Using Agile on Package Implementations

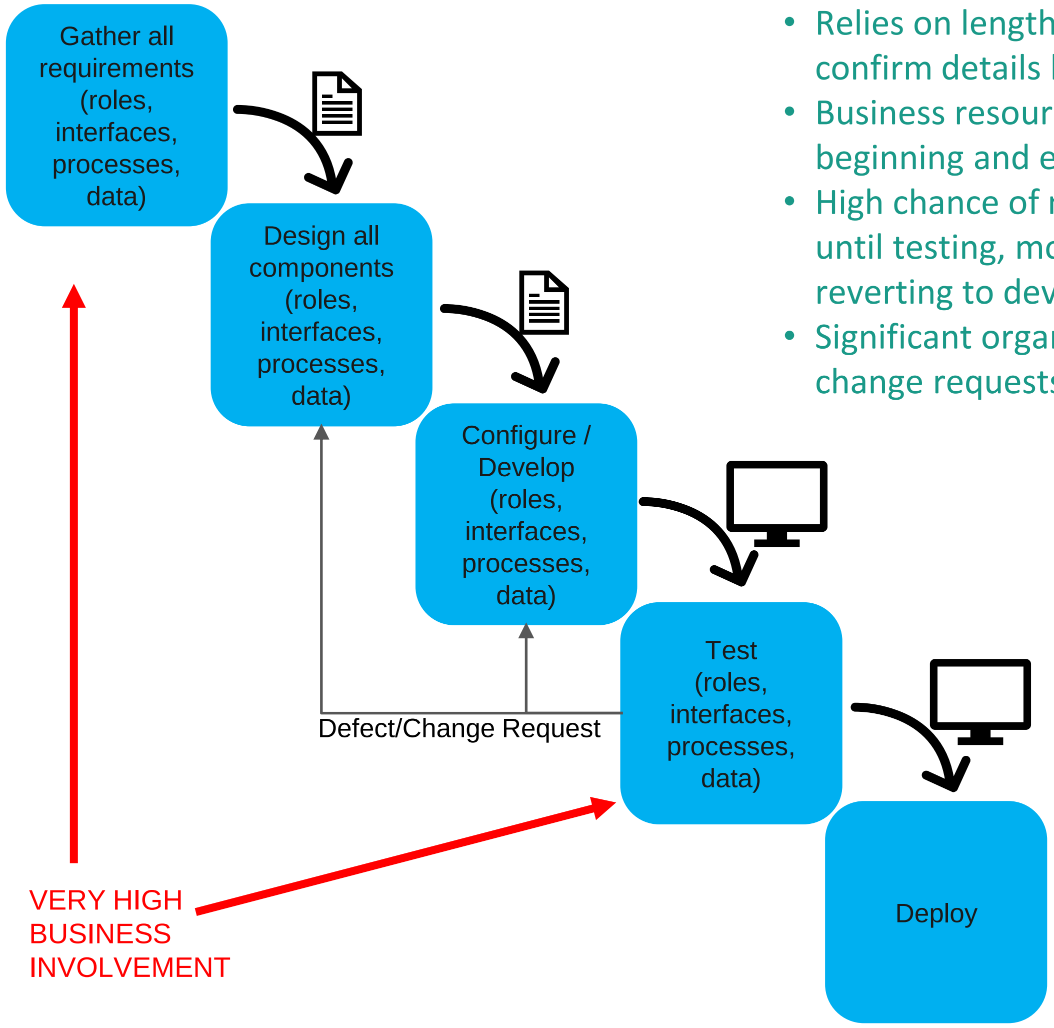
About this deck

- This deck is meant to be a sample set of slides to explain to business people who aren't typically familiar with project, product, or software lifecycles.
- There are 2 primary slides that are always needed: How a traditional implementation works, and how an Agile implementation works.
- Those slides are necessary but not sufficient for any given organization. Some may need more upfront context, some may need some specifics after. Thoughts on that content is provided but need to be fleshed out for your particular company and culture.

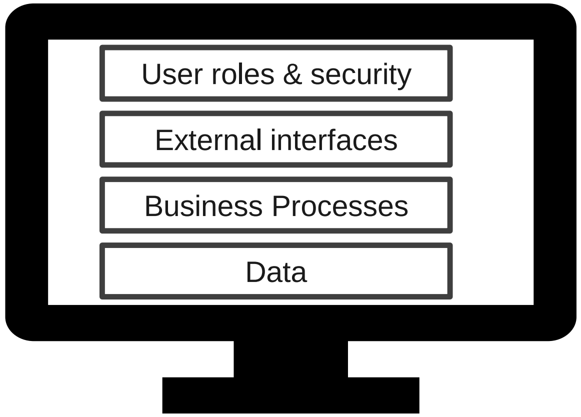
Setting the stage

- Put up front context here if needed. Items like “Why are we explaining this”, “Why is <your company> trying Agile”, etc

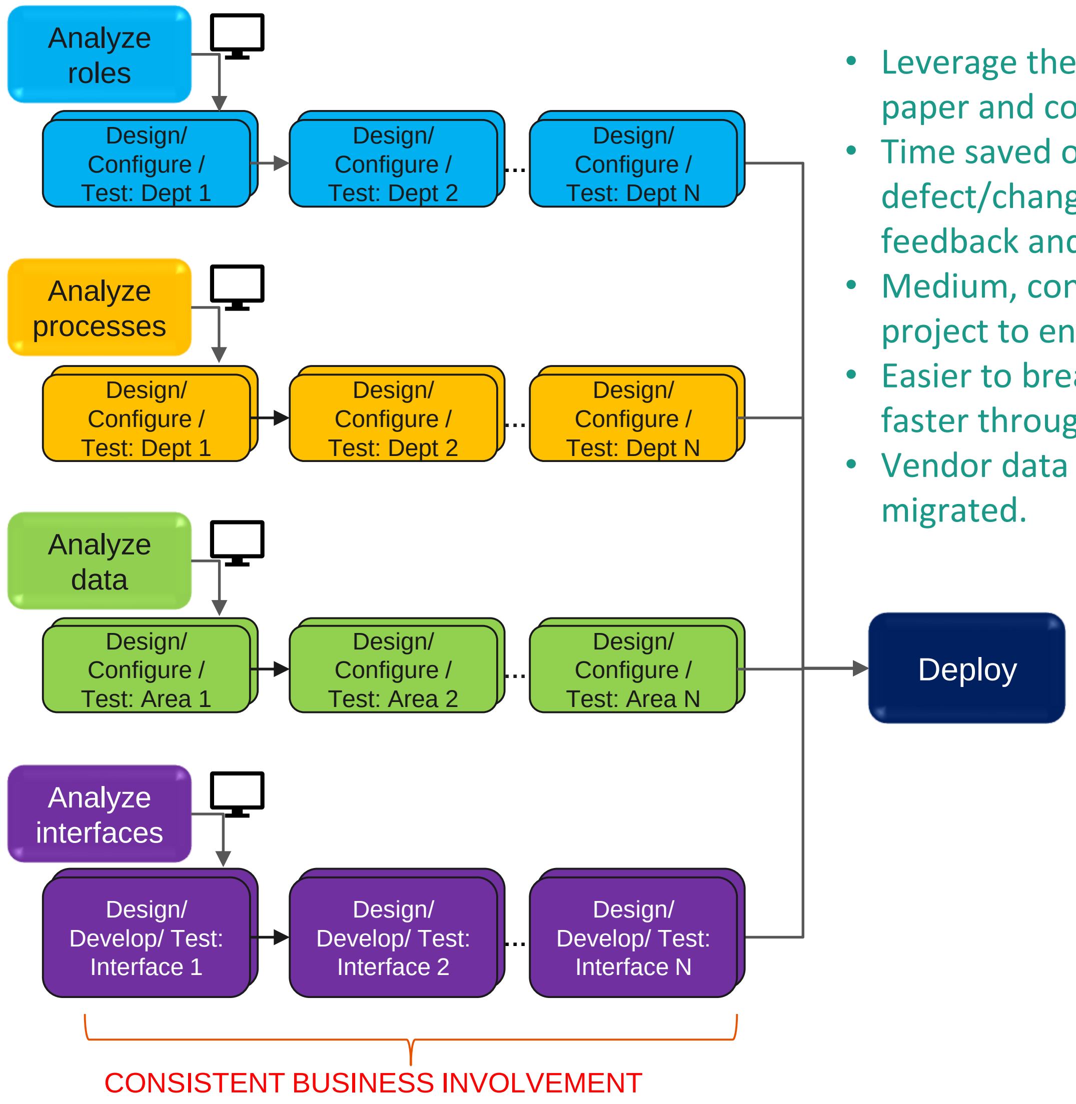
Waterfall (traditional) life cycle for package implementations


Gather all requirements (roles, interfaces, processes, data)
Design all components (roles, interfaces, processes, data)
Configure / Develop (roles, interfaces, processes, data)
Test (roles, interfaces, processes, data)
Deploy
Defect/Change Request
VERY HIGH BUSINESS INVOLVEMENT

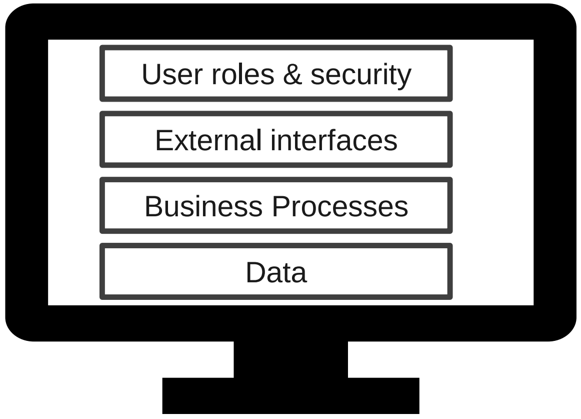
- Relies on lengthy paper documents for the initial stages and confirm details before moving on.
- Business resource usage fluctuates between very high in beginning and end and very limited in the middle.
- High chance of not realizing requirements were misinterpreted until testing, months later. Problems found may require reverting to develop or design stage based on severity.
- Significant organizational time to classify issues into defect or change requests



Agile lifecycle for package implementations


Analyze roles
Design/ Configure / Test: Dept 1
Design/ Configure / Test: Dept 2
...
Design/ Configure / Test: Dept N
Analyze processes
Design/ Configure / Test: Dept 1
Design/ Configure / Test: Dept 2
...
Design/ Configure / Test: Dept N
Analyze data
Design/ Configure / Test: Area 1
Design/ Configure / Test: Area 2
...
Design/ Configure / Test: Area N
Analyze interfaces
Design/ Develop/ Test: Interface 1
Design/ Develop/ Test: Interface 2
...
Design/ Develop/ Test: Interface N
Deploy
CONSISTENT BUSINESS INVOLVEMENT

- Leverage the completeness of a vendor solution to skip paper and configure requirements in a working system.
- Time saved on writing and reviewing paper and the defect/change request process can be spent on user feedback and vendor modifications.
- Medium, consistent business involvement throughout project to ensure alignment with business needs.
- Easier to break up into multiple teams for parallel work and faster throughput.
- Vendor data used as a proxy until company data has been migrated.



Your role in making <initiative> successful

Some company cultures work better when individuals have specific guidance on how they can help. Here are some example thoughts using far too many words. Trim based on what would resonate with your culture, and expand where needed:

- <Initiative> will deliver working code instead of paper. The first iteration may have material shortcomings. That’s okay, understanding what you like and dislike is important. Provide feedback quickly, so <vendor> can modify and return.
- Providing initial and high level feedback in 1 day is better than performing a detailed look and providing feedback after 1 week. The more iterations we can deliver, the better the solution will fit our needs.
- (If the vendor has provided you with a demo environment with sample data) Go learn the vanilla <package name> so you understand the system basics such as navigation, terminology, and look & feel. Then when we get our configured system you’re familiar with foundational concepts.