

LIST OF FIGURES

Figure 2-1: A tree data structure and two aliasing references to its internal nodes.	16
Figure 2-2: A local call can affect all reachable data	17
Figure 2-3: Call-by-reference semantics can be maintained with remote references.	18
Figure 2-4: State after steps 1 and 2 of the algorithm. Remote procedure <code>foo</code> has performed modifications to the server version of the data.	22
Figure 2-5: State after steps 3 and 4 of the algorithm. The modified objects (even the ones no longer reachable through <code>tree</code>) are copied back to the client. The two linear representations are “matched”—i.e., used to create a map from modified to original versions of old objects.	22
Figure 2-6: State after step 5 of the algorithm. All original versions of old objects are updated to reflect the modified versions.	22
Figure 2-7: State after step 6 of the algorithm. All new objects are updated to point to the original versions of old objects instead of their modified versions. All modified old objects and their linear representation can now be deallocated. The result is identical to Figure 2-3.	22
Figure 2-8: Changes after execution of method.	25
Figure 2-9: Under DCE RPC, the changes to data that became unreachable from <code>t</code> will not be restored on the client site.	26
Figure 2-10: NRMIzer GUI.	36
Figure 3-1: Simplified fragment of XDoclet template to generate the aspect code. Template parameters are shown emphasized. Their value is set by XDoclet based on program text or on user annotations in the source file.	48
Figure 4-1: Example user interaction with J-Orchestra. An application controlling speech output is partitioned so that the machine doing the speech	

synthesis is different from the machine controlling the application through a GUI.	63
Figure 4-2: Results of the indirect reference approach schematically. Proxy objects could point to their targets either locally or over the network.	68
Figure 4-3: J-Orchestra classification criteria. For simplicity, we assume a “pure Java” application: no unmodifiable application classes exist.	71
Figure 4-4: Results of the J-Orchestra rewrite schematically. Proxy objects could point to their targets either locally or over the network.	80
Figure 4-5: Mobile code refers to anchored objects indirectly (through proxies) but anchored code refers to the same objects directly. Each kind of reference should be derivable from the other.	81
Figure 4-6: The results of a query on the creation sites of class p.MyThread.	88
Figure 4-7: The zigzag deadlock problem in Java RMI.	97
Figure 4-8: Using thread id equivalence classes to solve the “zigzag deadlock problem” in Java RMI.	101
Figure 4-9: The appletizing perspective code view of a centralized Java GUI application.	116
Figure 4-10: Violating the Swing threading design invariant: someGUIOp method is invoked on a thread different than <i>Event-Disp-Thread</i> , if no special care is taken.	122
Figure 4-11: An automatically generated HTML file for deploying the appletized Jarminator application.	125
Figure 5-1: UML class diagram of the Thermal Plate Simulator functionality.	158
Figure 5-2: Kimura architecture: (a) the original system; (b) the reengineered Kimura2 system.	169
Figure 6-1: (a): Original system classes hierarchy.	182
Figure 6-1: (b): Replicating system classes in a user package (“UP”).....	182
Figure 6-2: (a): Original system class File (with a native method) and subclass TXFile (without native dependencies).	196

Figure 6-2:	(b): Result of the user-level indirection transformation, with safe access to non-public fields of class File.....	196
Figure 6-3:	(a): A File class hierarchy.....	197
Figure 6-3:	(b): Removing subclassing restrictions.....	197
Figure 8-1:	(a): A general remote call mechanism: a subset of the client heap, reachable from p , can be sent to the server, to be updated against a subset of the server heap.....	220
Figure 8-1:	(b): A general remote call mechanism: param p is returned to the client and restored in place.	220