

# 软件测试报告

## 目录

<b>1</b>	<b>概述</b>	<b>2</b>
1.1	相关文档 . . . . .	2
<b>2</b>	<b>需求</b>	<b>2</b>
<b>3</b>	<b>测试平台</b>	<b>2</b>
<b>4</b>	<b>测试报告说明</b>	<b>2</b>
<b>5</b>	<b>自动化测试</b>	<b>3</b>
5.1	SESim.Models.Employee . . . . .	3
5.2	SESim.Models.Company . . . . .	3
5.3	SESim.Models.ContractFactory . . . . .	4
5.4	SESim.Controllers.Persistent.SaveController . . . . .	5
5.5	Sesim.Helpers.Ceras.UlidFormatter . . . . .	5
<b>6</b>	<b>手动测试</b>	<b>6</b>
6.1	读取配置 . . . . .	6
6.2	选择主菜单选项 . . . . .	6
6.3	读写设置项 . . . . .	8
6.4	游戏内部运行 . . . . .	8
<b>7</b>	<b>性能测试结果</b>	<b>10</b>
<b>8</b>	<b>其他测试结果</b>	<b>11</b>
<b>9</b>	<b>不符合项列表</b>	<b>11</b>
<b>10</b>	<b>测试结论</b>	<b>11</b>

# 1 概述

项目简介、术语定义、参考资料见相关文档，此处不再赘述。

## 1.1 相关文档

1. 开发需求说明书 (<https://github.com/01010101lzy/software-engineering-simulator/blob/master/docs/homework/spec.md>)
2. 软件开发计划书 ([https://github.com/01010101lzy/software-engineering-simulator/blob/master/docs/homework/dev\\_plan.md](https://github.com/01010101lzy/software-engineering-simulator/blob/master/docs/homework/dev_plan.md))
3. 软件设计说明书 ([https://github.com/01010101lzy/software-engineering-simulator/blob/master/docs/homework/design\\_manual.md](https://github.com/01010101lzy/software-engineering-simulator/blob/master/docs/homework/design_manual.md))

## 2 需求

关于功能、性能的需求，见《软件设计说明书》。

## 3 测试平台

单元测试、集成测试、部分验收测试：Travis CI 持续集成平台，Linux 系统。

手动测试：Surface Pro 4 (Intel i5-6300U, Intel HD Graphics 620)。

## 4 测试报告说明

测试报告中的所有单项报告都按照以下格式编写：

### <测试的功能名称>

<测试说明>

( "前置条件： " <前置条件> )?

( "后置条件： " <后置条件> )? // 在前置、后置条件不存在时不写

"输入与预期输出： "

( <编号> ". " (<该步骤的输入> | " (" <该步骤执行时的状态 " ) " ) "; " <该步骤的预期输出> ". " )+

( <备注> )?

( <测试通过状况> <测试人> <测试时间> )? // 对于自动化测试此项省略

## 5 自动化测试

所有自动化测试均已通过。具体的测试结果可以在 <https://travis-ci.com/01010101lzy/software-engineering-simulator/> 查看。

下面列出了所有的单元测试内容。

### 5.1 SESim.Models.Employee

测试对象代码: Assets/lib/models/Employee.cs。

测试代码: Assets/Tests/Models/EmployeeTest.cs。

#### 5.1.1 EmployeeEfficiencyTest

员工的效率应当符合给出的函数定义。

输入与预期输出:

1. 在其他条件不变时, 基础倍率与效率成正比。
  1. 输入已知倍率的特性“java”计算; 得到正常效率 3。
  2. 输入已知倍率为 0 的特性“lua”计算; 得到效率为 0。
  3. 输入未知倍率的特性“javascript”计算; 得到效率为 0。
2. 在工作时间变化时, 效率先升高再降低。
  1. 输入工作时间为 0 (0 h); 效率为最高效率的一半。
  2. 输入工作时间为 90 (0.3 h); 效率为最高效率。
  3. 输入工作时间为 600 (2 h); 效率为最高效率。
  4. 输入工作时间为 1200 (4 h); 效率为最高效率的一半。
  5. 输入工作时间为 1800 (6 h); 效率为 0。
3. 工作效率随健康降低而降低。
  1. 输入健康为 0.5; 效率降为一半。
  2. 输入健康为 0; 效率为 0。
4. 工作效率随压力先升高再降低。
  1. 输入压力为 0; 效率略低于基准效率。
  2. 输入压力为 0.35; 效率高于基准效率。
  3. 输入压力为 1; 效率低于基准效率。

备注:

- 工作时间、健康和压力值都是程序内部计算生成的数值, 不会出现输入错误数值的情况。

### 5.2 SESim.Models.Company

测试对象代码: Assets/lib/models/Company.cs。

测试代码: Assets/Tests/Models/CompanyTest.cs。

### 5.2.1 CompanyUpdateTest

在工作时间段，公司应当更新其所有接单的进度。

输入与预期输出：

1. 在  $ut = 0$  时刻调用公司更新；公司订单进度无变化。
2. 在  $ut = 2700$  (09:00) 时刻调用公司更新；公司订单进度增加。

### 5.2.2 AddRemoveContractTest

测试公司可以正常地添加和删除订单。

输入和预期输出：

1. 向公司添加一个新的订单；订单数量增加 1，公司订单列表包含新订单。
2. 从公司删除该订单；订单数量减少 1，公司订单列表不包含新订单。

### 5.2.3 AddRemoveEmployeeTest

测试公司可以正常地添加和删除员工。

输入和预期输出：

1. 向公司添加一个新的员工；员工数量增加 1，公司员工列表包含新员工。
2. 从公司删除该员工；员工数量减少 1，公司员工列表不包含新员工。

## 5.3 SESim.Models.ContractFactory

测试对象代码：Assets/lib/models/ContractFactory.cs。

测试代码：Assets/Tests/Models/ContractFactoryTest.cs。

### 5.3.1 ReadFactoryTest

测试订单生成器可以正确地解析一个配置文件，给自己赋值。

输入和预期输出：

1. 输入一份正确的配置文件；生成的订单生成器各属性值如配置所述。
2. 输入一份错误的配置文件；订单生成器报错并取消生成。

### 5.3.2 ContractGenerationTest

测试订单生成器可以正确地按照其配置生成一个订单。

输入和预期输出：

1. 让前文所述的正确配置文件生成一份订单；订单的配置与配置相符。

## 5.4 SESim.Controllers.Persistent.SaveController

测试对象代码: Assets/lib/gameplay/controllers/persistent/SaveController.cs

测试代码: Assets/Tests/Controllers/Persistent/SaveControllerTest.cs

### 5.4.1 SerializationTest

测试存档可以被使用的 Ceras 序列化器成功转换成二进制格式, 并转换回来。

输入和预期输出:

1. 用 Ceras 序列化一个存档。
2. 将被序列化的存档用 Ceras 反序列化回来; 两个存档应该相等。

### 5.4.2 MetadataSerializationTest

测试元数据可以被使用的 Ceras 序列化器成功转换成二进制格式, 并转换回来。

输入和预期输出:

1. 用 Ceras 序列化一个元数据。
2. 将被序列化的元数据用 Ceras 反序列化回来; 两个元数据应该相等。

## 5.5 Sesim.Helpers.Ceras.UlidFormatter

测试对象代码: Assets/lib/helpers/ceras/Deserializers.cs

测试代码: Assets/Tests/Helpers/Ceras/UlidFormatterTest.vs

### 5.5.1 TestSerializer

测试自定义的序列化器可以成功将 Ulid 序列化为指定的形式。

输入与预期输出:

1. 序列化一个随机的 Ulid; 序列化结果应当与该 ID 的二进制表示完全一致。

### 5.5.2 TestDeserializer

测试自定义的序列化器可以成功将二进制形式的数据反序列化为 Ulid。

输入与预期输出:

1. 序列化一个随机的 Ulid, 再反序列化它; 反序列化结果应当与原始 ID 完全一致。

## 6 手动测试

由于自动化测试系统的局限性，一些如界面显示和与人交互的功能不能通过自动测试系统完成。在这些方面，我们进行了手动测试。

此外，所有软件用例均使用手动测试确保可以正常运行。

### 6.1 读取配置

用例：

- 读取配置和设置

#### 6.1.1 读取配置和设置

测试游戏可以正常地读取配置文件。

前置条件：游戏根目录中存在文件夹和文件：GameData/Sesim/androidMarketContract.conf（正常配置）、GameData/Sesim/androidMarketContract\_wrong.conf（错误配置）

输入和预期输出：

1. 双击可执行文件启动游戏；游戏显示加载界面。
2. (游戏加载中)；加载界面显示开始读取两个文件。
3. (游戏加载中)；加载界面显示成功读取正常配置的文件，无法读取错误配置的文件。
4. (游戏加载完成)；游戏进入主菜单。

测试通过 - Rynco, 2019.06.15

### 6.2 选择主菜单选项

用例：

- 开始新游戏
- 加载存档
- 选择设置
- 查看制作人员表
- 退出

#### 6.2.1 开始新游戏

测试新游戏可以被成功创建。

前置条件：加载完成并进入主菜单

输入和预期输出：

1. 玩家点击“新游戏”（Game::new）按钮；新游戏被创建。
2. (新游戏创建后)；进入游戏界面。

测试通过 - Rynco, 2019.06.13

### 6.2.2 加载存档

测试游戏可以正常展示文件夹中的所有存档，并加载指定的存档。

前置条件：

- 加载完成并进入主菜单
- 游戏根目录 Saves 文件夹中有存档

输入和预期输出：

1. 玩家点击“加载存档”（Game::load）按钮；游戏展示存档选择界面。
2. （存档选择界面中）；存档信息与实际信息一致。
3. 玩家选择存档，点击“加载”（Load）按钮；游戏加载存档。
4. （在存档加载完成之后）；进入游戏界面。

备注：在存档文件损坏的时候，由于反序列化不会成功，游戏不会加载存档并进入游戏界面。这个地方因为时间问题没有做相应的提示。

测试通过 - Rynco, 2019.06.13

### 6.2.3 删除存档

测试在游戏内可以删除指定的存档。

前置条件：

- 加载完成并进入主菜单
- 游戏根目录 Saves 文件夹中有存档

输入和预期输出：

1. 玩家点击“加载存档”（Game::load）按钮；游戏展示存档选择界面。
2. 玩家选择存档，点击“删除”（Load）按钮；存档界面刷新，选择的存档消失。

备注：如果选择的存档在展示出来之后被从其他途径删除，这次删除不会成功，但是结果与预期的一致。

测试通过 - Rynco, 2019.06.13

### 6.2.4 查看设置

游戏没有可以设置的内容，略过。

### 6.2.5 查看制作人员表

制作人员表没有实装，略过。

### 6.2.6 退出游戏

测试游戏可以通过玩家点击退出按钮关闭退出。

前置条件：加载完成并进入主菜单

输入和预期输出：

1. 玩家点击“退出”按钮；游戏退出。
2. 玩家点击窗体关闭按钮；游戏退出。
3. 玩家在键盘上输入 Alt+F4；游戏退出。

测试通过 - Rynco, 2019.6.12

## 6.3 读写设置项

游戏没有可设置的内容，略过。

## 6.4 游戏内部运行

用例：

- 每帧固定更新
- 玩家招聘员工
- 玩家解雇员工
- 玩家接单
- 玩家完成任务
- 玩家放弃任务
- 玩家调整时间加速倍率

注：

- 本需求下的用例全部带有前置条件“游戏中”，在以下的叙述中将被省略。

### 6.4.1 每帧固定更新

测试游戏每一帧更新的内容是否与预期一致。

前置条件：进入游戏界面

输入和预期输出：

1. (没有输入，游戏自动运行)；根据当前帧对应的游戏时间、员工基础效率等信息计算员工的当前效率；
2. (没有输入，游戏自动运行)；根据员工效率和项目信息计算员工在一帧内完成的工作量；
3. (没有输入，游戏自动运行)；根据工作量更新项目信息；
4. (没有输入，游戏自动运行)；更新员工的熟练度、基础效率、压力值、健康值等信息；
5. (没有输入，游戏自动运行)；更新公司的声望、资金、员工总状态等信息；
6. (没有输入，游戏自动运行)；更新可招聘人员、可接单列表等信息；
7. (没有输入，游戏自动运行)；更新办公室环境视图中员工的位置和动作；



备注：以上所有内容在一帧内运行完成。

测试通过 - Rynco, 2019.6.10

#### **6.4.2 玩家招聘员工**

测试玩家可否正常进行游戏内招聘员工的操作。

前置条件：可招聘列表里有员工。

输入和预期输出：

1. 玩家在可招聘人员列表中选择员工点击招聘；员工从可招聘列表进入公司员工列表。

#### **6.4.3 玩家解雇员工**

测试玩家可否正常进行游戏内招聘员工的操作。

前置条件：员工列表中有员工。

输入和预期输出：

1. 玩家从员工列表内选择员工进行解雇；员工被从公司员工列表中删除。

#### **6.4.4 玩家接单**

测试玩家可否正常接收订单。

前置条件：

- 玩家打开任务列表
- 任务列表内有可以接的任务，玩家也有相应的人员和资金

输入和预期输出：

1. 玩家在可接任务列表内选择需要接受的任务；游戏将任务加入已接单列表，公司增加相应的声望、资金等。
2. 玩家在弹出的菜单内安排任务信息；游戏内应用安排的信息。

测试通过 - Rynco, 2019.06.15

#### 6.4.5 玩家放弃任务

测试玩家可否正常放弃任务。

前置条件：

- 玩家打开任务列表
- 任务列表内有可以放弃的任务，玩家也有相应的人员和资金

输入和预期输出：

1. 玩家在活动任务列表内选择需要放弃的任务；任务被放弃，公司损失相应的声望、资金等，任务相关资源人员被释放。

还没测试 - Rynco, 2019.06.15

#### 6.4.6 玩家完成任务

测试已经完成的任務可否正常給公司帶來收入和聲望。

前置条件：

- 任务列表内有即将完成的任务

输入和预期输出：

1. (自动执行，在任务完成的那一刻)；公司增加相应的资金和声望，任务相关资源被释放。

测试成功 - Rynco, 2019.06.13

#### 6.4.7 玩家调整时间加速倍率

测试玩家可否正常调整游戏的时间加速倍率。

前置条件：游戏运行中

输入和预期输出：

1. 玩家按键盘上 . 键；时间流速变成原来的两倍。连续按键情况下，最大 128x。
2. 玩家按键盘上 , 键；时间流速变成原来的一半。连续按键情况下，最小 1x。

测试成功 - Rynco, 2019.06.02

### 7 性能测试结果

游戏在测试平台可以跑满 30fps，用户不会感受到卡顿。

## 8 其他测试结果

游戏中的各项界面元素排列正常、内容清晰，符合游戏的需求，便于用户使用。

游戏生成的可执行文件在其支持的系统中均运行良好。

## 9 不符合项列表

无。

部分低优先级的功能截至开发周期结束时尚未实装进游戏，故未能测试。这些功能将在以后的开发中补完。  
包括：- 设置界面（如果需要）- 随机事件的生成 - 公司贷款功能 - 员工各项属性更精细的调整 - 制作人员列表（如果需要）

## 10 测试结论

测试完成于 2019.06.15。

游戏中所有已经存在的功能均符合测试结果预期。

测试组组长签字：

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA256

测试结果符合系统要求和预期。

Rynco Maekawa <lynzrand@outlook.com> @ 2019.06.15

-----BEGIN PGP SIGNATURE-----

iHUEARYIAB0WIIQQsiA9Dum8YGRE7gopqPm20T1rWfQUCXQTYcwAKCRBqPm20T1rW  
fSPWAQCzRj77bvmBwTQbu3239WBgeCKoSx/TM/yF21EAsSmGmwEA4fPLnApWxS6T  
cu+puGVdbaabHKzjceMyw+Y0F/RDcAc=  
=jTqq

-----END PGP SIGNATURE-----