Group 1:
The points our group came up with were:

- Error-Proneness: If a user completes a process incorrectly, i.e. the output is accept instead of reject, then it is possible that many automated scripts are run. There is no clear way to undo the affects that will occur if an incorrect option is chosen.
- Principle of Semiotic Clarity: Process outputs are shortened to a single letter output in a plan. This single letter is ambiguous. This relates to the Principle of Perceptual Discriminability and to consistency. An example of this is shown in the Quality Assurance Plan where the Test Lead has output R=Reproduced while the Programmer has output R=Resolve_Rejected.
- Complexity Management: Sub plans provide a way to encapsulate complexity, they provide abstraction but also introduce hidden dependencies.
- The paper mentions obligations, dependencies between different plans that are not sub plans of each other. There is no visual indication of an obligation between sub plans, which causes both hidden dependencies and a lack of visual expressiveness.
- With respect to dual coding, the technique reduces the need for large amount of explanatory text by using sub plans. At a high level the text 'test feature' is sufficient, and if further details are required the sub plan can be examined.
- Viscosity: Adding additional processes to a plan appears to be simple, but there is no auto arrangement if the plan becomes crowded. The lack of auto arrangement though allows for layout order to be used as secondary notation.
- Graphics Economic: The number of symbols within the language is small, making it easy to learn. Despite this some improvements could be made to the and/or nodes, whose depiction is not particularly visually expressive.
- Visibiliity: The entire plan cannot be viewed in a single window. This is intended as abstraction is a major part of the design. There was no mention of whether a user could view multiple plans concurrently for comparison.

Group 2:

<u>COGNITIVE DIMENSIONS:</u>

- **Abstraction Gradient:**
  - high-level of abstraction
  - can have sub-maps of sub-maps
- **Closeness of Mapping:**
  - intuitive how it relates to the process
- **Consistency:**
  - Few symbols & pretty straightforward
  - Various symbols are confusing: such as the AND & OR
  - Took a while to understand what the multiple lines around the 'Review Teams' entity meant.
  - Sub-plans can show hidden dependencies.
- **Diffuseness:**
  - Every symbol has a specific meaning
- **Error-Proneness:**
  - Options are clear
- **Heard mental Operations:**

    - Pretty clear to see how the process works - just follow the paths with the steps of the process you're concerned with and it can be pretty easy to figure out.
- **Premature Commitment:**
    - Cannot move to next step unless you know the outcome (or options) of a specific stage therefore cannot really make a premature commitment.
- **Progressive Evaluation:**
    - You can tell what steps you have taken so far
    - You cannot tell how many times you have looped a stage. You can only tell where you are.
- **Role Expressiveness:**
    - Quite visible to see who's doing what.
- **Secondary Notation:**
    - Layout is used to group stages which are related. For example the testing and development stages are places in one corner while the management stage is placed in a distance to the rest.
- **Viscosity:**
    - Relatively easy to connect stages using lines and options.
    - With a high level of abstraction it makes it easier to insert sub-elements/sub-maps under a stage/element.
- **Visibility:**
    - Cannot view picture/process all in one go due to it's hierarchical structure (sub-maps).

__PHYSICS OF NOTATIONS:__

- **Semiotic Clarity:**
    - Start & Finish symbols are the same - although the finish can be used to mean the start of something else.
    - Different semantic constructs for entities with sub-maps?
- **Discriminability:**
    - Small circles (for 'options') can be hard to see and can be confused with stages.
- **Semantic Transparency:**
    - Start & Stop symbols look like Stop signs.
    - Arrows are confusing (Figure 5: arrow on condition node is pointing the wrong way)
- **Complexity Management:**
    - Have sub-maps to deal with complexity
- **Cognitive Integration:**
    - Can't see info on sub-maps from top-level
    - Finish symbol from sub-map changes to a Option symbol in the main process map
    - And symbol used as both a explicitor and combiner
- **Visual Expressiveness:**
    - No colour, shading, texture, etc. Just sizes are different. Could use more shapes.
- **Dual Coding:**
    - Text used to describe what options mean.
- **Graphic Economy:**

    - Manageable because not symbols are used.
- **Cognitive Fit**:
    - Not necessary in this case

Group 3:

Regatta Evaluations

1. The Regatta notation design is easy understandable, gives a clear view of relationships within roles. However one symbol was used to represent more than one meaning, as the graph below shows the marked symbols are exactly different operations. Also the Start and Processed nodes share the same hexagon symbol. The semiotic is not clarified and is Symbol Overload in details.

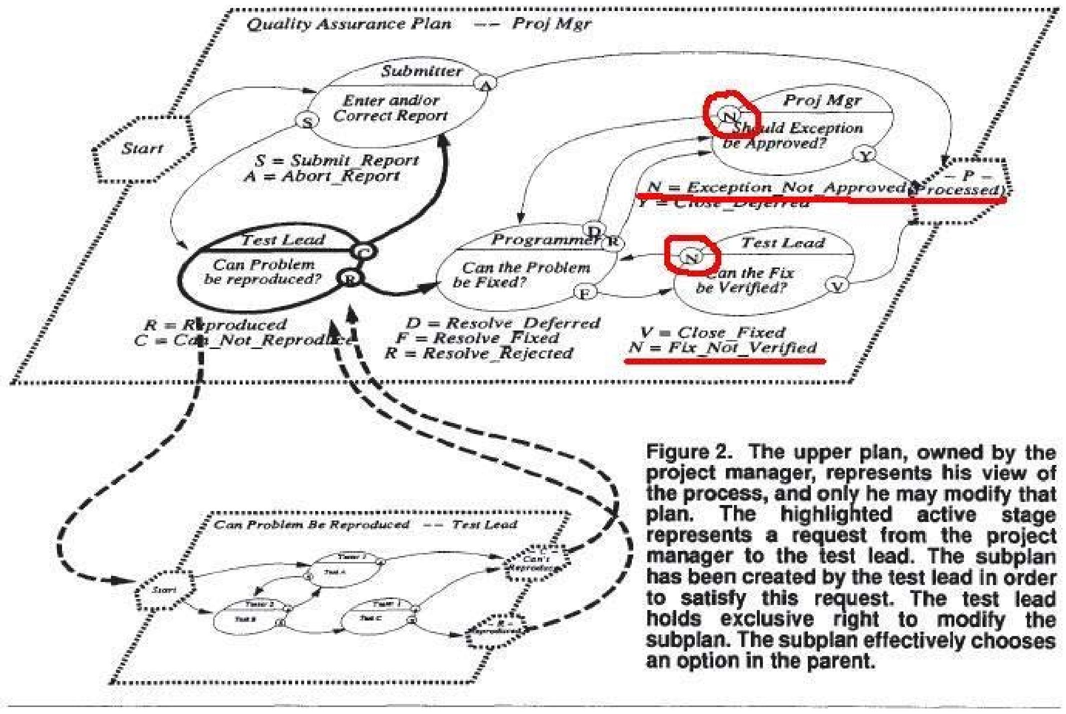

Figure 2. The upper plan, owned by the project manager, represents his view of the process, and only he may modify that plan. The highlighted active stage represents a request from the project manager to the test lead. The subplan has been created by the test lead in order to satisfy this request. The test lead holds exclusive right to modify the subplan. The subplan effectively chooses an option in the parent.

2. The Regatta hasn't been designed well with visual expressiveness. It only has a few symbols to represent complex tasks. Some shape or coloured symbols might be introduced to illustrate complex relationships clearer. However, the new symbol design needs to have a balance of diffuseness to make the design both compactly and clarify.
3. The Regatta has a good hierarchy design. The "Plan" and "Subplan" represent multiple levels of abstraction which are complexity manageable. However the increase of abstraction may affect the hidden dependencies, a small change of "Plan" part may affect the "Subplan". With the existing examples we can see, Regatta has two levels of hierarchy: the level "Plan" and "Subplan" ("Stages" might not be treated as another hierarchy since it is part of "Plan") which are enough for a common system design. We hope there will not be more hierarchies under "Subplan" which will significantly increase the difficulty of sorting out hidden dependencies.

4. A Secondary Notation might be introduced to have a clearer view of Regatta. Imaging to highlight the stages by processing sequences (the parallel processing might be grouped as one), we know exactly step by step of the whole "Plan", and Regatta might be easier understood by both experts and novices. In the other hand, it reduces the viscosity of diagram.
5. The Regatta uses arrow line and text explanation separately to illustrate relationships within "Options". It's quite good when doing complex designs because this can provide a neat view of whole diagram. However, when doing small or medium designs, why not to combine the relationship arrow line and text explanation together? Because, it's really uncomfortable to find the pairs of relationships and "Options". Therefore, a good balance between graphics and texts so called Hybrid Symbols (Principal of Dual Coding) need to be considered.
6. The Regatta has abstraction levels of Plan-Stage-Option, each Option can active one stage. If we design an "OR" operator (which means logic "or"), when two Options want to active one stage, they can be joined together with "OR" symbol then point to the destination Option. Therefore we have a new level of abstraction (Plan-Stage-Option-Operator), which brings an easier view of Regatta diagram (visibility increased).

Group4:

At the first glance the visual language looks complex but very well detailed. Those of us who failed to find/read the paper had trouble understanding some of the meanings behind the notations. But most of the notations in this visual language were straight forward.

When we went into detailed specifics of the VL, we began to notice lots of issues, and inconsistencies according to the "cognitive dimensions framework and physics of notation". We went through the 9 principles and evaluated how the VL achieved or failed to achieve these principles.

The first principle, principle of semiotic clarity requires 1:1 correspondence between constructs and symbols. Although the examples mostly showed 1:1 correspondence between constructs and symbols, we almost overlooked the AND/OR (circle with a + in the middle). It was pointed out that the 2 instances of the symbol appearing in the example actually represent 2 different constructs. For anyone not have memorised the key/meaning of these symbols this would be a very common mistake/assumption. The VL clearly needs to use a different symbol to represent these 2 constructs. We also discussed how the Start and End would benefit from different symbols as that would be very easy to read the flow of the diagram.

Secondly we looked at the second principle which states different symbols should be clearly distinguishable. Since the problem discussed above uses the same symbol for multiple constructs, this principle is not really broken. We thought the VL used somewhat clear and distinguishable symbols.

Thirdly it states that the symbols appearance should represent their meaning. The same issue discussed earlier where the use of the circle with + inside in most people's opinion represents somewhat an ADD operation. But in this VL they use it for splitting decisions/paths. This would surely confuse people, especially people who do not fully understand this VL in great detail.

Principle of complexity is pretty well managed in this case. The VL supports most of the basic functionality, parallelism, and even multiple layers.

Group 5:

Principle of Semiotic Clarity: There is redundancy with the use of the graphical symbols that represent the concept of an exit point i.e. the rhombus and the circles. Symbol overload exists because the start and exit concepts are represented by the same symbol.

Principle of Perceptual Discriminability: The symbols used are easy to discriminate against each other. Even though the exit symbols on the large ovals are both from the same family, their sizes are very different which makes it easy to differentiate their meaning. Because of the smaller circles location being on the border of the larger circle there is an implicit notion of relationship between the two.

Principle of Semantic Transparency: The visual representations used are very basic and do not imply any meaning. The exception to this is the use of arrows which are an accepted means of conveying a transition from one object to another.

Principle of Complexity Management: Subtasks can be created offsetting complexity issues. However it is not clear as to what extent the software tool offsets the *hidden dependencies* that will be created from sub tasking, e.g. the subtask will not be *visible* and so the tool needs to compensate for this somehow. There is no explicit mechanism for dealing with growing amounts of text that represent the meanings of symbols as the diagram grows. To offset this, the designers could have been more *visually expressive* with their symbols so that so much text wasn't necessary.

Visual Expressiveness: The diagrams are not very visually expressive. The position of the text could be located closer to the things they represent, to allow better mapping between the things they are trying to describe e.g. the text could be put along the arrows. Also the current exit symbols could be grouped into several generalised categories and symbols created for each of these. With both of these ideas, when someone looks at a diagram they can instantly recognise the general effect of an exit symbol and also determine its specific meaning by reading the description on the arrow.

Principle of Graphic Economy: The amount of symbols used is very cognitively manageable. However this means that the diagram is visually expressionless, adversely affecting the semantic transparency of the diagram.

Dual Coding: Text is used in a replacement rather than a supplementary manner, more work needs to be made into the expressiveness of the symbols.

Group 6:

Abstraction:

The VL allows subplan for any stage in the plan so that each level of the plan can have enough abstraction owned by different level of people.

Closeness of mapping:

Consistency:

The representation of input and out is very simple and easy to understand.

Diffuseness:

The VL represents the plan clearly except using the same symbol to represent different meaning of options

Error-proness:

Maybe it is not an easy task to recognise any looping flow in the plan

Hard mental operation:

The option symbol seems only symbolic and not perceptual enough to represent the condition.

Hidden dependency:

Having a subplan creates dependency to the entire plan as a whole. A reader may need to go deep in order to understand the whole plan. However, this hidden dependency enhances collaboration which allows the readers with different knowledge level to be assigned to the related subplan.

Premature Commitment

Progressive Evaluation

The VL allows the the task to be executed before its detail plan is built.

Role-expressiveness:

It is very easy to understand what kind of condition is required for a stage to produce a result and which next recipient it will goes to.

Secondery notation:

Viscosity

If there is a change on option condition in a parent level stage, it may require an effort to go deep into different level of its subplan to adapt to the change. However, a change only related to the flow order is required, the subplan will not be affected at all which can be seen as a advantage.

Visibility:

The VL allows visual representation of parallel execution. The subplan hides the some detail of a stage which saves some space

Trade-off:

High abstraction may increase hidden dependencies as the VL allows unlimited level of subplan if a creator needs to have.

High abstraction may enhance visibility as it would simplified the plan into different level which becomes understandable to the reader with different knowledge level

High abstraction may increase viscosity if a creator wants to do some change on the options because changing the meaning of an option on a stage may require a fundamental change to its subplan. On the other hand, the abstraction may decrease viscosity as the creator may not need to worry too much about its subplan detail to add or subtract any of the stage.