MAKING (BUSINESS) SENSE OF AUTOMATED TESTING

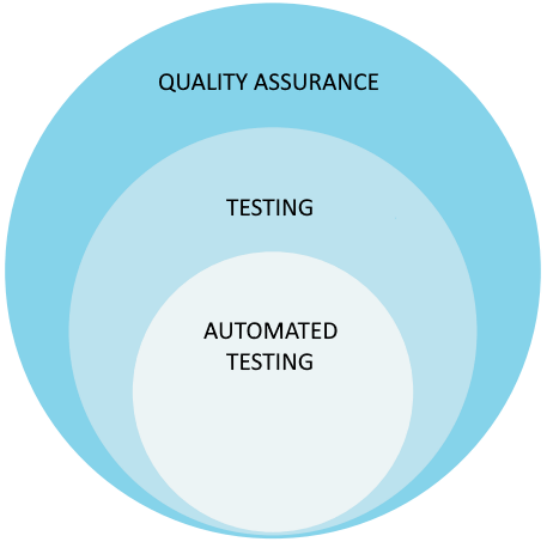
Iancho Dimitrov
linkedin.com/in/iancho
@iandim

Agile Testing Days, Potsdam *November 7th, 2019*

TEST AUTOMATION?

GOAL

To **efficiently and sustainably** provide **actionable insight** on the state of quality through **efficient test execution**.



MAKING (BUSINESS) SENSE?

PRICE-VALUE EQUATION

VALUE = PERCEIVED BENEFITS – PERCEIVED COSTS

EARNED VALUE = REALIZED BENEFITS – ACTUAL COSTS

YOUR PERSPECTIVE?

Trophies | Scars | Part-of-my-job

MY PERSPECTIVE

PLAN

1. WHY CARE?
2. HOW – BEFORE
3. HOW – DURING
4. DISCUSSION

WHY CARE?

- Test Automation **is not always**
  - appropriate / necessary
  - cost effective

- Even when appropriate and cost effective it **often fails** to deliver results – and to survive

WHY CARE?

Possible BENEFITS

- reduced
  - cost of testing
  - time needed for testing
    (i.e. “time to market/production”)
- more time spent by QA Engineers on meaningful work with higher added value

WHY CARE?

Possible BENEFITS

- test EXECUTION coverage
  - broader (incl. versions, environments, devices)
  - deeper (variations of test data)
  - tests that are impossible to do manually
- DevOps, CI/CD

WHY CARE?

Possible BENEFITS

- test more often –> catch bugs early –> reduced bug-fixing cost
- higher reliability (people make mistakes in repetitive tasks)
- motivated, fulfilled and happy QA Engineers
- can be used for monitoring on production

WHY CARE?

Source: "The Evolution of Test Automation 2018"
QASymphony and TechWell Survey

What benefits have you experienced as a result of adopting test automation?

(Respondents could select more than one answer)

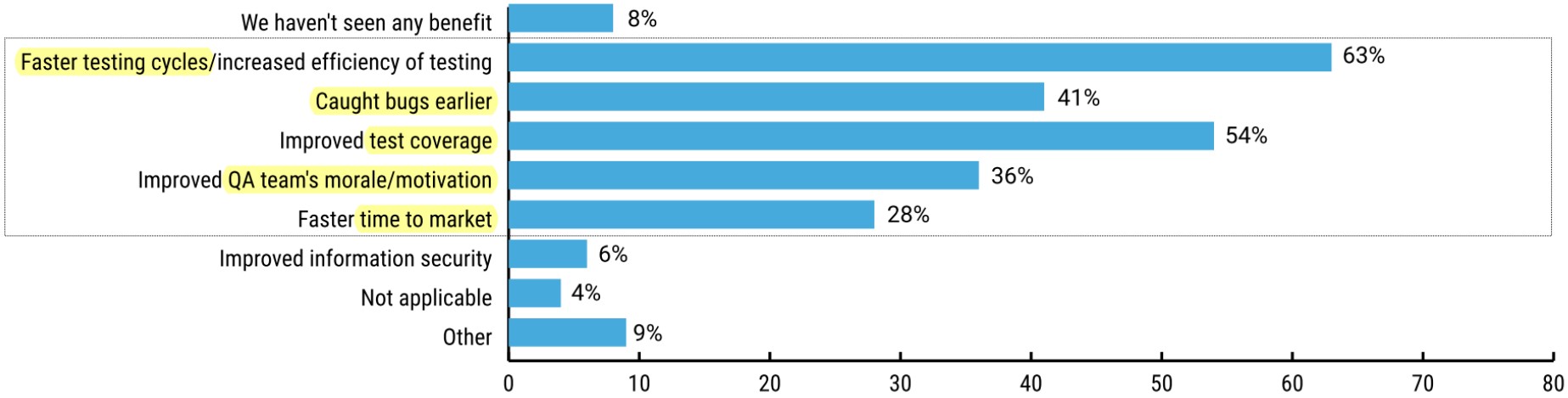


Fig. 9 Percentage of Respondents

WHY CARE?

…but it comes with its COSTS

- commercial tools - licenses
- infrastructure resources
- team members training
- PoC implementation

WHY CARE?

…but it comes with its COSTS

- new tests design and implementation
- (!) framework architecture setup, maintenance & enhancement
- (!) tests execution, results analysis, reporting and defect tracking
- (!) tests maintenance
- test data management

WHY CARE?

Major RISKS

- focus on tool selection – instead of on strategy, plan and process
- slow start, high initial costs – and canceled prematurely
- **(!)** exploding maintenance costs (changes in SUT, platforms, etc.)
- lack of needed skills – and high hopes on “snake oil” tools

WHY CARE?

Major RISKS

- wrong expectations to replace almost all other QA activities
- wrong expectations for quick payback
- **(!)** failure to achieve recognizable goals (esp. with none defined)
- unhappy QA Engineers
- TA assets to become “shelfware” – in full or partially

WHY CARE?

Source: "The Evolution of Test Automation 2018"
QASymphony and TechWell Survey

What challenges have you experienced with test automation?

(Respondents could select more than one answer)

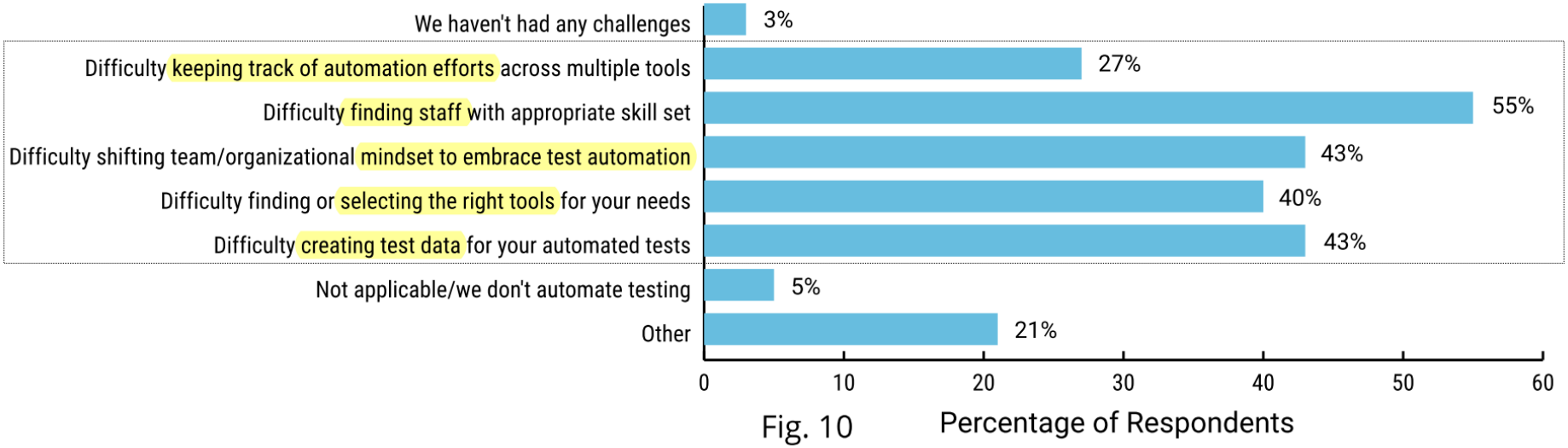
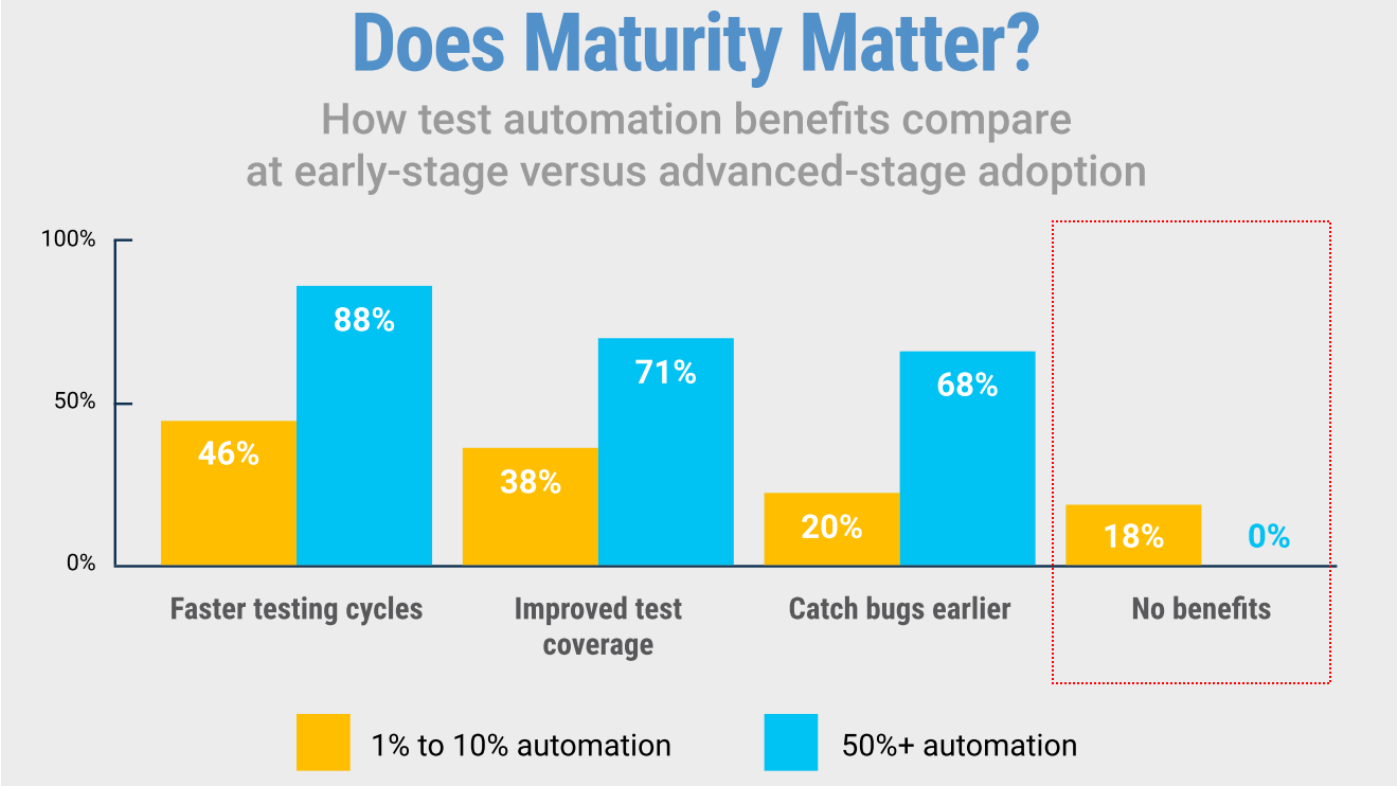


Fig. 10

WHY CARE?

Source: "The Evolution of Test Automation 2018"
QASymphony and TechWell Survey



WHY CARE?

WRAP-UP

- Test Automation is **NOT a fit for every project**

- Many possible **benefits** – but they come at a **cost**

- Many **risks** – achieving long-term success **not trivial**

- Hard to start **and** carry on

YOUR PERSPECTIVE

AGREE?

PLAN

1. WHY CARE?
2. HOW – BEFORE
3. HOW – DURING
4. DISCUSSION

HOW – BEFORE

1	**TAKE IT SERIOUSLY!**	(LONG-TERM INITIATIVE BY ITS OWN)
2	**EVALUATE FOR YOURSELF FIRST**	(THINK AS A PM / BUSINESS MANAGER)
3	**PHASED APPROACH**	
4	**EXPERIENCED TEAM**	
5	**MOTIVATED PROPOSAL**	(WITH THE BIG PICTURE IN MIND)
6	**DO A SUCCESSFUL POC → CALIBRATE TRIAL PHASE PROPOSAL → MOVE ON**	

HOW – BEFORE

TAKE IT SERIOUSLY!

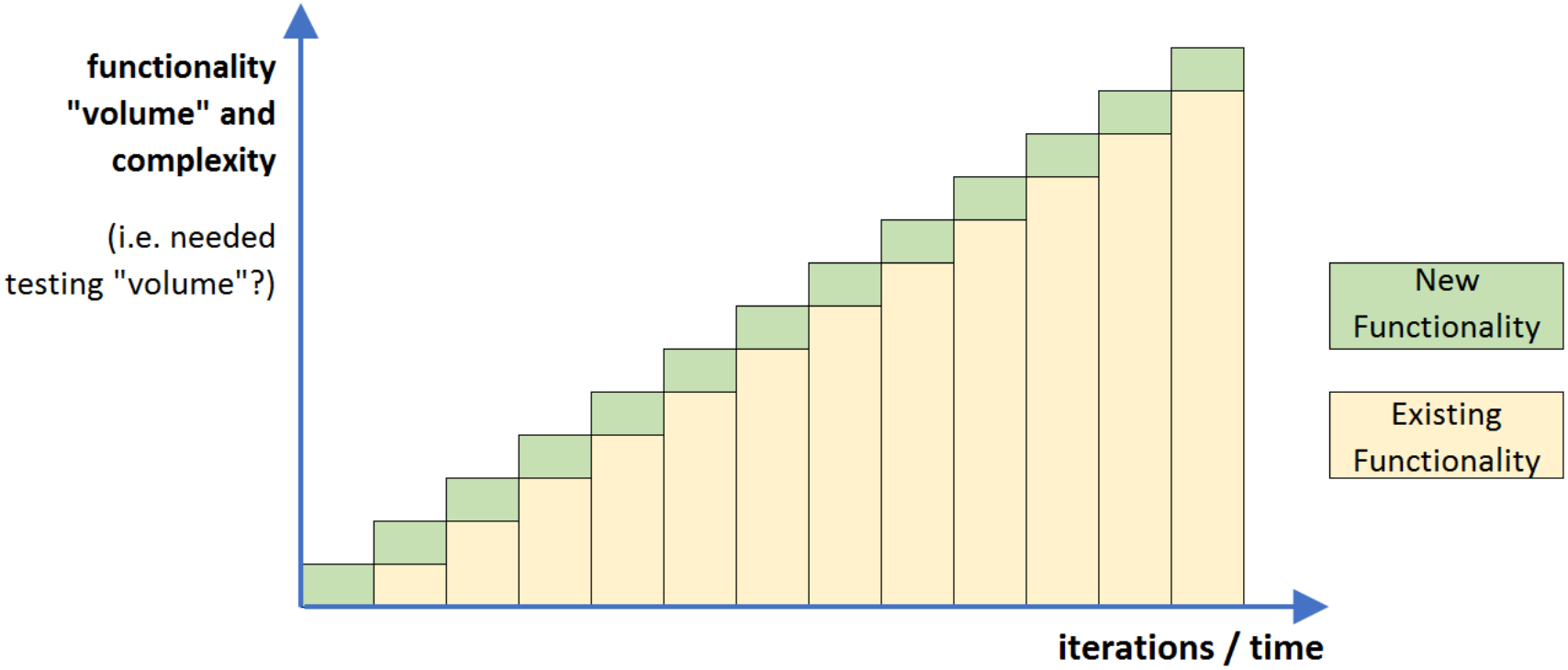
a separate long-term ~~project~~ initiative

HOW – BEFORE

2. Evaluate for yourself first – benefit multipliers?

- long-term project?
- release often?
- many combinations to test? ENVs, devices, OSs, data, etc.
- CI/CD?

TESTING NEEDED OVER TIME



HOW – BEFORE

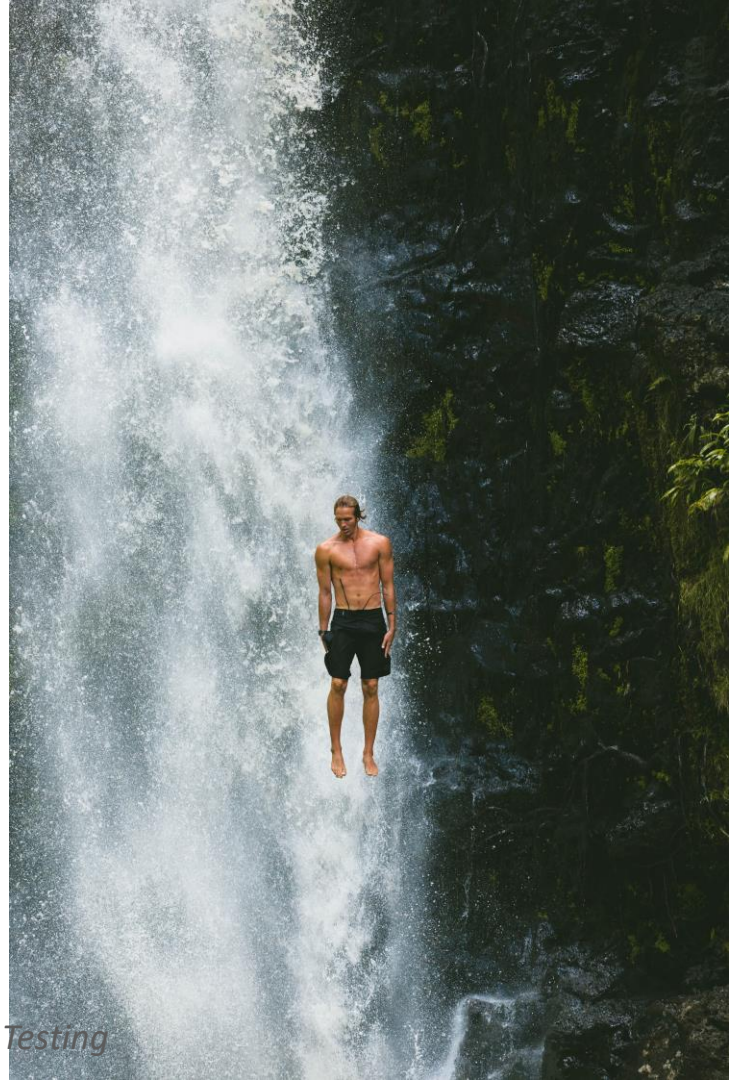
3. Propose a phased approach

- Research & Analysis (invest in it!)
- Proof of Concept - 2-4 weeks
- Trial Phase - 2-3 months
- Ongoing initiative - long term

HOW – BEFORE

4. Gather an experienced team

- TA (initiatives) Leader
- TA Technical Expert / Architect



HOW – BEFORE

5. Motivated proposal

- initial findings – TA applicable?
- a comprehensive plan for the PoC
- high-level plan for the Trial Phase
- expected benefits

HOW – BEFORE

5. In the project plans include:

- costs/efforts estimation (budget)
- expected results & deliverables
- risks & mitigation plan
- team and time allocation
- implementation timeline

…and be pessimistic in all estimations

HOW – BEFORE

6. Move on – step by step

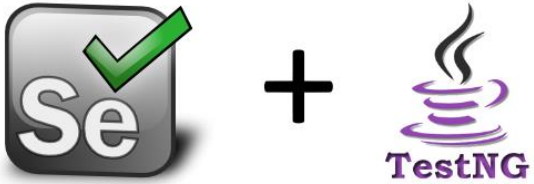
- **Get buy-in** from the Budget Owner (adapt proposal if necessary)
- **Do a successful PoC** – and demonstrate the results
- Prepare an **updated and detailed** TA Trial Phase project plan
- **Pitch the TA Trial Phase proposal**, get approval and go forward
- Make sure to **keep the key stakeholder(s) up to date** with progress at all times

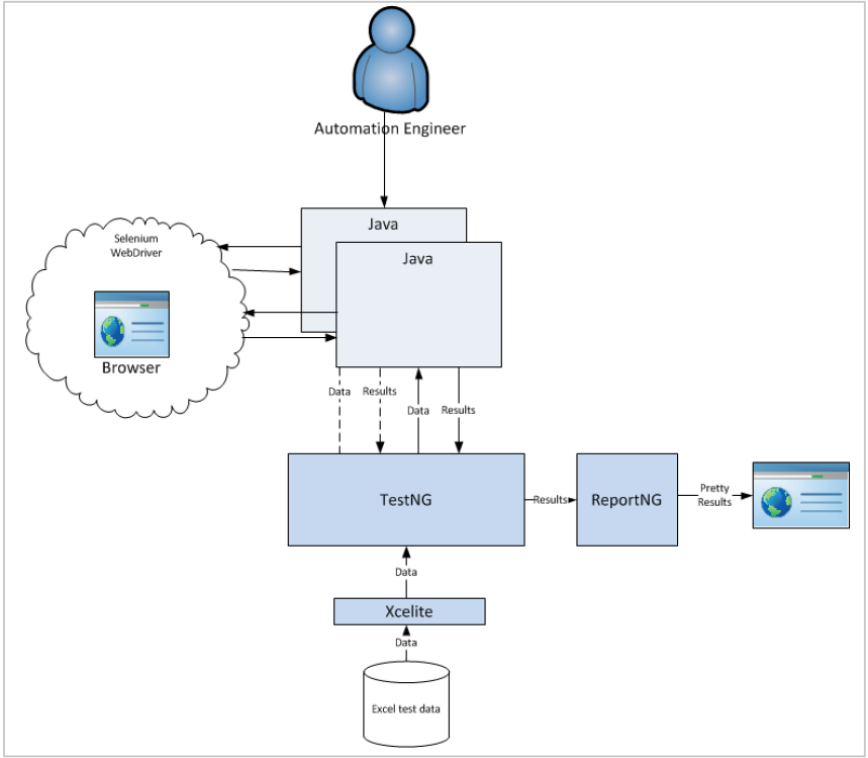
HOW – BEFORE

S

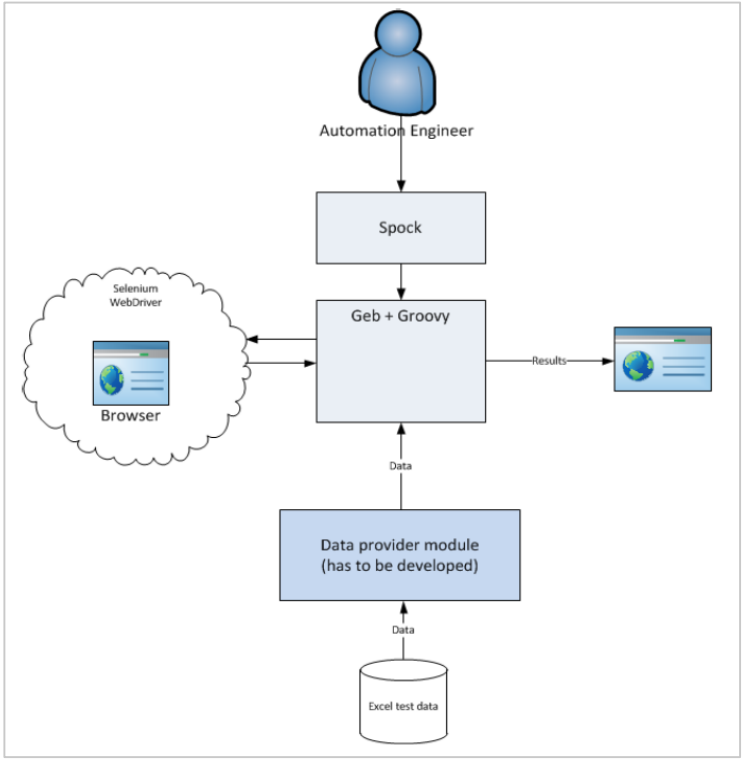
1	**TAKE IT SERIOUSLY!**	(LONG-TERM INITIATIVE BY ITS OWN)
2	**EVALUATE FOR YOURSELF FIRST**	(THINK AS A PM / BUSINESS MANAGER)
3	**PHASED APPROACH**	
4	**EXPERIENCED TEAM**	
5	**MOTIVATED PROPOSAL**	(WITH THE BIG PICTURE IN MIND)
6	**DO A SUCCESSFUL POC → CALIBRATE TRIAL PHASE PROPOSAL → MOVE ON**	

TOOLSET SELECTION – COMPARISON





TOOLSET SELECTION – COMPARISON



TOOLSET SELECTION – COMPARISON

+

+ SPOCK

Automated Testing for **Application Name –**

Technology Stack Research Report (v. 2.0)

Contents

TOOLSET SELECTION – COMPARISON

ID	Category	Geb + Spock	C.	Selenium + TestNG + JUnit
1	foundation	Uses Web Driver (Selenium) to interact with the browser.	=	Uses WebDriver (Selenium) to interact with the browser.
2	test fundamental	Web elements can be waited for both implicitly and explicitly – these two techniques are needed to test applications that are asynchronously loading page elements. Especially as the latter appear with various delays.	=	Web elements can be waited for both implicitly and explicitly – these two techniques are needed to test applications that are asynchronously loading page elements. Especially as the latter appear with various delays.
3	test fundamental	Test data can be fed in through Excel, CSV, etc. files.	=	Test data can be fed in through Excel, CSV, etc. files.
…	…	…	=	…
6	understandability	More human-like scripts.	>	More verbose code.
8	test advanced	Can handle branching navigation natively (meaning different next pages opening for different values selected on the current page).	>	More complex code needed to handle branching navigation.
9	reusability	Dynamic language makes it easier to write test utilities.	>	More expertise needed to create test utilities.
…	…	…	>	…
12	Maintainability	Written in Groovy, language that is not used by current team members.	<	Plain Java, the language in which the applications are written.
13	development	Poor IDE support.	<	Good IDE support.
14	Reliability	Exists for about six years, although community support is improving, still lagging behind.	<	The most popular and used technology stack. Open-sourced. Big community. One can rely on good support.
…	…	…	<	…

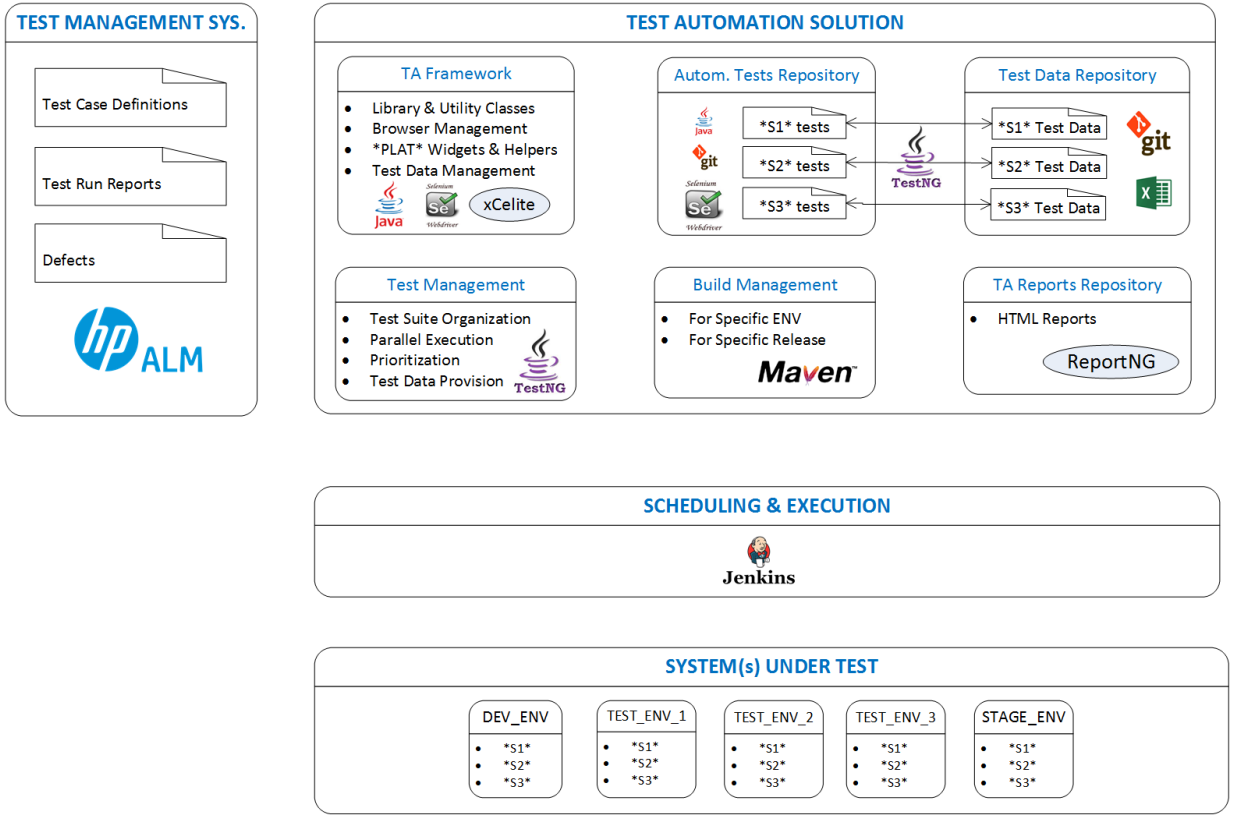
TOOLSET SELECTION – COMPARISON

+

+ SPOCK

ID	Aspect	"Winner"	Rationale
6	Test project backbone	Selenium + TestNG	All the addons implemented for the **Application Name 2** solution are listed above. Most of those will have to be implemented anew for the **Application Name** if the technological stack is switched (all but the JS framework).
7	Setup	Selenium + TestNG	It took some hours to get the **Application Name 2** automation working in **Client Name** infrastructure. It should be expected that similar amount of time will be needed for Geb-based solution.
8	Page waits	Geb + Spock	Geb has more configurable conditions for page loads that will make the tests more stable and precise.
9	Iframe support	Geb + Spock	As Java is static and strongly typed it was a struggle to get solution to the iFrame problem half as neat as is provided out-of-the-box in Geb.
10	Maturity	Selenium + TestNG	The fact that Selenium has far bigger community justifies why there are many more plugins and extensions available for this stack. Furthermore, as there are more resources in internet, problems are easier to resolve with this stack.
11	Branching navigation	Geb + Spock	In several occasions in **Application Name 2** application pages in a user flow show only if specific conditions are met (e.g. confirmation of deletion if related objects exist). Geb handles such situations a lot more neatly than what is done in **Application Name 2**.
12	Test data loading	Selenium + TestNG	With **Application Name 2** stack both Xcelite and TestNG make it easier to separate test data from scripts. With Geb there's no out-of-the-box solution.
13	Understandability	Geb + Spock	Geb tests will definitely be easier to understand. However, it should be noted that writing in Geb still requires technical expertise and cannot be done by non-technical people.
14	Side effects	Selenium + TestNG	Although dynamic languages allow you to do more easily extensions that require more code and tweaks in static ones the effect of a single not-well thought modification can be very negative and hard to trace.
15	Dynamicity	Geb + Spock	It is easier to write test utilities in dynamic languages.
16	IDE support	Selenium + TestNG	Spock is so specific DSL that there is no IDE that supports it well. For example compilation errors will be reported and auto completion and navigation options will be limited. This will affect the efficiency of the people creating the automated tests.
17	Language knowledge	Selenium + TestNG	Java is better known in **Client Name**, thus it will be easier for everyone in the team to write in the **Application Name 2** stack.
18	Customizable test reports	Selenium + TestNG	While we were able to get nicely looking test report in **Application Name 2** the report of Geb execution is very basic and there are far fewer options for improvement.
19	Relative element selection	Geb + Spock	In **Application Name 2** solution we had to copy-paste several times the common part of xPath between related elements. This means that if the DOM of the targeted page was changed in a way that affects the common part it had to be changed in several places. This will be resolved if Geb is used.

TEST AUTOMATION SOLUTION – ARCHITECTURE



PLAN

1. WHY CARE?
2. HOW – BEFORE
3. HOW – DURING
4. DISCUSSION

HOW – DURING

1	TAKE IT SERIOUSLY! (LONG-TERM INITIATIVE BY ITS OWN)
2	TA-SPECIFIC KPIS + MEASURABLE & REALISTIC GOALS
3	REUSABLE COMPONENTS
4	"AUTOMATED" TEST AUTOMATION
5	MEASURE AND COMMUNICATE
6	PRIORITIZE / DECIDE WHAT TO AUTOMATE

HOW – DURING

7	PROCESS & RESPONSIBILITIES
8	MAKE THE MOST OF TA (EXECUTION) RESULTS
9	TRACEABILITY BETWEEN ARTIFACTS
10	DESIGN FOR AUTOMATION
11	CROSS-FUNCTIONAL TEAM MEMBERS INVOLVEMENT

HOW – DURING

TAKE IT SERIOUSLY!

a separate long-term ~~project~~ initiative

DO TEST AUTOMATION RIGHT

+

MEASURE AND COMMUNICATE

DO TEST AUTOMATION RIGHT

2. Set & track TA-Specific Measurable Goals. Examples:

- Regression Test Suite for **”Sprint X-1”** – 100% automated
- **Smoke Test Suite** – up-to-date all the time
- All test **run results analyzed and distributed** in time (e.g. < 2 hrs.)
- **Effort spent for maintenance** < 15%, and all tests working

DO TEST AUTOMATION RIGHT

2. Set & track Measurable & Realistic Goals.
Anti-patterns:

- "test coverage"
  (analyzed and well-reported test runs matter) S
- "test automation of sprint delivery, within the sprint" S
- bugs found

DO TEST AUTOMATION RIGHT

3. Reusable components

- 2 aspects – technical & functional
- well documented
- (!) strict process for technical design

…crucial for both speed of implementation & maintenance!

REUSABLE COMPONENTS

Test Case ID	Test Case Steps	Module ID	Module Description / Notes	Module Used In
	1, 2, 3	1		All tests except T03
	4,5	2		All tests except T03
	6,7,8,9	3		All tests except T02
	10,11,12,13,14,15,16	4		All tests except T02
	1,2,3	1	< see module description above >	< see above >
	4,5	2	< see module description above >	< see above >
	6,7	5		T02, T04, T05, T06
	8,9	6		T02, T06,
	1,2,3,4,5	7		T03 and all next test cases with Prolongation option will reuse this (currently one such test)
	6,7,8,9	3	< see module description above >	<see above>
	10,11,12,13,14,15	4	< see module description above >	<see above>
	1,2,3	1	< see module description above >	<see above>
	4,5	2	< see module description above >	<see above>
	6,7	5	< see module description above >	<see above>

REUSABLE COMPONENTS

	8,9	8	[illegible]	T04, T06, [illegible]
	10,11,12,13,14,15	3	< see module description above >	<see above>
	16-22	4	< see module description above >	<see above>
[illegible]	1,2,3	1	< see module description above >	<see above>
	4,5	2	< see module description above >	<see above>
	6,7	5	< see module description above >	<see above>
	8,9	9	[illegible]	T05 and T06
	10-14	3	< see module description above >	<see above>
	15-22	4	< see module description above >	<see above>
[illegible]	1,2,3	1	< see module description above >	<see above>
	4,5	2	< see module description above >	<see above>
	6,7	5	< see module description above >	<see above>
	8,9	9	< see module description above >	<see above>
	10,11,12,13	8	< see module description above >	<see above>
	14,15,16,17	6	< see module description above >	<see above>
	18-22	3	< see module description above >	<see above>
	23-29	4	< see module description above >	<see above>
[illegible]	1,2,3	1	< see module description above >	<see above>
	4,5	2	< see module description above >	<see above>
	6-10	3	< see module description above >	<see above>

REUSABLE COMPONENTS

Name:	03_B[illegible]		Component			For Clarifi-cation	User Inte-rections
Steps	Step description	Implementation plan & hints	Reu-sable	Modi-fy	New		
1	Open a existin[illegible]	[illegible] -> selectK[illegible] see ApproveK[illegible]	1	-	-	-	6
2	1. Open "[illegible]" section 2. Click on "[illegible] 3. Create a new [illegible]it with [illegible]	1. Use K[illegible]NavigationPanelPage -> selectNavigationLinkByName to navigate to "Si[illegible] 2. Si[illegible] -> createN[illegible] Ne[illegible]	2	1	-	-	16
3	.....	.....	2	1	-	-	16
4	.....	.....	2	-	1	-	4
5	.....	.....	2	1	-	-	16
6	.....	.....	-	1	-	-	6
7	.....	.....	-	-	1	-	2
8	.....	.....	1	-	-	-	36

Steps count	9
User iteraction	102
Reusable components	10
Commponents which we can use with some modifications	4
New components	2
Steps fo clarification	0
Summary Implementaion time	1-2
Summary Testing time (mansdays)	1.5

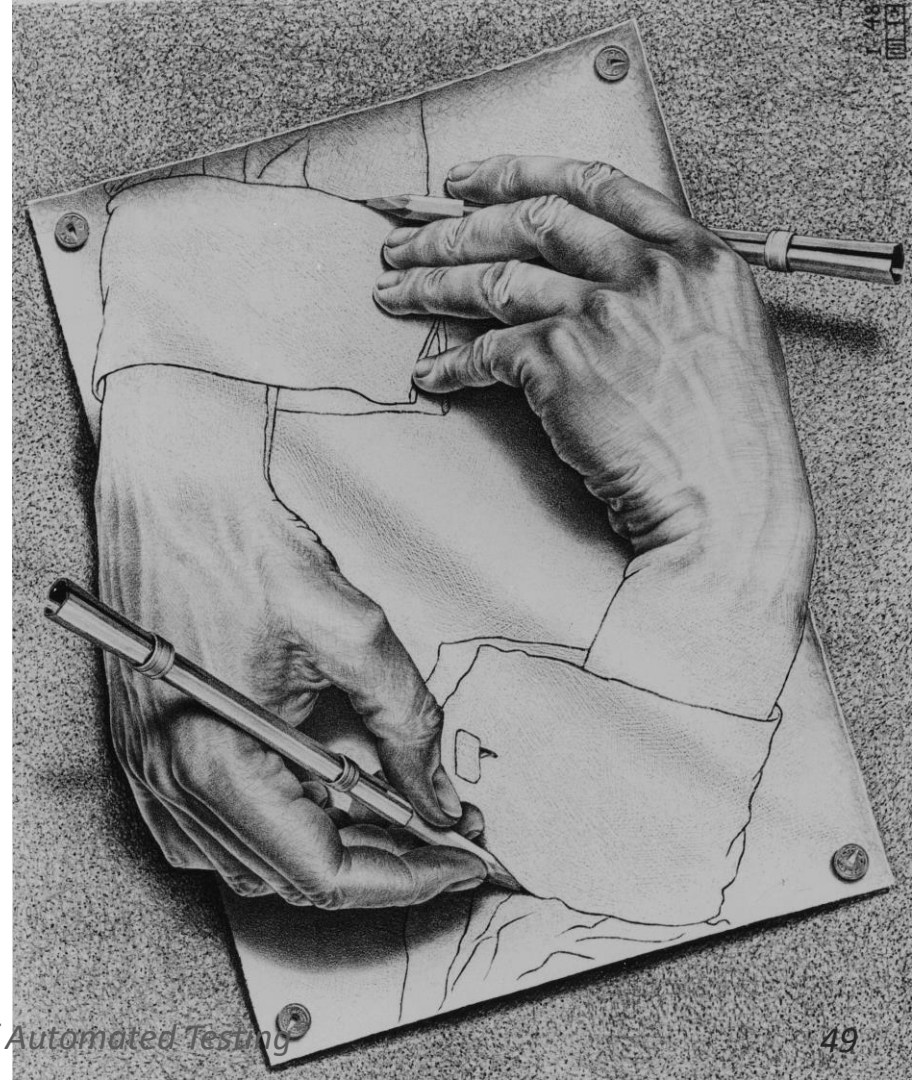
REUSABLE COMPONENTS

	Project	Test case name	Steps count	User Iteraction	Number of reusable components	Number of components need modification	Number of new components	Steps for clarification	Implementation Estimation (mandays)	Testing Estimation (mandays)
1		09_	13	158	8	7	1	1	1	1-2
2		04_	19	151	13	2	2	2	1	1
3		03_E	6	57	3	2	2	0	0.5	1
4		03_I	6	64	2	2	2	0	1	1
5		03_I	7	63	3	2	3	0	0.5-1	0.5-1
6		03_	8	89	8	3	3	0	1-2	1.5
7		02_I	32	261	10	2	18	3	5-6	2
8		06_Ei bs	6	298	12	0	14	0	1	1
9		06_	4	139	6	0	9	0	1	1

DO T.A. RIGHT

4. Automated "test automation"

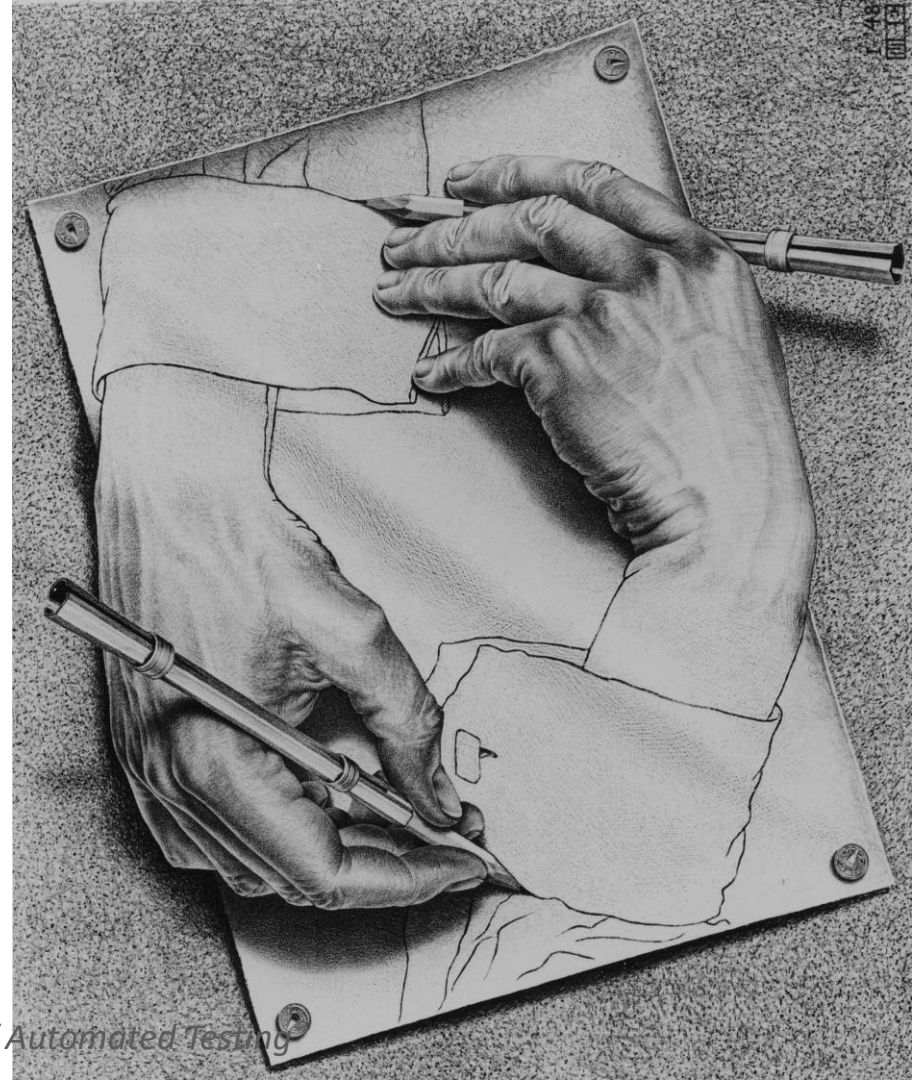
- configurable runs – different ENV/devices/SUT version
- test data prep. & clean-up
- scheduling
- parallel runs



DO T.A. RIGHT

4. Automated "test automation"

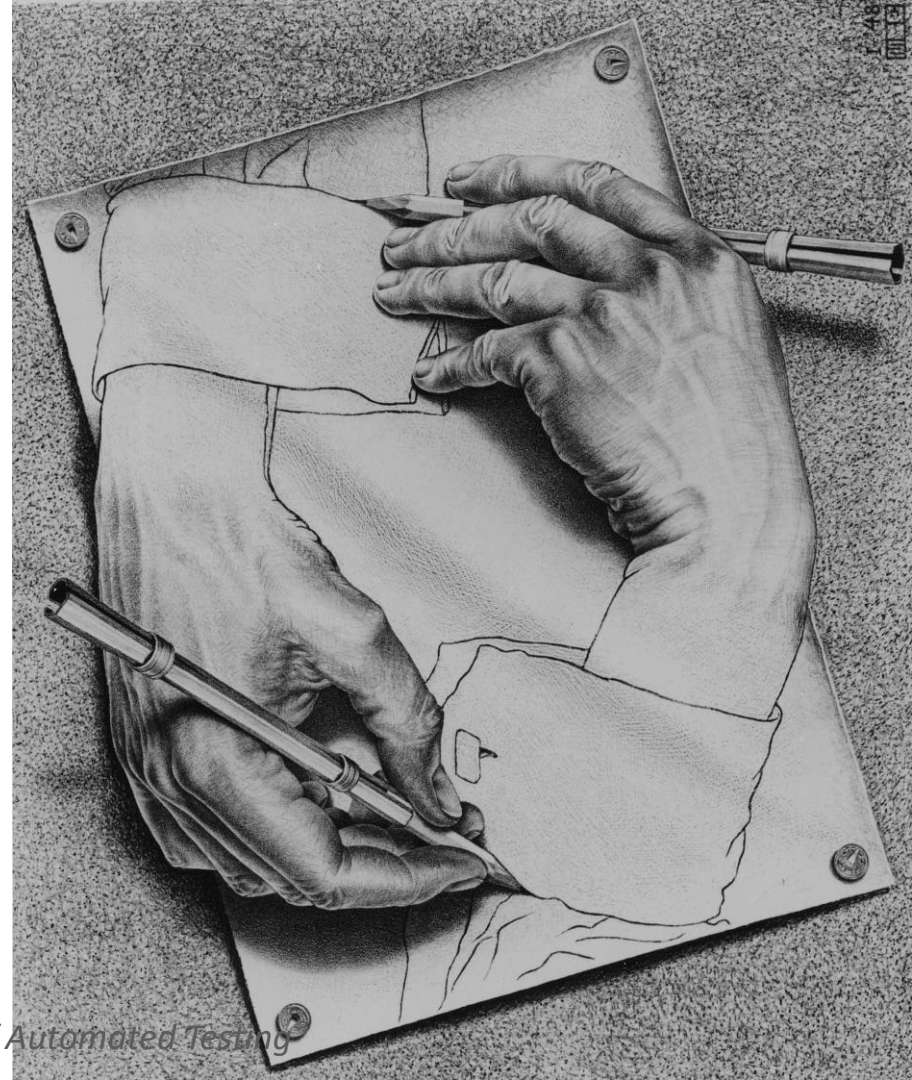
- 100% separation of Data from Tests
- (user friendly) reports, accessible by all team members – incl. screenshots, video recordings, logs



DO T.A. RIGHT

4. Automated "test automation"

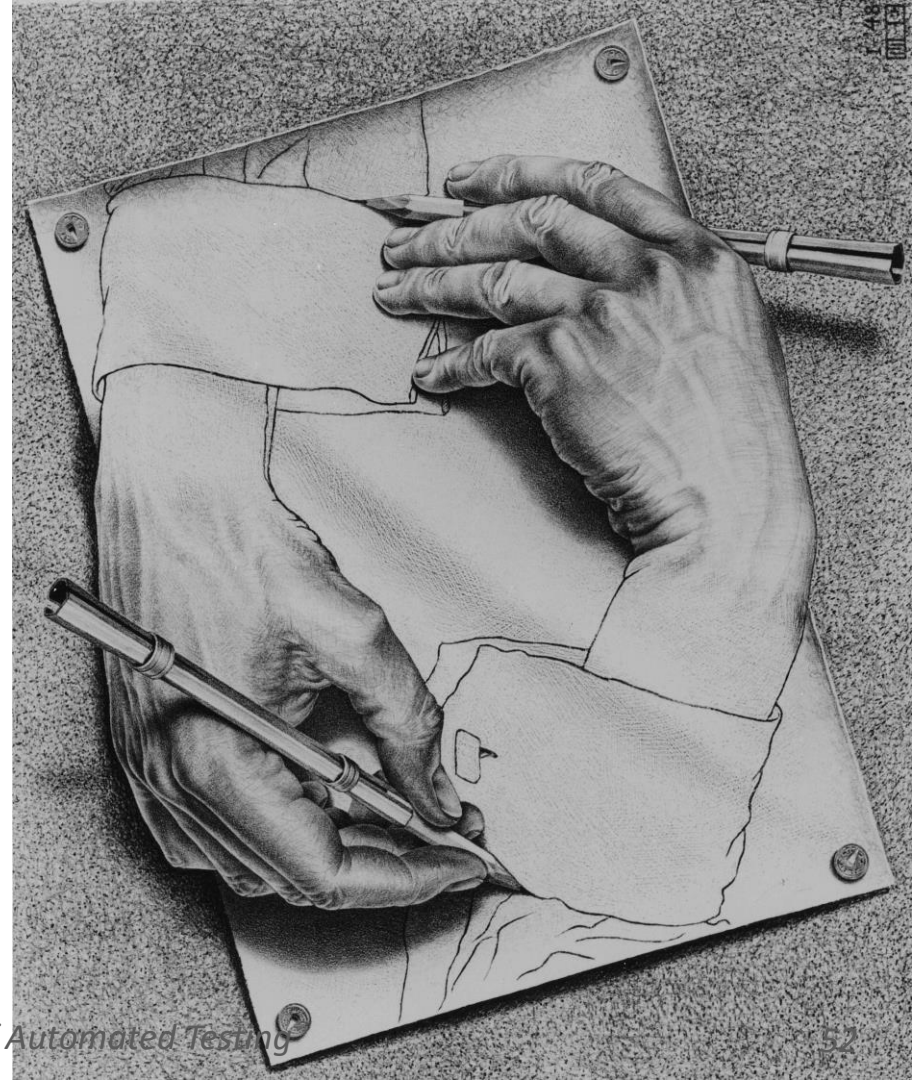
- integration with Test Management system
- non-TA team members enabled to execute test suites



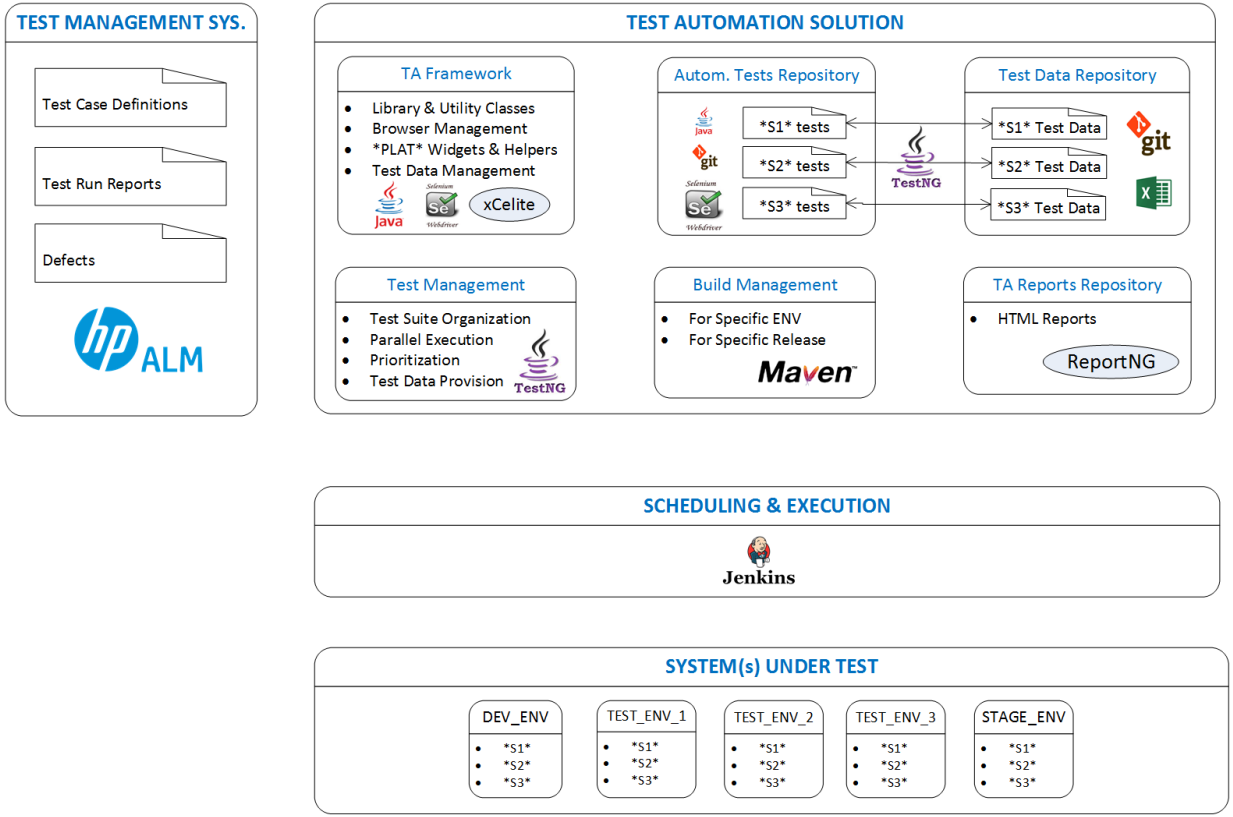
DO T.A. RIGHT

4. Automated "test automation"

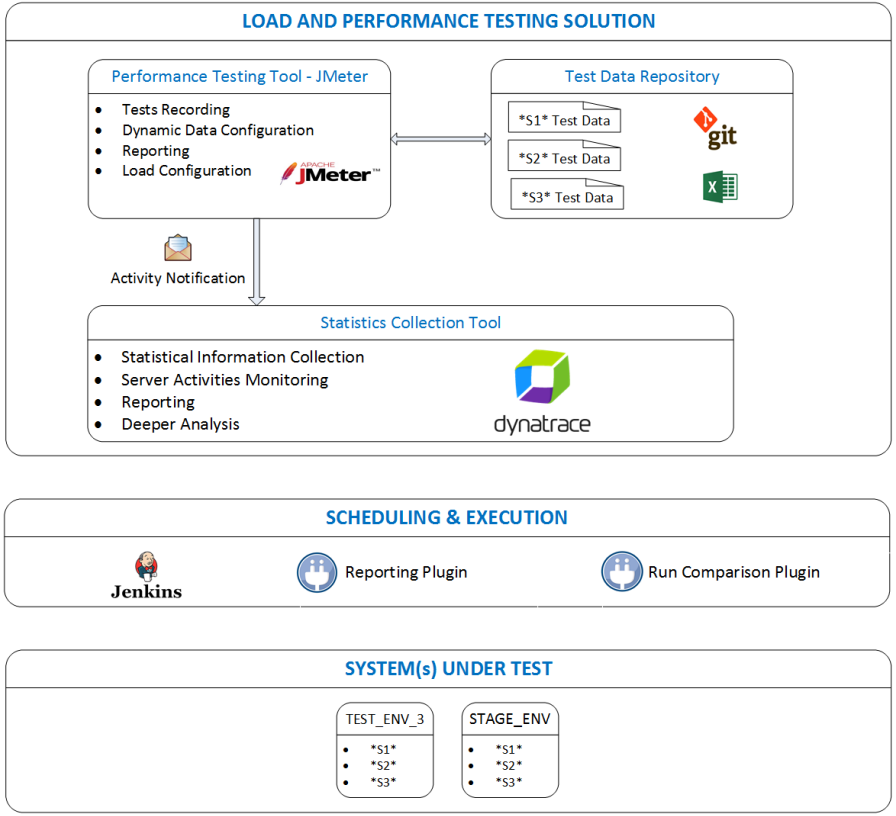
- treat it as a software development initiative:
  - start with the end in mind
  - architecture
  - code reviews



AUTOMATED TEST AUTOMATION



TEST AUTOMATION SOLUTION – ARCHITECTURE



5. MEASURE AND COMMUNICATE

MEASURE AND COMMUNICATE

Track and provide transparency on:

- Results achieved
- Time spent
- Work done
- Resources used

…regularly – on Sprint/Release basis, but at least once in 2 or 3 weeks.

MEASURE AND COMMUNICATE

Sample KPIs for Execution, Results Analysis and Reporting:

- Number **Test Suite runs**
- Number **Test runs** (and % of successful runs)
- Number **Executed "User actions"**(→ man-hours saved)
- Percentage "Test Run Results" **reports delivered in time**
- Number Automated Test cycle time

MEASURE AND COMMUNICATE

Sample KPIs for New Tests Design & Implementation:

- Percentage — Sprint **regression TA goal achieved** (X out of Y)
- Number — **New tests automated** + “user actions”
- Number — **New reusable components** created
- Number — **New Smoke tests** automated
- Number — **Test Cases designed** for automation (and %)

MEASURE AND COMMUNICATE

Sample KPI for Maintenance:

- Number Existing tests adapted/maintained

Framework/Tooling Enhancement:

- < highlights on framework enhancements implemented >

MEASURE AND COMMUNICATE

Sample KPIs for Time spent (quantity & percentage) on:

- run results analysis and reporting
- new tests implementation
- test maintenance
- framework/tooling enhancement
- other – test data preparation, process optimization, etc.

MEASURE AND COMMUNICATE

KPIs on Doing Test Automation Right (continuous improvement)

- **track progress** on achieving the goals in your prioritized list
- **track "health" status** of all items already achieved (as they tend to deteriorate if not taken care of)

MEASURE AND COMMUNICATE

The regular TA Status Report should be:

- short, but comprehensive – 2-3 slides
- addressed to and discussed with key stakeholders:

  - Project Manager / Budget Owner

  - Test Manager

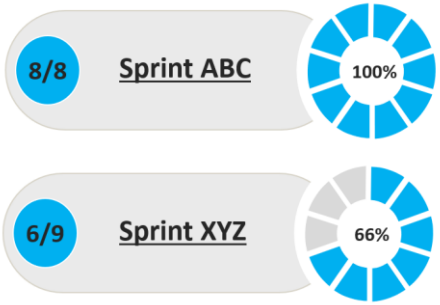
  - Scrum Master / Dev Team Leader

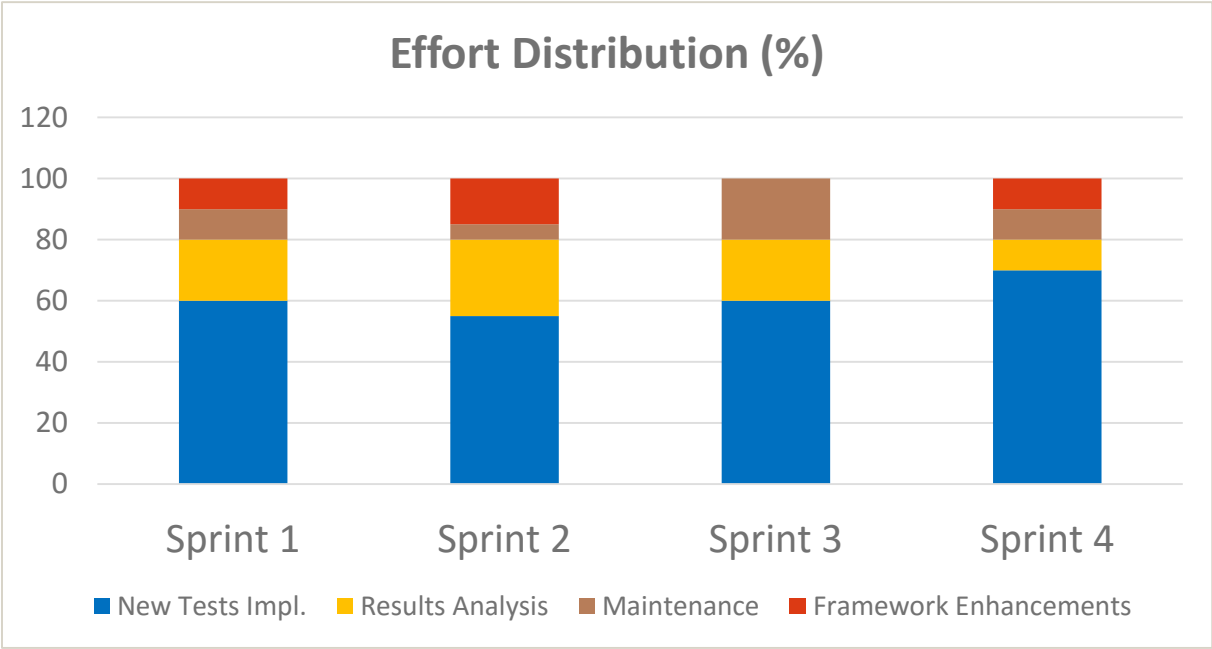
  - Product Owner / Lead BA / Product Manager

MEASURE AND COMMUNICATE

TA Status Report – sample elements

Sprint TA Scope DONE





MEASURE AND COMMUNICATE

TA Status Report – sample elements

ID	Goal / KPI	Status	Details
1	Test results are analyzed, and reports are distributed in time		2 of 31 test run reports have been delayed
2	Existing tests are maintained and up-to-date		All are up-to-date currently.
3	TA scope for the passed sprint is defined and tests are described		3 out of 9 tests are still pending description
4	Initiative: “Configurable environment-specific runs”		No progress has been made during this sprint due to vacations and sick leaves
…	…		…

MEASURE AND COMMUNICATE

Number of Tests in each Suite – and number of User Actions automated with each test suite

	Test Suite	Number of Tests	User Actions	Notes
1	[illegible]	15	763	[illegible]
2	[illegible]	9	151	[illegible]
3	[illegible]	17	629	[illegible]
4	[illegible]	1 (860)	12 040	[illegible]
5	[illegible]	57	3 021	[illegible]

MEASURE AND COMMUNICATE

Tests scheduled and run regularly on the different environments:

- daily – run each night
- weekly – run each Saturday night

	Test Suite	DEV_ENV	TEST_ENV-1	TEST_ENV-2	STAGE_ENV
1	[illegible]	daily	daily	daily	weekly
2	[illegible]	daily	daily	daily	weekly
3	[illegible]	daily	daily	daily	*
4	[illegible]	N / A	N / A	daily	weekly
5	[illegible]	< ad hoc >	< ad hoc >	daily	weekly

MEASURE AND COMMUNICATE

Number of Tests and User Actions automatically executed DAILY

	Test Suite	Tests	User Actions	Daily Runs	Total Test Run	Total User Actions Run
1	[illegible]	15	763	3	45	2 289
2	[illegible]	9	151	3	27	453
3	[illegible]	17	629	3	51	1 887
4	[illegible]	860	12 040	2	1 720	24 080
				Total:	1 843	28 709

MEASURE AND COMMUNICATE

Number of full Test Suite runs to date (statistics through Jenkins by 14.07.2016) – incl. nightly and ad-hoc runs

	Test Suite	DEV_ENV	TEST_ENV-1	TEST_ENV-2	STAGE_ENV	Total
1	[illegible]	176	133	65	< TBC >	**374**
2	[illegible]	45	13	15	< TBC >	**73**
3	[illegible]	154	8	8	N / A	**170**
4	[illegible]	N / A	N / A	60	< TBC >	**60**
5	[illegible]	389	391	119	99	**998**

#6 “BONUS” SECTION

PRIORITIZE AND DECIDE WHAT TO AUTOMATE

Parameters for **Categorization of User Stories**:

- intensity of use / number of users - on a scale of 1 to 3
- importance/criticality of the features - on a scale of 1 to 3

Parameters for **Categorization of Test Cases** covering a User Story:

- being appropriate for automation - Yes / No
- effort/time needed to execute manually - on a scale of 1 to 3 or man-days
- need for re-execution within a cycle (e.g. different input data) - on a scale of 1 to 3
- will be included in different test suites:
    - Smoke Test Suite - Yes / No
    - Regression Test Suite - Yes / No
    - Acceptance Test Suite - Yes / No
- complexity for implementing as automated test - on a scale of 1 to 3
- effort needed for implementation as automated test - in man-hours / man-days
- will automating it provide options for economy of scale? - Yes / No
  (will it allow its new components to be reused in other tests)

HOW – DURING

1	**TAKE IT SERIOUSLY!** (LONG-TERM INITIATIVE BY ITS OWN)
2	**TA-SPECIFIC KPIS + MEASURABLE & REALISTIC GOALS**
3	**REUSABLE COMPONENTS**
4	**"AUTOMATED" TEST AUTOMATION**
5	**MEASURE AND COMMUNICATE**
6	**PRIORITIZE / DECIDE WHAT TO AUTOMATE**

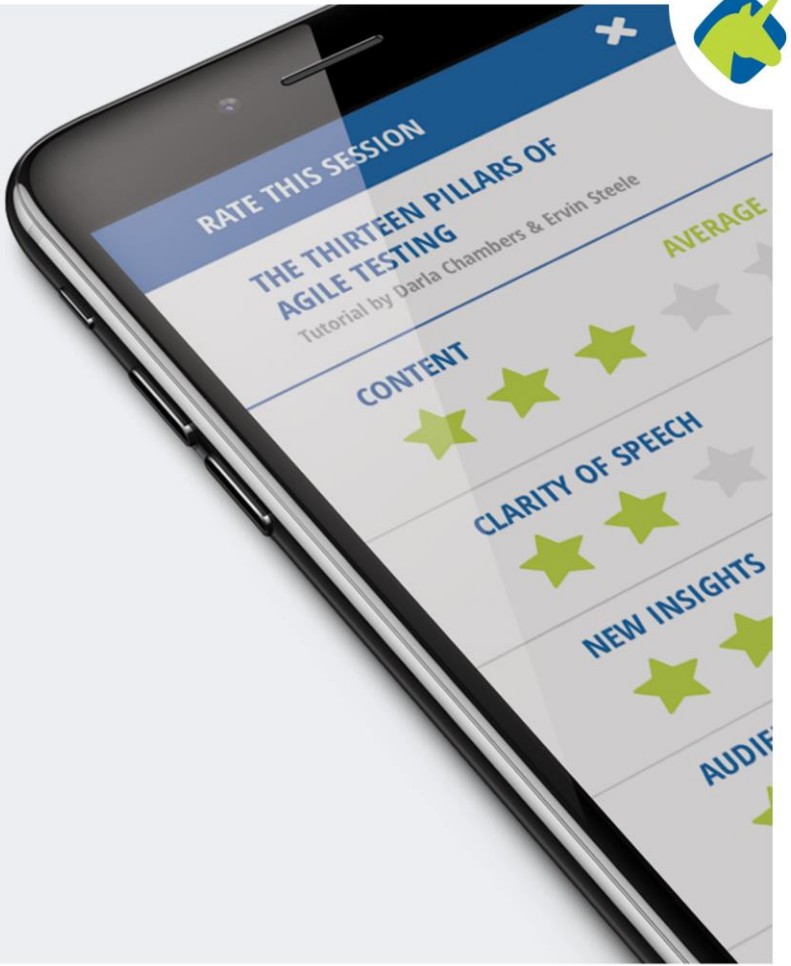
HOW – DURING

7	PROCESS & RESPONSIBILITIES
8	MAKE THE MOST OF TA (EXECUTION) RESULTS
9	TRACEABILITY BETWEEN ARTIFACTS
10	DESIGN FOR AUTOMATION
11	CROSS-FUNCTIONAL TEAM MEMBERS INVOLVEMENT

Thanks for your attention!

Feedback welcome!

Go to **agiletestingdays.com/session-ratings**
and give your rating!



VIELEN DANK FOR YOUR ATTENTION!

Iancho Dimitrov

iancho.d@gmail.com

linkedin.com/in/iancho

@iandim

<BACKUP SLIDES>

DO TEST AUTOMATION RIGHT

Set up Process & Responsibilities for:

- planning & case design → technical design → implementation
- run → results analysis → reporting & defect tracking
- **(!)** proactive test maintenance
- framework/tooling enhancement

DO TEST AUTOMATION RIGHT

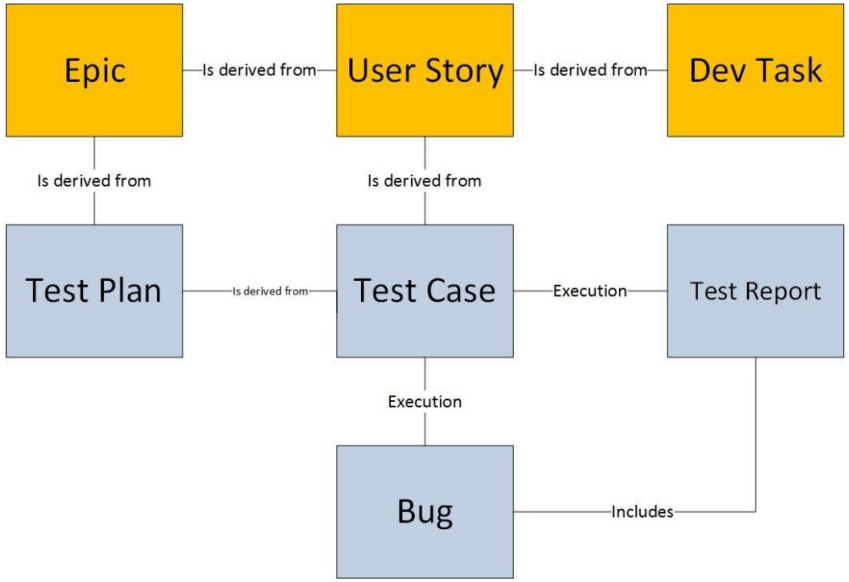
Make the most of TA (execution) results

- distribute run results to all relevant team members (mailgroups)
- meaningful descriptions – incl. screenshots, input data, logs
- regularly update all team members on progress on key TA goals:
  - could be once in 3, 4 or 6 months
  - semi-formal gathering (e.g. a Lunch&Learn/Brown-bag session)
  - gather feedback

DO TEST AUTOMATION RIGHT

Assure links & traceability between artifacts:

- User Story / Feature
- Test Case
- Test Run Reports
- Defects



DO TEST AUTOMATION RIGHT

Design for automation:

- the right test cases
- designed for automated testing
- decomposed to the needed level of reusable components

(vs. automating tests designed for manual testing)

DO TEST AUTOMATION RIGHT

Cross-functional team members involvement in TA

- Developers – assuring TA-friendly software
- BA/PO/PMs – design & prioritization, esp. “what’s next”
- Test Management – integration of TA in the QA Strategy & Plan

…and vice versa - TA Engineers – biz domain; features, requirements

HOW - AFTER

- **Do Test Automation Right** - so many things to address:
  - make a goals list – prioritized
  - reap the low-hanging fruit (things mostly within your control)
  - work on achieving the goals, track progress

- **Measure** | **Communicate** | **Optimize**

DO TEST AUTOMATION RIGHT!

- Don’t expect to achieve everything quickly
- Make a custom list for your initiative – and prioritize it
- Reap the low-hanging fruit
- Set deadlines, track progress – and work on these one step at a time

THANK YOU ONCE AGAIN ☺!

Iancho Dimitrov

iancho.d@gmail.com

LinkedIn.com/in/iancho