Advances in Computer Science Research, volume 77

8th International Conference on Management and Computer Science (ICMCS 2018)

Research on Test Automated Framework for Reverse Generation of Use Cases

Zhiqi Guo[1, a], Jiacong Zhao[2,b] and Yi Guan[3,c,*]

[1,2,3]Dalian Neusoft Institute of Information, Dalian 116000, Liaoning, China

[a]guozhiqi@neusoft.edu.cn, [b]zhaojiacong@neusoft.edu.cn, [c]guanyi@neusoft.edu.cn

*the Corresponding author

Keywords: Automation framework; Data parameterized management; Scenario design

Abstract. In the beginning of the framework technology, it was used to solve or deal with the complex problems of the application system. The traditional software testing process has always been basically a manual execution of test cases, and the same common test cases are needed in the entire software life cycle. Repeatedly performed many times, especially in the verification testing phase such as regression testing. In the traditional testing process, there is a lack of effective test data generation methods, which results in the generated test cases failing to cover all system requirements. In this situation, the author proposes the test theory of reverse generation of use cases and implements the prototype tool of this automation technology framework. The framework reverses test case generation based on test tool recording scripts; it can also locally update objects; parameterizes test data management and filters data by data model to ensure full path coverage of test data; and objects and recordings. The objects that cannot be obtained in the process are attribute-modified; the graphical "one-click" test scenario design makes complex tests simple and clear.

Introduction

In today's world, information technology is changing rapidly, which strongly promotes the development of social productivity. Various types of application software emerge in an endless stream, demands are updated quickly, data is increasing, project cycles are shortened, and the emergence of technologies such as big data and blockchain impose more stringent requirements on software quality. At the same time, it also proposes traditional software quality control methods. New challenges. The Nineteenth Congress pointed out that promoting the deep integration of the Internet, big data, artificial intelligence, and the real economy is an important part of deepening structural reforms on the supply side. With the deep integration of these emerging technologies and the traditional real economy, quality assurance is essential. One of the important links. In the traditional software testing process, the design, generation, and writing of test cases still require the participation of a large number of testers. With the change of requirements, a lot of changes need to be made to the design of test cases, and there is also a lack of a combination of business relationships. The in-depth exploration of constraints and operation sequences makes it difficult to fully cover the test data combination, resulting in insignificant improvements in test efficiency and test quality [1].

This paper takes the test technology of test case reverse generation as the background, combines the traditional software test technology and method to carry out research and analysis, puts forward the automated test framework technology of reverse generation of test cases, dynamic management of test data, and uses the reserved interface technology to automatically generate the use cases. Unavailable page element object attribute modification [2], according to the actual business needs increase and modify the input, output, checkpoints and other functions, and the scene design using a graphical design, thus improving the software test business coverage ratio, with important Theoretical and practical values [3].

Related Work

Why Automated Testing. It is widely known that the most frequent work in the testing of on-line application systems is regression testing. When new requirements and new features come online, it is necessary to ensure that the existing business functions are not affected, and often the regression testing of existing services consumes time. And manpower is the largest, and most of it is repetitive work. Based on this, automated testing can reduce manpower and improve coverage of regression testing, ensure that existing businesses are correct, and improve test execution efficiency [4-5]. The following are some of the advantages of automated testing summarized:
1. Improve the coverage of regression testing and reduce the risk of defects found after the system goes online [6];
2. Improve the efficiency of test execution, which will facilitate rapid identification and repair of defects;
3. automated test scripts with high reusability;
4. enhance the test accuracy, stability, improve the software trustworthiness;
5. Save labor costs, make full use of resources, and perform distributed testing;
6. shorten the software development, test cycle, the product faster to market.

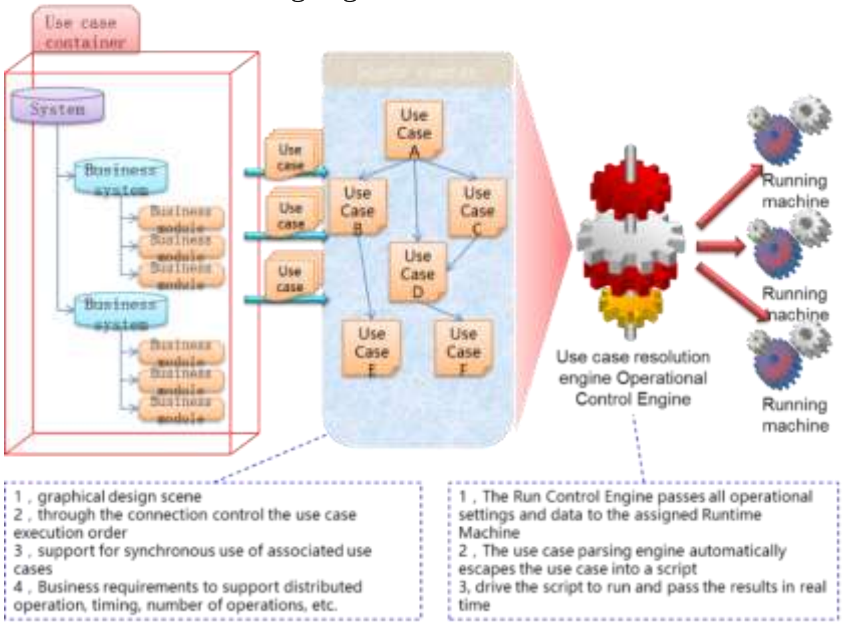
Since the use of automated testing for regression testing has absolute advantages, why is it not widely used? The following will analyze the problems existing in traditional automated test applications and provide solutions to these problems.

Problems and Pain Points in Automated Test Application. Traditional automated testing requires that testers have strong development capabilities to write automated test scripts. This virtually increases the access threshold for automated testing. It also requires more automated testers to complete complex code maintenance. Even if testers have the ability to develop automated scripts, subsequent script maintenance will become more and more complex with the frequent upgrade of the system, because each time the program is updated and updated, test scripts need to be revised or even rewritten. In addition to the project's own development costs, it also requires an additional increase in the workload of developing test scripts, which ultimately leads to uncontrollable costs. As such, how can we successfully implement automated testing? This article proposes the following points that have solved existing problems.

To solve the problem. Judging whether the implementation of automation is successful from the perspective of software engineering, mainly from the perspective of quality, cost, duration, that is, whether to improve the efficiency of the work; whether to improve the quality of software; whether to reasonably control the cost. For these situations, the testing framework mentioned in this paper focuses on the reverse generation of automated test cases, parameterization of test data, graphical design of test scenarios, analysis of control results of automated test operations, etc., and combines visual control of use-case steps and data. The test methods such as the path and the statement used in the traditional test method are automatically generated to complete the test design, test execution, defect analysis, test tracking, and regression test of the software test. The main contributions are as follows:
1. Use case management, including: use case reverse generation and use case steps and data visualization management; test objects, operations, data management; logical relationship management between use cases.
2. Data management, including: automated management of internal relationships of test data; automatic management of relationships between different test fields; automatic generation of test data; automatic generation of test data combinations.
3. Scenario design includes: service flow management and control technology; pre- and post-management control of service flows; load balancing policies for scenario operations; and scheduled scheduling control of scenarios.
4. Analysis of operation control results, including: automatic operation of the job; real-time monitoring of the operation results; automatic error processing; automatic defect delivery and defect analysis.

Use Case Reverse Generated Automated Test Framework

Combining with the existing deficiencies of the existing automated testing process and the need for regression testing, this paper presents an automated test framework for reverse generation of use cases, as shown in the following Fig.1:



1 , graphical design scene
2 , through the connection control the use case execution order
3 , support for synchronous use of associated use cases
4 , Business requirements to support distributed operation, timing, number of operations, etc.

1 , The Run Control Engine passes all operational settings and data to the assigned Runtime Machine
2 , The use case parsing engine automatically escapes the use case into a script
3, drive the script to run and pass the results in real time

Figure 1. System function architecture diagram

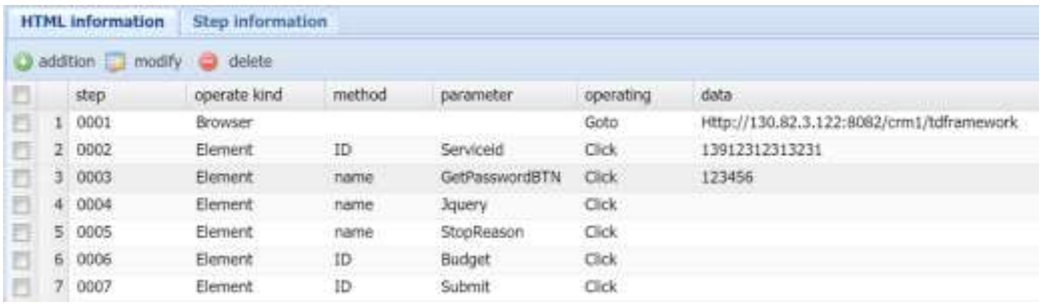
1. There is no need to manually grab objects and write test scripts. The system provides automatic detection of page elements and automatically generates test cases against the set rules.
2. based on the automatic detection of page elements function to achieve visualized test design mode.
3. one-button scene design, through the painting method to complete the automated test design work, after the design is completed, the execution engine can run the entire test process.
4. Support dynamic analysis of real-time test quality for project scale, coverage of use cases, coverage of tested objects, and implementation coverage.
5. can be customized, multi-dimensional report analysis and display functions and distributed scheduling execution.

Technology Implementation

This paper developed and implemented a test automation framework for reverse generation of use cases, including five functional modules, namely use case design management, data management, scenario management, operation monitoring, and result reporting.

Use Case Design Management. It mainly implements functions such as use case generation, use case reverse generation, use case management, use case association settings, and data use between use cases.

Use case generation: Analyze the automated recording script and implement "Automatically escaping test cases", The program is implemented as shown in Fig.2.



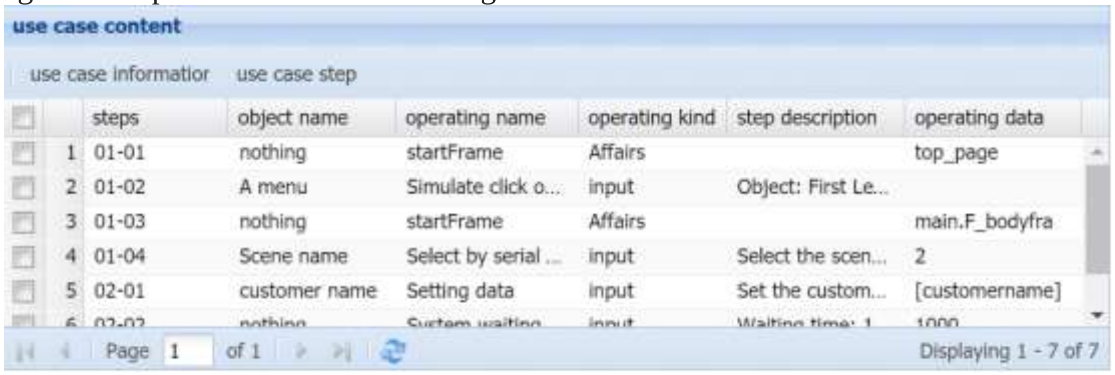
HTML information | Step information

addition | modify | delete

	step	operate kind	method	parameter	operating	data
1	0001	Browser			Goto	Http://130.82.3.122:8082/crm1/tdframework
2	0002	Element	ID	Serviceid	Click	13912312313231
3	0003	Element	name	GetPasswordBTN	Click	123456
4	0004	Element	name	Jquery	Click	
5	0005	Element	name	StopReason	Click	
6	0006	Element	ID	Budget	Click	
7	0007	Element	ID	Submit	Click	

Figure 2. Use case generation

Reverse use case generation: using escaped test cases, secondary editing of page element objects, configuration of operation on page element objects, re-editing of event types, and parameterization of page element object attribute data to achieve reverse generation Test the effect of the use case, The program is implemented as shown in Fig.3.



use case content

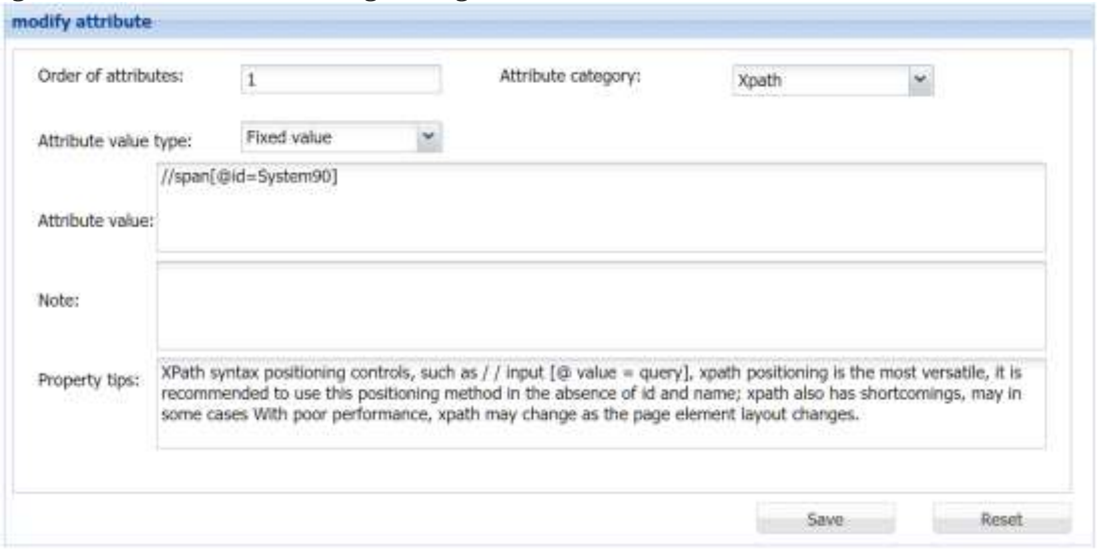
use case informatior | use case step

	steps	object name	operating name	operating kind	step description	operating data
1	01-01	nothing	startFrame	Affairs		top_page
2	01-02	A menu	Simulate click o...	input	Object: First Le...	
3	01-03	nothing	startFrame	Affairs		main.F_bodyfra
4	01-04	Scene name	Select by serial ...	input	Select the scen...	2
5	02-01	customer name	Setting data	input	Set the custom...	[customername]
6	02-02	nothing	System waiting	input	Waiting time: 1	1000

Page 1 of 1 Displaying 1 - 7 of 7

Figure 3. Use case content

Use case management: When an application program modifies a page element and re-runs a regression test, it only needs to modify the corresponding object attribute information. There is no need to re-prepare the use case. Supports custom functions to meet the individual requirements of operating functions, As shown in Fig. 4 ,Fig.5below.



modify attribute

Order of attributes: 1

Attribute category: Xpath

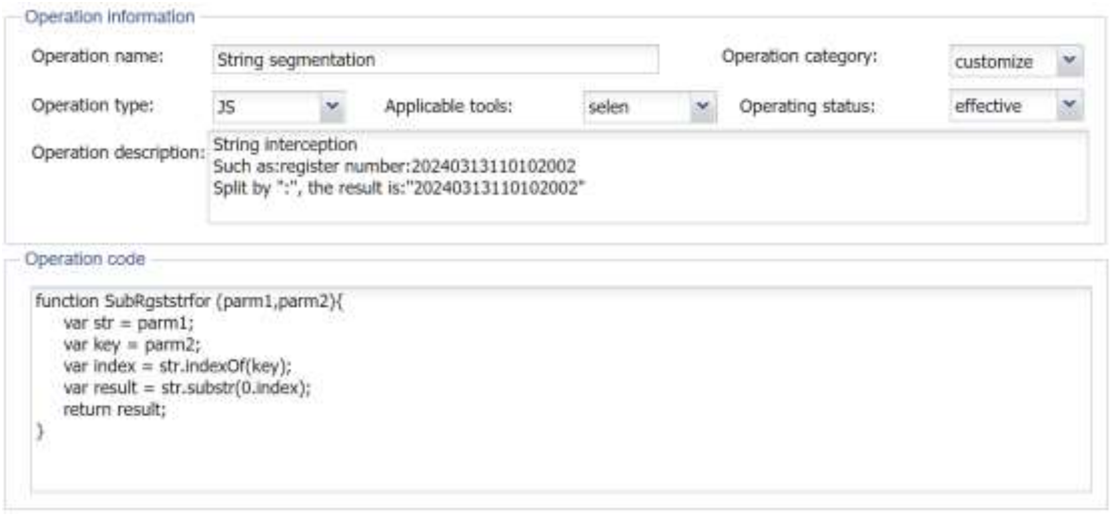
Attribute value type: Fixed value

Attribute value: //span[@id=System90]

Note:

Property tips: XPath syntax positioning controls, such as / / input [@ value = query], xpath positioning is the most versatile, it is recommended to use this positioning method in the absence of id and name; xpath also has shortcomings, may in some cases With poor performance, xpath may change as the page element layout changes.

Figure 4. Modify use case attribute information



Operation information

Operation name: String segmentation

Operation category: customize

Operation type: JS

Applicable tools: selen

Operating status: effective

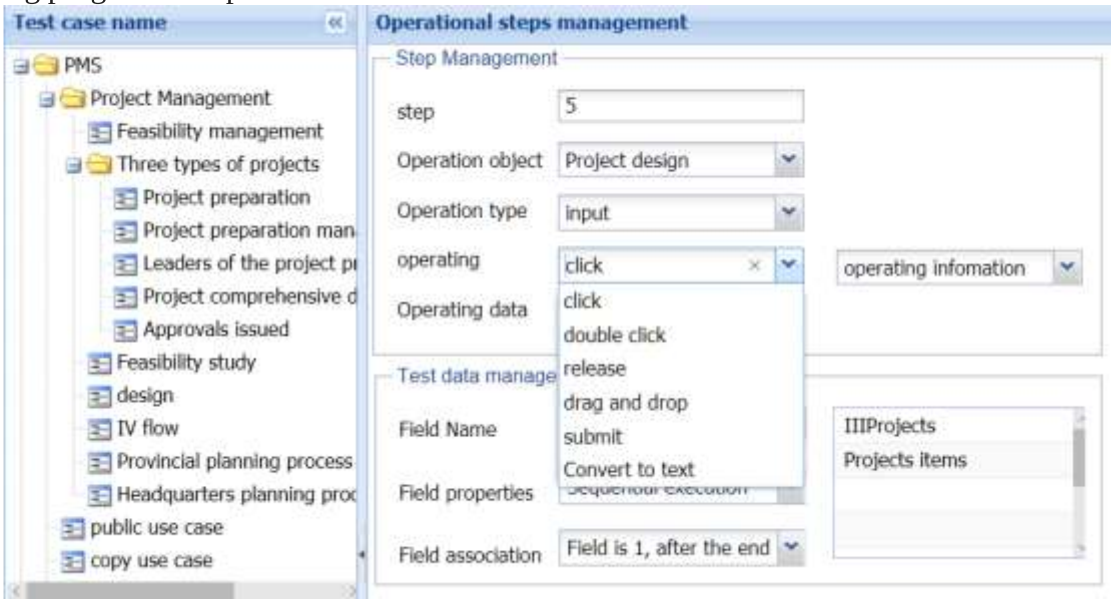
Operation description: String interception
Such as:register number:20240313110102002
Split by ":", the result is:"20240313110102002"

Operation code

```
function SubRgststrfor (parm1,parm2){
    var str = parm1;
    var key = parm2;
    var index = str.indexOf(key);
    var result = str.substr(0.index);
    return result;
}
```

Figure 5. Modify use case operation information

Use Case Steps Management: You can manually modify the operation steps of the object, The following program is implemented.



Test case name

- PMS
  - Project Management
    - Feasibility management
    - Three types of projects
      - Project preparation
      - Project preparation man
      - Leaders of the project pı
      - Project comprehensive d
      - Approvals issued
    - Feasibility study
    - design
    - IV flow
    - Provincial planning process
    - Headquarters planning proc
  - public use case
  - copy use case

Operational steps management

Step Management

step: 5

Operation object: Project design

Operation type: input

operating: click

operating infomation

Operating data

click
double click
release
drag and drop
submit
Convert to text

Test data manage

Field Name

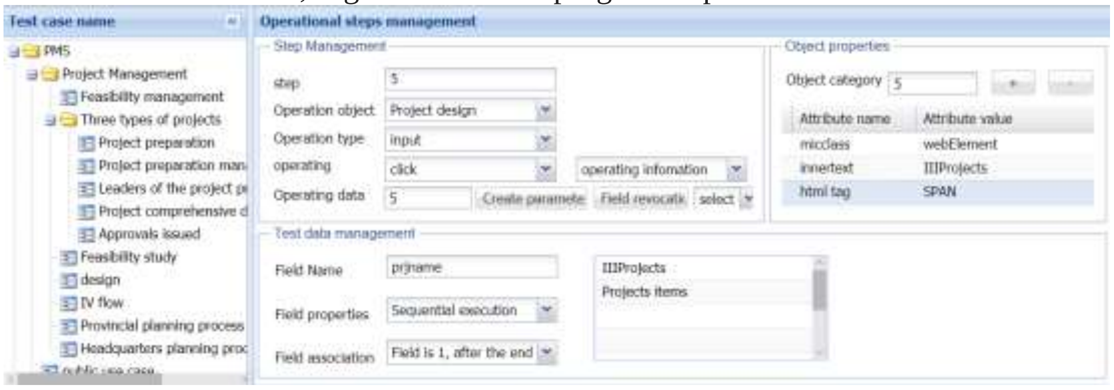
Field properties

Field association: Field is 1, after the end

IIIProjects
Projects items

Figure 6. Operational steps management

Use Case Steps Parametric Management: Performs parameterized management of use case test data, implements use case execution in terms of logical complexity, improves path coverage, and reduces missed measurements, Fig.7 below is the program implementation content.



Test case name

- PMS
  - Project Management
    - Feasibility management
    - Three types of projects
      - Project preparation
      - Project preparation man
      - Leaders of the project pı
      - Project comprehensive d
      - Approvals issued
    - Feasibility study
    - design
    - IV flow
    - Provincial planning process
    - Headquarters planning proc
  - public use case

Operational steps management

Step Management

step: 5

Operation object: Project design

Operation type: input

operating: click

operating infomation

Operating data: 5

Create paramete | Field revocatic | select

Object properties

Object category: 5

Attribute name	Attribute value
micclass	webElement
innertext	IIIProjects
html tag	SPAN

Test data management

Field Name: prjname

Field properties: Sequential execution

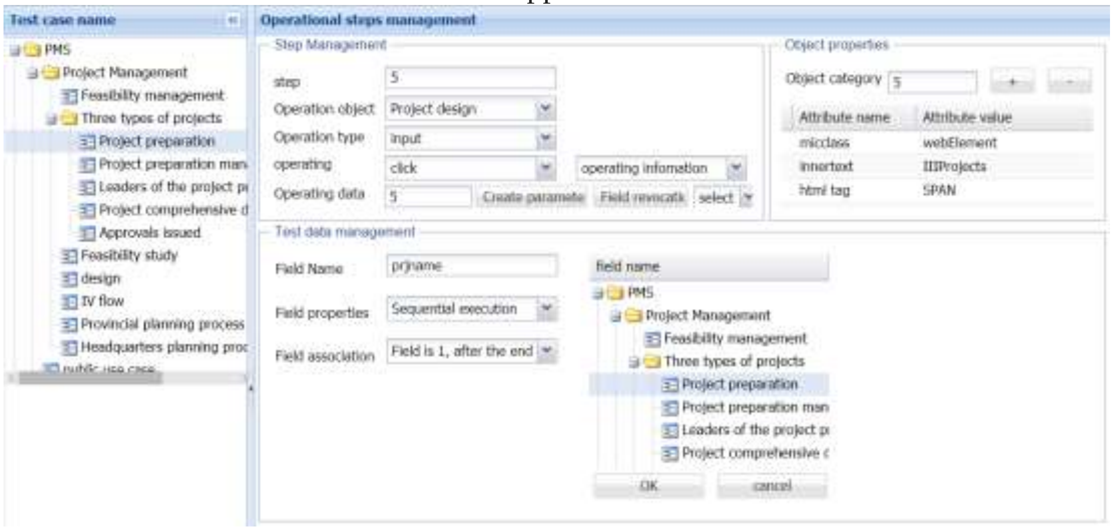
Field association: Field is 1, after the end

IIIProjects
Projects items

Figure 7. Use case steps parametric management

Use case association settings: establish associations between use cases, complete use case level data sharing and scene "one-click" design operations.

Use-case data calls: Use case-level data for application and isolation.



PMS
- Project Management
  - Feasibility management
  - Three types of projects
    - Project preparation
    - Project preparation man
    - Leaders of the project pı
    - Project comprehensive d
    - Approvals issued
  - Feasibility study
  - design
  - IV flow
  - Provincial planning process
  - Headquarters planning proc
- public use case

step	5
Operation object	Project design
Operation type	input
operating	click
	operating infomation
Operating data	5

Object category 5

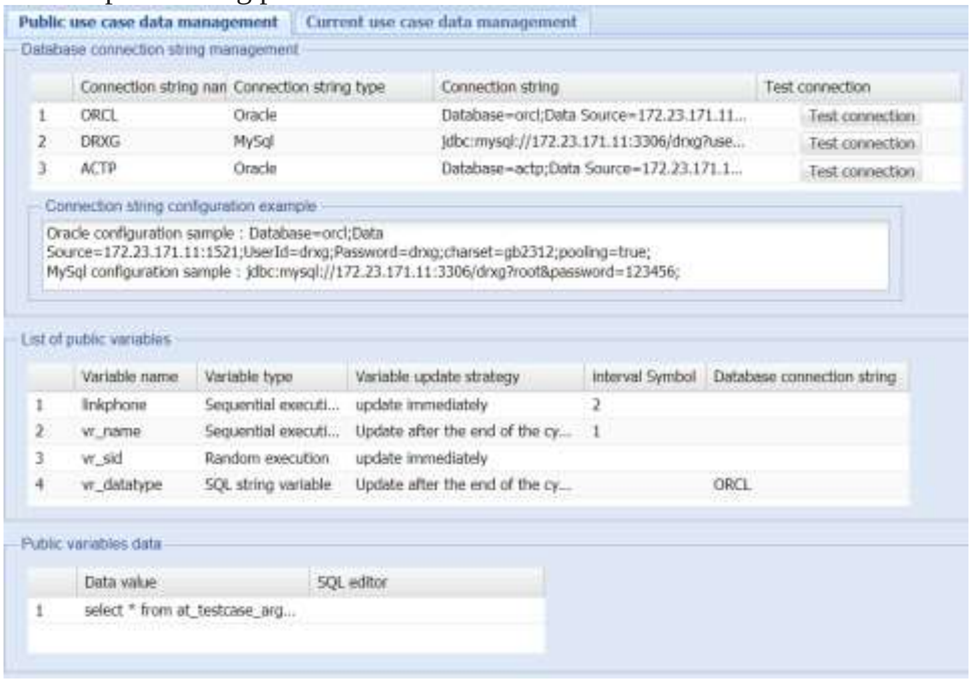
Attribute name	Attribute value
micclass	webElement
innertext	IIIProjects
html tag	SPAN

Field Name	prjname
Field properties	Sequential execution
Field association	Field is 1, after the end

field name
PMS
- Project Management
  - Feasibility management
  - Three types of projects
    - Project preparation
    - Project preparation man
    - Leaders of the project pı
    - Project comprehensive c

Figure 8. Use case association settings

Data Management. Realize the maintenance of variable rules for the data used in the use case; provide a variety of data import forms; generate test data according to the established rules; generate use case data through SQL statements and verify that the SQL syntax conforms to the rules. Only need to configure the use case test data program can be automatically identified, reducing the manual input data repeat testing process.


Public use case data management
Current use case data management
Database connection string management
Connection string configuration example
List of public variables
Public variables data

	Connection string nan	Connection string type	Connection string	Test connection
1	ORCL	Oracle	Database=orcl;Data Source=172.23.171.11...	Test connection
2	DRXG	MySql	jdbc:mysql://172.23.171.11:3306/drxg?use...	Test connection
3	ACTP	Oracle	Database=actp;Data Source=172.23.171.1...	Test connection

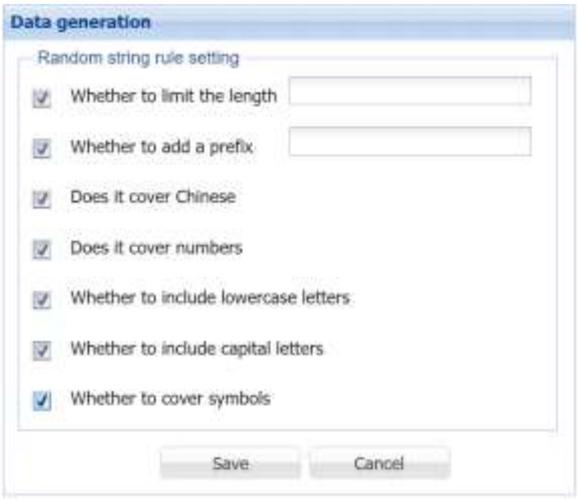
```
Oracle configuration sample : Database=orcl;Data
Source=172.23.171.11:1521;UserId=drxg;Password=drxg;charset=gb2312;pooling=true;
MySql configuration sample : jdbc:mysql://172.23.171.11:3306/drxg?root&password=123456;
```

	Variable name	Variable type	Variable update strategy	interval Symbol	Database connection string
1	linkphone	Sequential executi...	update immediately	2	
2	vr_name	Sequential executi...	Update after the end of the cy...	1	
3	vr_sid	Random execution	update immediately		
4	vr_datatype	SQL string variable	Update after the end of the cy...		ORCL

	Data value	SQL editor
1	select * from at_testcase_arg...	

Figure 9. Use case data management

Fig.10 below is Data Generation Rule Configuration.

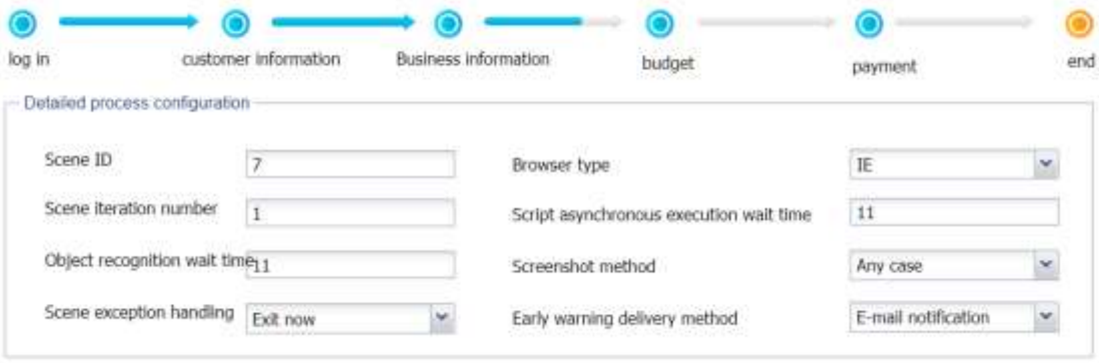

Data generation

Random string rule setting

- ✓ Whether to limit the length
- ✓ Whether to add a prefix
- ✓ Does it cover Chinese
- ✓ Does it cover numbers
- ✓ Whether to include lowercase letters
- ✓ Whether to include capital letters
- ✓ Whether to cover symbols

Figure 10. Data Generation Rule Configuration

Scene Management. According to the business requirements, the drag and drop combination of any node can be used to complete the scenario configuration. The scenario can be configured in detail according to the execution time and other early warning requirements, As shown below.

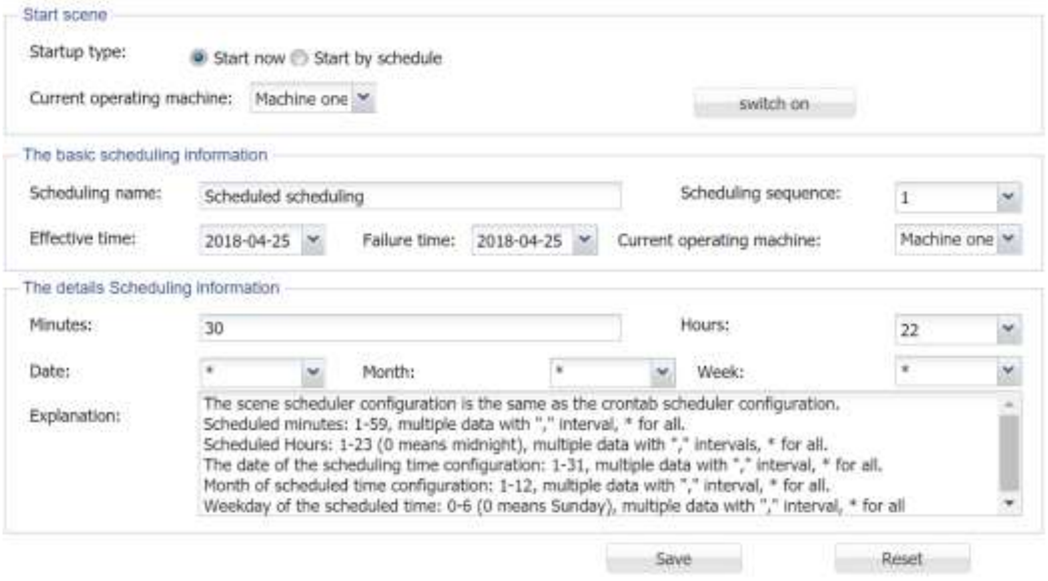


Detailed process configuration

Scene ID	7	Browser type	IE
Scene iteration number	1	Script asynchronous execution wait time	11
Object recognition wait time	11	Screenshot method	Any case
Scene exception handling	Exit now	Early warning delivery method	E-mail notification

Figure 11. Scene management

Manually start and stop scenarios for abnormal situations or debugging, and support configuration of job scheduling information based on actual business requirements.


Start scene

Startup type: [✓ Start now] Start by schedule

Current operating machine: Machine one

The basic scheduling information

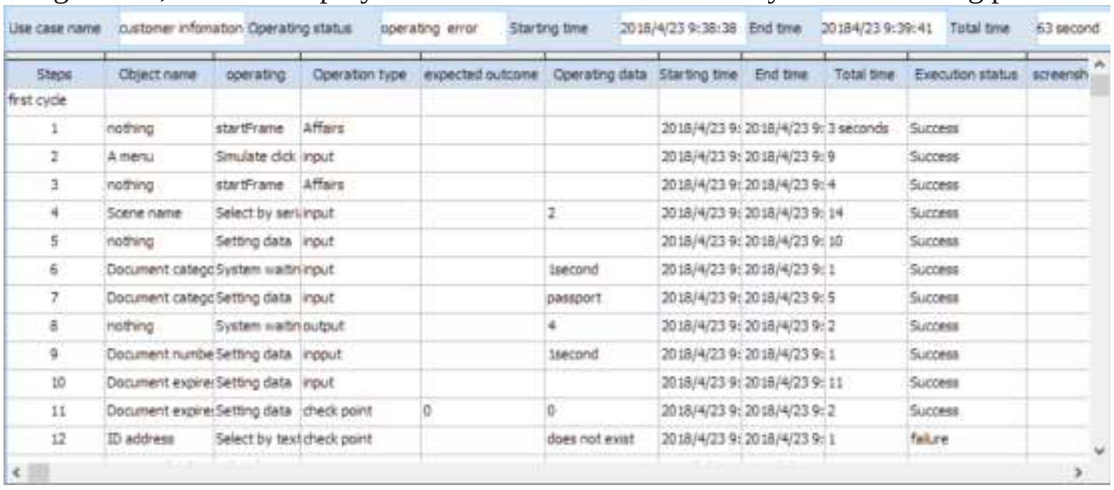
Scheduling name: Scheduled scheduling
Scheduling sequence: 1
Effective time: 2018-04-25
Failure time: 2018-04-25
Current operating machine: Machine one

The details Scheduling information

Minutes: 30
Hours: 22
Date: *
Month: *
Week: *
Explanation: The scene scheduler configuration is the same as the crontab scheduler configuration.
Scheduled minutes: 1-59, multiple data with "," interval, * for all.
Scheduled Hours: 1-23 (0 means midnight), multiple data with "," intervals, * for all.
The date of the scheduling time configuration: 1-31, multiple data with "," interval, * for all.
Month of scheduled time configuration: 1-12, multiple data with "," interval, * for all.
Weekday of the scheduled time: 0-6 (0 means Sunday), multiple data with "," interval, * for all

Figure 12. Scene manual operation

Operation Monitoring. Realize detailed indicator monitoring according to business requirements, including total duration of scene execution monitoring, single-step duration monitoring, time-consuming monitoring of specific transactions, and each step records operation information and running results, and can display screenshots of the results. Easy to find running problems.

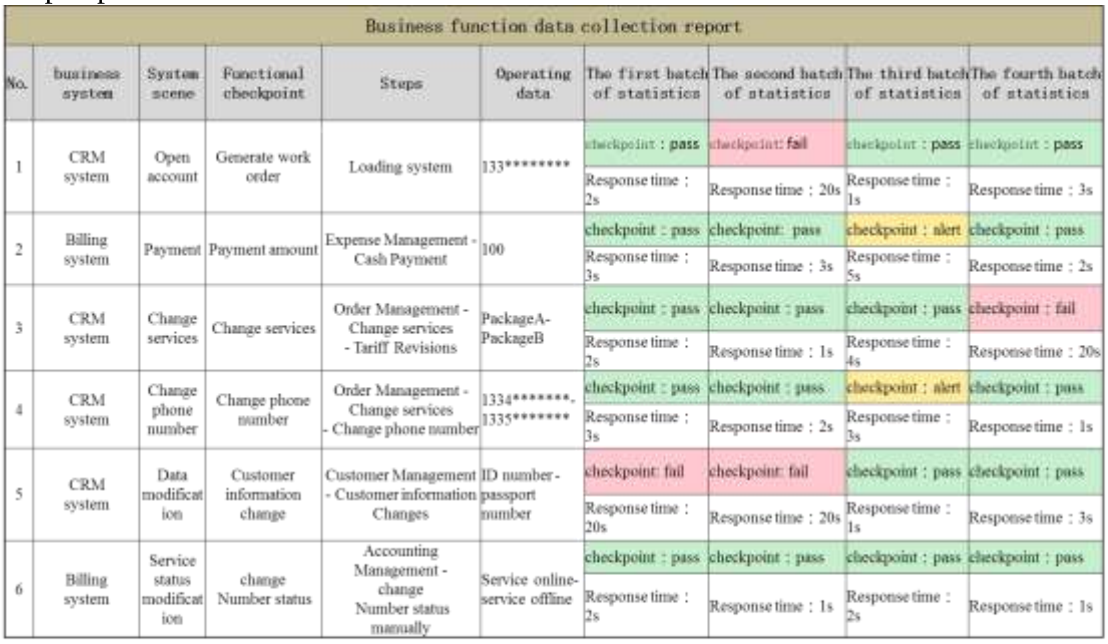


Use case name	customer infomation	Operating status	operating error	Starting time	2018/4/23 9:38:38	End time	20184/23 9:39:41	Total time	63 second

Steps	Object name	operating	Operation type	expected outcome	Operating data	Starting time	End time	Total time	Execution status	screensh
first cycle										
1	nothing	startFrame	Affairs			2018/4/23 9:	2018/4/23 9:	3 seconds	Success	
2	A menu	Simulate click	input			2018/4/23 9:	2018/4/23 9:	9	Success	
3	nothing	startFrame	Affairs			2018/4/23 9:	2018/4/23 9:	4	Success	
4	Scene name	Select by seri	input		2	2018/4/23 9:	2018/4/23 9:	14	Success	
5	nothing	Setting data	input			2018/4/23 9:	2018/4/23 9:	10	Success	
6	Document catego	System waitin	input		1second	2018/4/23 9:	2018/4/23 9:	1	Success	
7	Document catego	Setting data	input		passport	2018/4/23 9:	2018/4/23 9:	5	Success	
8	nothing	System waitin	output		4	2018/4/23 9:	2018/4/23 9:	2	Success	
9	Document numbe	Setting data	inpput		1second	2018/4/23 9:	2018/4/23 9:	1	Success	
10	Document expire	Setting data	input			2018/4/23 9:	2018/4/23 9:	11	Success	
11	Document expire	Setting data	check point	0	0	2018/4/23 9:	2018/4/23 9:	2	Success	
12	ID address	Select by text	check point		does not exist	2018/4/23 9:	2018/4/23 9:	1	failure	

Figure 13. Real-time monitoring of running results

Results Report. Achieve multi-dimensional (functional indicators, performance indicators), a variety of style charts, easy to make judgments on the test results, software quality indicators from a variety of perspectives.



Business function data collection report

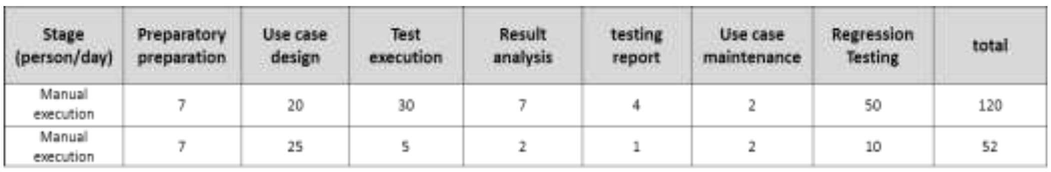
No.	business system	System scene	Functional checkpoint	Steps	Operating data	The first batch of statistics	The second batch of statistics	The third batch of statistics	The fourth batch of statistics
1	CRM system	Open account	Generate work order	Loading system	133********	checkpoint：pass	checkpoint: fail	checkpoint：pass	checkpoint：pass
						Response time：2s	Response time：20s	Response time：1s	Response time：3s
2	Billing system	Payment	Payment amount	Expense Management - Cash Payment	100	checkpoint：pass	checkpoint: pass	checkpoint：alert	checkpoint：pass
						Response time：3s	Response time：3s	Response time：5s	Response time：2s
3	CRM system	Change services	Change services	Order Management - Change services - Tariff Revisions	PackageA-PackageB	checkpoint：pass	checkpoint：pass	checkpoint：pass	checkpoint：fail
						Response time：2s	Response time：1s	Response time：4s	Response time：20s
4	CRM system	Change phone number	Change phone number	Order Management - Change services - Change phone number	1334*******, 1335*******	checkpoint：pass	checkpoint：pass	checkpoint：alert	checkpoint：pass
						Response time：3s	Response time：2s	Response time：3s	Response time：1s
5	CRM system	Data modification	Customer information change	Customer Management - Customer information Changes	ID number - passport number	checkpoint: fail	checkpoint: fail	checkpoint：pass	checkpoint：pass
						Response time：20s	Response time：20s	Response time：1s	Response time：3s
6	Billing system	Service status modification	change Number status	Accounting Management - change Number status manually	Service online-service offline	checkpoint：pass	checkpoint：pass	checkpoint：pass	checkpoint：pass
						Response time：2s	Response time：1s	Response time：2s	Response time：1s

Figure 14. Results report

Conclusion

By comparing the time spent in manual testing and automated testing of various projects in line projects, the following conclusions are drawn: The time-saving ratio of automated testing compared to manual testing is approximately 56%.

Table 1 Time-consuming comparison of two different modes



Stage (person/day)	Preparatory preparation	Use case design	Test execution	Result analysis	testing report	Use case maintenance	Regression Testing	total
Manual execution	7	20	30	7	4	2	50	120
Manual execution	7	25	5	2	1	2	10	52

The following are the conclusions obtained from other metrics:

Functional coverage has improved greatly, with automated test coverage exceeding 70%.

Process branch coverage is wide, and special node process testing is more comprehensive.

Significant improvement in test efficiency, making it easier for regression testing to save time and manpower.

Test preparation data is greatly shortened.

Verification of the suitability of old system data for new features.

The reuse of resources is significantly improved, and only a few modifications are required between different versions to complete the test execution.

Test quality has been greatly improved to ensure the stability of other related functions.

Avoidance of human test negligence and errors, improve software trust.

Test cycle, 50% to 70% less than manual test.

Test environment configuration is reduced by more than 30% over pure automation tool configuration.

Reduced automated test maintenance costs by 60%.

Facts have proven that automated testing can improve the testing efficiency and shorten the project cycle by adopting automated means in use case design, test execution, and test tracking.

References

[1] Li Yu, Fang Jianyong, Jiang Meng. Web Services Automation Testing Framework for Demand Coverage[J]. Computer Science and Exploration, 2017, 11(11): 1747-1763.

[2] Wu Lijin, Han Xinyu, Zhang Kai, Tang Longli. A Non-intrusive GUI Automation Test System Design[J]. Computer Measurement & Control, 2017, 25(12):49-53.

[3] Qiao Yanru, Chen Xiaoxuan, Ouyang Min. Design and Implementation of an Automated Test Platform Based on Robot Framework[J]. Railway Computer Application, 2017, 26(10):8-11.

[4] Li Bohui, Xu Haihong, Ruan Chengming, Cao Jian\'an, Wu Hao. Design and Implementation of an Automated Test System Based on JSON[J]. Measure and Control Technology, 2017, 36(04): 120-123+129.

[5] Tian Hongyun, Liu Qingkai, Cheng Jie, Yang Zhang, Shan Yahui. An Automated Test Platform for High Performance Numerical Simulation Software[J]. Computer Engineering and Science, 2017, 39(11): 1980-1985.

[6] Li Hongfei, Zhang Fusheng. Design and application of automated test platform for manned spacecraft[J]. Spacecraft Engineering, 2017, 26(04): 116-121.