Software Project Management

Chapter Four

Selection of an appropriate project approach

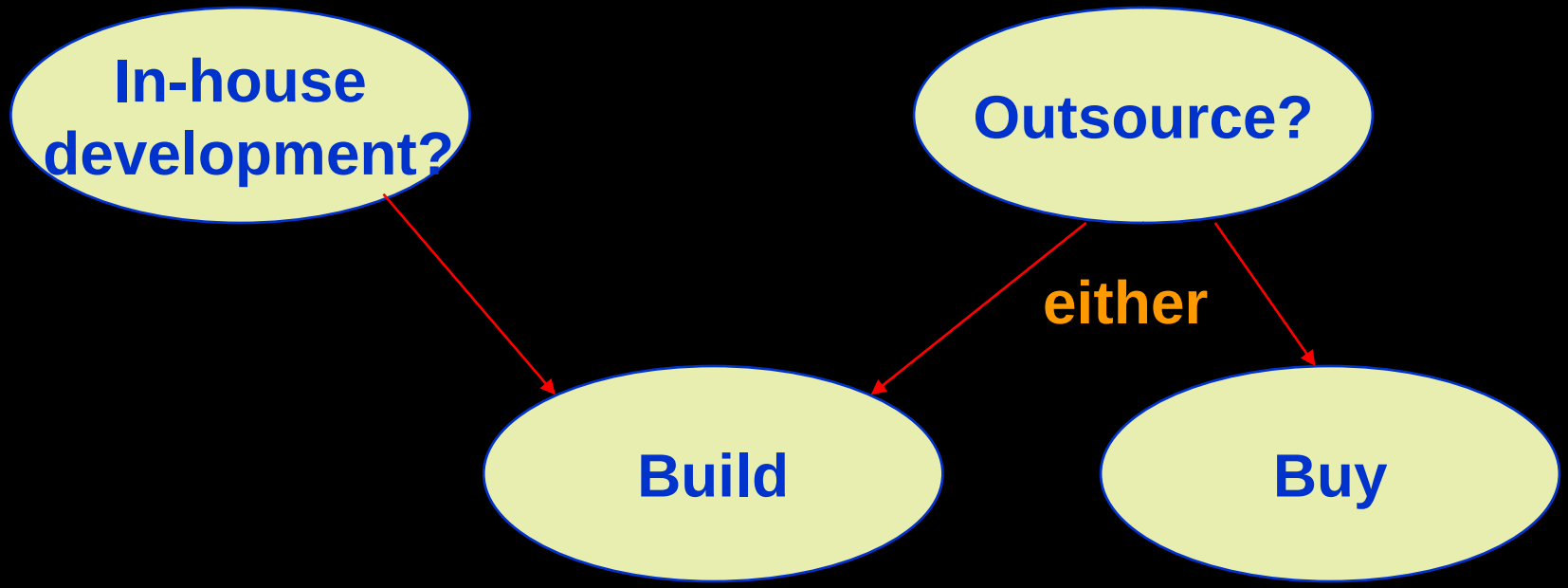
Outline of lecture

- Building versus buying software
- Taking account of the characteristics of the project
- Process models
  - Waterfall
  - Prototyping and iterative approaches
  - Incremental delivery
- Agile approaches

Selection of project approaches

- This lecture concerned with choosing the right approach to a particular project: variously called *technical planning*, *project analysis*, *methods engineering* and *methods tailoring*
- In-house: often the methods to be used dictated by organizational standards
- Suppliers: need for tailoring as different customers have different needs

Build or buy?



Some advantages of off-the-shelf (OTS) software

- Cheaper as supplier can spread development costs over a large number of customers
- Software already exists
  - Can be trialled by potential customer
  - No delay while software being developed
- Where there have been existing users, bugs are likely to have been found and eradicated

Some possible disadvantages of off-the-shelf

- Customer will have same application as everyone else: no competitive advantage, *but* competitive advantage may come from the *way* application is used
- Customer may need to change the way they work in order to fit in with OTS application
- Customer does not own the code and cannot change it
- Danger of over-reliance on a single supplier

General approach

- Look at risks and uncertainties e.g.
  - are requirement well understood?
  - are technologies to be used well understood?
- Look at the type of application being built e.g.
  - information system? embedded system?
  - criticality? differences between target and development environments?
- Clients' own requirements
  - need to use a particular method

Structure versus speed of delivery

Structured approach

- Also called ‘heavyweight’ approaches
- Step-by-step methods where each step and intermediate product is carefully defined
- Emphasis on getting quality right first time
- Example: use of UML and USDP
- Future vision: Model-Driven Architecture (MDA). UML supplemented with Object Constraint Language, press the button and application code generated from the UML/OCL model

Structure versus speed of delivery

Agile methods

- Emphasis on speed of delivery rather than documentation
- RAD Rapid application development emphasized use of quickly developed prototypes
- JAD Joint application development. Requirements are identified and agreed in intensive workshops with users

Processes versus Process Models

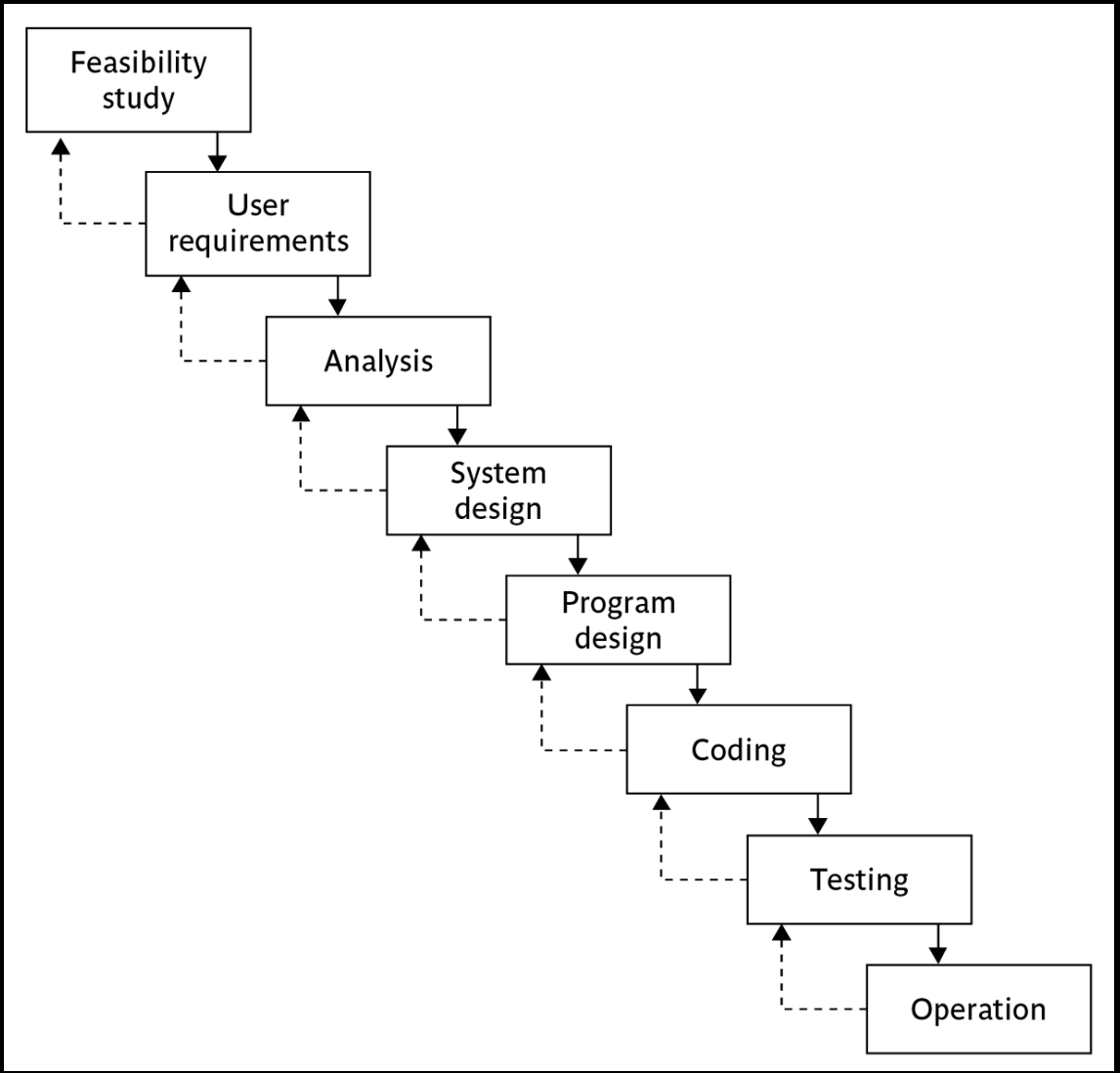
- Starting from the inception stage:
  - A product undergoes a series of transformations through a few identifiable stages
  - Until it is fully developed and released to the customer.
  - This forms its life cycle or development process.
- Life cycle model (also called a process model):
  - A graphical or textual representation of the life cycle.

Choice of process models

- ‘waterfall’ also known as ‘one-shot’, ‘once-through’
- incremental delivery
- evolutionary development

Also use of ‘agile methods’ e.g. extreme programming

Waterfall


Feasibility study
User requirements
Analysis
System design
Program design
Coding
Testing
Operation

Waterfall

- the ‘classical’ model
- imposes structure on the project
- every stage needs to be checked and signed off
- BUT
  - limited scope for iteration
- V model approach is an extension of waterfall where different testing phases are identified which check the quality of different development phases

Evolutionary delivery: prototyping

‘ *An iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions’*

Sprague and McNurlin

main types

- ‘throw away’ prototypes
- evolutionary prototypes

what is being prototyped?

- human-computer interface
- functionality

Reasons for prototyping

- learning by doing
- improved communication
- improved user involvement
- a feedback loop is established
- reduces the need for documentation
- reduces maintenance costs i.e. changes after the application goes live
- prototype can be used for producing expected results

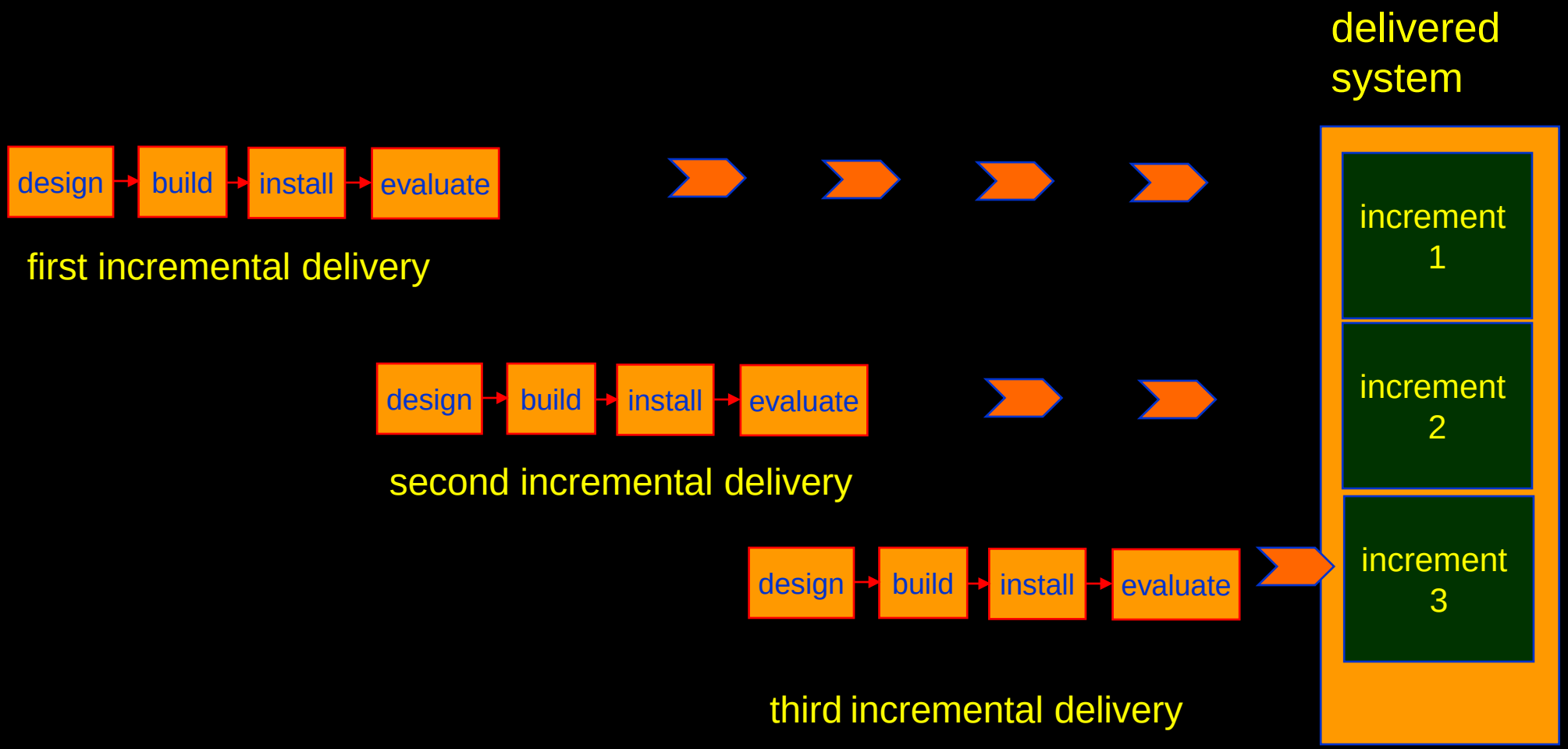
prototyping: some dangers

- users may misunderstand the role of the prototype
- lack of project control and standards possible
- additional expense of building prototype
- focus on user-friendly interface could be at expense of machine efficiency

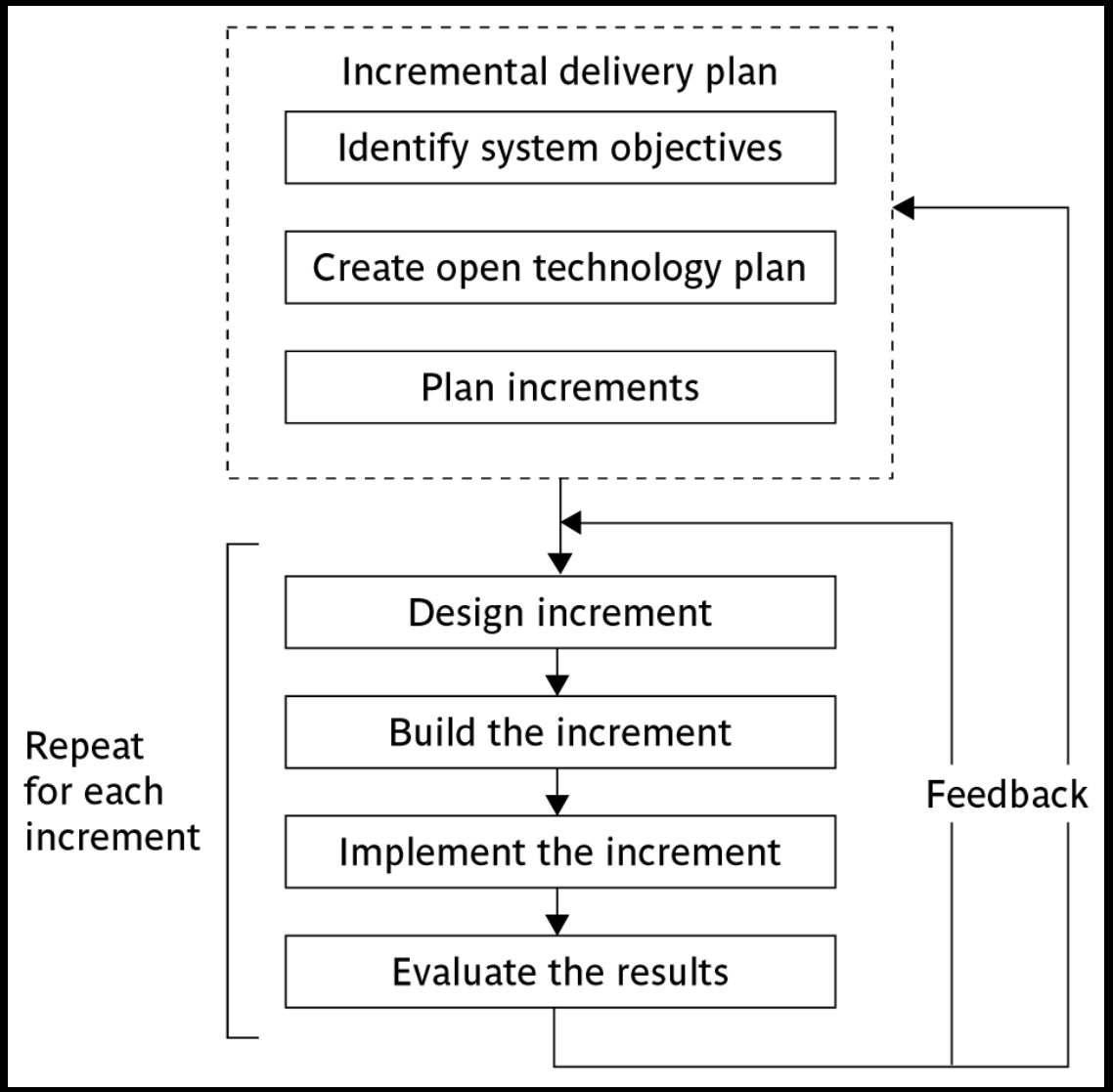
other ways of categorizing prototyping

- what is being learnt?
  - organizational prototype
  - hardware/software prototype (‘experimental’)
  - application prototype (‘exploratory’)
- to what extent
  - mock-ups
  - simulated interaction
  - partial working models: *vertical* versus *horizontal*

Incremental delivery



The incremental process



Incremental approach: benefits

- feedback from early stages used in developing latter stages
- shorter development thresholds
- user gets some benefits earlier
- project may be put aside temporarily
- reduces ‘gold-plating’

BUT there are some possible disadvantages

- loss of economy of scale
- ‘software breakage’

Overview of incremental plan

- steps ideally 1% to 5% of the total project
- non-computer steps should be included
- ideal if a step takes one month or less:
  - not more than three months
- each step should deliver some benefit to the user
- some steps will be physically dependent on others

which step first?

- some steps will be pre-requisite because of physical dependencies
- others may be in any order
- value to cost ratios may be used
  - V/C where
  - V is a score 1-10 representing value to customer
  - C is a score 0-10 representing value to developers

V/C ratios: an example

step	value	cost	ratio	
profit reports	9	1	9	2nd
online database	1	9	0.11	5th
ad hoc enquiry	5	5	1	4th
purchasing plans	9	4	2.25	3rd
profit- based pay for managers	9	0	inf	1st

Genesis of ‘Agile’ methods

Structured development methods have several disadvantages

- produce large amounts of documentation which can largely remain unread
- documentation has to be kept up to date
- division into specialist groups and need to follow procedures stifles communication
- users can be excluded from decision process
- long lead times to deliver anything etc. etc

The answer? ‘Agile’ methods?

Agile Methods

- Agile is an umbrella term that refers to a group of development processes:
  - Crystal technologies
  - Atern (formerly DSDM)
  - Feature-driven development
  - Scrum
  - Extreme Programming (XP)
- Similar themes:
  - Some variations

Important Themes of Agile Methods

- Base on the incremental approach:
  - At a time, only one increment is planned, developed, and then deployed at the customer site.
- Agile model emphasizes face-to-face communication over written documents.
- An agile project usually includes a customer representative in the team.
- Agile development projects usually deploy pair programming.

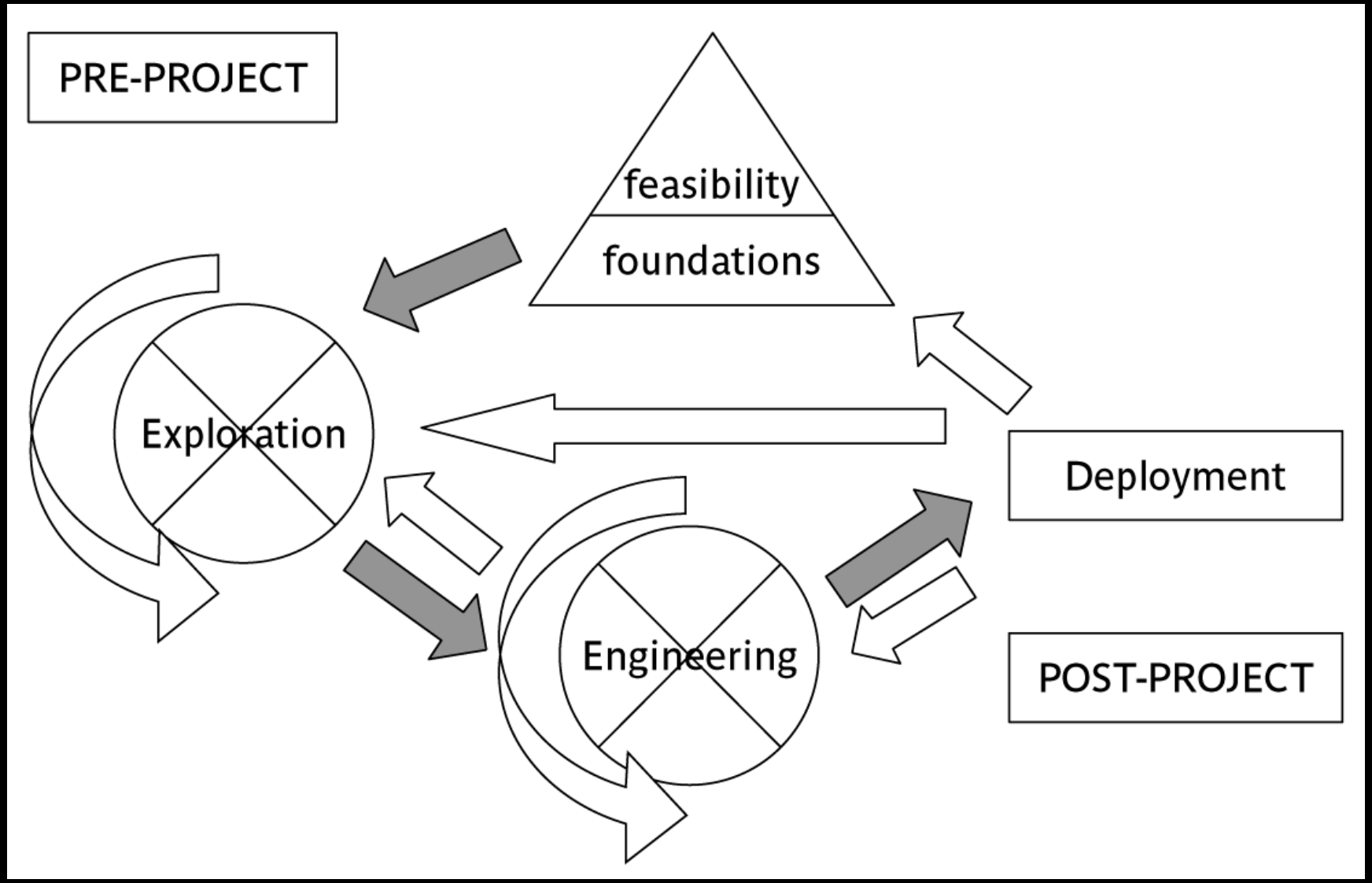
Atern/Dynamic system development method (DSDM)

- UK-based consortium
- *arguably* DSDM can be seen as replacement for SSADM
- DSDM is more a project management approach than a development approach
- Can still use DFDs, LDSs etc!
- An update of DSDM has been badged as ‘Atern’

Six core Atern/DSDM principles

1. Focus on business need
2. Delivery on time – use of time-boxing
3. Collaborate
4. Never compromise qualitiy
5. Deliver iteratively
6. Build incrementally

Atern/DSDM framework



Atern DSDM: time-boxing

- *time-box* fixed deadline by which *something* has to be delivered
- typically two to six weeks
- MOSCOW priorities
  - Must have - essential
  - Should have - very important, but system could operate without
  - Could have
  - Want - but probably won't get!

Extreme Programming Model

- Extreme programming (XP) was proposed by Kent Beck in 1999.
- The methodology got its name from the fact that:
  - Recommends taking the best practices to extreme levels.
  - If something is good, why not do it all the time.

Extreme programming

- increments of one to three weeks
  - customer can suggest improvement at any point
- argued that distinction between design and building of software are artificial
- code to be developed to meet current needs only
- frequent re-factoring to keep code structured

Taking Good Practices to Extreme

- If code review is good:
  - Always review --- pair programming
- If testing is good:
  - Continually write and execute test cases --- test-driven development
- If incremental development is good:
  - Come up with new increments every few days
- If simplicity is good:
  - Create the simplest design that will support only the currently required functionality.

Taking Good Practices to Extreme

- If design is good,
  - everybody will design daily (refactoring)
- If architecture is important,
  - everybody will work at defining and refining the architecture (metaphor)
- If integration testing is important,
  - build and integrate test several times a day (continuous integration)

extreme programming - contd

- developers work in pairs
- test cases and expected results devised *before* software design
- after testing of increment, test cases added to a consolidated set of test cases

Limitations of extreme programming

- Reliance on availability of high quality developers
- Dependence on personal knowledge – after development knowledge of software may decay making future development less easy
- Rationale for decisions may be lost e.g. which test case checks a particular requirement
- Reuse of existing code less likely

Grady Booch's concern

Booch, an OO authority, is concerned that with requirements driven projects:

'Conceptual integrity sometimes suffers because this is little motivation to deal with scalability, extensibility, portability, or reusability beyond what any vague requirement might imply'

Tendency towards a large number of discrete functions with little common infrastructure?

Project Characteristics that Suggest Suitability of Extreme Programming

- Projects involving new technology or research projects.
  - In this case, the requirements change rapidly and unforeseen technical problems need to be resolved.
- Small projects:
  - These are easily developed using extreme programming.

Scrum

- One of the “agile processes”
- Self-organizing teams
- Product progresses in a series of month-long “sprints”
- Requirements are captured as items in a list of “product backlog”

Sprints

- Scrum projects make progress in a series of “sprints”
- Analogous to XP iterations
- Target duration is one month
  - +/- a week or two
- During a sprint, a product increment is designed, coded, and tested

Key Roles in Scrum Process

- Product Owner
  - Acts on behalf of customers to represent their interests.
- Development Team
  - Team of five-nine people with cross-functional skill sets.
- Scrum Master
  - Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

Scrum Ceremonies

- Sprint Planning Meeting
- Sprint
- Daily Scrum
- Sprint Review Meeting

Sprint Planning

- In this meeting, the product owner and the team members decide which Backlog Items the Team will work on in the next sprint
- Scrum Master should ensure that the Team agrees to realistic goals

Sprint

- Fundamental process flow of Scrum
- A month-long iteration, during which an incremental product functionality completed
- NO outside influence can interfere with the Scrum team during the Sprint
- Each Sprint begins with the Daily Scrum Meeting

Daily Scrum

- Held daily:
  - Short meeting
  - Lasts for about 15mins only

- Main objective is to answer three questions:
  - What did you do yesterday
  - What will you do today?
  - What obstacles are in your way?

Sprint Review Meeting

- Team presents what it accomplished during the sprint
  - Typically takes the form of a demo of new features or underlying architecture
- Informal meeting:
  - The preparation time should not exceed about 2-hours

Sprint Artefacts

- Product backlog
- Sprint backlog
- Sprint burndown chart

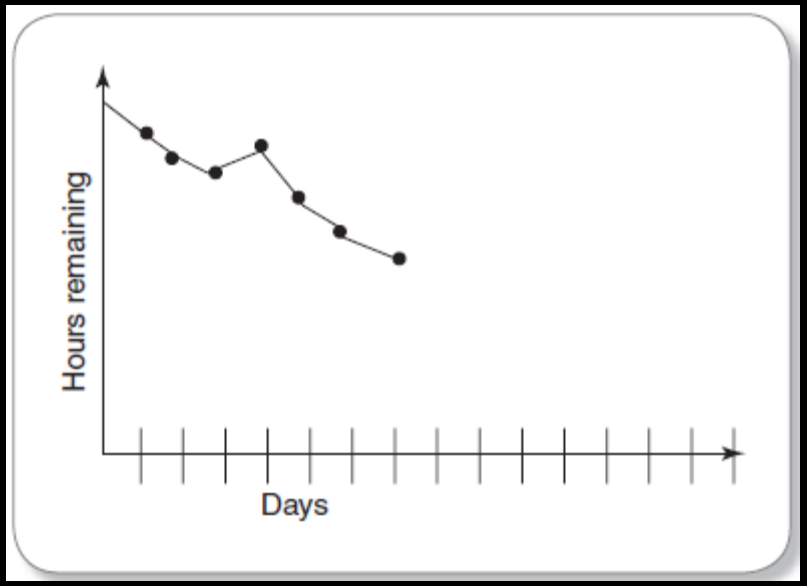
Product Backlog

- A list of all desired work on the project --- usually a combination of :
  - story-based work (e.g. “let user search and replace”)
  - task-based work (“improve exception handling”)
- List is prioritized by the Product Owner

Sprint Backlog

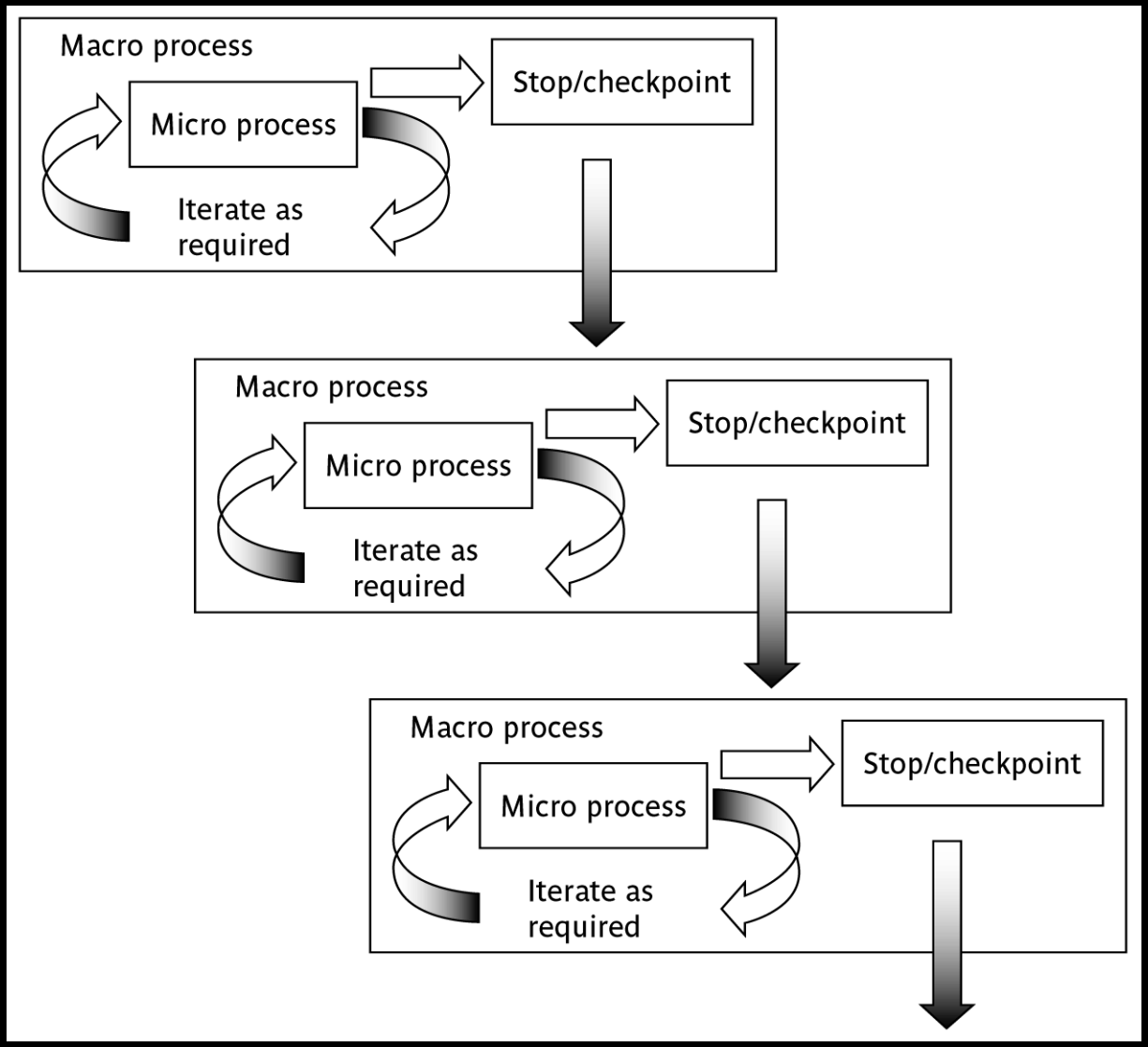
- A subset of Product Backlog Items, which define the work for a Sprint
  - Created by Team members
  - Each Item has it’s own status
  - Updated daily

Sprint Burndown Chart

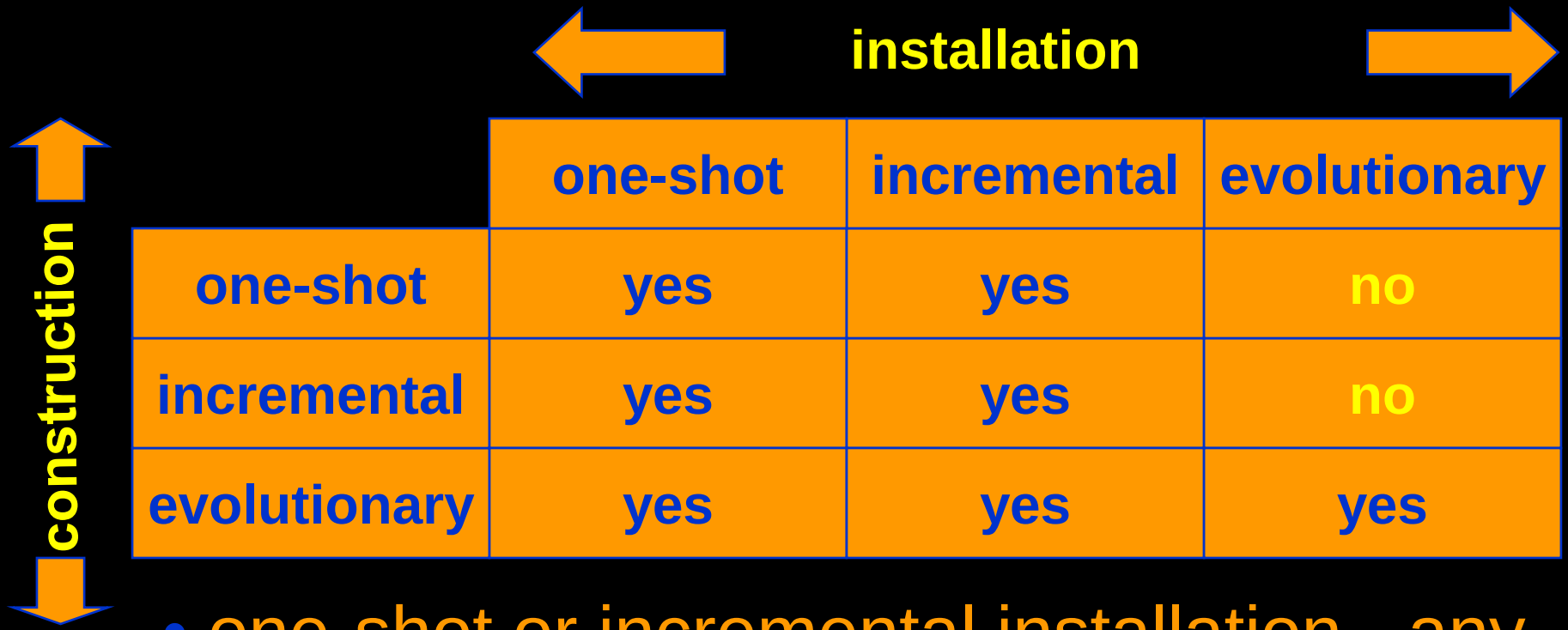


- Day-wise depicts the total Sprint Backlog hours remaining
- Ideally should burn down to zero to the end of the Sprint

macro and micro processes



combinations of approach



	one-shot	incremental	evolutionary
one-shot	yes	yes	no
incremental	yes	yes	no
evolutionary	yes	yes	yes

- one-shot or incremental installation - any construction approach possible
- evolutionary installation implies evolutionary construction

‘rules of thumb’ about approach to be used

IF uncertainty is high

THEN use evolutionary approach

IF complexity is high but uncertainty is not

THEN use incremental approach

IF uncertainty and complexity both low

THEN use one-shot

IF schedule is tight

THEN use evolutionary or incremental