单元测试计划
（Unit Test Plan）

1 引言

1.1 目的

本文档为 Google 开源项目 Guava 以下模块的单元测试活动提供范围、方法、资源和进度方面的指导。

本文档的读者主要是开发经理和开发人员。

1.2 测试策略

以类为单元，采用独立的单元测试策略，通过设计相应的驱动和桩的方法来测试类中的方法。在选择类中被测方法时，根据方法的规模和复杂度进行判定。非空非注释代码行数 LOC>20，或者复杂度 VG>3 的方法进行单元测试，其他方法不进行单元测试。

对于子类的测试采用分层增量测试（Hierarchical Incremental Testing）策略，对子类的变化部分设计新的测试用例，与父类相同的部分则重用父类的测试用例。

执行单元测试的次序是根据《软件设计说明》中的用例实现交互图，从图中最小依赖关系的类开始测试，再逐步扩大到依赖关系较强的类，直至所有类测试完毕。

1.3 范围

单元测试包含了计划阶段、设计阶段、实现阶段和执行阶段四个阶段。本单元测试计划是整个软件开发项目中的一部分，起始于详细设计阶段，直到单元测试阶段结束后终止。该计划主要处理与 MiniLibrary 系统单元测试有关的任务安排、资源需求、人力需求、风险管理、进度安排等内容。

1.4 参考文献

《软件需求规格说明（Software Requirement Specification）》

《软件设计说明（Software Design Descriptions）》

《用户界面规格说明（User Interface Specification）》

1.5 术语

无。

2 测试项目

根据《软件设计说明》中的详细设计内容，单元测试的测试项目如 2.1-2.8 小节所示。

2.1 Net 模块

1 设计类标识:Net 设计类

方法标识符	方法名	代码行（LOC）	复杂度（VG）
NET-01	HostAndPort	5	1
NET-02	fromString()	21	*8*
NET-03	getHost	1	1
NET-04	getHostText	1	1
NET-05	getHostAndPortFromBracketed Host()	30	*2*
NET-06	fromParts	5	1
NET-07	toString()	20	4
NET-08	withDefaultPort	7	2
NET-09	equals()	20	4
NET-10	requireBracketsForIPv6	4	1
NET-11	fromValid()	22	4
NET-12	requireBracketsForIPv6	1	1
NET-13	hashCode	1	1

NET-14	ipStringToBytes()	25	16
NET-15	InetAddress	7	2
NET-16	textToNumericFormatV6()	50	14
NET-17	textToNumericFormatV4	12	2
NET-18	compressLongestRunOfZeroes()	21	8
NET-19	forUriStringNoThrow()	20	4
NET-20	convertDottedQuadToHex()	11	2
NET-21	getCoercedIPv4Address()	40	6
NET-22	parseOctet()	7	1
NET-23	hextetsToIPv6String()	20	8
NET-24	parseHextet()	7	2
NET-25	findPublicSuffix()	23	6
NET-26	bytesToInetAddress()	7	2
NET-27	withParameter()	20	4
NET-28	toAddrString()	16	2
NET-29	parse()	35	8
NET-30	escape()	72	7

2.2 IO模块

1 设计类标识:IO设计类

方法标识符	方法名	代码行（LOC）	复杂度（VG）
IO-01	encodingStream()	43	7
IO-02	decodingStream()	50	9
IO-03	size()	25	5
IO-04	contentEquals()	24	4
IO-05	toByteArray()	30	4
IO-06	copy()	27	6
IO-07	close()	22	5
IO-08	add()	26	15
IO-09	deleteRecursively()	33	9
IO-10	deleteDirectoryContents()	22	6
IO-11	isSubtypeOf()	25	8
IO-12	method()	28	1
IO-13	constructor()	28	1
IO-14	toGenericType()	25	3
IO-15	visit()	28	14
IO-16	reset()	18	7
IO-17	update()	18	3
IO-18	createTempDir()	19	3
IO-19	createParentDirs()	11	3
IO-20	move()	14	4
IO-21	readLines()	17	1
IO-22	flush()	10	3
IO-23	finishLine()	8	1

IO-24	equal()	15	3
IO-25	getFileExtension()	9	2
IO-26	read()	10	3
IO-27	skip()	14	1
IO-28	initialize()	8	3
IO-29	resolveParameterizedType()	10	1
IO-30	resolveType()	14	5

2.3 Collect 模块

1 设计类标识:Collect 设计类

方法标识符	方法名	代码行（LOC）	复杂度（VG）
COLLECT-01	concat()	22	2
COLLECT-02	zip()	28	2
COLLECT-03	filterKeys()	17	5
COLLECT-04	removeOccurrences()	20	4
COLLECT-05	symmetricDifference()	46	4
COLLECT-06	getOnlyElement()	17	4
COLLECT-07	subMap()	24	8
COLLECT-08	subset()	23	8
COLLECT-09	toTable()	27	2
COLLECT-10	containsAll()	21	5
COLLECT-11	contains()	16	6
COLLECT-12	drain()	30	4
COLLECT-13	drainUninterruptibly()	39	11
COLLECT-14	retainOccurrencesImpl()	20	4
COLLECT-15	contains()	16	6
COLLECT-16	Filter()	10	2
COLLECT-17	setCountImpl()	11	3
COLLECT-18	newPriorityQueue()	9	2
COLLECT-19	newLinkedBlockingDeque()	8	2
COLLECT-20	canonical()	6	1
COLLECT-21	readResolve()	7	2
COLLECT-22	newHashSet()	5	1
COLLECT-23	makeComplementByHand()	6	1
COLLECT-24	reverse()	11	4
COLLECT-25	addAllImp()	9	2
COLLECT-26	unmodifiableNavigableMap()	11	2
COLLECT-27	containsEntryImpl()	6	2
COLLECT-28	pollNext()	9	2
COLLECT-29	checkNonnegative()	5	2
COLLECT-30	advance()	10	2

2.4 Reflect 模块

1 设计类标识:Reflect 设计类

方法标识符	方法名	代码行	复杂度

		(LOC)	(VG)
REFLECT-01	invoke()	27	*6*
REFLECT-02	scanJar()	20	4
REFLECT-03	getClassPathFromManifest()	25	6
REFLECT-04	getClassPathEntries()	21	6
REFLECT-05	scanDirectory()	26	7
REFLECT-06	visitWildcardType()	23	4
REFLECT-07	visitParameterizedType()	27	4
REFLECT-08	resolveType()	15	5
REFLECT-09	capture()	36	9
REFLECT-10	getGenericSuperclass()	18	4
REFLECT-11	getGenericInterfaces()	16	4
REFLECT-12	getSupertype()	20	4
REFLECT-13	isSubtypeOf()	25	8
REFLECT-14	isSubtypeOfParameterizedTyp e()	16	4
REFLECT-15	isSubtypeOfArrayType()	15	4
REFLECT-16	getTopLevelClasses()	10	3
REFLECT-17	getTopLevelClassesRecursive ()	11	3
REFLECT-18	getSimpleName()	12	3
REFLECT-19	equals()	18	2
REFLECT-20	getExceptionTypes()	11	2
REFLECT-21	visitGenericArrayType()	9	2
REFLECT-22	resolveParameterizedType()	10	1
REFLECT-23	resolve()	14	3
REFLECT-24	getSubtype()	18	3
REFLECT-25	delegate()	16	2
REFLECT-26	getOwnerTypeIfPresent()	9	3
REFLECT-27	toString()	13	1
REFLECT-28	load()	8	2
REFLECT-29	scanDirectory()	6	3
REFLECT-30	toFile()	10	2

2.6 Math 模块

1 设计类标识:Math 设计类

方法标识符	方法名	代码行 (LOC)	复杂度 (VG)
MATH-01	log2()	38	8
MATH-02	log10()	50	6
MATH-03	Sqrt()	27	5
MATH-04	SqrtFloor()	23	2
MATH-05	Factorial()	55	2
MATH-06	Binomial()	40	3
MATH-07	RoundIntermediate()	60	14
MATH-08	BigToDouble()	22	2
MATH-09	SaturatedPow()	43	8

MATH-10	Binomial()	25	2
MATH-11	Pow()	34	8
MATH-12	Divide()	42	4
MATH-13	Gcd()	21	2
MATH-14	CheckedPow()	39	11
MATH-15	IsPrime()	31	4
MATH-16	SaturatedMultiply()	19	2
MATH-17	SaturatedSubtract()	7	2
MATH-18	SaturatedAdd()	7	2
MATH-19	CheckedMultiply()	15	2
MATH-20	CheckedSubtract()	5	1
MATH-21	ListProduct()	16	2
MATH-22	Mean()	12	2
MATH-23	FuzzyCompare()	11	3
MATH-24	Factorial()	12	2
MATH-25	RoundToBigInteger()	10	2
MATH-26	LinearTransformation()	8	4
MATH-27	Equals()	13	2
MATH-28	ToString()	14	2
MATH-29	FromByteArray()	13	1
MATH-30	AddAll()	16	2

3 被测函数

根据测试策略中制定的被测方法选取标准，被测函数如表 1 所示。

3.1 Net 模块

表 3-1 Net 被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
NET-02	fromString()	21	*8*
NET-05	getHostAndPortFromBracketedHost()	30	*2*
NET-07	toString()	20	4
NET-09	equals()	20	4
NET-11	fromValid()	22	4
NET-14	ipStringToBytes()	25	16
NET-16	textToNumericFormatV6()	50	14
NET-18	compressLongestRunOfZeroes()	21	8
NET-19	forUriStringNoThrow()	20	4
NET-21	getCoercedIPv4Address()	40	6
NET-23	hextetsToIPv6String()	20	8
NET-25	findPublicSuffix()	23	6
NET-27	withParameter()	20	4
NET-29	parse()	35	8
NET-30	escape()	72	7

3.2 IO 模块

表 3-2 IO 被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
IO-01	encodingStream()	43	7
IO-02	decodingStream()	50	9
IO-03	size()	25	5
IO-04	contentEquals()	24	4
IO-05	toByteArray()	30	4
IO-06	copy()	27	6
IO-07	close()	22	5
IO-08	add()	26	15
IO-09	deleteRecursively()	33	9
IO-10	deleteDirectoryContents()	22	6
IO-11	isSubtypeOf()	25	8
IO-12	method()	28	1
IO-13	constructor()	28	1
IO-14	toGenericType()	25	3
IO-15	visit()	28	14

3.3 Collect 模块

表 3-3 Collect 被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
COLLECT-01	concat()	22	2
COLLECT-02	zip()	28	2
COLLECT-03	filterKeys()	17	5
COLLECT-04	removeOccurrences()	20	4
COLLECT-05	symmetricDifference()	46	4
COLLECT-06	getOnlyElement()	17	4
COLLECT-07	subMap()	24	8
COLLECT-08	subset()	23	8
COLLECT-09	toTable()	27	2
COLLECT-10	containsAll()	21	5
COLLECT-11	contains()	16	6
COLLECT-12	drain()	30	4
COLLECT-13	drainUninterruptibly()	39	11
COLLECT-14	retainOccurrencesImpl()	20	4
COLLECT-15	contains()	16	6

3.4 Reflect 模块

表 3-4 Reflect 被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
REFLECT-01	invoke()	27	*6*
REFLECT-02	scanJar()	20	4

REFLECT-03	getClassPathFromManifest()	25	6
REFLECT-04	getClassPathEntries()	21	6
REFLECT-05	scanDirectory()	26	7
REFLECT-06	visitWildcardType()	23	4
REFLECT-07	visitParameterizedType()	27	4
REFLECT-08	resolveType()	15	5
REFLECT-09	capture()	36	9
REFLECT-10	getGenericSuperclass()	18	4
REFLECT-11	getGenericInterfaces()	16	4
REFLECT-12	getSupertype()	20	4
REFLECT-13	isSubtypeOf()	25	8
REFLECT-14	isSubtypeOfParameterizedType()	16	4
REFLECT-15	isSubtypeOfArrayType()	15	4

3.5 Math 模块

表 3-5 Math 被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
MATH-01	log2()	38	8
MATH-02	log10()	50	6
MATH-03	Sqrt()	27	5
MATH-04	SqrtFloor()	23	2
MATH-05	Factorial()	55	2
MATH-06	Binomial()	40	3
MATH-07	RoundIntermediate()	60	14
MATH-08	BigToDouble()	22	2
MATH-09	SaturatedPow()	43	8
MATH-10	Binomial()	25	2
MATH-11	Pow()	34	8
MATH-12	Divide()	42	4
MATH-13	Gcd()	21	2
MATH-14	CheckedPow()	39	11
MATH-15	IsPrime()	31	4

4 不被测函数

对不满足测试策略中被测方法选取标准的方法将不进行单元测试，但这些方法必须经过严格代码检视，以保证不会出现一些低级性的错误，并且在集成测试阶段统一验证其接口功能的正确性。不被测函数如表 2 所示。

4.1 Net 模块

表 4-1 NET 不被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
NET-01	HostAndPort	5	1
NET-03	getHost	1	1

NET-04	getHostText	1	1
NET-06	fromParts	5	1
NET-08	withDefaultPort	7	2
NET-10	requireBracketsForIPv6	4	1
NET-12	requireBracketsForIPv6	1	1
NET-13	hashCode	1	1
NET-15	InetAddress	7	2
NET-17	textToNumericFormatV4	12	2
NET-20	convertDottedQuadToHex()	11	2
NET-22	parseOctet()	7	1
NET-24	parseHextet()	7	2
NET-26	bytesToInetAddress()	7	2
NET-28	toAddrString()	16	2

4.2 IO 模块

表 4-2 IO 不被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
IO-16	reset()	18	7
IO-17	update()	18	3
IO-18	createTempDir()	19	3
IO-19	createParentDirs()	11	3
IO-20	move()	14	4
IO-21	readLines()	17	1
IO-22	flush()	10	3
IO-23	finishLine()	8	1
IO-24	equal()	15	3
IO-25	getFileExtension()	9	2
IO-26	read()	10	3
IO-27	skip()	14	1
IO-28	initialize()	8	3
IO-29	resolveParameterizedType()	10	1
IO-30	resolveType()	14	5

4.3 Collect 模块

表 4-3 Collect 不被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
COLLECT-16	Filter()	10	2
COLLECT-17	setCountImpl()	11	3
COLLECT-18	newPriorityQueue()	9	2
COLLECT-19	newLinkedBlockingDeque()	8	2
COLLECT-20	canonical()	6	1
COLLECT-21	readResolve()	7	2
COLLECT-22	newHashSet()	5	1
COLLECT-23	makeComplementByHand()	6	1
COLLECT-24	reverse()	11	4

COLLECT-25	addAllImp()	9	2
COLLECT-26	unmodifiableNavigableMap()	11	2
COLLECT-27	containsEntryImpl()	6	2
COLLECT-28	pollNext()	9	2
COLLECT-29	checkNonnegative()	5	2
COLLECT-30	advance()	10	2

4.4 Reflect 模块

表 4-4 Reflect 不被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
REFLECT-16	getTopLevelClasses()	10	3
REFLECT-17	getTopLevelClassesRecursive()	11	3
REFLECT-18	getSimpleName()	12	3
REFLECT-19	equals()	18	2
REFLECT-20	getExceptionTypes()	11	2
REFLECT-21	visitGenericArrayType()	9	2
REFLECT-22	resolveParameterizedType()	10	1
REFLECT-23	resolve()	14	3
REFLECT-24	getSubtype()	18	3
REFLECT-25	delegate()	16	2
REFLECT-26	getOwnerTypeIfPresent()	9	3
REFLECT-27	toString()	13	1
REFLECT-28	load()	8	2
REFLECT-29	scanDirectory()	6	3
REFLECT-30	toFile()	10	2

4.5 Math 模块

表 4-4 Math 不被测函数

方法标识符	方法名	代码行（LOC）	复杂度（VG）
MATH-16	SaturatedMultiply()	19	2
MATH-17	SaturatedSubtract()	7	2
MATH-18	SaturatedAdd()	7	2
MATH-19	CheckedMultiply()	15	2
MATH-20	CheckedSubtract()	5	1
MATH-21	ListProduct()	16	2
MATH-22	Mean()	12	2
MATH-23	FuzzyCompare()	11	3
MATH-24	Factorial()	12	2
MATH-25	RoundToBigInteger()	10	2
MATH-26	LinearTransformation()	8	4
MATH-27	Equals()	13	2
MATH-28	ToString()	14	2

MATH-29	FromByteArray()	13	1
MATH-30	AddAll()	16	2

5 测试方法

根据类规约和操作规约构建测试用例，利用传统等价类划分法、边界值分析法、判定表法等黑盒测试技术对边界值、正常值、错误值等情况进行全面测试，以覆盖所有前置条件和后置条件组合。

对具有特殊需求的类辅以以下两种方法设计测试用例：

（1）根据状态转换图构建测试用例。该方法根据被测试的类的对象所处的状态以及状态之间的转移来构造测试用例，对状态之间和状态内部的每一转换及其可能发生的异常转换、转换的监护条件等进行全面测试。

（2）基于实现构建测试用例。该方法利用传统逻辑覆盖法、数据流分析法等白盒测试技术对程序的逻辑结构或数据流进行测试，以达到一定的代码覆盖率。

更详细的测试策略描述参考《单元测试说明》。

6 测试通过/失败标准

测试通过的标准表述如下：

- 所有单元测试的用例都被执行并通过；
- 所有发现的缺陷都被修正并回归测试过；
- 所有被测对象的前置条件和后置条件组合覆盖率达到 100%，或能明确给出不需要达到的理由；
- 单元测试报告被权签人批准。

测试失败标准表述如下：

- 严重缺陷密度大于 15 个/kLOC；
- 发现软件结构有重大设计问题，其修改会导致 20%以上的接口、功能、数量的变化，进一步测试相关特性已经无意义；
- 发现关键功能未被设计，该功能的设计会导致 20%以上的接口、功能、数量的变化，进一步测试相关特性已经无意义；

测试结果审批过程：

开发人员提交单元测试报告→开发经理签字并提交 SQA→SQA 对报告进行评审并签字（测试经理参与）→产品经理签字。

7 测试挂起/恢复的条件

测试挂起的条件有：

- 当某个类在单元测试执行过程中发现有阻塞用例的时候，该类的单元测试被挂起。
- 当有 20%以上的被测类都遇到有阻塞用例的时候，所有类的单元测试被挂起。
- 当出现有新增需求的时候，与该需求相关的所有类的单元测试被挂起。
- 当开发人员提出要进行设计变更的时候，相关类的单元测试将被挂起。

测试恢复的条件有：

- 测试被挂起的条件已经被解决。
- 需要恢复测试的对象达到单元测试入口条件，在这里要求这些被测对象已经通过代码走读（要提交走读报告）和语法检查（要提交检查结果）。

8 单元测试交付物

- 单元测试计划（Unit Test Plan）；
- 单元测试设计规格（Unit Test Design Specification）；
- 单元测试用例规格（Unit Test Case Specification）；
- 单元测试用例脚本；
- 单元测试驱动和桩代码；

- 单元测试执行日志（Unit Test Log）；
- 单元测试报告（Unit Test Report）。

9 单元测试任务

单元测试任务表参考表 3。

表 3 单元测试任务表

任务标识	任务描述	责任人	优先级	依赖关系
UT-TASK-001	单元测试计划写作	开发经理	高	
UT-TASK-003	单元测试计划评审	SQA	中	UT-TASK-001
UT-TASK-005	单元测试计划修改	开发经理	中	UT-TASK-003
UT-TASK-007	单元测试设计规格写作	某开发人员	中	UT-TASK-003
UT-TASK-009	单元测试设计规格评审	SQA	中	UT-TASK-007
UT-TASK-011	单元测试设计规格修改	某开发人员	中	UT-TASK-009
UT-TASK-013	单元测试用例规格写作	某开发人员	高	UT-TASK-009
UT-TASK-015	单元测试用例规格评审	SQA	中	UT-TASK-013
UT-TASK-017	单元测试用例规格修改	某开发人员	中	UT-TASK-015
UT-TASK-019	单元测试驱动、桩、用例脚本代码实现	某开发人员	中	UT-TASK-015
UT-TASK-021	驱动、桩、脚本代码走读	SQA	低	UT-TASK-019
UT-TASK-023	驱动、桩、脚本代码修改	某开发人员	低	UT-TASK-021
UT-TASK-025	单元测试执行及回归	某开发人员	高	UT-TASK-023
UT-TASK-027	单元测试报告	开发经理	高	UT-TASK-025
UT-TASK-029	单元测试报告审批	产品经理	高	UT-TASK-027

10 环境需求

10.1 硬件需求

10.2 软件需求

10.3 测试工具

Logiscope 4.0、JUnit 4.0。

10.4 其他

11 角色和职责

单元测试角色和职责参考表 4。

表 4 单元测试角色和职责对应表

角色	职责
产品经理	解决资源（包括人、工具等）需求，对单元测试结果进行监督
开发经理	制定单元测试计划，安排单元测试任务
测试经理	参与单元测试结果验收
SQA	对单元测试过程（包括代码走读、正规检视活动）进行监控
开发人员	完成单元测试需要的输入，并完成单元测试设计规格、单元测试用例规格、单元测试规程的制定，执行单元测试，记录发现问题，修改问题，并负责问题的回归测试。与此同时，负责定位问题和解决问题

12 人员及培训

- 需要 2 名一年以上工作经验的开发人员，并且他们应在详细设计开始之后全职投入到单元测试项目组中；
- 在详细设计完成之前，需要完成对项目需求、系统设计、详细设计、单元测试技术、单元测试脚本技术方面的培训；
- 在编码完成之前要完成缺陷电子流使用、测试日志表格使用、测试工具使用的培训；
- 以上培训大约需要花费每人 20 人时的工作量。

13 单元测试进度

单元测试进度安排参考表 5。

表 5 单元测试进度安排表

任务标识	任务描述	起始日期	周期/天
UT-TASK-001	单元测试计划写作	系统设计结束后 2 天内	
UT-TASK-003	单元测试计划评审	单元测试计划完成后 1 天内	
UT-TASK-005	单元测试计划修改	单元测试计划评审完成后 1 天内	
UT-TASK-007	单元测试设计规格写作	单元测试计划评审完成后 2 天内	
UT-TASK-009	单元测试设计规格评审	单元测试设计规格完成后 1 天内	
UT-TASK-011	单元测试设计规格修改	单元测试设计规格评审完成后 1 天内	
UT-TASK-013	单元测试用例规格写作	单元测试设计规格评审完成后 2 天内	
UT-TASK-015	单元测试用例规格评审	单元测试用例规格完成后 1 天内	
UT-TASK-017	单元测试用例规格修改	单元测试用例规格评审完成后 1 天内	
UT-TASK-019	单元测试驱动、桩、用例脚本代码实现	单元测试用例规格评审完成后 2 天内，并且编码阶段已经开始	
UT-TASK-021	单元测试驱动、桩、脚本代码走读	单元测试驱动、桩、用例脚本代码完成后 1 天	
UT-TASK-023	单元测试驱动、桩、脚本代码修改	单元测试驱动、桩、脚本代码走读后 1 天	
UT-TASK-025	单元测试执行及回归	单元测试驱动、桩、脚本代码走读后 2 天，并且编码阶段已经结束	
UT-TASK-027	单元测试报告	单元测试执行及回归完成后 1 天内	
UT-TASK-029	单元测试报告审批	单元测试报告完成后 1 天内	
	风险预留时间	单元测试阶段工作中任意时候	
	单元测试阶段里程碑时间点	2009-11-15	

14 风险和应急计划

风险和应急计划安排见表 6。

表 6 风险和应急计划

风险 ID	风险描述	责任人	优先级	规避措施	应急计划
1	人员无法及时到位	开发经理	高	1. 在产品的预算中体现这部分需求 2. 定期催促人力资源部进行资源协调 3. 从可能空闲的产品部中物色人员	1. 推迟进度计划 2. 进行招聘 3. 考虑工作外包
2	人员技能不符合要求	开发经理	中	1. 在人力预算中给出人员技能要求 2. 对提供的人员进行技能面试 3. 从其他产品部门协调有能力的人员	1. 提高培训的强度 2. 加强培训效果监控 3. 对工作输出加强检视
3	测试工具无法到位	某员工	低	1. 在产品预算中尽早体现这部分需求 2. 联系其他产品，看是否有空闲的 License 3. 尽早联系工具代理商，	1. 采用人工分析程序规模 2. 允许使用规模估计值 3. 自行开发测试

				洽谈采购事宜	工具

15 审批

计划提交人签字：　　日期：

开发经理签字：　　日期：

产品经理签字：　　日期：