palinQA meetup

BUDAPEST

5 Tips to Plan Your Software Product Development with Agile Testing in Mind

István Novák

Agile coach, architect

Adaptive Consulting

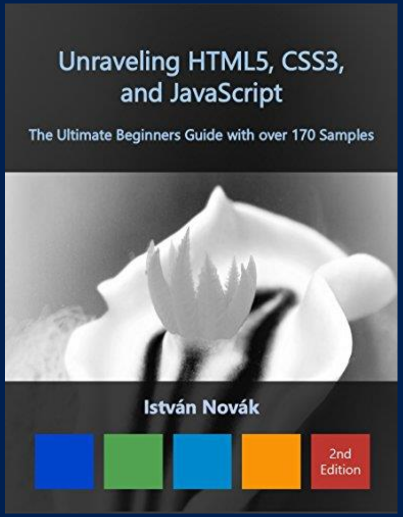
www.adaptiveconsulting.hu, istvannovak.net

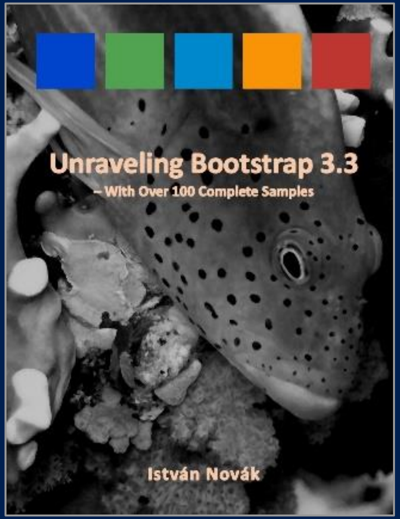
PalinQA meetup, February 28, 2017

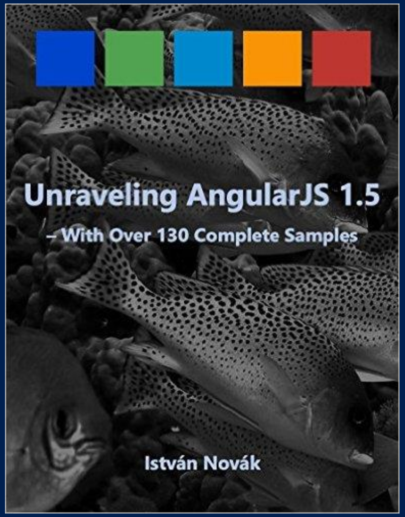
About István

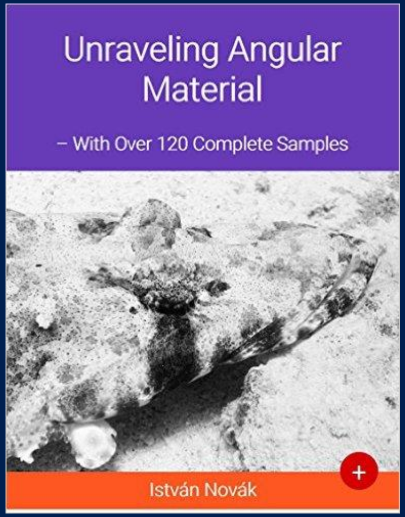
Agile architect and coach, technical book writer
Husband, father of two daughters (18 and 20)
Microsoft MVP, long distance runner, rabid fan of scuba diving

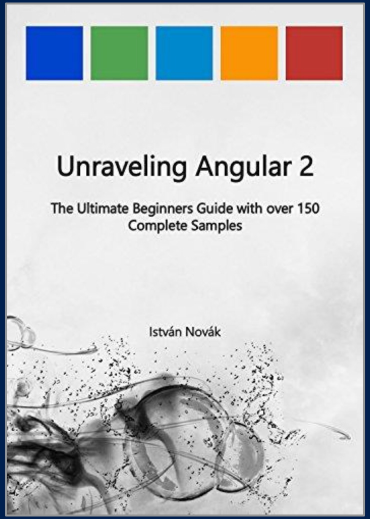
Author of **Unraveling Series** (Kindle Books)


Unraveling HTML5, CSS3, and JavaScript
The Ultimate Beginners Guide with over 170 Samples
István Novák
2nd Edition


Unraveling Bootstrap 3.3
– With Over 100 Complete Samples
István Novák


Unraveling AngularJS 1.5
– With Over 130 Complete Samples
István Novák


Unraveling Angular Material
– With Over 120 Complete Samples
István Novák


Unraveling Angular 2
The Ultimate Beginners Guide with over 150 Complete Samples
István Novák

The Fundamental Goal of Testing

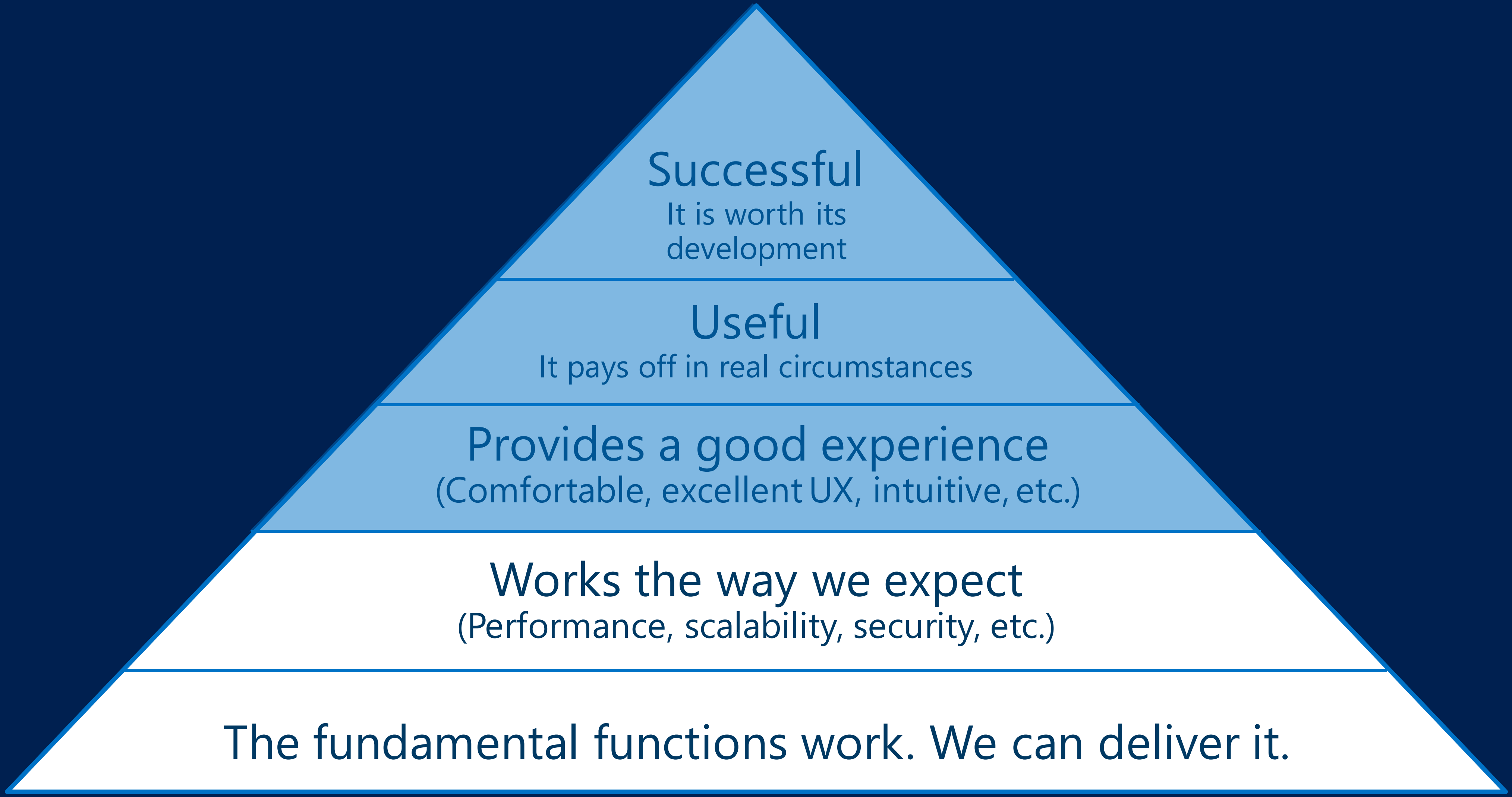
Approaches

We want to make sure of creating a software product that complies with its specification

How can we check that a particular specification leads to an excellent product?

We want to prove that the software product is appropiate for serving its original purpose; it solves the problem we aimed when creating it

The Levels of Testing


Successful
It is worth its development
Useful
It pays off in real circumstances
Provides a good experience
(Comfortable, excellent UX, intuitive, etc.)
Works the way we expect
(Performance, scalability, security, etc.)
The fundamental functions work. We can deliver it.

Modern, Agile Way of QA

QA is *not a separate* process element.

It is an *organic part* of the daily work.

Tip #1

Challenge Hypotheses

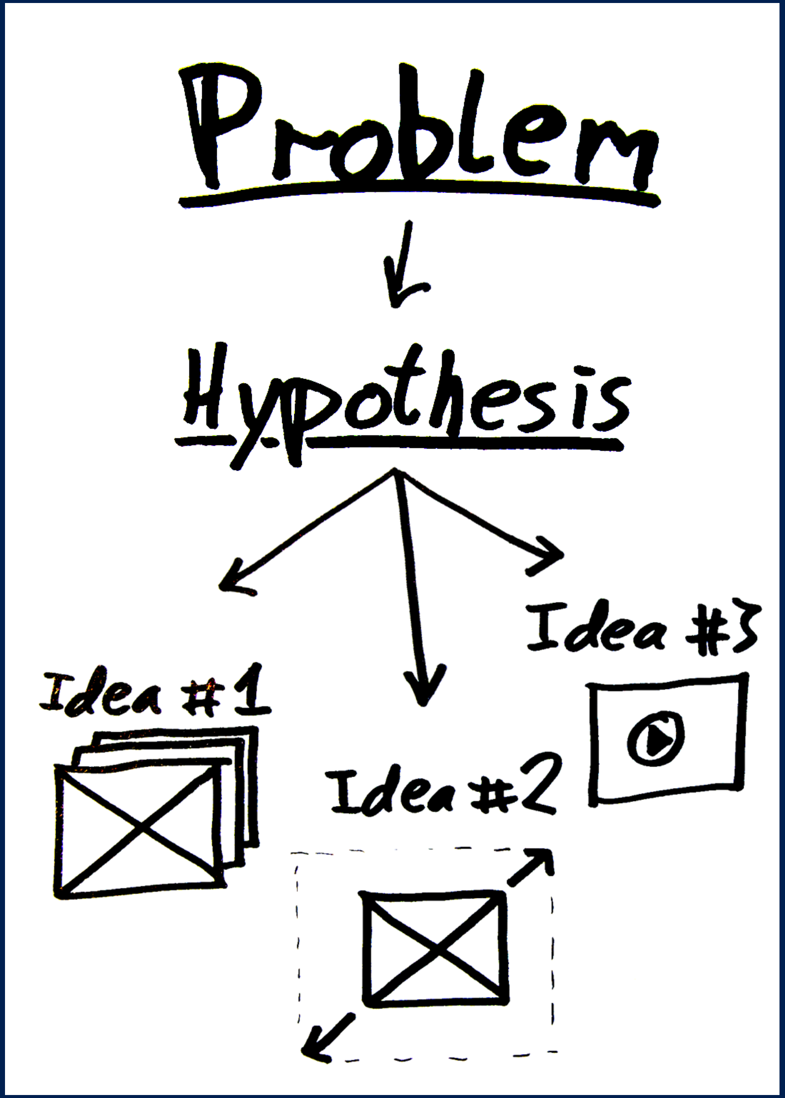
Important Questions

What problem does the product solve?

Would it be successful provided it solves the problem?

Does it truly address the problem?

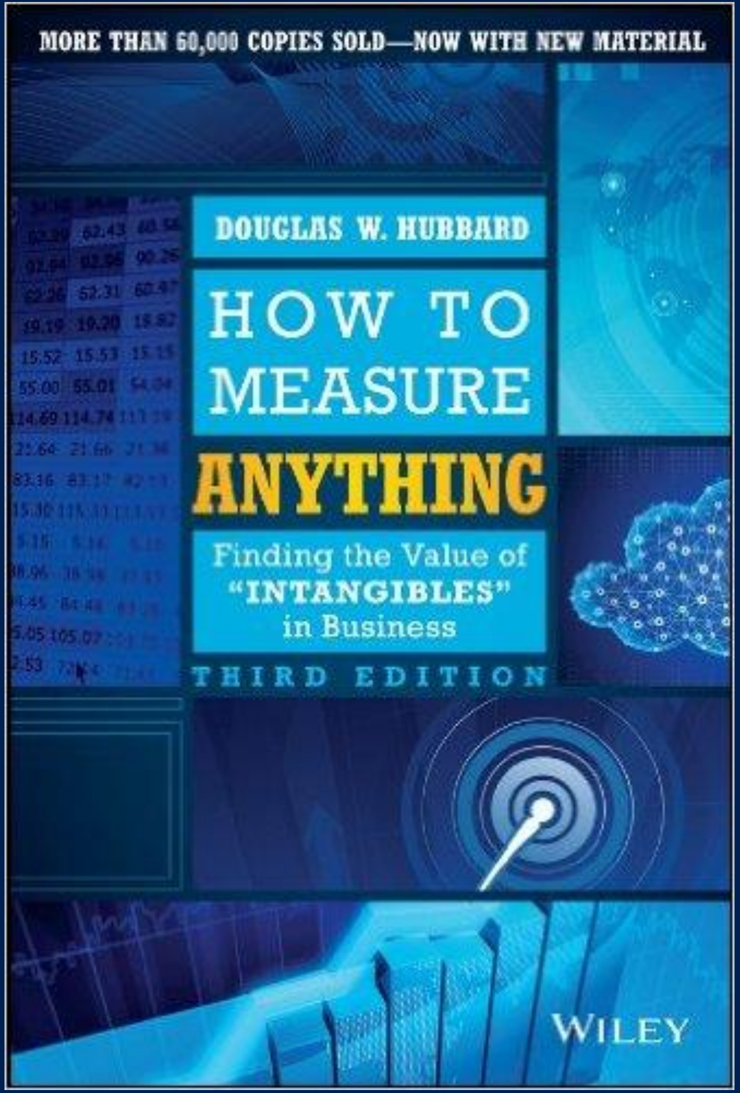
How can we check that the answers to these questions – the hypotheses – are valid?



Hypothesis Evaluation

Unless a hypothesis can be proved *to be valid*, the feature might not be one we should develop

Book Tip


MORE THAN 60,000 COPIES SOLD—NOW WITH NEW MATERIAL
DOUGLAS W. HUBBARD
HOW TO MEASURE ANYTHING
Finding the Value of "INTANGIBLES" in Business
THIRD EDITION
WILEY

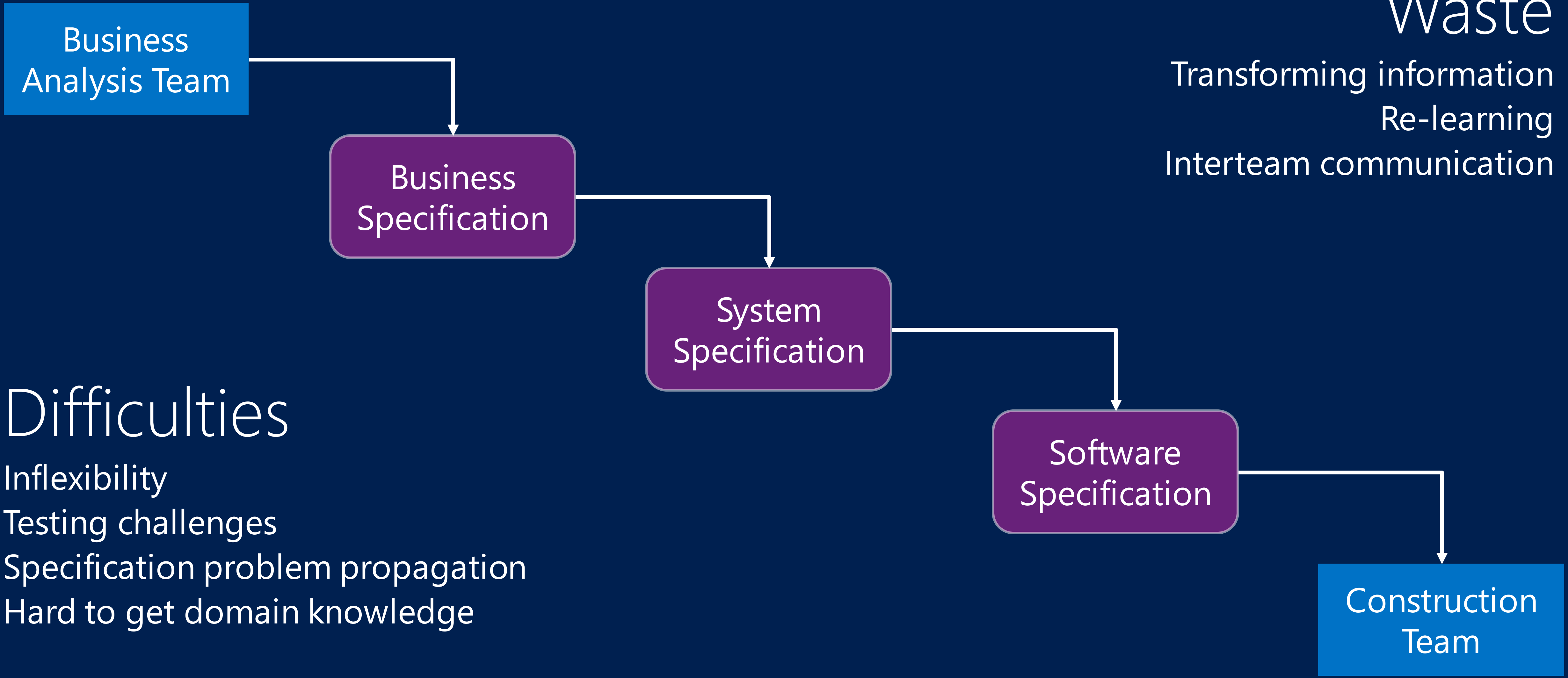
How To Measure Anything

by Douglas W. Hubbard

Tip #2

Do *Not* Develop Directly from Traditional Specification Documents

Traditional Way of Using a Specification



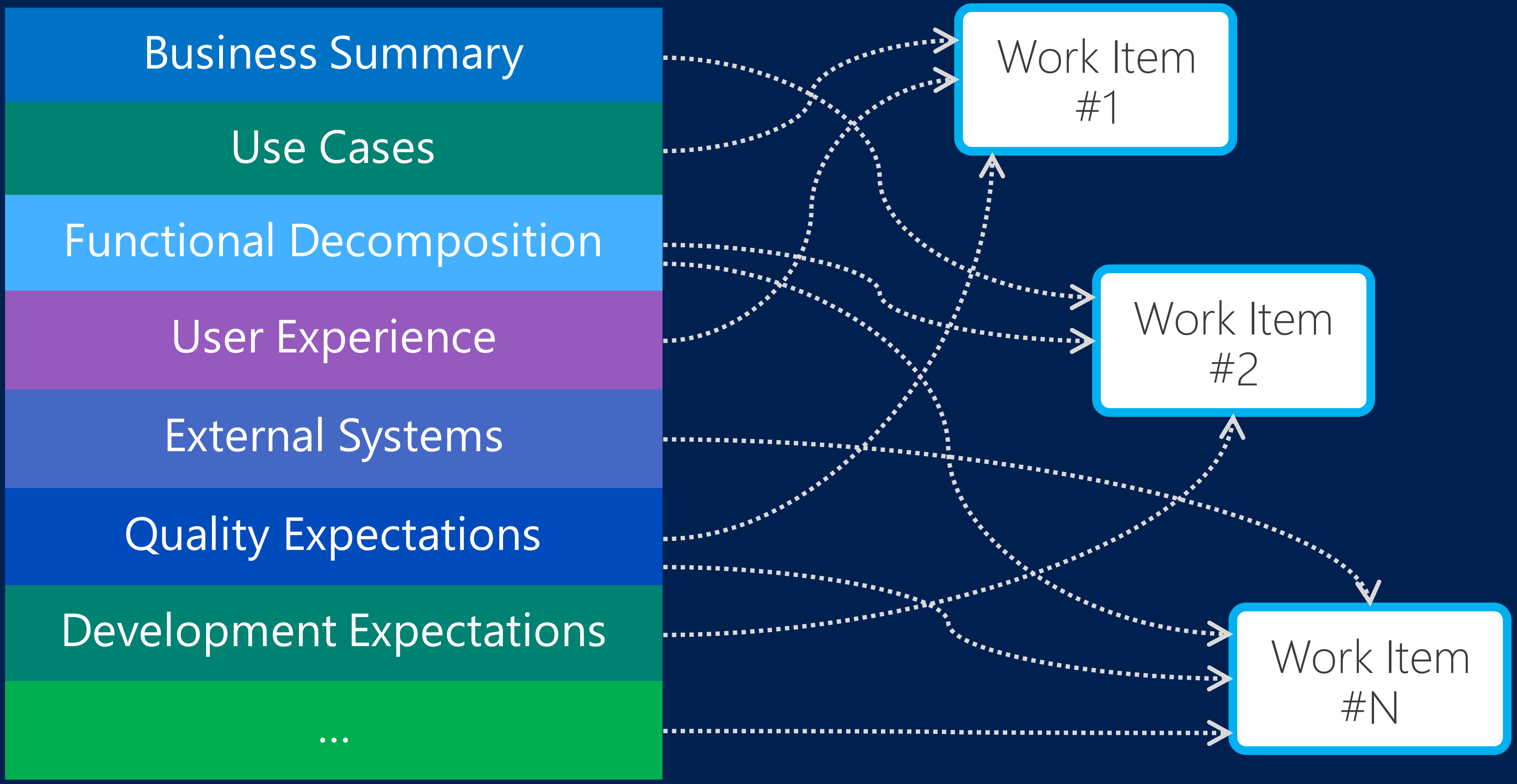
Waste

Transforming information
Re-learning
Interteam communication

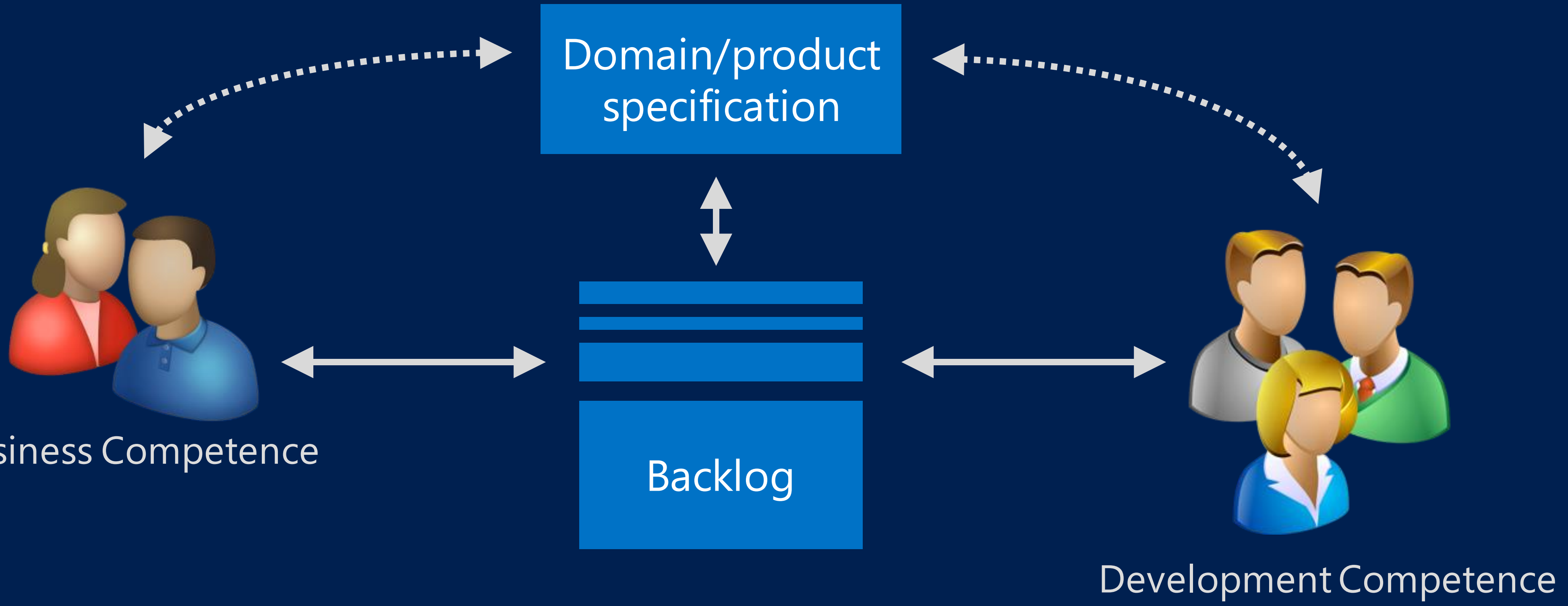
Difficulties

Inflexibility
Testing challenges
Specification problem propagation
Hard to get domain knowledge

Searching for Work Items



Develop from Backlog



Got It?

Specification is not evil

You'd better not work without specification

Developing from traditional specification documents is challenging

Tip #3

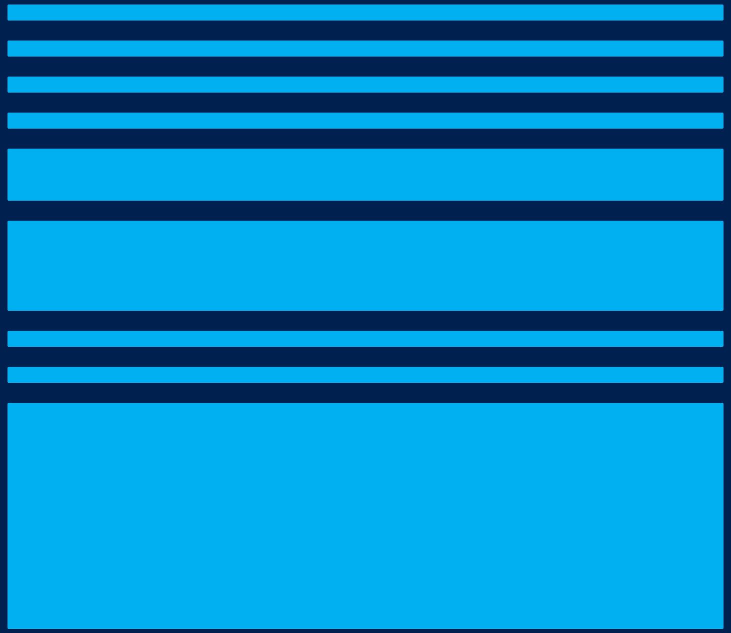
Build Your Stories Around Testing and Demo

Quick Recap – Product Backlog

Ordered list of functions and product features we intend to work out in order to implement the product.

„User voice" form:

As a [**stakeholder**] I'd like
to use a [**feature**]
so that I can achieve a [**goal**]



Smart User Story Structure

Definition: <who?>, <what?>, <for what purpose?>

Acceptance Criteria

Test/Demo Scenarios

Ensure the creativity of the development team

Define when do we take the story into account as completed

We prove that the completed story satisfies the acceptance criteria

Smart User Story Structure

Definition: <who?>, <what?>, <for what purpose?>

Acceptance Criteria

Test/Demo Scenarios

Test and Demo Scenarios

Avoid detailed steps and be concrete

The e-shop user wants to search in accent-insensitive way to get a better-matching result set

Demo scenario:

#1: The user searches for „sor” → 14 match is returned

#2: The user searches for „sör” → 14 match is returned, the same as in step #1

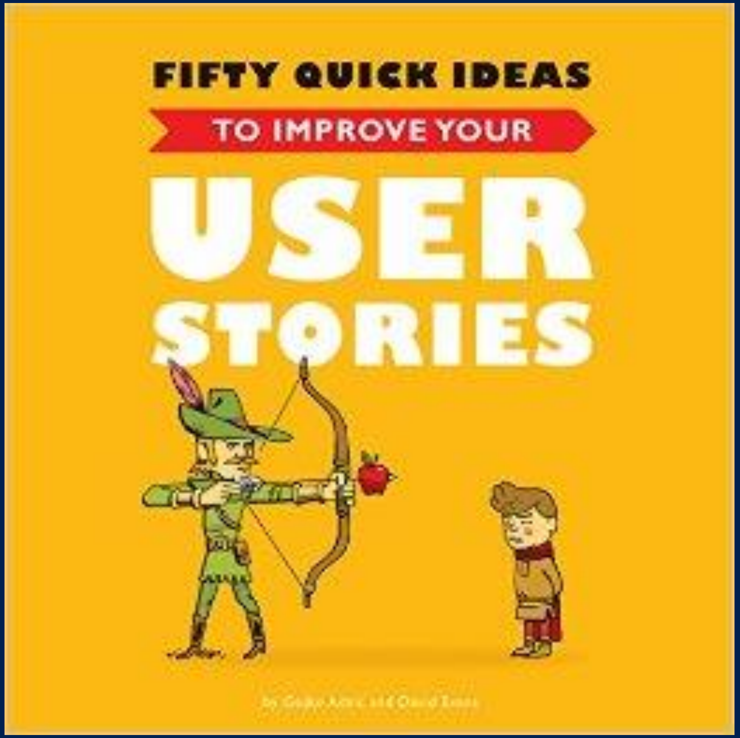
„Big Chunk" Warning Signs

Too many scenarios

Too many steps in the demo scenario

Going through the demo takes long time

Book Tip



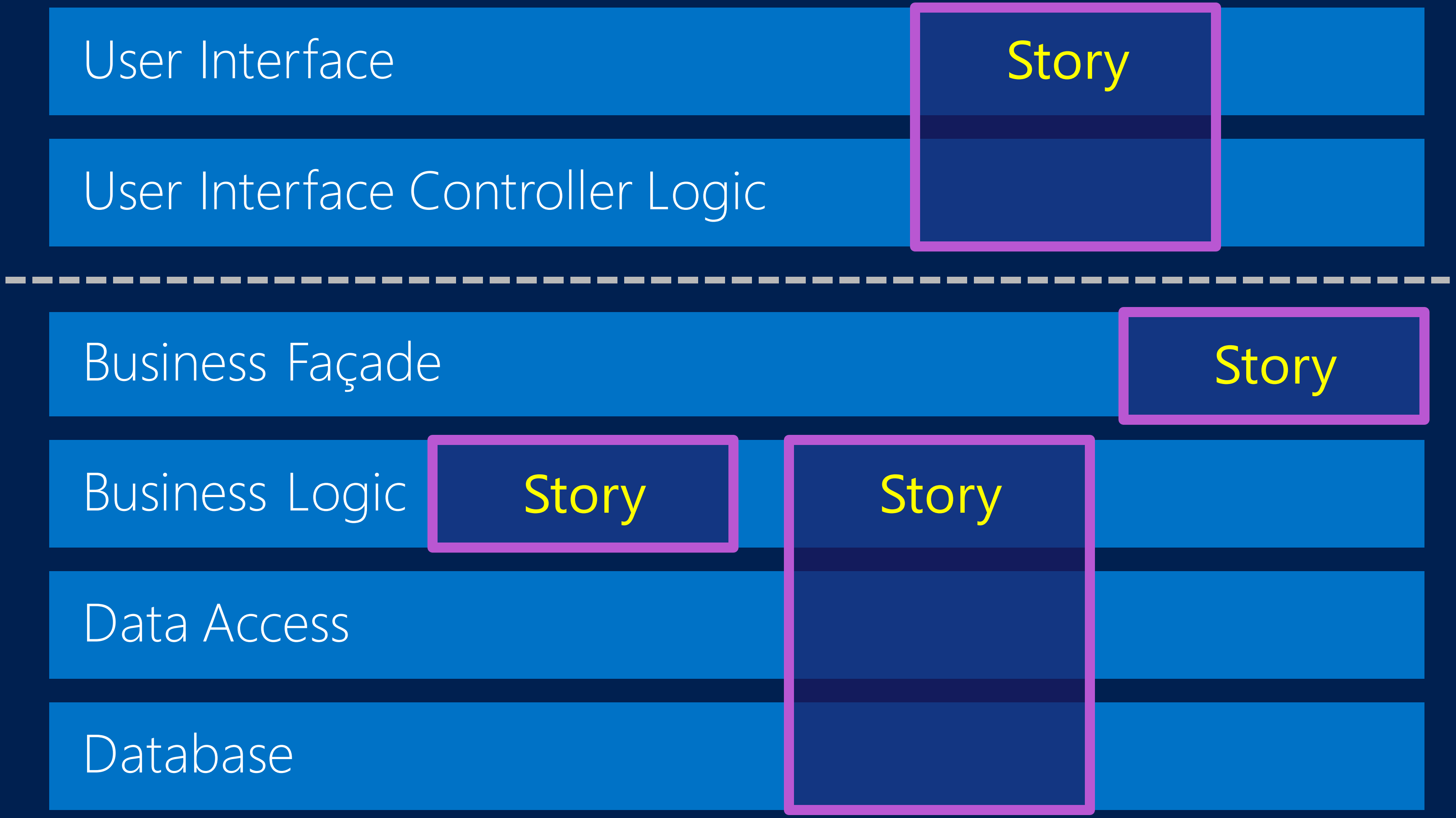
Fifty Quick Ideas to Improve User Stories

by Gojko Adzic, David Evans

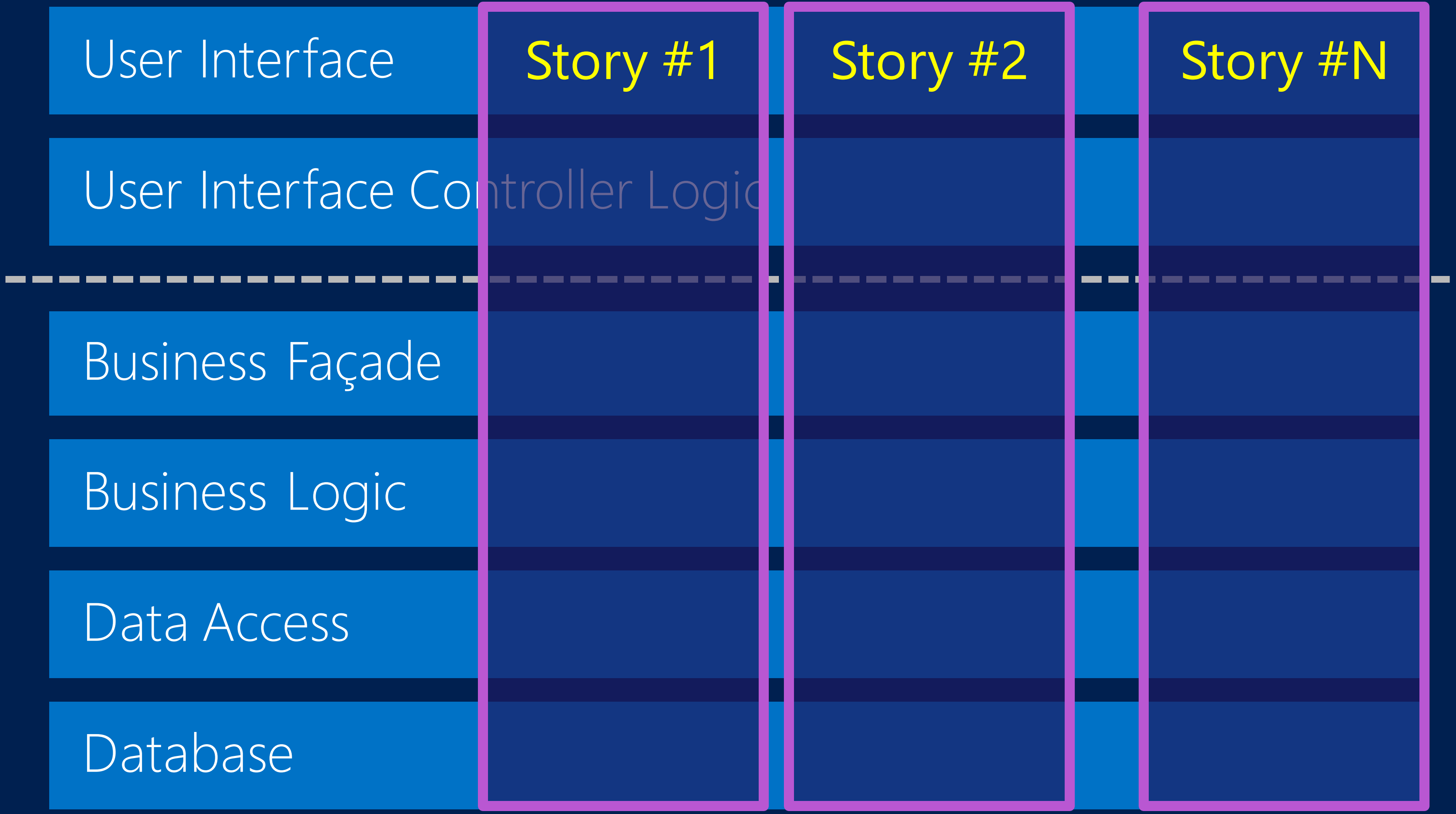
Tip #4

Think Vertically

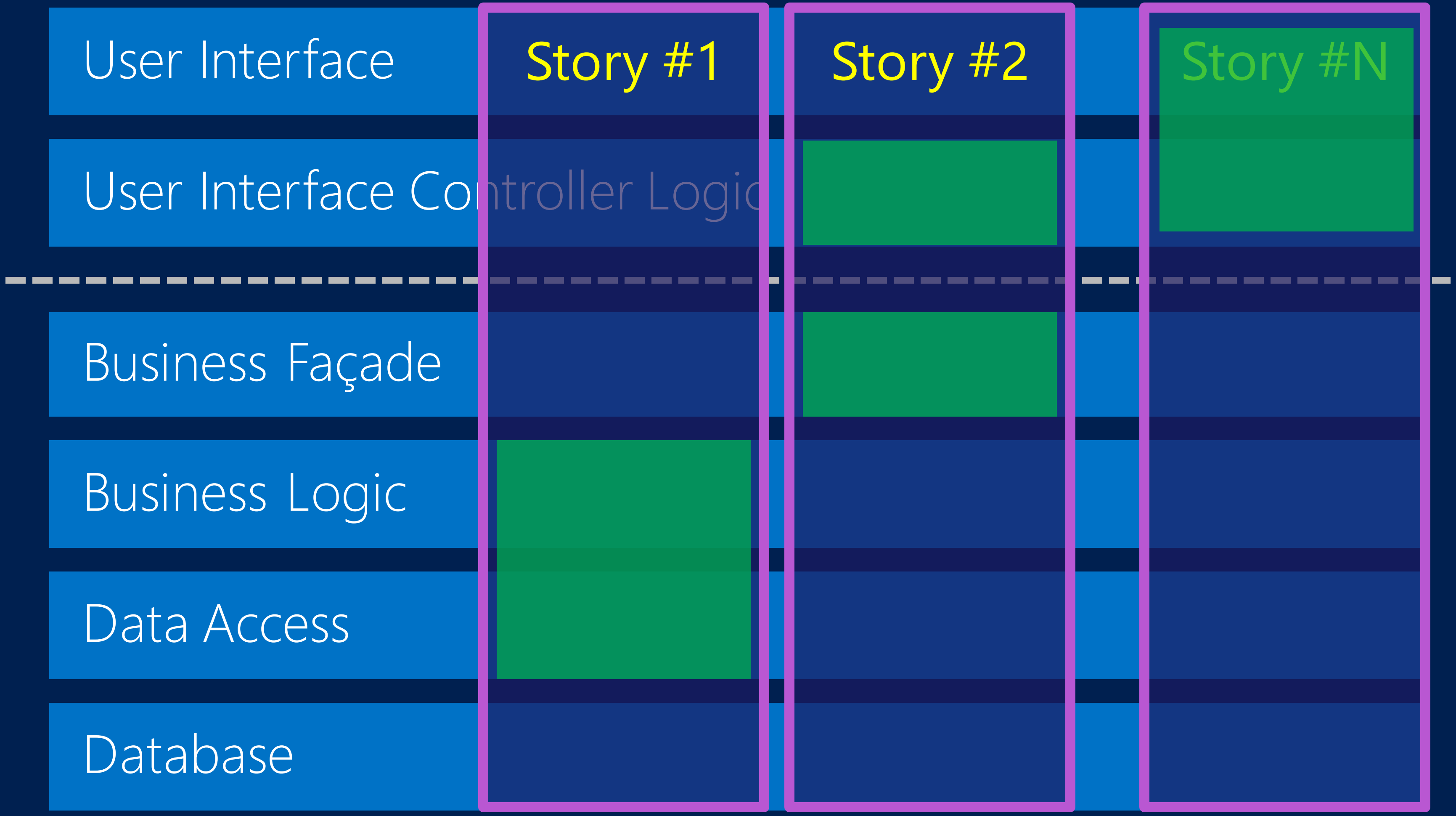
Avoid Horizontal User Stories



Lead Stories Through All Layers

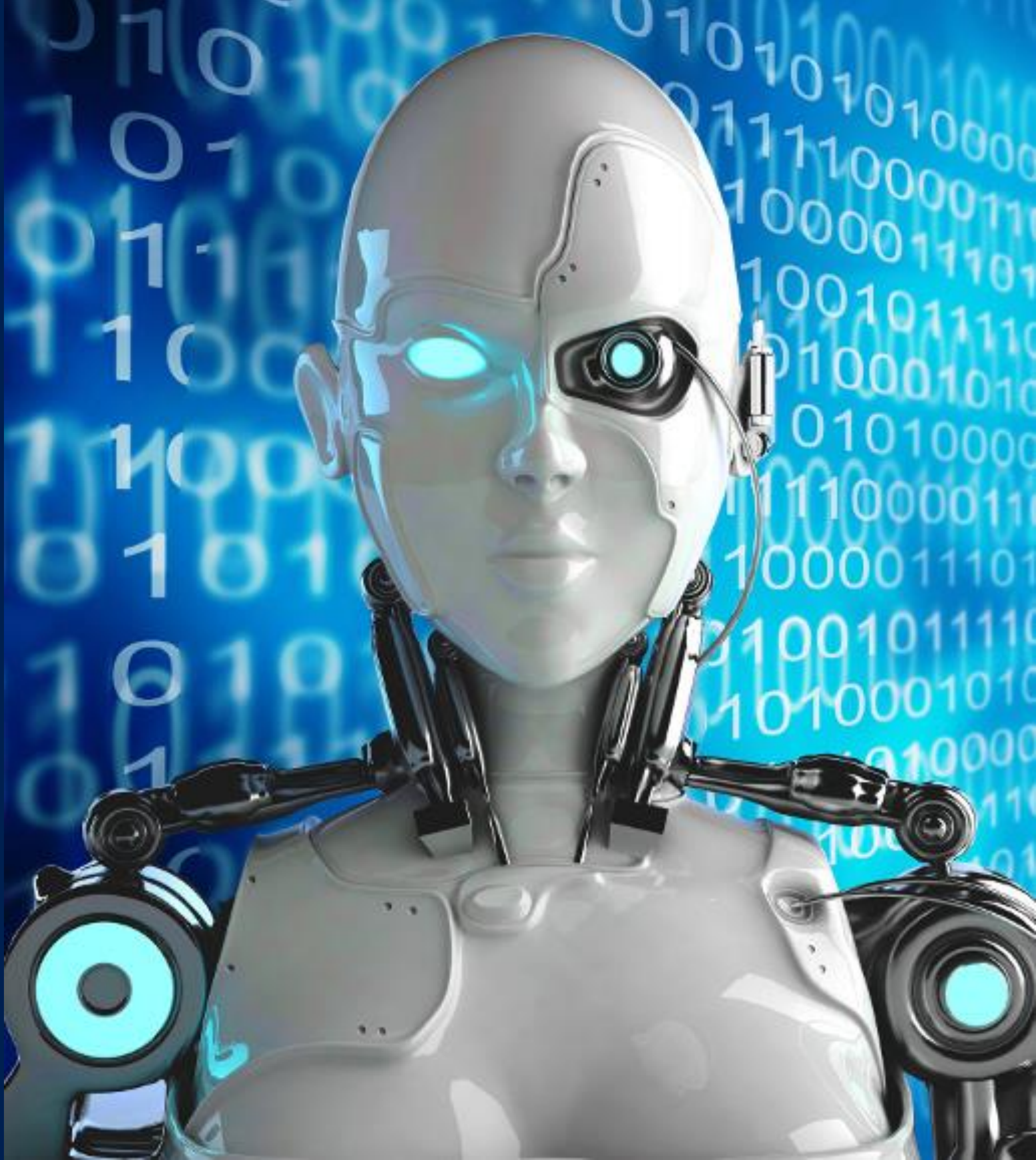


Implement The Necessary Layers



Tip #5

Create Your Test Automation Wisely



Why I Love Automatic Tests

Automatic tests are repeatable – practically with any frequency

Coding task complete → corresponding automatic tests are implemented and run

They make regression testing easy

The best way to shake up a team when fighting with technical debt

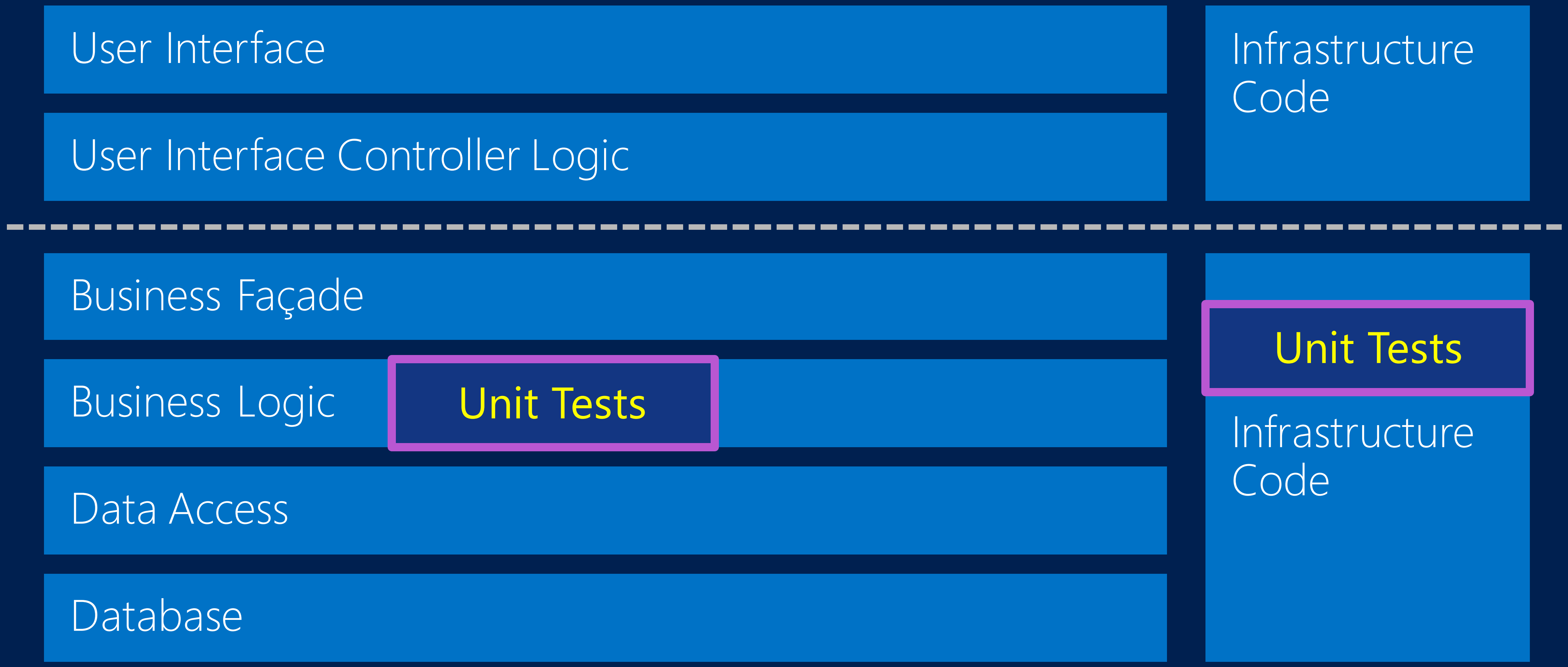
Refactoring the code to testable makes its structure more robust

Well-structured unit tests are API documentations, too

(Automatic Testing ≠ Unit Testing)

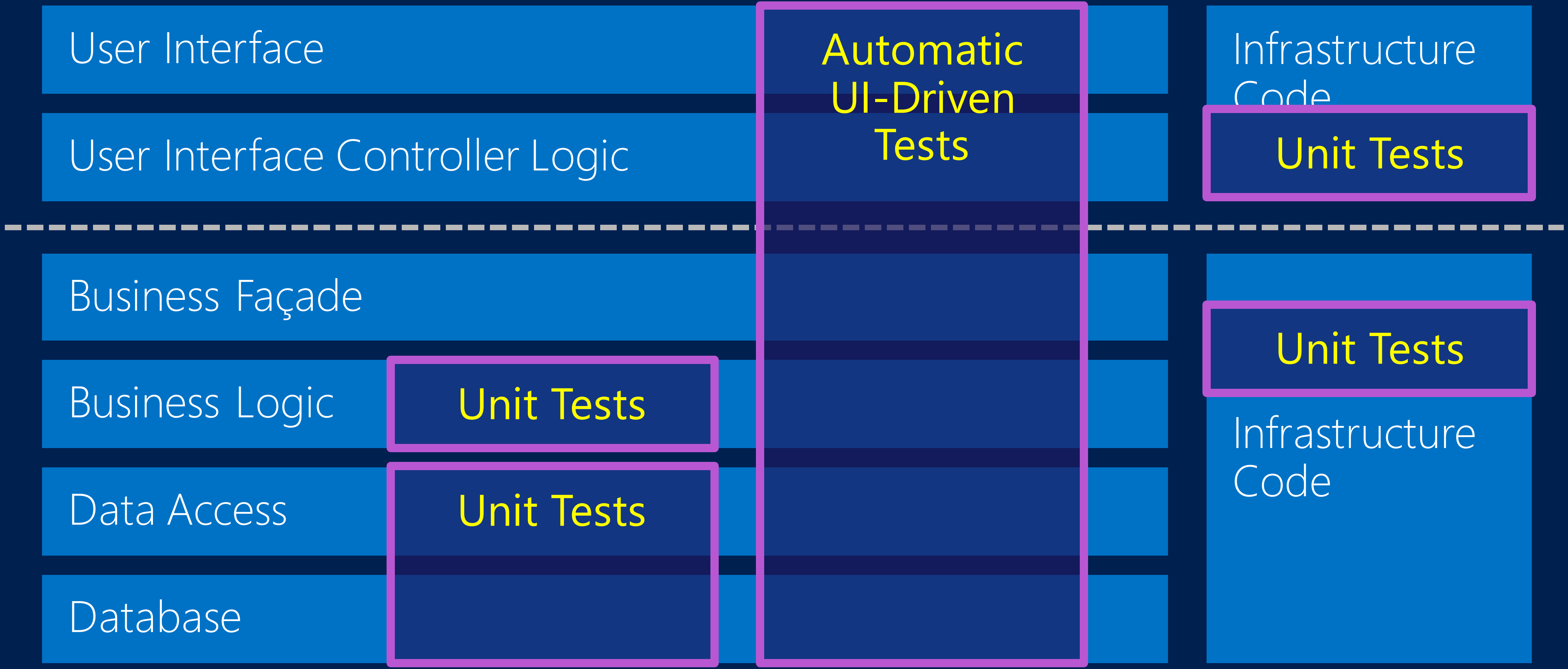
What to Cover with Automatic Tests

Under-Represented Testing



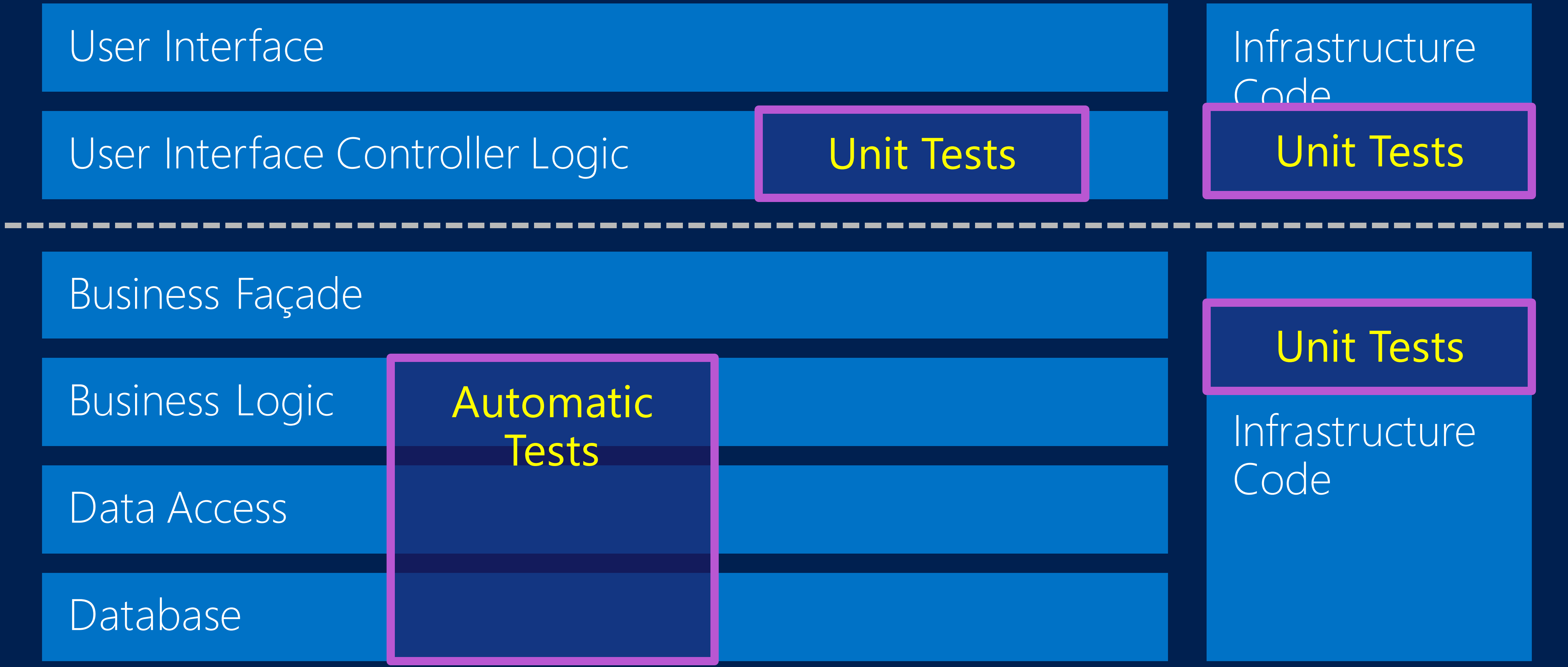
What to Cover with Automatic Tests

Over-Enforced Testing



What to Cover with Automatic Tests

A Good Starting Point…



Takeaways

#1: Challenge Hypotheses

#2: Do Not Develop Directly from Traditional Specification Documents

#3: Build Your Stories Around Testing and Demo

#4: Think Vertically

#5: Create Your Test Automation Wisely

Questions?

Novák István

dotneteer@hotmail.com