IBM Global Business Services

Engineering and Business Software

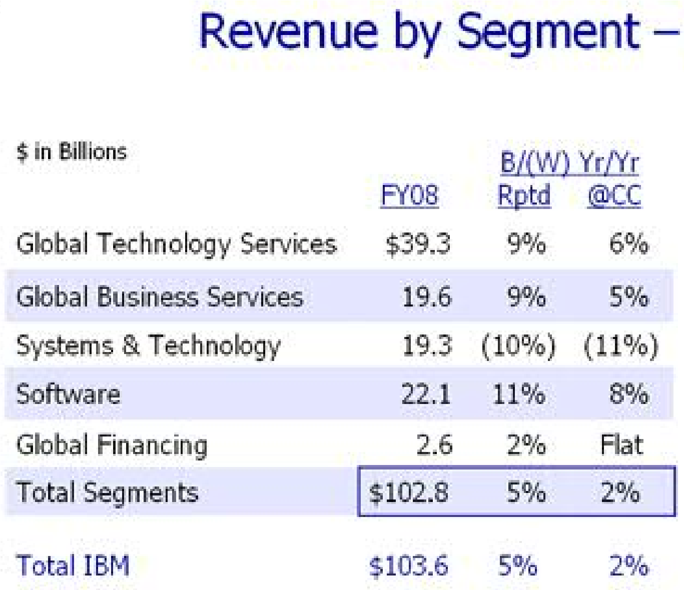
Reflections on ‘Engineering and Software’ by Michael Jackson

John Cameron
May 2009
cameronj@acm.org

ICSE 2009
VANCOUVER

Software-intensive systems are central to 2 of 5 IBM business segments: Software and Global Business Services (GBS). ‘Delivery excellence’ is one of GBS’s top-level management metrics.

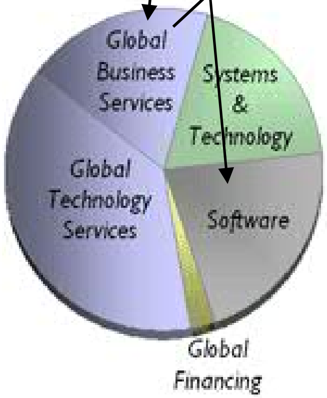
Two business segments are based on software-intensive systems



Revenue by Segment – FY 2008

$ in Billions

	FY08	B/(W) Yr/Yr Rptd	@CC
Global Technology Services	$39.3	9%	6%
Global Business Services	19.6	9%	5%
Systems & Technology	19.3	(10%)	(11%)
Software	22.1	11%	8%
Global Financing	2.6	2%	Flat
Total Segments	$102.8	5%	2%
Total IBM	$103.6	5%	2%



Same scorecard used at all levels of GBS

- One top-level metric is ‘Delivery Excellence’
  - on time
  - within budget
  - customer satisfaction
  - …
- Other top level metrics are as you’d expect: revenue, profit, signings, …
- All but impossible to score well on all metrics simultaneously
- Intense focus on any metric that is deemed to be poorly performing
  - intense focus on Delivery Excellence means that many managers and senior technical people are searching for ways to improve

The management process almost inevitably results in intense scrutiny on each headline metric at least every 3 or 4 years.

Do businesses like GBS think software engineering is important? Frankly, my dear, they don't give a damn!

- A $20bn business
  - Business improvement based on technology
    - with software-intensive systems
  - Good (very good) at it
- Intense scrutiny on 'delivery excellence'
  - = successful project delivery
    - time / budget / customer satisfaction
- BUT … they don't look to 'software engineering' for help with delivery excellence

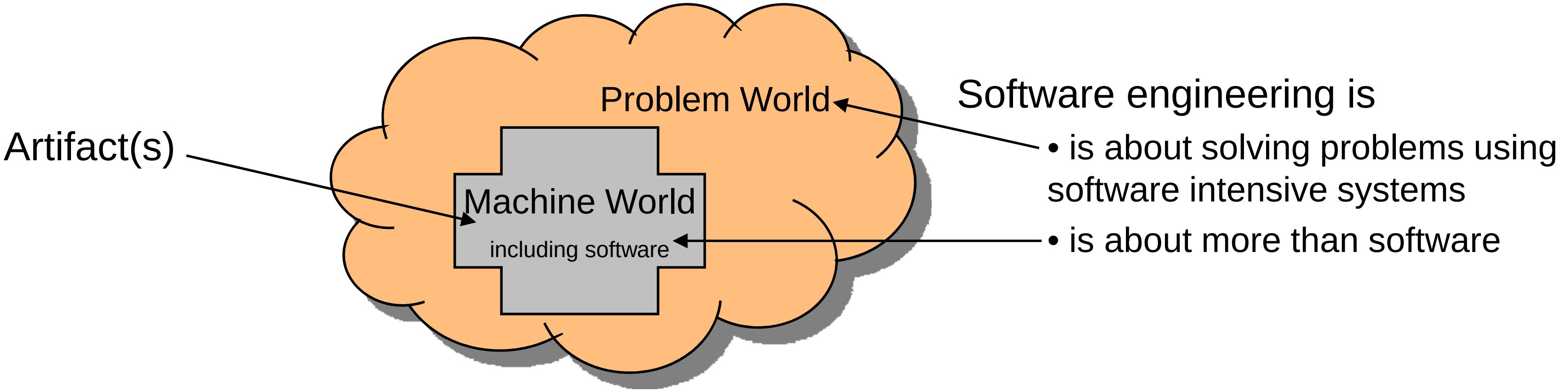
- Even though they clearly do 'software engineering'
    - requirements,
    - Design
    - Construction
    - Test
    - Maintenance
    - configuration management
    - process
    - Methods
    - Tools
    - Quality

      (the section headings in SWEBOK)

Software engineering is (should be) about much more than software

Engineering is ‘the practice of organising the design and construction of any artifice which transforms the physical world around us to meet some recognised need’

G F C Rogers, quoted in Michael Jackson’s ‘Engineering and Software’



Software engineering is

- is about solving problems using software intensive systems
- is about more than software

Requirements in the problem world are satisfied by the behaviour of a suitably interacting machine

Software Engineering (in journals, at conferences) isn’t interested in the problem world of business

No surprise then that GBS and its competitors are not much interested in ‘software engineering’

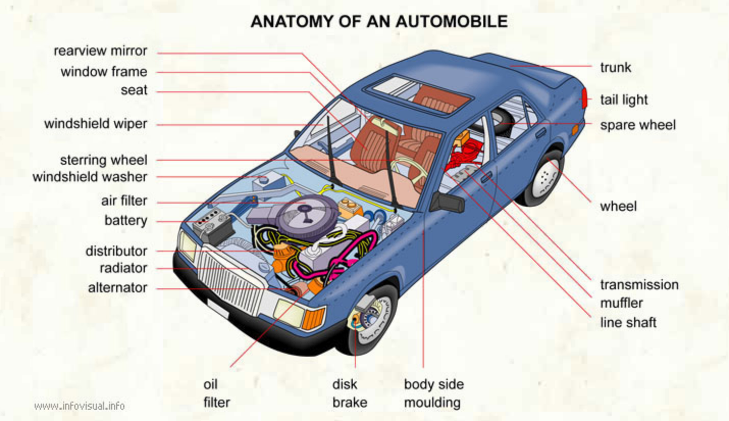
Traditional engineering practice is based on evolving 'normal designs'

from 'Engineering and Software', Michael Jackson

Radical Design

v

Normal Design



Specialisation

'In engineering the primary benefit of artifact specialisation is the emergence, adoption and evolution of *normal design*'

Normal Design

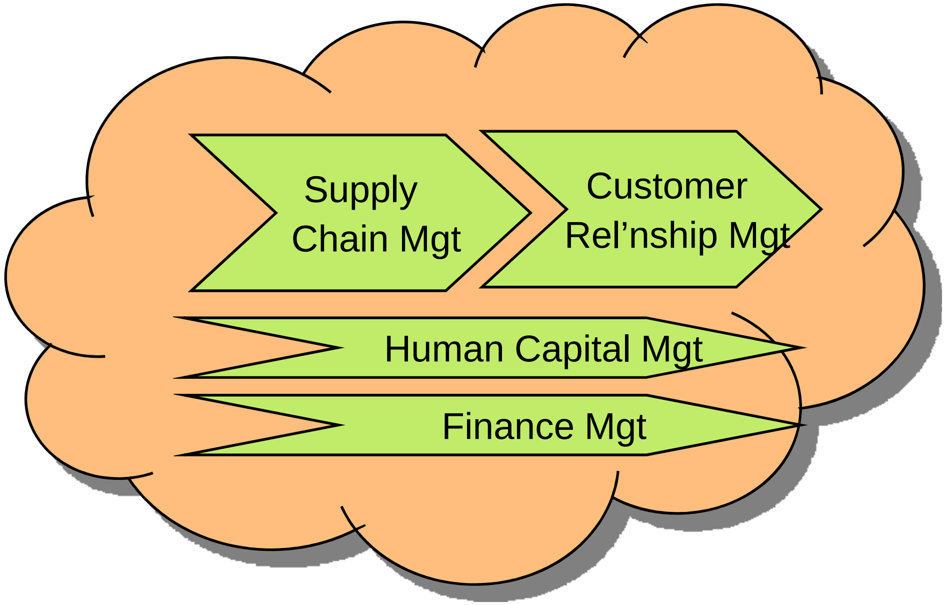
- specialisation, especially of artefacts (artefacts have their own problem world)
- defines its own requirements space
- comes with other assets, e.g.
  - estimating model

An aspiration for software engineering ?

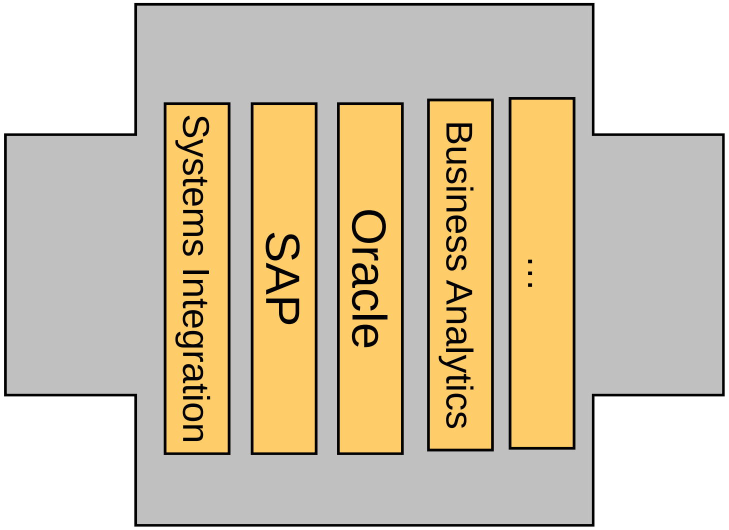
Hang on a minute … !
In a crude way we do this already – not as a future aspiration

Some organisational specialisations in GBS:-

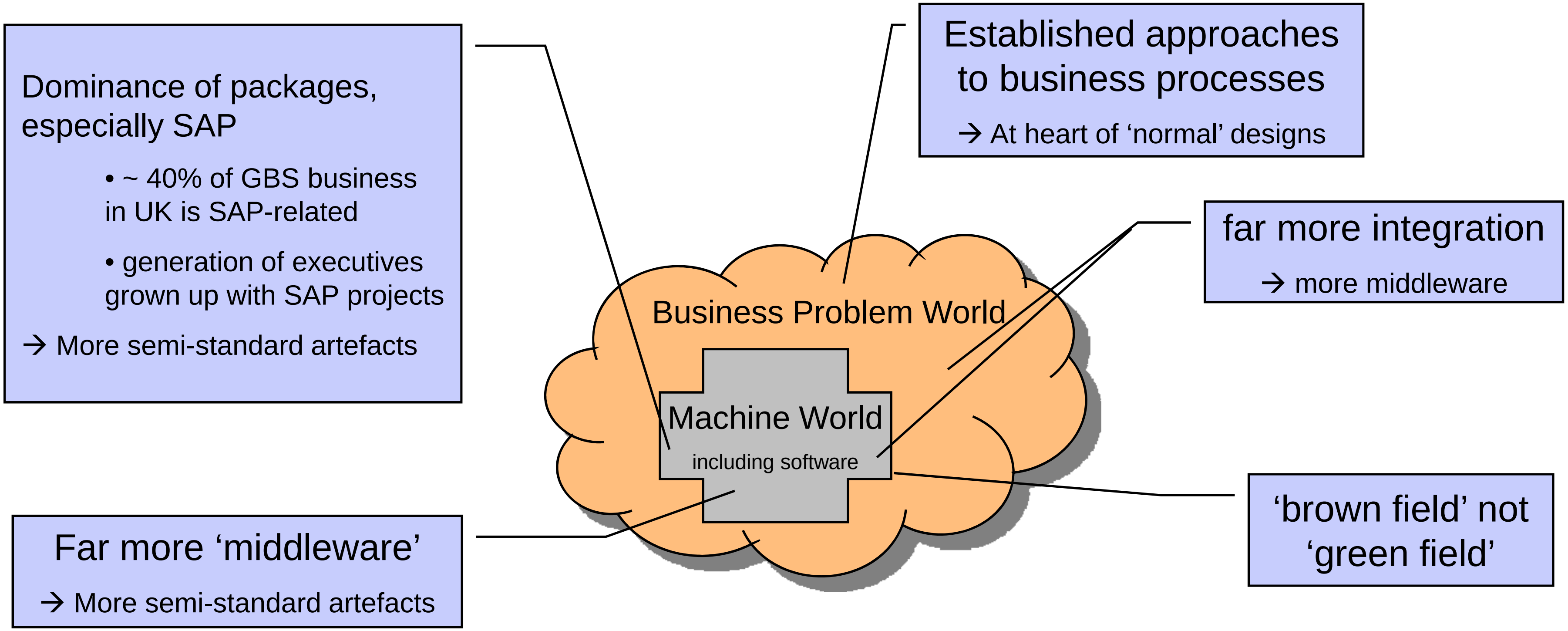
Problem World Specialisations



Machine World Specialisations



Changes in Business Systems Development in the last 20 years have encouraged specialisation and the development of ‘normal’ designs based on packages and middleware.



Dominance of packages, especially SAP

- ~ 40% of GBS business in UK is SAP-related
- generation of executives grown up with SAP projects

→ More semi-standard artefacts

Established approaches to business processes

→ At heart of ‘normal’ designs

far more integration

→ more middleware

‘brown field’ not ‘green field’

Far more ‘middleware’

→ More semi-standard artefacts

+ global supply chain for resources

→ encourages specialisation

+ emphasis on Systems Integration versus simple application development

→ encourages specialisation

Normal Designs come with their own Requirements

Scenario

- The engineer (architect) is disorientated
  - can't get started
  - 'a detailed requirements document is the prerequisite for any useful work'
- A few weeks later …
  - The engineer is quite content
    - working on the design
    - not even very curious about the requirements
  - Detailed requirements still not written
- Diagnosis
  - Disoriented only while uncertain which normal design to use
  - Once the normal design is known, the requirements are largely known

- A Normal Design defines a requirements space
  - Requirement work involves
    - Setting parameters, configurations
    - Fit-gap analysis - Variations, extensions
  - The variation/configuration is used where requirements would be used
    - Basis for: Development, Testing, Explanations to stakeholders, Training material, …

- Requirements somewhat demoted
- Higher level somewhat promoted
  - Scope
  - Intent
  - Precis

Requirements are not all they are cracked up to be

To summarise:-

1. Executives in consulting and systems integration businesses do not naturally and routinely look to 'software engineering' when they want to improve the performance of their projects

2. An obvious reason: 'software engineering' (textbooks, journals, conferences) is not concerned with the business problem world

3. Michael Jackson argues that:-
   - Software engineering will mature if and when it admits more specialisation
     - The key specialisation is by artefact
     - Specialisation involves a community of practitioners (engineers, architects, …)
   - Most engineering is concerned with normal (as opposed to radical) design
     - Normal design is only possible if there is specialisation
     - A project typically parameterises/configures/extends/varies a normal design

4. The people and organisations that de facto do 'software engineering' for business already organise themselves around specialisations
   - They already have a (pedestrian) realisation of Michael Jackson's vision
   - Specialise mainly by business process area and associated package artefacts, but also by middleware artefact

5. One consequence of 'normal design' is a reinterpretation of the role of 'requirements'
   - Many requirements come 'for free' with the normal design, which defines a restricted requirements space
     - + parameters, configuration, extension relative to the 'normal requirements' e.g. via a 'fit-gap analysis'
   - Above that
     - Scope (inc choice of normal design) is important
     - Precis is important (what you explain to senior sponsors becomes definitive)