针对第三单元的三次作业和课程内容，撰写技术博客

• (1)调研，然后总结介绍规格化设计的大致发展历史和为什么得到了人们的重视

• (2)按照作业，针对自己所被报告的规格bug以及雷同的规格bug（限于 bug树的限制，对手无法上报），列一个表格分析

• 规格bug类别（采用bug树上的名称）、每个出现所对应方法的代码行数

• (3)分析自己规格bug的产生原因

• (4)分别列举5个前置条件和5个后置条件的不好写法，并给出改进写法

• (5)按照作业分析被报的功能bug与规格bug在方法上的聚集关系

• 即给出表格：方法名、功能bug数、规格bug数

• (6)归纳自己在设计规格和撰写规格的基本思路和体会

1. 调研

软件形式化方法最早可追溯到20世纪50年代后期对于程序设计语言编译技术的研究，即J.Backus提出BNF描述Algol60语言的语法，出现了各种语法分析程序自动生成器以及语法制导的编译方法，使得编译系统的开发从“手工艺制作方式”发展成具有牢固理论基础的系统方法。形式化方法的研究高潮始于20世纪60年代后期，针对当时所谓“软件危机”,人们提出种种解决方法,归纳起来有两类：一是采用工程方法来组织、管理软件的开发过程；二是深入探讨程序和程序开发过程的规律，建立严密的理论，以其用来指导软件开发实践。前者导致“软件工程”的出现和发展，后者则推动了形式化方法的深入研究。经过30多年的研究和应用，如今人们在形式化方法这一领域取得了大量、重要的成果，从早期最简单的形式化方法——一阶谓词演算方法到现在的应用于不同领域、不同阶段的基于逻辑、状态机、网络、进程代数、代数等众多形式化方法。形式化方法的发展趋势逐渐融入软件开发过程的各个阶段,从需求分析、功能描述(规约)、(体系结构/算法)设计、编程、测试直至维护。

形式化方法的研究和应用已经有40多年的历史了，其产生是Dijkstra和Hoare在程序验证方面的工作和Scott，Stratchey以及其他学者在程序语义方面的工作基础上发展起来的，从最简单的形式化方法到80年代较为复杂的Z语言，直至形成具有代表性的形式规范语言——面向实时及分布式的LOTOS语言和面向对象的Z++语言等。形式化方法已经形成了较为完整的体系。

早期的形式化方法主要局限于程序的形式化验证，早期的规范语言中，有两种有代表性的，由工具支持的语言是AFFIRM语言和OBJ语言，主要功能也是进行程序验证。80年代产生了Z语言和VDM语言这两种形式语言，它们都是基于模型的形式规范语言，它们有一个共同的弱点是较难确定该语言描述的软件总体结构和范围。80年代末期，形式化方法发展中的另一个有代表性的方向是对代数方法和软件技术结合的AMAST研究，HartmutEhrig领导研究并创造了ACT方法，开发了相应的使用环境。此外，使用代数方法的规范与语言还有一些较有影响的语言，比如PLUSS和FOOPS等。

1. 规格BUG

|  |  |  |  |
| --- | --- | --- | --- |
| 作业次数 | 规格bug数量 | 名称 | 雷同bug数量 |
| homework\_9 | 1 | Effects内容为实现算法 | 0 |
| homework\_10 | 1 | Requeirs逻辑错误 | 0 |
| homework\_11 | 8 | 见下表 | 0 |

|  |  |  |
| --- | --- | --- |
| 方法名 | 代码行数 | bug类型 |
| Request | 8 | Effects内容为实现算法 |
| FileLoader.parseMap | 32 | Requeirs逻辑错误 |
| Taxi | 17 | Modified不完整 |
| Taxi.move | 24 | Modified逻辑错误 |
| Taxi.printFinishedTask | 11 | Modified不完整 |
| TraceableTaxi.initTaxi | 11 | Modified不完整 |
| FileLoader.Load | 25 | Effects逻辑错误 |
| LightSystem | 14 | Effects逻辑错误 |
| FlowRecord.add | 10 | Effects逻辑错误 |
| FlowRecord.count | 13 | Effects内容为实现算法 |

1. 规格BUG产生原因

除了最后一次碰到一个心狠手辣的，前两次的朋友都没有太纠结JSF，我自己也从来没有扣分在JSF上（仅有的一个也取消了）所以没有在上面太花时间。尤其是在最后一次作业中，关于新增内容出于懒惰，没有更改JSF。但是没想到，还真有人依靠这个拿分，最后还是倒霉被人抓住。虽然不爽，没有怨言。

1. 不好写法及其改进

可能JSF一直也没有认真写，主要精力主要还是在逻辑的正确性上。

前置条件的：

第一个：

/\*\*

\* @REQUIRES: taxiGUI!=null; lightSystem!=null;

\* @MODIFIES: this.location; this.taxiTime; this.tickTock; this.taxiMap;

\* this.taxiGUI ;

\* @EFFECTS: this.taxiTime == TaxiSys.MAINTIME; this.taxiGUI == taxiGUI;

\* this.location == taxiMap.getPoint(random.nextInt(TaxiSys.SIZE),

\* random.nextInt(TaxiSys.SIZE)); this.tickTock == 0;

\* this.taxiMap.addTaxi(location, this);

\*/

public abstract void initTaxi(TaxiGUI taxiGUI, LightSystem lightSystem);

问题：忽略新增的变量lightSystem

改进：

/\*\*

\* @REQUIRES: taxiGUI!=null;

\* @MODIFIES: this.location; this.taxiTime; this.tickTock; this.taxiMap;

\* this.taxiGUI ;

\* @EFFECTS: this.taxiTime == TaxiSys.MAINTIME; this.taxiGUI == taxiGUI;

\* this.location == taxiMap.getPoint(random.nextInt(TaxiSys.SIZE),

\* random.nextInt(TaxiSys.SIZE)); this.tickTock == 0;

\* this.taxiMap.addTaxi(location, this);

\*/

public abstract void initTaxi(TaxiGUI taxiGUI, LightSystem lightSystem);

第二个：

/\*\*

\* @REQUIRES: point!=null;

\* @MODIFIES: this.taxiMap; this.location;

\* @EFFECTS: Move the taxi without a target. Change the taxiList of point on the

\* taxiMap.

\*/

protected void wander() ;

问题：出线冗余的Requires

改进：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this.taxiMap; this.location;

\* @EFFECTS: Move the taxi without a target. Change the taxiList of point on the

\* taxiMap.

\*/

protected void wander() ;

第三个：

/\*\*

\* @MODIFIES: None;

\* @EFFECTS: (\old(stamps.get(n - 1)==long))==>(times.get(n -

\* 1)==\old(times.get(n - 1))+=num); (\old(stamps.get(n -

\* 1)==long))==>(stamps.get(n - 1)==long)&&(times.get(n - 1)==num);

\* @THREAD\_REQUIRES \locked(\this);

\* @THREAD\_EFFECTS: \locked(this);

\*/

public synchronized void add(long time, int num) {

int n = stamps.size();

if (stamps.get(n - 1) < time) {

stamps.add(time);

times.add(num);

} else {

int count = times.get(n - 1);

times.set(n - 1, count + num);

}

}

问题：对于此函数，其实要求time >= stamps.get(stamps.size());没有考虑time < stamps.get(stamps.size())

改进：

/\*\*

\* @MODIFIES: time >= stamps.get(stamps.size());

\* @EFFECTS: (\old(stamps.get(n - 1)==long))==>(times.get(n -

\* 1)==\old(times.get(n - 1))+=num); (\old(stamps.get(n -

\* 1)==long))==>(stamps.get(n - 1)==long)&&(times.get(n - 1)==num);

\* @THREAD\_REQUIRES \locked(\this);

\* @THREAD\_EFFECTS: \locked(this);

\*/

public synchronized void add(long time, int num) {

int n = stamps.size();

if (stamps.get(n - 1) < time) {

stamps.add(time);

times.add(num);

} else {

int count = times.get(n - 1);

times.set(n - 1, count + num);

}

}

第四个：

/\*\*

\* @REQUIRES: string != null;

\* @MODIFIES: this.message;

\* @EFFECTS: this.message.contains(string);

\*/

public void addMessage(String string) {

message.add(string);

}

问题：对于基本类型可以不用写requires

改进：

/\*\*

\* @MODIFIES: this.message;

\* @EFFECTS: this.message.contains(string);

\*/

public void addMessage(String string) {

message.add(string);

}

第五个：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS: (messages.get(index)!=null) ==> \result ==

\* messages.get(index)&&index =

\* \old(index)-1;exception\_behavior(Exception e);

\*/

public RequestMessage previous() throws Exception {

index--;

try {

RequestMessage requestMessage = messages.get(index);

return requestMessage;

} catch (Exception e) {

index++;

throw new Exception("Illegal Request");

}

}

问题：前置对于数字大小有限制

改进：

/\*\*

\* @REQUIRES: index - 1 < 0 || index - 1 >= messages.size();

\* @MODIFIES: None;

\* @EFFECTS: (messages.get(index)!=null) ==> \result ==

\* messages.get(index)&&index =

\* \old(index)-1;exception\_behavior(Exception e);

\*/

public RequestMessage previous() throws Exception {

index--;

try {

RequestMessage requestMessage = messages.get(index);

return requestMessage;

} catch (Exception e) {

index++;

throw new Exception("Illegal Request");

}

}

后置条件的：

第一个：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS: normal\_behavior:(messages.get(\old(index))!=null) ==> \result ==

\* messages.get(index)&&index =

\* \old(index)+1;exception\_behavior(Exception e);

\*/

public RequestMessage next() throws Exception {

try {

RequestMessage requestMessage = messages.get(index);

index++;

return requestMessage;

} catch (Exception e) {

throw new Exception("Illegal Request");

}

}

问题：对于exception\_behavior和normal\_behavior的理解不足。由于后续没有编程作业，没有进行改进，只进行了完善和修改。

改进：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS: (messages.get(\old(index))!=null) ==> \result ==

\* messages.get(index)&&index = \old(index)+1;

\*/

public RequestMessage next() throws Exception {

try {

RequestMessage requestMessage = messages.get(index);

index++;

return requestMessage;

} catch (Exception e) {

throw new Exception("Illegal Request");

}

}

第二个：

/\*\*

\* @REQUIRES: taxiMap!=null;

\* @MODIFIES: this.taxiMap; this.ID; this.credit; this.state; this.location;

\* this.requestList; this.arrivedTarget; this.gotCustomer;

\* this.executingRequest; this.customerPoint; this.targetPoint;

\* this.executeState this.path; this.flow;this.messages;

\* @EFFECTS: this.taxiMap == taxiMap; this.ID == number++; this.credit == 0;

\* this.state == State.WANDERING; this.location == null;

\* this.requestList != null; this.arrivedTarget == false;

\* this.gotCustomer == false; this.executingRequest == null;

\* this.customerPoint == null; this.targetPoint == null;

\* this.executeState == 0; this.path == null; this.flow == null;

\* this.messages != null;

\*/

public Taxi(TaxiMap taxiMap) {

this.taxiMap = taxiMap;

this.ID = number++;

this.credit = 0;

this.state = State.WANDERING;

this.messages = new ArrayList<RequestMessage>();

this.location = null;

this.requestList = new ArrayList<>();

this.arrivedTarget = false;

this.gotCustomer = false;

this.executingRequest = null;

this.customerPoint = null;

this.targetPoint = null;

this.executeState = 0;

this.path = null;

this.flow = null;

}

问题：关于构造器，有一些类的属性比较多，关于这些属性很多需要进行初始赋值，由于我每一个初始赋值都写入了JSF（按照要求，这样似乎也没问题）导致自己构造器的JSF非常冗长，看起来十分累赘。

改进：

/\*\*

\* @REQUIRES: taxiMap!=null;

\* @MODIFIES: this;

\* @EFFECTS: (initial all the field);

\*/

public Taxi(TaxiMap taxiMap) {

this.taxiMap = taxiMap;

this.ID = number++;

this.credit = 0;

this.state = State.WANDERING;

this.messages = new ArrayList<RequestMessage>();

this.location = null;

this.requestList = new ArrayList<>();

this.arrivedTarget = false;

this.gotCustomer = false;

this.executingRequest = null;

this.customerPoint = null;

this.targetPoint = null;

this.executeState = 0;

this.path = null;

this.flow = null;

}

第三个：

/\*\*

\* @REQUIRES:None;

\* @MODIFIES:None;

\* @EFFECTS:(all the conditions satisfied) ==> \result == true;

\*/

public boolean repOK() {

return pointMap != null;

}

问题：使用了不必要的自然语言。

改进：

/\*\*

\* @REQUIRES:None;

\* @MODIFIES:None;

\* @EFFECTS: \result == (pointMap != null)；

\*/

public boolean repOK() {

return pointMap != null;

}

第四个：

/\*\*

\* @REQUIRES: taxiMap!=null;

\* @MODIFIES: this.taxiMap; this.ID; this.credit; this.state; this.location;

\* this.requestList; this.arrivedTarget; this.gotCustomer;

\* this.executingRequest; this.customerPoint; this.targetPoint;

\* this.executeState this.path; this.flow;

\* @EFFECTS: this.taxiMap == taxiMap; this.ID == number++; this.credit == 0;

\* this.state == State.WANDERING; this.location == null;

\* this.requestList == !=null; this.arrivedTarget == false;

\* this.gotCustomer == false; this.executingRequest == null;

\* this.customerPoint == null; this.targetPoint == null;

\* this.executeState == 0; this.path == null; this.flow == null;

\*/

public Taxi(TaxiMap taxiMap) {

this.taxiMap = taxiMap;

this.ID = number++;

this.credit = 0;

this.state = State.WANDERING;

this.messages = new ArrayList<RequestMessage>();

this.location = null;

this.requestList = new ArrayList<>();

this.arrivedTarget = false;

this.gotCustomer = false;

this.executingRequest = null;

this.customerPoint = null;

this.targetPoint = null;

this.executeState = 0;

this.path = null;

this.flow = null;

}

问题：新增的message作为新的属性，没有算入effects和modified

改进：

/\*\*

\* @REQUIRES: taxiMap!=null;

\* @MODIFIES: this.taxiMap; this.ID; this.credit; this.state; this.location;

\* this.requestList; this.arrivedTarget; this.gotCustomer;

\* this.executingRequest; this.customerPoint; this.targetPoint;

\* this.executeState this.path; this.flow;this.messages;

\* @EFFECTS: this.taxiMap == taxiMap; this.ID == number++; this.credit == 0;

\* this.state == State.WANDERING; this.location == null;

\* this.requestList != null; this.arrivedTarget == false;

\* this.gotCustomer == false; this.executingRequest == null;

\* this.customerPoint == null; this.targetPoint == null;

\* this.executeState == 0; this.path == null; this.flow == null;

\* this.messages != null;

\*/

public Taxi(TaxiMap taxiMap) {

this.taxiMap = taxiMap;

this.ID = number++;

this.credit = 0;

this.state = State.WANDERING;

this.messages = new ArrayList<RequestMessage>();

this.location = null;

this.requestList = new ArrayList<>();

this.arrivedTarget = false;

this.gotCustomer = false;

this.executingRequest = null;

this.customerPoint = null;

this.targetPoint = null;

this.executeState = 0;

this.path = null;

this.flow = null;

}

第五个：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this.pointMap; this.taxiHere;

\* @EFFECTS: taxiHere = new ArrayList[TaxiSys.SQUARESIZE]; pointMap = new

\* Point[TaxiSys.SQUARESIZE]; (\all int i; 0<=i<TaxiSys.SQUARESIZE;

\* pointMap[i] != null); (\all int i; 0<=i<TaxiSys.SQUARESIZE;

\* taxiHere[i] != null);

\*/

@SuppressWarnings("unchecked")

public TaxiMap() {

taxiHere = new LinkedBlockingQueue[TaxiSys.SQUARESIZE];

pointMap = new Point[TaxiSys.SQUARESIZE];

matrix = new int[6405][6405];

for (int i = 0; i < TaxiSys.SQUARESIZE; i++) {

pointMap[i] = new Point(i / 80, i % 80);

}

for (int i = 0; i < TaxiSys.SQUARESIZE; i++) {

taxiHere[i] = new LinkedBlockingQueue<Taxi>();

}

}

问题：taxiHere = new ArrayList[TaxiSys.SQUARESIZE];不妥，这容易理解为再新开一个arraylist。

改进：

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this.pointMap; this.taxiHere;

\* @EFFECTS: taxiHere != null; pointMap != null; (\all int i;

\* 0<=i<TaxiSys.SQUARESIZE; pointMap[i] != null); (\all int i;

\* 0<=i<TaxiSys.SQUARESIZE; taxiHere[i] != null);

\*/

@SuppressWarnings("unchecked")

public TaxiMap() {

taxiHere = new LinkedBlockingQueue[TaxiSys.SQUARESIZE];

pointMap = new Point[TaxiSys.SQUARESIZE];

matrix = new int[6405][6405];

for (int i = 0; i < TaxiSys.SQUARESIZE; i++) {

pointMap[i] = new Point(i / 80, i % 80);

}

for (int i = 0; i < TaxiSys.SQUARESIZE; i++) {

taxiHere[i] = new LinkedBlockingQueue<Taxi>();

}

}

1. 聚集关系

实际上没有任何聚集关系，唯一我承认的功能BUG在于我请求的窗口期不是3000ms而是2800ms，这个是我的数学逻辑错误。而此方法由于比较复杂，最终使用的是自然语言未被报JSF。所有被报的规格BUG与功能BUG为任何重叠。我认为这样的情况下，强行为了完成要求而列表格没有实际意义。所以此处选择放弃这个要求。

1. 思路体会

由于习惯问题肯定还是先写了代码才写的规格，所以规格照着代码进行逻辑上的翻译就可以了，虽然可能这样违背了老师的想法，但是我们本来还是认为逻辑重要性大于JSF正确性的。

体会：对于为了拿一个好成绩，逻辑的正确性其实没有JSF写得好那么重要。小小的设计不妥产生的bug通常也就是丢个个位数的分数，JSF写不好，可以让人接近无效。几次作业下来，体验非常差，每天人人自危，生怕自己被狠狠扣分。一个人因为JSF被扣分就会自己也转变为这样的杀手，抓JSF来拿分的人几乎是以指数增长。仿佛让人看到了人性的脆弱和不堪。想起尼采的那句话，当你凝望深渊，深渊也在凝望着你。