package edu.buaa.sei.datamodel;

import edu.buaa.sei.run.Publisher;

public class Dependency {

private Publisher srcPublisher;

private Publisher dstPublisher;

private SendData sendData;

private double time;

private double reliability;

private LostPackage lostPackage;

public double getReliability() {

return reliability;

}

public void setReliability(double reliability) {

this.reliability = reliability;

}

public SendData getSendData() {

return sendData;

}

public void setSendData(SendData sendData) {

this.sendData = sendData;

}

public double getTime() {

return time;

}

public void setTime(double time) {

this.time = time;

}

public LostPackage getLostPackage() {

return lostPackage;

}

public void setLostPackage(LostPackage lostPackage) {

this.lostPackage = lostPackage;

}

public Publisher getSrcPublisher() {

return srcPublisher;

}

public void setSrcPublisher(Publisher srcPublisher) {

this.srcPublisher = srcPublisher;

}

public Publisher getDstPublisher() {

return dstPublisher;

}

public void setDstPublisher(Publisher dstPublisher) {

this.dstPublisher = dstPublisher;

}

}

package edu.buaa.sei.datamodel;

public class LostPackage {

private int firstLostPackage;

private int secondLostPackage;

private String baseDependencyId;

public int getFirstLostPackage() {

return firstLostPackage;

}

public void setFirstLostPackage(int firstLostPackage) {

this.firstLostPackage = firstLostPackage;

}

public int getSecondLostPackage() {

return secondLostPackage;

}

public void setSecondLostPackage(int secondLostPackage) {

this.secondLostPackage = secondLostPackage;

}

public String getBaseDependencyId() {

return baseDependencyId;

}

public void setBaseDependencyId(String baseDependencyId) {

this.baseDependencyId = baseDependencyId;

}

}

package edu.buaa.sei.datamodel;

public class Message {

public String title;

public String name;

public String id;

public double time;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public double getTime() {

return time;

}

public void setTime(double time) {

this.time = time;

}

}

package edu.buaa.sei.datamodel;

import java.util.ArrayList;

public class Process {

public String name;

public ArrayList<String> threads = new ArrayList<String>();

private double needTime;

private double timeUsed;

public void addThreads(String str) {

threads.add(str);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public ArrayList<String> getThreads() {

return threads;

}

public void setThreads(ArrayList<String> threads) {

this.threads = threads;

}

public double getNeedTime() {

return needTime;

}

public void setNeedTime(double needTime) {

this.needTime = needTime;

}

public double getTimeUsed() {

return timeUsed;

}

public void setTimeUsed(double timeUsed) {

this.timeUsed = timeUsed;

}

}

package edu.buaa.sei.datamodel;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.utils.StringHandle;

public class Receiver {

private ArrayList<Message> messageList = new ArrayList<Message>();

public double getFIFOSendTime() {

System.out.println("calaulating time from message list, which size is "

+ messageList.size());

double time = 0;

for (int i = 0; i < messageList.size(); i++) {

time += messageList.get(i).getTime();

}

System.out.println("FIFO: time used " + time + "ms.");

return time;

}

public boolean containDumplicate(Message m) {

for (int i = 0; i < messageList.size(); i++) {

if (messageList.get(i).title.compareTo(m.title) == 0)

return true;

}

return false;

}

private Message findTimeById(String id, NodeList list) {

for (int temp = 0; temp < list.getLength(); temp++) {

Node nNode = (Node) list.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String timeId = eElement.getAttribute("base\_NamedElement");

if (timeId.equals(id)) {

Message mes = new Message();

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("execTime")) {

String timeStr = node.getFirstChild()

.getNodeValue();

String timeStr1 = StringHandle

.delUnusedStr(timeStr);

String[] str = timeStr1.split(",");

if (str.length < 2)

continue;

mes.time = Double.valueOf(str[0]);

return mes;

}

}

}

}

}

}

return null;

}

public void getReveiver(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("message");

NodeList timeList = doc.getElementsByTagName("GQAM:GaStep");

// System.out.println("Reveiver message count : " + nList.getLength());

messageList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String type = eElement.getAttribute("xmi:type");

if (type.equals("uml:Message")) {

Message mes = new Message();

String nameStr = eElement.getAttribute("name");

String[] strList = nameStr.split(": ");

if (strList.length < 2)

continue;

mes.title = strList[0];

mes.name = strList[1];

mes.id = eElement.getAttribute("xmi:id");

Message mes\_t = findTimeById(mes.id, timeList);

if (mes\_t == null)

continue;

mes.time = mes\_t.time;

if (containDumplicate(mes))

continue;

// System.out.println("title: " + mes.title + ", name: "

// + mes.name + ", time: " + mes.time);

messageList.add(mes);

}

}

}

// System.out.println("valid count : " + messageList.size());

}

}

package edu.buaa.sei.datamodel;

public class SendData {

private int DataNum;

private int DataSize;//kb

private String dependencyDstId;

public String getDependencyDstId() {

return dependencyDstId;

}

public void setDependencyDstId(String dependencyDstId) {

this.dependencyDstId = dependencyDstId;

}

public int getDataNum() {

return DataNum;

}

public void setDataNum(int dataNum) {

DataNum = dataNum;

}

public int getDataSize() {

return DataSize;

}

public void setDataSize(int dataSize) {

DataSize = dataSize;

}

}

package edu.buaa.sei.datamodel;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.utils.StringHandle;

public class Sender {

private ArrayList<Message> messageList = new ArrayList<Message>();

public double getFIFOSendTime() {

System.out.println("calaulating time from message list, which size is "

+ messageList.size());

double time = 0;

for (int i = 0; i < messageList.size(); i++) {

time += messageList.get(i).getTime();

}

System.out.println("FIFO: time used " + time + "ms.");

return time;

}

public ArrayList<Double> getTimeTableTime(int threads, int timeTableUnit) {

ArrayList<Double> timeList = new ArrayList<Double>();

ArrayList<TimeTable> timeTableList = new ArrayList<TimeTable>();

int threads\_t = threads;

double neededTime = getFIFOSendTime();

for (int i = 0; i < threads; i++) {

TimeTable timeTable = new TimeTable();

timeTable.setTimeUsed(0);

timeTable.setNeededTime(neededTime);

timeTableList.add(timeTable);

}

double timeNow = 0;

double timePerProcess;

while (threads\_t > 0) {

timePerProcess = (double)timeTableUnit/(double)threads\_t;//ms

for (int i = 0; i < timeTableList.size(); i++) {

if (timeTableList.get(i).getNeededTime() <= 0) {//this process done.

continue;

} else if (timeTableList.get(i).getNeededTime() > timePerProcess) {

timeTableList.get(i).setNeededTime(timeTableList.get(i).getNeededTime() - timePerProcess);

timeNow += timePerProcess;

} else {

timeNow += timeTableList.get(i).getNeededTime();

timeTableList.get(i).setNeededTime(0);

timeTableList.get(i).setTimeUsed(timeNow);

threads\_t--;

}

}

}//while

for (int i = 0; i < threads; i++) {

timeList.add(timeTableList.get(i).getTimeUsed());

}

return timeList;

}

public boolean containDumplicate(Message m) {

for (int i = 0; i < messageList.size(); i++) {

if (messageList.get(i).title.compareTo(m.title) == 0)

return true;

}

return false;

}

private Message findTimeById(String id, NodeList list) {

for (int temp = 0; temp < list.getLength(); temp++) {

Node nNode = (Node) list.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String timeId = eElement.getAttribute("base\_NamedElement");

if (timeId.equals(id)) {

Message mes = new Message();

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("execTime")) {

String timeStr = node.getFirstChild()

.getNodeValue();

String timeStr1 = StringHandle

.delUnusedStr(timeStr);

String[] str = timeStr1.split(",");

if (str.length < 2)

continue;

mes.time = Double.valueOf(str[0]);

return mes;

}

}

}

}

}

}

return null;

}

public void getSender(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("message");

NodeList timeList = doc.getElementsByTagName("GQAM:GaStep");

// System.out.println("Sender message count : " + nList.getLength());

messageList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String type = eElement.getAttribute("xmi:type");

if (type.equals("uml:Message")) {

Message mes = new Message();

String nameStr = eElement.getAttribute("name");

String[] strList = nameStr.split(": ");

if (strList.length < 2)

continue;

mes.title = strList[0];

mes.name = strList[1];

mes.id = eElement.getAttribute("xmi:id");

Message mes\_t = findTimeById(mes.id, timeList);

if (mes\_t == null)

continue;

mes.time = mes\_t.time;

if (containDumplicate(mes))

continue;

// System.out.println("title: " + mes.title + ", name: "

// + mes.name + ", time: " + mes.time);

messageList.add(mes);

}

}

}

// System.out.println("valid count : " + messageList.size());

}

public ArrayList<Message> getMessageList() {

return messageList;

}

public void setMessageList(ArrayList<Message> messageList) {

this.messageList = messageList;

}

}

package edu.buaa.sei.datamodel;

public class TimeTable {

private double neededTime;

private double timeUsed;

public double getNeededTime() {

return neededTime;

}

public void setNeededTime(double neededTime) {

this.neededTime = neededTime;

}

public double getTimeUsed() {

return timeUsed;

}

public void setTimeUsed(double timeUsed) {

this.timeUsed = timeUsed;

}

}

package edu.buaa.sei.utils;

import java.util.Random;

public class RandomGenerator {

public static double getARandomNum(double min, double max) {

Random random = new Random();

double s = (random.nextDouble()\*max)%(max-min+1) + min;

return s;

}

public static double getARandomNumD(double min, double max) {

Random random = new Random();

double s = random.nextDouble()\*max;

while (s < min)

s = random.nextDouble()\*max;

return s;

}

}

package edu.buaa.sei.utils;

public class StringHandle {

public static String delUnusedStr(String str) {

String rstr = null;

for (int i = 0; i < str.length(); i++) {

if (str.charAt(i) == '{' || str.charAt(i) == '}' || str.charAt(i) == ' ') {

continue;

} else {

if (rstr == null)

rstr = String.*valueOf*(str.charAt(i));

else

rstr += str.charAt(i);

}

}

return rstr;

}

public static int getKbFromStr(String str) {

char metric = str.charAt(str.length()-1);

int multiSize = 0;

if (metric == 'M' || metric == 'm')

multiSize = 1024;

else if (metric == 'G' || metric == 'g')

multiSize = 1024\*1024;

else if (metric == 'K' || metric == 'k')

multiSize = 1;

else

System.*out*.println("no such metric : " + metric);

String num = str.substring(0, str.length()-1);

int rv = Integer.*valueOf*(num);

return rv\*multiSize;

}

}

package edu.buaa.sei.run;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.datamodel.Dependency;

import edu.buaa.sei.datamodel.LostPackage;

import edu.buaa.sei.datamodel.Message;

import edu.buaa.sei.datamodel.SendData;

import edu.buaa.sei.datamodel.Sender;

import edu.buaa.sei.utils.RandomGenerator;

import edu.buaa.sei.utils.StringHandle;

public class DDS {

ArrayList<Dependency> dependencyList = new ArrayList<Dependency>();

ArrayList<Publisher> publisherList = new ArrayList<Publisher>();

ArrayList <LostPackage> lostPackageList = new ArrayList <LostPackage>();

ArrayList <SendData> sendDataList = new ArrayList <SendData>();

public Publisher getPublisherByBonId(String sonId) {

for (int i = 0; i < publisherList.size(); i++) {

Publisher pub = publisherList.get(i);

for (int j = 0; j < pub.getSonId().size(); j++) {

String sonIdStr = pub.getSonId().get(j);

if (sonIdStr.equals(sonId)) {

return pub;

}

}

}

return null;

}

public LostPackage getLostPackageByDepId(String depID) {

for (int i = 0; i < lostPackageList.size(); i++) {

LostPackage lostPackage = lostPackageList.get(i);

if (lostPackage.getBaseDependencyId().equals(depID)) {

return lostPackage;

}

}

return null;

}

public SendData getSendDataByDepDstId(String depID) {

for (int i = 0; i < sendDataList.size(); i++) {

SendData sendData = sendDataList.get(i);

if (sendData.getDependencyDstId().equals(depID)) {

return sendData;

}

}

return null;

}

public void getAllDependency(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("packagedElement");

getALLPublisher(umlPath);

getAllLostPackage(umlPath);

getAllSendData(umlPath);

dependencyList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String type = eElement.getAttribute("xmi:type");

if (type.equals("uml:Dependency")) {

Dependency dep = new Dependency();

String srcId = eElement.getAttribute("client");

String dstId = eElement.getAttribute("supplier");

Publisher srcPub = getPublisherByBonId(srcId);

Publisher dstPub = getPublisherByBonId(dstId);

dep.setSrcPublisher(srcPub);

dep.setDstPublisher(dstPub);

String depId = eElement.getAttribute("xmi:id");

dep.setLostPackage(getLostPackageByDepId(depId));

dep.setSendData(getSendDataByDepDstId(srcPub.getPublisherId()));

dependencyList.add(dep);

}

}

}

}

public void printDependencyInfo() {

for (int i = 0; i < dependencyList.size(); i++) {

Dependency dep = dependencyList.get(i);

System.out.println(dep.getSrcPublisher().getPublisherName() + " --> " + dep.getDstPublisher().getPublisherName()

+ ", send data size: " + dep.getSendData().getDataSize() + "Kb, num: " + dep.getSendData().getDataNum()

+ ", lost package 1st: " + dep.getLostPackage().getFirstLostPackage()

+ ", 2nd: " + dep.getLostPackage().getSecondLostPackage()

+ ", time: " + dep.getTime() + ", reliability: " + dep.getReliability());

}

}

public void calculateDependency(Sender sender) {

//calculate send time.

//calculate transmite time.

for (int i = 0; i < dependencyList.size(); i++) {

Dependency dep = dependencyList.get(i);

}

}

public double recursiveGetTime(Publisher pub) {

double minTime = 99999999;

for (int i = 0; i < dependencyList.size(); i++) {

Dependency dep = dependencyList.get(i);

if (dep.getDstPublisher().getPublisherId().equals(pub.getPublisherId())) {

double curTime = dep.getTime() + recursiveGetTime(dep.getSrcPublisher());

if (curTime <= minTime) {

minTime = curTime;

}

}

}

if (minTime == 99999999)

return 0;

return minTime;

}

public double recursiveGetReliability(Publisher pub) {

double reliability = 1;

for (int i = 0; i < dependencyList.size(); i++) {

Dependency dep = dependencyList.get(i);

if (dep.getDstPublisher().getPublisherId().equals(pub.getPublisherId())) {

reliability \*= dep.getReliability() \* recursiveGetReliability(dep.getSrcPublisher());

}

}

return reliability;

}

public void calculateTime() {

ArrayList<Publisher> leafPublisher = getLeafPublisher();

for (int i = 0; i < leafPublisher.size(); i++) {

Publisher pub = leafPublisher.get(i);

double time = recursiveGetTime(pub);

System.out.printf("%s : %.3f ms.\n", pub.getPublisherName(), time);

}

}

public void calculateReliability() {

ArrayList<Publisher> leafPublisher = getLeafPublisher();

for (int i = 0; i < leafPublisher.size(); i++) {

Publisher pub = leafPublisher.get(i);

double reliability = recursiveGetReliability(pub);

System.out.printf("%s : %.3f%%.\n", pub.getPublisherName(), reliability\*100);

}

}

public void showResults() {

ArrayList<Publisher> leafPublisher = getLeafPublisher();

System.out.printf("\n%7s\t%10s\t%6s\n", "name", "reliability(%s)", "time(ms)");

for (int i = 0; i < leafPublisher.size(); i++) {

Publisher pub = leafPublisher.get(i);

double reliability = recursiveGetReliability(pub);

double time = recursiveGetTime(pub);

System.out.printf("%7s\t%10.3f\t%6.3f\n", pub.getPublisherName(), reliability\*100, time);

}

}

public ArrayList<Publisher> getRootPublisher() {

ArrayList<Publisher> rl = new ArrayList<Publisher>();

return rl;

}

public boolean isLeafPublisher(Publisher pub) {

for (int i = 0; i < dependencyList.size(); i++) {

Dependency dep = dependencyList.get(i);

if (dep.getSrcPublisher().getPublisherId().equals(pub.getPublisherId())) {

return false;

}

}

return true;

}

public ArrayList<Publisher> getLeafPublisher() {

ArrayList<Publisher> rl = new ArrayList<Publisher>();

for (int i = 0; i < publisherList.size(); i++) {

Publisher pub = publisherList.get(i);

if (isLeafPublisher(pub)) {

rl.add(pub);

}

}

return rl;

}

public void getAllLostPackage(String umlPath) throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("Profile:lostpackage");

lostPackageList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String baseDependencyId = eElement.getAttribute("base\_Dependency");

String firstLostPackageStr = eElement.getAttribute("FirstLostPackge");

String secondLostPackageStr = eElement.getAttribute("SecondLostPackage");

LostPackage lostPackage = new LostPackage();

lostPackage.setBaseDependencyId(baseDependencyId);

if (firstLostPackageStr != null) {

if (firstLostPackageStr.length() != 0)

lostPackage.setFirstLostPackage(Integer.valueOf(firstLostPackageStr));

}

if (secondLostPackageStr != null) {

if (secondLostPackageStr.length() != 0)

lostPackage.setSecondLostPackage(Integer.valueOf(secondLostPackageStr));

}

lostPackageList.add(lostPackage);

}

}

}

public void getAllSendData(String umlPath) throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("SW\_Interaction:MessageComResource");

sendDataList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String baseDependencyId = eElement.getAttribute("base\_Classifier");

Publisher pub = getPublisherByBonId(baseDependencyId);

SendData sendData = new SendData();

sendData.setDependencyDstId(pub.getPublisherId());

int size = 0;

int count = 0;

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("capacity")) {

String str = node.getFirstChild().getNodeValue();

String[] strList = str.split("\\\*");

if (strList.length < 2)

continue;

int count\_t = Integer.valueOf(strList[1]);

int sizeKB = StringHandle.getKbFromStr(strList[0]);

size += sizeKB\*count\_t;

count += count\_t;

}

}

}

sendData.setDataSize(size);

sendData.setDataNum(count);

sendDataList.add(sendData);

}

}

}

public void getALLPublisher(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("nestedClassifier");

publisherList.clear();

// scan xml and get all valid message.

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String type = eElement.getAttribute("xmi:type");

if (type.equals("uml:Device")) {

Dependency dep = new Dependency();

String id = eElement.getAttribute("xmi:id");

String name = eElement.getAttribute("name");

Publisher publisher = new Publisher();

publisher.setPublisherId(id);

publisher.setPublisherName(name);

ArrayList<String> sonIdList = new ArrayList<String>();

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("nestedClassifier")) {

if (((Element)node).getAttribute("xmi:type").equals("uml:Artifact")) {

String sonId = ((Element)node).getAttribute("xmi:id");

sonIdList.add(sonId);

}

}

}

}

publisher.setSonId(sonIdList);

publisherList.add(publisher);

}

}

}

}

public void calculateDependencyTime() {

}

public void calculateDDSTime() {

}

}

package edu.buaa.sei.run;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.datamodel.Message;

import edu.buaa.sei.datamodel.Process;

import edu.buaa.sei.utils.StringHandle;

public class NodeSend {

private String schedPolicy;

private ArrayList<Process> pList = new ArrayList<Process>();

public void parseNodeSend(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("GRM:Scheduler");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

//get schedPolicy, FIFO timeTable priority.

String type = eElement.getAttribute("schedPolicy");

schedPolicy = type;

break;

}

}

System.out.println("SchedPolicy:" + schedPolicy);

}

private void getProcessContent(Document doc, String[] processList) {

pList.clear();

NodeList nList = doc.getElementsByTagName("packagedElement");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String processStr = eElement.getAttribute("name");

for (int i = 0; i < processList.length; i++) {

String name = processList[i] + ":Task";

if (name.compareTo(processStr) == 0) {

Process process = new Process();

process.name = processList[i];

System.out.println("find " + process.name);

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("ownedOperation")) {

Element e = (Element) node;

String ownedOperationName = e.getAttribute("name");

process.addThreads(ownedOperationName);

System.out.println("add thread " + ownedOperationName);

}

}

}

pList.add(process);

break;

}

}

}

}

}

public String getSchedPolicy() {

return schedPolicy;

}

public void setSchedPolicy(String schedPolicy) {

this.schedPolicy = schedPolicy;

}

public ArrayList<Process> getpList() {

return pList;

}

public void setpList(ArrayList<Process> pList) {

this.pList = pList;

}

}

package edu.buaa.sei.run;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.datamodel.Message;

import edu.buaa.sei.utils.StringHandle;

public class Publisher {

private String publisherName;

private String publisherId;

private ArrayList<String> sonId = new ArrayList<String> ();

public String getPublisherName() {

return publisherName;

}

public void setPublisherName(String publisherName) {

this.publisherName = publisherName;

}

public String getPublisherId() {

return publisherId;

}

public void setPublisherId(String publisherId) {

this.publisherId = publisherId;

}

public ArrayList<String> getSonId() {

return sonId;

}

public void setSonId(ArrayList<String> sonId) {

this.sonId = sonId;

}

}

package edu.buaa.sei.run;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import edu.buaa.sei.datamodel.Message;

import edu.buaa.sei.utils.RandomGenerator;

import edu.buaa.sei.utils.StringHandle;

public class Transporter {

// FIFO

// FixedPriority

// TimeTableDrivenff

private String schedPolicy;

private double paskageUnitSize = 1.4;

private double totalsize;// kb

private double minKb;

private double maxKb;

public void getTransporter(String umlPath)

throws ParserConfigurationException, SAXException, IOException {

File fXmlFile = new File(umlPath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("GQAM:GaCommStep");

int size\_kb = -1;

String transportId = null;

double maxThroughPut = 0, minThroughput = 0;

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = (Node) nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

transportId = eElement.getAttribute("base\_NamedElement");

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("msgSize")) {

String sizeStr = node.getFirstChild()

.getNodeValue();

size\_kb = StringHandle.getKbFromStr(sizeStr);

// System.out.println("size:" + size\_kb + "kb");

break;

}

}

}

if (size\_kb != -1)

break;

}

}

// get MaxThroughPut and MinThroughput

NodeList throughPutList = doc

.getElementsByTagName("Profile:SendMessage");

for (int temp = 0; temp < throughPutList.getLength(); temp++) {

Node nNode = (Node) throughPutList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

String id = eElement.getAttribute("base\_Message");

if (id.compareTo(transportId) == 0) {

for (Node node = nNode.getFirstChild(); node != null; node = node

.getNextSibling()) {

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("MaxThroughPut")) {

String maxStr = ((Element) node)

.getAttribute("precision");

// System.out.println("max:" + maxStr);

maxThroughPut = Double.valueOf(maxStr);

}

}

if (node.getNodeType() == Node.ELEMENT\_NODE) {

if (node.getNodeName().equals("MinThroughput")) {

String minStr = ((Element) node)

.getAttribute("precision");

// System.out.println("min:" + minStr);

minThroughput = Double.valueOf(minStr);

}

}

}

}

}

}

totalsize = size\_kb;

minKb = minThroughput;

maxKb = maxThroughPut;

}

public double calculateTransportFIFO() {

double time = 0;

// int packageNum = (int) Math.ceil(totalsize / paskageUnitSize);

double bandWidth = RandomGenerator.getARandomNum(minKb, maxKb);

time = (totalsize / bandWidth) \* 1000;

System.out.println("transport time used " + time + "ms.");

return time;

}

}

package edu.buaa.sei.run;

import java.io.IOException;

import java.util.ArrayList;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import edu.buaa.sei.datamodel.Message;

import edu.buaa.sei.datamodel.Process;

import edu.buaa.sei.datamodel.Receiver;

import edu.buaa.sei.datamodel.Sender;

public class Wrapper {

public static void main(String[] args) throws IOException, ParserConfigurationException, SAXException {

//setp 1: parse uml.

NodeSend nodeSend = new NodeSend();

nodeSend.parseNodeSend("CaseStudy/NodeSend.uml");

String schedPolicy = nodeSend.getSchedPolicy();

Sender sender = new Sender();

sender.getSender("CaseStudy/send.uml");

Receiver receiver = new Receiver();

receiver.getReveiver("CaseStudy/receive.uml");

Transporter transporter = new Transporter();

transporter.getTransporter("CaseStudy/transport.uml");

//step 2:

DDS dds = new DDS();

dds.getAllDependency("CaseStudy/publisher.uml");

dds.calculateDependency(sender);

dds.showResults();

}

}

package edu.buaa.sei.run;

import java.util.ArrayList;

import java.util.List;

public class BaseModelManager {

List<UMLComponent> umlCompList = new ArrayList<UMLComponent>();

List<UMLDependency> umlDepList = new ArrayList<UMLDependency>();

List<UMLMessage> umlMsgList = new ArrayList<UMLMessage>();

String startCompName;

String endCompName;

private static BaseModelManager instance;

public static BaseModelManager INS = instance();

private static BaseModelManager instance() {

if (instance == null) {

instance = new BaseModelManager();

}

return instance;

}

public void addUMLComponent(UMLComponent comp) {

umlCompList.add(comp);

}

public void addUMLDependency(UMLDependency dep) {

umlDepList.add(dep);

}

public void addUMLMessage(UMLMessage msg) {

umlMsgList.add(msg);

}

public UMLComponent findUMLComponent(String compName) {

for (UMLComponent comp : umlCompList) {

if (comp.getCompName().equals(compName))

return comp;

}

return null;

}

public List<UMLDependency> findUMLDependency(String compName, String type) {

List<UMLDependency> resultDepList = new ArrayList<UMLDependency>();

for (UMLDependency dep : umlDepList) {

if ((type.equals("supplier") && dep.getSupplier().getCompName()

.equals(compName))

|| (type.equals("client") && dep.getClient().getCompName()

.equals(compName)))

resultDepList.add(dep);

}

return resultDepList;

}

public List<UMLDependency> findUMLDependency(UMLComponent supplier,

UMLComponent client) {

List<UMLDependency> resultDepList = new ArrayList<UMLDependency>();

for (UMLDependency dep : umlDepList) {

if (dep.getSupplier().equals(supplier)

&& dep.getClient().equals(client))

resultDepList.add(dep);

}

return resultDepList;

}

public List<UMLMessage> findUMLMessage(String compName, String type) {

List<UMLMessage> resultMsgList = new ArrayList<UMLMessage>();

for (UMLMessage msg : umlMsgList) {

if ((type.equals("sender") && msg.getSender().equals(compName))

|| (type.equals("receiver") && msg.getReceiver().equals(

compName)))

resultMsgList.add(msg);

}

return resultMsgList;

}

public List<UMLComponent> getUmlCompList() {

return umlCompList;

}

public List<UMLDependency> getUmlDepList() {

return umlDepList;

}

public List<UMLMessage> getUmlMsgList() {

return umlMsgList;

}

public String getStartCompName() {

return startCompName;

}

public void setStartCompName(String startCompName) {

this.startCompName = startCompName;

}

public String getEndCompName() {

return endCompName;

}

public void setEndCompName(String endCompName) {

this.endCompName = endCompName;

}

}

package edu.buaa.sei.run;

import java.util.ArrayList;

import java.util.List;

public class BaseModelManager\_V2 {

List<UMLClassIns> umlClaInsList = new ArrayList<UMLClassIns>();

List<UMLMessage\_V2> umlMsgList = new ArrayList<UMLMessage\_V2>();

String startInsName;

String endInsName = "finishNode";

private static BaseModelManager\_V2 instance;

public static BaseModelManager\_V2 INS = instance();

private static BaseModelManager\_V2 instance() {

if (instance == null) {

instance = new BaseModelManager\_V2();

}

return instance;

}

public void addUMLMessage(UMLMessage\_V2 msg) {

umlMsgList.add(msg);

}

public List<UMLMessage\_V2> findUMLMessage(String claInsName, String type) {

List<UMLMessage\_V2> resultMsgList = new ArrayList<UMLMessage\_V2>();

for (UMLMessage\_V2 msg : umlMsgList) {

if ((type.equals("sender") && msg.getSender().getInsName()

.equals(claInsName))

|| (type.equals("receiver") && msg.getReceiver()

.getInsName().equals(claInsName)))

resultMsgList.add(msg);

}

return resultMsgList;

}

public List<UMLMessage\_V2> getUmlMsgList() {

return umlMsgList;

}

public void addUMLClaIns(UMLClassIns umlClaIns) {

this.umlClaInsList.add(umlClaIns);

}

public List<UMLClassIns> findInsByClaName(String umlClaName) {

List<UMLClassIns> resultList = new ArrayList<UMLClassIns>();

for (UMLClassIns umlClaIns : this.umlClaInsList) {

if (umlClaIns.getClassName().equals(umlClaName))

resultList.add(umlClaIns);

}

return resultList;

}

public UMLClassIns findInsByInsName(String insName) {

for (UMLClassIns umlClaIns : this.umlClaInsList) {

if (umlClaIns.getInsName().equals(insName))

return umlClaIns;

}

return null;

}

public String getStartInsName() {

return startInsName;

}

public void setStartInsName(String startInsName) {

this.startInsName = startInsName;

}

public String getEndInsName() {

return endInsName;

}

public void setEndInsName(String endInsName) {

this.endInsName = endInsName;

}

public List<UMLClassIns> getUmlClaInsList() {

return umlClaInsList;

}

public void clearAll() {

this.umlMsgList.clear();

this.umlClaInsList.clear();

this.startInsName = "";

}

}

package edu.buaa.sei.run;

import java.util.ArrayList;

import java.util.List;

public class ClaInsFragment {

String claInsName;

List<TacticInfo> tiList;

public ClaInsFragment(String claInsName, List<TacticInfo> tiList) {

super();

this.claInsName = claInsName;

this.tiList = tiList;

}

public String getClaInsName() {

return claInsName;

}

public List<TacticInfo> getTiList() {

return tiList;

}

}

package edu.buaa.sei.run;

public class Constants {

public final static int MCOST\_TYPE\_AC[] = { 1, 10 };

public final static int MCOST\_TYPE\_AP[] = { 1, 10 };

public final static int MCOST\_TYPE\_RP[] = { 1, 10 };

public final static int MCOST\_TYPE\_MA[] = { 1, 10 };

public final static int MCOST\_TYPE\_MR[] = { 1, 10 };

public final static String T\_HEARTBEAT = "Heartbeat";

public final static String T\_PING\_ECHO = "Ping/Echo";

public final static String T\_TIMESTAMP = "TimeStamp";

public final static String T\_SANITY\_CHECKING = "SanityChecking";

public final static String T\_CONDITION\_MONITORING = "ConditionMonitoring";

public final static String T\_EXCEPTION\_DETECTION = "ExceptionDetection";

public final static String T\_SELF\_TEST = "SelfTest";

public final static String T\_ACTIVE\_REDUNDANCY = "ActiveRedundancy";

public final static String T\_PASSIVE\_REDUNDANCY = "PassiveRedundancy";

public final static String T\_VOTING = "Voting";

public final static String T\_STATE\_RESYNCHRONIZATION = "StateResynchronization";

public final static String T\_CHECKPOINT\_ROLLBACK = "Checkpoint/Rollback";

public final static String T\_REMOVAL\_FROM\_SERVICE = "RemovalFromService";

public final static String T\_TRANSACTIONS = "Transactions";

public final static String T\_EXCEPTION\_PREVENTION = "ExceptionPrevention";

public final static String T\_INCREASE\_CONPETENCE\_SET = "IncreaseConpetenceSet";

public final static String T\_EXCEPTION\_HANDLING = "ExceptionHandling";

public final static String T\_SOFTWARE\_UPGRADE = "SoftwareUpgrade";

public final static String T\_RETRY = "Retry";

public final static String T\_IGONRE\_FAULTY\_BEHAVIOR = "IgonreFaultyBehavior";

public final static String T\_DEGRADATION = "Degradation";

public final static String T\_RECONFIGURATION = "Reconfiguration";

public final static int MAX\_NUM\_BACKUP = 3;

public final static String FAULT\_DETECTION\_TACTIC[] = { T\_HEARTBEAT,

T\_PING\_ECHO, T\_TIMESTAMP, T\_SANITY\_CHECKING,

T\_CONDITION\_MONITORING, T\_EXCEPTION\_DETECTION, T\_SELF\_TEST };

public final static String REDUNDANCY\_TACTIC[] = { T\_ACTIVE\_REDUNDANCY,

T\_PASSIVE\_REDUNDANCY, T\_VOTING };

public final static String FAULT\_RECOVERY\_TACTIC[] = {

T\_STATE\_RESYNCHRONIZATION, T\_CHECKPOINT\_ROLLBACK };

public final static String FAULT\_PREVENTION\_TACTIC[] = {

T\_REMOVAL\_FROM\_SERVICE, T\_TRANSACTIONS, T\_EXCEPTION\_PREVENTION,

T\_INCREASE\_CONPETENCE\_SET };

public final static String FAULT\_REPAIR\_TACTIC[] = { T\_EXCEPTION\_HANDLING,

T\_SOFTWARE\_UPGRADE, T\_RETRY, T\_IGONRE\_FAULTY\_BEHAVIOR,

T\_DEGRADATION, T\_RECONFIGURATION };

}

package edu.buaa.sei.run;

public class SearchResult {

double fitness;

double bestSR;

double bestCost;

int[] v;

public SearchResult(double fitness, double bestSR, double bestCost, int[] v) {

super();

this.fitness = fitness;

this.bestSR = bestSR;

this.bestCost = bestCost;

this.v = v;

}

public String getFitness() {

return "" + fitness;

}

public String getBestSRAndCost() {

return bestSR + "," + bestCost;

}

public String getV() {

String temp = "" + this.v[0];

for (int i = 1; i < this.v.length; i++) {

temp = temp + "," + this.v[i];

}

return temp;

}

}

package edu.buaa.sei.run;

public class TacticInfo {

int tacticType;

int minRedNum, maxRedNum;

public TacticInfo(int tacticType, int minRedNum, int maxRedNum) {

super();

this.tacticType = tacticType;

this.minRedNum = minRedNum;

this.maxRedNum = maxRedNum;

}

public int getTacticType() {

return tacticType;

}

public int getMinRedNum() {

return minRedNum;

}

public int getMaxRedNum() {

return maxRedNum;

}

}

package edu.buaa.sei.run;

import java.util.ArrayList;

import java.util.List;

public class UMLClassIns {

String insName;

String className;

double tacticSafety = 1.0;

double reliability;

double failureRate;

double cost;

double meanFR;

double lowFR;

double highFR;

List<UMLClassIns> assInsList = new ArrayList<UMLClassIns>();

public UMLClassIns(String insName, String className) {

super();

this.insName = insName;

this.className = className;

}

public void addassIns(UMLClassIns assIns) {

this.assInsList.add(assIns);

}

public String getInsName() {

return insName;

}

public double getCost() {

return cost;

}

public void setCost(double cost) {

this.cost = cost;

}

public double getMeanFR() {

return meanFR;

}

public void setMeanFR(double meanFR) {

this.meanFR = meanFR;

}

public double getLowFR() {

return lowFR;

}

public void setLowFR(double lowFR) {

this.lowFR = lowFR;

}

public double getHighFR() {

return highFR;

}

public void setHighFR(double highFR) {

this.highFR = highFR;

}

public double getFailureRate() {

return failureRate;

}

public void setFailureRate(double failureRate) {

this.failureRate = failureRate;

}

public String getClassName() {

return className;

}

public List<UMLClassIns> getAssInsList() {

return assInsList;

}

public double getReliability() {

return reliability;

}

public void setReliability(double reliability) {

this.reliability = reliability;

}

public double getTacticSafety() {

return tacticSafety;

}

public void setTacticSafety(double tacticSafety) {

this.tacticSafety = tacticSafety;

}

}

package edu.buaa.sei.run;

public class UMLDependency {

String depName;

UMLComponent supplier;

UMLComponent client;

public UMLDependency(String depName, UMLComponent supplier,

UMLComponent client) {

super();

this.depName = depName;

this.supplier = supplier;

this.client = client;

}

public UMLComponent getSupplier() {

return supplier;

}

public void setSupplier(UMLComponent supplier) {

this.supplier = supplier;

}

public UMLComponent getClient() {

return client;

}

public void setClient(UMLComponent client) {

this.client = client;

}

public String getDepName() {

return depName;

}

public void setDepName(String depName) {

this.depName = depName;

}

}

package edu.buaa.sei.run;

public class UMLMessage {

String msgName;

UMLComponent sender;

UMLComponent receiver;

public UMLMessage(String msgName, UMLComponent sender, UMLComponent receiver) {

super();

this.msgName = msgName;

this.sender = sender;

this.receiver = receiver;

}

public String getMsgName() {

return msgName;

}

public UMLComponent getSender() {

return sender;

}

public UMLComponent getReceiver() {

return receiver;

}

}

package edu.buaa.sei.run;

public class UMLMessage\_V2 {

String msgName;

UMLClassIns sender;

UMLClassIns receiver;

public UMLMessage\_V2(String msgName, UMLClassIns sender, UMLClassIns receiver) {

super();

this.msgName = msgName;

this.sender = sender;

this.receiver = receiver;

}

public String getMsgName() {

return msgName;

}

public UMLClassIns getSender() {

return sender;

}

public UMLClassIns getReceiver() {

return receiver;

}

}

package edu.buaa.sei.run;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Iterator;

import java.util.List;

import java.util.PriorityQueue;

import edu.buaa.sei.clock.SimulationClock;

import edu.buaa.sei.event.IEvent;

import edu.buaa.sei.resource.ResourceContainer;

/\*\*

\* Class controls simulation.

\* @author sei

\*

\*/

public class SimulationController implements ISimulationController {

// containers in this simulation

private List<ResourceContainer> containers;

// event list

private PriorityQueue<IEvent> eventList;

// single instance in this simulation engine.

public static SimulationController instance = new SimulationController();

public SimulationClock clock;

// creation of this class is forbidden.

private SimulationController() {

eventList = new PriorityQueue<IEvent>(1, new Comparator<IEvent>() {

public int compare(IEvent first, IEvent second) {

if (first.scheduledAtTime() - second.scheduledAtTime() < 0) return -1;

if (first.scheduledAtTime() - second.scheduledAtTime() == 0) return 0;

return 1;

}

});

eventList.clear();

clock = new SimulationClock();

containers = new ArrayList<ResourceContainer>();

}

// get current time;

@Override

public int currentTime() {

return clock.getCurrentTime();

}

// add container

@Override

public boolean addContainer(ResourceContainer container) {

if (containers.contains(container)) return false;

containers.add(container);

return true;

}

// add event to event list.

@Override

public void addEvent(IEvent event) {

eventList.add(event);

}

// start simulation

@Override

public void start() {

// start all the container.

Iterator<ResourceContainer> iter = containers.iterator();

while (iter.hasNext()) {

iter.next().start();

}

// Iterate all the event to process.

while(!eventList.isEmpty()) {

IEvent event = eventList.remove();

clock.setCurrentTime(event.scheduledAtTime());

event.eventRoutine();

}

}

// stop simulation

@Override

public void stop() {

// left empty.

}

// pause simulation

@Override

public void pause() {

// left empty.

}

// resume simulation

@Override

public void resume() {

// left empty.

}

}

package edu.buaa.sei.run;

public class SimulationClock {

private int currentTime;

public int getCurrentTime() {

return currentTime;

}

public void setCurrentTime(int currentTime) {

this.currentTime = currentTime;

}

public void updateCurrentTime(int plus) {

this.currentTime += plus;

}

}

package edu.buaa.sei.run;

import java.util.UUID;

/\*\*

\* A simulated event, which can be scheduled to occur at a specified point

\* time in the simulated future. When reaching the specified simulated time instant,

\* the eventRoutine method is being executed. In this way, this class supports the

\* so-called event-scheduling simulation modeling.

\* @author sei

\*

\*/

public interface IEvent {

/\*\*

\* Schedules this event to occur in delay simulated time units.

\* @param process

\* @param delay the period of simulated time to wait before this event is executed.

\*/

public void schedule();

/\*\*

\* cancel this event

\*/

public void cancelEvent();

/\*\*

\* return the time this event is about to occur.

\* @return

\*/

public int scheduledAtTime();

/\*\*

\* Executes the simulation logic associated with this event.

\* Notice, that this method is not intended to be called by clients. Instead,

\* the event scheduler of the respective simulation library invokes this method

\* as soon as the simulation is reached at which the event has bean scheduled.

\*

\* @param process the process associated with this event.

\*/

public void eventRoutine();

/\*\*

\* Unique identifier of this event.

\*/

public UUID getId();

}

package edu.buaa.sei.run;

import java.util.UUID;

import edu.buaa.sei.controller.SimulationController;

import edu.buaa.sei.util.util;

public abstract class AbstractEvent implements IEvent {

// Unique identifier of this event.

private UUID id;

// time this event is about to finished.

private int time;

public AbstractEvent(int time) {

this.id = util.generateId();

this.time = time;

}

// Get identifier of this event.

@Override

public UUID getId() {

return id;

}

/\*\*

\* Test the equal of two events.

\*/

@Override

public boolean equals(Object obj) {

if (obj instanceof IEvent)

return ((IEvent) obj).getId() == getId();

return false;

}

/\*\*

\* Put this event to event handler.

\*/

@Override

public void schedule() {

SimulationController.instance.addEvent(this);

}

/\*\*

\* Time this event is about to vanish.

\*/

@Override

public int scheduledAtTime() {

return time;

}

@Override

public void cancelEvent() {

// left empty.

}

}

package edu.buaa.sei.run;

import edu.buaa.sei.processes.ISchedulableProcess;

import edu.buaa.sei.resource.active.ActiveResourceProcessor;

public class ProcessorFinishedEvent extends AbstractEvent {

// processor produces this event.

private ActiveResourceProcessor processor;

// process execute this event.

private ISchedulableProcess process;

/\*\*

\* Default constructor of this class.

\* @param processor

\* @param process

\*/

public ProcessorFinishedEvent(ActiveResourceProcessor processor,

ISchedulableProcess process, int time) {

super(time);

this.process = process;

this.processor = processor;

}

@Override

public void eventRoutine() {

// When process finished. Proceed with the next instruction.

processor.dispatchProcess(process);

processor.release(process);

}

}

package edu.buaa.sei.run;

import edu.buaa.sei.processes.ISchedulableProcess;

import edu.buaa.sei.resource.active.ActiveResourceStorage;

public class StorageFinishedEvent extends AbstractEvent {

// storage resource associated with this event.

private ActiveResourceStorage resource;

// Process associated with this event. left for extension.

private ISchedulableProcess process;

/\*\*

\* Default constructor for this class.

\* @param resource

\* @param process

\* @param time

\*/

public StorageFinishedEvent(ActiveResourceStorage resource,

ISchedulableProcess process, int time) {

super(time);

this.resource = resource;

this.process = process;

}

/\*\*

\* Forward the execution to next request.

\*/

@Override

public void eventRoutine() {

process.getProcessor().process(process);

resource.release(process);

}

}

package edu.buaa.sei.run;

import java.util.List;

import edu.buaa.sei.processes.ISchedulableProcess;

import edu.buaa.sei.resource.active.ActiveResourceProcessor;

/\*\*

\* Load balancer for new created process only. Process is executed

\* in the same processor until finished. Currently, the load balancer

\* does not consume time.

\* @author sei

\*

\*/

public interface ILoadBalancer {

/\*\*

\* Add a processor to this load balancer. If this processor has been

\* added before, return false. Otherwise return true.

\* @param processor

\* @return

\*/

public boolean addProcessor(ActiveResourceProcessor processor);

/\*\*

\*

\* @param process

\* @return

\*/

public void addProcess(ISchedulableProcess process);

}

package edu.buaa.sei.run;

import java.util.Iterator;

import java.util.List;

import edu.buaa.sei.processes.ISchedulableProcess;

import edu.buaa.sei.resource.active.ActiveResourceProcessor;

/\*\*

\* Load balancer for new created process only. Process is executed

\* in the same processor until finished. In each computation node,

\* there could be many processors but only one storage resource.

\* There may be many communication resource, but it's not allocated

\* by balancer, but allocated by developer.

\* @author sei

\*

\*/

public class LoadBalancer implements ILoadBalancer {

List<ActiveResourceProcessor> processors;

public LoadBalancer(List<ActiveResourceProcessor> processors) {

this.processors = processors;

}

@Override

public boolean addProcessor(ActiveResourceProcessor processor) {

if (processors.contains(processor))

return false;

processors.add(processor);

return true;

}

@Override

public void addProcess(ISchedulableProcess process) {

Iterator<ActiveResourceProcessor> iter = processors.iterator();

int length = Integer.MAX\_VALUE;

ActiveResourceProcessor processor = null, tmp = null;

while (iter.hasNext()) {

tmp = iter.next();

if (length > tmp.getWaitingLength()) {

processor = tmp;

length = tmp.getWaitingLength();

}

}

process.setProcessor(processor);

processor.process(process);

}

}

package edu.buaa.sei.run;

public class AbstractResource {

// resource instances number.

private int capacity;

// name of this resource, many resource may have the same name

// to show the type.

private String name = "";

// id of this resource, unique.

private String id = "";

public AbstractResource(String name, String id, int capacity) {

this.name = name;

this.id = id;

this.capacity = capacity;

}

public int getCapacity() {

return capacity;

}

public void setCapacity(int capacity) {

this.capacity = capacity;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

@Override

public boolean equals(Object obj) {

if (obj != null && obj instanceof AbstractResource)

return id == ((AbstractResource) obj).id;

return false;

}

}