## RIVAR -- Rich Interfaces for Verifiable Aspect Reuse

Collection of empirical data on assumptions made by aspect programmers about the context in which their aspects will be woven.

In the table below, enter information for each advise on a separate line. Use additional lines for different assumptions. Enter assumptions in English text giving as much detail as needed to completely describe the assumption. Coding and classification will be performed in a separate step.

Project: *GlassBox*
Version: *2.0, downloaded 18/02/2010, focusing on monitor subproject; AOP@Work articles*

Adivce data:

| File | Lines | Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.) | Assumption Description | Comment | Ron's Comments |
|---|---|---|---|---|---|
| g.agent.ErrorContainment.aj | 39--48 | AOP@Work article; Interpretation of Code | Assumes any around advice will be suitably enclosed in subaspects named '*..*Around' so that it will not be matched by this pointcut. | See AOP@Work article for an explanation why we do not want to contain exceptions in around advice. | This is a work-around for an AspectJ compiler limitation that you can't exclude around advice in the AspectJ pointcut language. |
| g.config.EagerConfiguration.aj | 14--17 | Comment | This aspect is packaged with the monitor but is NOT deployed by default. | | |
| g.config.extension.api.PluginTracking.aj | | | | No relevant assumptions as far as I can see. | |
| g.monitor.resource.AbstractFtpMonitor.aj | 19--20 | Interpretation of Code | Assumes the super aspect AbstractMonitor uses the monitorPoint pointcut to identify the behaviour to be monitored. | | |
| g.monitor.resource.AbstractFtpMonitor.aj | 22--24 | Interpretation of Code | Assumes that the object exposed by monitorPoint will also be handed as a parameter to getKey() | | |
| g.monitor.resource.BerkeleyDbMonitor.aj | 14--16; 18--20; 22--24 | Interpretation of Code | Assumes implementing MonitoredType will ensure Glassbox is started when an instance of this type is created. | This assumes bootstrap/glassbox.config.AutoInitialization.aj to be deployed | It is also possible to initialize Glassbox through other means; in fact that's the norm. So this aspect only depends on having initialized the system. In general, Glassbox is coded to allow it to run without effect until the system has been properly initialized, although until initialization, it has no effect (the aspects are disabled). Bugs are possible, of course. This same comment applies to all the cases below of the same assumption. |
| g.monitor.resource.BerkeleyDbMonitor.aj | 14--16; 18--20; 22--24 | Comment; Interpretation of Code | Assumes MonitoredType will be used as a marker interface by LogManagement.aj to switch off logging, if logging is deployed. | This is an interesting assumption, because a marker interface that has been defined explicitly for one aspect is implicitly used by another aspect. However, given the specific comment '/*don't manage logging for this*/' in BerkeleyDbMonitor, this is an explicitly made assumption rather than something that happens and that the base code (i.e., this aspect) can be oblivious of. | |
| g.monitor.resource.BerkeleyDbMonitor.aj | 51--53; 55--59; 61--63; with 46--47 | Interpretation of Code | Assumes that monitorEnd() is advised to endNormally/endException for the current response created via createResponse | This doesn't use the monitorStart()/monitorPoint() abstract pointcuts defined in the super aspect, because they have bundled all handling of BerkeleyDB stuff into one aspect. | |
| g.monitor.resource.BerkeleyDbMonitor.aj | 82--84 | | | Not sure why this has been overridden at all. It would appear it is never called. In any case, the layer returned isn't exactly what is needed anyway. | Yes it is older code that could be deleted. |
| g.monitor.resource.BerkeleyXmlDbMonitor.aj | 32--34; 36--38; 40--41 | Comment; Interpretation of Code | Assumes MonitoredType will be used as a marker interface by LogManagement.aj to switch off logging, if logging is deployed. | This is an interesting assumption, because a marker interface that has been defined explicitly for one aspect is implicitly used by another aspect. However, given the specific comment '/*don't manage logging for this*/' in BerkeleyDbMonitor, this is an explicitly made assumption rather than something that happens and that the base code (i.e., this aspect) can be oblivious of. | |
| g.monitor.resource.BerkeleyXmlDbMonitor.aj | 32--34; 36--38; 40--41 | Interpretation of Code | Assumes implementing MonitoredType will ensure Glassbox is started when an instance of this type is created. | This assumes bootstrap/glassbox.config.AutoInitialization.aj to be deployed | |
| g.monitor.resource.BerkeleyXmlDbMonitor.aj | 94--100; 108--114; 122--134; 150--154; 156--172 | Interpretation of Code | Assumes that monitorEnd() is advised to endNormally/endException for the current response created via createResponse | This doesn't use the monitorStart()/monitorPoint() abstract pointcuts defined in the super aspect, because they have bundled all handling of BerkeleyXmlDB stuff into one aspect.<br>Particularly interesting for the advice on lines 156--172, as this actually doesn't create a response for every join point. Some interesting use of if-pointcut in the definition of monitorEnd here! | |
| g.monitor.resource.BerkeleyXmlDbMonitor.aj | 174--176 | | | Not sure why this has been overridden at all. It would appear it is never called. In any case, the layer returned isn't exactly what is needed anyway. | |
| g.monitor.resource.BufferFlushMonitor.aj | 23--25 | Interpretation of Code | Assumes super aspect to endNormally/endException after monitorEnd | This might be a bit nitpicky. However, it still is an assumption on how the super-aspect works. | |

## RIVAR -- Rich Interfaces for Verifiable Aspect Reuse

Collection of empirical data on assumptions made by aspect programmers about the context in which their aspects will be woven.

In the table below, enter information for each advise on a separate line. Use additional lines for different assumptions. Enter assumptions in English text giving as much detail as needed to completely describe the assumption. Coding and classification will be performed in a separate step.

Project: *GlassBox*
Version: *2.0, downloaded 18/02/2010, focusing on monitor subproject; AOP@Work articles*

Adivce data:

| File | Lines | Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.) | Assumption Description | Comment | Ron's Comments |
|---|---|---|---|---|---|
| g.monitor.resource.CommonsHttpMonitor.aj | 18--24 | Comment; Interpretation of Code | Assumes MonitoredType will be used as a marker interface by LogManagement.aj to switch off logging, if logging is deployed. | This is an interesting assumption, because a marker interface that has been defined explicitly for one aspect is implicitly used by another aspect. However, given the specific comment '/*don't manage logging for this*/' in BerkeleyDbMonitor, this is an explicitly made assumption rather than something that happens and that the base code (i.e., this aspect) can be oblivious of. | |
| g.monitor.resource.CommonsHttpMonitor.aj | 18--24 | Interpretation of Code | Assumes implementing MonitoredType will ensure Glassbox is started when an instance of this type is created. | This assumes bootstrap/glassbox.config.AutoInitialization.aj to be deployed | |
| g.monitor.resource.CommonsHttpMonitor.aj | 44--56 | Interpretation of Code | Assumes all responses thus opened will be closed correctly by the advice for monitorEnd. In particular, here this seems to assume that executeOnMethod1 and executeOnMethod2 are a complete decomposition of executeOnMethod. | Interesting: This could have been made safe by simply removing the explicit advice and renaming monitorEnd into monitorPoint (providing the parameters and using executeOnMethod1 and executeOnMethod2 explicitly). Has this not been done because the implementor of CommonsHttpMonitor didn't know this was an option? Or is there another reason? | Are you suggesting code like this:<br>protected pointcut monitorPoint(Object httpMethod) :<br>topLevelExecuteOnMethod() && (args(httpMethod, ..) &&<br>!args(org.apache.commons.httpclient.HostConfiguration, ..) \|\|<br><br>args(org.apache.commons.httpclient.HostConfiguration, httpMethod, ..));<br><br>Unfortunately, the AspectJ compiler won't accept that, giving an error:<br><br>ambiguous binding of parameter(s) identifier across '\|\|' in pointcut CommonsHttpMonitor.aj<br><br>The only way executeOnMethod can match but not 1 or 2 is a no-arg method. I've now fixed this issue by requiring at least 1 argument in executeOnMethod (thereby changing from an assumption to a tautology), since it's better to have consistent advice on begin/end if the base system evolves in unanticipated ways. |
| g.monitor.resource.EjbCallMonitor.aj | 36--50 | Interpretation of Code | Assumes all responses thus opened will be closed correctly by the advice for monitorEnd. | This is essentially the case. monitorEnd() adds '&& this(Object)', which excludes static calls, similarly, the two before advices include this(ejb) which excludes static calls. | Basically this is assuming that javax.ejb.EJBObject and javax.ejb.EJBHome extend java.rmi.Remote, which has been true for more than 12 years and is unlikely to change. |
| g.monitor.resource.EjbOperationMonitor.aj | 32--38 | Interpretation of Code | Assumes all responses thus opened will be closed correctly by the advice for monitorEnd. | | |
| g.monitor.resource.EjbOperationMonitor.aj | 24 | Interpretation of Code | Assumes request enabling is required at this level. | This is both an assumption on the behaviour of topLevelPoint in the super aspect and on the control flow in the base that implies that this is the right point to do this. | |
| g.monitor.resource.EmailMonitor.aj | 16--17 | Interpretation of Code | Assumes that monitorPoint() defines points to be monitored using a response structure. | An assumption on the super aspect. | |
| g.monitor.resource.JakartaFtpMonitor.aj | 13--18 | Interpretation of Code | Assumes defining this pointcut will define a monitor point | An assumption on the super aspect. | |
| g.monitor.resource.JakartaFtpMonitor.aj | 24--26 | Interpretation of Code | Assumes openConnection will appropriately be used to close responses as well. | | |
| g.monitor.resource.JakartaFtpMonitor.aj | 28 | Interpretation of Code | Assumes the super aspect uses monitorPoint rather than monitorStart/monitorEnd to define points to be monitored. | | |
| g.monitor.resource.JaxmMonitor.aj | 19 | Interpretation of Code | Assumes that monitorPoint() defines points to be monitored using a response structure. | | |
| g.monitor.resource.JaxmMonitor.aj | 21--23 | Interpretation of Code | Assumes the argument of monitorPoint will be passed on to getKey() | | |
| g.monitor.resource.JdbcMonitor.aj | 35--52 | Comment; Interpretation of Code | Assumes MonitoredType will be used as a marker interface by LogManagement.aj to switch off logging, if logging is deployed. | This is an interesting assumption, because a marker interface that has been defined explicitly for one aspect is implicitly used by another aspect. However, given the specific comment '/*don't manage logging for this*/' in BerkeleyDbMonitor, this is an explicitly made assumption rather than something that happens and that the base code (i.e., this aspect) can be oblivious of. | |
| g.monitor.resource.JdbcMonitor.aj | 35--52 | Interpretation of Code | Assumes implementing MonitoredType will ensure Glassbox is started when an instance of this type is created. | This assumes bootstrap/glassbox.config.AutoInitialization.aj to be deployed | |
| g.monitor.resource.JdbcMonitor.aj | 153--215 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.resource.JndiMonitor.aj | 30--31 | Interpretation of Code | Assumes that monitorPoint() defines points to be monitored using a response structure. | | |

## RIVAR -- Rich Interfaces for Verifiable Aspect Reuse

Collection of empirical data on assumptions made by aspect programmers about the context in which their aspects will be woven.

In the table below, enter information for each advise on a separate line. Use additional lines for different assumptions. Enter assumptions in English text giving as much detail as needed to completely describe the assumption. Coding and classification will be performed in a separate step.

Project: *GlassBox*
Version: *2.0, downloaded 18/02/2010, focusing on monitor subproject; AOP@Work articles*

Adivce data:

| File | Lines | Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.) | Assumption Description | Comment | Ron's Comments |
|---|---|---|---|---|---|
| g.monitor.resource.JxtaOperationMonitor.aj | 11--12 | Interpretation of Code | Assumes super aspect will use classControllerExecTarget | | |
| g.monitor.resource.JxtaSocketMonitor.aj | 22--23 | Interpretation of Code | Assumes that monitorPoint() defines points to be monitored using a response structure. | | |
| g.monitor.resource.JxtaSocketMonitor.aj | 25--31 | Interpretation of Code | Assumes JxtaSocketMonitor has precedence over AbstractMonitor (otherwise the implementation of getKey would not work) | Note that this is standard AspectJ semantics, so the real assumption here is that this precedence relations is not changed by any other aspect through an explicit declare precedence. | Good point. It's safer to make the assumption an explicit requirement, so I added<br>declare precedence: JxtaSocketMonitor, AbstractMonitor; |
| g.monitor.resource.JxtaSocketMonitor.aj | 37--43 | Interpretation of Code | Assumes the argument of monitorPoint will be passed on to getKey() | | |
| g.monitor.resource.LogMonitor.aj | 18 | Interpretation of Code | Assumes that monitoredPublicMethods() defines methods to be monitored. | | |
| g.monitor.resource.RemoteCallMonitor.aj | 37--41 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | Interesting case, because monitorEnd potentially matches more joinpoints as it also includes static calls. | Good point. Improved by changing the relevant code to avoid assumptions, like so:<br><br>public pointcut remoteExecution(Remote remote) :<br>within(Remote+) && execution(public * *(..) throws RemoteException) && this(remote);<br><br>public pointcut endPoint(Remote remote) :<br>!within(javax.ejb.EJBObject+) && !within(javax.ejb.EJBHome+) && remoteExecution(remote);<br><br>protected pointcut monitorEnd() : endPoint(*);<br><br>before(Remote remote) : endPoint(remote) { |
| g.monitor.resource.SftpMonitor.aj | 17--19 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.resource.SftpMonitor.aj | 17--21 | Interpretation of Code | Assumes AbstractFtpMonitor uses monitorPoint to define its own measurements rather than monitorBegin/monitorEnd | | |
| g.monitor.ui.DwrMonitor.aj | 41--47 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.ui.GwtMonitor.aj | 28--34 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | Interesting case, because monitorEnd potentially matches more joinpoints as it also includes static calls. | Also fixed by requiring this in the base pointcut. |
| g.monitor.ui.MvcFrameworkMonitor.aj | 53--63 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.ui.PortletMonitor.aj | 24--34 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.ui.ServletRequestMonitor.aj | 70 | Comment; Interpretation of Code | Assumes MonitoredType will be used as a marker interface by LogManagement.aj to switch off logging, if logging is deployed. | This is an interesting assumption, because a marker interface that has been defined explicitly for one aspect is implicitly used by another aspect. However, given the specific comment '/*don't manage logging for this*/' in BerkeleyDbMonitor, this is an explicitly made assumption rather than something that happens and that the base code (i.e., this aspect) can be oblivious of. | |
| g.monitor.ui.ServletRequestMonitor.aj | 70 | Interpretation of Code | Assumes implementing MonitoredType will ensure Glassbox is started when an instance of this type is created. | This assumes bootstrap/glassbox.config.AutoInitialization.aj to be deployed | |
| g.monitor.ui.ServletRequestMonitor.aj | 76 | Interpretation of Code | Assumes all points measured also are top-level entry points | | |
| g.monitor.ui.ServletRequestMonitor.aj | 78--85;<br>107--139 | Interpretation of Code | Assumes all responses thus opened will also be closed by the super aspect again. | | |
| g.monitor.ui.SpringMvcRequestMonitor.aj | 32--33;<br>41--42 | Interpretation of Code | Assumes setting these pointcuts will lead to desirable logging | An assumption on the super aspect. | |
| g.monitor.ui.StrutsRequestMonitor.aj | 45--49 | Interpretation of Code | Assumes setting these pointcuts will lead to desirable logging | An assumption on the super aspect. | |
| g.monitor.ui.TemplateOperationMonitor.aj | 23--24;<br>28--29 | Interpretation of Code | Assumes setting these pointcuts will lead to desirable logging | An assumption on the super aspect. | |
| g.monitor.AbstractHandlerTracking.aj | 9 | Interpretation of Code | Assumes that subaspects will define scope such that this does not conflict with other exception handling, especially where this uses recordException, too. | This is a bit vague at the moment and needs more analysis | If more than one method handles exceptions, it will just record the state more than once - that might be valid, although the assumption would be that a subaspect overriding the base knows what it's doing. |
| g.monitor.AbstractMonitor.aj | 33--35;<br>44--50 | Interpretation of Code | Assumes monitorBegin() and monitorEnd() are matched up so that as many Resources are created as are removed within one control flow. Also assumes that this match up leads to correct nesting. | There's some rudimentary checking takes place in AbstractMonitorClass.getValidResponse, but it doesn't actually enforce proper nesting completely. | Indeed, I don't think there are good options for adding more explicit checking of paired begin/end responses It might be better to require explicit identification of a unique id for a given type of response that is begun or ended, adding some programming overhead to the monitor interface to reduce the risk of mismatch. In practice, this assumption is problematic and has been a significant source of problems, both in system initialization scenarios and in debugging new monitors. Any other thoughts for how to avoid such assumptions? |
| g.monitor.AbstractMonitorControl.aj | | | | No relevant assumptions as far as I can see. | |

**RIVAR -- Rich Interfaces for Verifiable Aspect Reuse**

Collection of empirical data on assumptions made by aspect programmers about the context in which their aspects will be woven.

In the table below, enter information for each advise on a separate line. Use additional lines for different assumptions. Enter assumptions in English text giving as much detail as needed to completely describe the assumption. Coding and classification will be performed in a separate step.

Project: *GlassBox*
Version: *2.0, downloaded 18/02/2010, focusing on monitor subproject; AOP@Work articles*

Adivce data:

| File | Lines | Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.) | Assumption Description | Comment | Ron's Comments |
|---|---|---|---|---|---|
| g.monitor.MethodMonitor.aj | | | | No relevant assumptions as far as I can see. They just go and ignore almost all the super aspect provides. However, this makes the above assumption about nesting of resources even more interesting, as it is now completely out of the control of AbstractMonitor | Yes the method monitor replaces the machinery of the AbstractMonitor to provide for a lower overhead monitoring mechanism. However, it assumes that subaspects will use its pointcuts rather than directly invoking begin/end methods, since getValidResponse isn't used here. That is a design flaw that should be addressed. |
| g.monitor.NativeMonitor.aj | | | | No relevant assumptions as far as I can see. | |
| g.policy.ApiPolicy.aj | | | | No relevant assumptions as far as I can see. | |
| g.policy.ContractChecking.aj | | | | No relevant assumptions as far as I can see. | |
| g.response.DefaultResponseFactory.aj | | | | No relevant assumptions as far as I can see. | |
| g.response.ResponseFactory.aj | | | | No relevant assumptions as far as I can see. | |
| g.response.ResponseInvariants.aj | | | | No relevant assumptions as far as I can see. | |
| g.thread.context.MonitorContextLoaderManagement.aj | | | | No relevant assumptions as far as I can see. | |
| g.util.jmx.JmxManagement.aj | | | | No relevant assumptions as far as I can see. | |
| g.util.jmx.MonitorJmxManagement.aj | | | | No relevant assumptions as far as I can see. | |
| g.util.logging.api.LogManagement.aj | 65--136 | Interpretation of Code | Expects base code to be aware of this aspect so that it knows it can call these ITDs. | | |
| g.util.SimpleObserverProtocol.aj | | | | No relevant assumptions as far as I can see. | The base code needs to be aware of this aspect to be able to call these ITDs, also. |