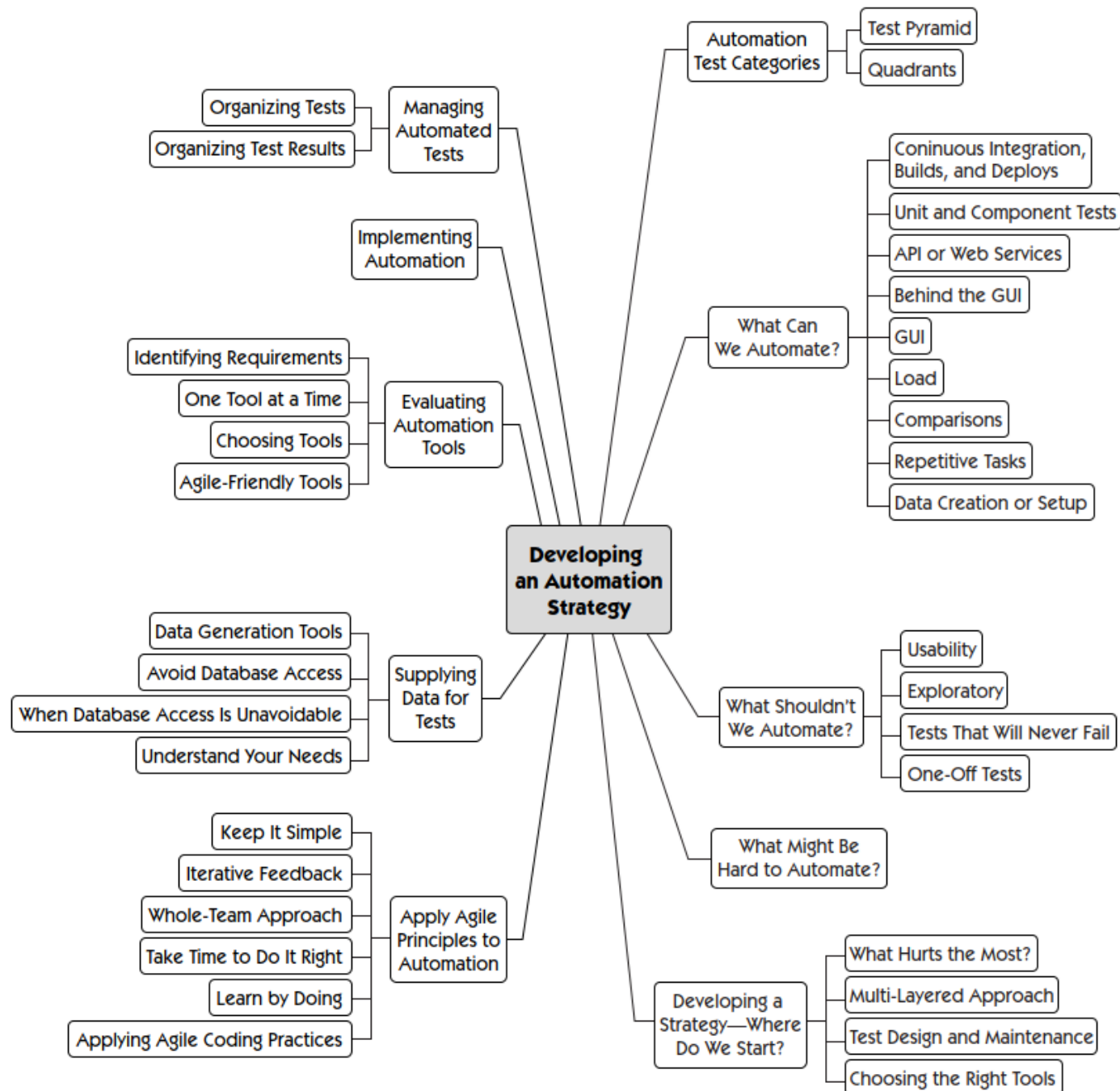


# Test Automation


## Concepts and Strategy

*By Samer Zein, PhD*

# **Part 1: Why Automate? And What are the Common Obstacles?**



# The Focus

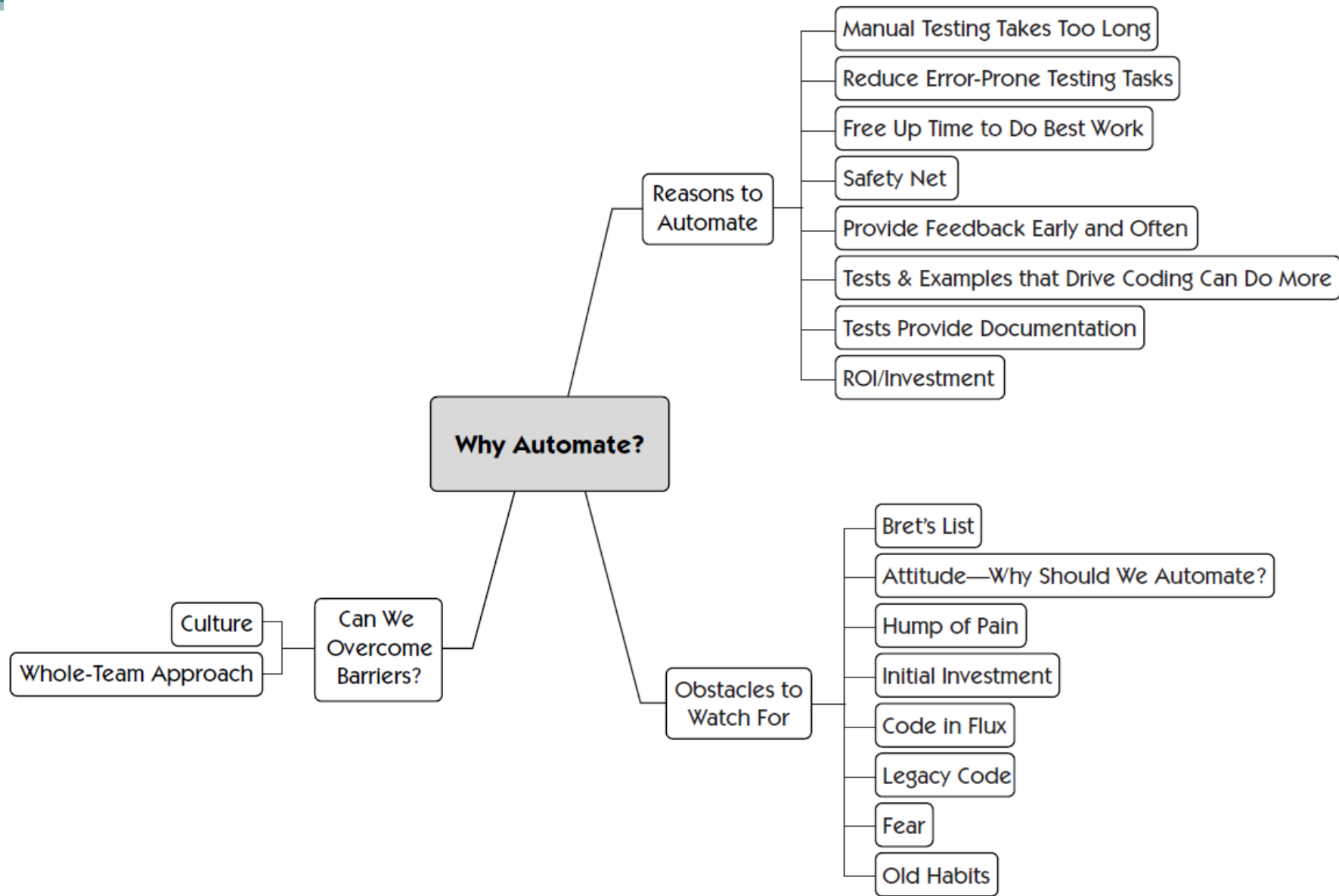
- Automation is a vast topic, it includes:
  - ▣ **Automated builds,**
  - ▣ **Automated deployment**
  - ▣ **And Automated Testing** 
    - **Why we need automation?**
    - **What are the common obstacles to achieve it?**

# Fact

- Automation is a vast topic that requires a course on itself!
- It includes tasks such as writing simple shell scripts, session management, and creating robust test automation suites.
- The range of automation tools rises exponentially!

# Test Automation is a Core Agile Practice

- Agile projects depend on **automation**.
- Good-enough automation frees the team to deliver high-quality code **frequently**.
- It provides a framework that lets the team maximize its **velocity** while maintaining a **high standard**.



# Why Automate?

## Manual Testing Takes Too Long:

- It simply takes **too long** to complete all of the necessary testing **manually**.
- The time to test everything grows longer and longer, sometimes **exponentially**!
- Remember the black-box testing example, you will need to repeat it daily if it is manual!
- Need to have testing everyday.
- Impossible to do **manual** regression testing daily.
- In order to keep pace with development.



# Why Automate?

## Manual Processes Are Error Prone:

- Manual testing gets **repetitive**, especially if you're following scripted tests.
- Manual tests get **boring** very quickly.
- It's way too easy to **make mistakes** and overlook even simple **bugs**.
- Get short in time ☐      cut down testing time ☐  
miss lots of bugs.

# Why Automate?

## Automation Frees People to Do Their Best Work:

- The Dream of a Tester: **Having continual builds run all of the unit tests and the functional regression tests**
- This will free the testers to do their best work: **Exploratory Testing!**
- Having not only **more time**, but also more **mental energy!**

# Why Automate?

## Automated Regression Tests Provide a Safety Net:

- Most practitioners who have been in the software business for some years know the **risk** of modifying code that is not covered by automation tests.
- Knowing the code has sufficient coverage by automated regression tests gives a great feeling of **confidence**.
- Using regression tests we will find errors **immediately** thus fixing them in a **short time**.

# Why Automate?

## Tests and Examples that Drive Coding Can Do More

- Committing to automated testing can include
  - TDD
  - And SDD (Story-test Driven Development)
- Produces high quality code
- Deep compliance with user requirements.
- Better code design.

# Why Automate?

## Tests are great documentation

- Automated tests have to be **constantly updated**.
- Thus they become accurate **live documentation** of the system.
- On the other hand, it is very hard to keep updating **static documentation** (technical document reproduced every sprint)

# Why Automate?

## Great ROI:

- All of the reasons just presented contribute to the bottom line and the payback of automation:
  - **Free time to test the application to the limits**
  - **Better time to market.**
  - **Defects are fixed in short time as opposed to manual testing.**

# Things that get in the way of automation:

## Bret's List:

- Only using the spare time to do automation will not get you much far.
- Lack of clear goals.
- Lack of experience,
- Huge turnover
- Reaction to desperation
- Having it as a way of fun or reputation.

# Things that get in the way of automation:

- Additional List:
  - **Programmers' attitude.**
  - **The learning curve.**
  - **Initial investment**
  - **Code that is always in Flux.**

Many people have experienced test automation efforts that didn't pay off. Their organization may have purchased a vendor capture-playback tool, given it to the QA team, and expected it to solve all of the automation problems. Such tools often sit on a shelf gathering dust. There may have been thousands of lines of GUI test scripts generated, with no one left who knows what they do, or the test scripts that are impossible to maintain are no longer useful.



# Automation Tools and Incorrect Design of Test Scripts Real Case

I walked into an organization as a new QA manager. One of my tasks was to evaluate the current automated test scripts and increase the test coverage. A vendor tool had been purchased a few years earlier, and the testers who had developed the initial suite were no longer with the organization. One of the new testers hired was trying to learn the tool and was adding tests to the suite.

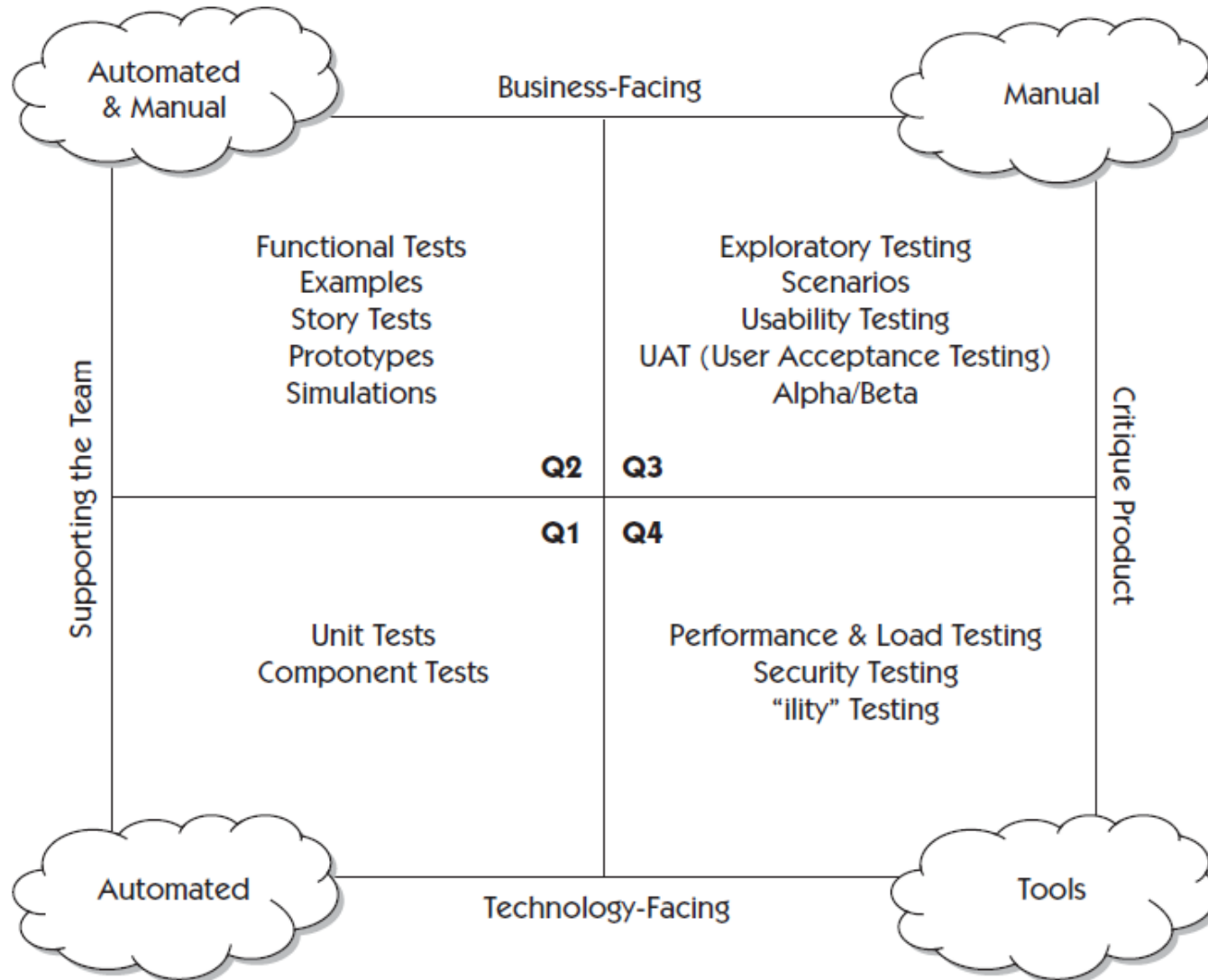
The first thing I did was ask this tester to do an assessment on the test suite to see what the coverage actually was. She spent a week just trying to understand how the tests were organized. I started poking around as well and found that the existing tests were very poorly designed and had very little value.

We stopped adding more tests and instead spent a little bit of time understanding what the goal was for our test automation. As it turned out, the vendor tool could not do what we really needed it to do, so we cancelled the licenses and found an open source tool that met our needs.

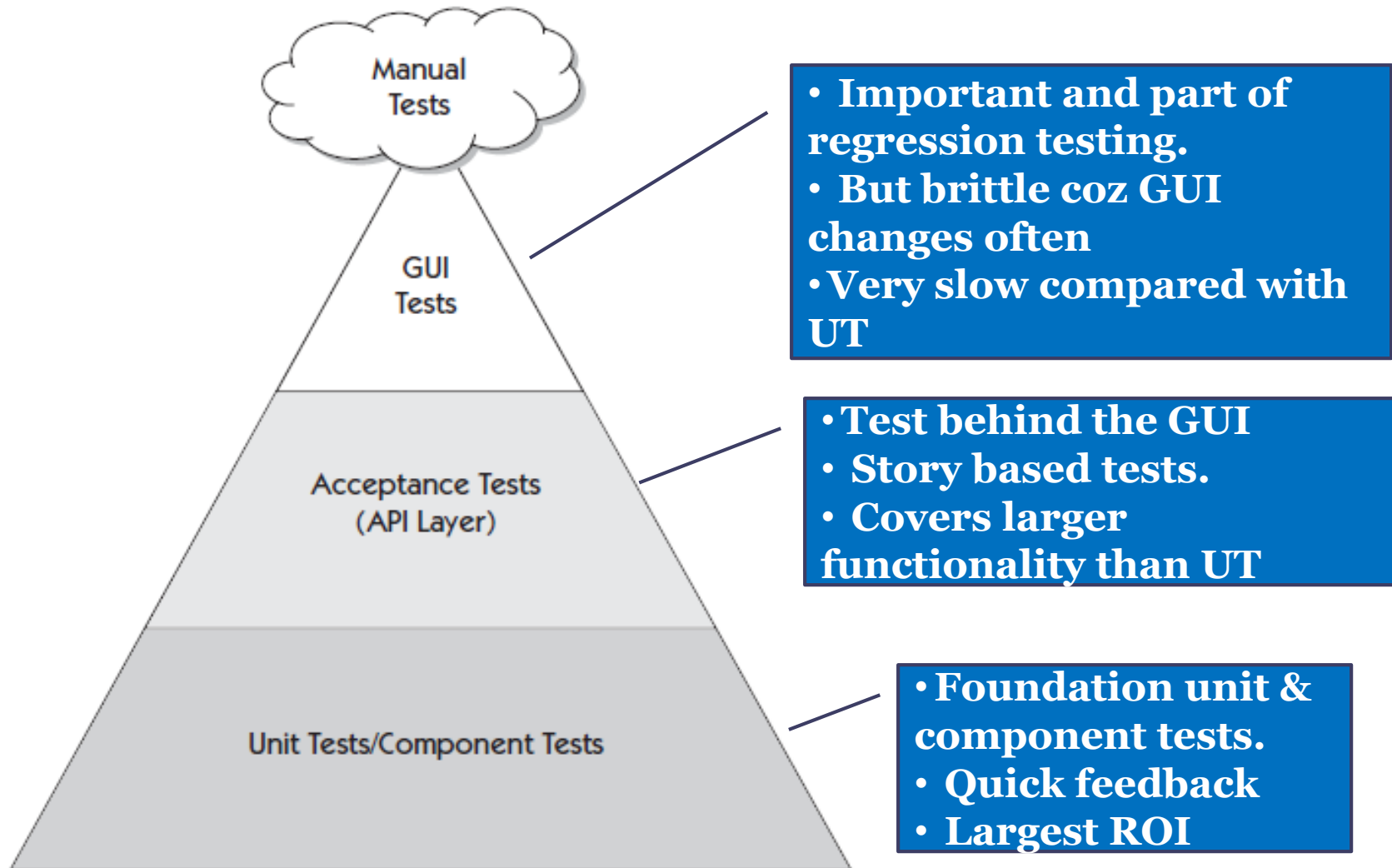
We still had to spend time learning the new open source tool, but that investment would have been made if we'd stayed with the original vendor tool anyhow, because no one on the team knew how to use the original tool.

# Part 2: An Automation Test Strategy

# Agile Testing Quadrants



# Agile Test Automation Pyramid

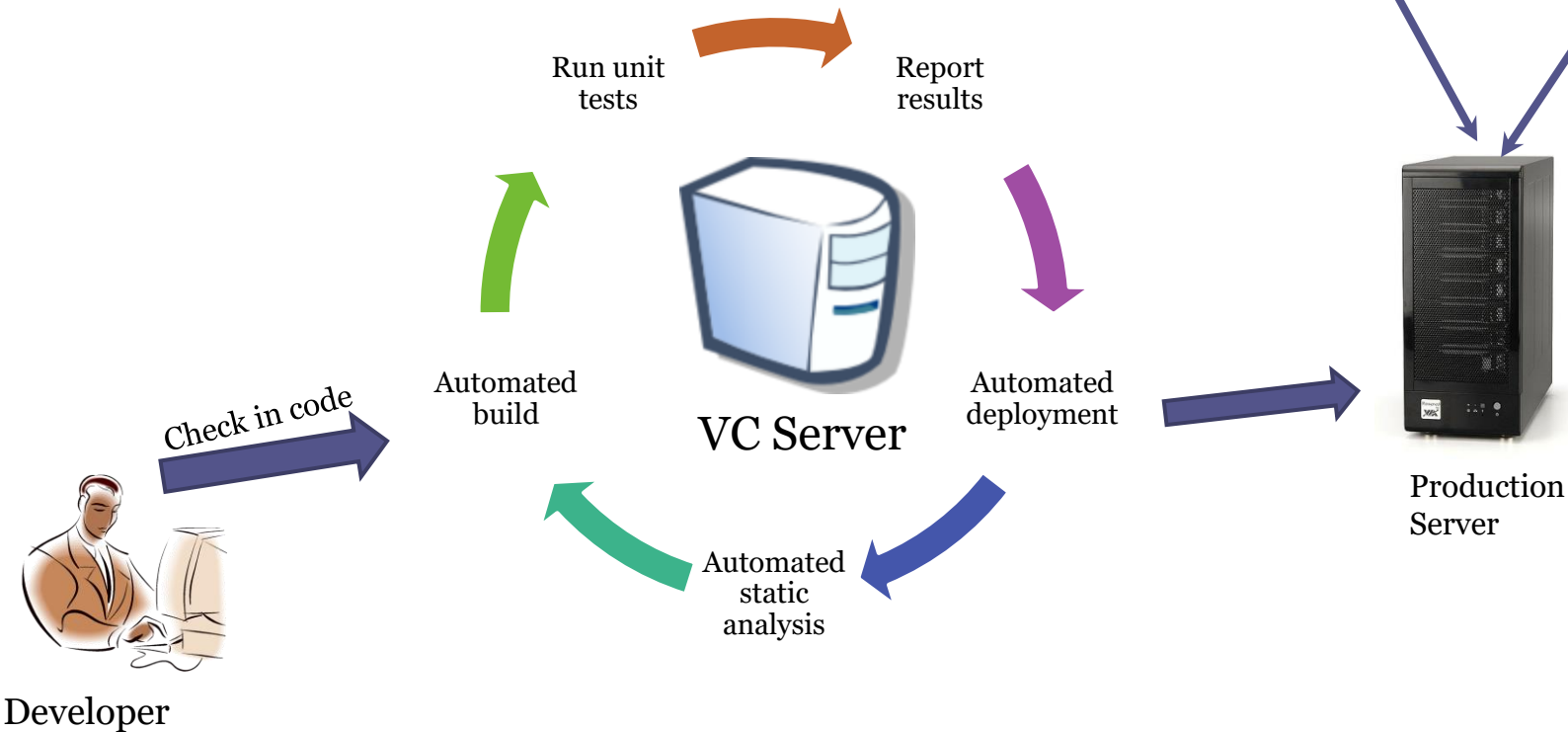


# Important Notes

- In Agile development methods, the tester can **pair with developers** to build the tests.
- Manual exploratory testing is an **effective** way to find functional defects, but if we don't have **enough** automated regression tests, we probably spend **all of our time** madly trying to keep up with manual regression testing.

# Automated Build, Integration and Deployment

- More unit tests
- integration tests
- automated UI
- performance, load
- Exploratory tests



# What Else Can be Automated?

- API or Web Service Testing
  - Special tools to read test data from spread sheets and run them against web services (Ruby)
- Test Behind the GUI
  - Fit and Fitness tools
- Testing the GUI
- Load Tests
- Repetitive Tasks
- Data Creation and Setup

# What Should not be Automated?

- Usability Testing
  - **You will need real person do test the usability of the system.**
- Exploratory Testing
  - **Learn more about the system, push the system to the limits, try different pattern of scenarios**
- Tests that will never Fail



# Important Consideration for Test Automation

- Need to have clear objectives for automation.
- Choosing the right tools.
- Test Design and Maintenance
- Whole team approach
- Avoid Database Access