Tree Diagrams

David Hallowell
Six Sigma Advantage, Inc.

Aren't Tree Diagrams Plain and Simple?

With the complexity of many of the tools in the Six Sigma kit it's easy to look at tree diagrams as fairly simple and routine. Experience shows, though, that there are enough pitfalls encountered and benefits missed that it could be worth consolidating a few time-won guidelines and tips.

We'll focus on four kinds of " trees" and hierarchical diagrams that become part of many Six Sigma projects:

- Cause and Effect Diagrams
- Y to x flowdown
- Functional Analysis
- Abstraction (KJ and Affinity) Diagrams

Each of these has a specific thrust and strength -that can be surprisingly challenging to capture when one tries to help a real team to build one or more of them. There are enough similarities in the required data and the building processes for these diagrams that teams can tangle them up a bit,- potentially dulling the result.

Getting the full strength from these simple tools requires special attention on what makes each unique – which takes a little vigilance with the team fac\ititation during their construction.

We'll overview each of these diagrams or trees with a brief profile and a few tips. ALong the way we'll look at some subtle ways that they can become inadvertently blurred with one another - hoping to raise the right flags to prevent that in the next cases that you encounter.

Cause and Effect Diagrams

One of the original, tried and true, basic tools needs little introduction. Beginning with an 'effect' that has been observed and verified, a team repeatedly asks "Why?" does this situation exist? When each answer describes a situation that is at least a contributor, the "Why?" question moves down a level to consider why that happens. When done well this can lead a team down, from the original effect (like "Less than expected gas mileage"), which is not directly actionable, to root causes that they can do something about.

The subtle art in bulding these trees seems to be. at each level, crafting the wording of answers to be factual, at the right level of detail and within the team and project's area of concern. Those things together make the next "Why?" question meaningful and direct it to the next level down, toward root causes.

Too often, an answer that is too vague, judgmental, and/or outside the team's area of influence (like "bad management" or "poor training") gets posted in the tree (Fig 1). Such a node makes it hard to pose the next "Why?"question - stifling progress on the drilldown on that branch. Figure 2 shows a better choice, with a factual answer in report language, in the scope of the project's area of concern. A next "Why?" question can be usefully posed, surfacing further detail, as shown in Figure 4.

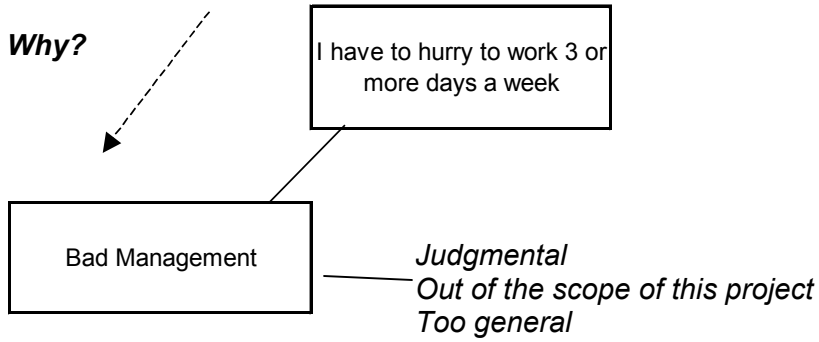

Fig. 1 Common Pitfall: Out of Scope Answer Confuses the Next "Why?" Question

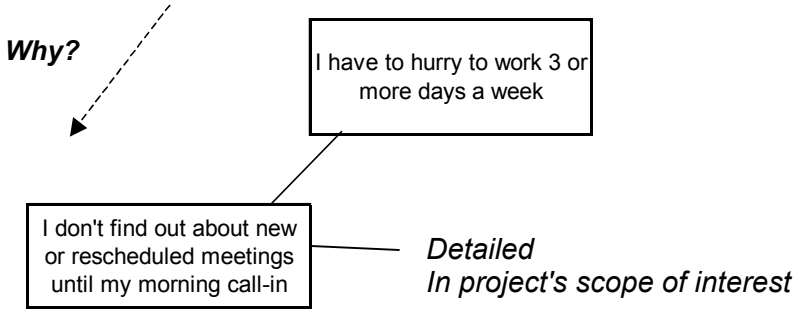

Fig. 2 Better: Detailed, In Scope Answer Enables the Next "Why?" Question

Another common pitfall in cause and effect work is branching to answers that describe 'the parts of' the problem (where it is, when, etc.) (Fig 3). That doesn't answer the original "Why?" question very well and it doesn't set things up very clearly for the next one. Better to revisit and reword such answers to more clearly propel the detail around "Why?"

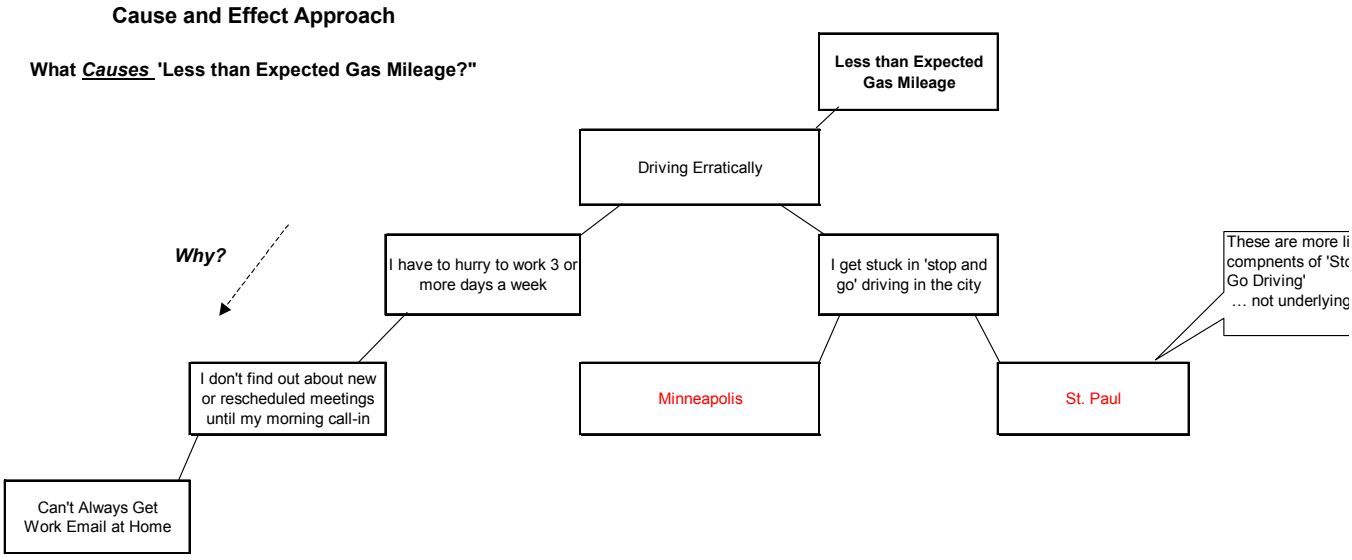

Fig 3: Another Pitfall - Branches that don't answer the "Why?" question

Y to x Flowdown

A Y-to-x begins with an important results measure (the Y) and asks the question, "What factors drive this Y?" While that is not completely different than the cause and effect question, the thrust and content of this tree want to be a distinctly different. Eacn node in the tree should describe a measure - a factor that can take on different values. Factors can describe measures that range from continuous (like time, and capacity) to categoric (like small, medium, large) - but they should all describe measures.

Figure 4 is a part of a simple Y to x tree, cast in the same general area as the cause and effect in FIgure 3. Even though the spirit of the inquiry is similar in each of these cases, by posing the queston about driving factors the Y to x calls for different language in the node labels, and it drives to a different kind of lower level result, with the identified x's. Each node should describe measure - a factor that can take on different values.

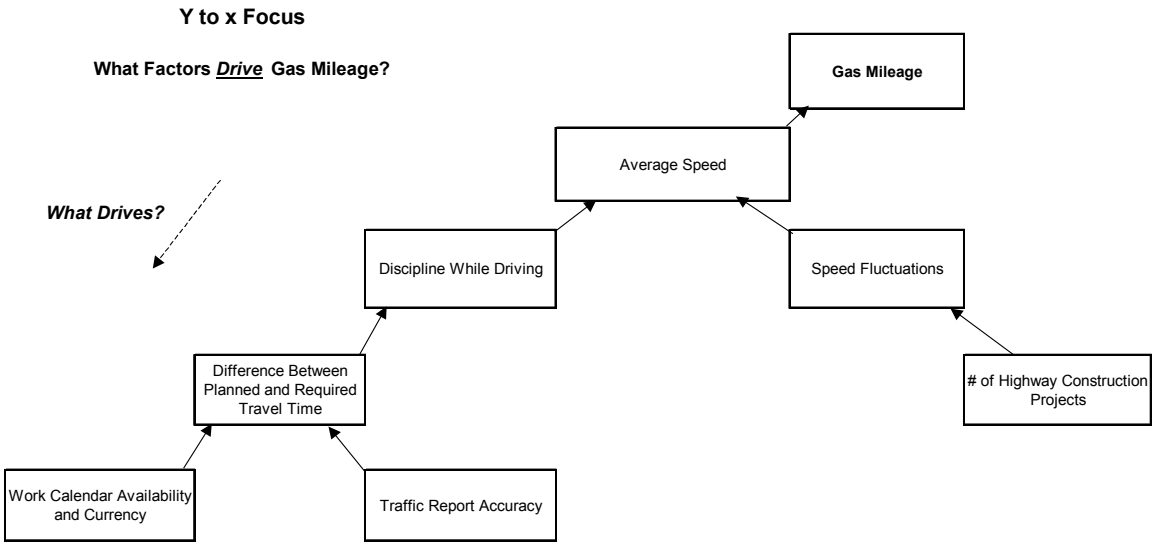

Fig. 4. Part of a Simple Y to x Tree

.

Figure 5 shows part of a Y to x tree for a medical device, illustrating the value in sticking with measures all the way down to the lower level x's - as they can be classified as 'controllable' or 'uncontrollable' or 'noise' factors. While the results Y is generally not a measure that can be influenced directly - the lower level x's should be. In DMAIC projects controllable x's, with verified impact, and in the team's sphere of influence, are are used to drive the the Y in the direction of project gpals. In DFSS projects, the x's and their influences inform design decisions and adjustments to optimize performance during design and implementation

While building Y to x trees some teams are tempted to insert causes, like “Getting a Good Enough Sample to Read” instead of a measure like “Sample Adequacy.” While both phrases describe what’s important, the tree is better served with labels that all read like factors that can take on different values.

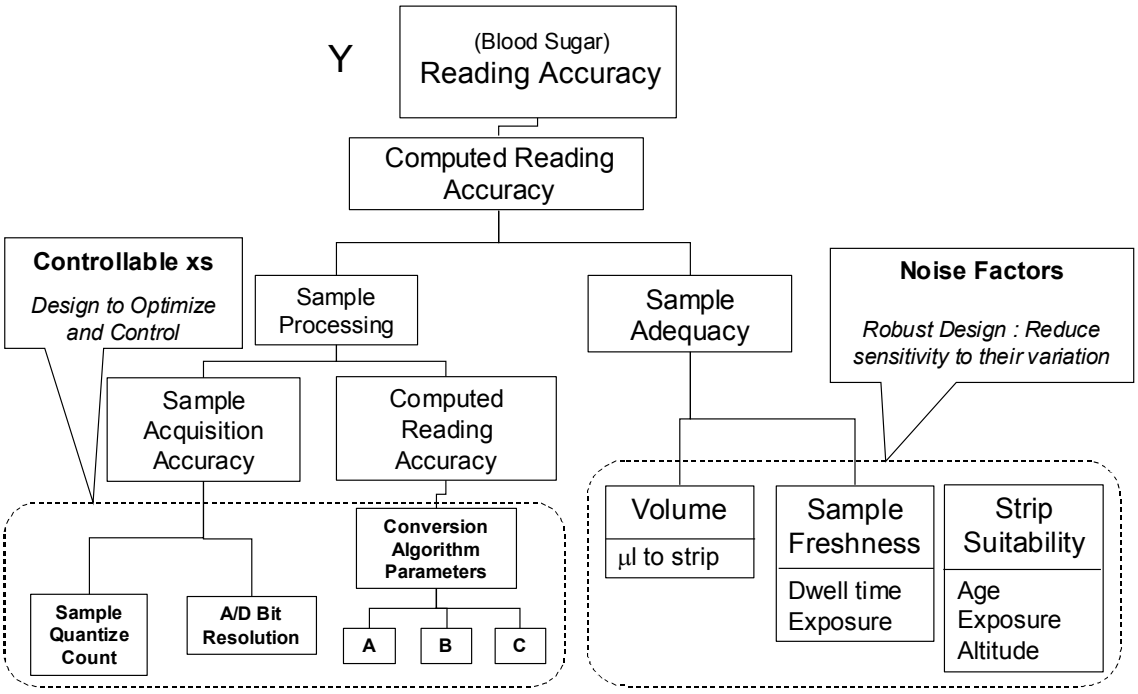

Fig. 5 Y to x flowdown (section) for medical device meter

Functional Analysis

As we've described, a Y to x flowdown focuses on the results **measures** and drivers connected with critical requirements. A related but different perspective is gained by identifying and organizing the important **functionality** in an emerging or existing product or process. This calls for a special 'lens' that highlights the functions delivered. Figure 6 shows such a view, for the same system viewed previously (in Figured 5) through the 'measures' lens.

Functional analysis dates way back as an engineering method, always using verbs to precisely describe functions. More recently, object-oriented thinking has developed 'use cases' which broaden the use (still centered on verbs) to software and business systems. (note - could insert ref to my article 'Use Cases and Measures').

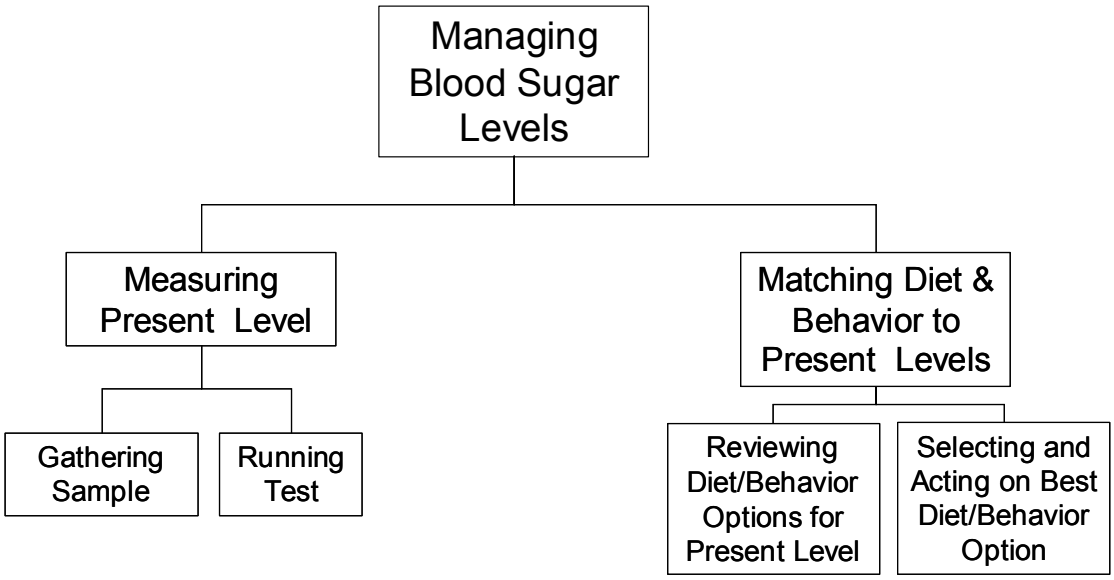

FIg. 6 Preliminary Functional Analysis Tree (section) for a Medical Device

Figure 7 shows how trees can be reviewed and detailed by scanning each level asking "Are these the only (causes, measures, functions…) that convey a complete picture?" Additional latent requirements were discovered, shown as functionality that increased the breadth and/or depth of some branches.

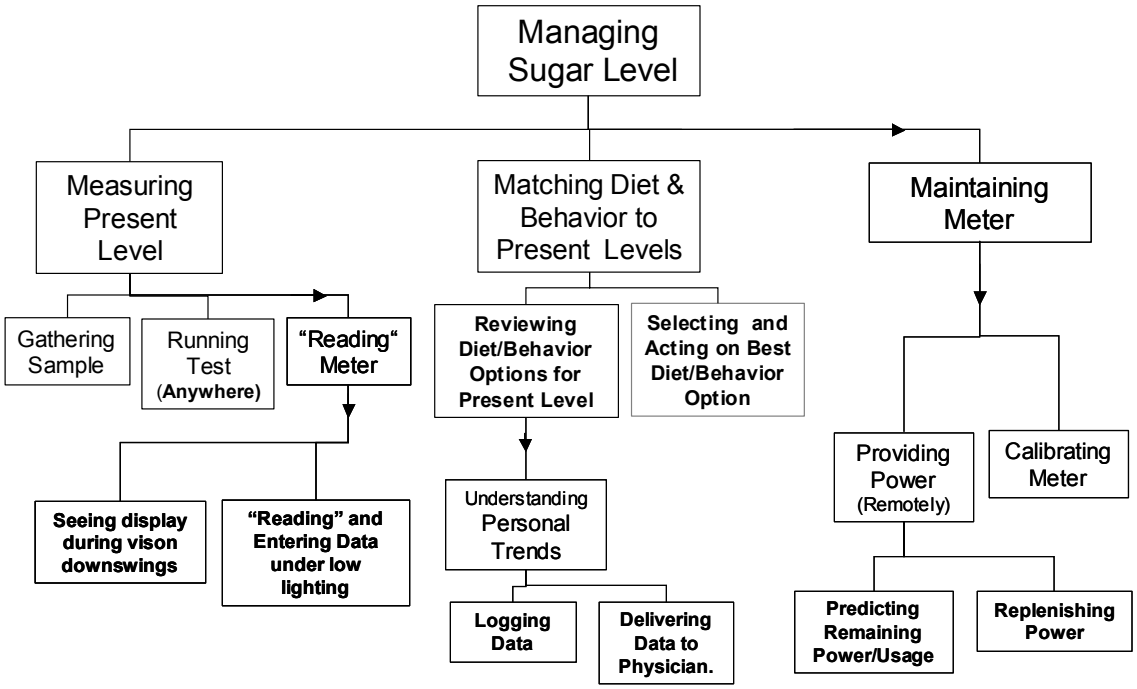

Fig. 7 Improved Functional Analysis Tree - with latent and more detailed functions identified

A functional tree it is much easier to read and review if each node label focuses on a positive, active voice verb (like “measuring, gathering, reading, etc. in Fig. 7). If a team slips into labels that describe measures or where or how the functionality happens it is worth pulling them back to that simple ‘verbs’ discipline.

KJ or Affinitiy Diagrams

A KJ (a language processing tool named for the initials of its originator, Jiro Kawakita) or properly done affinity diagram organizes facts in a tree-like hierarchy. Unique among the tools considered here because they are built from the bottom up, a KJ applies the rules of abstraction discover and articulate key messages at the top of the tree (the blue headline in Figure 8). These few concepts or themes distill the meaning that may not be immediately evident when looking at the many lower level facts (the black data elements in Figure 8).

Teams building a KJ may slip into 'cause and effect' thinking, thinking about why things happen rather than distilling how, when, and where. Every lower level in a KJ tree should be a good example of the data above (at the next lower level of abstraction). (Fig. 9).

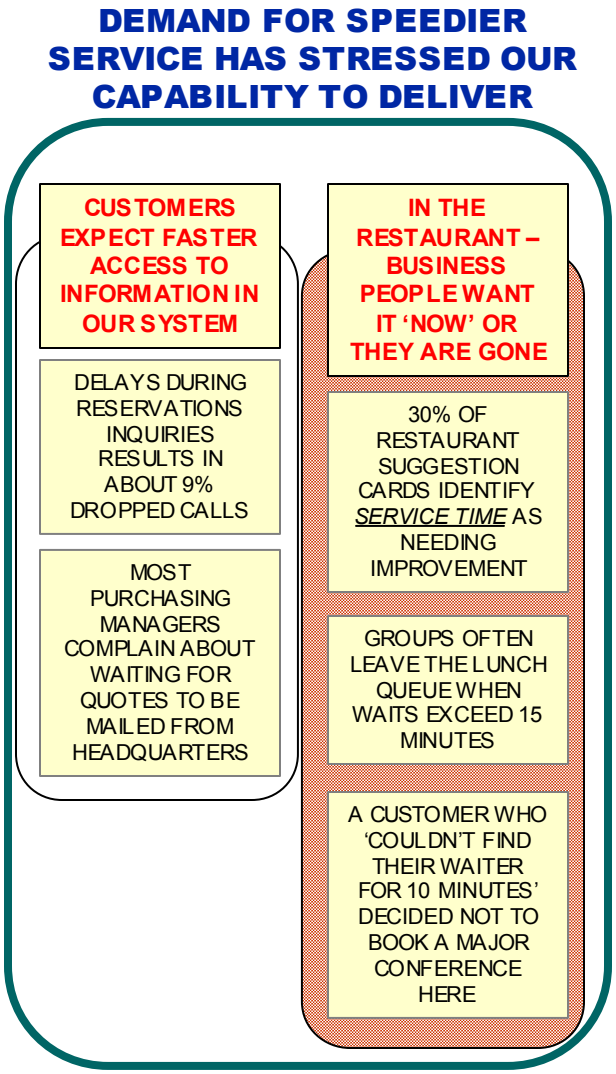

Fig. 8: Section of a KJ Diagram - an Abstraction Tree

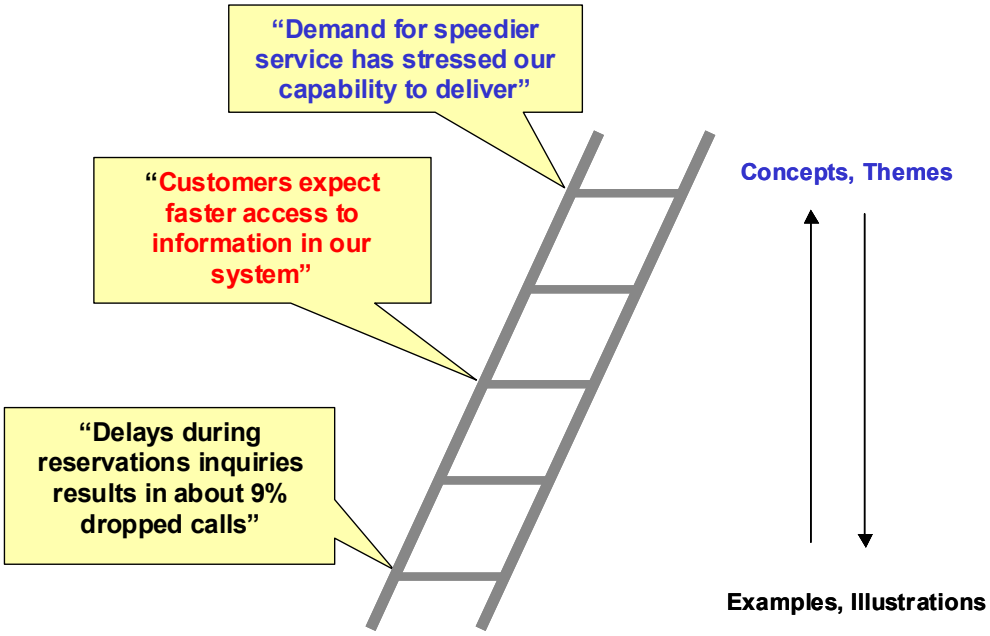

Fig. 9 KJ Follows the Ladder of Abstraction, Not Cause and Effect

Wrapup

We've looked at four types of 'trees' paying special attention to the data and construction logic that makes each one unique. Table 1 outlines the applications and tips for the tables we've revisited. The hope is these tips and insights may help you out a bit the next time you find yourself approaching one of these familiar tools.

	Cause and Effect	Y to x	Functional	Abstraction Tree (KJ and Affinity)
Starting Pont	A documented 'effect'	A results measure (dependent variable)	One or more functions delivered by a product or process	Facts that answer a 'theme question'
Construction	**Top –down**, starting with the effect, asking "Why?" in nested, branching pattern to surface fundamental causes	**Top down**, asking the question "What factors may drive changes in the measure at the current node?"	From **top, bottom, or middle** – organizing a group of connected functions from the general view (at the top) to the detailed 'what's involved?' view toward the bottom.	**Bottom up**, Understanding and grouping factual answers to a theme question using rules of abstraction. Discovering and reporting themes that may not have been evident in the lower level raw data.
Node Wording	Factual situations, described unambiguously	Describes 'factors' = variables that can change value	Focus on the positive, active voice verb that most clearly describes the node's functionality	Factual 'report language' (free of judgment, emotion, or inference).
Tips	Anticipate the next "Why?" question when wording each answer – crafting it to drill down from the previous node while 'opening the way' to a plausible next 'Why?"	Think of all the factors that may drive the variable at the next level up – whether they are presently understood or controlled.	Spend some time coming up with just the right verb.. e.g. is it "identify" or "select" or "report" ? – find the verb with just the right sense.	Use the 'is-a' test for the precise generalization that abstraction should accomplish. Each node in the tree should be a reasonable example or substitute for each node above it.
Pitfalls	Answers that blur or cut off drilldown on a branch of the tree: • Broad and/or judgmental answers – reduce the focus of the next "Why?" • Answers describing 'parts or elements' of the problem (like "Where?" or "When" instead of "Why").	Introducing node labels that are not clearly factors. For example 'data translation time (a clear factor) is better than 'database' or 'data access' for a factor the could drive Response Time'	Mingling measures (like "forecast accuracy') and nouns (like 'incoming data') with the functions.	Using cause and effect thinking to drive relationships in the tree. People often say 'I know why that happens" or "this causes that
Application	Uncover root causes that are actionable – to change the problematic 'effect'	Identify and classify factors (independent variables) that may drive an important results variable	Identify general and specific functionality that operates in concert in a product or process area of interest. The tree structure helps check for completeness and reports the analysis in way that can hide or expose detail as appropriate to different audiences.	Distill fragments of factual data to find messages and themes that are often not evident in raw data by itself. Powerfully and succinctly report the insights derived by the team constructing the KJ type tree.

Table 1: Summarizing the Discussion