AGILE BA CHEAT SHEET

SECOND EDITION

The aim of this cheat sheet is to provide Agile Business Analysts with a quick reference card that may help them with their day to day work. For further details on Agile requirements management visit out website.

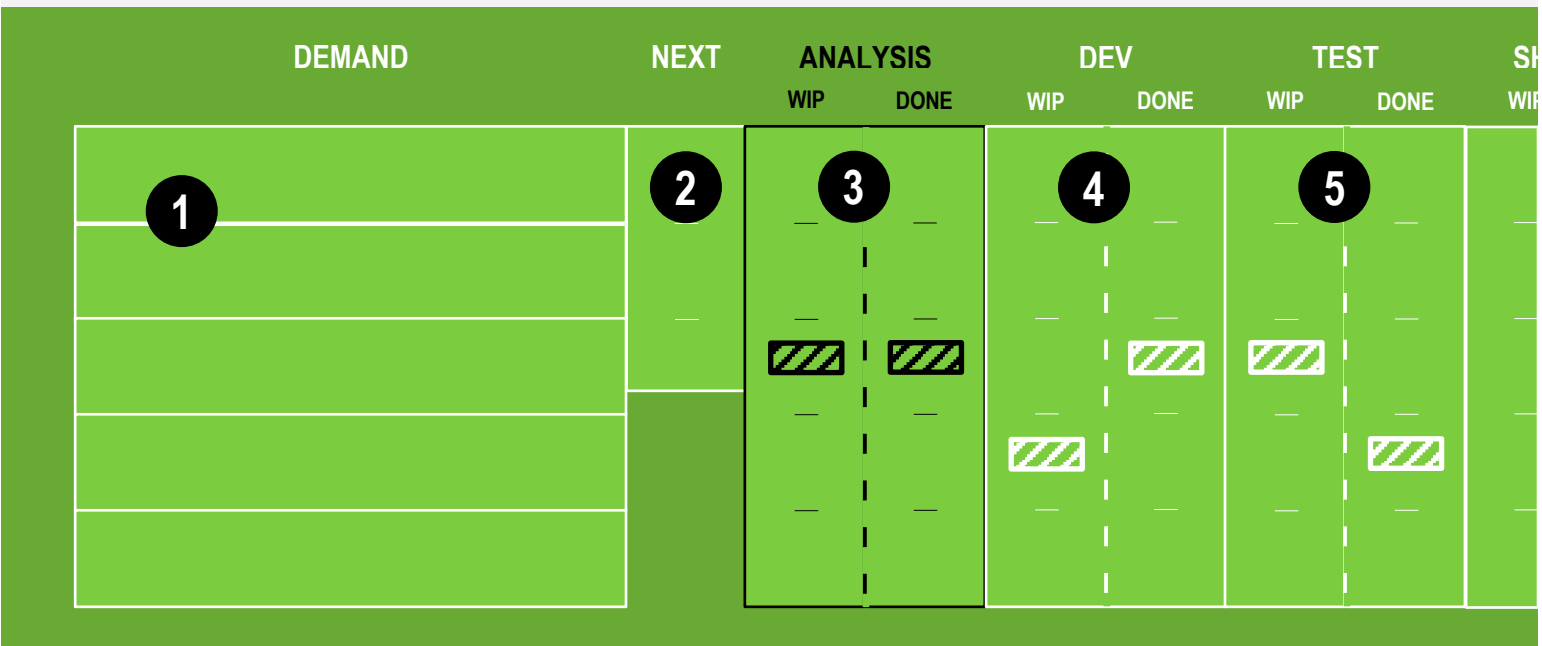
UK.LINKEDIN.COM/IN/SOLUTIONEER

@IANCARROLLUK

0844 357 1024

IAN@SOLUTIONEERS.CO.UK

WWW.SOLUTIONEERS.CO.UK



1) It's your job as BA to work closely with the Product Owner to constantly groom the demand coming into the team. Look for work that keeps getting trumped by other work to see if it's genuinely needed. The correct level of story detail at this stage could be as little as the title of the story written on the card. Getting into too much detail at this point risks waste as you may not have access to Dev's and Tester's or the story may get trumped by other work. Remember, the longer the period of time between a story being written and a developer picking it up degrades quality and requires a greater effort of hand over. This is why we never hand-over a story but instead write stories **with** Dev's and Tester's.

2) The NEXT column is used to drive out priorities and identify the next work item for the team to start next. Make sure you only pull from this column WHEN YOU HAVE THE CAPACITY TO DEAL WITH IT. If you pull too much work in, then you will just become a bottleneck and create a bigger hand-off with the rest of the team. It's easy to fall into the trap of thinking you're helping the team go faster by getting ahead with story writing but you are in fact slowing the team down. You want to do just enough up front work to allow you to focus more on fleshing out the story detail (if needed) closely WITH developers and testers. If you hit your WIP limits go downstream to help out in Dev & Test (see point 4&5 below). Remember the golden rule: Always look downstream for more work before pulling work from the left.

3) STORY WRITING

Writing stories is a group activity with Product Owner, UX, Developers, and Testers. You can use a number of formats to articulate your requirements:

- As a (user role), I want (goal), so that (benefit)
- <action> <result> <object>
- As <who> <when> <where>, I <what> because <why>.

Articulating Acceptance Criteria (from BDD):

- Given <scenario>, when <action>, then <result>

Other artefacts that teams often include in their stories are wireframes, graphical assets, test data packages, technical diagrams. Basically, include anything that helps teams to gain a common understanding of what is required.

What's the correct level of detail?
It depends! Use the Cynefin model to assess what level of detail is needed. For example, if the requirement is *Simple or Obvious* then maybe a simple descriptive title on a card is enough. *Complicated* stories may need more detail such as using the story formats above but with more prescriptive supporting artefacts such as wireframes or data. For *Complex* stories, don't constrain the team but simply write the problem statement using one of the formats above and let them explore options and solutions with you.

For large projects or pieces of work, run an inception to de-risk and understand key areas such as Biz Case, Customer Journey, and Technical Architecture.

Non-Functional Requirements (NFR's)
Don't forget your NFR's! Run workshops to understand, Accessibility, Availability, Compliance, Efficiency, Extensibility, Interoperability, Maintainability, Performance, Privacy, Quality, Resilience, Response Time, Scalability, Security, Testability, Usability…
Minimal Viable Product (MVP) - the minimal amount of functionality that can be produced to test an idea or a hypothesis. This does not represent the full go to market product. It is used to validate assumptions quickly.
Minimal Marketable Product (MMP) – the minimal amount of functionality that can be produced to take a product to market.
Story Sizing

- Never, ever, size in hours, days, or any unit of time.
- The purpose of sizing is to validate if you've broken down your stories enough
- Use relative T-Shirt sizing – S, M, L, XL. Start by finding the biggest story and put that in XL. Then find the smallest and put that in S. Now simply offset the remaining stories. Break down L's or XL's. Repeat.

Story Splitting

- Workflow steps
- Business Rule Variations
- Major Effort
- Simple / Complex
- Variations in Data
- Data Entry Methods
- Defer Performance
- Operations (e.g. CRUD)
- Breakout a spike

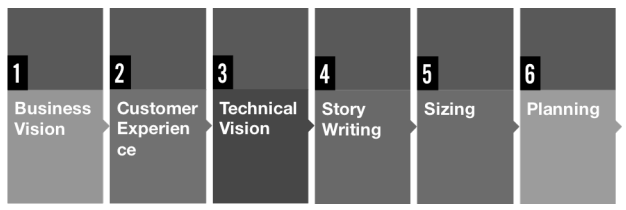
Look at your acceptance criteria - each acceptance criteria often represents a split!

4) It's really critical that you work closely with Developers. A tight feedback loop is required. Work on building strong relationships with all developers. You should be regularly pairing up with Developers to answer any queries or clarify any requirements. Encourage Developers to regularly showcase their work to you and constantly listen out for conversations between developers that you can contribute constructively to.

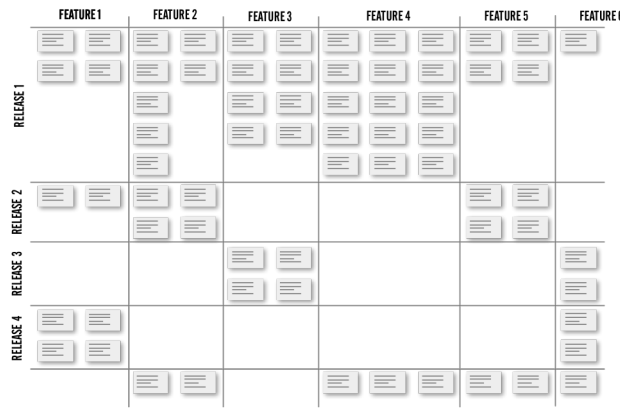
5) It's very common in an Agile software development team for Business Analysts to 'swarm' downstream to help out in Test. As with pairing up with Developers, you can do the same with Testers. A clear understanding of how the software is tested is critical and a key area for Business Analysts to work with Testers to feed into the Test Strategy. You may also be called upon to actually do some hands-on testing. Keep yourself close to how the software is evolving.

RECOMMENDED READING

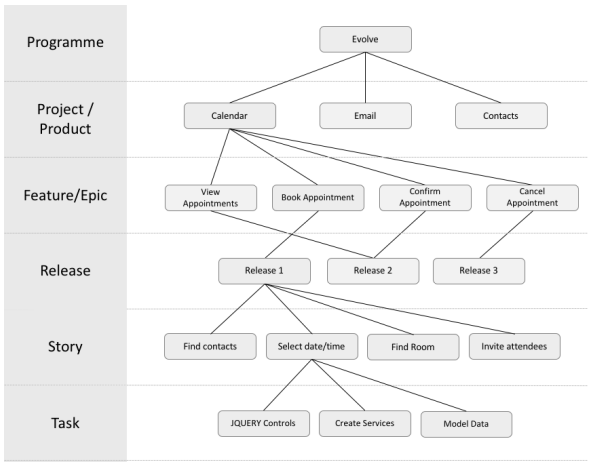
STRUCTURE OF AN INCEPTION



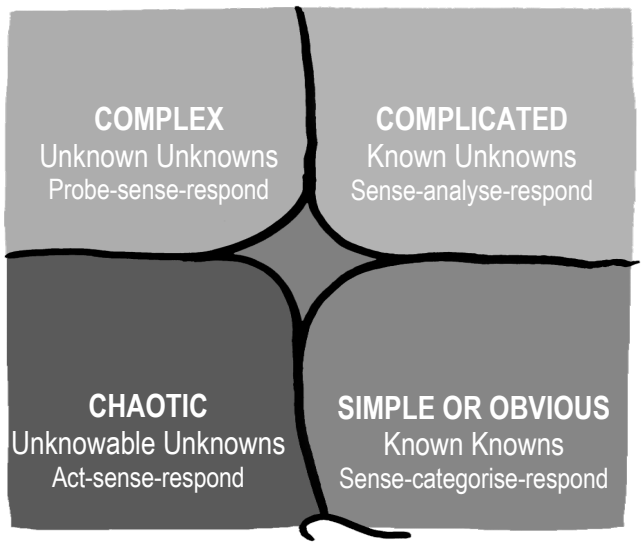
RELEASE PLANNING



REQUIREMENTS HIERARCHY



CYNEFIN



SOLUTIONEERS