Documenting using Doxygen

Benny Åkesson

29 oktober 2003

1 Introduction to Doxygen

Doxygen is a system for documenting code. For a programmer already familiar with Java it can be described as a super set to Javadoc with more tags and support for more languages such as C, C++ and C#. The added functionality allows the traditional API view with class and method listings to be extended with normal text pages which comes in very handy when creating a reference manual.

Doxygen has the ability to create a reference manual in many useful formats. The same lines of documentation can be used to generate HTML, latex, RTF, XML and man page output. Information specific to every format can be specified with tags to allow, for example, HTML formatting and for pictures to be present in some formats but not in all of them.

Using Doxygen the reference manual is expressed directly in the files containing the source code keeping all information in one place. This means that properly documenting classes and methods pays off since those very lines of comments appear in the manual automatically.

Doxygen is easy to use for a number of reasons. Anyone with experience from Javadoc will notice that almost everything they know will be applicable when using Doxygen and learning about the extra functions and tags is easy using the online manual. It's also convenient that the output can be formatted using latex or HTML tags so one can use whatever comes naturally. Apart from being easy to use Doxygen is also very expressive and configurable. Images and latex mathematical formulas may be used which allows the reference manual to look good and be well suitable to express the mathematics of the problems properly. Inheritance diagrams are automatically generated and more graphs can be included if some external software is included. Source code may be inlined in the documentation and examples can easily be included.

Doxygen is free software and is available on most platforms.

2 Designing with Doxygen

We intend to use Doxygen all the way in the project. That means we will use it when designing as well as during the main implementation. We want you to design by creating and documenting classes and methods that you will need as stub code. The methods and classes created will have to be commented in order for the design to be presentable. With only little time available for design we will not require you to describe all parameters and return values properly but if you

can manage you will save a lot of time in the end. Remember to go through the Java API properly when designing with 0 standard classes or you might end up creating a design that will not be sufficient. Blocking methods calls are common when dealing with networks and streams so more threads may be required than one might initially think.

Apart from delivering your design as PDF output we want you to provide an UML class diagram to create overview. This diagram may be drawn by hand or using any tool you see fit. An online HTML version of the design is not necessary but would be pleasing and can be done with very little extra work.

The PDF file is created from the latex output using the automatically generated Makefile when you type `make pdf` in the latex directory that is created when Doxygen is run (provided that latex output is enabled in the config file).

If you publish your documentation online there is no need to inline code in the generated documentation. All code will be present in the HTML version if it's properly documented. Not inlining code saves some paper and goes easy on your printing account.

The benefits of using Doxygen while designing is that the documentation created during the design will be available directly in the code when the implementation starts. That gives you a head start in documenting your project.

Remember to write the name of your tutor before handing it in unless it's delivered personally.

3 Where to go and what to do

To behold the power of Doxygen please refer to the deadlock detection application that may be downloaded from the course homepage. The application provides HTML and PDF output which clearly shows how Doxygen uses the same information to create reference manuals for very different purposes. Check out the file `ResourceKeeper.java` in the source code to see how the documentation was created. The contents of that file may be used as a template for your own documentation.

You will also need to familiarize yourselves with the Doxygen manual which is available online. Use your favorite browser and connect to `http://www.doxygen.org`.

4 Examples

4.1 Class documentation

```
/**
 * This class is a monitor that keeps track of resource allocations.
 * Concurrently running threads should request their resource allocations
 * through a keeper to get blocked in a resource queue until the requested
 * resource is ready. Once the resource is deallocated by the currently
 * running thread it will be allocated to the next one in the resource
 * queue. The resource keeper keeps track of allocations and will notify
 * when an allocation may result in a deadlock.
 */
public class ResourceKeeper {
```

4.2 Method documentation

```
/**
 * This method dumps the names of the threads queuing in a resource
 * queue along with the name of the resource to which the queue
 * belongs. This is used for output generation.
 *
 * @param theQueue the resource queue to dump into a string
 * @param id the name of the resource to which the queue belongs
 *
 * @return a string representation of the queue
 */
private synchronized String queueToString(LinkedList theQueue, String id) {
```

4.3 Text pages

The example page below has been cut down somewhat so that there will be more emphasis on the Doxygen tags. The contents of the page may, for that reason, not make much sense. The HTML tables are not necessary but it makes the HTML output look nicer. References between pages are done more easily using the `ref`-tag.

```
/**
 \page scenarios Scenarios and Scenario Description Files

 \htmlonly
 <table width="600" align=center>
 <tr><td>
 \endhtmlonly

 \section scenariosection Scenarios

 A scenario could be described as ...

 The predefined scenarios are stored along with the source code.
 Uncompress the source and step into /deadlock/simulation/scenarios.

 \section scenariofile Scenario Description Files

 \subsection scenariofile_overview Scenario Description File overview

 Scenario Description Files are ...

 <i>example:</i> \n
 <tt># This is a comment and will be ignored by the parser.</tt>\n

  An allocation scheme is defined by ...

 \htmlonly
 <table width="100%">
 <tr>
```

```
<td align="left">
<a href="howtouse.html">How to use</a>
</td>
<td align="right">
<a href="compile.html">Compiling and running the simulator</a>
</td>
</tr>
</table>
</td></tr></table>
\endhtmlonly
*/
```