PROBLEMS AND FUTURES IN SOFTWARE ENGINEERING

OR

accelerating the development of an engineering discipline

(An Iconoclastic view)

for HK-IVE

by

Karl Reed, MSc, FACS, FIE(Aust)
Director, ACS Tech. Board (CS&SE),
President, IWCASE(STEP) Board
Assoc. Prof. at La Trobe Univ.
This presentation is about need to systemically transform software development into an Engineering discipline.

We take as read

- the size and economic importance of software in the global economy..
  (its some number like US$500B....)
- the impossibility of a silver bullet,

Our objective ...take an engineering approach. benchmark.... identify, define goal,

SYSTEMATICALLY “CREATE“ THE DELTA
This Presentation will...

- Argue the current discipline of Software Engineering is doing well, but we need to refocus our goals

- Identify the "missing links", items of methodology etc. which are required for software development to complete the transition to Engineering.

- Propose steps which will address some of these problems,
Our Approach...

- We have a complex phenomena,

- We need a combination of short-term, near-term ad long-term goals,

- We identify two distinct issues:-
  - Methodological/technique
  - Philosophical

  What are the real differences between software development and engineering? (Baragry)

Karl Reed, La Trobe University
But First-some problems...

- design an email system which sends a message to a particular user,
- design a “tickler” system, which tells you that you should look at file on a particular date and time.
- Count the number of occurrences of different words in a text file...
- Given a text file of records containing suburb names and their postcodes, check that each entry is correct,
- Given a text file with records containing sales incidents in the form of a product code and the no. sold, produce a list of the total no. sold for each product code

We’ll return to these later.
Why Software Engineering?

☞ First proposed at the Nato Conference on SE in 1968

☞ Concern about the lack of order and discipline in s/w development.

☞ Belief that Engineering was better, and that engineers had more control over development.

☞ Tended to associate maths with engineering.

☞ Early definition …[Bauer, 1969]

“The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines” (not a good definition!)
Why Software Engineering? (cont’d)

Need for care....

Problems!!! Our understanding of Engineering is imperfect!

* A DANGER...we may classify current development practice as Software Engineering!
Firstly, what is Engineering?  

(Reed)

Engineers have

- problem-solving mind-set, using

- established techniques “derived” from theoretical disciplines…(but which often preceded them...)

- to design systems and artifacts capable of construction and maintenance …predictably & economically (iteratively)

Processes must be repeatable!!

Identical specifications should lead to recognisable identical systems.
MUCH ENGINEERING DESIGN KNOWLEDGE IS EMPIRICAL AND "RULE OF THUMB"

The differences between engineers and software developers…

Engineers explicitly…differentiate between…

- "problems" whose solution can be achieved using "prescribed" methods, and

- situations where these methods do not appear to exist..

ENGINEERS WORK WITH A DEFINED FRAMEWORK..

- Common, Coherent Universe of Discourse! (terms, methods, techniques) Reed-Baragry, and others)

- Theoretical basis of knowledge not always visible
“Engineers… design artefacts to interface with the real world…”

S/W developers… attempt to build models of real-world phenomena (Baragry1997)"

finally..

ENGINEERS DON’T BUILD SYSTEMS!!

the result of an “engineering” process is a set of design documents and plans which will be use by someone else of lesser training (but higher aptitude)

Compare with software development....
Engineering is...

- Design-reasoning explicit....
  - Choices in design cannot be made without "reasoning", which by its nature, is recorded,
  - Documentation is in fact "design reasoning recording"
  - Documentation is a by product of the process, NOT an external requirement by mandate

- Based upon "abstractions" validated by physical laws..
  - Constrains the solution space,
  - Provides definite failure modes,
  - Simplifies the process of developing a universe of discourse..

- None of this is true for s/w..

- Existence of physical laws is neither necessary nor sufficient for engineering properties..
Why do we need Software Engineering?

- Too much software being written… (>20B lines of new code per year?)
- "Maintenance" costs too high (40-80% of developers budgets)

We can’t build “evolving” software

Difficult to understand existing systems

- "Quality" of delivered code (errors, performance, reliability etc.) not always adequate (cf errors in a cpu chip)
- Software becoming pervasive, effecting all phases of our lives.

(gives a new meaning to "safety critical" and environmentally sound).

- In general, unable to predict performance of s/w.
- Have difficulty in validating development technology

Especially methodologies

- Measurement culture developing slowly

Karl Reed, La Trobe University

HK-IVE
Jan 2000
Why do we need Software Engineering? (cont'd)

Difficult to fix "appropriate" design scale!

(Do we need a six-lane freeway or a footbridge to cross the creek? In S/W, either will do!)
Why do we need Software Engineering? (cont'd)

Difficult to fix "appropriate" design scale!

(Do we need a six-lane freeway or a footbridge to cross the creek? In S/W, either will do!)

Is Software Engineering Possible?

Software development is a "theory building exercise" (baragry)

Some people believe that the inherent creativity of s/w development means that it cannot become "engineering".

(The texas board of engineers disagrees!)
How well are we doing?

- It is argued that computer designers and manufacturers do better than software developers...

Not so.....!!!!!

- Compare the purchase-cost of Delphi or Foxbase with a mainframe equivalent 20 years ago... (Jones)... reductions per unit of delivered end-user functionality of $10^2$ to $10^3$

- Extremely large complex systems, deployed with very large-scale usage,

- successful package, tool and “utility” builders around for >30 years

- A better comparison.. cost developing Windows NT vs design of and plant costs for a new Pentium (Reed)
The Importance of Software...

- S/W IS AN ENABLING TECHNOLOGY...
  increasing competitiveness of existing industry...
  creating new industries...

- S/W IS A MAJOR INT'L COMMODITY...
  WORLD MARKETS > US$80B
  Major national initiatives to promote product development and SE research in several countries...
  Japan...Europe...Singapore...
  Korea...Taiwan...

SOFTWARE PRODUCTION CAPABILITY A MAJOR ECONOMIC LEVER
WHAT IS SOFTWARE ENGINEERING?

THE IDENTIFICATION OF, DESIGN, CONSTRUCTION AND WHOLE OF LIFE CYCLE MAINTENANCE OF (SOCIALY) USEFUL SYSTEMS …using…

- Prescribed (validated) design and implementation techniques,
- Standardised components,
- Tools,
- Iteration of design to achieve and determine both functional and performance objectives,

…to…

- Predetermined resource and time constraints,
- Predetermined reliability, safety, quality and environmental standards

…in a…

- Well managed fashion. (Reed, after Fairley and with Zucconi)

Karl Reed, La Trobe University
Jan 2000
ENGINEERING

IS

- DESIGN REASONING EXPLICIT
- CONSERVATIVE
SOFTWARE ENGINEERING RESEARCH?

“The development of new techniques to improve Software Engineering Practice…” (Reed)

Improving SE research...

we need to...

➢ Focus on actual practice!...we know that methodologies are not followed

➢ Emphasise “validation”,

➢ Extend empirical studies

➢ Avoid “applied” computer science

➢ Use experienced researchers

In addition we need...

➢ More large-scale Industrial research centres (e.g. SEI, Fraunhofer)

➢ Agenda setting by commercial developers, the s/w industry, and computer industry
SOFTWARE ENGINEERING RESEARCH?

Over optimistic sometimes.

CASE, integrated program development environments …

STRUCTURED PROGRAMMING
More clarity with terminology...

E.G. “process” its relationship to improving development:-

A product relates to “technology” in five ways.. the technology

a. embodied by the product, 
   Its capabilities, in function, performance etc.

b. of its components, 
   the parts, materials, sub-systems

c. used in its production, 
   tools, design techniques

d. the human skills used in production,

e. used in production process 
   organisation of production, process
ISSUES TO BE TACKLED...

- CODIFICATION OF (EXISTING) KNOWLEDGE (ESSENTIALLY EXTENDED REUSE OF DESIGN AS WELL AS COMPONENTS)
- VALIDATION OF DESIGN PROCEDURES
- DEVELOPMENT OF VALIDATED CONSTRUCTION PROCEDURES...leading to maintainable systems...

INCORPORATING

- IMPLEMENTATION RESOURCE AND TIME ESTIMATION
- RUN-TIME RESOURCE UTILIZATION AND PERFORMANCE PREDICTION
- REPEATABILITY AND TRACE-ABILITY OF DESIGN AND CONSTRUCTION

BASED UPON
STANDARDS AND ACCEPTED PRACTICE
FOLLOWED BY
- TECHNOLOGY TRANSFER
Revisiting our earlier problems

▷ design an email system which sends a message to a particular user,

▷ design a “tickler” system, which tells you that you should look at file on a particular date and time.

▷ Count the number of occurrences of different words in a text file...

▷ Given a text file of records containing suburb names and their postcodes, check that each entry is correct,

▷ Given a text file with records containing sales incidents in the form of a product code and the no. sold, produce a list of the total no. sold for each product code
Some interesting problems…


2. Design "Prescriptions"

3. Implications of "maintenance" and extendability for implementation techniques,

4. S/W reuse and its impact on methodology,

5. Component-Based design

6. Philosophy of "design" and "architecture"

7. Philosophy of "software development” and relationship to engineering

Validated methods, tools and languages etc. are required!
Some interesting problems, and current status…
(cont'd)

   There is a gap between theory and practice…and the sanitised versions of methodology

Consider the so-called "waterfall model", ascribed to Royce (1970).

There are two problems…
   A. Is that what Royce actually said?

   B. What are the implications of the model for s/w development?
Assumptions...

1. information produced by Pi-1 is adequate for completion of Pi,
   Implies types of diagrams, documents, text etc. reflect the intellectual material needed for completion of the next phase.
   Some research, and anecdotal evidence contradicts this.
   
   e.g. It is believed that productivity is high when ONE developer handles analysis thru unit-test.

Implies...
   Need for improved diagraming and design-recording approaches.
Waterfall Model... problems...

2. phases can only be performed in order,
3. no phases can be performed in parallel

Implied by 1., but, assumes no knowledge acquisition is required at $P_j > i$ which may be independent of $P_i$. If they need not be performed in order, then may be they can be performed in parallel.

Experience shows that major problems can exist which are largely independent of the application specification, being heavily embedded in the implementation domains.(this has implications for a number of areas, e.g., s/w architecture, project design, etc.)
Waterfall Model... problems...
Assumptions...

4. all decisions made at Pi<j remain valid when Pj is reached,

Implies...

No knowledge down stream of Pi is needed by Pi, OR...

If such knowledge exists, it will be apparent at Pi (see problem 1.)

Experience shows that this simply not true, and, that in the worst case, the original requirement for the system may be invalid by the time it is complete!
Some interesting problems, and current status... (cont'd)

**Current developments in S/W process modelling...**

Boehm's spiral model...

Osterweil's process programming(?)...

Prototyping...(not new, see Royce)...

Hesse “embedded diamond”

**Remaining issues...**

*Project Plan tailoring, i.e., how does one select a process model, and tailor it to a particular project?*

**Actual process models used in the real world?**

AN INTERESTING ISSUE .... THE UNIQUE NATURE OF SOFTWARE DEVELOPMENT VS THE UNCRTITICAL ADOPTION OF ENGINEERING MODELS(Baragry/Cleary/Reed)?
Software Development vs Engineering...The Uncritical use of Analogues

How much of our view of s/w development is pre-conditioned by our desire that it be like engineering. (Cleary/Reed)

In Engineering, construction dominates, In S/W, design dominates (Reed/Cleary)

Above impacts our view of project planning (Cleary/Reed)

Impact of philosophical differences on in underpinning concepts and approaches (Baragry/Reed)

Impact of intellectual process in s/w development on process (Reed, McDermott)

May be we are wrong! (Baragry etc.)
Some interesting problems...(cont'd)

2. Design "Prescriptions"

Existing methodologies lack detailed procedures for moving from one "level" or "phase" to another.

E.G. expanding a Data Flow Diagram to show more detail, or, a State Transition Diagram.

The processes are "arbitrary" in the sense that there are many acceptable solutions, determined by developer's skill, prejudice, etc.

Other Engineering Disciplines require calculations, use workbooks, standard components & practice etc., restricting the number of possible implementations!

E.G. the design of a flange to join pipes of known diameter carrying gases at known pressure and flow.

which explains why no-one documents design...
Some interesting problems, ... (cont'd)

2. Design "Prescriptions"

Current developments...

§ Slow recognition of this fact, but, little explicit method. (Some in testing, O-O, for inheritance, but, otherwise little activity...)

§ Smith & Williams are working towards "engineering" approaches to performance engineering.

Future Research

Identify missing links, develop prescriptive validated methods
An aside…

Some research suggests that methodologies are not actually followed in practice…(Curtis, Siddiqi, Silverman)

If this is true, then…

- Documentation occurs after the fact,
- It cannot reflect the design process that actually occurred…
- It will not aid maintainers (hence re-engineering)
- Implementers will not actually document design until they cannot complete the design without the documentation…

In other words,

when the design process can only be performed with calculations, explicit decisions, etc. then it will be documented!
Some interesting problems, ...(cont'd)

3. Implications of "maintenance" and extendability for implementation techniques,

⇒ S/W is extensively modified…

§ Specifications change,
§ Customer requirements change,
§ Delivery platforms change…

To address this, we need…

DESIGN FOR MODIFIABILITY

DARPA INITIATIVE IN “EVOLVING SYSTEMS”
Some interesting problems, ...(cont'd)

3. DESIGN FOR MODIFIABILITY

IN THIS CASE, THE TECHNOLOGY EXISTS, BUT NEEDS TO BE FORMALLY CONSOLIDATED INTO A PRESCRIPTIVE METHOD!

Technology available includes…

§ Table-driven,
§ Information hiding,
§ Exception handling,
§ "Come-From" statements…
§ State Transition Diagrams,
§ Interpretive techniques,
§ Application generators using compiler-writing techniques,
§ Use of error handling and “defensive” programming as points to extend systems
§ Message-switching architecture as basis for evolution
Some interesting problems, ...(cont'd)

3. DESIGN FOR MODIFIABILITY

TO ACHIEVE THIS, WE CAN...

- Seek possible generalizations of function,
- Fabricate functions from common modules, using table driven approaches,
- Imbed logic in tables,
- Reverse the “contraction of generality”(Reed)

ETC, ETC.

IN THIS CASE, WE HAVE A PROBLEM AND THE "CRUDE" TECHNOLOGY TO SOLVE IT...

ALL WE NEED IS CONSOLIDATION AND INTEGRATION
Some interesting problems…

4. S/W reuse and its impact on methodology,

PROBLEMS…

§ Current methodologies do not generate reuse-able components.

This requires iterative design and "late commitment" to component choices…

§ Current research tends to focus on sophisticated technology for reuse library management.

Experience with other "reuse" based disciplines suggests that simple, widely known libraries of components and designs UNSUPPORTED BY COMPLEX TECHNOLOGY work very, very well
Some interesting problems...

4. S/W reuse and its impact on methodology,

PROBLEMS and we are making progress...

§ Need for frame-works for reuse, "plug-in" modules for s/w racks and frames...

**Concepts exist in S/W bus and MLI's**

§ Organisational frame-works needed...

See Basili's Experience
"factory" ("studio" is better. cf, "Software Factory", a poor term)

§ reuse MUST extend to complete and partial designs... **There should be standardised designs, re-usable in systems development. Evidence is that these do in fact exist.**

§ "Domain-isomorphism" is important

Architectures, designs etc. which are reusable in different application domains are possible-and, implementations

Karl Reed, La Trobe University

HK-IVE
Jan 2000
The ultimate goal here is "graphical" system composition...

Graphical icons representing components are "clipped" in to architecture/program diagrams to produce a system.

This is not really new, we do it with PC's and Mac's all the time???

Work is in progress in this area, e.g. program visualisation.
Some interesting problems…

4. S/W reuse and its impact on methodology,

There is significant work in this area, but, it still needs consolidation…
Some interesting problems...

5. Component-Based design

This is the other side of reuse, but addresses a different issue...

Design procedures assuming the prior existence of standardised components e.g. COTS...

Difficulties are considered to exist here...

- Absence of physical laws forcing component nature...software is infinitely malleable...

- Absence of “hard” encapsulation ... s/w is (almost) infinitely accessible...

- Absence of speed/size benefits... s/w libraries (this need not be true...less of an issue with modern cpu's)

- **ABSENCE OF A COMPONENT-BASED DESIGN EDUCATION**
  (warrants some explanation)
Some interesting problems...

5. Component-Based design

Addressing these issues...

S/W components ...

- can and are being developed...
  commercial component libraries are now being marketed... C++
  (also subject of US Govt. funded research)

- net-sourced components(?)

- can be treated as immutable

- It is possible to train people to think in terms of component-based design... this is done for electronic engineers

- (component libraries were common in the late 1960's. Where did we go wrong?)
5. Component-Based design (cont’d)

Hard-encapsulation” vs “modifyable components…

Issue of s/w vs engineering...

Component based engineering design tends to assume immutable components…but s/w components are infinitely malleable…the intersection. is predictable and controlled malleability in components(Reed).

MANY OF THE ISSUES MENTIONED SO FAR ARE RELATED TO THE GENERAL ISSUES OF "REUSE" AND TO DESIGN…
Some interesting problems…

6. Philosophy of "design" and "architecture"

There is a deeper issues here…

What constitutes design in S/W?

Current "design" paradigms are primarily "working-result-oriented", i.e., so long as it works, the design is considered successful.

Constraint-based design (performance, reliability, quality, safety etc.) leads into uncharted territory.

How does the concept of architecture "fit" into s/w design?

(What is this anyway? )

Cox proposes a "product-centric design" paradigm, based on reuse.

Basili advocates an experience factory, re-using design at various levels… and goal-based adaptation of methodology (cf GQM)

Reed advocates prescriptive design, "reasoning explicit"
6. Philosophy of "design" and "architecture"

"Architecture" seems to involve...

- Standard arrangements of components...(Shaw identifies 5)

- Concentration on aesthetic and functional requirements NOT performance, primarily

- Rules for arranging components when standard architectures do not apply.

- Can be influenced by “non-functional” requirements

- Based on poor analogy with "built" architectures (Baragry/Reed)
6. Philosophy of "design" and "architecture"

Various levels of reuse of design (cf "ordinary" architecture)

We can identify a number of levels of component design which …

§ is "completed", hence is not performed, and has no effect on the final system.

§ is known to be achievable, hence incompleteness is irrelevant, but may impact final system.

§ is known to be achievable, but may need to be completed to ensure final system is "correct".

§ is not known to be achievable …cf Sydney opera house.

This can be understood easily in terms of standard building architecture.
6. Philosophy of "design" and "architecture"

Some observations...

§ The concept of “architecture” in the traditional sense may not be appropriate for software (Baragary)

§ Systems may have multiple “architectures” of a system implemented as a message switch...

§ It may not be possible to determine architecture in advance(Reed)

§ It may be possible to “retrofit” and architecture(Reed)

The problem of S/W architecture.

§ Derived from the "built" architectures

§ Multiple architectures exist for a single system.. some call these views (?)

§ Architecture is not visibly discernable (Baragry)

§ Is it static, dynamic (?)
Additional major technical problems to be solved...(we need engineering methodologies, NOT simply new theory!)

- "Powerful" formal methods, cf Laplace xforms v DE's
- Improved Diagraming Techniques,
  § Diagrams should be executable, cf. circuit diagrams etc.
  § Diagrams should reflect nature of (most) problems... async. communicating. parallel processes.
- "Efficient" test techniques
- "Testability" driven implementation techniques
  § Reduction in logical complexity of programs, perhaps through separation of control and data
- "Consistent" design-implementation test methodology...STD's suggested by Zucconi and Reed
Impact of poor studies of process and design procedures...

Problems with early CASE?

A pessimistic view...

- Automated methodologies which may not work... preventing work arounds.

What might come from detailed studies of existing practice...

- Knowledge of existing design procedures,

- Precise data on the useful knowledge used in the transition from one phase to the next, hence, knowledge of the design documents which are useful,

- Discovery of new process models, and rules for choosing them,

AND, perhaps, the design of new tools, based on genuine CAD, implementing prescriptive procedures!

Karl Reed, La Trobe University
Issues to be addressed...

- absence of undergraduate s.e. education, and relevant "mind-set".

- academic appointment and promotion rules need to recognise experience for teachers

- find ways of tackling research issues arising from industry’s needs
Leveraging the current state of research...

- STUDY OF EXISTING PRACTICE AND ARTIFACTS

- STUDY OF PAST PRACTICE AND ARTIFACTS

- ACCESS TO TOOLS  *(very expensive...)*

- DEVELOPMENT OF EXPERIMENTALLY VALIDATED ..

  § PROGRAMMING STYLES AND LANGUAGES

  § DESIGN AND CONSTRUCTION PROCEDURES

  *these include diagraming systems, documentation techniques, tools, libraries, etc.*

  *(FOLLOWING CCITT AND U.S. MILITARY AIRCRAFT PROCUREMENT MODELS)*

- APPROPRIATE EDUCATION
Issues to be addressed...

THE SOLUTIONS TO THESE PROBLEMS REQUIRE....

A NEW DISCIPLINE.....

SOFTWARE ARCHEOLOGY??

"WE ARE DISCARDING VALUABLE KNOWLEDGE AT AN ALARMING RATE, IGNORING THAT WHICH MAY BE USED BY FUTURE GENERATIONS, SIGNIFICANTLY RETARDING OUR RATE OF PROGRESS (Reed)"

"What do the rubbish bins of the software shops of the world reveal?" (ibid)

also.. Software Engineering Empericism... much of what we need may actually already exist
Leveraging the current state of research...

- LEVEL OF EXPERIENCE OF RESEARCHERS
  - "DEEP" PROBLEMS REQUIRING EXTENSIVE EXPERIENCE AND KNOWLEDGE AS WELL AS ABILITY
  - CONSOLIDATION OF KNOWLEDGE, A TASK FOR EXPERIENCED PERSONNEL
  - NATURE OF TRAINING OF NEW GRADUATES

- LEVEL OF EXPERIENCE OF SUPERVISORS
  - IN-DEPTH S/W CONSTRUCTION EXPERIENCE..NOT CONTINUOUS UNIV. BACKGROUND (we'll explore this further)
DIFFICULTIES FOR RESEARCHERS (CONT)

⇒ SCALE OF PROJECTS NEEDED IN MANY AREAS

• MOVEMENT OF RESEARCH AREAS INTO INDUSTRIAL DOMAIN MEANS ACADEMIA MAY NOT BE ABLE TO COMPETE, (e.g. CASE, Hypertext)

• CONTROLLED EXPERIMENTS ON LARGE SCALE PRODUCTION NEEDED TO VALIDATE CONCEPTS (e.g. design procedures, construction techniques, managerial, estimating, metrics, etc.)
Leveraging the current state of research...

 SCALE OF PROJECTS NEEDED IN MANY AREAS (cont)
   • VALIDATION OF SYSTEM STRUCTURE PROPOSALS (distributed, network, o.s., etc)

 MULTI-DISCIPLINARY STUDIES NEEDED
   • METRICS, ACTUAL SYSTEM DEVELOPMENT PRACTICE.

SUMMARY...PROGRESS ON MAJOR PROBLEMS REQUIRES SOME REVISION OF CURRENT ACADEMIC AGENDAS

.. FUND INDUSTRIAL STRENGTH RESEARCH, INSIDE ACADEMIA AND OUT...

.. NEW DOCTORAL DEGREE D.S.E. (?)

.. MORE UND. GRADUATE S.E. DEGREES
Conclusion

WE HAVE IDENTIFIED...

- General properties of Engineering Discipline Identified as applied to SE,
- Specific missing links needed complete the transition to an Engineering Discipline,
- Factors inhibiting progress,
- Absence of relevant research, except for NASA/SEL and a few others,

And noted...

HOW DO WE PROCEED?
the way forward…

- Establish s/w development process monitoring and improvement programs, with developers and academia,
- Set research programs aimed at specific problems from those issues listed,
  (recognize applied nature of the research, and the specific goals being set.)
- Create opportunities for experienced (20yr) vets to undertake research,
- Study the relationship between engineering and s/w development
- DEVELOP UNDERGRADUATE SE PROGRAMS, DETERMINED ENTIRELY BY AN ENGINEERING MIND-SET!
Thank You...