Intent-Explicit Behavior Specs for Domain Functions

David Gelperin
ClearSpecs Enterprises

Rapid development tools enable the implementation of many software functions in a few hours or days. As implementation costs and times have decreased, the need to specify domain functionality has decreased. Sometimes, it is smarter to build something quickly in response to vague descriptions so customers and developers can gain deeper understanding of the real requirements.

If specification of a domain function is useful, e.g. because of mission criticality or functional complexity, consider using **Intent-Explicit Behavior (IEB)** specs just in time. IEB specs use simple algebra, rather than explicit values, to express intent. They contain details about intent so a manual or automated test can be created to fulfill that intent. For example, a variable might be assigned the value "maximum" or "minimum" rather than an explicit value. Test details like setup and wrap-up are not included. These specs record common understanding about the details of a domain function.

Current situation

Many Agile methods specify requirements with user stories including acceptance criteria, stakeholder conversations, and executable tests. Agile projects are free to use other types of "useful" specifications. From an information granularity perspective, Agile uses course-grained (user stories) and fine-grained (acceptance criteria and executable tests) specifications. The team decides whether to rely solely on these to discover and communicate functional details or to use other (unidentified) types of specifications. When the function being developed is simple or familiar to the developers, conversations and tests may be enough.

When a function is neither simple nor familiar, we believe some form of medium-grained specification is useful to clarify the details. We propose that IEB specs fill this role.

IEB specs

These specs combine elements of:

- User stories
- Use cases
- Meaningful variable names
  e.g. "loan amount"
- Simple algebra
  e.g. loan amount up by (requested amount – account balance)
- Simple logical expressions
  e.g. requested amount available
  & requested amount >= min withdrawal
  & requested amount <= remaining daily limit

& requested amount <= (account balance + credit limit – loan amount)

- Natural language, both controlled and uncontrolled <in comments>
- Action contracts
- Pseudocode (for post-condition alternatives)
- Logic tables

They enclose comments and references in < >.

IEB specs can be used to create comprehensive boundary-value test patterns. In addition, they contain sufficient detail to create (1) tests for an array of test platforms e.g. FitNesse or Cucumber, and (2) rules for an array of business rule engines.

IEB specs are not appropriate for detailing the requirements for algorithm-intensive functions such as compiler functions, data mining functions, constraint-programing functions, graphic functions, or audio functions. They are unnecessarily complex for specifying the behavior of reactive systems such as flight control systems.

IEB spec example

Start IEB spec

System: ATM

As an account holder (AH)
I want to get cash from an ATM
So I can have cash anytime I want it

Risk factors

a. Regularity of Usage = periodic	< uniform, periodic, irregular >
b. Range of Usage = medium to heavy	< heavy, medium, light >
c. Failure Impact = high	< high, medium, low >

Basic success path: <Basic paths are optional, but useful for interactive systems>

<ATM steps reference their associated action contracts and logic tables>

AH inserts ATM card for scanning

ATM displays successful scan message
<**unsuccessful scan** action contracts>

AH enters PIN

ATM acknowledges valid PIN and displays menu of functions
<**invalid PIN** action contract>

AH selects withdrawal and specifies amount

ATM dispenses requested amount and adjusts AH balances
<**successful and unsuccessful withdrawal** action contracts>

Action contracts:

1. **unsuccessful scan** action contracts

   Constant conditions: **Card state**

Card state	Displayed message	Card disposition
scan failed	scan failed, please try again	returned
inactive	please activate card	returned
suspended	card retained	kept
cancelled	card retained	kept

2. **invalid PIN** action contract

   Constant conditions: **PIN**

PIN	Displayed message	Card disposition
invalid	PIN invalid, please try again	returned

3. successful and unsuccessful ATM withdrawal action contracts

<*minimum withdrawal* is the smallest amount that a specific ATM will dispense. It is constant for a specific ATM, but may differ for different ATMs>

<Each AH has a *daily ATM withdrawal limit*. Different AHs may have different limits.>

Action	**Dispense requested amount**	**Deny withdrawal request**
Pre-conditions	requested amount available & requested amount >= min withdrawal & requested amount <= remaining daily limit & requested amount <= (account balance + credit limit – loan amount)	
Constant conditions	ATM sign-on successful & withdrawal request verified OK	ATM sign-on successful & (withdrawal request verified NG OR requested amount unavailable OR requested amount < min withdrawal OR requested amount > remaining daily limit OR requested amount > (account balance + credit limit – loan amount))
Post-conditions	OK withdrawal message and menu of functions displayed & remaining daily limit down by requested amount & CASE 1: **if** requested amount <= account balance **then** account balance down by (requested amount) CASE 2: **if** account balance = 0 **then** loan amount up by (requested amount) CASE 3: **otherwise** loan amount up by (requested amount – account balance) & account balance set to 0	Denial message displayed as "invalid withdrawal request" or "insufficient cash in ATM" or "requested amount too small" or "requested amount over daily limit" or "insufficient funds in your account"

End IEB spec

Advantages of IEB Specs

The goal of IEB specs is "extreme" readability. A set of IEB specs is easier to check for adequacy than a large set of executable test cases. Early development and stakeholder review of IEB specs and their associated test cases is a powerful educational and communication strategy since these specs and their derivable test cases provide detailed, real-world examples of required behavior.

Action Contracts include BDD scenarios.

Specifying requirements with IEB specs has the following advantages:

- Behaviors are specified with essential and precise details
- The intent behind each behavior is clearly stated
- An action contract can be used to produce a rule-covering, boundary-valued suite of test patterns.
- Review of test patterns can be used to validate the specs and show developer understanding of the associated functionality
- Increases application testability when the pre, constant, and post conditions of the action contracts are inserted in the code as assertions
- Validated test cases can be used to produce tests for an array of test platforms
- Validated specs can be used to produce rules for an array of business rule engines
- Effort estimates will be more accurate because of the details in action contracts
- Specs can be incrementally developed i.e., user stores and basic paths, then action contracts and logic tables

Developing IEB specs is related to, but differs from test-driven development (TDD, BDD, ATDD). These other approaches entail the specification of executable test, but executability sacrifices readability because intent is replaced by explicit values, which may be arbitrary.

Reverse engineering intent from executable tests may be a challenge and is unnecessary with IEB specs. Why would 05/01/2016 and 09/30/2016 be in a test set?