Application Case Study: Introduction

6.871-- Lecture 4
Questions About ...

• The Task
  – Is this the right problem to solve?
    • Is it important?
    • Is it valuable?
  – Can it be done?
  – How can progress be measured?
  – How will you know if it succeeds?
  – If you build a system, will anyone use it?
    • Who, and why?
Questions About ...

- The Task
  - If you build it, who will maintain it?
  - If you build it
    - Who will benefit from it?
    - Who will be threatened by it?

- The Technology
  - What can it do?
Knowledge Based Systems Can

• Replicate knowledge and expertise
  – *If only we had 5 more of Sally*…
Knowledge Based Systems Can

• Preserve knowledge and expertise
  Corporate Memory
    – Joe’s getting ready to retire.

• Embed knowledge and expertise
  – Is it #*1 or ##2 to call-forward?!!
Knowledge Based Systems Can

• Make knowledge accessible
  – Oh, HERE it is, on page 412 of volume 6.
Knowledge Based Systems Can

• Apply knowledge consistently over time
  Provide an environment for knowledge standardization and growth
  – Why can’t they do it in Chicago the way they do it in Seattle?
  – Why does every plant have to keep re-learning this?
  – E.g. American Express Authorizer’s Assistant
Knowledge Based Systems Can

• Leverage the expert
  Why can’t we use Phil’s time more productively?

• Improve practice; support the average
  We can never find and train enough skilled people.
Knowledge Based Systems Can

• Help avoid disaster.  
  *How did that slip through?*

• Help manage change? 
  *Fifty new products this year! A technical success, and a marketing disaster.*
Knowledge Based Systems Can

• Distribute corporate policy
  *Why don’t the salesman read any of the 100 memos we sent this quarter?*

• Solve a variety of “part assembly” tasks.
  *I can’t keep track of all the combinations.*
Analysis: What Is It?

• What is the task?
  – Specify in terms of input and output.
• When is it done and why?
• How often?
• How fast must it be done?
• How much does one “run” cost?
• What value is produced by a run?
Analysis: How Is It Done?

• Who does it?
• What do they do?
• How do they get trained?
• How available are they?
• How is the task organized?
• How accurately should it be done?
• What goes well about it now?
• What goes badly?
Analysis: Mistakes

• What is the nature and origin of a mistake?
  – What kinds of things go wrong?
  – Why?
    • too much detail
    • too much change
    • too much info to absorb
    • insufficiently trained people
    • too simple
    • too routine
Analysis: Mistakes

• What are the consequences of a mistake?
  – time: how much?
  – money: how much
  – image

• If something goes wrong now?
  – who spots it
  – who fixes it
  – who gets blamed
The Technical Case

- Character of the problem
  - Narrow domain of application
  - Knowledge overload
    - Many different outcomes
    - Few outcomes but a lot to know
  - Task involves symbolic reasoning
  - Task uses symbolic information
  - No adequate algorithmic solution
  - Takes 20 minutes to a few days
  - Incremental progress is possible
  - Repetitive
The Technical Case

- **Character of the knowledge**
  - Substantial specialized knowledge/expertise required
    ⇒ accumulating relevant knowledge takes time
  - Knowledge is *relatively* stable
  - There are recognized experts
  - … but too few of them
  - … or they have other tasks that are more rewarding
    (for several senses of reward)
The Technical Case

• **Character of the knowledge**
  – Experts are provably better than the amateur
    • Measure the difference
      – What dimension: speed, accuracy?
      – What is the right answer?
      – The experts can communicate the relevant knowledge
    – They can communicate it to you
      • You can become at least a talented amateur
    – One expert is enough (or, one chief expert)
The Technical Case

• Character of the solution:
  – useful accuracy is reachable

• The skill is routinely taught

• Data and cases studies are readily available
  – Dead center cases
  – Extreme cases
  – Informative canonical cases
The Business Case

• Define the character of the payoff
  – revenue
  – improved competitive position
  – quality
  – speed
  – uniformity
  – cost reduction
  – new, different product
  – staff retention
  – staff reduction
The Business Case

• Calibrate the size of the payoff
  – What is half the distance to the expert worth?

• Determine the chance for leverage
The Organizational Case

• An enthusiastic, committed expert is available
• Who will use it?
• End-users are identifiable/identifiable
• End-users are enthusiastic
  – Do they agree that
    • the problem exists?
    • the problem is important?
    • the program solves their problem?
The Organizational Case

• The organizational culture will support its use

• The answer is worth the difficulties
  – learning to use it, using it
If It’s The **First** Problem

- Select one where knowledge is fairly clear
  - Needs formalization, not discovery
    eg. Procedures, manuals, etc.

- Select one that’s too small

- Select one that matters

- Set up a skunkworks
Project Design

Expert-level performance is difficult, so...

- Adopt an evolutionary approach
  It gets you started
  Useful wherever you stop
Project Design

• Build an assistant
  Inherently low profile
  Leverages the operator
  Keeps lines of accountability clear

• Manage expectations

• Provide a smooth adoption path

• Provide follow-on and support
Project Construction

• You don’t know what you’re trying to build
  Recall checkbook vs. supermarket
  – Not formally definable
  – Can’t anticipate all contingencies
    • Can’t specify procedure
  – Human performance is the metric
  – The task will change out from under you
Project Construction

• Nature of the solution changes

• Nature of the construction process changes
Rapid Prototyping

• Construction process involves
  – Intertwining of specification and implementation
  – Experimentation
  – Three-month prototype
    • prevents optimization
    • encourages experimentation
    • early feedback on technology and conception

PROTOTYPE ⇔ ENHANCE ⇒ SPECIFY ⇒ CODE
Rapid Prototyping: Advantages

• Handle ill-defined tasks.
• Check problem conception.
• Secure user buy-in.
• Refine user requirements.
• Refine production and integration requirements.
• Something works all the time.
• Get management support.
• It happens anyway.
Field Test and After

- Where to field test
  - Who wants it and is knowledgeable enough to evaluate it?

- KB development is never done
  - Determine who can take over

- What will happen to the expert?
  - attrition?
  - work on harder problems?
  - extend the knowledge base?
Design for Evolution

• If it’s a success, how long will they use it for?
• If they use it, what else will they want?
• What do you suspect will happen to the hardware and software infrastructure that the application will rely on?
• How closely coupled to the underlying infrastructure will you need to be?
  – Will they let you do that?
  – Are there standard ways to do it?
  – How pervasive will these be in the end application?