Patterns in Test Automation: Issues and Solutions

Prepared and presented by

Seretta Gamba

seretta.gamba@steria-mummert-iss.de
srttgmb@yahoo.com

Dorothy Graham

@DorothyGraham
info@dorothygraham.co.uk
www.DorothyGraham.co.uk

Tutorial objectives

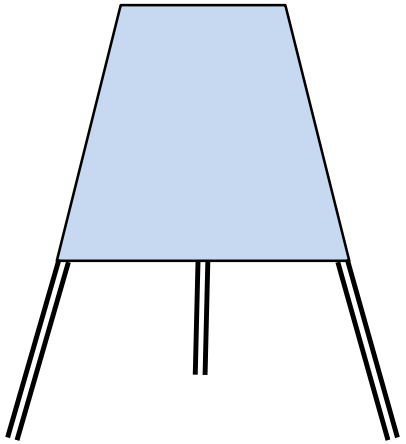
- help you recognize test automation issues
- explore ways of solving automation issues using patterns
- work on example scenarios*

- NOTE: system level automation, not unit level
- NOTE: test automation patterns, not test patterns

*using Test Automation Patterns wiki **on your laptop or internet-enabled device**

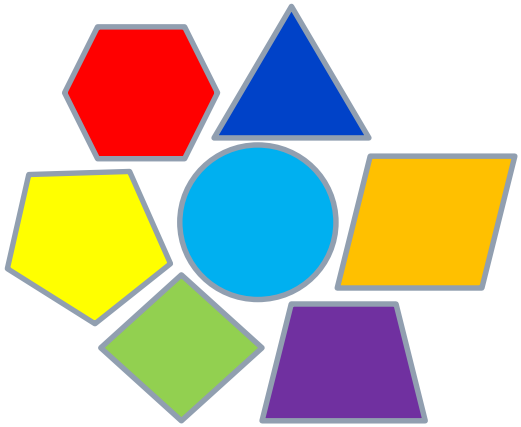
Issues (problems)

- what problems affect you in test automation?
  - what causes the most trouble?
  - what takes longer than it should?
  - what is holding you back in your automation?
  - what are you struggling with?
  - what would you most like to improve?



Agenda

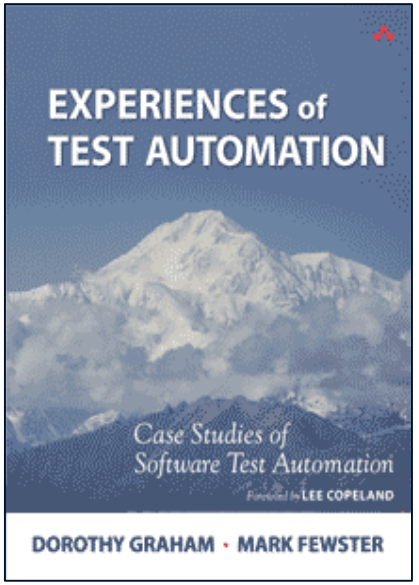
- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion



Origin of patterns in software

- 1970s: architect Christopher Alexander describes patterns for buildings (e.g. light from 2 walls)
- 1987: Kent Beck & Ward Cunningham experimented & presented at OOPSLA
- 1994: “Gang of Four” wrote a book applying pattern principles to software design
  - “Design Patterns: elements of reusable object-oriented software”, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- now widely accepted in software development community

Origin of Test Automation Patterns



- Experiences book published 2012
- Seretta contributed Ch 21
  - Automation through the back door, by supporting manual testing
- she is a developer, used to using patterns
- when she read the other chapters, she thought: “Patterns! Patterns!”
- she wrote up test automation patterns as a book & asked Dot to join her
- Word didn’t work very well, so we made a wiki

Test automation patterns wiki

- purpose: share information, ideas and experiences about test automation, using issues and patterns
- mind-maps summarize issues and patterns (not clickable as links)
- issues and patterns are classified into sub-sections (e.g. management, process, etc.)

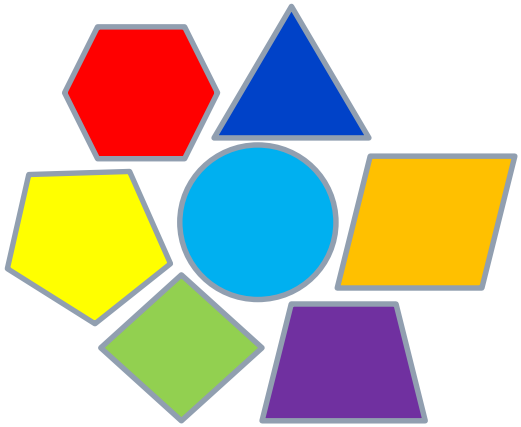
- wiki started 28 January 2013

About the patterns in the wiki

- the wiki has a limited set of patterns
- patterns use other patterns
- we are focusing on system-level automation rather than unit test automation
  - See Gerard Meszaros' book "xUnit Test Patterns: Refactoring test code" for unit test patterns
- a "work in progress"
- a new way of presenting known information

Agenda

- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion



Test Automation Issues

- An *Issue* is a problem that testers and/or test automators encounter when trying to do test automation

- They are things that take longer than they should, that hold you back in what you want to automate, or that are just too much trouble

Test Automation Issues (~75)

- **Process Issues**: problems that occur when the test automation process has not yet reached the necessary maturity
- **Management Issues**: problems that occur when management has not given the necessary support (budget, resources, team)
- **Design Issues:** problems that occur when an efficient testware architecture and maintainability are not built in from the very beginning
- **Execution Issues:** problems that occur when the automated tests are run (unpredictable results etc)

Test Automation Patterns

- General reusable solution to a commonly occurring problem within a given context
- a way of solving an issue or problem in test automation that has worked in practice for many people
- a pattern is not prescriptive or a set of steps, but rather ideas to consider to see if they would apply in your situation or context

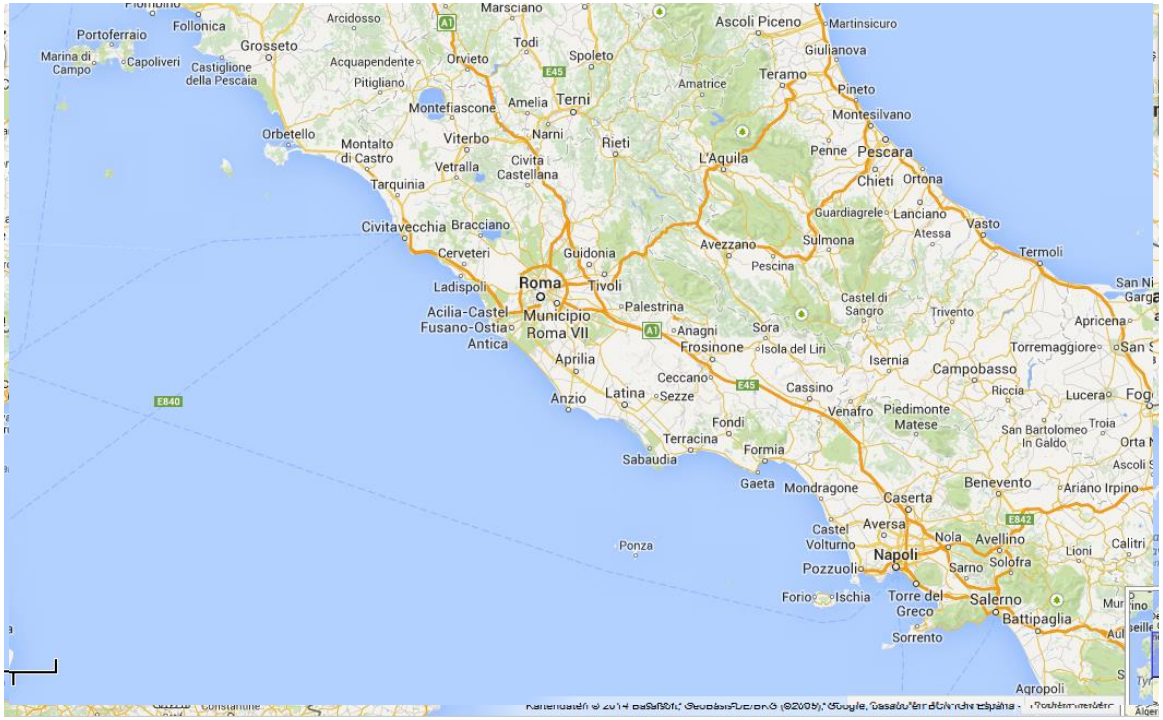
Pattern ➔ expert knowledge proven by repeated experience

Test Automation Patterns (~75)

- **Process Patterns**: how the test automation process should be set up or how it can be improved
- **Management Patterns**: how to manage test automation both as an autonomous project or integrated in the development process
- **Design Patterns:** how to design the test automation testware so that it will be efficient and easy to maintain
- **Execution Patterns:** how to take care that test execution is easy and reliable

General Issues

- General issues are the issues that one feels when test automation is not going as expected, but one cannot really pinpoint a specific problem, because usually more than one is causing the pain
- General issues contain other more specific issues



How to use general issues

1. Select the general issue that best describes the most pressing problem in your test automation.
2. Examine what kind of issues can cause it (probably more than one).
3. Select the one that you deem most pressing.
4. Repeat steps 2. and 3. until you reach the specific issue that needs to be solved first. It will suggest the patterns needed to solve it.
5. Select the pattern(s) that best fit your context and apply it/them.
6. If you still have problems, start again from step 1.

Why use general issues

- You don‘t have to study all the issues and patterns (50+70)
- The issues lead you to the causes of your problems
- You can focus on the most effective patterns
- You can repeat the process as often as you want

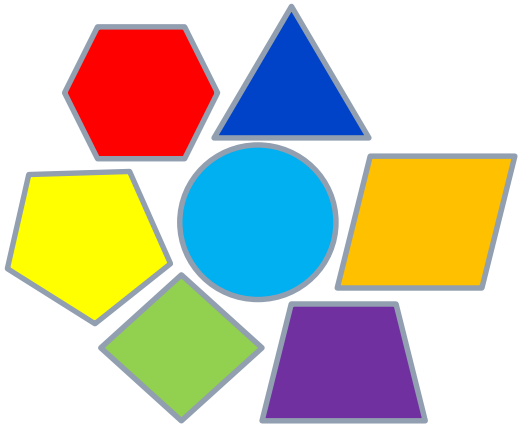
General Issues: Example

- One or two participants explain their most pressing issue
- Using the General issues we will find what patterns are most appropriate to solve them

* We will use an offline version of the wiki for this tutorial

Agenda

- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion



Seretta's story

- company background
  - Steria-Mummert-ISS, Hamburg / Germany
  - Standard Software for insurances
- my background
  - developer
  - test manager
- 2001, wanted to start automation
  - *NO PREVIOUS AUTOMATION*

Resolving Patterns:

1. SET CLEAR GOALS
2. MANAGEMENT SUPPORT
3. DEDICATED RESOURCES
4. RIGHT TOOLS
5. AUTOMATION ROLES
6. PLAN SUPPORT ACTIVITIES
7. MAINTAINABLE TESTWARE
8. AUTOMATE WHAT'S NEEDED
9. TAKE SMALL STEPS
10. UNATTENDED TEST EXECUTION

automate regression tests for the registration product

We chose an older tool that could drive our application reliably

I knew how to write good code for the automation

Done

Done

Done

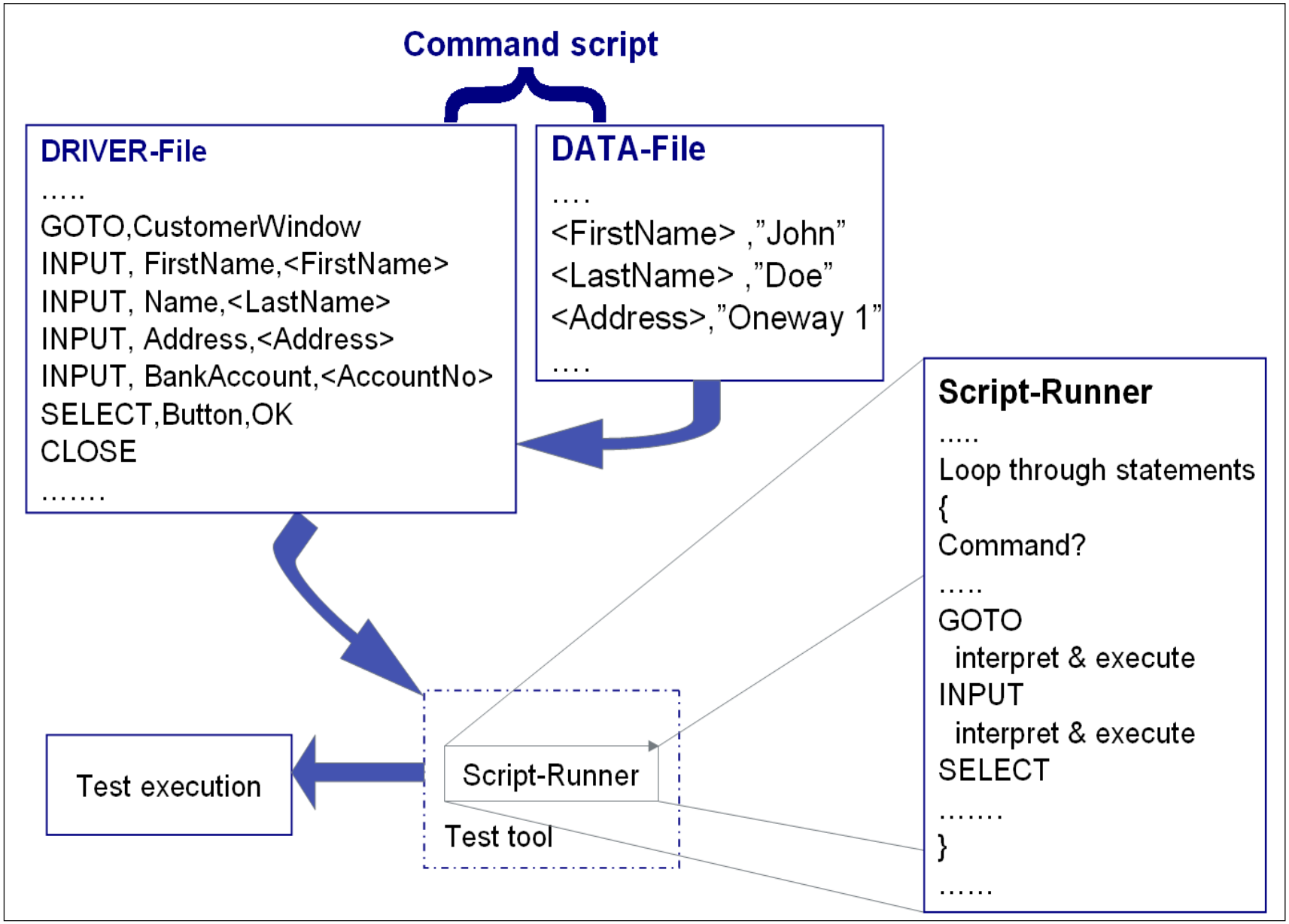
MAINTAINABLE TESTWARE

- **Summary**:
  Design testware so it doesn't have to be updated for every little change in the Software Under Test (SUT)
- **Description**:
  - Identify the most costly and/or most frequent changes
  - Design testware to cope best with those changes
- **Implementation**
  - If objects are frequently renamed use OBJECT MAP or devise a translation table for names
  - GOOD DEVELOPMENT PROCESS and GOOD PROGRAMMING PRACTICES
  - DOMAIN-DRIVEN TESTING using Keywords

Command-driven testing

- developed a keyword-type test architecture
  - minimize scripts that interface directly to the tool
    - **TOOL INDEPENDENCE**
  - organized into a Driver-part and a Data-part
    - **DATA-DRIVEN TESTING**
  - developed ISS Test Station to make it all work
    - **TEST AUTOMATION FRAMEWORK**
  - keywords as simple commands
    - e.g. “INPUT”, “SELECT”
    - **KEYWORD-DRIVEN TESTING**

Command script example



Success

- the automation gave us good benefits:
  - we had a good architecture and framework
  - easy to add new automated tests
  - company-wide automation standards
  - automated everything we had planned
  - expected our ratio of automated to manual tests to continue to grow

What happened next: Stagnation

- marketing success = automation problem
  - increase in business, new features
  - developers and testers needed on business-critical tasks, taken off automation
  - leading to new issues:
    - *INADEQUATE SUPPORT*
    - *INADEQUATE COMMUNICATION*
    - *NO INFO ON CHANGES*
  - no time for automation, which would have given them more time!

Patterns missed:

1. SET CLEAR GOALS
2. MANAGEMENT SUPPORT
3. DEDICATED RESOURCES
4. RIGHT TOOLS
5. AUTOMATION ROLES
6. PLAN SUPPORT ACTIVITIES
7. MAINTAINABLE TESTWARE
8. AUTOMATE WHAT'S NEEDED
9. TAKE SMALL STEPS
10. UNATTENDED TEST EXECUTION

I was doing the automation in my "spare time"

Initiated by management, but not enough

I was automator, tester, developer and project leader

at first I didn't need support, didn't think about this

MANAGEMENT SUPPORT

Implementation

Build a convincing TEST AUTOMATION BUSINESS CASE. Test automation can be quite expensive and requires, especially at the beginning, a lot of effort.

No business case ➔

- No real commitment from management to continue to support test automation,
- Just as for testing in general, test automation doesn't come first!

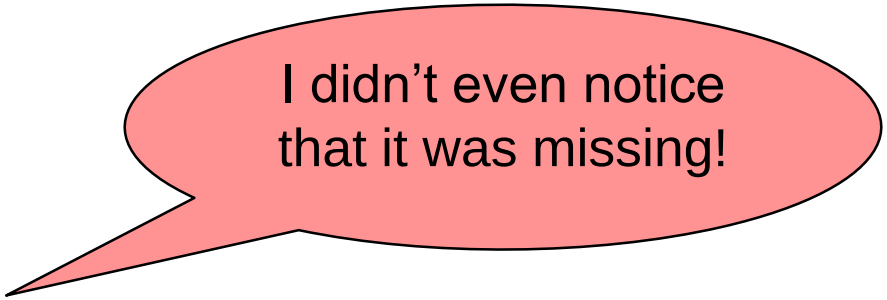
Management initiation of the automation was good, but it wasn't enough – needs ongoing support

MANAGEMENT SUPPORT

Implementation

A good way to convince management is to DO A PILOT. In this way they can actually “touch” the advantages of test automation and it will be much easier to win them over.

- SET CLEAR GOALS
- RIGHT TOOLS
- AUTOMATION ROLES
- TAKE SMALL STEPS
- Take time for debriefing when you are through and don't forget to LEARN FROM MISTAKES



Other Issues

- ***INADEQUATE TEAM***
  I could not select the team and had to manage with colleagues who were not needed for development!

- **AUTOMATION ROLES**
- **FULL TIME JOB**

If you have a pattern, that at least confronts you with the problem from the beginning. You cannot say afterwards that you didn't think about it!

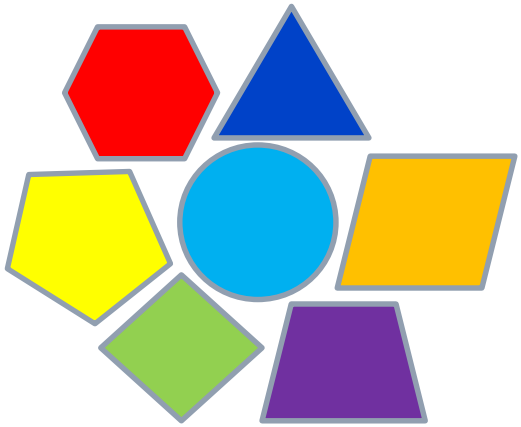
.

Case study conclusion

- we unknowingly applied some patterns
  - ✓ enabled us to achieve success
- we didn't think about other patterns
  - ➢ they caused us serious problems later on
- many other people have similar experiences
  - some succeed, some don't
  - the test automation patterns can help you on the road to success

Agenda

- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion

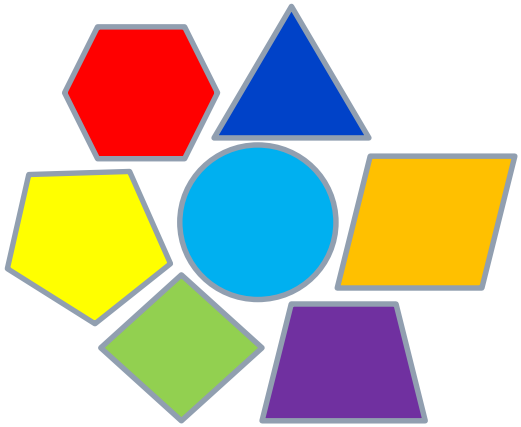


Exercise

- 5 scenarios
  - choose 1 to start, other(s) afterwards
- Load the offline wiki from the stick
- Explore the offline wiki with a partner
- Using the General Issues, identify the problems for the person in the scenario
  - what issue(s)?
  - what pattern(s) would be most appropriate in this context?
- solution ideas will be given

Agenda

- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion



More about implementing patterns

- using a few examples
  - CLEAR GOALS
  - ABSTRACTION LEVELS
  - DATA-DRIVEN vs KEYWORD-DRIVEN
- other patterns (chosen by you)
  - (time permitting!)

Pattern: SET CLEAR GOALS

Define the automation goals / objectives from the very beginning in a way that is clear and understandable to all.

Context

This pattern is always applicable, although the goals may be different at different stages.

- when you first consider test automation
- when an automation initiative is getting started
- when your automation is going well
- when you want to revitalize a stalled automation effort.

If you don‘t know what your goals are, how do you know that you are going in the right direction?

Pattern: SET CLEAR GOALS

Description

- automation objectives defined clearly up front
  ➔ no disappointment later.

- Inform your managers about what is feasible and what is not.

Goals should be measurable so that you can tell if you have achieved them or not

Pattern: SET CLEAR GOALS

Implementation examples

- Selected regression tests should run x-times faster and y-times more often.
- Tests too complex to perform manually are to be automated. Define exactly which ones.

Not good goals

- Confuse the goals for automation with goals for testing.
- Automate all manual tests

Potential problems

Be sure to remind management that it may not be possible or advisable to automate every test case.

Good objectives/goals for automation?

- run regression tests evenings and weekends

  only if they are worthwhile tests!

- increase test coverage

  can be a good one but depends what is meant by “test” coverage

- run tests tedious and error-prone if run manually

  good automation objective

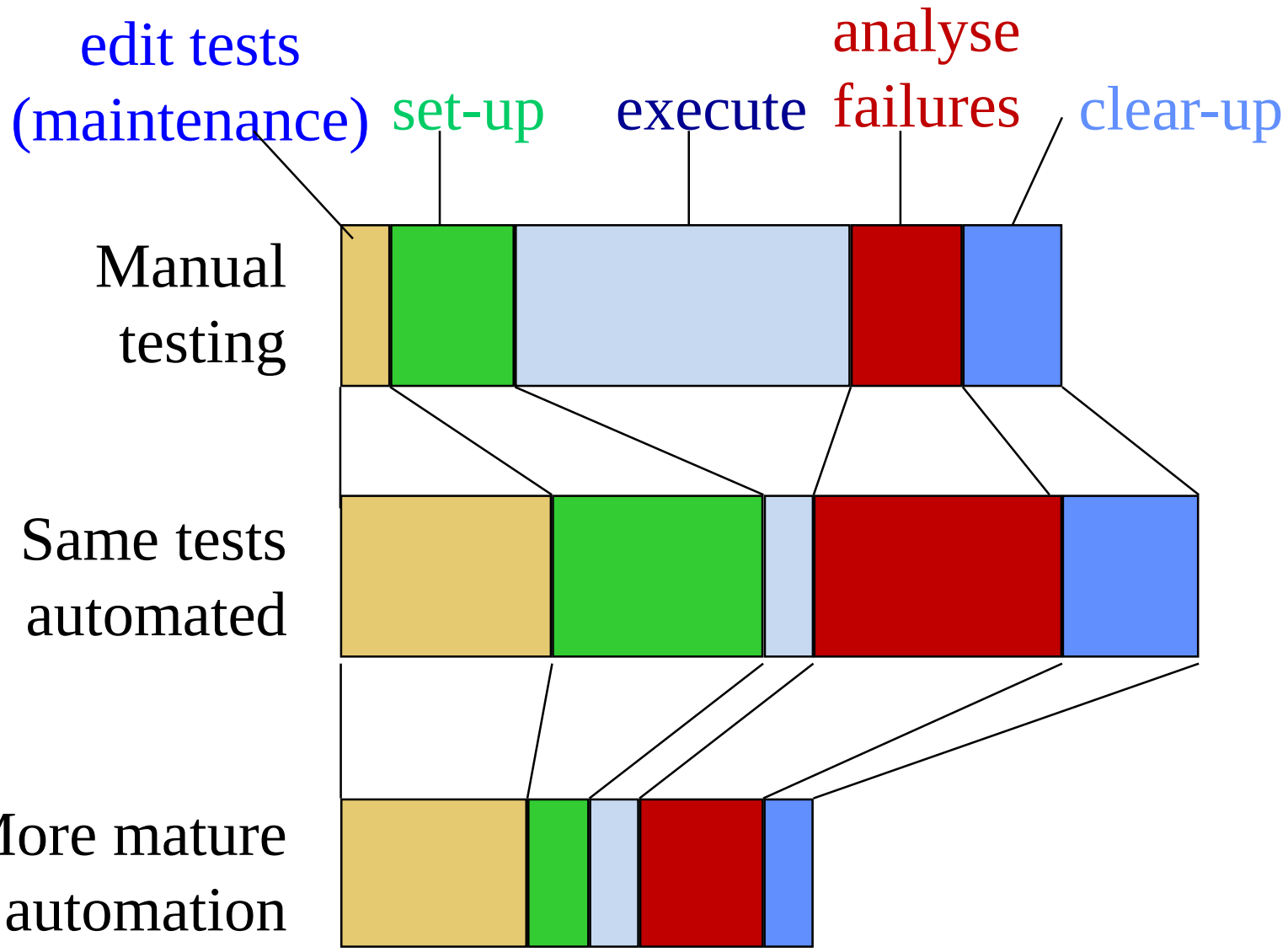
- gain confidence in the system

  an objective for testing, but automated regression tests help achieve it

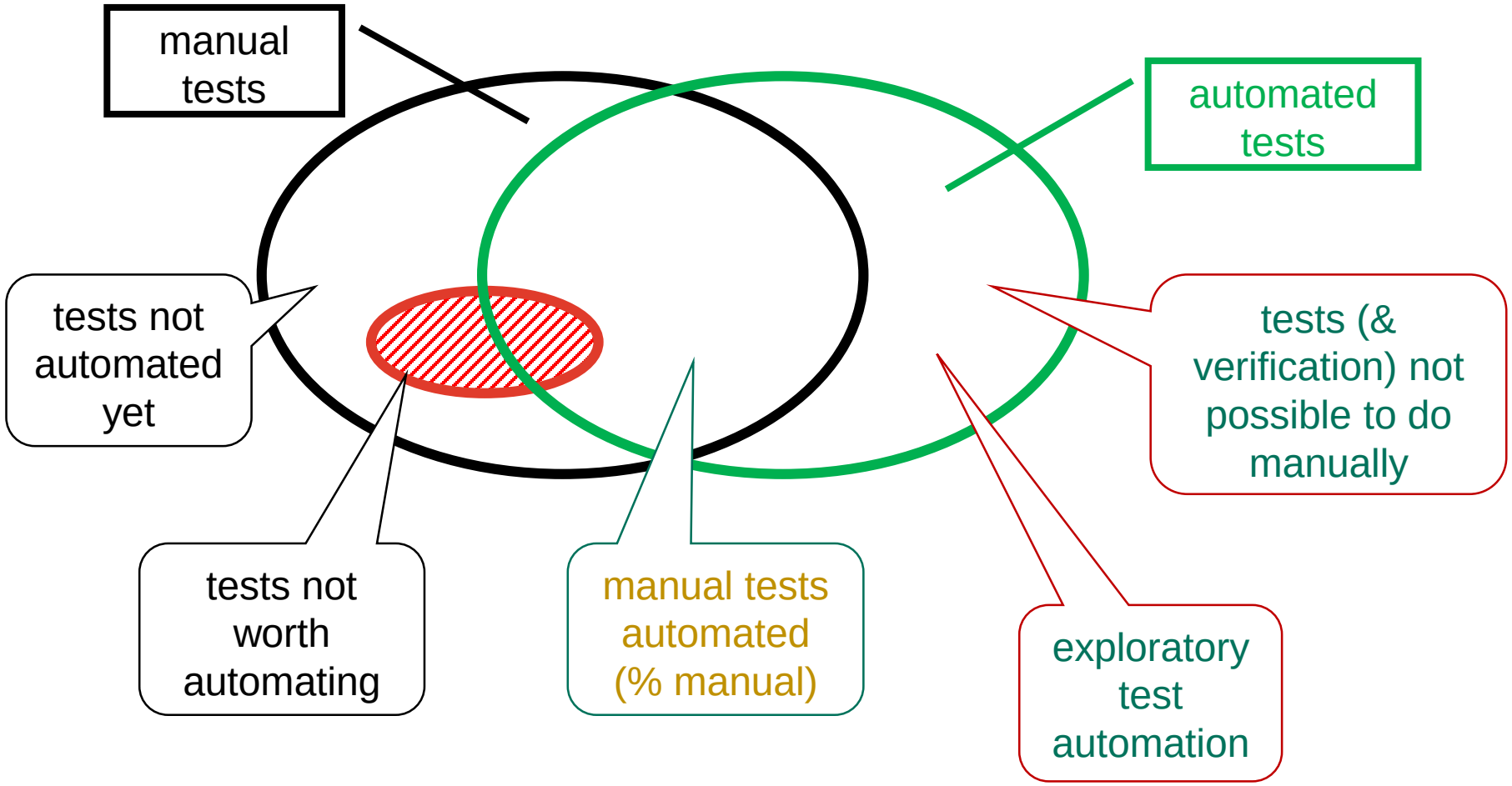
- reduce the number of defects found by users

  good objective for testing, maybe not a good objective for automation!

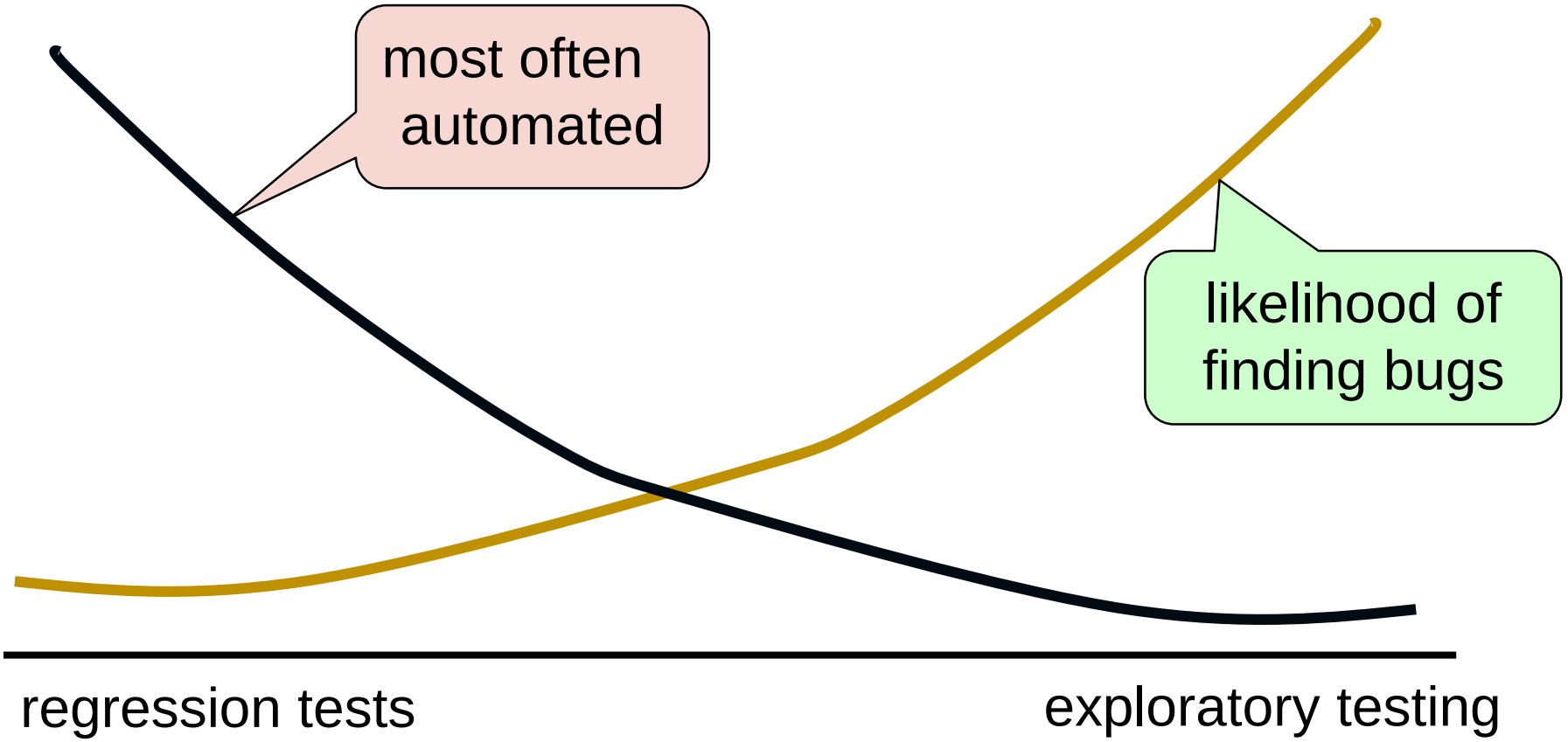
Reduce test execution time



Automate x% of the manual tests?



Tests usually automated?



automation objective – find lots of bugs?

No! - not for regression test automation

Find more bugs?

- tests find bugs, not automation
- automation is a mechanism for running tests
- the bug-finding ability of a test is not affected by the manner in which it is executed
- this can be a dangerous objective
  - especially for regression automation!

Automated tests	Manual Scripted	Exploratory	Fix Verification
9.3%	**24.0%**	**58.2%**	**8.4%**

Experiences of Test Automation, Ch 27, p 503, Ed Allen & Brian Newma

Pattern: ABSTRACTION LEVELS

Build testware that has one or more abstraction layers (or levels of separation)

Context

Apply this pattern for long-lasting and maintainable automation

- not necessary for disposable scripts

If you don’t take this into consideration in the design of your testware architecture, you are likely to have *HIGH MAINTENANCE COSTS* and will not achieve the success you could have had.

Pattern: ABSTRACTION LEVELS

Description

- separate your testware from the tool-specific scripting language as much as possible
- make your scripts modular and independent (following GOOD PROGRAMMING PRACTICES)
- build an interface so that testers and others can easily write and run automated tests

Pattern: ABSTRACTION LEVELS

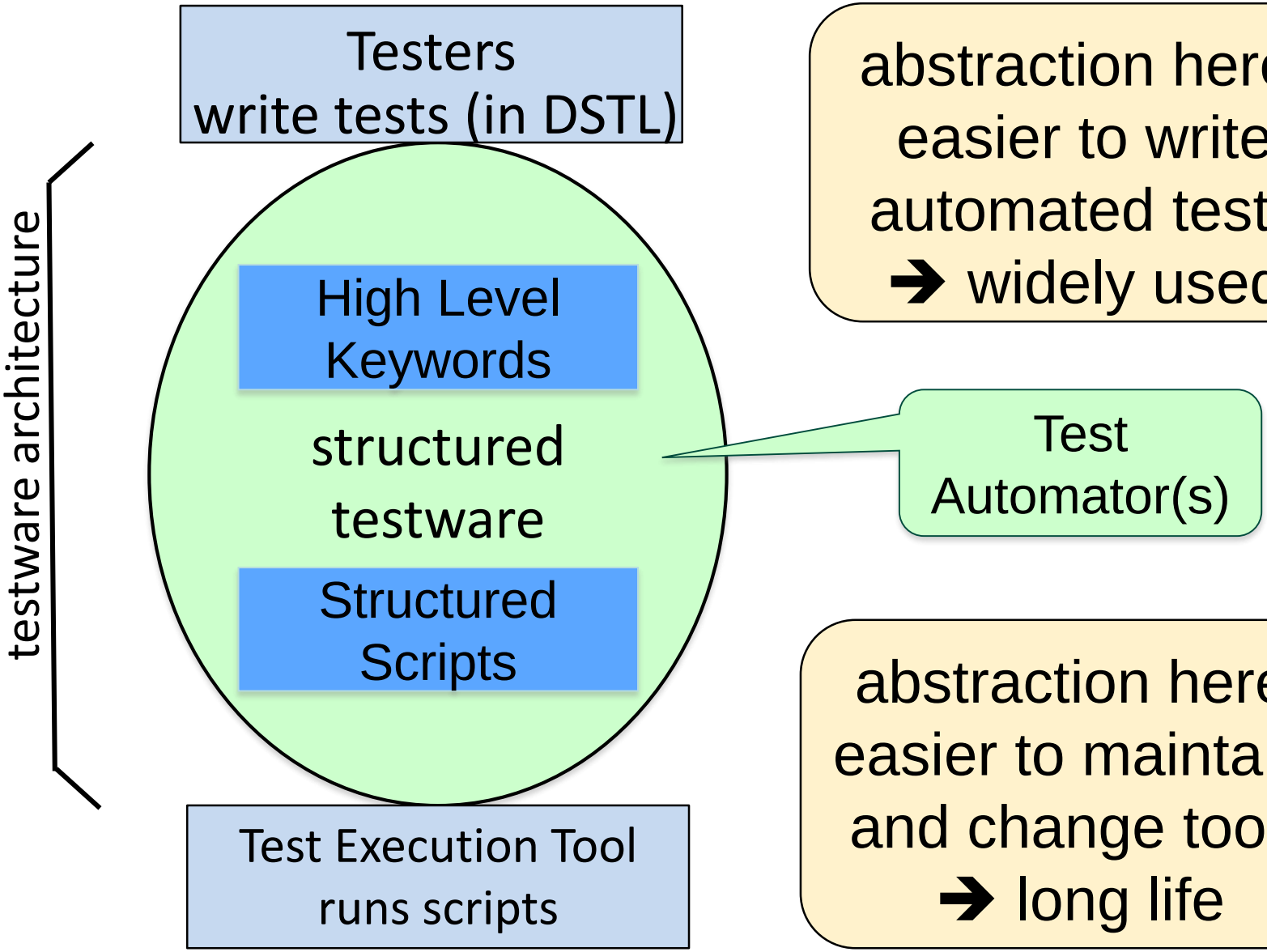
Implementation examples

- DATA-DRIVEN TESTING
- KEYWORD-DRIVEN TESTING
- DOMAIN-DRIVEN TESTING
- MODEL-BASED TESTING

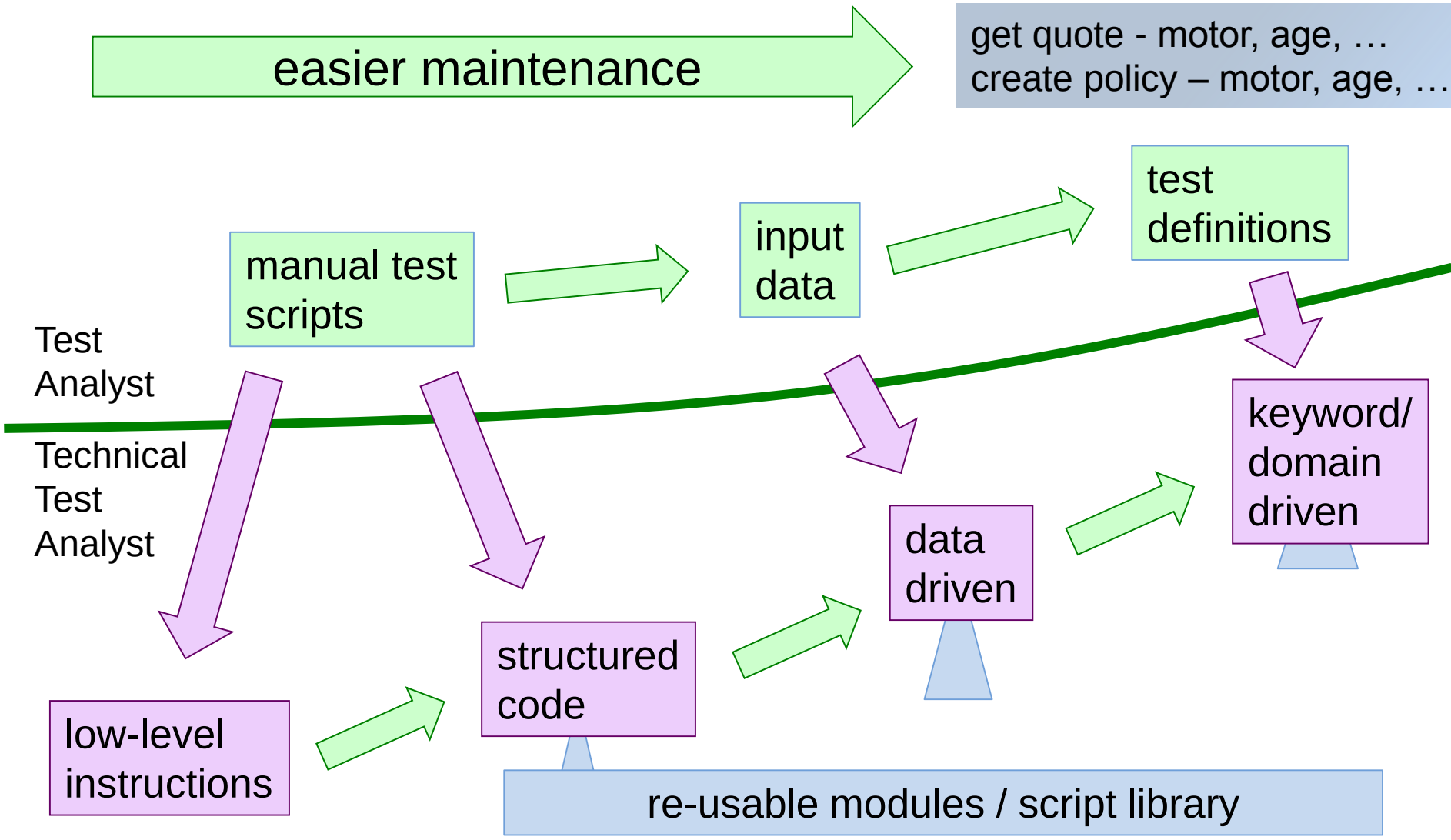
Potential problems

It takes time and effort to build a good TESTWARE ARCHITECTURE that uses levels of abstraction

Abstraction levels



Progression of aut. implementation



Data-driven example

Name	Earn1	Earn2	Earn3
Pooh	100	150	125
Piglet	75	90	80
Roo	120	110	65

```
Sub RunTests()
    For each Row in file
        Call OpenApplication("TaxCalculator")
        Open TaxCalcData.csv
        Read Name
        Read Earn1
        Read Earn2
        Read Earn3
        Call CalculateTax(Name, Earn1, _
            Earn2, Earn3)
        Call SaveAsAndClose(Test & " Results")
    Next Row
End Sub
```

Keyword-driven example

Keyword file (option 1)

```
CalculateTax Pooh, 100, 150, 125
CalculateTax Piglet, 75, 90, 80
CalculateTax Roo, 120, 110, 65
```

Keyword file (option 2)

```
CalculateTax
    Pooh, 100, 150, 125
    Piglet, 75, 90, 80
    Roo, 120, 110, 65
```

Keyword file (option 3)

```
CalculateTax Datafile.txt
```

Datafile.txt

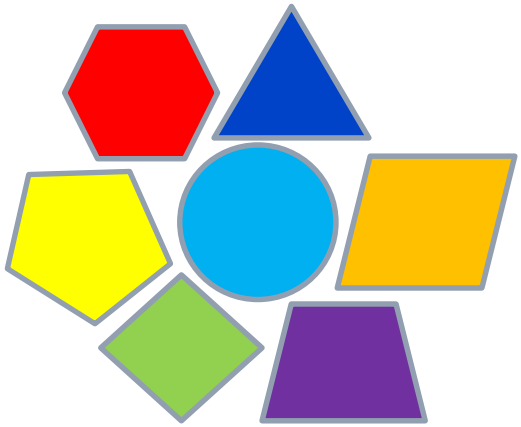
Name	Earn1	Earn2	Earn3
Pooh	100	150	125
Piglet	75	90	80
Roo	120	110	65

Keyword file (Robot Framework)

Test Case	Action	Arg1	Arg2	Arg3	Arg4
Check Tax	Calculate Tax	Pooh	100	150	125
	Calculate Tax	Piglet	75	90	80
	Calculate Tax	Roo	120	110	65

Agenda

- Background
- Issues and Patterns
- Case study
- Exercise
- More about some patterns
- Conclusion



Feedback/discussion

- discussion of issues, patterns, wiki
  - did you use the General Issues?
    - were they helpful?
  - what issues & patterns did you look at?
  - what was most useful for you?
  - what else would be helpful?
  - any problems finding what you wanted?
  - how will you apply the patterns to your own automation?

Using the wiki

- this is a new resource that will grow
- the more you add, the more valuable it will be
  - please add your experiences of patterns, tips, problems, solutions
  - feel free to email us with your ideas
    - we can put your text on the wiki
    - or give you access to edit the wiki yourself
- we welcome feedback on the wiki ☺

Finally

- Thank you for coming today!
- Any questions now?
- Contact us afterwards:
  - info@dorothygraham.co.uk
  - srttgmb@yahoo.com
- All the best with your test automation!

WIKI: testautomationpatterns.wikispaces.com