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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.alternative.AbstractAlternativeFeature.aj				No relevant assumptions as far as I can tell.
l.m.alternative.MusicSelector.aj	49--62	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.alternative.MusicSelector.aj	49--62	Interpretation of the Code	Assumes Display.getDisplay(controller.midlet).getCurrent() returns a list, and in particular a list that shows a selection of different media that are marked up as 'Music' or sth else.	
l.m.alternative.MusicSelector.aj	49--62	Interpretation of the Code	Assumes setMusicController has been invoked on controller from outside this aspect.	This is an inter-aspect dependency: It assumes that either l.m.alternative.PhotoAndMusicAndVideo or l.m.alternative.photoMusic.PhotoAndMusicAspect have been deployed, as they are setting the value in after advice for the startApp pointcut.
l.m.alternative.MusicSelector.aj	49--62	Interpretation of the Code	Assumes setMusicAlbumData has been invoked on controller from outside this aspect.	This is an inter-aspect dependency: It assumes that either l.m.alternative.PhotoAndMusicAndVideo or l.m.alternative.photoMusic.PhotoAndMusicAspect have been deployed, as they are setting the value in after advice for the startApp pointcut.
l.m.alternative.MusicSelector.aj	49--62	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.alternative.OneAlternativeFeature.aj	18--20	Interpretation of the Code	Assumes that no other code adds an 'Exit' command to the menu.	
l.m.alternative.PhotoAndMusicAndVideo.aj	23--24			Why does this not inherit from AbstractAlternativeFeature?
l.m.alternative.PhotoAndMusicAndVideo.aj	26--57	Interpretation of the Code	Assumes intertype declarations from a number of other aspects.	MusicSelector, PhotoSelector, VideoSelector, but also some others yet to be determined.
l.m.alternative.PhotoAndMusicAndVideo.aj	65--76	Interpretation of the Code	Assumes that the advice after startApp is always executed before this advice in any run of the application.	
l.m.alternative.PhotoSelector.aj	25--38	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.alternative.PhotoSelector.aj	25--38	Interpretation of the Code	Assumes Display.getDisplay(controller.midlet).getCurrent() returns a list, and in particular a list that shows a selection of different media that are marked up as 'Photos' or sth else.	
l.m.alternative.PhotoSelector.aj	25--38	Interpretation of the Code	Assumes presence of imageController declaration and that it has been set appropriately before this advice is invoked.	This is an inter-aspect dependency: It assumes that l.m.alternative.TwoAlternativeFeatures and l.m.alternative.PhotoAndMusicAndVideo or PhotoAndMusicAspect have been deployed.
l.m.alternative.PhotoSelector.aj	25--38	Interpretation of the Code	Assumes imageAlbumData has been declared and set appropriately.	This is an inter-aspect dependency: It assumes that l.m.alternative.TwoAlternativeFeatures and either l.m.alternative.PhotoAndMusicAndVideo or l.m.alternative.photoMusic.PhotoAndMusicAspect have been deployed, as they are setting the value in after advice for the startApp pointcut.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.alternative.PhotoSelector.aj	25--38	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.alternative.TwoAlternativeFeaturs.aj	32--34	Interpretation of the Code	Assumes that no other code adds an 'Back' command to the menu.	
l.m.alternative.VideoSelector.aj	49--56	Interpretation of the Code	Assumes that the return value of handleCommandAction is not important and that video selection should always take priority.	Apart from this, also makes a lot of assumptions similarly to the other XYSelector aspects above. However, this one is interesting as it is different from the assumptions up there and it is not immediately clear why this would be so.
l.m.alternative.music.AbstractMusicAspect.aj				No relevant assumptions as far as I can tell.
l.m.alternative.music.MusicAspect.aj	64--78	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.alternative.music.MusicAspect.aj	64--78	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.alternative.music.MusicAspect.aj	64--78	Interpretation of the Code	Assumes that the 'Play' command has been added to the menu, that is that the advise at lines 156--161 and 167--171 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.alternative.music.MusicAspect.aj	85--118	Interpretation of the Code	Assumes that the Save command is indeed handled within handleCommand already and will be invoked also for musical media.	The first part can be checked by inspecting the joinpoint shadows: If there are none, then this assumption is invalid. The second part seems more difficult. It seems again to be an assumption on the overall setup. LTL or similar can be useful here again, but what would the appropriate state predicates be?
l.m.alternative.music.MusicAspect.aj	116	Comment	The comment states "This should be the return value from method handleCommandAction." indicating that the developers didn't quite see how they could express this in their aspect. They could have used two coordinated advices that hand the return value over appropriately based on the assumption that the value is not changed in between. This assumption is what I find interesting about this example.	Note, there is another issue about the duplication of exception handlers here. I believe, they could have avoided that by listing these exceptions as to be thrown by the advice, thus effectively placing an assumption on the weaving context that the exception would be handled there.
l.m.alternative.music.MusicAspect.aj	156--161	Interpretation of the Code	Assumes that no other code adds a command labelled 'Play'.	
l.m.alternative.music.MusicAspect.aj	167--171	Interpretation of the Code	Assumes that no other code introduces a screen type that has the integer value 2.	

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.alternative.music.MusicAspect.aj	181--184	Interpretation of the Code	Assunes that no other code will add a "Type of media" input field.	It would seem a bit odd for this to be in MusicAspect, as it certainly also is relevant for something like video. So, conversely, there is probably an assumption here that Video will never be used without Music?
l.m.alternative.music.MusicNotPhotoNotVideo.aj				No relevant assumptions as far as I can tell.
l.m.alternative.music.optional.CopyAndMusic.aj	21--22	Interpretation of the Code	Not an assumption here, but one for CopyMultiMediaAspect: This pointcut refines the pointcut there, so we should be able to express some more assumptions about the pointcut in CopyMultiMediaAspect already (e.g., the assumption about the meaning of the return value).	
l.m.alternative.music.optional.CopyAndMusic.aj	33--36	Interpretation of the Code	Assumes that no other code will add a "Copy" command.	Strangely enough, this is not inherited from CopyMultiMediaAspect, which assumes this label to be there...
l.m.alternative.musicvideo.MusicOrVideo.aj	17--32	Interpretation of the Code	Assumes that some other code will know of these introductions and use them appropriately.	Such dependencies between aspects will of course be checked by seeing that if some code calls these methods this aspect must be deployed. However, this particular aspect doesn't add anything to the system's behaviour unless its introductions are invoked somewhere else, as it doesn't actually use them in any way anywhere else. So, assuming that developers do not intentionally produce dead code, this makes an assumption on other code. In this case, this seems to assume that the aspect is deployed together with MusicMediaAccessor. An error in the implementation of MusicMediaAccessor seems to be that it doesn't actually use the constants defined in the aspect, but instead adds the strings directly.
l.m.alternative.photo.AbstractPhotoAspect.aj				No relevant assumptions as far as I can tell.
l.m.alternative.photo.PhotoAspect.aj	56--72	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.alternative.photo.PhotoAspect.aj	56--72	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.alternative.photo.PhotoAspect.aj	56--72	Interpretation of the Code	Assumes a command labeled 'View' has been added to the list of available commands.	
l.m.alternative.photo.PhotoAspect.aj	101--106	Interpretation of the Code	Assumes no other code adds a command labelled 'View'.	
l.m.alternative.photo.PhotoAspect.aj	101--106; 112--	Interpretation of the Code	Assumes no other code defines a screen type valued 1.	
l.m.alternative.photo.PhotoNotVideoNotMusic.aj				No relevant assumptions as far as I can tell.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.alternative.photo.exceptionblocks.ScreensAspectEH.aj	16--17	Interpretation of the Code	The fact that this doesn't select calls to loadImage, but rather selects calls to the constructor directly encodes an assumption that loadImage is only ever invoked from within this constructor.	I'm really not quite sure why this is a useful aspect in the first place. It seems to introduce more dependencies than it resolves. Anyway, the only other relevant assumption that I can identify is somewhat reversed to the other ones: PhotoViewScreen.new makes an assumption that this aspect will be deployed and will soften the two exceptions for this particular constructor.
l.m.alternative.photo.optional.CopyAndPhoto.aj	26--27	Interpretation of the Code	Assumes that there are no other constructors in PhotoViewScreen.	
l.m.alternative.photo.optional.CopyAndPhoto.aj	29--31	Interpretation of the Code	Assumes that the constructor doesn't itself add a command named 'Copy'.	
l.m.alternative.photoMusic.PhotoAndMusicAspect.aj	35--36			Why is this aspect not simply derived from AbstractAlternativeFeature?
l.m.alternative.photoMusic.PhotoAndMusicAspect.aj	38--64	Interpretation of the Code	Assumes that MusicSelector and PhotoSelector have also been deployed, but VideoSelector apparently has not.	Otherwise, I would assume this code should also invoke stuff from VideoSelector. Really, of course, the assumption is not on a particular aspect deployment, but on the presence of particular methods.
l.m.alternative.photoMusic.PhotoAndMusicAspect.aj	72--83	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling goToPreviousScreen. Also assumes that other handlers will return true when they have handled goToPreviousScreen.	
l.m.alternative.photoMusic.PhotoAndMusicAspect.aj	72--83	Interpretation of the Code	Assumes setMainMenu has been invoked from somewhere before.	This is satisfied because setMainMenu is invoked from within this aspect.
l.m.alternative.video.AbstractVideoAspect.aj				No relevant assumptions as far as I can tell.
l.m.alternative.video.VideoAspect.aj	28--40			One thing that will need further detailed analysis (or maybe just a discussion with Eduardo) is to understand how the various setup code in advice for startApp is meant to interact and form a meaningful whole. I'm sure there are interesting assumptions in here that might be used to check the correctness of precedence statements.
l.m.alternative.video.VideoAspect.aj	44--48	Interpretation of the Code	Assumes that if this aspect is deployed, AlbumData mediaAccessors will always be VideoMediaAccessors.	Otherwise, there should be more defensive programming that also covers other cases.
l.m.alternative.video.VideoAspect.aj	56--71	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.alternative.video.VideoAspect.aj	56--71	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.alternative.video.VideoAspect.aj	56--71	Interpretation of the Code	Assumes that the 'Play Video' command has been added to the menu, that is that the advise at lines 110--115 and 120--124 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.alternative.video.VideoAspect.aj	110--115	Interpretation of the Code	Assumes that no other code adds a command labelled 'Play Video'.	
l.m.alternative.video.VideoAspect.aj	120--124	Interpretation of the Code	Assumes that no other code introduces a screen type that has the integer value 3.	
l.m.alternative.video.VideoNotPhotoNotMusic.aj	20--23	Interpretation of the Code	Assumes that some instance of AbstractVideoAspect has also been deployed.	Otherwise, it would need to extend that aspect.
l.m.alternative.video.optional.CopyAndVideo.aj	31--34	Interpretation of the Code	Assumes that no other code adds a command labelled 'Copy'.	Implicitly assumes that l.m.alternative.photo.optional.CopyAndPhoto is not deployed.
l.m.aspects.exceptionblocks.ControllerAspectEH.aj	27--34	Interpretation of the Code	Assumes AlbumData.deleteAlbum is only ever invoked from AlbumController.handleCommand.	Note that this could be fixed by changing the pointcut to a general call(AlbumData.deleteAlbum) without the explicitly added withincode. Not sure I fully understand why the exception handling performed needs to add this specific constraint. In the current code base, deleteAlbum is only ever called from that handleCommand method anyway. CLARIFY WITH EDUARDO (above is another similar case)
l.m.aspects.exceptionblocks.DataModelAspectEH.aj	108--116	Interpretation of the Code	Assumes MediaAccessor.loadAlbums is only ever invoked from AlbumData.getAlbumNames.	Similar to above. Here, though it would have made sense to have a negative withincode to exclude the case where loadAlbums is recursively invoked. CLARIFY WITH EDUARDO
l.m.aspects.exceptionblocks.UtilAspectEH.aj	19--21	Interpretation of the Code	Assumes that MediaUtil.readMediaAsByteArray is only used to read in images; that is, that the class is not reused for videos or music.	In fact, though, it is at least also used for music, which may lead to slightly misleading error messages. On the other hand, the pointcut here is a good example of where it does make sense to use withincode for an exception handling aspect: The advise handles exceptions from Class.getResourceAsStream and, thus, does need the context to determine how to correctly interpret them.
l.m.aspects.exceptionblocks.UtilAspectEH.aj	29--37	Interpretation of the Code	Assumes that internalReadMediaAsByteArray is only invoked from readMediaAsByteArray	This is probably true, as it is a private operation, but then why is the withincode clause needed in the first place?
l.m.optional.MusicAndOptionalFeatures.aj				No relevant assumptions as far as I can tell.
l.m.optional.OptionalFeaturesButVideo.aj				No relevant assumptions as far as I can tell.
l.m.optional.OptionalFeatureAspect.aj				No relevant assumptions as far as I can tell.
l.m.optional.SortingAndFavorites.aj				No relevant assumptions as far as I can tell.
l.m.optional.SortingAndFavoritesAndCopy.aj				No relevant assumptions as far as I can tell.
l.m.optional.SortingAndFavoritesAndCopyAndSMS.aj				No relevant assumptions as far as I can tell.
l.m.optional.VideoAndOptionalFeatures.aj				No relevant assumptions as far as I can tell.
l.m.optional.capturephoto.CapturePhotoAspect.aj	23--42	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.optional.capturephoto.CapturePhotoAspect.aj	23--42	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.capturephoto.CapturePhotoAspect.aj	23--42	Interpretation of the Code	Assumes that the 'Capture Photo' command has been added to the menu, that is that the advise at lines 134--139 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.optional.capturephoto.CapturePhotoAspect.aj	59--73	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.optional.capturephoto.CapturePhotoAspect.aj	59--73	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.capturephoto.CapturePhotoAspect.aj	59--73	Interpretation of the Code	Assumes that the 'Take Photo' command has been added to the menu, that is that the advise at lines 84--90 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.optional.capturephoto.CapturePhotoAspect.aj	84--90	Interpretation of the Code	Assumes no other code introduces a command labelled 'Take photo'.	
l.m.optional.capturephoto.CapturePhotoAspect.aj	84--90	Interpretation of the Code	Assumes no other code introduces a screen type with ordinal value 1 for CaptureVideoScreen.	
l.m.optional.capturephoto.CapturePhotoAspect.aj	134--139	Interpretation of the Code	Assumes no other code introduces a command labelled 'Capture photo'.	
l.m.optional.capturevideo.CaptureVideoAspect.aj	26--32	Interpretation of the Code	Assumes no other code introduces a screen type with ordinal value 2 for CaptureVideoScreen.	
l.m.optional.capturevideo.CaptureVideoAspect.aj	74--79	Interpretation of the Code	Assumes no other code introduces a command labelled 'Capture Video'.	
l.m.optional.capturevideo.CaptureVideoAspect.aj	87--101	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.optional.capturevideo.CaptureVideoAspect.aj	87--101	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.capturevideo.CaptureVideoAspect.aj	87--101	Interpretation of the Code	Assumes that the 'Capture Video' command has been added to the menu, that is that the advise at lines 74--79 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.optional.copy.CopyAspect.aj	36--41	Interpretation of the Code	Assumes that PhotoViewController is an appropriate controller for implementing copying and will not be advised by other aspects to change this behaviour.	
l.m.optional.copy.CopyAspect.aj	53--74	Interpretation of the Code	Again, an inverse assumption: Code that invokes this method (which is not invoked from CopyAspect!) assumes CopyAspect has been deployed.	
l.m.optional.copy.CopyMultiMediaAspect.aj	46--48	Interpretation of the Code	Assumes CopyAspect to also be deployed.	Not so exciting here, as the two aspects at least reside in the same package.
l.m.optional.copy.CopyMultiMediaAspect.aj	55-57	Interpretation of the Code	Assumes that mediaName is not changed between invokation of this advice and of the advice on lines 61--116.	Not a problem for a single-threaded phone application, but will become problematic in a multi-threaded environment.
l.m.optional.copy.CopyMultiMediaAspect.aj	61--116	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.
l.m.optional.copy.CopyMultiMediaAspect.aj	61--116	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.copy.CopyMultiMediaAspect.aj	61--116	Interpretation of the Code	Assumes that the 'Copy' and 'Save Item' commands have been added to the menu.	This is an interesting variant of the theme: The two commands are not introduced within this aspect, but otherwise.
l.m.optional.favourites.FavouritesAspect.aj	40--76	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.optional.favourites.FavouritesAspect.aj	40--76	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.favourites.FavouritesAspect.aj	40--76	Interpretation of the Code	Assumes that the 'Set Favorite' and 'View Favorites' commands have been added to the menu; that is that the advice on lines 144--148 hs been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.optional.favourites.FavouritesAspect.aj	40--76	Interpretation of the Code	Assumes (on line 69) that favorite will be maintained appropriately per controller instance.	This works for single-threaded phone applications, but may become problematic with multi-threaded contexts. Actually, there seem to also be a few typos in the ITDs dealing with favorite.
l.m.optional.favourites.FavouritesAspect.aj	144--148	Interpretation of the Code	Assumes no other code introduces commands labelled 'Set Favorite' and 'View Favorites'.	
l.m.optional.favourites.FavouritesAspect.aj	144--148	Interpretation of the Code	Assumes l.m.optional.favourites.PersisteFavoritesAspect.aj has been deployed also and has been used for serialising the media data read.	An interesting assumption as it may span a number of runs of the application.
l.m.optional.favourites.PersisteFavoritesAspect.aj	22--32	Interpretation of the Code	Relies on consistent aspect ordering between aspects serialising and deserialising media data.	Otherwise, fields added by other aspects might be mistaken for fields added by this aspect, as they are only referenced by their relative order.
l.m.optional.sms.SMSAspect.aj	23--25	Interpretation of the Code	SmSOrCapturePhoto assumes this method is there.	
l.m.optional.sms.SMSAspect.aj	27--29	Interpretation of the Code	SMSReceiverController assumes this method is there.	
l.m.optional.sms.SMSAspect.aj	27--29	Interpretation of the Code	Assumes the image will additionally be passed to the constructor of PhotoViewScreen in the usual manner so that it will be displayed.	An alternative would seem to be to only maintain an appropriate flag or to use loadImage to ensure the image was copied/referenced appropriately.
l.m.optional.sms.SMSAspect.aj	37--39	Interpretation of the Code	Assumes no other code introduces a command labelled 'Send Photo by SMS'.	
l.m.optional.sms.SMSAspect.aj	47--49	Interpretation of the Code	SMSReceiverController assumes this method is there.	
l.m.optional.smsorcapturephoto.SmSOrCapturePhoto.aj				No relevant assumptions as far as I can tell.
l.m.optional.smsorcapturephotoorvideo.SmsOrCapturePhotoOrVideo.aj		Interpretation of the Code		Seems to assume that someone will call this. Not sure how it relates to methods such as getImage introduced in SMSAspect
l.m.optional.sorting.SortingAspect.aj	65--83	Interpretation of the Code	Assumes all user interaction is treated by handleCommandAction.	This may be encoded in the architecture, but is certainly not clarified as a dependency for this advice.

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: *MobileMedia* l.m.X stands for lancs.mobilemedia.X
Version: *7*

Adivce data:

File	Lines	Source of Assumption (e.g., comment, interview, mailing list, interpretation of code, etc.)	Assumption Description	Comment
l.m.optional.sorting.SortingAspect.aj	65--83	Interpretation of the Code	Assumes that returning true will stop other potential aspects handling command. Also assumes that other command handlers will return true when they have eaten an command.	This is a bit nitpicky. It seems a pretty standard protocol, but it is currently not documented (only informally in l.m.core.ui.controller.ControllerInterface, but not as an assumption of the aspect). Sometimes this sort of protocol is implemented one way (true for 'yes I'm done'), sometimes the opposite way (true for 'command still up for grabs'), so making this assumption explicit should certainly help with reuse or base-code evolution.
l.m.optional.sorting.SortingAspect.aj	65--83	Interpretation of the Code	Assumes that the 'Sort by View' command has been added to the menu; that is that the advice on lines 171--174 has been run.	This seems an interesting form of cross-dependency. Formally expressing this as an LTL formula requiring that we have passed that other advice before entering this advice doesn't make sense (it would be almost vacously true as there is no other way this advice could ever be entered). Instead, it requires that for at least one object that has been associated with this advice, we have previously registered the play command. NEEDS MORE THOUGHT
l.m.optional.sorting.SortingAspect.aj	65--83	Interpretation of the Code	Assumes (on line 76) that sort will be maintained appropriately per controller instance.	A potential for issues in synchronisation.
l.m.optional.sorting.SortingAspect.aj	171--174	Interpretation of the Code	Assumes no other code introduces a command labelled 'Sort by Views'.	
l.m.optional.sorting.SortingAspect.aj	183--201; 209--215	Interpretation of the Code	Makes an assumption about correct relative ordering of serialising and deserialising aspects.	An interesting assumption as it may span a number of runs of the application. Different from the case above, however, here the code for serialising and deserialising is in the same aspect, which may make ordering quite difficult.