



# Intent Based Test Case Scripting

**Tool-Agnostic "Intent/Idea → Test Case" Transformation Service**

From Natural language to the "Activity plan" and from there to "Test case"

[Challenge](#) | [Solution](#) | [How it works](#) | [Platform capabilities](#) | [5 steps](#) | [Glossary](#)

Can Davutoglu, February 2026





## The QiTASC promise

**While your domain knowledge becomes  
the script, we make the underlying  
toolchain invisible.**





## Challenges

## Benefits

### Technical limitations

Limits for intricate scenarios. Understand the context of ambiguous intents. Framework dependency coupling. Non-deterministic output variations.

### Organization

Team skepticism of AI-generated code. Workflow disruption requires training. Governance: approval and version control. Skill transition: testers become AI supervisors.

### Process overhead

Significant initial knowledge base setup. Substantial training data corpus required. Human review burden initially. Two-stage latency vs. direct generation.

### Quality risks

Over-reliance reduces manual testing skills. Syntactically correct but logically flawed tests. AI reproduces training biases. Generated tests are harder to debug.

### Efficient development

Minutes vs. hours for test creation. Standardized patterns, reduced variance. Generate hundreds of test variations. Template updates affect all future tests.

### Team empowerment

Junior testers produce senior-quality tests. Product owners, BAs contribute directly. Tribal knowledge codified. New team members productive immediately.

### Business value

Parallel test development with features. Less manual effort, fewer rework cycles. Comprehensive testing, early defect detection. Fast test adaptation to requirement changes.

### Improved quality

AI suggests edge cases humans miss. Best practices encoded in knowledge base. Built-in syntax/logic validation. Self-documenting through manual plan.

# 3 reasons why your intent-based test case scripting will be successful.

1

Separate concerns in 2 stages: Stage 1 defines what to test while Stage 2 handles execution (how). This creates a clear validation checkpoint with human review, dual validation (content and syntax), and iterative refinement to reduce errors.

2

The Intent → Manual Plan → Executable code flow provides a transparent audit trail for compliance and requirements mapping, while enabling non-technical stakeholders to validate business logic before code generation.

3

By decoupling intent from implementation, StepDef libraries, DSLs, and execution frameworks can change independently. It allows domain knowledge to evolve and multiple executable outputs to be generated from the same manual plan.



# Comparison: Traditional <> Intent-based scripting

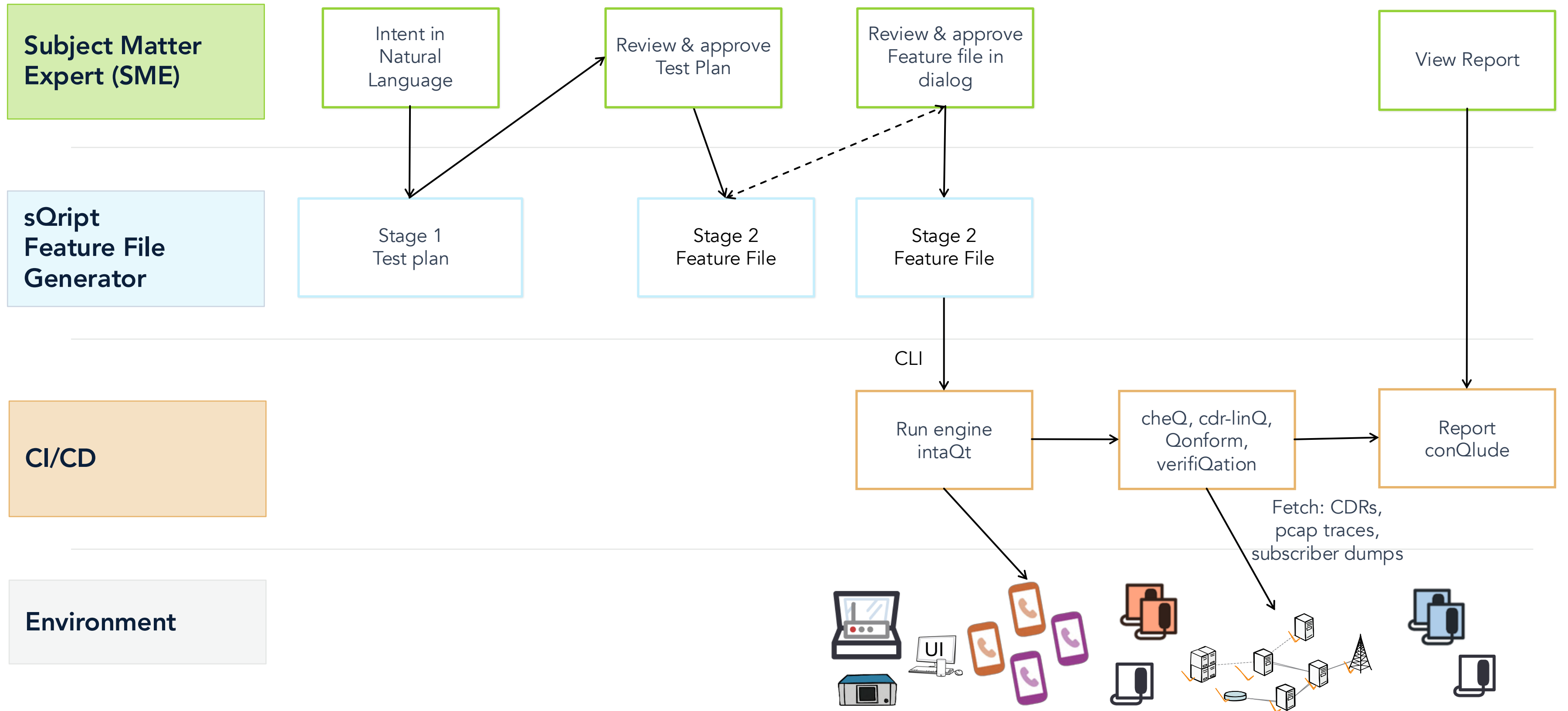


Feature	Traditional (Human-Centric)	Intent-Based (System-Centric)	The Strategic Benefit
<b>Primary Focus</b>	"How" to script (Technical implementation)	"What" to test (Business intent)	<b>Higher business value:</b> Focus shifts to logic, not syntax.
<b>Required Skillset</b>	Technical scripting & tool expertise	Domain knowledge & product logic	<b>Democratization:</b> More team members can contribute.
<b>Script Style</b>	Individual & varied: Every tester has their own "style" and quirks.	Architectural consistency: Every file follows exact same pattern and rules.	<b>Low cognitive load:</b> Faster onboarding and easier peer reviews.
<b>Maintenance</b>	Manual & brittle: Changes require hunting through thousands of lines of code.	Automated & Resilient: Update the "Intent" or "StepDef," and scripts regenerate.	<b>Reduced technical debt:</b> Prevents "script rot" over time.
<b>Quality Control</b>	Manual peer reviews (prone to human oversight)	Multi-agent validation (Syntax + Content Checkers)	<b>Industrial reliability:</b> Continuous, 24/7 quality enforcement.
<b>Scalability</b>	Linear (More tests = more people needed)	Exponential (One expert manages many AI-generated tests)	<b>Force multiplier:</b> Dramatically higher output per person.



# How it works

## Process for intent-based test case scripting



## SME Value Transformation & Benefits

### Before: SME Time Allocation

Learning test frameworks & syntax	30%
Writing & debugging test code	40%
Maintaining existing tests	20%
Strategic test design & edge cases	10%

### After: SME Time Allocation

Identifying edge cases & scenarios	40%
Domain expertise & validation	35%
Test strategy & coverage planning	20%
Reviewing generated tests	5%

### Result:

SMEs spend 95% of time on strategic value activities vs. 10% before. They become test architects, not test coders.

How it works

## Strategic benefits in numbers



**10x**

faster test creation



**60%**

cost reduction



**3x**

coverage increase



**80%**

defect reduction



**50%**

faster time-to-market



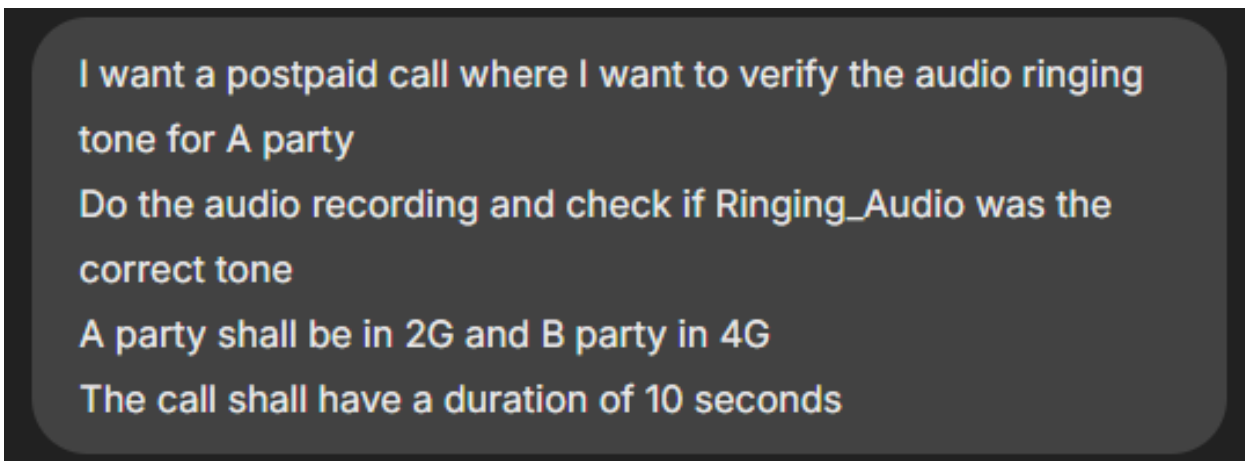
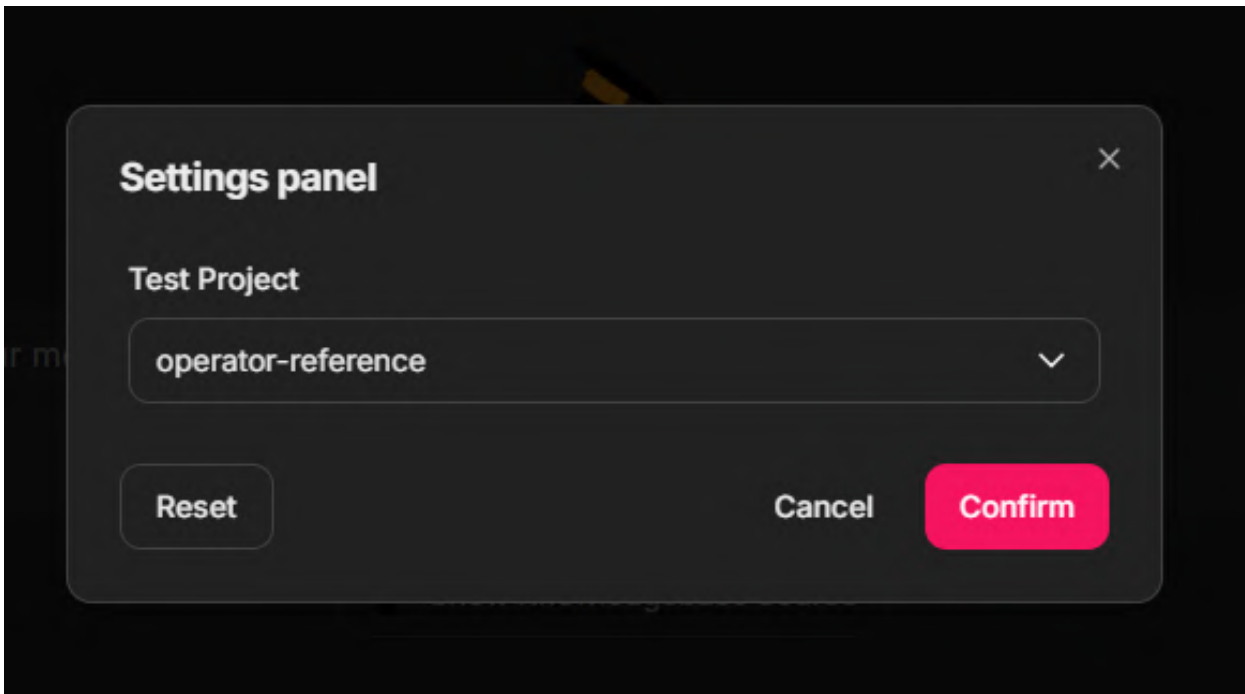
**100%**

team enablement

## How it works

# Stage 1

1<sup>st</sup> step: Test Plan generation as an intermediate step



## How it works

# Stage 2

2<sup>nd</sup> step: Feature file has been generated

```
#-> Prerequisites

#-> Resource Allocation
Given a phone as A:
    * with profile POSTPAID_2G
Given a phone as B:
    * with profile POSTPAID_4G

#-> Phone specific preparation
And ensure settings on phone B:
    * without call forwarding
And ensure settings on phones A,B:
    * with airplane mode enabled
And wait for 5 seconds
And ensure settings on phones A,B:
    * with airplane mode disabled
And store date "StartDate"

#-> Execution

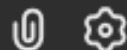
#-> Evidence
And start trace with profile "tcap"

#-> Action
And start audio recording on A
When A dials the number B.number
And within 30 seconds, B detects an incoming call from A.number
And within 30 seconds, A connects
And wait for 10 seconds
And A ends the call and, within 15 seconds, he and B disconnect
And stop audio recording on A and save as RING_REC
And stop trace

#-> Verification

And ensure audio recordings match:
    * recording = "RING_REC"
    * reference = "Ringing_Audio"
    * minimum similarity score = 20%
And store date "StopDate"
And set cdr search for phones "A,B"
```

Type your message here...



## Tasks

Done

- 1 ✓ Thinking...
- 2 ✓ Thinking...
- 3 ✓ Searching documentation for 'Profiles'
- 4 ✓ Searching documentation for 'Ensuring Device Settings on Phones'
- 5 ✓ Searching documentation for 'Airplane mode enabled/disabled'
- 6 ✓ Searching documentation for 'Call Forwarding'
- 7 ✓ Searching documentation for 'Trace collection and validation'
- 8 ✓ Searching documentation for 'Make Call'
- 9 ✓ Searching documentation for 'Starting Audio Recording on a Device'
- 10 ✓ Searching documentation for 'Detecting Incoming Calls on a Device'
- 11 ✓ Searching documentation for 'Set CDR search values'
- 12 ✓ Searching documentation for 'Store dates for CDR search'
- 13 ✓ Searching documentation for 'Verify'
- 14 ✓ Thinking...
- 15 ✓ Searching documentation for 'Audio Recording'
- 16 ✓ Thinking...
- 17 ✓ Thinking...
- 18 ✓ Searching documentation for 'Ensuring Device Settings on Phones'
- 19 ✓ Searching documentation for 'Airplane mode enabled/disabled'
- 20 ✓ Searching documentation for 'Call Forwarding'

## How it works

# Stage 2

---

Dialog to optimize the test case,  
add one more verification

I would like to check within the call if A is still in 2G and B is still in 4G



```
And start audio recording on A
When A dials the number B.number
And within 30 seconds, B detects an incoming call from A.number
And within 30 seconds, A connects
And wait for 10 seconds
And verify A.network == "2G"
And verify B.network == "4G"
And A ends the call and, within 15 seconds, he and B disconnect
And stop audio recording on A and save as RING_REC
And stop trace

#-> Verification

And ensure audio recordings match:
```

# Project Setup: Required Activities for new project



Category	Activity	Responsibility
Strategy & Logic	<b>Domain Prompt Template(s):</b> Updating the core instructions to include new business rules, product logic, and domain-specific constraints.	Prompt Engineer / Business Analyst
Technical Core	<b>StepDef Library + Conventions:</b> Developing new code building blocks and defining how parameters (e.g., IDs, prices) must be passed.	Test Automation Engineer
Knowledge Base	<b>Example Feature Files / Training Corpus:</b> Curating a "Golden Set" of new test cases to serve as reference points (Few-Shot Learning) for the LLM.	Domain Expert / QA Lead
Quality Gates	<b>Validation Rules (Syntax + Content):</b> Adjusting the "Checker Agents" to recognize new technical syntax and validate against updated business requirements.	QA Architect / AI Engineer
Infrastructure	<b>CI/CD Pipeline Stages and Gates:</b> Updating the automation triggers and ensuring the verification loop is integrated into the deployment flow.	DevOps Engineer
Enablement	<b>Documentation (README, Onboarding):</b> Updating manuals and guides to reflect new system capabilities and instructions for the users.	Technical Writer / Product Owner
Environment	<b>Test Data &amp; Env Dependencies:</b> Aligning data availability and backend system states with the new test scenarios to ensure execution.	Environment Manager / Data Architect



---

**Get in touch, and consolidate your tool landscape!**

**Can Davutoglu**

Founder & CEO

Mail [can.davutoglu@qitasc.com](mailto:can.davutoglu@qitasc.com)

Web [www.qitasc.com](http://www.qitasc.com)

# Glossary (A-Z)

---



**4G/5G:** Fourth-/fifth-generation mobile access

**ACS:** Auto Configuration Server

**ADSL:** Asymmetric Digital Subscriber Line

**CI:** Continuous Integration

**CLI:** Command-Line

**CPE:** Customer Premises

**DSCP:** Differentiated Services Code

**DSL:** Digital Subscriber

**E2E:** End-to-end

**FTTx:** Fiber-to-the-x (fiber access variants)

**GHz:** Gigahertz (Wi-Fi bands: 2.4/5/6 GHz)

**IP:** Internet Protocol

**IPTV:** IP Television (service validation, KPIs)

**KPI:** Key Performance Indicator

**LAN:** Local Area Network

**MOS:** Mean Opinion Score (voice QoE metric)

**NAT:** Network Address Translation

**PCAP/pcap(s):** Packet capture file(s)

**PDU:** Power Distribution

**QoE:** Quality of Experience

**QoS:** Quality of Service

**RAL: Router Access Layer**

**RBAC: Role Based Access Control**

**RCA:** Root Cause Analysis

**RCE:** Router Configuration Engine

**RSSI:** Received Signal Strength Indicator

**SDSL:** Symmetric Digital Subscriber Line

**SSID:** Service Set Identifier (Wi-Fi network name)

**SSH:** Secure Shell

**TCP:** Transmission Control Protocol

**TR-069:** Broadband Forum CPE WAN management protocol

**TWAMP:** Two-Way Active Measurement Protocol

**VLAN:** Virtual LAN

**VoIP:** Voice over IP

**WAN:** Wide Area Network

**Web-UI:** Web-based user interface

**Wi-Fi:** Wireless LAN

**p95:** 95th percentile