We need better ways to analyze a software design and predict the value its implementation will offer to a customer or to its producer.
Predicting Value from Design

Engineering design

Engineers . . .

- iterate through design alternatives
- reconcile client’s constraints
- consider cost & utility as well as capability
- recognize that early decisions affect later costs

... but ...

... but ...
Engineering design

> Engineers . . .
> - iterate through design alternatives
> - reconcile client’s constraints
> - consider cost & utility as well as capability
> - recognize that early decisions affect later costs

. . . but . . .

> Software engineers . . .
> - lack adequate techniques for early analysis of design
> - design for component spec rather than client expectation
> - rarely include cost as 1st-class design consideration

Why does early design evaluation matter?

> Cost of repair
> - Fixing problems after delivery often costs 100x more than fixing them in requirements and design
> - Up to half of effort goes to avoidable rework
>   - “avoidable rework” is effort spent fixing problems that could have been avoided or fixed earlier with less effort
> - Early reviews can catch most of the errors

-- Boehm/Basili, IEEE Computer, 2001
Predicting Value from Design

Why does early design evaluation matter?

- **Cost of repair**
  - Fixing problems after delivery often costs 100x more than fixing them in requirements and design
  - Up to half of effort goes to avoidable rework
    - “avoidable rework” is effort spent fixing problems that could have been avoided or fixed earlier with less effort
  - Early reviews can catch most of the errors

... but ...

- **Confidence in estimates is lowest early in a project**

---

Barry Boehm, Institute for Software Research, International

Mary Shaw 10/5/2005
Predicting Value from Design

Why does early design evaluation matter?

- Cost of repair
  - Fixing problems after delivery often costs 100x more than fixing them in requirements and design
  - Up to half of effort goes to avoidable rework
    - “avoidable rework” is effort spent fixing problems that could have been avoided or fixed earlier with less effort
  - Early reviews can catch most of the errors

... but ...

- Confidence in estimates is lowest early in a project
- Early decisions commit most of the resources
Costs, commitment, and uncertainty

Engineering involves deciding how to make irreversible commitments in the face of uncertainty

Risk-aware view: costs committed to date
Usual view: cumulative costs incurred to date

Current software design evaluation

Relatively little attention to early design evaluation
Software-centric evaluations
Minor role for costs other than development
Sparse, scattered, inconsistent evaluation methods
Predicting Value from Design

Current software design evaluation

- Relatively little attention to early design evaluation
  - Leverage lower cost of change during design
- Software-centric evaluations
  - Consider user-specific preferences, or perceived value
- Minor role for costs other than development
  - Expand role for larger-scale economic issues
- Sparse, scattered, inconsistent evaluation methods
  - Find ways to use models together

What needs to be done?

- Make early predictive design evaluation viable
  - Identify existing techniques that apply early
  - Explain them in a consistent way
  - Determine how to compose them
  - Develop new techniques
- Provide a unifying model
  - Be explicit about interfaces
  - Be clear about method and confidence
- Support it with tools
Economists’ view of value

A firm’s goal is typically to maximize total revenue minus cost of the inputs, represented by
\[
\max \left[ (B(z) - C(y)) \right] \text{ such that } F(y, z) \leq 0
\]

Here
- In vector \( z \), \( z_j \) represents quantity of product \( j \) sold
- \( B(z) \) is the total revenue from selling those products
- In vector \( y \), \( y_i \) represents quantity of input \( i \) consumed
- \( C(y) \) is the total cost of those inputs
- \( F(y, z) \) is a vector, as well, so \( F(y, z) \leq 0 \) represents a list of equations representing constraints on the problem
Early, code–free, design evaluation

- Target of evaluation
  - very high level design, before “software design” methods start elaborating the box and line diagrams
  - evaluation that weighs costs as well as capabilities
  - evaluation that recognizes user needs and preferences
  - evaluation that does not depend on access to code

- Long-term objective: framework to unify models
  - general, to handle models for various specific attributes
  - open-ended, esp. with respect to the aspects considered
  - flexible, handling various levels of detail and precision

Model for predictive analysis of design

\[ U(d, \theta) = B(x, \theta) - C(d, x, m) \text{ for } \{ x : F(d, x, m) \}, \text{ where } x = P(d, m) \]

Value \( U \) of design \( d \) to a client with preferences \( \theta \) is benefit \( B \) net of cost \( C \) provided the desired result \( x \) is achievable and attributes \( x \) of implementation are predicted by \( P \)

Let
- \( d \) be a design in some appropriate notation
- \( x \) be in \( A^n \) an open-ended vector of capabilities
- \( v \) be in \( V^n \) a multidimensional value space
- \( m \) be in some notation a development method
- \( \theta \) express user pref a multidimensional utility space
- \( B \) express benefits predicted value \( v \) of \( x \) to user with pref \( \theta \)
- \( C \) express costs cost \( v \) of getting \( x \) from \( d \) with method \( m \)
- \( F \) checks feasibility whether \( d \) with \( x \) can be achieved with \( m \)
- \( P \) predicts capabilities attributes \( x \) that \( m \) will deliver for \( d \)
Predicting Value from Design

Plan

- Role of early design evaluation
- Model for predictive analysis of design
- Techniques for predicting value from design
- Framework for composing and comparing the techniques
- Scenarios for use
- Open problems

Basic value proposition

\[ U = B - C \]

Following economics, value is benefit net of cost

Adopting a software tool will cost $X, and it will save you $Y, right away, on your current project.

\[ U = Y - X \]
Value based on product attributes

\[ U(d) = B(x) - C(x) \]

The value of a design is the benefit, net of cost, of the implementation as represented by its capabilities.

Let \( d \) be a design in some appropriate notation
\( x \) be in \( \mathbb{R}^n \) an open-ended vector of capabilities
\( v \) be in \( \mathbb{R} \) value in dollars

\( B \) express benefits predicted value \( v \) of \( x \) to user
\( C \) express costs cost \( v \) of getting or using \( x \)

Example 2: Choosing a representation

- You store maps to view and edit in drawing package
- Only 1 of every 50 reads leads to a write
- Cost: $10K per sec read/write, $0.1/KB storage
- You get data for your typical data sets:

<table>
<thead>
<tr>
<th>File type</th>
<th>Seconds to open (read)</th>
<th>Seconds to write (save or export)</th>
<th>File size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>6</td>
<td>93</td>
<td>6243</td>
</tr>
<tr>
<td>EMF</td>
<td>9</td>
<td>88</td>
<td>17908</td>
</tr>
<tr>
<td>EPS</td>
<td>5</td>
<td>17</td>
<td>20909</td>
</tr>
<tr>
<td>PDF</td>
<td>7</td>
<td>95</td>
<td>6243</td>
</tr>
<tr>
<td>WMF</td>
<td>5</td>
<td>86</td>
<td>11038</td>
</tr>
</tbody>
</table>
Ex 3: Determining value of features

- For spreadsheets,
  - Adherence to dominant standard ➔ 46% higher price
  - 1% increase in installed base ➔ 0.75% increase in price
  - Quality-adjusted prices over 5 years declined 16%/year
- Hedonic model a good predictor
  - Hedonic model estimates value of product aspects to consumer’s utility or pleasure; it assumes price is a function of product features

Econometric analysis of spreadsheet market, 1987-92
Predicting Value from Design

Predicting attributes from design

\[ U(d, x) = B(x, q) - C(x, q) \]

where \( x = P(d) \)

We often need to predict the implementation properties \( x \) before the code is written.

Let \( d \) be a design in some appropriate notation

\( x \) be in \( \mathbb{R}^n \), an open-ended vector of capabilities

\( v \) be in \( \mathbb{R} \), value in dollars

\( B \) express benefits, \( C \) express costs

\( P \) predicts capability \( \frac{d}{x} \) of implementation of \( d \)

Ex 4: Predicting size from function points

COCOMO Early Design

- Examine design to count function points

<table>
<thead>
<tr>
<th>Type</th>
<th>Complexity Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Internal logical files</td>
<td>7</td>
</tr>
<tr>
<td>External interface files</td>
<td>5</td>
</tr>
<tr>
<td>... etc ...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Choose programming language
- Use pre-calibrated table to estimate code size

<table>
<thead>
<tr>
<th>Language</th>
<th>Ada 95</th>
<th>C++</th>
<th>Java</th>
<th>PERL</th>
<th>VB 5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC per Fcn Pt</td>
<td>49</td>
<td>55</td>
<td>53</td>
<td>27</td>
<td>34</td>
</tr>
</tbody>
</table>

-- Boehm, COCOMO II, 2000
Ex 5: Predicting mobile performance

Given a configuration of applications to support a user task, what will its resource requirements be?

Design $d$ is “configuration” expressed as

$$\{ \text{<application, (QoS settings)>} \}$$

$$\{ \text{<Windows Media Player, (24 fps, 300x200, high quality audio)>}, \text{<MS Word, ()>}, \text{<Firefox, (5 s, text)>} \}$$

Resource use of configuration
Ex 5: Predicting mobile performance

Empirical profiling yields resource usage
Implementation attributes maintain distinctions among resource consumers:

\{<\text{application}, (\text{QoS settings}), \text{resource usage}>\}

\{<\text{Windows Media Player},
\quad (24 \text{ fps}, 300\times200, \text{high quality audio}),
\quad (25\%, 256 \text{ Kpbs}, 30 \text{ MB})>,
\text{ <MS Word},
\quad (2\%, 0 \text{ Kpbs}, 28 \text{ MB}>,
\text{ <Firefox},
\quad (5 \text{ s, text}),
\quad (8\%, 56 \text{ Kpbs}, 10 \text{ MB})>\}

Time ≠ Money

\[ U(d) = B(x) - C(d, x, m) \quad \text{where} \quad x = P(d) \]

Capabilities \( x \) and values \( v \) are multidimensional; they may be measured on different scales

\begin{align*}
\text{Let} \quad & d \quad \text{be a design in some appropriate notation} \\
\text{in} \quad & \text{in}\ A^n \quad \text{open-ended vector of arbitrary attributes} \\
\text{be in} \quad & \text{in}\ V^n \quad \text{open-ended vector of arbitrary attributes} \\
\text{B express} \quad & \text{predicted value} \ v \ \text{of} \ x \ \text{to user} \\
\text{C express} \quad & \text{cost} \ v \ \text{of getting} \ x \ \text{from} \ d \ \text{with method} \ m \\
\text{P predicts} \quad & \text{capabilities} \ x \ \text{that} \ m \ \text{will deliver for} \ d
\end{align*}
Multidimensional Cost Analysis

- Different factors in a problem are appropriately measured in different ways
  - Dollars, computer resources, user distraction, staff time, reputation, schedule, lives lost
- It’s tempting to convert everything to dollars, but this can lead to …
  - Loss of information related to different properties
  - Errors by converting nominal, ordinal, or interval scales to a ratio scale
  - Loss of flexibility by early choice of conversion
  - Confusion of precision with accuracy
- Many analysis techniques require a single cost unit, but you should delay conversion as long as possible

Properties of Resources

- **Perishable**: lost if not used
  - Perishable: bandwidth
  - Nonperishable: disk space
- **Fungible**: convertible to other resources
  - Complete: common currency
  - Partial: bandwidth vs CPU (compression)
  - None: calendar time vs staff months
- **Rival**: use by one person precludes use by another
  - Rival: money, labor, bandwidth
  - Nonrival: information goods
- **Measurement scale**: appropriate scale & operations
  - Nominal, ordinal, interval, ratio
Ex 6: Algorithmic Complexity

Analysis of algorithms tells you how running time will scale with problem size
- A sort algorithm might be $O(n \log n)$
- Scalability is not a scalar attribute!!

In this case
- $d$, the design, is the pseudo-code of the sort algorithm
- $x$, the capabilities, is $O(n \log n)$ scalability
- $v$, the value space, includes a scalability dimension
- $m$, the development method, is a programming technique
- $P$ predicts competent implementation $\Rightarrow$ expected runtime
- $C$ is the cost (e.g., performance) of $O(n \log n)$ execution time

Considering development method

We don’t have the code during early design, so we have to predict the implementation properties $x$ assuming $d$ is implemented by method $m$

Let $d$ be a design in some appropriate notation
$x$ be in $R^n$, an open-ended vector of capabilities
$v$ be in $V^n$, a multidimensional value space
$m$ be in some notation, a development method

$U(d) = B(x) - C(x)$ where $x = P(d,m)$

$B$ express benefits predicted value $v$ of $x$ to user
$C$ express costs cost $v$ of getting $x$ from $d$ with method $m$

$P$ predicts capability capabilities $x$ that $m$ will deliver for
Ex 6a: Algorithmic Complexity, again

- Analysis of algorithms tells you how running time will scale with problem size
  - A sort algorithm might be $O(n \log n)$
- In this case
  - $d$, the design, is the pseudo-code of the sort algorithm
  - $x$, the capabilities, is $O(n \log n)$ scalability
  - $v$, the value space, includes a scalability dimension
  - $m$, the development method, is a programming technique
- $P$ predicts competent implementation $\Rightarrow$ expected runtime
- $C$ is the cost (e.g., performance) of $O(n \log n)$ execution time
- Implementation must be competent, not just correct
  - I once saw an $O(n^3)$ implementation in a class assignment!

Ex 7: COCOMO II Early Design Model

- COCOMO predicts effort (PM) & schedule (TDEV)
  - $PM = A \times (Size)^E \prod \Sigma EM_i$ where $E = B + 0.01 \Sigma SF_j$
  - $A, B$ are calibrated to 161 projects in the database
  - $EM_i$ and $SF_j$ characterize project and developers
  - $TDEV$ is similar
- But it depends on Size, and LOC aren’t known early
  - Count unadjusted function points (UFP) in requirements
  - Use COCOMO II’s conversion table (previous example!!)
  - $Size = KSLOC$ (programming language, UFP)
Ex 7: Predicting development effort

\[ C(d, x, m) = C(\text{Size }, x, <A, B, EM_j, SF_k>) = <PM> \]
\[ = < A \times \text{Size}^E \Pi_i EM_i, > \text{ where } E = B + 0.01\sum_j SF_j \]
\[ = < A \times \text{KSLOC}(pl, UFP(d))^E \Pi_i EM_i > \]

With nominal values for A, B, SF\_j, EM\_j
\[ = < 2.94 \times \text{KSLOC}(pl, UFP(d))^{1.0997} > \]

For 100KSLOC system,
\[ = < 465.3153 \text{ person-months} > \]

Client-focused Value

\[ U(d, \theta) = B(x, \theta) - C(d, x, m) \]

Most significantly, value can only be reckoned relative to the needs and preferences (utilities) of a stakeholder – in this case, the client or user

Let \( d \) be a design in some appropriate notation
\( x \) be in \( R^n \) an open-ended vector of capabilities
\( v \) be in \( V^n \) a multidimensional value space
\( m \) be in some notation a development method
\( \theta \) express user pref a multidimensional utility space
\( B \) express benefits predicted value \( v \) of \( x \) to user with pref \( \theta \)
\( C \) express costs cost \( v \) of getting \( x \) from \( d \) with method \( m \)

\( P \) predicts capability capabilities \( x \) that \( m \) will deliver for
Ex 8: Mobile configuration utility

\[ U(d, \theta) = B(x,\theta) - C(d,x,m) \]

where \( x = P(d,m) \)

We previously saw prediction of \( x \) from \( d \)

\( x \) is qualities of delivered service (e.g. video fidelity)

\( d \) is application configuration (player + editor)

\( v \) is <user utility, seq of configurations, resource use>

Objective is a sequence of configurations \( d \) with the
that best satisfies each user’s personal preferences \( \theta \)

<table>
<thead>
<tr>
<th>Video player</th>
<th>Windows media</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RealPlayer</td>
<td>0.8</td>
</tr>
<tr>
<td>Frame rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 fps</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>18 fps</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>24 fps</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

... etc ...

Ex 8: Mobile configuration utility

- **Utility**: Benefit of configuration
- **\( \theta \)**: user preferences
- **\( x \)**: quality of configuration
- **\( d \)**: capability point

Institute for Software Research, International

Mary Shaw 10/5/2005
**Ex 8: Mobile configuration utility**

For the configuration design point

\{ <Windows Media Player,
(24 fps, 300x200, high quality audio),
(25%, 256 Kpbs, 30 MB)>,

... etc ...
\}

The utility is weighted by attribute

<player, frame rate, frame size, audio> \sim <.5, 1.0, .5, 1.0>

Then the player component of the utility is

\[ 0.5 \times \theta(\text{Media Player}) + 1.0 \times \theta(24 \text{ fps}) + 0.5 \times \theta(300\times200) + 1.0 \times \theta(\text{high}) \]

\[ = 0.5 + 1.0 + 0.5 + 1.0 \]

\[ = 3.0 \]

---

**Uncertainty in values**

\[ U(d, \theta) = B(x,\theta) - C(d,x,m) \]

where \( x = P(d,m) \)

Capabilities \( x \) and values of \( B, C \) may be contingent and uncertain, so the value space may express uncertainty such as ranges, probabilities, future values

Let \( d \) be a design in some appropriate notation

\( x \) be in \( R^n \) an open-ended vector of capabilities

\( v \) be in \( V^n \) a multidimensional value space

\( m \) be in some notation a development method

\( \theta \) express user pref a multidimensional utility space

\( B \) express benefits predicted value \( v \) of \( x \) to user with pref \( \theta \)

\( C \) express costs cost \( v \) of getting \( x \) from \( d \) with method \( m \)

\( P \) predicts capability capabilities \( x \) that \( m \) will deliver for
Ex 9: Present Value Analysis

Purchase or license a component?
- Benefit $60K/year, realized at end of year
- License cost $50K/year, due at beginning of year
- Purchase cost $120K, at beginning
- Interest rate 5%/year

<table>
<thead>
<tr>
<th>End yr</th>
<th>Purchase</th>
<th>License</th>
<th>Benefit</th>
<th>1/(1+I)^N</th>
<th>Purchase</th>
<th>License</th>
<th>Benefit</th>
<th>Val</th>
<th>pur</th>
<th>lic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>120</td>
<td>50</td>
<td>60</td>
<td>1.00</td>
<td>120.00</td>
<td>50.00</td>
<td>-</td>
<td>120.00</td>
<td>50.00</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>60</td>
<td>0.95</td>
<td>47.62</td>
<td>57.14</td>
<td>120.00</td>
<td>97.62</td>
<td>57.14</td>
<td>120.00</td>
<td>97.62</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>60</td>
<td>0.91</td>
<td>45.35</td>
<td>54.42</td>
<td>120.00</td>
<td>111.56</td>
<td>84.44</td>
<td>13.95</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>60</td>
<td>0.86</td>
<td>43.19</td>
<td>51.83</td>
<td>120.00</td>
<td>142.97</td>
<td>111.56</td>
<td>13.95</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>92</td>
<td>0.82</td>
<td>49.36</td>
<td>120.00</td>
<td>186.16</td>
<td>212.76</td>
<td>92.76</td>
<td>20.92</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>120</td>
<td>200</td>
<td>240</td>
<td>120.00</td>
<td>186.16</td>
<td>212.76</td>
<td>186.16</td>
<td>212.76</td>
<td>92.76</td>
<td>20.92</td>
</tr>
</tbody>
</table>

Economic Value in a SW Project

Note the times at which variables are evaluated
- Development cost (I) is PV at time 0 of development cost
- Asset value (C) and Operation cost (M) are PV at time T
- Risk (d) is used as discount rate to move C&M to 0
- Flexibility value (Ω) measures value of strategic flexibility

\[ \text{NPV} = \frac{(C-M)}{(1+d)^T} - I + \Omega \]

—Erdogmus, Comparative evaluation of development strategies with NPV, EDSER-I, 1999
Plan

Role of early design evaluation
Model for predictive analysis of design

Techniques for predicting value from design

Framework for composing and comparing the techniques

Scenarios for use
Open problems

Usage scenarios

- Evaluating a given design, comparing products
  - Most of the previous examples explore this scenario
- Composing evaluation functions
  - COCOMO Early Design composes code size estimate with the effort and schedule estimators
- Optimizing among design alternatives
  - We show dynamic reconfiguration for mobile devices
- Deciding what design information to write down
  - Look at the design representations used the the predictors that may be appropriate
- Exploring tradeoff spaces
  - We now show how to use COCOMO in this way
Recall: COCOMO II Early Design Model

- COCOMO predicts effort (PM) & schedule (TDEV)
  \[ PM = A \cdot (\text{Size})^E \Pi EM_i \text{ where } E = B + 0.01 \Sigma SF_j \]
  - \( A, B \) are calibrated to 161 projects in the database
  - \( EM_i \) and \( SF_j \) characterize project and developers
  - TDEV is similar

- But it depends on Size, and LOC aren’t known early
  - Count unadjusted function points (UFP) in requirements
  - Use COCOMO II’s conversion table (previous example!!)

Size = KSLOC(programming language, UFP)

Ex 10: Tradeoffs in development costs

- Most of \( EM_i \) and \( SF_j \) describe development method, but four describe characteristics of the product
  - SCHED (required development schedule constraint)
  - RCPX (required reliability and complexity)
  - RUSE (required reusability)
  - PDIF (platform difficulty)

- We can restate the Early Design estimators to retain these as parameters
  - For simplicity, use only RCPX, SCHED
COCOMO II, Product Factors Isolated

\[ U(d) = \text{\textsc{\#}} \ C(d,x,m) \]  
\[ \text{where } x = P(d,m) \]

\[ \begin{align*}  
\text{\#} x &= <\text{RCPX, SCHED}>, x_i \text{ in } \{\text{XL, VL, L, N, H, VH, XH}\} \\
\text{\#} d \text{ is Size} &= \text{KSLOC} (\text{prog lang, UFP(rqts)}) \\
\text{\#} v \text{ is value space} &= <\text{PM, TDEV, RCPX, SCHED}> \\
\text{\#} m \text{ is encoded in the adaptive factors} &= <A, B, \Sigma_{i \neq \text{RCPX, SCHED}} SF_i> \\
\text{\# COCOMO (P) then predicts the cost element of } v &= \text{PM} = A (\text{Size})^E \prod_{i \neq \text{RCPX, SCHED}} EM_i \times EM_{\text{RCPX}} \times EM_{\text{SCHED}} \\
\text{where } E &= B + 0.01 \Sigma_{i} SF_i 
\end{align*} \]

\[ C(d,x,m) 
= C(d, <\text{RCPX, SCHED}>, <A, B, \Sigma_{i \neq \text{RCPX, SCHED}} SF_i>) \\
= <\text{PM, TDEV, RCPX, SCHED}> \\
= <A \times \text{Size}^E \prod_{i \neq \text{RCPX, SCHED}} EM_i \times EM_{\text{RCPX}} \times EM_{\text{SCHED}}, TDEV, RCPX, SCHED> \\
\text{where } E &= B + 0.01 \Sigma_{i} SF_i \\
= <A \times \text{KSLOC} (\text{pl, UFP}(d))^{1.0997} \times EM_{\text{RCPX}} \times EM_{\text{SCHED}}, TDEV, RCPX, SCHED> \\
\text{With nominal values for } A, B, SF_i \text{ all } EM_i \text{ but RCPX, SCHED} &= <2.94 \times \text{KSLOC} (\text{pl, UFP}(d))^{1.0997} \times EM_{\text{RCPX}} \times EM_{\text{SCHED}}, TDEV, RCPX, SCHED> \\
\text{For 100KSLOC system,} &= <465.3153 \times EM_{\text{RCPX}} \times EM_{\text{SCHED}}, TDEV, RCPX, SCHED> 
\]
**Ex 11: Utility–based Adaptive Configuration**

- Ubiquitous computing systems are resource-limited
  - Processor power, bandwidth, battery life, storage capacity, media fidelity, user distraction, …
- Users require different capabilities at different times
  - Editing, email, viewing movies, mapping, …
  - Dynamic preferences for quantity and quality of service
- Abstract capabilities can be provided by different combinations of services
  - Specific editors, browsers, mailers, players, …
- Use utility theory and linear/integer programming to find best sequence of configuration
- Vahe Poladian (5th year PhD student)
  - Papers in EDSER4, ICSE’04
Two Tasks Using Two Resources

- 100 units of memory line: capacity limit based on memory
- 120 units of battery line: capacity limit based on battery

feasible region

User Utility for Task Combinations

# map lookups vs. # emails processed
Predicting Value from Design

Plan

Role of early design evaluation
Model for predictive analysis of design

Techniques for predicting value from design

Framework for composing and comparing the techniques

Scenarios for use

Open problems

Review: Examples

Toy examples
1. Value as simple benefit minus cost
2. Selection of representation for a task
9. Present value analysis for buy vs license decision

Real models
3. Feature value from econometric analysis of spreadsheets
6. Performance prediction based on algorithmic complexity
7. Schedule and effort from COCOMO II
4. KSLOC prediction from requirements via function points
10. RCPX & SCHED tradeoffs from COCOMO II

Current and recent research
Multidimensional costs
5, 8, 11. User-oriented configuration of mobile devices
Other examples

- Security Attribute Evaluation Method (SAEM, Butler)
  - Elicit client’s threat, asset protection priorities ($\theta$)
  - Evaluate per-threat countermeasure effectiveness
    ($x = P(d,m)$) of candidate security technology to add ($d$)
  - Weight countermeasures by priorities ($B(x,\theta)$)
- Cognitive modeling for UIs (Keystroke, GOMS)
  - Design UI and select common tasks
  - Use cognitive model to predict task times ($x = P(d,m)$)
- Real options to evaluate delayed decision
  - Additional cost now to preserve flexibility
  - Cost to exercise flexibility later
    - $C(d,x,m)$ expresses implementation and design cost now
    - $B(x,\theta)$ expresses option value for exercising flexibility later

FAQ

- Is it sound? No, it’s light!
- Is the model correct? Maybe not, it’s a first cut
- Is it complete? No, it’s opportunistic
- Is it universal? No, it takes user view of value
- Does it work? Maybe. We’ll see
- So, is it useful? We already think so
- What does it not do? Things that need code
We need better ways to analyze a software design and predict the value its implementation will offer to a customer or to its producer.

Many techniques provide early, but selective, evaluation.

They are not organized to use systematically.

Economic view offers promise for unification.