

# Agile2023

THE SCOTLAND EXPERIENCE

PRESENTED BY



JOIN US TODAY!

#AGILE2023SCOTLAND

LEARN MORE



# Gaspar Nagy

## Tree in the Forest: Managing Details in BDD Scenarios



Agile  
Alliance  
PRESENTS

Agile 2023  
THE SCOTLAND EXPERIENCE

#AGILE2023SCOTLAND

bdd newsletter: [bddaddict.com](http://bddaddict.com)

CAUTION!

**bdd addict**

on air

given . when . then

# Gáspár Nagy

coach, trainer and bdd addict  
creator of SpecFlow

[gaspar@specsolutions.eu](mailto:gaspar@specsolutions.eu)

<https://specsolutions.eu>

@gasparnagy

<http://bddbooks.com>



# Today...

- Super brief BDD intro
- Scenarios as Tests
- Scenarios as Specification
- Types of Details
- Implicit Context
- Wrap-up, Q&A

*BDD in 3 minutes...*

# BDD scenario: bridge between requirements and the solution



make requirements testable

Scenario: User votes up a question  
Given a question asked with 2 votes  
And the user is authenticated  
When the user votes up the question  
Then the votes should be changed to 3



make tests understandable

***Behaviour Driven Development*** is about  
understanding, documenting & validating  
business requirements  
through illustrative scenarios

# Scenarios are discovered through examples...

“Examples” are a lightweight form of describing the “scenario”

They fit better for collaborative discussions

**Another question contains the same word**

- Questions:

- 1) What is SpecFlow
- 2) What is Cucumber

\* Start asking: “Best SpecFlow practices”  
=> Suggested: 1)

discovery

Having discussions based on such complex scenarios is hard...

Scenario: Another question contains the same word  
Given there are questions asked as

```
| title                |  
| What is SpecFlow?  |  
| What is Cucumber? |
```

And the user is authenticated

*formulation* When the user starts asking a question as

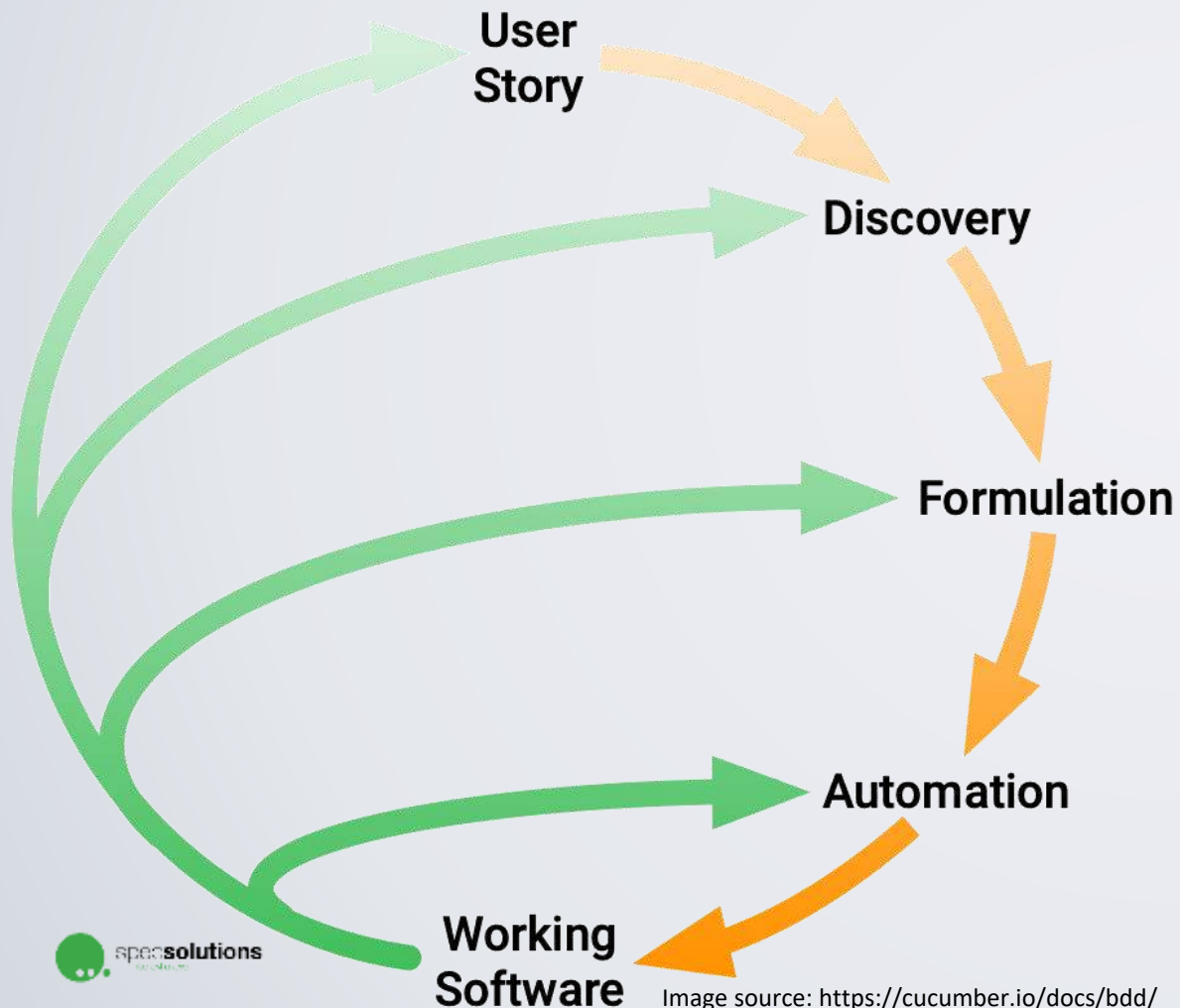
```
| title                |  
| Best SpecFlow practices |
```

Then the suggestions list should be

```
| title                |  
| What is SpecFlow?  |
```



# BDD Practices



## Discovery

Shared understanding is established through collaboration and structured conversations

## Formulation

The examples of system behaviour are documented as scenarios

## Automation

Scenarios are automated to be able to verify the behaviour of the system

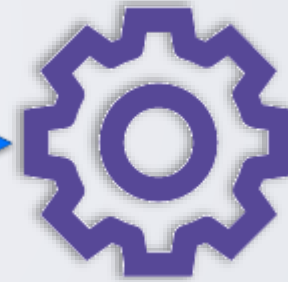
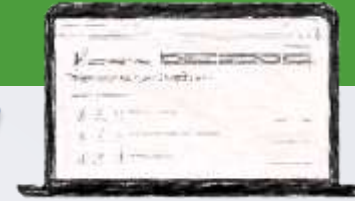
# Purposes of scenarios

Scenario: The one where...  
Given some context  
When perform an action  
Then the outcome happens

illustration of  
the requirements

automated test

documentation of  
the behavior



implementation phase  
(test-driven)

maintenance phase  
(regression)

BDD is not (only) testing!

# *Scenarios as Tests*

# In order to use scenarios as tests...

- Executable – In order to gain trust in that we deliver what was agreed
- Objective – the execution should always be performed in the same way, otherwise we are not testing the same thing
- Concrete – at the end the application has to be exercised based on the scenarios with concrete details

All *details* that are required for exercising the application has to be available *during execution* latest

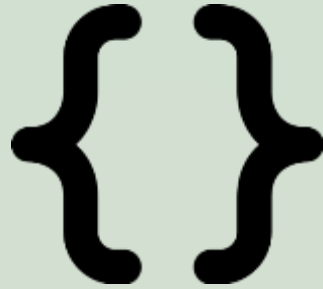
# This brings up a couple of interesting questions...

- What are those *details*? What kind of details we should consider?
- How do we agree on those details?
- Where do we document (store/code) those details?

# Where to document the details?



In the text of  
the **scenario** in  
the feature file



In the  
automation  
**code**



In **external**  
sources (Excel,  
existing data,  
etc.)

managed by code

The details are not removed from the scenario, but  
**pushed down** to the code



# ONLINE POLL: Pro / Con analysis for details

# Scenario

## *Details in Gherkin*

### Pro

- Visible
- Easier test result analysis (debugging)

### Con

- Makes scenario hard to read
- Maintainability problems

VS

# Code

## *Details coded or processed by code*

### Pro

- Easy to reuse
- Different abstractions can be defined
- Maintainable
- Visible during execution (dynamic visibility)

### Con

- Harder to see details in code (static visibility)
- Code is not a shared asset

# Without details, how do we know what data was used for testing?

*# Including question and user details...*

Scenario: The user votes up a question

Given there is a question asked as

title	body	votes
How to write better BDD scenarios?	I need help	2

And the user "Marvin" is authenticated with password "1234"

When the user votes up the question

Then the vote count of the question should be changed to 3

*# Omitting incidental details*

Scenario: The user votes up a question

Given there is a question asked with 2 votes

And the user is authenticated

When the user votes up the question

Then the vote count of the question should be changed to 3

Given there is a question asked with 2 votes

Question: Title: T8f43d0a6410141e ..., Votes: 2, AskedBy: Jeltz

-> done: QuestionStepDefinitions.GivenThereIsAQuestionAskedWithVotes(2)...

And the user is authenticated

Perform Login: Input: Name: Marvin, Password: 1234

-> done: AuthStepDefinitions.GivenTheUserIsAuthenticated() (0.0s)

When the user votes up the question

Perform VoteQuestion: Input: (16b5df88-0311-4b9d-bb86-2bf617c11cdf, Up)

-> done: QuestionVotingStepDefinitions.WhenTheUserVotesUpTheQuestion(Up) (0.0s)

Then the vote count of the question should be changed to 3

-> done: QuestionVotingStepDefinitions.ThenTheVoteCountOfTheQuestion...

It is easier to manage details in code,  
but we need to work extra for visibility

# Scenarios as Specification

# How to make scenarios to good specification

- Easy to read, review & discuss
  - Not too long (brief)
  - Free of “noise”
  - Business language (ubiquitous language)
- Should only change when the requirement changes
  - No technical/solution details
  - Focus on the what (intention) and not the how (steps to achieve)
- Concrete, so that the validity can be checked
  - Contain essential (relevant) details
  - Contain real/concrete data
  - Should focus on a single business rule

# 6 BRIEF principles of good scenarios

- **Business language** — Business terminology aids cross-discipline collaboration
- **Real data** — Using actual data helps reveal assumptions and edge cases
- **Intention revealing** — Describe the desired outcomes, rather than the mechanism of how they are achieved
- **Essential** — Omit any information that doesn't directly illustrate behaviour
- **Focused** — Each scenario should only illustrate a single rule
  
- **Brief** — Shorter scenarios are easier to read, understand and maintain

# What details are essential?

*There are no fixed guidelines to decide on this... and the answer might be context specific...*

- The details that influence the outcome (=Then steps)
- The details that are relevant for the “rule” (=AC, business rule, requirement) the scenario illustrates (“focuses on”, purpose)



Deciding on essential details in scenarios without clear outcome and purpose is not possible

You need to fix those first

# Types of Details

# EXERCISE:

## Identify different detail types

- Work in pairs or 3 people mini-groups
- Find at least one example of the detail types on the right
- Is that particular detail necessary in the scenario?

# Many details are unnecessary

Rule: Dispatched orders cannot be modified in the app

Scenario: Customer attempts to modify a dispatched order

Given the Earth is spinning

And there is electricity for the servers

And the system has been started

And the system has been initialized

And the system is not in maintenance mode

And there are the following products

name	unit price	status
Flipchart	250	active
Sticky Notes	18	active

And there is a customer

name	user name	password
Agile Alliance	agall	secret

And the customer "Agile Alliance" lives at

street	city
18 Holyrood Park Road	Edinburgh

And the customer "Agile Alliance" has an order

And the customer "Agile Alliance" has an order

status	payment method	issued
created	bank_transfer	7/20/2013

And the order "0293445" has an item

item id	product	quantity
743268	Flipchart	13

Rule: Dispatched orders cannot be modified in the app

Scenario: Customer attempts to modify a dispatched order

Given there is a customer with a dispatched order

When the customer attempts to start changing in the order

Then the changes should fail with "error-cannot-modify-dispatched-orders"

*The data is not always unnecessary!*

Rule: The countries we cannot ship ("no-ship") cannot be selected

Scenario: Customer attempts to select a no-ship country

Given a customer started an order

When they select "North Pole" as shipping country

Then the country selection should fail with "no-ship-country"

# Detail types

## Scenario Related

## Implicit Context Related

1. Entity property

2. Entity existence

3. Hierarchical data

4. Technical details

5. Contextual details (scenario execution context)

6. Baseline

7. Workflow steps

8. System status

# 1. Entity property

Given the customer has an order

status	payment method	issue date	due date	
created	bank_transfer	7/25/2023	8/24/2023	

*used defaults*

Given the customer has an order

*defaults can be overridden*

Given the customer has a **dispatched** order

Given the customer has an order

status	
dispatched	

- Many properties of the entity might not be relevant for the scenario
- These properties change and extended independently of the scenario
- Push down: maintain a default (test) values for the properties of the entities in code

## 2. Entity existence

Given there is a customer

And the customer has an address

street	city	
18 Holyrood Park Road	Edinburgh	

Given there is a customer

*pre-populate with  
default address*

*required prerequisites*

Given there is a customer

And there is an order of the customer

Given there is an order

*ensure customer  
prerequisite*

- The existence of some entities might be required for execution, but not relevant for the outcome (e.g. address record of a customer; customer for an order)
- These requirements might change independently of the scenario
- Push down:
  - populate the records with defaults automatically
  - ensure that all prerequisite is created

# 3. Hierarchical data

Given there is a customer "AA"  
And the customer "AA" has an order "0293445"  
And the order "0293445" has an item

item id	product	quantity
268	Flipchart	13

And the order item "268" has a comment "..."

Given there is an order with

product	quantity	comment
Flipchart	13	...

*relevant  
details only*

- Some tests require a hierarchical set of data
- Complex data hierarchies are hard to understand, so should be simplified
- Push down:
  - ensure that prerequisite is created
  - build up hierarchy as a story by relevant steps
  - use custom format to express (e.g. yaml)
  - use internal IDs



# 4. Technical details

When the user clicks on button

```
"/[*[@id="orders_0293445"]/td[3]/a"
```

*identified  
business action*

When the user attempts to start changing in the order

- Technical details and steps might be needed to execute a particular business action (e.g. enter the name to a textbox with ID “tb1234”)
- These details are bound to the implementation, not the specification
- Documenting this details in scenarios requires intensive maintenance
- Push down:
  - Identify & express business needs or actions
  - Translate these to the technical details in code (reusable!)

# 5. Contextual details (scenario execution context)

Given there is a customer "AA"  
And the customer "AA" has an order "0293445"  
When the customer "AA" changes  
the order "0293445"

Given the customer has an order  
When they change the order

refer to  
context

*can use more explicit form as well*

Given the customer has an order  
When the customer changes the order

- The “story” that the scenario explains might need to introduce details that are referred in later steps – they are part of the scenario execution context
- Duplicating these details makes the scenario verbose and hard to understand
- Push down:
  - Store these contextual details in scenario-lifetime storage provided by the BDD framework (World, ScenarioContext)
  - Resolve the references to these details from the storage

# Detail types

## Scenario Related

## Implicit Context Related

1. Entity property

2. Entity existence

3. Hierarchical data

4. Technical details

5. Contextual details (scenario execution context)

6. Baseline

7. Workflow steps

8. System status

# *Implicit Context*

# Anatomy of a scenario

- Given -> Context
- When -> Action (triggering action)
- Then -> Outcome

The “Given” step(s) specify the context (preconditions, system state) that is required to specify in order to perform the action and verify the outcome.

# Do we always have a context?

Given the Earth is spinning  
And the system has been started  
And there are the following products

name	unit price	
Flipchart	250	
Sticky Notes	18	

And there is a customer with a dispatched order

Assumed	Needs automation actions	Make it implicit?
Yes	No	Yes, always
Yes	Yes	Yes, always
Maybe	Yes	Depends...
No	Yes	Never

# Implicit

## Context

### Pro

- Maintained by code
- Can hide technical details

### Con

- Requires team agreement & knowledge to understand the scenarios
- Tracking the dependencies in code might be more complex

VS

# Explicit

## Context

### Pro

- Can be used to declare essential details
- Highlights the necessity of the prerequisite

### Con

- Might cause unnecessary duplication in scenarios
- Scenarios require more maintenance
- Makes scenarios longer and harder to understand

Implicit context is always present

Whether you make it explicit is a team decision, that  
you might need to document



# Detail types

## Scenario Related

## Implicit Context Related

1. Entity property

2. Entity existence

3. Hierarchical data

4. Technical details

5. Contextual details (scenario execution context)

6. Baseline

7. Workflow steps

8. System status

# 6. Baseline

Given there are the following products

name	unit price
Flipchart	250
Sticky Notes	18

And there is an order with

product	quantity
Flipchart	13

Given there is an order with

product	quantity
Flipchart	13

*Flipchart and  
Sticky Notes are  
always there*

- The scenarios of a product might need “usual” data for expressing the examples
- Many team has an implicit agreement to use the same data over and over, so it might became noise
- Push down:
  - Make a team agreement of “default” test data and make sure those can be used without explicitly listing them
  - Special scenarios can still override
  - There might be multiple baseline sets

# Personas in “Baseline” details

**NAME:** Femi **AGE:** 35

**FROM:** Lagos, Nigeria

**LIVES:** Lagos, Nigeria

**OCCUPATION:** Entrepreneur

**BIOGRAPHY**  
Femi is the founder of EduChap, a children's education app. Coming from a long line of educators, he believes deeply in the role of education to empower the next generation and transform his country.

After getting his MBA from the prestigious University of Lagos, Femi worked at a computer business owned by a family friend. Though his salary was modest, Femi worked hard and spent little. After six years, he left and, with his savings and some support from family, started EduChap.

Today, most of Femi's internet use is for work. Running a startup, he is working seemingly 24/7. Checking and sending email on his phone is often the last thing he does every night before falling asleep and the first thing he does when he wakes up in the morning.

Given how much he is on the move for sitting in Lagos traffic, his internet usage largely happens via his mobile phone. His iPhone 6 is his primary gateway online—he uses it for email, browsing, LinkedIn, and constantly checking on EduChap. His Blackberry is mostly for social media—he likes Twitter and Facebook for news—and for managing contacts. His MacBook Air is kept at the office for word processing, presentations, and some light browsing. He spends ₦5,000 to ₦10,000 (\$17.70 to \$35.40) on monthly data plans and can use up to 1 GB in a day. Last month, his friend told him about a great new MyFi service in Lagos. Femi needs ₦18,000 to get set-up on the plan and is willing to give it a shot after his current plan expires. He had tried a couple of these types of plans in the past and had left them because the service deteriorates after a while.

Femi knows that for EduChap to be successful he must invest in his own professional growth. He is currently trying to improve his coding skills (via international sites such as lynda.com, w3schools.com, etc.) and knowledge about primary school curricula and pedagogy (via basic online research to find relevant articles and events).

Femi developed a fondness for Wikipedia at UNILAG. His professors would tell him not to use, but Femi couldn't understand why, beyond the fact that they wanted to profit from their own textbook sales. He also noticed his professors using Wikipedia material for their lectures and thought this double standard was unfair. He eventually figured out a loophole: he would use Wikipedia content, but just to cite other sources. Today, a modified version of his practice continues: he uses Wikipedia content for EduChap modules and supplements it with information from other sources.

**DEVICE USE**

**iPhone 6**  
**PRIMARY USE:** Voice calling, emailing, LinkedIn, and EduChap.  
**NETWORK:** Etisalat. Spends ₦5,000 (\$17.70) on a 5GB monthly plan for his smartphone and to tether his other devices. Spends another ₦1,500 (\$5.30) weekly for calls.  
**APPS USED:** In order of preference: Twitter, Facebook, WhatsApp, Instagram, Wikipedia.

**Blackberry Z10**  
**PRIMARY USE:** Social media, and voice calling.  
**NETWORK:** Etisalat. Spends ₦2,000 (\$7.70) for data on his Blackberry and ₦500 (\$1.77) on airtime top-up each month. Taps up 2 to 4 times a month.

**iPad**  
**PRIMARY USE:** Reading, taking and looking at photos.  
**DETAILS:** Not data-enabled, only accesses internet through WiFi and tethering. iPad was a status symbol—frequently carried, but with limited usage.

**Macbook Air**  
**PRIMARY USE:** Primary desktop device for word processing, presentations, etc.  
**DETAILS:** Kept at the office, primarily for typing, preparing presentations, and light browsing while at his desk.

**REBOOT**

Low Digital Confidence ————— High Digital Confidence

Low Economic Status ————— High Economic Status

Absenteeism of Wikipedia

Access to Internet

Picture source: Wikipedia

- A persona is a fictional character created to represent a user type that might use a [...] product in a similar way. (Wikipedia)
  - Well known to the team
  - Helps discussions as it saves time of explaining the context
- Similarly you might have fictional, well-known data that we use in discussions
  - Jenny Martin refers to these as “Data Personas” (<https://jennyjmar.com/2016/04/15/data-personas/>)
- Personas and Data Personas are also useful for specification & testing!

# 7. Workflow steps

Given the customer has logged in  
And the customer dismissed the newsletter subscription  
And the customer started to create an order  
And the customer added the line to the order  
| product | quantity |  
| Flipchart | 13 |  
And the customer has placed the order

*removed non-relevant workflow steps*

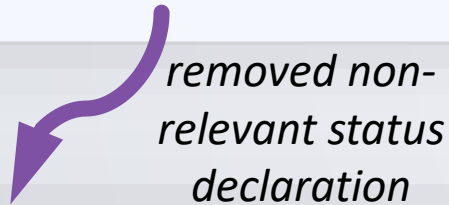


Given the customer has placed an order with  
| product | quantity |  
| Flipchart | 13 |

- The different user or workflow steps might also represent details that are not relevant
- Keeping these details in the scenario causes maintainability issues once the workflow changes
- It is better to state the intentions and the relevant steps only
- Push down:
  - Express intention revealing steps and ensure the state by repaying the necessary steps or by applying a shortcut in the automation code

# 6. System status

Given there is a customer  
And the customer is not blocked



Given there is a customer

- The details might be shown as declaration of the system or entity status
- Declaring “normal” status makes the scenarios harder to understand
- Push down:
  - Remove unnecessary status declarations from the scenario
  - Optionally add precondition checks to different actions to make sure that the system is in the expected state

Wrap-up

# We covered...

- That scenarios need details for execution, but you don't necessarily need to store those in the scenario itself, but push down to automation code
- That in order to use scenarios as specification, we need to make them focused and express only essential details in the steps
- That there are different detail types, watch for them and apply the mentioned “push-down” strategies if required
- That context is specified by the “Given” steps, but there are always implicit context too

Be aware of the details

Push them down

Make a team agreement on implicit details





*Thank you!*

Gáspár Nagy

coach • trainer • bdd addict • creator of specflow  
“The BDD Books” series • <http://bddbooks.com>  
@gasparnagy • [gaspar@specsolutions.eu](mailto:gaspar@specsolutions.eu)



PRESENTS

# Agile2023

THE SCOTLAND EXPERIENCE

**Join Agile Alliance today!**

Become an Agile Alliance member and help support our non-profit mission, while gaining access to valuable benefits like online events, in-person conference discounts, and event session videos.



#AGILE2023SCOTLAND