citi
for projects and programmes

Iterative project management practices
are easier than Agile methods

David Tuffs
Independent Consultant

Thomas Docker
Chairman, CITI Group

The Agile manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Some comments on Agile methods

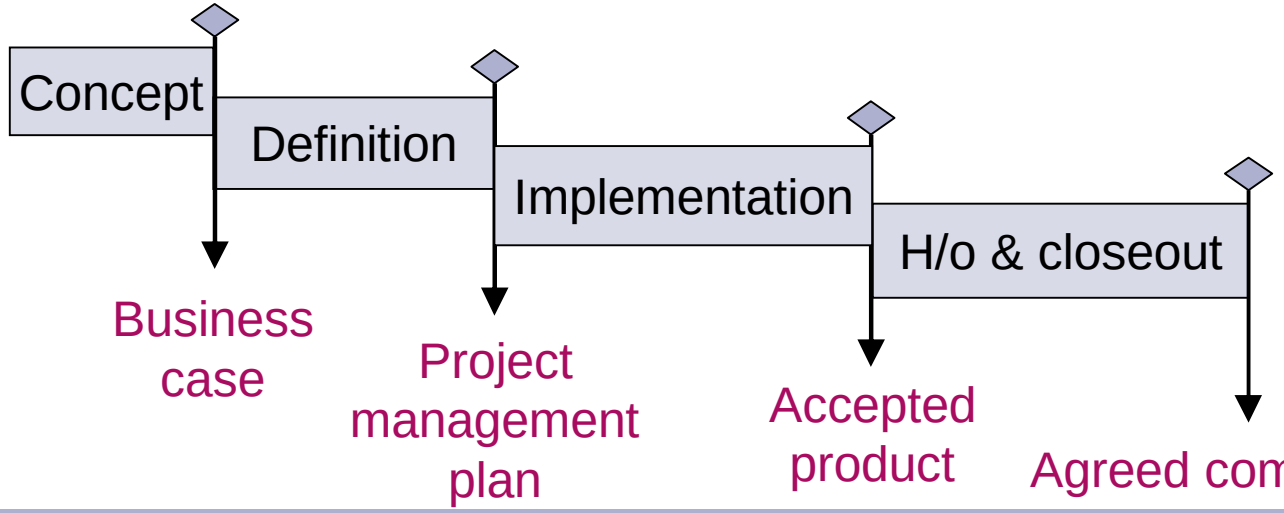
We advocate iterative projects, however …

- ‘Agile’ is an unhelpful term!
  - ***agile*** adj.: nimble, quick-moving, active, lively
  - antonym: ***clumsy***. A very pejorative term for any residual linear projects!
- Some Agile exponents are doing the cause no favours
  - “The role of management in an agile project is to fetch pizza” – *Kent Beck*
- All the Agile approaches are very similar under the covers
- The *distillation* of everything Agile is just a **set of practices** …
  - … as advocated by Ivar Jacobson

Life cycles key relationships

Project management life cycle

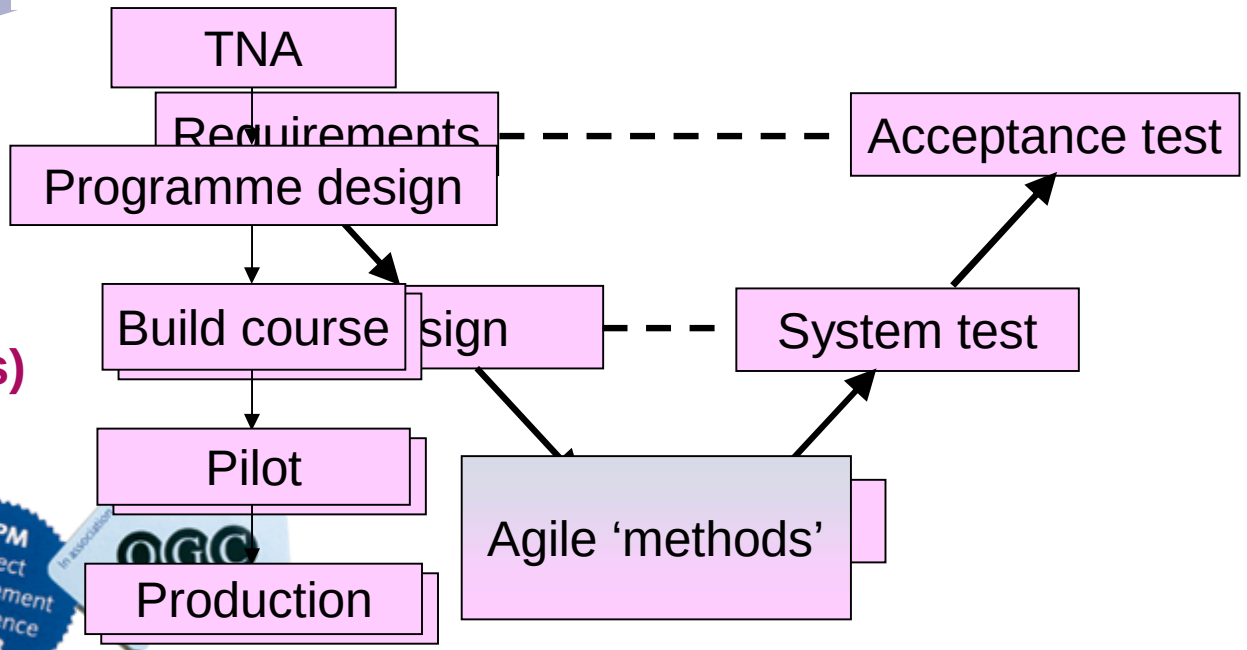
(One instance)



Constrains

Product (deliverable) life cycle

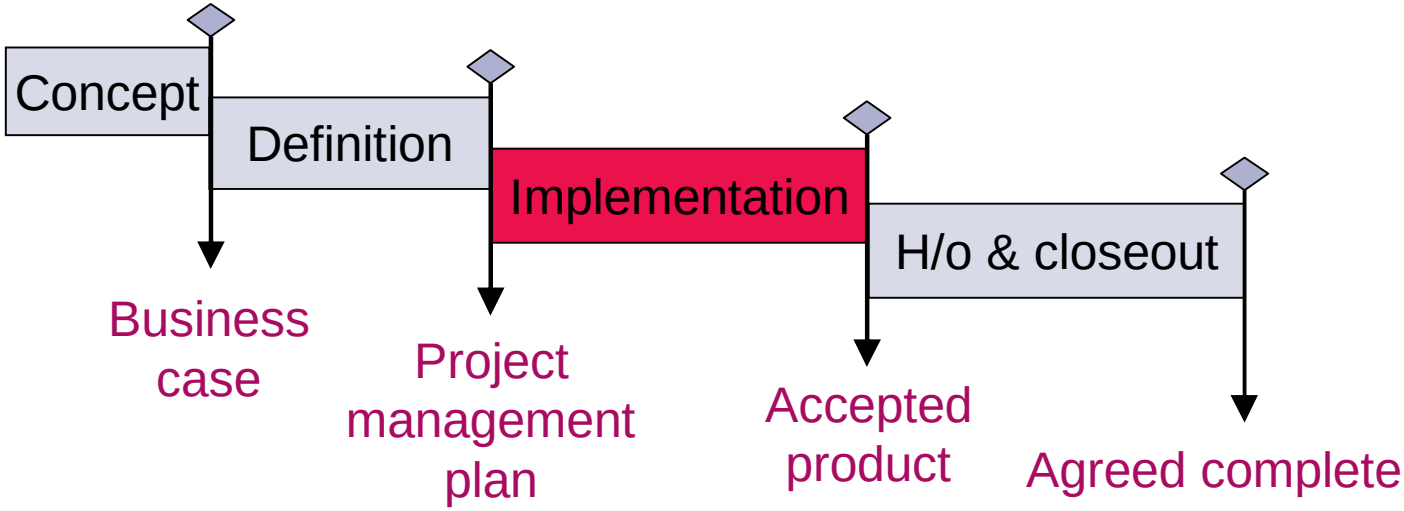
(Many instances)



Life cycles one-shot approach

Project management life cycle

(One instance)

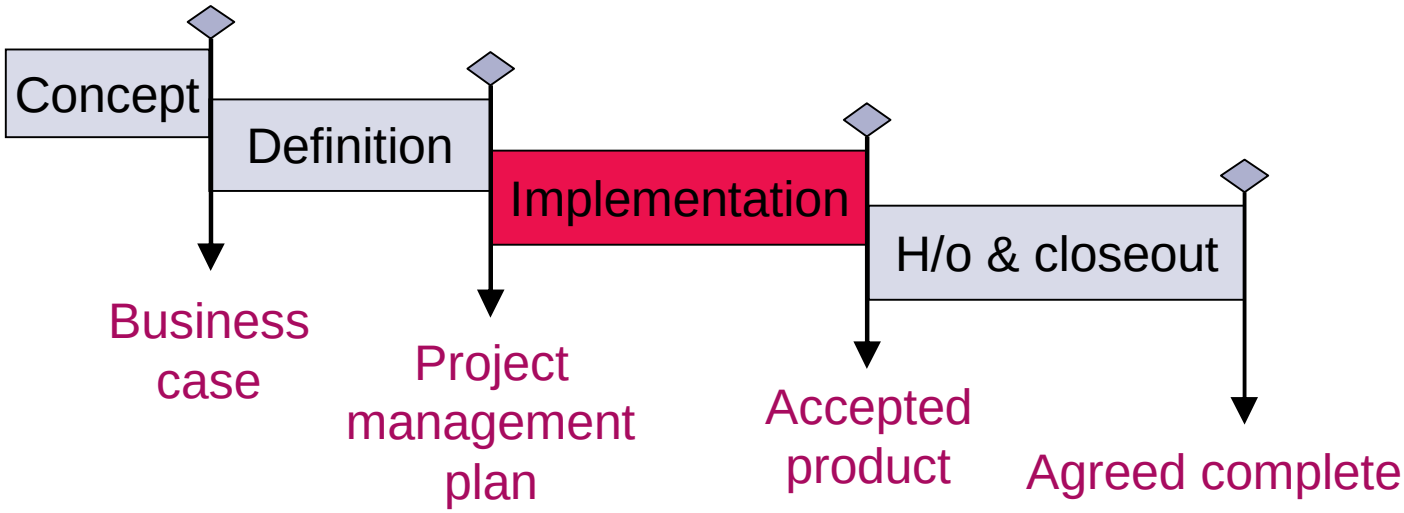


Project practices	*Product practices*
Ü Business requirements	**Design**
Ü Project plan	***complete specification***
Ü Governance gates	**Build and test**
	bundled / unstructured
	Integrate
	at the end
	Test
	planned phase
	Delivery
	user acceptance test

Life cycles iterative project management

Project management life cycle

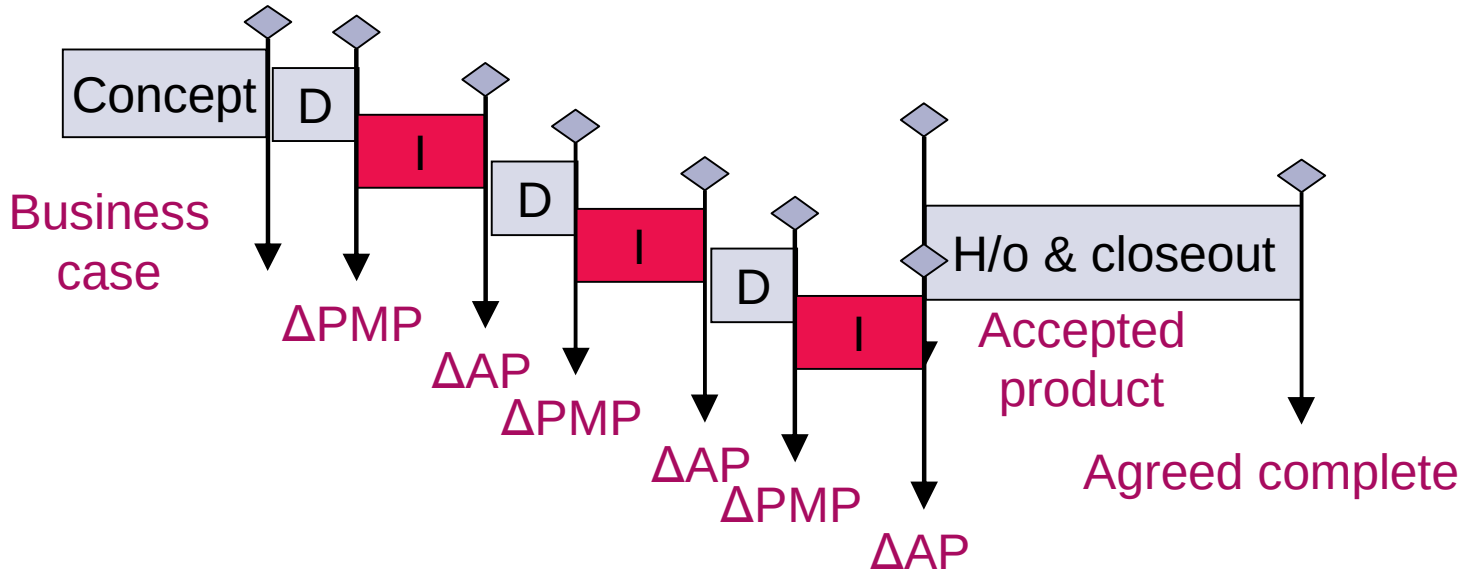
(One instance)



Life cycles iterative project management

Project management life cycle

(One instance)



- Ü **Volatile requirements**
- Ü **Time-boxed plans**
- Ü **Users reviews**
- Ü **Continuous integration**

Design
timeboxed
Build and test
by requirement priority
Integrate and test
daily / weekly
Delivery
ongoing acceptance
delivery-ready prototypes

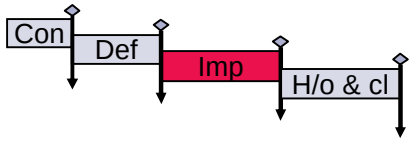
Evolutionary development…

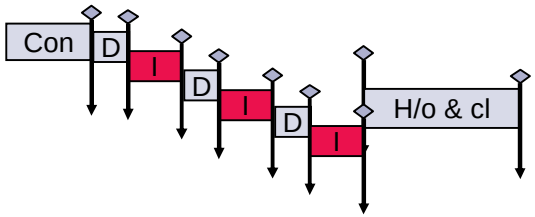
- Vision
- Design authority
- Product breakdown structure
- Configuration management
- Change control

…doesn't work without an 'architectural' plan and good project practices

Methodology and practices

Lifecycle





Constrains

Project practices

Ü Business requirements Ü Project plan Ü Governance gates	Ü Prioritised requirements Ü Rolling plans Ü Stepwise refinement Ü Some ongoing integr'n	Ü Volatile requirements Ü Time-boxed plans Ü Users reviews Ü Continuous integration

Constrains

Continuum of change

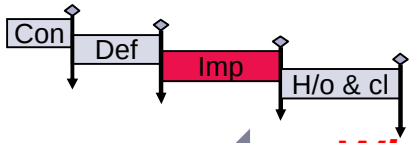
Constrains

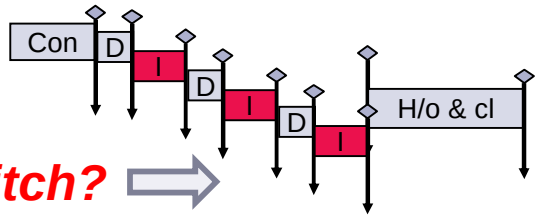
Product practices

Design *complete specification* **Build and test** *bundled / unstructured* **Integrate** *at the end* **Test** *planned phase* **Deliver** *user acceptance test*	**Design** *rolling design* **Build and test** *by requirement priority* **Integrate** *ongoing + end phase* **Test** *ongoing + end phase* **Deliver** *user acceptance test*	**Design** *timeboxed* **Build and test** *by requirement priority* **Integrate and test** *daily / weekly* **Delivery** *ongoing acceptance* *delivery-ready prototypes*

Methodology and practices

Lifecycle





When is the best time to switch?

Constrains

Project practices

Ü Business requirements Ü Project plan Ü Governance gates	Ü Prioritised requirements Ü Rolling plans Ü Stepwise refinement Ü Some ongoing integr'n	Ü Volatile requirements Ü Time-boxed plans Ü Users reviews Ü Continuous integration

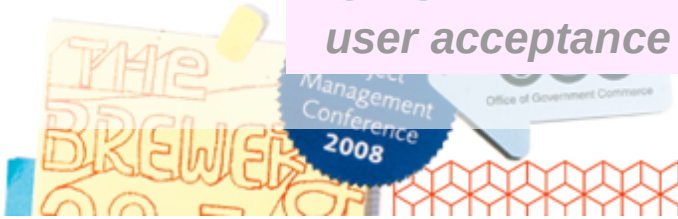
Constrains

Continuum of change

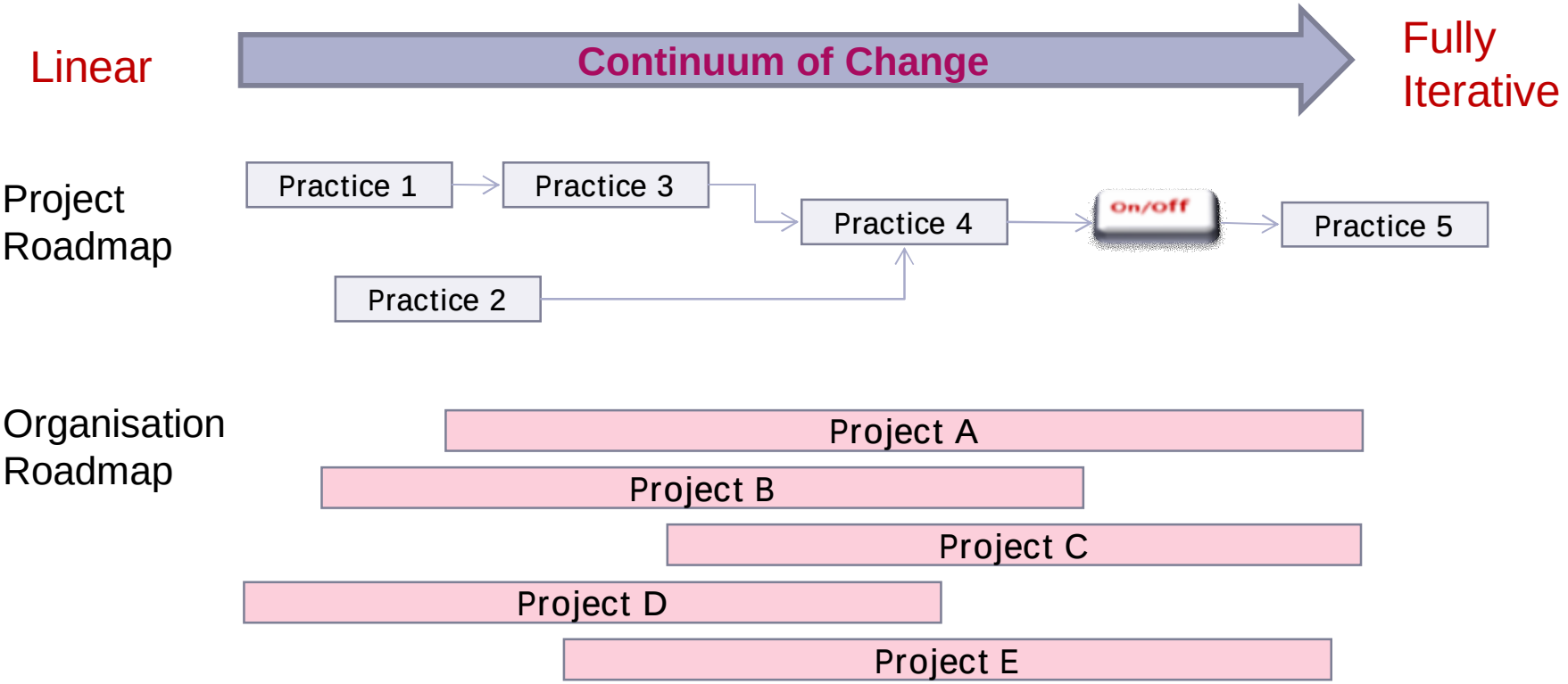
Constrains

Product practices

Design *complete specification* **Build and test** *bundled / unstructured* **Integrate** *at the end* **Test** *planned phase* **Deliver** *user acceptance test*	**Design** *rolling design* **Build and test** *by requirement priority* **Integrate** *ongoing + end phase* **Test** *ongoing + end phase* **Deliver** *user acceptance test*	**Design** *time-boxed* **Build and test** *by requirement priority* **Integrate and test** *daily / weekly* **Delivery** *ongoing acceptance* *delivery-ready prototypes*



Project and organisation roadmaps



- Not all projects will start from the same place
- Possibly not all projects will end up fully iterative
- But every project has a better place to get to – in safety

Conclusions

- Don't believe everything you read in Agile books!
- Lifecycles constrain project practices and t
  practices
- Switching to iterati
  managed as such
- ive and
  tices by the practitioners
- tation to t e methodology switch too early!

Iterative is good engineering discipline -- not 'cutting corners'

citi
for projects and programmes

Thank you

Questions and discussion

David Tuffs

Thomas Docker