc,i,new XRandom()) );
                        }
                        catch( Exception e )
                        {       error("Invalid 'supplier-normal-distribution'
parameter:"+e);
                                return  false;
                        }
                }
   else
                if( s.compareTo("supplier-uni")==0 )
                { StringTokenizer st=new StringTokenizer(p1);
     int max=st.countTokens();
     if(max<=1)
     { error("Invalid 'supplier-uniform' parameter: incomplete");
      return false;
     }
     int arrival_rate[]= new int[max];
     for(int i=0;i<max;i++)
     {
      try
                        {       arrival_rate[i]= Integer.parseInt(st.nextToken());
      }
                        catch( Exception e )
                        {       error("Invalid 'supplier-uniform-distribution' parameter
("+i+"):"+e);
                                return  false;
                        }
     }
                 put( new SupplyUni(cellc,arrival_rate,new XRandom()) );
                }
   else
                if( s.compareTo("terminator")==0 )
                {       put( new Terminate(cellc) );
                }
                else
                if( s.compareTo("qbox")==0 )
                {       StringTokenizer st = new StringTokenizer( p1 );
                        int     max     = st.countTokens();
                        if( max<=0 )
                        {       error("Invalid 'box' parameter: no parameter");
                                return  false;
                        }
                        int     sv[]    = new int[max];
                        for( int i=0; i<max; i++ )
                        {       try
                                {       sv[i]   = Integer.parseInt( st.nextToken() );
                                }
                                catch( Exception e )
                                {       error("Invalid 'box' parameter("+i+"):"+e);
                                        return  false;
                                }
                        }
                        put( new QBox( cellc, sv, new SGate(0), new RGate(1) ) );
```

```
                }
                else
                if( s.compareTo("link")==0 )
                {       StringTokenizer st = new StringTokenizer( p1 );
                        int     max     = st.countTokens();
                        if( max<=3 )
                        {       error("Invalid 'link' parameter: less parameter");
                                return  false;
                        }
                        String  type    = st.nextToken();
                        int     lf[]    = new int[64];
                        int     lt[]    = new int[64];
    int  lp[]   = new int[64];
                        int     fc      = 0,
                                tc      = 0,
     pc= 0,
                                i       = 1;
                        try
                        {
                                for( ; i<max; i++ )
                                {       String  ss      = st.nextToken();
                                        if( ss.compareTo("to")==0 )
                                break;
                                        lf[fc++]        = Integer.parseInt( ss );
                                }
                                for( i++; i<max; i++ )
                                {       String  sss     = st.nextToken();
                                        if( sss.compareTo("with")==0 )
                                                break;
                                        lt[tc++]        = Integer.parseInt( sss );
                                }
                                for( i++; i<max; i++ )
                                        lp[pc++]        = Integer.parseInt( st.nextToken()
);
                        }
                        catch( Exception e )
                        {       error("Invalid 'link'
parameter("+i+"):"+fc+"/"+tc+":"+e);
                                return  false;
                        }
                if( fc==0 )
                {       error("Invalid 'link' parameter: No any from node");
                        return  false;
                }
                if( tc==0 )
                {       error("Invalid 'link' parameter: No any to node");
                        return  false;
                }
                int src[]  = new int[fc];
                System.arraycopy(lf,0,src,0,fc);
                int dst[]  = new int[tc];
                System.arraycopy(lt,0,dst,0,tc);
                if( type.compareTo("random")==0 )
                        mlink( src, dst, new RGate(0) );
                else
                if( type.compareTo("sequence")==0 )
                        mlink( src, dst, new SGate(0) );
                else
```

```
                                if( type.compareTo("probability")==0 )
                                    {
                                        if( pc==0 )
                                            error("Invalid 'poplink' need 'with [pop
list]");
                                        int llp[]       = new int[pc];
                                          System.arraycopy(lp,0,llp,0,pc);
                                          mlink( src, dst, new PGate(0,llp) );
                                }
                    else
                    { error("Invalid 'link' type parameter");
                     return  false;
                    }
                    }
                    else
                if( s.compareTo("reset")==0 )
                {     for( int i=0; i<cellc; i++ )
                          cell[i].reset();
                      for( int i=0; i<gatec; i++ )
                          gate[i].reset();
                      time  = 0;
                }
                else
                if(s.compareTo("arrival-rate")==0)
                { info("Arrival Rate [Lamda] : ");
                      for( int i=0; i<cellc; i++ )
                     if( cell[i] instanceof QBox )
                     { QBox   qb     = ((QBox)cell[i]).dumpQBox();
                      Float  c      = new Float(qb.entry);
                      Float  _time=  new Float(time);
                      System.out.println(_time);
                      System.out.println(c);
                      float   f       = c.floatValue()/_time.floatValue();
                      info(qb.name()+"="+f+" ");
                     }
                     info("");
                }
                else
                if( s.compareTo("throughput")==0 )
                {     info("Throughput[X]: ");
                      for( int i=0; i<cellc; i++ )
                          if( cell[i] instanceof QBox )
                          {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                              //find completion of each server
                              for(int j=0;j<qb.r_numserv();j++){
                                Float depart_each_server = new Float(qb.r_numcust(j));

info_no_linefeed("server"+j+"completion:"+depart_each_server.floatValue()+" |
"+"X="+depart_each_server.floatValue()/time+" , ");
                              }
                              //
                              Float  c      = new Float(qb.depart);
                              float f       = c.floatValue()/time;
                              info(qb.name()+"="+f+" ");
                          }
                      info("");
                }
                else
```

```
            if( s.compareTo("total-busy-time")==0 )
            {     info("busytime [B]: ");
                  for( int i=0; i<cellc; i++ )
                        if( cell[i] instanceof QBox )
                        {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                              //display each B depend on the Server
                              for(int j=0;j<qb.r_numserv();j++){
                                Float b = new Float(qb.r_serv_busytime(j));
                                info_no_linefeed("server"+j+":"+b.floatValue()+", ");
                              }
                              //busytime of Qbox
                              Float  c      = new Float(qb.r_serv_busytimes());
                              float f       = c.floatValue()/qb.r_numserv();
                              info(qb.name()+"="+c.floatValue()+" ");
                        }
                  info("");
            }
            else
            if( s.compareTo("mean-service-time")==0 )
            {     info("Mean Service Time[Ts]: ");
                  for( int i=0; i<cellc; i++ )
                        if( cell[i] instanceof QBox )
                        {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                              //display each B depend on the Server
                              for(int j=0;j<qb.r_numserv();j++){
                                Float b = new Float(qb.r_serv_busytime(j));
                                Float depart_each_server = new Float(qb.r_numcust(j));
                                info_no_linefeed("Ts/server"+j+":"+b.floatValue
()/depart_each_server.floatValue()+", ");
                              }
                              //mean-service-time fo Qbox

                              Float  c      = new Float(qb.r_serv_busytimes());
                              Float  completion = new Float(qb.depart);
                              float f       = c.floatValue()/qb.r_numserv();
                              float  Ts     = c.floatValue()/completion.floatValue();
                              info(qb.name()+"="+Ts+" ");
                        }
                  info("");
            }
            else
            if( s.compareTo("utilisation")==0 )
            {     info("Server Utilisation[U]: ");
                  for( int i=0; i<cellc; i++ )
                        if( cell[i] instanceof QBox )
                        {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                              //display each B depend on the Server
                              for(int j=0;j<qb.r_numserv();j++){
                                Float b = new Float(qb.r_serv_busytime(j));
                                Float depart_each_server = new Float(qb.r_numcust(j));
                                info_no_linefeed("U/server"+j+":"+b.floatValue()/time+",
");
                              }
                              //total utilization
                              Float  c      = new Float(qb.r_serv_busytimes());
                              float f       = c.floatValue()/qb.r_numserv();
                              float  U      = c.floatValue()/time;
                              info(qb.name()+"="+U+" ");
```

```
            }
        info("");
    }
    else
    if( s.compareTo("resident-time-sum")==0 )
    {    info("Residence Time Sum[EWi]: ");
        for( int i=0; i<cellc; i++ )
            if( cell[i] instanceof QBox )
            {    QBox   qb    = ((QBox)cell[i]).dumpQBox();
                int    ns    = qb.r_numserv();
                float  sum   = 0;
                for( int j=0; j<ns; j++ )
                {    sum    += qb.r_cust_residents(j);
                }
                info(qb.name()+"="+sum+" ");
            }
        info("");
    }
    else
    if( s.compareTo("mean-resident-time")==0 )
    {    info("Meam Residence Time[W]: ");
        for( int i=0; i<cellc; i++ )
            if( cell[i] instanceof QBox )
            {    QBox   qb    = ((QBox)cell[i]).dumpQBox();
                int    ns    = qb.r_numserv();
                float  sum   = 0;
                for( int j=0; j<ns; j++ )
                {    sum    += qb.r_cust_residents(j);
                }
                Float  completion = new Float(qb.depart);
                float W = sum/completion.floatValue();
                info(qb.name()+"="+W+" ");
            }
        info("");
    }
    else
    if( s.compareTo("mean-queue-time")==0 )
    {    info("Meam Queue Time[Wq]: ");
        for( int i=0; i<cellc; i++ )
            if( cell[i] instanceof QBox )
            {    QBox   qb    = ((QBox)cell[i]).dumpQBox();
                int    ns    = qb.r_numserv();
                float  sum   = 0;
                for( int j=0; j<ns; j++ )
                {    sum    += qb.r_cust_residents(j);
                }
                Float  c     = new Float(qb.r_serv_busytimes());
                Float  completion = new Float(qb.depart);
                float f      = c.floatValue()/qb.r_numserv();
                float  Ts    = c.floatValue()/completion.floatValue();
                float W = sum/completion.floatValue();
                float Wq= W-Ts;
                info(qb.name()+"="+Wq+" ");
            }
        info("");
    }
    else
    if( s.compareTo("mean-number-in-system")==0 )
```

```
        {     info("Mean Number In System[L]: ");
              for( int i=0; i<cellc; i++ )
                    if( cell[i] instanceof QBox )
                    {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                          int    ns     = qb.r_numserv();
                          float  sum    = 0;
                          for( int j=0; j<ns; j++ )
                          {     sum    += qb.r_cust_residents(j);
                          }
                          float L=sum/time;
                          info(qb.name()+"="+L+" ");
                    }
              info("");
        }
        else
        if( s.compareTo("mean-number-in-queue")==0 )
        {     info("Mean Number In Queue[Lq]: ");
              for( int i=0; i<cellc; i++ )
                    if( cell[i] instanceof QBox )
                    {     QBox   qb     = ((QBox)cell[i]).dumpQBox();
                          int    ns     = qb.r_numserv();
                          float  sum    = 0;
                          Float  c      = new Float(qb.r_serv_busytimes());
                          float f       = c.floatValue()/qb.r_numserv();
                          float  U      = c.floatValue()/time;
                          for( int j=0; j<ns; j++ )
                          {     sum    += qb.r_cust_residents(j);
                          }
                          float L=sum/time;
                          float Lq=L-U;
                          info(qb.name()+"="+Lq+" ");
                    }
              info("");
        }
        else
if( s.compareTo("load")==0 )
 load_file( p1 );
else
if( s.compareTo("save")==0 )
 save_file( p1 );
else
              {     info(" WARNING: unknow command '"+s+"'");
                    return  false;
              }
              return  true;
     }

     public void save_file( String filename )
     { try
{ FileWriter  out = new FileWriter( filename );
               out.write("; Queue Simulate File\n");
              out.write("; ----- Generator version 0.0.1 -----\n");
              out.write("; Number of queue(s) in Network = "+cellc+"\n");
               out.write("; Number of link(s) in Network  = "+gatec+"\n");
              out.write("debugc\t"+lc_debug+"\t;debug info counter\n");
              for( int i=0; i<cellc; i++ )
 { out.write( "locate "+cell[i].x+" "+cell[i].y+"\n");
                    out.write( cell[i].write()+"\n" );
```

```
    }
                for( int i=0; i<gatec; i++ )
                { out.write( "locate "+gate[i].x+" "+gate[i].y+"\n");
        out.write( gate[i].write()+"\n" );
    }
                 out.write("; end of file");
     out.close();
    }
    catch( Exception e )
    {     System.out.println("Can't load input file "+e);
                }
        }

        private int      lc_debug;

// Pannel control functions

        private void adjGate( int gidx )
        {       if( (gidx>=gatec)||(gidx<0) )
                        return;
                Gate    gatei   = gate[gidx].dump();
                int     src[]   = gatei.get_source();
                int     dst[]   = gatei.get_dest();
        int     x       = 0,
                        y       = 0,
                        w       = 0,
                        h       = 0;
                for( int i=0; i<src.length; i++ )
                {       RCell  csrc    = cell[src[i]].dump();
                for( int j=0; j<dst.length; j++ )
                        {       RCell  cdst    = cell[dst[j]].dump();
                        int     dx      = csrc.x-cdst.x,
                                        dy      = csrc.y-cdst.y;
                        if( Math.abs(dx) > w )
                                {       x       = csrc.x + dx/2;
                                        w       = (int)Math.abs(dx);
                                }
                                if( Math.abs(dy) > h )
                                {       y       = csrc.y + dy/2;
                                        h       = (int)Math.abs(dy);
                                }
                        }
                }
    if (x<0)
    x=(int)Math.abs(x);
                gatei.x = x;
    if(y<0)
    y=(int)Math.abs(y);
                gatei.y = y;

        }

    private  int dum_x,dum_y;
    private  boolean dum_mode  = false,
                     dum_c     = false;

    public void dumy_mode(boolean mode)
    { if( dum_mode!=mode )
```

```
    dum_c = false;
   dum_mode  = mode;
  }

  public int dumy_getx()
  { if( dum_c )
     return  dum_x;
   return  -1;
  }

  public int dumy_gety()
  { if( dum_c )
     return  dum_y;
   return  -1;
  }

  Image offscreen = null;
  Dimension offscreensize;
  Graphics og;

        public synchronized void paint( Graphics gg )
  { update(gg);
  }

        public synchronized void /*paint*/update( Graphics gg )
        {       Dimension d   = getSize();
        if ((offscreen == null) || (d.width != offscreensize.width) || (d.height !=
offscreensize.height))
    { offscreen = createImage(d.width, d.height);
            offscreensize = d;
            og = offscreen.getGraphics();
            og.setFont(getFont());
    }
                FontMetrics fm          = gg.getFontMetrics();
                Color   bgcolor         = getBackground();

    og.setFont(getFont());
                og.setColor( bgcolor );
                og.fillRect(0, 0, d.width, d.height);
    og.setColor( Color.gray ); //draw grid line
    for (int x=20; x < d.width; x+=20)
      og.drawLine(x,0,x,d.height);
    for (int y=20; y < d.height; y+=20)
      og.drawLine(0,y,d.width,y);

                for (int i=0; i<gatec; i++)
                {       Gate    gatei   = gate[i].dump();
                        int     src[]   = gatei.get_source();
                        int     dst[]   = gatei.get_dest();
     int prob[] = gatei.get_prob();
                        og.setColor( Color.black );
                        for( int ii=0; ii<src.length; ii++ )
                                og.drawLine(cell[src[ii]].x, cell[src[ii]].y, gatei.x,
  gatei.y);
      String lbl_prob;
                        for( int ii=0; ii<dst.length; ii++ ){
                                og.drawLine(cell[dst[ii]].x, cell[dst[ii]].y, gatei.x,
  gatei.y);
```

```
    if(prob[ii]>0){
    lbl_prob=Integer.toString(prob[ii]);
    og.drawString(lbl_prob,(cell[dst[ii]].x+gatei.x)/2,(cell[dst[ii]].y+gatei.y)/2);
    }
   }
                    String  lbl     = gatei.name();
                    og.setColor( gatei.pick ? Color.yellow : bgcolor );
                    int     fmw     = fm.stringWidth(lbl) + 10;
                            fmh     = fm.getHeight() + 4;
                            gatei.w= fmw;
                    og.fillRect(gatei.x-fmw/2, gatei.y-fmh/2, fmw, fmh);
                    og.setColor(Color.black);
                    og.drawRect(gatei.x-fmw/2, gatei.y-fmh/2, fmw-1, fmh-1);
                    og.drawString(lbl, gatei.x-(fmw-10)/2, (gatei.y-(fmh-4)/2) +
fm.getAscent());

                    if( gatei.info )
                    {       //lbl     = gate[i].count;
                            og.setColor( Color.black );
                            og.drawString( gatei.toString(), gatei.x-(fmw-10)/2,
gatei.y-2-(fmh-4)/2 );
                    }
            }
            for (int i=0; i<cellc; i++)
                    cell[i].paint(og, fm);

  gg.drawImage(offscreen, 0, 0, null);
    }

    int     fmh;            // fm height cache
    RCell   pickc;
    Gate    pickg;

    public synchronized boolean mouseDown(Event evt, int x, int y)
    { if( pickc!= null )
            {       pickc.pick      = false;
                    pickc   = null;
            }
            if( pickg!= null )
            {       pickg.pick      = false;
                    pickg   = null;
            }
            int     fmh2    = fmh/2;
      for( int i=0; i<cellc; i++ )
            {       RCell   c       = cell[i].dump();
                    int     x1      = c.x-c.w/2,
                            y1      = c.y-fmh2,
                            x2      = c.x+c.w/2,
                            y2      = c.y+fmh2;
                    if( x>=x1&&x<=x2&&y>=y1&&y<=y2 )
                    { if( dum_mode )
      parent.dummy(i,-1);
    else
    { c.pick  = true;
      pickc   = c.dump();
                            repaint();
      parent.status( c.toString() );
      if( pickc.types.compareTo("QBox")==0 )
```

```
        parent.pick( (QBox)pickc );
    }
                        return  true;
                }
            }
            for( int i=0; i<gatec; i++ )
            {       Gate    g       = gate[i].dump();
                    int     x1      = g.x-g.w/2,
                            y1      = g.y-fmh2,
                            x2      = g.x+g.w/2,
                            y2      = g.y+fmh2;
                    if( x>=x1&&x<=x2&&y>=y1&&y<=y2 )
                    { if( dum_mode )
     parent.dummy(-1,i);
    else
    { g.pick  = true;
     pickg    = g.dump();
                        repaint();
     parent.status( g.toString() );
    }
                        return  true;
                    }
            }

if( dum_mode )
{ dum_x = x;
 dum_y = y;
 dum_c = true;
 parent.dummy( x, y );
 return true;
}
            return true;
        }

        public synchronized boolean mouseDrag(Event evt, int x, int y)
        {       if( pickc!=null )
                {       pickc.x= x;
                        pickc.y = y;
                }
                else
                if( pickg!=null )
                {       pickg.x= x;
                        pickg.y= y;
                }
                else
                        return  true;
                repaint();
                return  true;
        }

        public synchronized boolean mouseUp(Event evt, int x, int y)
        {       if( pickc!=null )
                {       pickc.x= x;
                        pickc.y= y;
                        pickc.pick      = false;
                        pickc   = null;
                }
                else
```

```
                if( pickg!=null )
                {       pickg.x= x;
                        pickg.y= y;
                        pickg.pick      = false;
                        pickg   = null;
                }
                else
                        return  true;
                repaint();
                return  true;
        }
}
//-------------------------------------------------------------------------------------------
class QTGraph extends Canvas
{ QBox source   = null;
  int timeSpace = 0,
      peak      = 0,
      time;
  int state[];

  public void setSource(QBox source)
  { this.source = source;
  }

  public void setTimeSpace(int timeSpace)
  { this.timeSpace  = QSimPanel.timeSpace;
    /*this.timeSpace  = timeSpace;
    state = new int[timeSpace];*/
  }

  public void dump(int time)
  { //State ss;
    this.time = time;
    this.state= source.gstate;
/*    if( time < QSimPanel.timeSpace )
      System.out.print("length="+time+"[.."+state[time-2]+","+state[time-1]+"]");
    else
      System.out.print("length="+time+"[.."+state[QSimPanel.timeSpace-2]+","+state
[QSimPanel.timeSpace-1]+"]");*/
    //int startt  = time - timeSpace;
    peak  = 0;
    for( int i=0; i<QSimPanel.timeSpace; i++)
      if( peak < state[i] )
        peak  = state[i];
    System.out.println(" peak="+peak);
/*    int cc,count;
    for( peak=0,cc=0; cc<timeSpace; cc++)
      state[cc] = 0;
    for( cc=source.server.length-1; cc>=0; cc--)
    { ss  = source.server[cc].state();
      count = ss.get()-1;
      System.out.print("\ndump().count="+count+", startt="+startt);
      for( ; count>=0; count--)
      { System.out.print(" "+ss.servi[count]);
        if( ss.servi[count] >= startt )
        { int ccc = ss.servi[count] - startt;
          if( ccc < timeSpace )
          { state[ccc]++;
```

```
        if( state[ccc]>peak )
          peak  = state[ccc];
       }
      }
      else
        break;
    }
   }*/
  }

        public synchronized void paint( Graphics g )
        {       Dimension d   = getSize();
                FontMetrics fm        = g.getFontMetrics();
                Color   bgcolor       = getBackground();
  int hh  = d.height - 25;
  Integer h1  = new Integer(hh),
        h2  = new Integer(peak);

  float highRatio = h1.floatValue() / h2.floatValue();
  h1  = new Integer(d.width - 10);
  h2  = new Integer(timeSpace);
  float widthRatio= h1.floatValue() / h2.floatValue();

  g.setFont(getFont());
                //g.setColor( bgcolor );//draw graph area
  g.setColor( Color.white );
                g.fillRect(0, 0, d.width, d.height);

  int lastv = 0;
  for(int x=0; x<timeSpace; x++)
  { Integer ss  = new Integer(state[x]);
   Float h = new Float(highRatio * ss.floatValue() + 0.5),
        w1= new Float(x*widthRatio+0.5),
        w2= new Float(widthRatio);

   g.setColor( Color.gray );
   g.fillRect(w1.intValue()+20,hh-h.intValue()+15,w2.intValue()+1,h.intValue());
   if( ss.intValue()!=lastv )
   { g.setColor( Color.black );
    g.drawString(ss.toString(), w1.intValue()+20, hh-h.intValue()+10);
    lastv = ss.intValue();
   }
  }
  g.setColor( Color.black );
  g.drawLine(20,5,20,d.height-10);
  g.drawLine(20,d.height-10,d.width,d.height-10);
  if( timeSpace > 0 )
  { Integer s1  = new Integer(time);//draw label of max. time
   g.drawString("time="+s1.toString(),d.width - 40, d.height - 4);    }
 }
}
```

```
//Title:       Queueing Simulation Program
//Version:     1.0.0
//Copyright:   Copyright (c) 1998
//Author:      Boonyong Kaewbuddee
//Company:     KMITL
//Description: This program is a part of the master thesis.
//File         QsimControl.java

package QSim;

public interface QSimControl
{ public abstract void status(String msg);
  public abstract void message(String msg);
  public abstract void dummy(int x, int y);
  public abstract void pick(QBox cell);
}
```

```
//Title:       Queueing Simulation Program
//Version:     1.0.0
//Copyright:   Copyright (c) 1998
//Author:      Boonyong Kaewbuddee
//Company:     KMITL
//Description: This program is a part of the master thesis.
//File         QsimApplet.java

package QSim;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;
import java.util.StringTokenizer;

//-----------------------------------------------------------------------------------------------
  public class QSimApplet extends JApplet implements QSimControl{
  final Font f = new Font("TimesRoman",Font.BOLD,12);
  boolean isStandalone = false;
  String initFile;
  int editWidth = 640, editHeight = 400;

  CheckboxGroup cbType = new CheckboxGroup();
  Panel panel2 = new Panel();
  Panel panel3 = new Panel();
  Checkbox cbCQbox = new Checkbox();
  Checkbox cbCSourceExp = new Checkbox();
  Checkbox cbCSourceNor = new Checkbox();
  Checkbox cbCSourceUni = new Checkbox();
  Checkbox cbMove = new Checkbox();
  Checkbox cbCSink = new Checkbox();
  Checkbox cbLinkP = new Checkbox();
  Checkbox cbLinkR = new Checkbox();
  Checkbox cbLinkS = new Checkbox();
  Checkbox cbDel = new Checkbox();

  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  ScrollPane scrollPane1 = new ScrollPane();
  QSimPanel qSimPanel = new QSimPanel(256, this);
  QTGraph qgraph  = new QTGraph();
  int timespace  = 100;

  Panel panel1 = new Panel();

  TextArea textInfo = new TextArea(13,40);
  TextArea textCell = new TextArea(4,40);
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
```

```
  Panel panel5 = new Panel();

  Label label1 = new Label();
  TextField textCmd = new TextField(30);
  FileDialog fileLoad, fileSave;
  Button btCmd = new Button();

//Get a parameter value
  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  //Construct the applet
  public QSimApplet() {
  }

//Initialize the applet
  public void init() {
    try { initFile = this.getParameter("InitFile", ""); } catch (Exception e) {
e.printStackTrace(); }
    try {
    jbInit();
    }
    catch (Exception e) {
    e.printStackTrace();
    }
  }
  //static {
  //  try {
  //    //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.metal.MetalLookAndFeel());
  //    //UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.motif.MotifLookAndFeel());
  //    UIManager.setLookAndFeel(new
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
  //  }
  //  catch (Exception e) {}
  //}

//Component initialization
  private void jbInit() throws Exception {
    panel3.setLayout(flowLayout2);
    panel3.setFont(new Font("Helvetica", 0, 8));
    cbCQbox.setLabel("QBox");
    cbCQbox.setCheckboxGroup(cbType);

    cbCQbox.addMouseListener(new java.awt.event.MouseAdapter()
    {
```

```
  public void mouseClicked(MouseEvent e)
  {
   cbCQbox_mouseClicked(e);
  }
});
cbCSourceExp.setCheckboxGroup(cbType);
cbCSourceExp.addMouseListener(new java.awt.event.MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
   cbCSourceExp_mouseClicked(e);
  }
});
cbCSourceUni.setCheckboxGroup(cbType);
cbCSourceUni.addMouseListener(new java.awt.event.MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
   cbCSourceUni_mouseClicked(e);
  }
});
cbCSourceNor.setCheckboxGroup(cbType);

cbCSourceNor.addMouseListener(new java.awt.event.MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
   cbCSourceNor_mouseClicked(e);
  }
});
cbMove.setLabel("Move");
cbMove.setCheckboxGroup(cbType);

cbMove.addMouseListener(new java.awt.event.MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
   cbMove_mouseClicked(e);
  }
});
cbCSink.setCheckboxGroup(cbType);

cbCSink.addMouseListener(new java.awt.event.MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
   cbCSink_mouseClicked(e);
  }
});
cbLinkS.setCheckboxGroup(cbType);
```

```
cbLinkS.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  cbLinkS_mouseClicked(e);
 }
});
cbLinkR.setCheckboxGroup(cbType);

cbLinkR.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  cbLinkR_mouseClicked(e);
 }
});
cbLinkP.setCheckboxGroup(cbType);

cbLinkP.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  cbLinkP_mouseClicked(e);
 }
});
cbDel.setCheckboxGroup(cbType);

cbDel.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  cbDel_mouseClicked(e);
 }
});
qSimPanel.setBackground(Color.white);
qSimPanel.setBounds(new Rectangle(0, 39, 326, 384));
qgraph.setBounds(0,0,640,80);
panel1.setLayout(verticalFlowLayout2);
textInfo.setFont(f);
textInfo.setBackground(Color.white);
textInfo.setText("Simulated information:");
textCell.setFont(new Font("TimesRoman", 0, 12));
textCell.setBackground(Color.gray);
textCell.setText("Cell Status:");
panel5.setLayout(flowLayout1);
label1.setText("Command:");
textCmd.setFont(new Font("TimesRoman", 0, 12));
textCmd.setBackground(Color.white);
textCmd.setText("Command here");
cbLinkP.setLabel("PLink");
```

```
cbLinkR.setLabel("RLink");
cbLinkS.setLabel("SLink");
cbCSink.setLabel("Sink");
cbMove.setState(true);
cbCSourceExp.setLabel("SupExp");
cbCSourceNor.setLabel("SupNor");
cbCSourceUni.setLabel("SupUni");
cbDel.setLabel("Delete");
panel2.setLayout(verticalFlowLayout1);
this.setSize(640,560);
btCmd.setActionCommand("DoCmd");
panel6.setLocale(new java.util.Locale("th", "", ""));
panel6.setLayout(gridLayout1);
panel6.setFont(new Font("Helvetica", 0, 8));
btHelp.setLabel("Help");
btNew.setLabel("New");
btNew.setActionCommand("New");
btLoad.setLabel("Load");
btSave.setLabel("Save");
btRun.setLabel("Run");
numLoop.setBackground(Color.white);
numLoop.setText("1");

btSave.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  btSave_mouseClicked(e);
 }
});
btLoad.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  btLoad_mouseClicked(e);
 }
});
btNew.addMouseListener(new java.awt.event.MouseAdapter()
{
 public void mouseClicked(MouseEvent e)
 {
  btNew_mouseClicked(e);
 }
});
this.getContentPane().add(panel2, BorderLayout.SOUTH);
panel2.add(panel3, null);
panel3.add(numLoop, null);
panel3.add(btRun, null);
panel3.add(btSave, null);
panel3.add(btLoad, null);
```

```
    panel3.add(btNew, null);
    panel3.add(btHelp, null);
    panel3.add(cbMove, null);
    panel3.add(cbCSourceExp, null);
    panel3.add(cbCSourceNor, null);
    panel3.add(cbCSourceUni, null);
    panel3.add(cbCSink, null);
    panel3.add(cbCQbox, null);
    panel3.add(cbLinkP, null);
    panel3.add(cbLinkR, null);
    panel3.add(cbLinkS, null);
    panel3.add(cbDel, null);
    panel2.add(panel6, null);
    this.getContentPane().add(scrollPane1, BorderLayout.CENTER);
    scrollPane1.add(qSimPanel, null);
    this.getContentPane().add(panel1, BorderLayout.EAST);
    panel1.add(panel5, null);
    panel5.add(label1, null);
    panel5.add(textCmd, null);
    panel5.add(btCmd, null);
    panel6.add(qgraph, null);
    panel1.add(textInfo, null);
    panel1.add(textCell, null);

//    Frame frame = this.frame;
//    fileLoad  = new FileDialog(null,"Load sim file",FileDialog.LOAD);
//    fileSave  = new FileDialog(null,"Save sim file",FileDialog.SAVE);
    btCmd.addMouseListener(new java.awt.event.MouseAdapter()
    {
     public void mouseClicked(MouseEvent e)
     {
      btCmd_mouseClicked(e);
     }
    });
    btCmd.setLabel("Do");

    if( initFile.length() > 0)
    { qSimPanel.load_file( initFile );
     textInfo.setText( qSimPanel.getMsgInfo() );
    }
    qSimPanel.repaint();
  }

//Start the applet

  public void start() {
  }
//Stop the applet

  public void stop() {
```

```
  }

//Destroy the applet
  public void destroy() {
  }

//Get Applet information
  public String getAppletInfo() {
    return "Applet Information\n" +
          "initFile = " + initFile + "\n" +
          "Sim = " + qSimPanel;
  }

//Get parameter info
  public String[][] getParameterInfo() {
    String pinfo[][] =
    {
      {"initFile", "String", "Initial .sim file to be load on start."},
      {"editWidth", "int", "Initial width of edit area."},
      {"editHeight","int", "Initial height of edit area."},
    };
    return pinfo;
  }

  private int mode = 0;
  Panel panel6 = new Panel();
  FlowLayout flowLayout1 = new FlowLayout();
  Button btHelp = new Button();
  Button btNew = new Button();
  Button btLoad = new Button();
  Button btSave = new Button();
  Button btRun = new Button();
  TextField numLoop = new TextField();
  FlowLayout flowLayout2 = new FlowLayout();
  GridLayout gridLayout1 = new GridLayout();
  MenuBar menuBar1 = new MenuBar();
  MenuBar menuBar2 = new MenuBar();

  public void pick(QBox cell)
  { if( cell.types.compareTo("QBox")==0 )
    { qgraph.setSource(cell);
      qgraph.setTimeSpace(timespace);
      qgraph.dump(qSimPanel.time);
      qgraph.repaint();
    }
  }

  public void status(String msg)
  { textCell.setText(msg);
  }
```

```
public void message(String msg)
{ textInfo.setText(msg);
}

public void dummy(int x, int y)
{ if( mode==9 )
  { if( y<0 )
    { // confirm;
      System.out.println("delete cell:"+x);
      System.out.println("ret="+qSimPanel.del(x));
      qSimPanel.repaint();
    }
    if( x<0 )
    { // confirm;
      System.out.println("delete link:"+y);
      qSimPanel.rlink(y);
      qSimPanel.repaint();
    }
   return;
  }
  if( x<0||y<0 )
   return;
  qSimPanel.locatei(x,y);
  switch( mode )
  { case 1: textCmd.setText( "supplier-exp <n>" );
            break;
    case 2: textCmd.setText( "supplier-nor <n>" );
            break;
    case 3: textCmd.setText( "supplier-uni <low> <high>" );
            break;
    case 4: textCmd.setText( "terminater" );
            break;
    case 5: textCmd.setText( "qbox <n n ..>" );
            break;
    case 6: textCmd.setText( "link probability <n ..> to <n ..> with <p ..>" );
            break;
    case 7: textCmd.setText( "link random <n ..> to <n ..>" );
            break;
    case 8: textCmd.setText( "link sequence <n ..> to <n ..>" );
  }
}

void btLoad_mouseClicked(MouseEvent e)
{ fileLoad.show();
  String fname = fileLoad.getFile();
  if( fname!=null )
    qSimPanel.load_file( fname );
}
```

```
void btSave_mouseClicked(MouseEvent e)
{ fileSave.show();
  String fname  = fileSave.getFile();
  if( fname!=null )
    qSimPanel.save_file( fname );
}

void btCmd_mouseClicked(MouseEvent e)
{ StringTokenizer st = new StringTokenizer( textCmd.getText() );
  if( st.countTokens()>0 )
  { String  cmd = st.nextToken(),
            par = "";
    if( st.hasMoreTokens() )
    { par = st.nextToken();
      for(; st.hasMoreTokens(); par+=" "+st.nextToken());
    }
    qSimPanel.cmdline(cmd,par);
    qSimPanel.repaint();
  }
  textCmd.setText("");
  textInfo.setText( qSimPanel.getMsgInfo() );
}

void cbMove_mouseClicked(MouseEvent e)
{ mode  = 0;
  qSimPanel.dumy_mode( false );
  textCmd.setText("");
}

void cbCSourceExp_mouseClicked(MouseEvent e)
{ qSimPanel.dumy_mode( true );
  mode  = 1;
}
void cbCSourceNor_mouseClicked(MouseEvent e)
{ qSimPanel.dumy_mode( true );
  mode  = 2;
}
void cbCSourceUni_mouseClicked(MouseEvent e)
{ qSimPanel.dumy_mode( true );
  mode  = 3;
}
void cbCSink_mouseClicked(MouseEvent e)
{ qSimPanel.dumy_mode( true );
  mode  = 4;
}
void cbCQbox_mouseClicked(MouseEvent e)
{ qSimPanel.dumy_mode( true );
  mode  = 5;
}
void cbLinkP_mouseClicked(MouseEvent e)
```

```
 { qSimPanel.dumy_mode( true );
  mode  = 6;
 }
 void cbLinkR_mouseClicked(MouseEvent e)
 { qSimPanel.dumy_mode( true );
  mode  = 7;
 }
 void cbLinkS_mouseClicked(MouseEvent e)
 { qSimPanel.dumy_mode( true );
  mode  = 8;
 }
 void cbDel_mouseClicked(MouseEvent e)
 { qSimPanel.dumy_mode( true );
  mode  = 9;
  System.out.println("delete mode");
 }

 void btNew_mouseClicked(MouseEvent e)
 { qSimPanel.newPanel(256);
  qSimPanel.repaint();
 }
}
//The end of program.
```

โปรแกรมบรรจุในแผ่นดิสก์

โปรแกรมนี้พัฒนาด้วย Borland JBuilder 2.0 Client/Server Suite

Project name: Qsim.jpr

ภายใต้ Project name จะมีองค์ประกอบอยู่หลายส่วนคือ

1) HTML เพื่อเรียกรันโปรแกรมและเขียนเอกสาร
2) แฟ้ม โปรแกรม .java อาจมีได้หลายแฟ้ม
3) ฟังก์ชันของภาษาจาวา ที่โปรแกรมนี้เรียกใช้

การเปิดโปรแกรมให้เปิด Qsim.jpr แล้วผู้ใช้จะมองเห็นองค์ประกอบทุกอย่างในมุมมองเหมือน explore

ประวัติผู้เขียน

นายบุญยงค์ แก้วบุดดี เกิดเมื่อวันที่ 3 กันยายน 2512 ที่จังหวัดขอนแก่น สำเร็จการศึกษาระดับปริญญาตรี วิศวกรรมคอมพิวเตอร์ จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2534

ปีพุทธศักราช 2535 เข้าทำการในตำแหน่งวิศวกรระดับ 4 การไฟฟ้าฝ่ายผลิตแห่งประเทศไทย อ. บางกรวย จ. นนทบุรี และปัจจุบันดำรงตำแหน่งวิศวกรระดับ 6 สังกัดฝ่ายเทคโนโลยีสารสนเทศ ในตำแหน่ง System Programmer รับผิดชอบดูแลระบบคอมพิวเตอร์ไอบีเอ็มเมนเฟรมและเครือข่ายคอมพิวเตอร์ นอกจากนี้ยังทำหน้าเป็นวิทยากรบรรยายหลักสูตรคอมพิวเตอร์ให้กับฝ่ายเทคโนโลยีสารสนเทศและฝ่ายฝึกอบรม