Problem Statement Requirements

Due Date: Friday, Sept 20, 2002 before mid-night and at the BEGINNING of class on Friday, September 20, 2002.

Hand in the following (one per group):

When submitting a problem description, you should use the following structure as a guide.

- **Abstract**
  This concise presentation of your problem will be the basis for the participants to choose the problem they want to work on.

- **Description of a domain**
  This description can be accompanied by pictures, images, text or whatever. The purpose of this description is to explain the context in which the problem is placed. The text should define any terminology and should be understandable by laymen. All abbreviations must be explained, if possible in a glossary.

- **Block diagrams (Optional)**
  A block diagram of the problem domain, or any other type of diagram which may serve as a basis for the analysis and design, should be included here.

- **Description of the program that is wanted**
  This should include the goals of the program and the required functions. It should state what the program should achieve without enforcing any solutions.

- **Detailed requirements**
  Each team must create a more detailed 2-3 page of detailed requirements. Because of time limitations these should of course be restricted. An indication of performance and capacity requirements should be given.

- **Use cases and User Context**
  Each team must provide description of what your users do, how their need for the system arose, and what they would like to see to solve their problem. Use your imagination and put yourself in the user’s context. Describe all the use case in your system or meaningful handful of use cases. Use cases are well suited to provide hints for both the analysis and design processes, as design teams tend to work on them first. Description of the work context for the user/customer's point-of-view: why a system is needed, and what the system will do for them.

- **Interfaces**
  Descriptions of interfaces to other systems. This is of course optional when no interfaces to other systems are necessary.

- **References for further study**
  These are references to articles, other similar systems or implementations.

Sample Problem Statements or Requirements:

See sample problems on the web --- Check OOPSLA’s DesignFests
Grading Criteria at the end of the semester as part of the final project

<table>
<thead>
<tr>
<th>Item</th>
<th>% of Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustrations</td>
<td>15%</td>
</tr>
<tr>
<td>Completeness of requirements</td>
<td>40%</td>
</tr>
<tr>
<td>Organization</td>
<td>30%</td>
</tr>
<tr>
<td>Creativity</td>
<td>15%</td>
</tr>
</tbody>
</table>

1. **Illustrations**: This refers to any illustrated models, such as pictures, images, diagrams, that provide clear definition of the problem domain and/or system architecture.
2. **Completeness of requirements**: This refers to how completely the group has described the user context and requirements. Make sure all the possible scenarios or use cases are covered, including error recovery. For example, a requirement document for an e-mail system that leaves out the ability to reply to mail messages would be downgraded (of course, it may be a requirement that the user cannot reply to mail -- but that needs to be part of the requirement).
3. **Organization**: This refers to how well-organized and readable the requirements document is. If related requirements are strewn throughout the document, or the document is written poorly, it will be downgraded. Good organization that clarifies the requirements is desired.
4. **Creativity**: Creative requirements will be rewarded. Try to come up with some good ideas that fit the user's context.