

The question could be factual (How many TVs were sold in China in 1998?) or subjective (What's the best place to go dancing in Wichita?).

Anyone can bid to answer. The person will provide information on why he or she is qualified. If both parties agree, the answer is sent to the questioner, who pays the person who answered. If you're not satisfied with the answer, you can refuse to pay.

The site will build up eBay-style mechanisms for self-policing. Answers get a grade from 1 to 4, with comments. Questioners get rated by those who sell them answers. As ratings build, buyers and sellers can be more sure of what they're getting, InfoRocket says.

Top-rated answers will get posted in an archive and the author will get payments every time someone downloads that answer. Also, if you see a question you think a friend could answer, you can pass it on via e-mail with one click.

InfoRocket is aiming first at the mass market, not business. The company is counting on people becoming addicted to the site as they browse for questions they can answer and buy bits of information.

There are hurdles ahead, though. "Copyright and liability are our two biggest issues," says Matt Cassin, an InfoRocket founder. If someone privately sells an answer that contains research done at work, that could violate copyright rules. If someone posts an answer that, for instance, leads to an injury, that could trigger a lawsuit. Cassin says InfoRocket has mechanisms in place to limit those problems.

Your thoughts about this would be appreciated !!

+++++

KnowSite

As described in KnowBits: www.inforocket.com

+++++

10 Steps to Map Individual Knowledge

1. Decide what aspect of your knowledge you want to map.

Consider that you have the accumulated knowledge of your age. What is it that you specifically want to map? Are you solving a particular problem with the map? Are you interested in creating a picture of some particular aspect of what you know?

2. Choose an appropriate representation.

It is a good idea to have some idea of how you will represent your knowledge in the map. My personal preference is a graph - a visual representation that depicts concepts as nodes of the graph and relationships as edges. In this type of representation it is also a good idea to start out with a small set of relationships.

3. Make a list of initial concepts.

I like to think of a knowledge map as beginning from a root or central seed concept. For example, if I were mapping my knowledge about programming, I would make the central node of my graph a node named programming. If I were mapping my knowledge in an outline, then I might start the outline with the concept programming.

4. Connect initial concepts.

Using the root or initial concept as a starting point, connect the other initial concepts to the root concept. For example, using programming as the core concept, I might divide my knowledge map into concepts like C, Java, Visual Basic, Developing Algorithms, Coding, Data Structures, Testing, and Debugging. Note that this list is not meant to reflect a comprehensive list of programming related concepts, but rather a list that reflects my own view of my knowledge about programming. In outline form, I represent this as:

- Root: Programming
 - A. Languages
 - B. Developing Algorithms
 - C. Coding
 - D. Data Structures
 - E. Testing
 - F. Debugging

5. Iterate through your list

You can take the list to any level of detail. For example, I have knowledge of several different programming languages. I might wish to add this as another level of detail in my knowledge map.

- Root: Programming
 - A. Languages

1. C
2. Java
3. Visual Basic
4. C++
5. LISP
6. Prolog
7. Little Languages
 - a. Designing
 - b. Creating
- B. Developing Algorithms
- C. Coding
- D. Data Structures
- E. Testing
- F. Debugging

6. Identify the knowledge you are going to represent

So far, what we have done is to represent a structure for our knowledge. Having settled on a series of topics, what is the actual knowledge you will represent? Back to the original definition about knowledge - that which enables us to do things - we might consider recording rules of thumb (heuristics), processes, procedures, or techniques. Each of these is an attempt to represent something that enables us to do something. Under debugging, I have added three knowledge items.

- Root: Programming
- A. Languages
 1. C
 2. Java
 3. Visual Basic
 4. C++
 5. LISP
 6. Prolog
 7. Little Languages
 - a. Designing
 - b. Creating
 - B. Developing Algorithms
 - C. Coding
 - D. Data Structures
 - E. Testing
 - F. Debugging
 1. Ways to debug
 - a. Printing variables
 - b. Wolf trap

7. Express the Knowledge

How are you going to express the knowledge you want to convey? Typically we will write or speak to express our knowledge. For example, if we want to express a rule of thumb, then we might write something like,

If a program freezes a computer while it is running, this is usually the sign of a memory leak.

This is one way to express a rule of thumb or heuristic.

8. Situate the knowledge that you are expressing.

When you express the knowledge in step seven, it is removed from its original environment. For example,

If you use a pokeball you could catch a pokeman or it might escape.

In this example, unless you are a 7-something year old or a parent of a 7-something year old, you probably don't have any idea what this rule is about. A context for the rule would help identify when it was to be applied. In this case, the context is the Pokemann fad that kids are presently engaged in. By situate, I mean that it is important to set a context for the knowledge you are mapping.

9. Test the knowledge (optional but desirable)

There are different ways that you might test the knowledge. You might see if anyone else can use the knowledge in the form you recorded it. You might also get an assessment of the value of the knowledge.

10. Integrate the knowledge

If you decide to integrate the knowledge into a larger repository, you must make sure the repository has hooks for this kind of knowledge and your knowledge has the necessary "eyes" for the hooks.

These ten steps represent a framework for establishing your own knowledge map. There are a couple of things you must consider to assure that your knowledge map is useful. First, you must follow a consistent discipline to grow your

map. These ten steps can be used to grow your knowledge map as well as to create it. Consistency is important because it will enhance your ability to retrieve knowledge from the map. Second, you must keep your knowledge map fresh. Knowledge maps can rapidly become stale if they are not used and if they don't evolve over time. If you create a knowledge map, it is important to be committed to not letting it go stale.

+++++

If you are interested in learning more about knowledge work, subscribe to this newsletter by sending email to:

knowldgWORKSNews-on@lists.webvalence.com.

To unsubscribe send an e-mail to: knowldgWORKSNews-off@lists.webvalence.com.

You may type an x in the subject or body if your e-mail program requires.

Previous issues of the knowldgWORKS News are archived at <http://www.accsys-corp.com>.

+++++

Published by Dr. Randy M. Kaplan, and ACCSYS Corporation. This newsletter is the property of ACCSYS Corporation. No part may be reproduced in any form without permission from ACCSYS Corporation. Copyright (c) 1999 ACCSYS Corporation. All rights reserved. All contributed work remains the property of the authors.