Evaluation in the Face of Uncertainty: Anticipating Surprise and Responding to the Inevitable

AEA/CDC Summer Evaluation Institute
June 13 – 16, 2010
Atlanta GA

Jonathan A. Morell, Ph.D.
jamorell@jamorell.com
(734) 646-8622

© 2010 Guilford Publications
What is this workshop about?

Response to surprise
- Crisis response → advance planning

Disseminating knowledge
- Tactics for adding surprise to the evaluation mix

Community building
- More and better tactics
- More and better theory
- Archive of cases

Adding “surprise” to evaluation planning
- Funding
- Deadlines
- Logic models
- Measurement
- Program theory
- Research design
- Information use plans
- Defining role of evaluator
- Logistics of implementation
- **Planning to anticipate and respond to surprise**

In this workshop we will go heavy on tricks and tips, light on theory, explanation, or analysis of collected cases.
The goal is informed commitment to practical action

- When is the likelihood of surprise high?
- When will surprise disrupt evaluation?
- If probability of disruption is high, what can we do about it?

- No formula but theory and experience help
- No magic bullet but we can chip away at the problem

- Many choices, one actual design
- All have pros and cons
- Tradeoffs are inescapable
Some historical background

We know why unexpected events occur

Evaluation
- Goal free evaluation emphasizes what a program does, not what it claims
- Interactivity between evaluation and the program being evaluated

Explanations embedded in domain
- Marketing, education, drinking regulation, tobacco control, product development, welfare, and many others, I have no doubt.

Complex systems
- Uncertain environments, cross linkages, self organization, adaptation, feedback loops with different latencies, etc.

But what to do about it as evaluators?

Guaranteed solution
- Post-test only
- Experimental group only
- Unstructured data collection

But we want to do a lot better
You can never tell the future but some surprises are more foreseeable than others

- Get lucky
- Knowledge from stakeholders
- Good program theory
- Use research literature
- Use experts

**Theory**

- Limiting time frames
- Exploiting past experience

**Forecasting & program monitoring**

- System based logic **modeling**

**Retooling program theory**

- Agile methodology
- Data choices

---

Complex system behavior makes prediction impossible no matter how clever we are. PS – do not assume that complex systems are always unpredictable!
We don’t know exactly where the cats are but we can sweep them toward one side of the landscape, and tame the one’s that escape.
Programs and their evaluations have an essential similarity

- What will help us with unexpected program outcomes will also
- Help us with unexpected problems in conducting an evaluation because
- Both are similar social constructions
  - Resources (time, people, $)
  - Processes
  - Embedded in a social setting
  - To accomplish specific objectives
What are the practical and political reasons for surprise?

Any single organization has limited money, political capital, human capital, authority and power.
Narrow windows of opportunity.
Competition requires bold claims.
Resource owners have parochial interests.
Design expertise limited.
Collaboration across agency boundaries is very difficult.

Short term success is rewarded.
Partial solutions can accrue to major success over time.
Pursuing limited success with limited resources is justifiable.

Result:
- Narrow programs
- Simple program theories
- Small set of outcomes

Planners may know better but they are doing the best job they can. Evaluators have to follow.
What might an unforeseen but predictable outcome look like?

**Program**
- Post-natal care in Niger
- Formal fees
- Informal fees integrated into (hidden in) overall fee structure

**Innovation**
- NGO provides drugs and supplies
- Remove fees

**Results**
- Patients: drug hoarding (patients learned from previous programs)
- Staff: game system, new fees

- Experience with similar programs
- Psychology of self interest
- Common sense

*Something* like this will happen, even if we can’t say exactly what.
What might unforeseeable outcomes look like?

The problem is not sensitive to scale. We run into the same trouble with large and small problems.

If you built a logic model here

- Tutoring services
- Textbook publishers
- Reform advocates
- Dept. of Ed.
- Legislators
- Charter schools

Would it be valid here?

- Tutoring services
- Textbook publishers
- Reform advocates
- Legislators
- Charter schools
- Dept. of Ed.
- Legislators
- Tutoring services

School Reform

Niger

Health measures?
Agenda of other NGOs
Similar programs?
Replace lost income?
Family structure?
Government policy?

Health measures?
Agenda of other NGOs
Similar programs?
Replace lost income?
Family structure?
Government policy?

Fluctuating Environments

© 2010 Jonathan Morell
How much surprise should we expect?

- Where is the program in its life cycle? Start-up phase is unstable.
- How stable is the environment? The past not be a good guide, but maybe better than nothing.
- How robust has the innovation been over time and context?
- How rich and tight are the linkages?
- What is the “size” of the program relative to the boundaries of the system it is in?

<table>
<thead>
<tr>
<th>Linkages</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Whole school reform</td>
</tr>
<tr>
<td></td>
<td>Continuity of care</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>New reading curriculum</td>
</tr>
<tr>
<td></td>
<td>Pre-surgical checklist</td>
</tr>
</tbody>
</table>

Why is this advice problematic

- What does “big” mean with respect to a system and an innovation?
- What pattern of linkages qualify as “rich”?
- What feedback latency constitutes “tight”?
- What does “fast” mean with respect to a life cycle?
- Rich linkages might indicate both stability and fragility
- Small changes can have disproportionally large effects

But it still helps to ask the questions
How do we estimate the likelihood of surprise?

- **Fidelity and robustness**
  - Fidelity = extent program adheres to proven protocols
  - Robustness = program works when fidelity is low and context variation is high
  - Low fidelity + low robustness = high likelihood of surprise

- **Time erodes predictability**
  - Shifting environments
  - Longer feedback loops
  - Changing internal operations
  - New customer and stakeholder needs

- **R&D content**
  - Proven knowledge in novel setting, e.g. cross functional continuous process improvement in a poisonous labor/management climate
  - Novel program, e.g. injecting “consumer operator services” into a traditional mental health setting
  - Novel phenomenon, e.g. integrating Web 2.0 into routine organizational operations

But when uncertainty is high AND uncertainty is problematic for evaluation?

- Life cycle view
- Social/organizational view
Program x evaluation life cycles can help us understand when uncertainty is high and problematic

Evaluation life cycle
- Shorter or longer than program life cycle
- Begins sometime after program start (usually)
- Stages affect each other iteratively
- More spiral than waterfall form, but with some lag, all stages are present

Program life cycle
- Formal events, e.g. budgets, yearly plans, publishing RFPs
- Continual stream of micro-level changes and environmental adaptations with greater effects early on

Relationships between the life cycles affect unpleasant surprise
Multiple, short term studies
- Continuous process improvement
- Short time between cause and effect = inference with simpler methodology
- Pretesting and prototyping to test evaluation design
- Inherently sensitive to unexpected program activity

1:1 Correspondence between life cycles
- Fog of start up
- Surprise late in program life cycle can force early stage evaluation redesign
  - *Gets worse when design and data requirements must be stable over time*

Retrospective focus
- Emphasis on program in stable part of life cycle
- Program change, evolution relatively unimportant
<table>
<thead>
<tr>
<th>Stage where corrective action most useful</th>
<th>Design</th>
<th>Implementation</th>
<th>Data Collection</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Case 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Collection</td>
<td></td>
<td></td>
<td></td>
<td>Case 1</td>
</tr>
<tr>
<td>Data Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Case 2: Computer training
- Early discovery of disagreement over multiple stakeholders’ priorities
- Design reworked many times prior to evaluation implementation
- Design was able to satisfy all needs

### Case 1: Child Care
- Sponsor’s priority: ratio of caregivers to children
- Minimum ratio set by regulation
- Upper limit set by economics
- Restricted range → no significant findings
- Design problem discovered at analysis stage
- Evaluation question morphed: Impact of number of children per group.

We can also learn a lot by comparing evaluation stage where a problem is discovered to the stage where it is best fixed.
Where does surprise fall on the program x evaluation life cycles? 32 surprises from 18 cases

This is a CI-like approach. I wish this kind of evaluation were done.*

The common work we all know and love

- Not enough of this kind of work done?
- I could not find it?
- Less susceptible to surprise?

Where does surprise come from? A Social/organizational view is also helpful in understanding surprise

Program related

Environment
- Funder / regulatory decisions
- Program client / customer behavior

Internal activity
- Organizational behavior
- Program staff

Evaluation implementation
- access to interviewees
- IT system capabilities
- access to comparison groups
- innovation creep across groups
- ...
Where does surprise come from and how does it move through the system? 32 surprises from 18 cases

Program related

Environment
- Funder / regulatory decisions
- Program’s client / customer behavior

Internal activity
- Organizational behavior
- Program staff

Evaluation implementation
- Program – evaluation coordination
- access to interviewees
- IT system capabilities
- access to comparison groups
- innovation creep across groups
- …

Adjustments to evaluation

Design
e.g.:
- substitute time series for cross-group measures
- add interviews to explicate program theory
- New evaluation objectives

Procedure / logistics
e.g.:
- Change method of recruiting respondents
- Hire professional interviewers

Analysis / interpretation
e.g.:
- construct proxy variables
- redefine study as combined impact of control and experimental groups

© 2010 Guilford Publications
Some other useful ways of categorizing sources of surprise

- **Pilot tests / feasibility assessments:** Important but not infallible, e.g.
  - Last year’s data used to estimate power do not apply to current year
  - Query individuals who can answer for themselves but not for the organizational behavior

- **Resistance to evaluation**
  - People think they can speak for a program when they can’t
  - Levels are not static over time

- **Incorrect assumptions early in the evaluation life cycle**
  - Funders ask the wrong question
  - People think they can promise data but can’t deliver
These methods are most useful early in evaluation life cycle

**Foreseeable**
- Get lucky
- Knowledge from stakeholders
- Good program theory
- Use research literature
- Use experts

**Unforeseeable**
- Complex system behavior makes prediction impossible no matter how clever we are.
  PS – do not assume that complex systems are always unpredictable!

---

**Theory**
- Limiting time frames
- Exploiting past experience
Theory as a tactic for reducing surprise

- Why is theory useful?
  - Example 1: Program theory*
  - Example 2: Life cycle behavior
  - Example 3: Perfect Market

- Why is theory problematic?

- How can value be maximized and problems minimized?

Explanatory power helps look in the right place.
Too many to choose from
Choose more than one, choose wisely.

Program Theory

- Context specific
- Engages stakeholders
- Good framework for surfacing assumptions
- Captures knowledge of deep program experts
- Assures evaluation that will meet what stakeholders perceive as their needs

- Stakeholders cherished beliefs can be wrong
  - Limited to stakeholders’ perspectives
  - Not likely to capture much relevant knowledge
    - Similar programs in other contexts
    - Research literature
Theory examples: Life cycles

Ex #1 Worker participation safety program

Union member to evaluator: “These things last 5 years. They always do.”

Ex #2 CC Training

Welding

Verify job market → Instruction quality → Student learning → Student Placement

Y 1: High
Y 2: High
Y 3: High

Technology refresh rate: 60 months

Animation

Verify job market → Instruction quality → Student learning → Low

18 months
Recognizing that measurement and public reporting are powerful mechanisms to drive quality and efficiency improvement throughout the health care system, purchasers and consumers have embraced a vision of a transparent health care market, in which decision-making is supported by publicly reported comparative information. Our shared vision is that with this information, Americans will be better able to select hospitals, physicians, and treatments based on nationally standardized measures for clinical quality, consumer experience, equity, and efficiency. http://www.healthcaredisclosure.org/about/
Choosing Theories

Principles
- One is better than none
- A few are better than one
- Include stakeholders’ program theory
- Using more than a few is dysfunctional – too many variables and relationships
- Choices establish path dependency. Make sure all theories in pool are relevant

Thought Experiment
1. Stakeholders establish program theory
2. Recruit group of diverse experts
3. Experts choose 5 other relevant theories
4. Pick 1/5 at random
5. Add to stakeholder program theory
6. Develop evaluation
7. Pick another theory
8. Repeat

Result: Similarity across designs
- Same program
- Same stakeholders
- Same environment
- Same information needs

Result: All designs better than if only 1 used
- Stakeholders provide context specificity
- Other theories provide relevant
  - Variables
  - Relationships
Capitalizing on what we already know

Few programs are so unique that previous experience won’t decrease surprise

- Process knowledge: What happens to programs like mine in similar circumstances?
  - E.g. How do needle exchange and health eating programs fare at election time?
- What do we know about how programs like mine work?
  - E.g. Do threatening public service announcements encourage diabetics to monitor their blood sugar and control what they eat?
  - Literature reviews and interviews work

Example of using process knowledge to understand program behavior

<table>
<thead>
<tr>
<th>Use of pre-surgical checklists</th>
<th>Certainty of outcomes</th>
<th>Political Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Dissemination of evidence based practice data</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Example of using domain knowledge

- Tobacco control: Integrate person focus and environmental focus
- Problem: Not enough known about successful implementation in this context
- Solution: 1) Literature review of successful ecological implementations. 2) Theory based evaluation of application for tobacco control
Choosing knowledge domains: Principles are the same as with theory

- One is better than none
- A few are better than one
- Include stakeholders’ program expertise
- Using more than a few is dysfunctional – too many variables and relationships
- Choices establish path dependency. Make sure all candidates are relevant
We can minimize surprise by limiting temporal and causal distance, but we better be careful. A lot can happen as time marches on.
These methods are most useful for detecting leading indicators

- Get lucky
- Knowledge from stakeholders
- Good program theory
- Use research literature
- Use experts

- Complex system behavior makes prediction impossible no matter how clever we are. PS – do not assume that complex systems are always unpredictable!

Forecasting & program monitoring

System based logic modeling

The trick is to do a little better than the Delphic oracle
Use planning and monitoring techniques to revisit program and evaluation at various slices of their life cycles

- Assumptions underlying program success
  - Which are critical?
  - How robust or brittle?
  - Indicators of failure?
- Future states
  - What is the desired future?
  - What are the likely futures?
- Environmental conditions
  - Funding / Politics / Culture
  - Needs of service population, whether individuals of organizations
- Internal operations
  - Staff makeup, organizational structure/culture

How to get all this information?
- Stakeholders are necessary but not sufficient
- Identify all relevant domains
- Identify most relevant subset
- Query relevant subset frequently
- Rotate thorough the others
- Use case study methods
Example of how a program may change over time

The program: Improve safety by training managers

- Some program assumptions
  - Workers can interpret managers’ behavior
  - Safety $\rightarrow$ productivity
  - Safety + productivity $\rightarrow$ manager behavior
  - *No linkage with other CPI initiatives*
  - *No activity to sabotage program*

- Some evaluation assumptions
  - Need only manager, worker surveys + safety, productivity data
  - No confounds to causal inference

Simple program, program theory, and evaluation design, gets complicated.

Managers demonstrate interest in safety $\rightarrow$ Managers improve safety processes $\rightarrow$ Improved safety $\rightarrow$ Improved productivity $\rightarrow$ Workers perceive effort, act accordingly $\rightarrow$ Improved productivity

Company wide CPI program $\rightarrow$ New Discipline Policy $\rightarrow$ Negative relationship
How can an evaluation be designed to change?

- Get lucky
- Knowledge from stakeholders
- Good program theory
- Use research literature
- Use experts

Complex system behavior makes prediction impossible no matter how clever we are. PS – do not assume that complex systems are always unpredictable!

Data choices

Agile methodology

Retooling program theory
Data

Can the data be modified to meet new needs?
- Validated scales vs. open ended questions
- Custom programming vs. standard lookup
- Structured teacher observations during class vs. casual assessment by visitors

Is gatekeeper approval needed?
- OMB
- Air Force Survey Office
- Corporate VP

Are substitutes available without harming the intent of the evaluation?
- Self report → clinical record
- Direct cost → total cost

Are substitutes practical?
- Collection burden increase
- Development cost to move to new methods
- Switching time relative to deadline for getting data
  - E.g. Clinical records vs. patient report
# Agile methodology: Some definitions

<table>
<thead>
<tr>
<th>Agile</th>
<th>Ability to change quickly.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methodology</td>
<td>Logic in which observations are embedded</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Organizational entity</td>
</tr>
<tr>
<td></td>
<td>• Processes</td>
</tr>
<tr>
<td></td>
<td>• Resources</td>
</tr>
<tr>
<td></td>
<td>• Structures</td>
</tr>
<tr>
<td></td>
<td>Constructed to allow</td>
</tr>
<tr>
<td></td>
<td>• Data acquisition to feed</td>
</tr>
<tr>
<td></td>
<td>• Methodology that allows data interpretation</td>
</tr>
</tbody>
</table>

**How to make an evaluation agile**

- Flexible vs. rigid design elements
- Dependencies
- Boundaries
- Partition
- Retool program theory
Example of agile and brittle evaluation components

Methodology
- Two possible comparison groups
- Time series and cross sectional possibilities
- If any one comparison goes away others remain

Data
- Develop, validate fixed-choice instruments for pre-post training assessments
- Interviews ½ way through training for course improvement

Experimental Control Time Series

Then

Later

Accident rate
High
Low
Then
Training
Later

Data
- Interviews with workers soon after an accident to see why/if manager behavior affects safety
- Safety, accident, derailment statistics from IT systems to test primary outcome
What are the agile and brittle components?

<table>
<thead>
<tr>
<th>Data: Formative</th>
<th>Data: Summative</th>
<th>Design</th>
<th>Implications for Agility</th>
</tr>
</thead>
</table>
| Validated instrument test training quality | 2, beginning, end of training | | ▪ Time, cost: difficult to change instrument  
▪ Timing to training critical |
| Semi-structured questions: if/why managers change | 1 half way through for course improvement. | | ▪ Minimal effort to determine questions.  
▪ Variation around midpoint OK. |
| Validated safety culture scales | 3, start, end, 6 months post | | ▪ Time, $, difficult to change instrument.  
▪ First 2 timed to training. 3rd can move |
| Interviews: why manager behavior affects safety | Keyed to occurrence of accidents. | | ▪ Minimal time to determine questions.  
▪ Synchronize with accidents |
| Safety & accident stats | From company IT system | | ▪ Available any time  
▪ Not linked to training |
| 1- Control groups other parts of company | | | ▪ Difficult to implement.  
Considerable negotiation needed. |
| 2- Time series on accidents | | | ▪ Available from IT systems.  
Fallback if #1 disappears |
Number and Richness of Dependencies Affect Agility

- Which design serves the customer best?
- Which design is riskier?
- Which should be chosen?

### Design 1
- Same program
- Somewhat different evaluation questions
- Different in length of critical path

### Design 2

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Mar 28, '10</th>
<th>Apr 4, '10</th>
<th>Apr 11, '10</th>
<th>Apr 18, '10</th>
<th>Apr 25, '10</th>
<th>May 2, '10</th>
<th>May 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interviews as raw material for survey development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Survey developed from interviews</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Baseline data assessment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Conduct program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Posttest data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Final report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Interviews to provide contextual understanding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Evaluation context report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Determine how to get data from IT system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Baseline data assessment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Conduct program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Posttest data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Final report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2010 Jonathan Morell
Example 1: Evaluations that depend on managing boundaries are not agile

There are many good reasons to choose one or another design. Agility can be one of them.

<table>
<thead>
<tr>
<th>Organizational distance</th>
<th>Data collection burden</th>
<th>Data sensitivity</th>
<th>Agility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different leaders</td>
<td>Interviews</td>
<td>Labor / management interactions</td>
<td>Low. Renegotiating any evaluation condition is difficult.</td>
</tr>
<tr>
<td>Same leader</td>
<td>IT data</td>
<td>Technical capacity</td>
<td></td>
</tr>
</tbody>
</table>
Example 2: Evaluations that depend on managing boundaries are not agile

There are many good reasons to choose one or another design. Agility can be one of them.
One reason for partition in an evaluation design is agility

We always split our projects into phases in the service of practicality, methodology, and promoting knowledge use. E.g. pretests of instruments, pilot studies to estimate power, preliminary findings to test stakeholder needs, feasibility assessments

Agility can be another reason to think about partition

<table>
<thead>
<tr>
<th>Process example</th>
<th>Original</th>
<th>Innovation for Agility</th>
<th>Advantages and Disadvantages</th>
</tr>
</thead>
</table>
| Interview 1/2 way in training. Feedback on instruction | Interview some ¼, ½ and ¾ through | • More opportunity to see if program is working to plan. Chance to change outcome measures  
• Logistics more difficult  
• Opportunity for as much information as possible at ½ point is lost |

| Outcome example | Download IT data at end | Analyze at intervals | • Chance to see detect unexpected outcomes  
• More evaluation resources for analysis  
• Greater burden on company’s IT staff |

© 2010 Jonathan Morell
Selecting tactics: Sometimes more is not better

- Get lucky
- Knowledge from stakeholders
- Good program theory
- Use research literature
- Use experts

Complex system behavior makes prediction impossible no matter how clever we are.
PS – do not assume that complex systems are always unpredictable!

Foreseeable

Unforeseeable

Theory

- Limiting time frames
- Exploiting past experience

Forecasting & program monitoring
System based logic modeling

- Retooling program theory
- Agile methodology
- Data choices

But all together they can get us into a lot of trouble.

Any few of these may make sense.
**Example of how multiple tactics induce new problems: Buffering against promised interviews not materializing**

<table>
<thead>
<tr>
<th>Evaluation Scenario</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>Detailed information</td>
<td>Resistance to work not seen as serving clinical purpose</td>
</tr>
<tr>
<td>6 month follow-up, phone interviews by clinic staff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 month, as above</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrator assures cooperation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eliminate 1 data collection</th>
<th>Might get needed information</th>
<th>Less data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Still no guarantee of cooperation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Eliminate interviews, rely on IT information instead</th>
<th>No clinical cooperation needed</th>
<th>Sparse data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IT data often untrustworthy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A lot of work to vet systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Do both</th>
<th>Redundancy</th>
<th>Increased range of information</th>
<th>Multiple measures</th>
<th>Longer to design and implement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Need more diverse expertise on evaluation team</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hard to maintain integrity of evaluation over time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Nurture good relationships with clinical and IT staff</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Resources diverted, e.g. from analysis</td>
</tr>
</tbody>
</table>
Framework for appreciating trade-offs

- Any change to time, complexity or resources can affect the others
- Fixing technical or procedural problems can affect time, complexity or resources
- Delay exceeds stakeholder need for information, but not fixing the problem also risks not meeting stakeholders’ needs
How can we design maximum protection against surprise before problems set in?

- All tactics for dealing with surprise have social and technical aspects
- Ability to detect and adapt rises with # of people and groups to provide advice
- But as size increases so do problems of management and decision making

Tactics for dealing with surprise

Social

- Groups
  - Interaction with stakeholders
  - Group membership
  - Group behavior
  - Design component connectedness

Technical
• General good practice to engineer as much communication as possible along the evaluation life cycle
  • New or evolving needs
  • Evaluation findings
  • Insight on analysis
  • Redesign logic models
  • Also sets a context where minimal extra effort or complication needed to discuss unintended or unexpected program or evaluation behavior

• Split essential and non-essential members
  • Essential: stakeholders whose continued involvement is needed to
    • maintain the evaluation
    • make use of findings
  • Non-essential: weak claims on the program but advice can be useful.
    • Not on the critical path
    • Relatively low cost
    • Very many possible groups but
      • Some are better than none
      • Membership can rotate over time
Diverse input means larger groups. Larger groups are hard to manage
• Use special techniques to get small group behavior from large groups
• Delphi methods to avoid discord
• Loose groups, e.g. advisory boards meet just frequently enough to know the project and who can provide occasional useful advice
• It’s frequency, not just cost. Phone and Web conferencing lowers cost and increases amount of advice that can be purchased
• Split groups by recognize relative connectedness, e.g. sustainability and impact are related, but different enough to keep advisors separate.

• Evaluation plans differ in the number of critical paths among their components Make this one of the considerations. E.g.
  • 6 month follow-up data to design 12 month follow-up, or
  • Design instruments based on cross sectional analysis of past service recipients at 6 and 12 months
• Richness of dependencies. E.g.
  • Continual iteration: 1) Simulation to determine program performance + 2) empirical data collection or
  • Simulation after data collection