COMAQA.BY

QA automation community

How to Draw QA Architecture Automation in Few Slight Movements

Anton Semenchenko

About

Anton Semenchenko

- QA community COMAQA co-founder
- C++ community CoreHard co-founder
- “International IT” community InterIT co-founder
- 50+ international conferences organizer
- IT evangelist (300+ speeches on meetups and conferences)
- Field of scientific interest: using math for differential diagnostic in cardiology, psychiatry, virology and immunology
- C++ books editor
- QA Automation, QA Management, Low level development, Agile trainer
- ICAgile Certified Professional

How to Draw QA Architecture Automation in Few Slight Movements: goals

1. Show that **visual representation** of framework makes it easier to support the project, provide knowledge transfer and implement new features;
2. Provide a ready-to-go example of **how to draw architecture in few minutes** (for UI project, as example);
3. Demonstrate that **“Architecture” is not frightening** as it may sound like;
4. Show you **hints of drawing** schemes at interviews, knowledge-transfers, etc…

QA Automation Architecture as a technical challenge

1. Main challenge – **find evolutionary changing balance point**;
2. Solution should **not be too complicated**, or **too simple**;
3. **Finding that balance point which is changing from time to time - is the most complicated part of architecture**;
4. When balancing is low in quality - **business will financially suffer**;
5. Tech discussion about project **hypothetical growth** are always futile;
6. And tech understanding may be different from business needs – results in monstrously complex project;
7. **Solving** “**balancing Architecture challenge**” - by using **visualization** and **focusing**.

COMAQA.BY
QA automation community

Legend

Entity

Note

🟥⟷
🟦⟷
🟩⟷

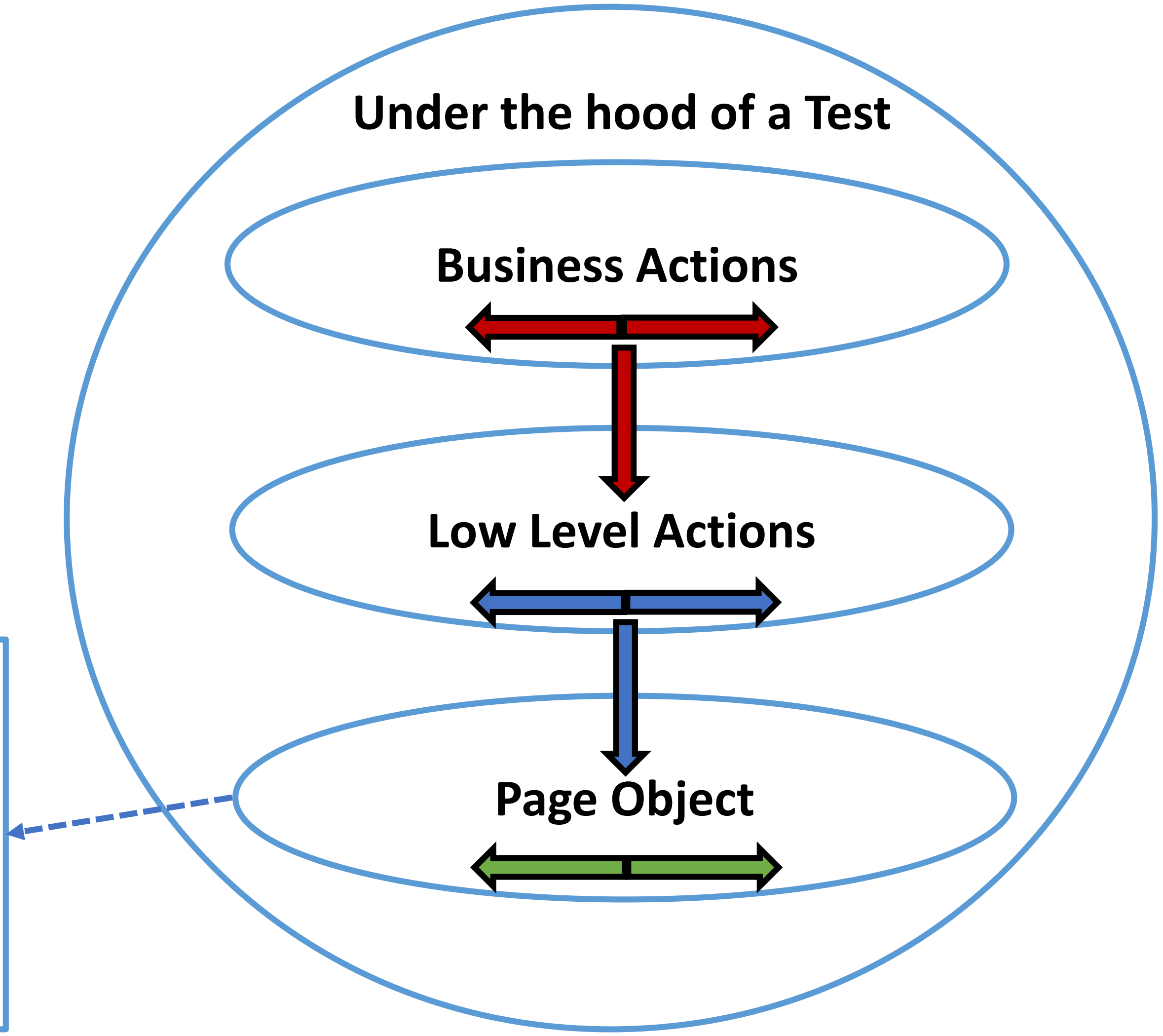
<u>Color</u> == level \ frequency of interactions inside one layer

→

<u>Direction</u> == use
<u>Bi-direction</u> == equal
<u>Weight</u> == frequency or probability of interactions
<u>Dashes</u> == potential relations

? — Topic that require additional investigation, and can be covered via conversation at community stand

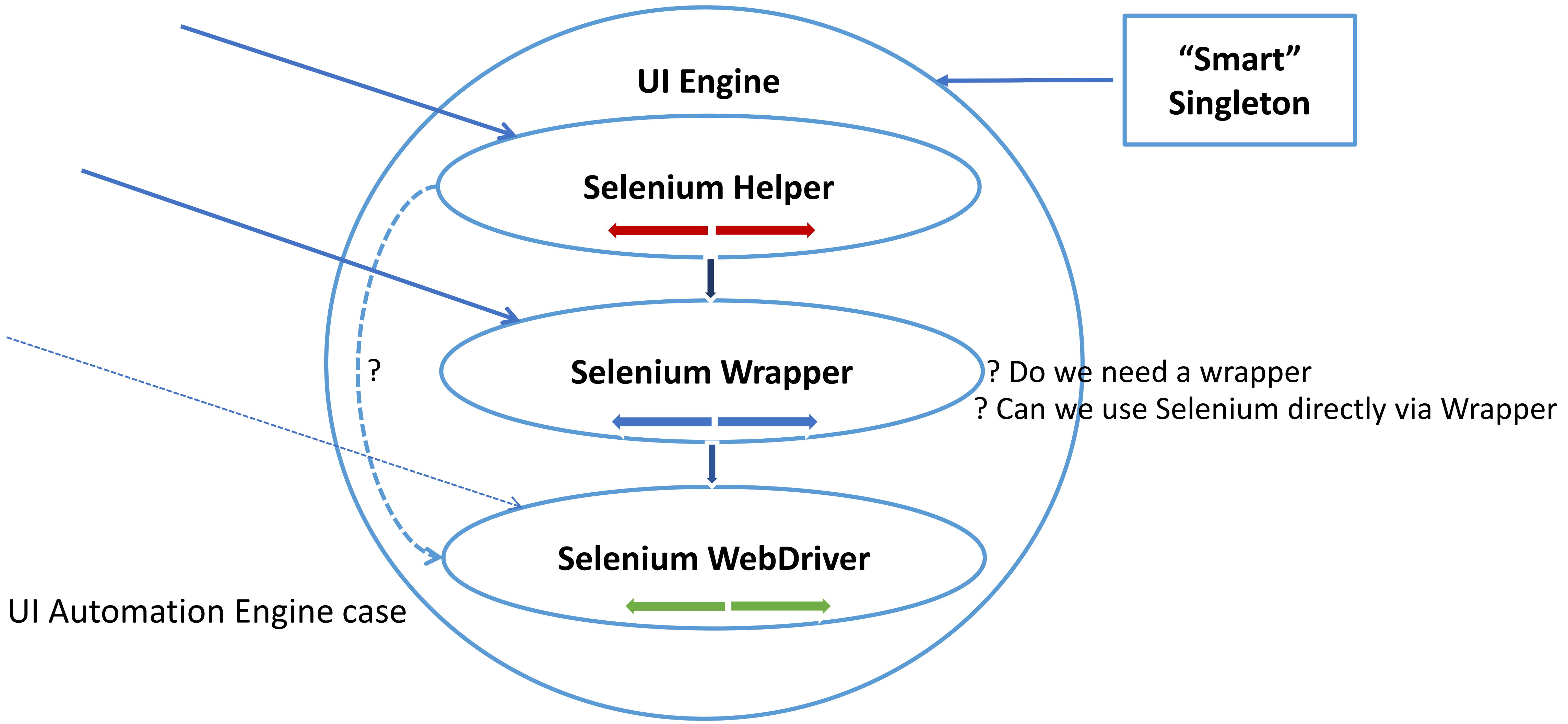
3 main layers & external data source with locators



External data source with locators, for example:

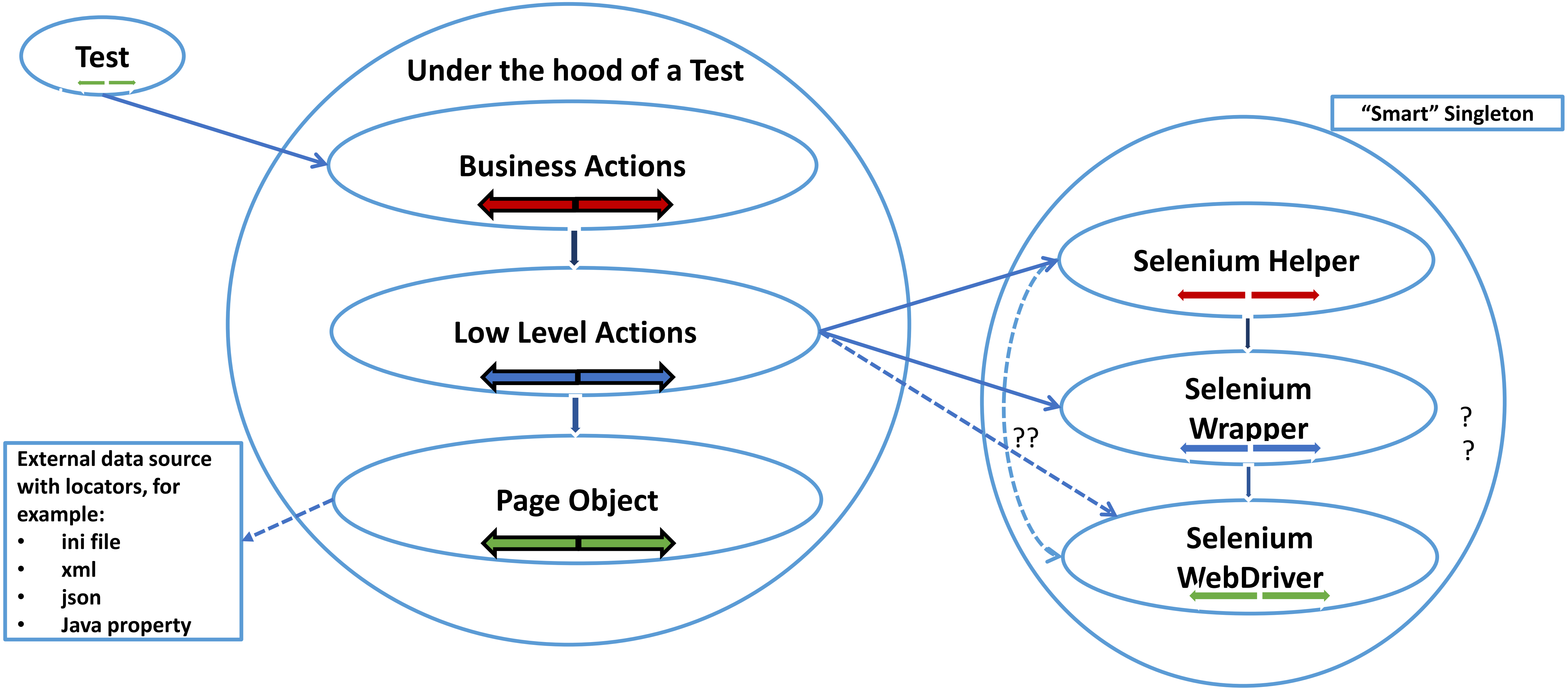
- **ini file**
- **xml**
- **json**
- **Java property**

3 main layers & any UI Engine



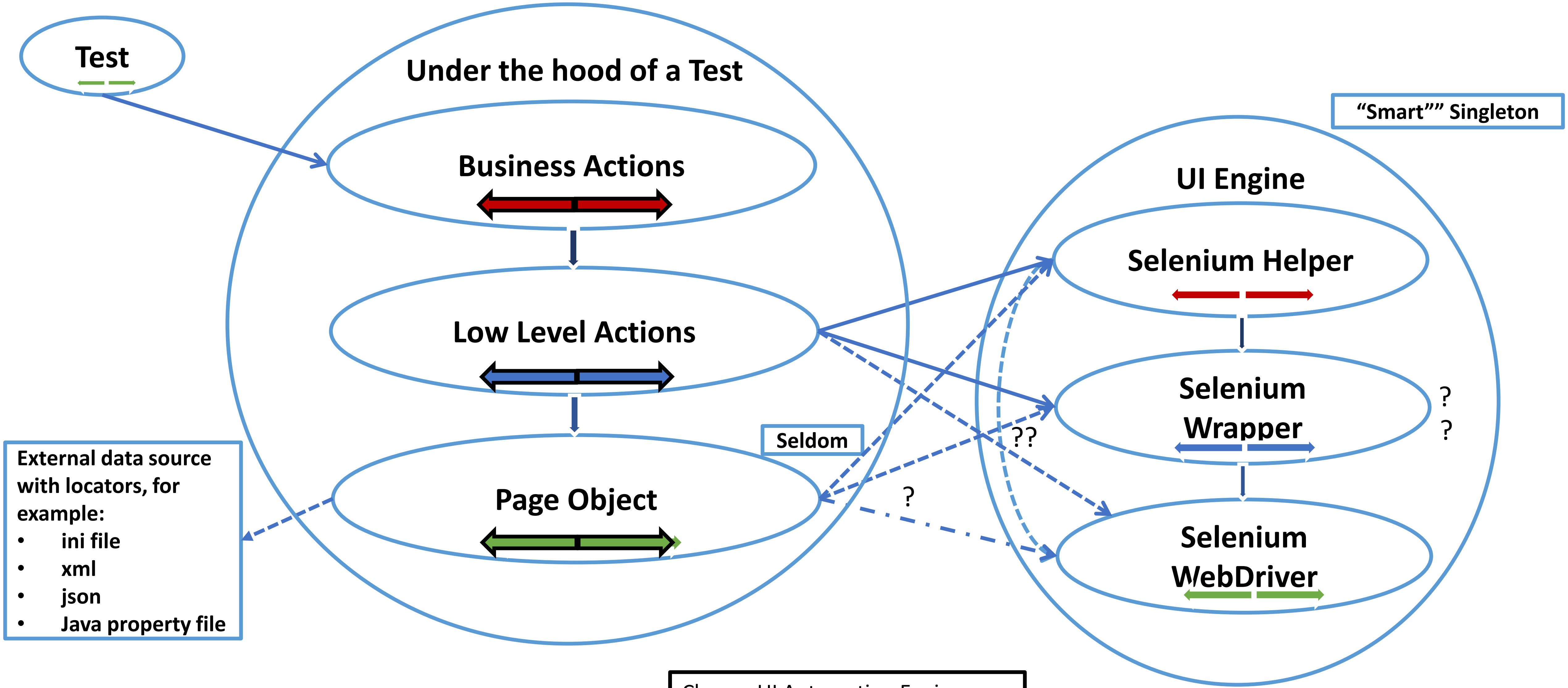
Change UI Automation Engine case

3 main layers & any UI Engine
Option 1: simpler => preferable



External data source with locators, for example:
- ini file
- xml
- json
- Java property

3 main layers & any UI Engine; Sption 2: more complicated - Yandex WebElements based



Change UI Automation Engine case

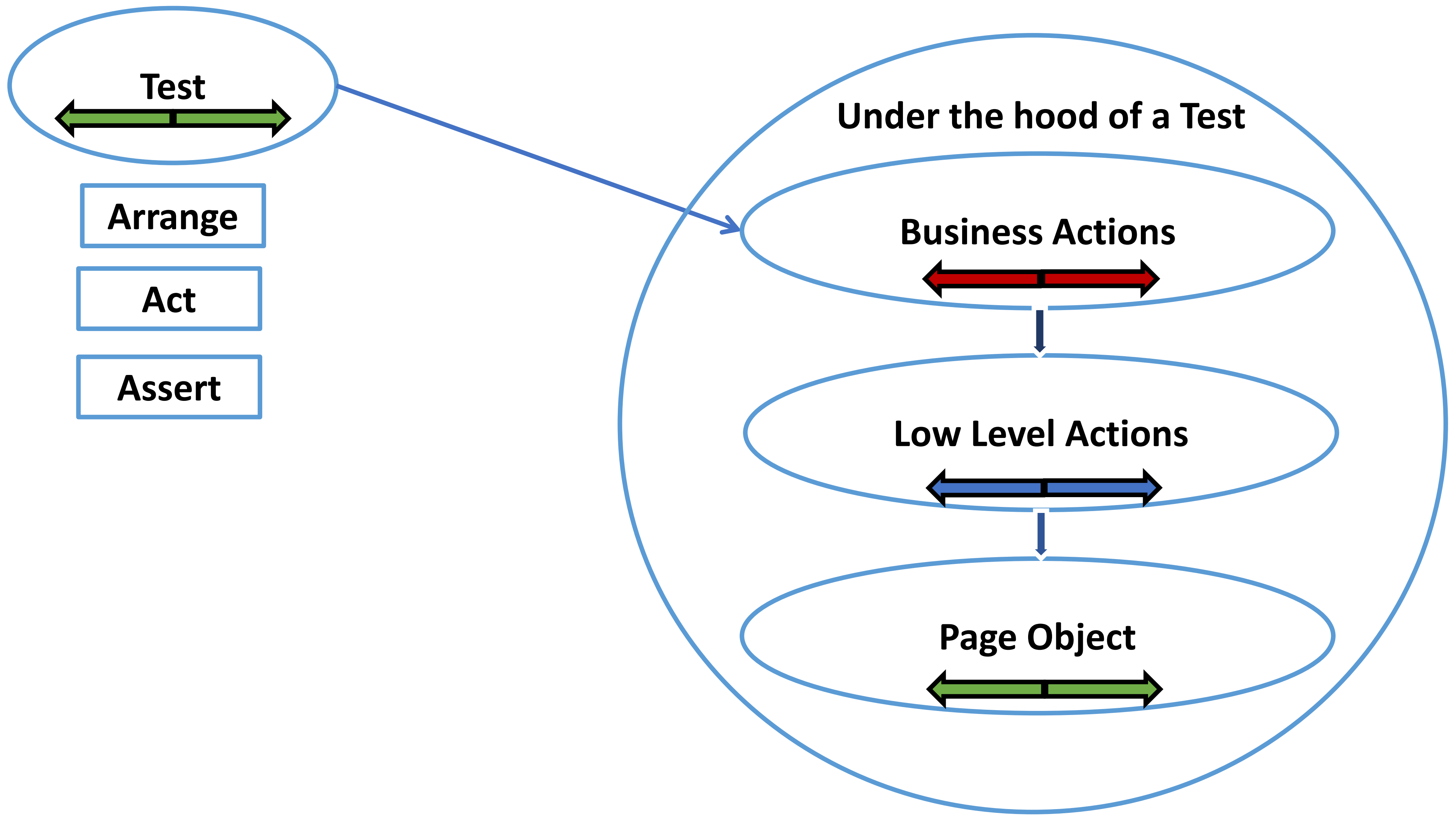
3A Rule
Arrange, **A**ct, **A**ssert test structure rule

Basically tests consist of 3 steps:

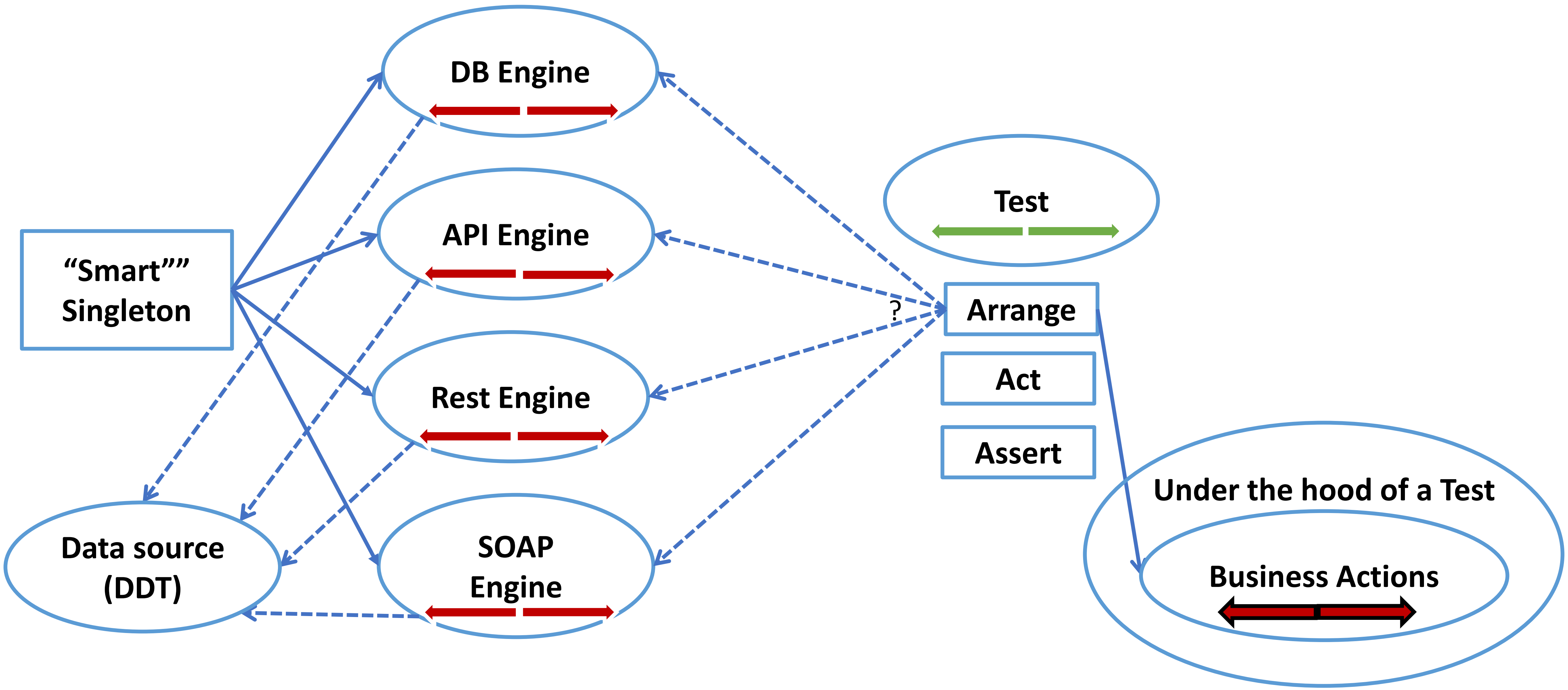
- **A**rrange - establishing the test environment;
- **A**ct - executing the logic;
- **A**ssert - verifying the expectation.

Test & 3A Rule

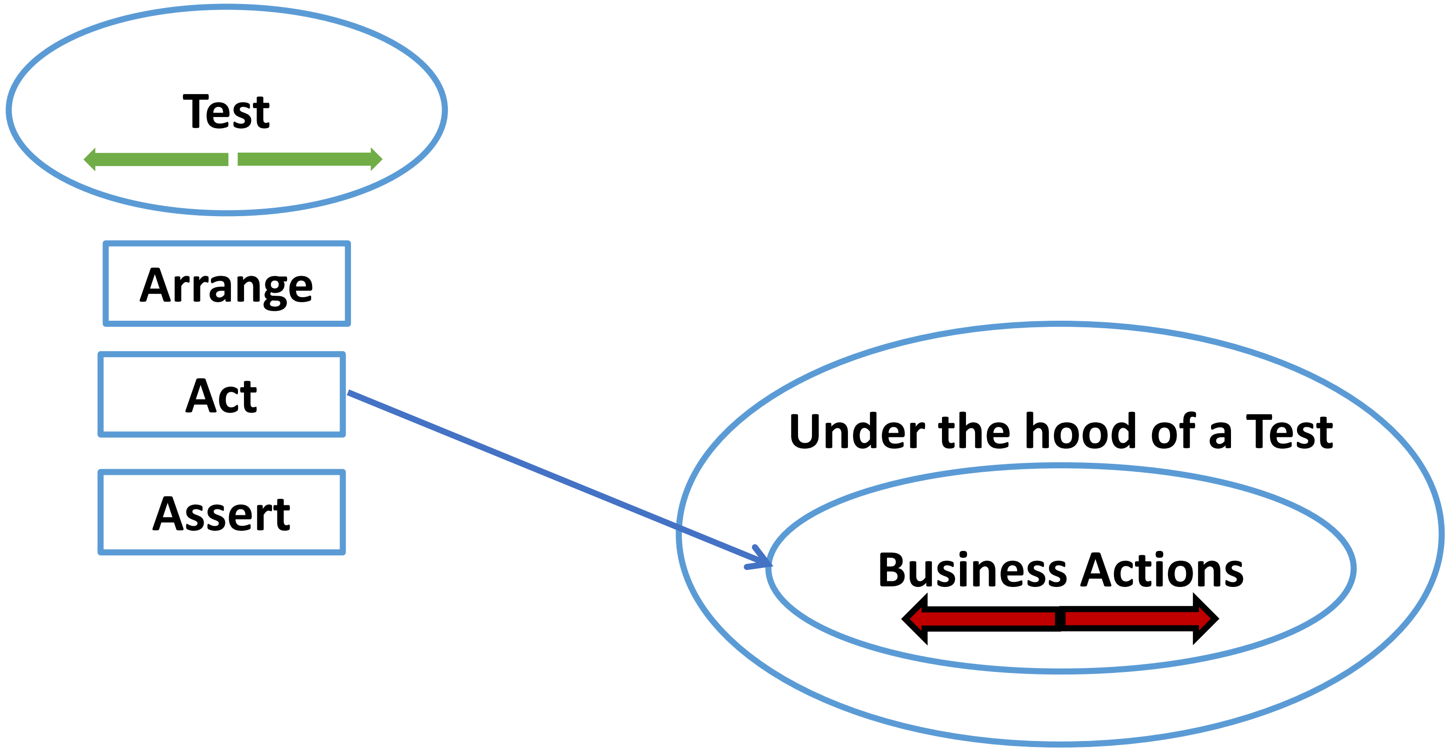
Arrange, Act, Assert test structure rule



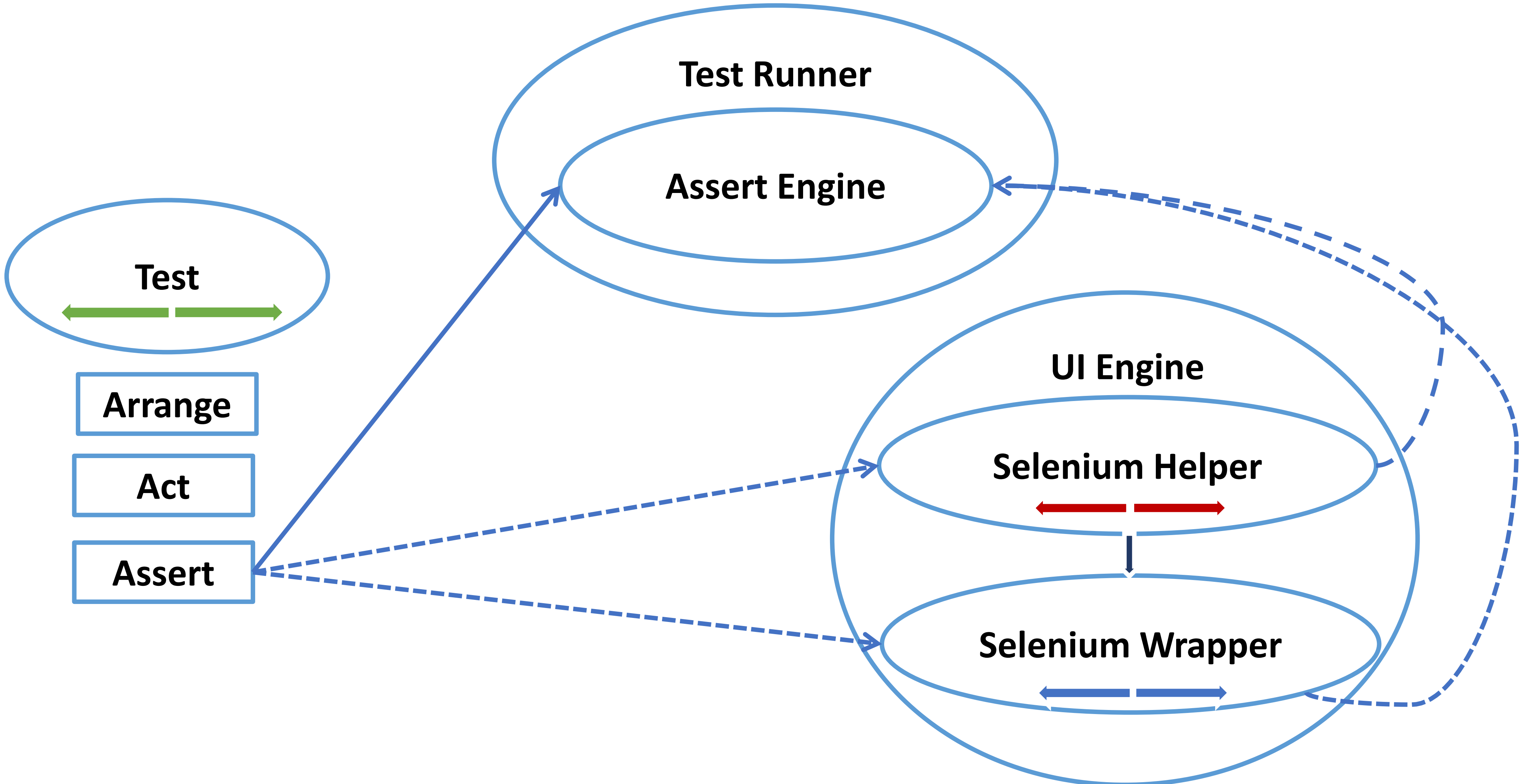
Test & 3A Rule - Arrange



Test & 3A Rule - Act

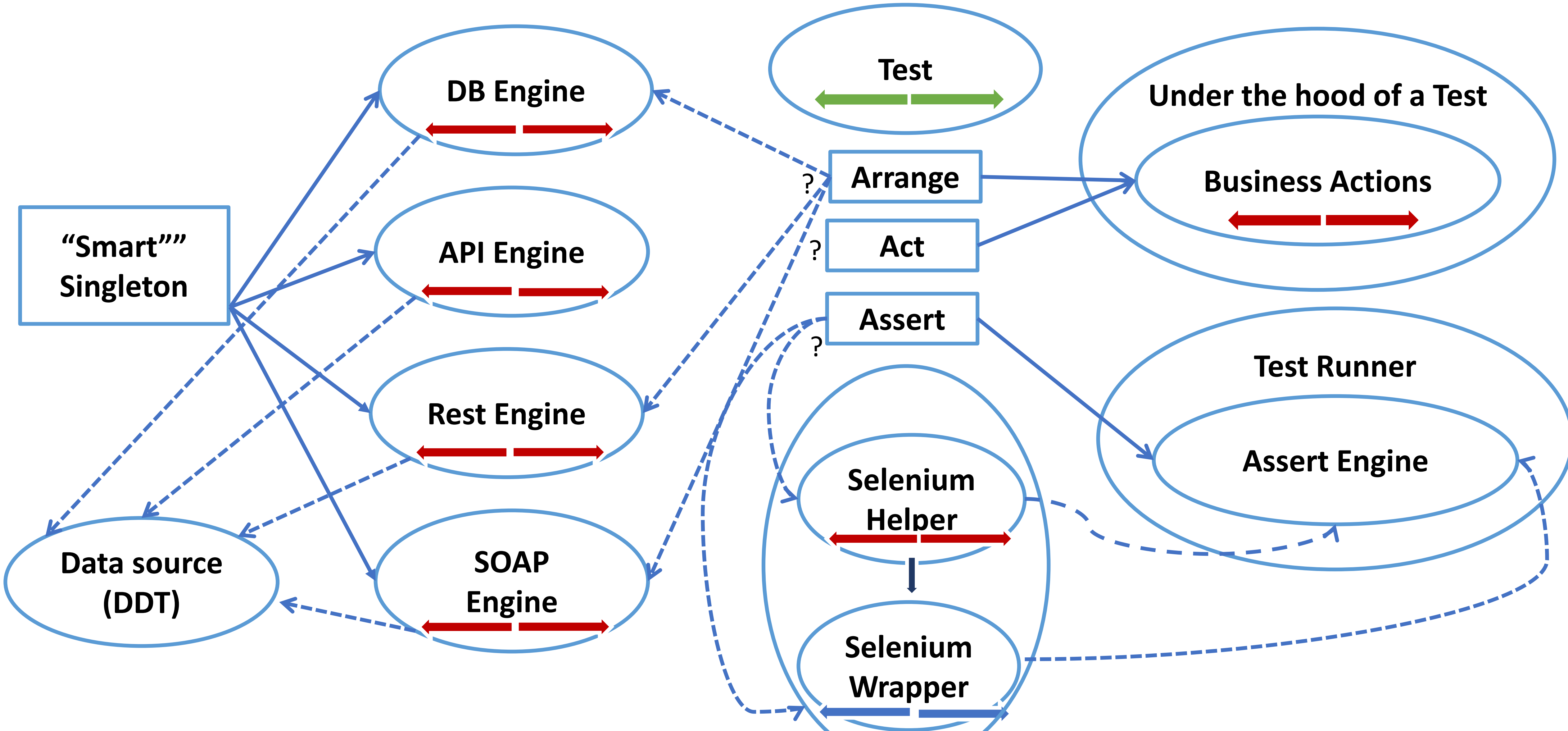


Test & 3A Rule - Assert

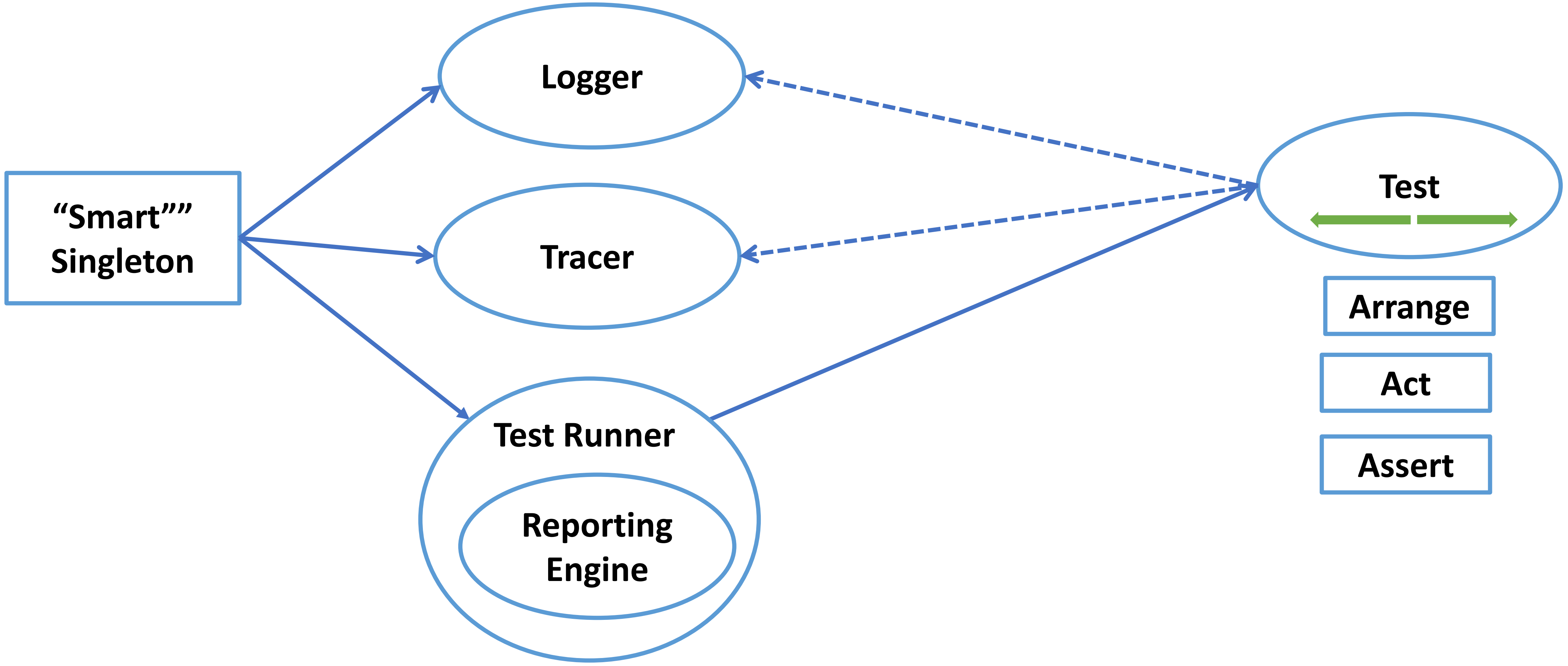


COMAQA.BY
QA automation community

Test & 3A Rule: Arrange, Act, Assert

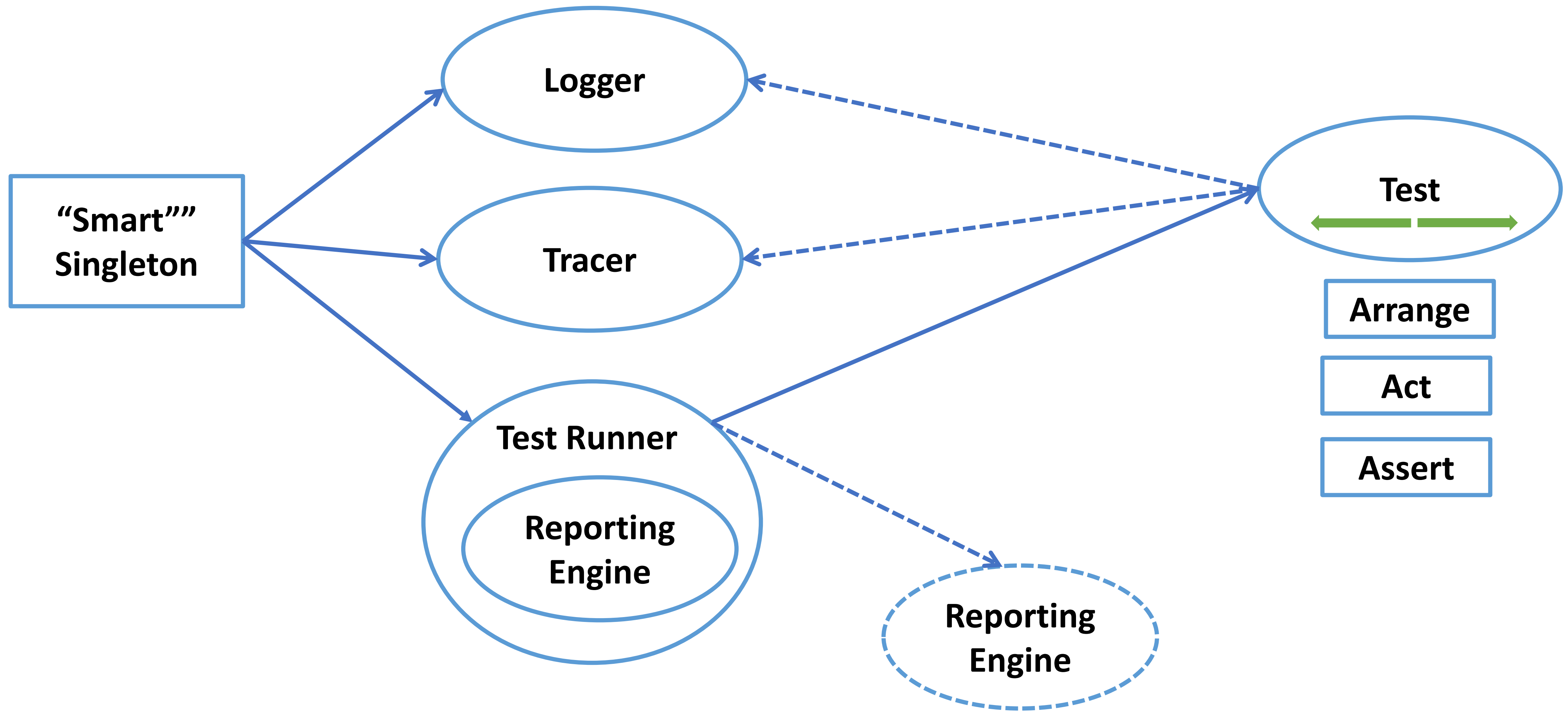


Test & “Global” Entities \ “Smart Singleton”

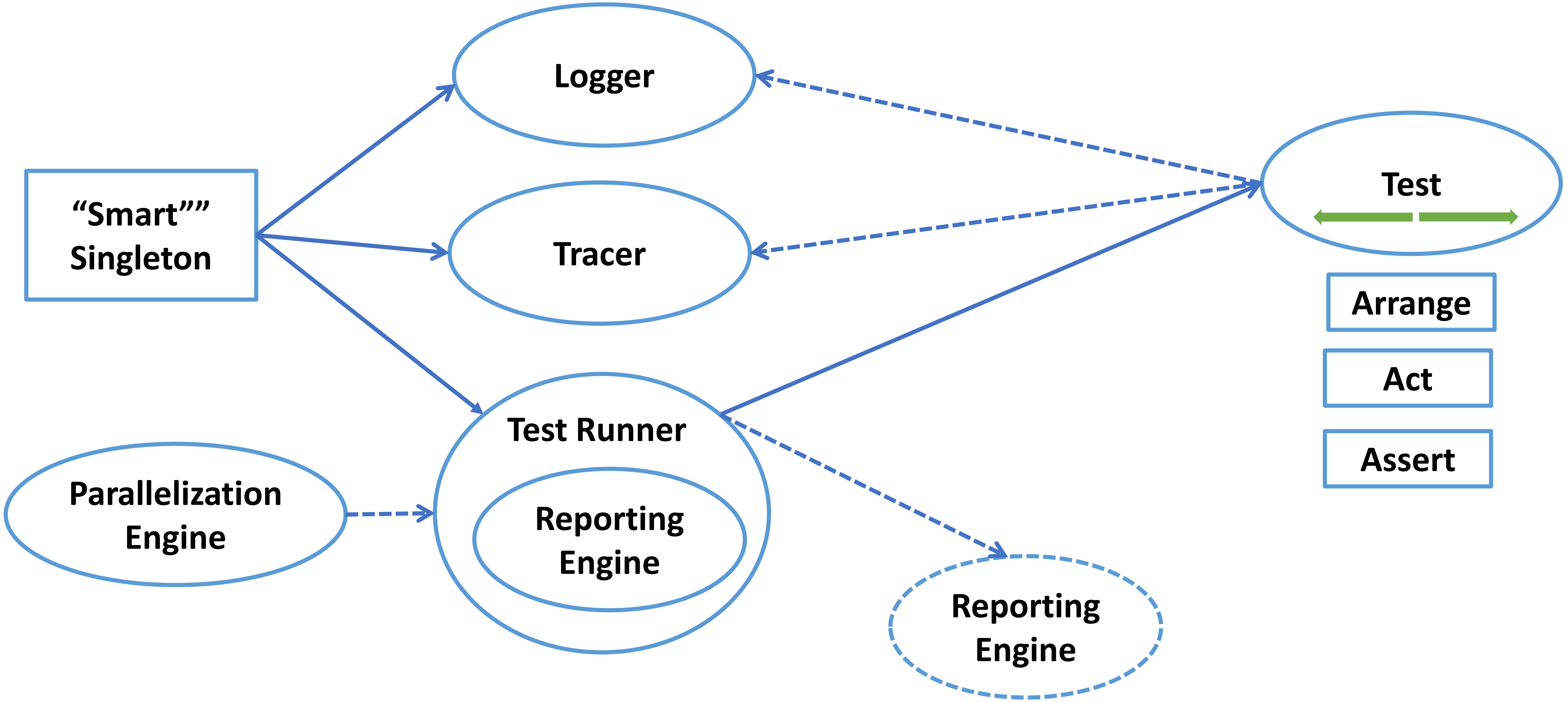


COMAQA.BY QA automation community

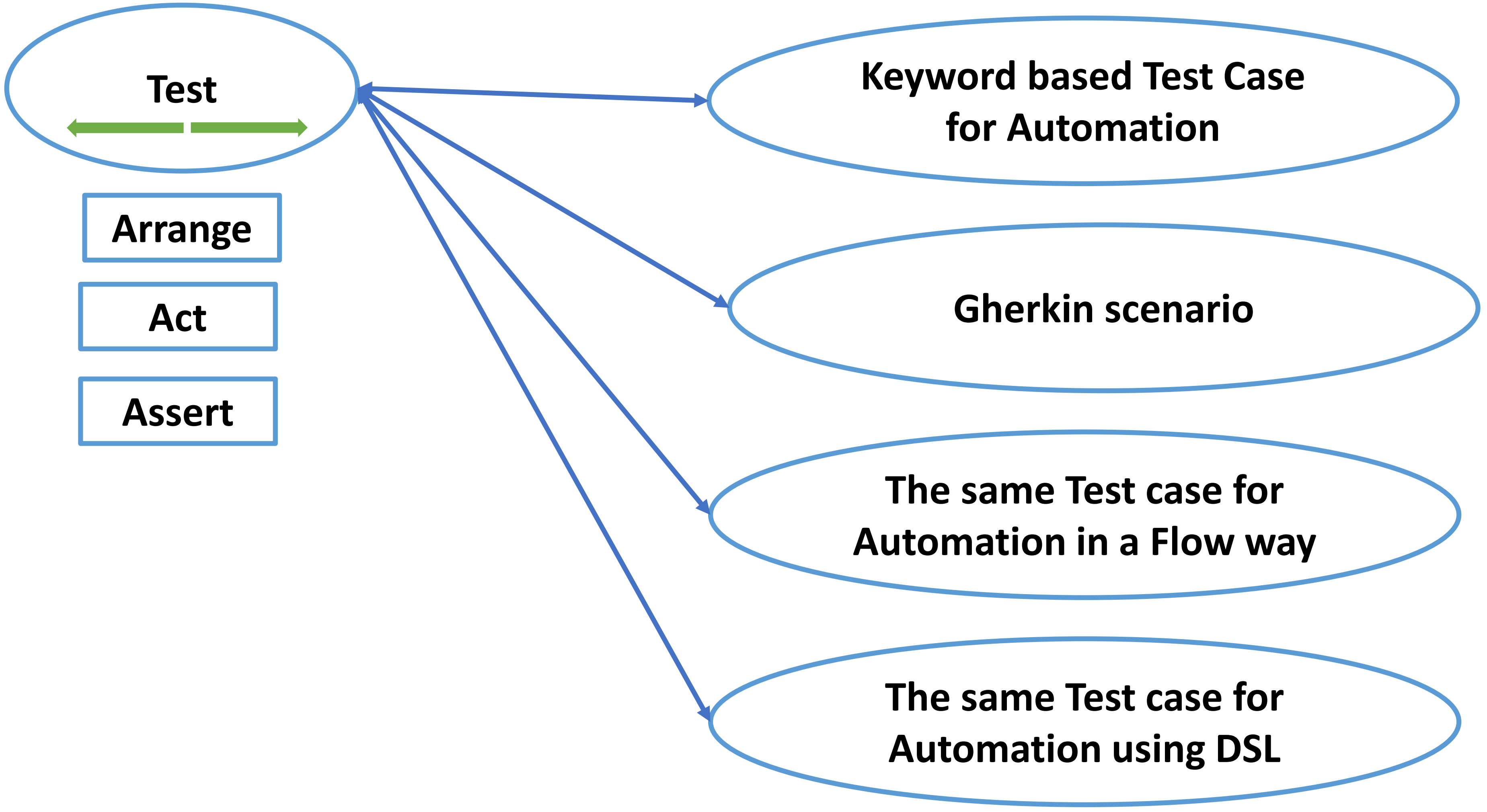
Test & “Global” Entities \ “Smart Singleton”



Test & “Global” Entities - run test in parallel



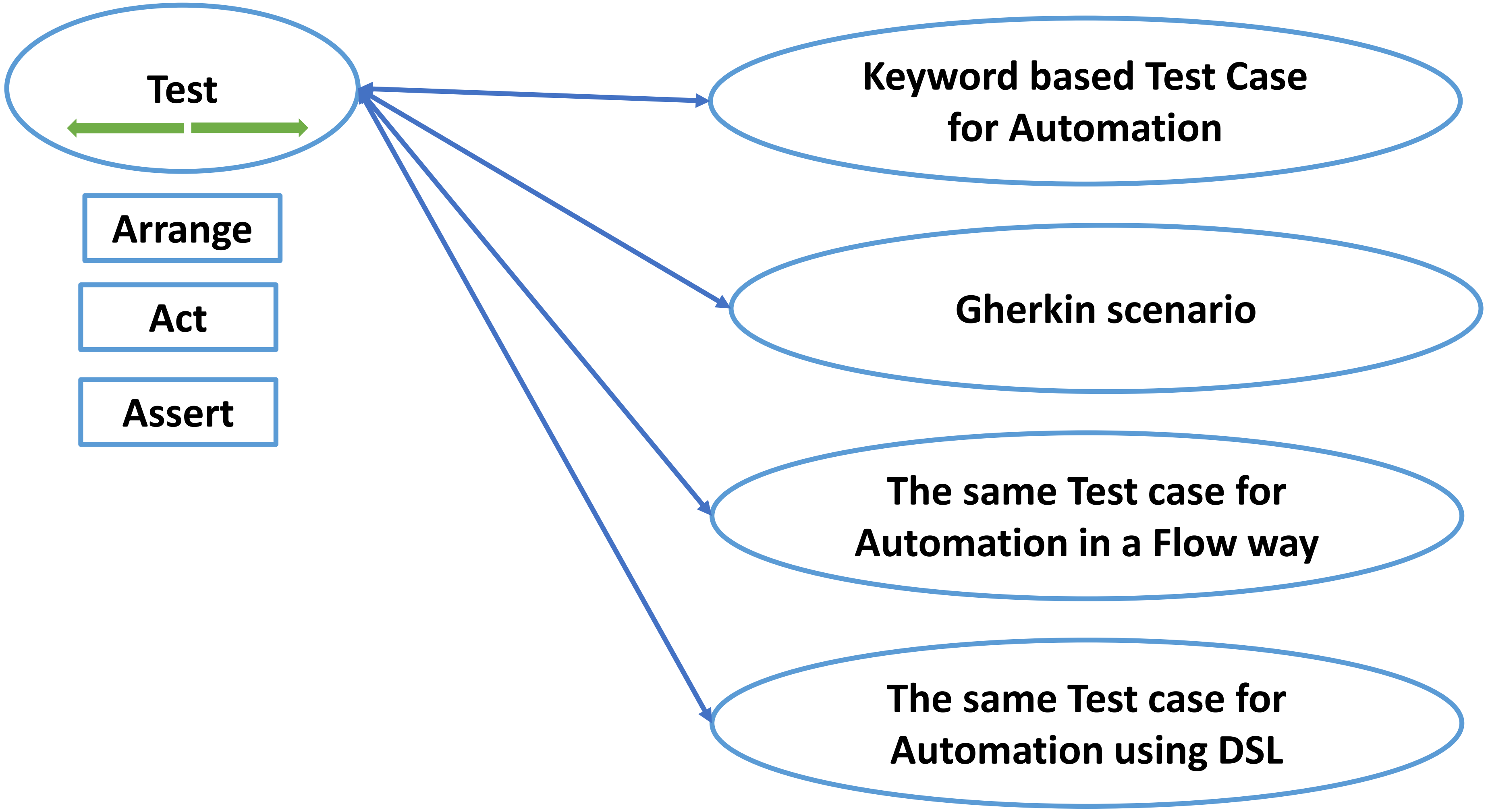
Test level mapping: 4 examples of entities mapping in one


Test
Arrange
Act
Assert
Keyword based Test Case for Automation
Gherkin scenario
The same Test case for Automation in a Flow way
The same Test case for Automation using DSL

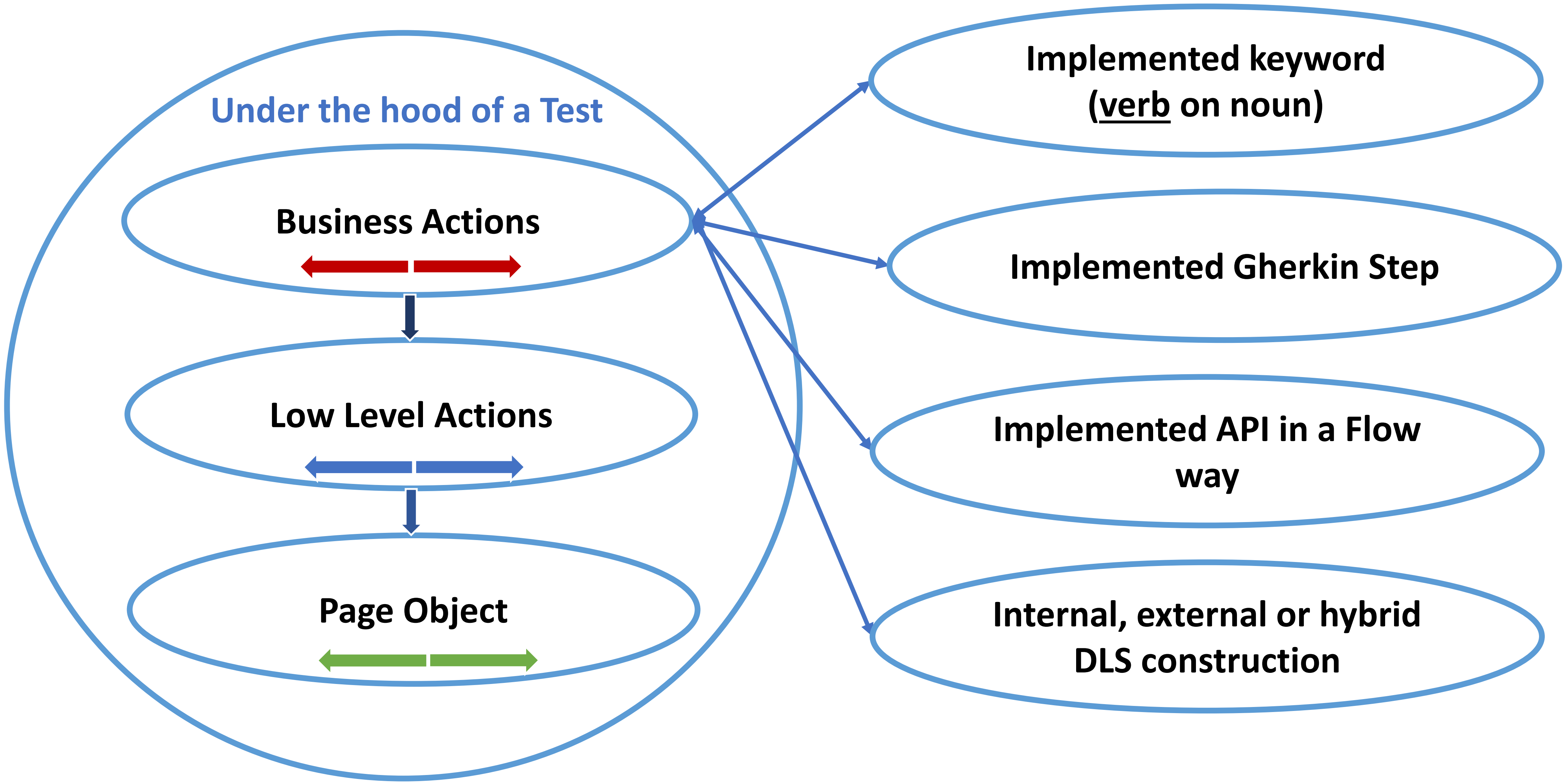
QA Automation approaches Keyword, BDD, Fluent interface, DSL

1. Ubiquitous language
2. Key word driven QA Automation
3. Behavior Driven Development (BDD) approach – as a special case of DSL based QA Automation solutions
4. Domain specific language (DSL) based QA Automation
5. Flow – as a way of implementation DSL\BDD
6. Flow- fluent interface is an implementation of an object oriented API that aims to provide more readable code.

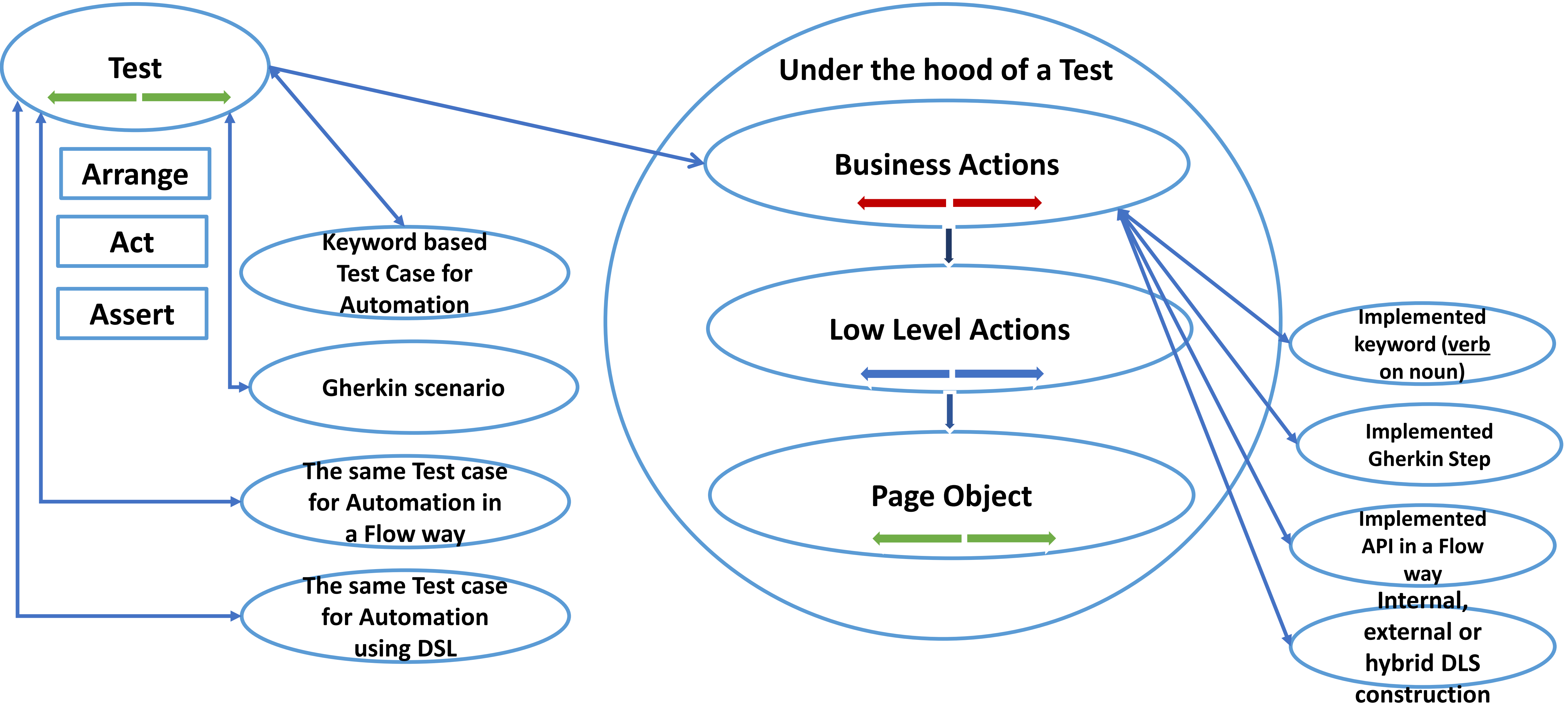
Test level mapping: 4 examples of entities mapping in one


Test
Arrange
Act
Assert
Keyword based Test Case for Automation
Gherkin scenario
The same Test case for Automation in a Flow way
The same Test case for Automation using DSL

Business Action level mapping: 4 examples of entities mapping in one


Under the hood of a Test
Business Actions
Low Level Actions
Page Object
Implemented keyword (verb on noun)
Implemented Gherkin Step
Implemented API in a Flow way
Internal, external or hybrid DLS construction

Test & 3A Rule: 4 examples of entities mapping in one


Test
Arrange
Act
Assert
Keyword based Test Case for Automation
Gherkin scenario
The same Test case for Automation in a Flow way
The same Test case for Automation using DSL
Under the hood of a Test
Business Actions
Low Level Actions
Page Object
Implemented keyword (verb on noun)
Implemented Gherkin Step
Implemented API in a Flow way
Internal, external or hybrid DLS construction

COMAQA.BY
QA automation community

Conclusion: 4 results of the presentation for technical specialists

1. visual representation makes it easier to understand the project when you new to it;
2. show you that architecture is not as frightening as it may sound like;
3. helpful tip for any interview – draw simple project’s diagram;
4. provided you with the presentation, which can be used as a cheat sheet.

COMAQA.BY
QA automation community

Conclusion: 3 results of the presentation for Business

1. solve “**balancing Architecture challenge**” – by creating diagrams;
2. avoid potential issues of **over complexing or over simplification** of the system;
3. **completing those saves money for our partners and increase quality of the final product.**

COMAQA.BY
QA automation community

Contact Information

Anton Semenchenko
Email: semenchenko_anton_v@tut.by
Skype: semenchenko_anton_v
Telegram: +375 33 33 46 120
Phones: +375 33 33 46 120 & +375 44 74 00 385
https://www.facebook.com/semenchenko.anton.v
https://www.linkedin.com/in/anton-semenchenko-612a926b/
https://twitter.com/comaqa

COMAQA.BY
QA automation community

Thank you for you attention!

Bid you farewell!

http://conference.comaqa.by/

https://comaqa.by/

https://www.youtube.com/channel/UCzAhXR53eIvHht9qmFPBVxg

QA Automation approaches Keyword, BDD, Fluent interface, DSL

- Domain specific language (DSL) based QA Automation
- Flow – as a way of implementation DSL\BDD
- Flow- fluent interface is an implementation of an object oriented API that aims to provide more readable code.
- A fluent interface is normally implemented by using method cascading (concretely method chaining) to relay the instruction context of a subsequent call (but a fluent interface entails more than just method chaining).
- Generally, the context is defined through the return value of a called method self-referential, where the new context is “equivalent” to the last context terminated through the return of a void context.

QA Automation approaches
Fluent interface by Martin Fowler

- “Building a fluent API like this leads to some unusual API habits.”
- “One of the most obvious ones are **setters that return a value**.”
- “The common convention in the curly brace world is that modifier methods are void, which I like because it follows the principle of CommandQuerySeparation. This convention does get in the way of a fluent interface, so I'm inclined to suspend the convention for this case.”
- “You should choose your return type based on what you need to continue fluent action.”
- “The key test of fluency, for us, is the Domain Specific Language quality. The more the use of the API has that language like flow, the more fluent it is”

COMAQA.BY
QA automation community

QA Automation approaches
DSL – Domain Specific Language

- DSL = Domain (ether technical or business … or both – Gherkin for specific domain) + Language
- Language = Dictionary + Structure
- Dictionary = Ubiquitous language
- Structure = some rules how to combine words (business terms) from dictionary in a proper ways (based on business logic)
- Way of implementation (one of the ways) – Flow Design Pattern

COMAQA.BY
QA automation community

QA Automation approaches
DSL by Martin Fowler

- The basic idea of a domain specific language (DSL) is a computer language that's targeted to a particular kind of problem (QA Automation or even QA Automation in exact domain), rather than a general purpose language that's aimed at any kind of software problem. Domain specific languages have been talked about, and used for almost as long as computing has been done.
- DSLs are very common in computing: examples include CSS, regular expressions, make, rake, ant, SQL, HQL, many bits of Rails, expectations in JMock …
- It's common to write tests using some form of DomainSpecificLanguage, such as Cucumber or an internal DSL. If you do this it's best to layer the testing DSL over the page objects so that you have a parser that translates DSL statements into calls on the page object.

QA Automation approaches
DSL types by Martin Fowler

- Internal DSLs are particular ways of using a host language to give the host language the feel of a particular language. This approach has recently been popularized by the Ruby community although it's had a long heritage in other languages - in particular Lisp. Although it's usually easier in low-ceremony languages like that, you can do effective internal DSLs in more mainstream languages like Java and C#. Internal DSLs are also referred to as embedded DSLs or FluentInterfaces

QA Automation approaches
DSL types by Martin Fowler

- External DSLs have their own custom syntax and you write a full parser to process them. There is a very strong tradition of doing this in the Unix community. Many XML configurations have ended up as external DSLs, although XML's syntax is badly suited to this purpose.
- Mixed (internal with external)
- Graphical DSLs requires a tool along the lines of a Language Workbench.