

EXAMPLE – DOMAIN ASSET MANAGER

This document contains an example of how the SMartyPerspective technique works for the Domain Asset Manager (DAM) scenario for defect detection of the diagram with embedded defects of the Mobile Media Software Product Line (SPL) use-case diagram.

1. Mobile Media Software Product Line

Mobile Media (MM) is an SPL composed of applications (products) that manipulate music, videos and photos for mobile devices such as smartphones and tablets. Provides support for managing (creating, deleting, viewing, executing and sending) different types of media (Young, 2005; Geraldi and OliveiraJr, 2017).

Part of the description of the Mobile Media use cases was taken from the work of Choma (2017) and are presented as follow:

UC1: Log in

Category: mandatory

Description: User enters login and password and the system validates the data.

UC2: Send Media

Category: optional

Description: User selects and sends a certain media to another device.

UC3: Manage Album

Category: mandatory

Description: User performs general control of album entries. It encompasses the operations of creating, deleting and listing albums.

UC4: Manage Media

Category: mandatory

Description: User performs general control of media records. It encompasses the operations of creating, deleting and listing records.

UC5: Manage Favourite Media

Category: optional

Description: User performs control of favorite media. It encompasses the operations of setting favorite media and listing favorite media.

UC6: Play Media

Category: mandatory

Description: User execute a media according to the possibilities of the product.

UC7: Add Media to Album

Category: mandatory

Description: User selects a media and links it to a certain album stored in the system.

UC8: Link Media with Address Book Entry

Category: optional

Description: User links media to a particular contact so that it plays when there is a call from that contact.

UC9: View/Hear Media from Incoming Caller

Category: optional

Description: When receiving a call, the system plays the media linked to the contact.

UC10: Label Files

Category: optional

Description: User inserts a label for a certain media, which can be video, photo or music.

2. Use Case Diagram

In Figure 1, the use case diagram for Mobile Media is presented with defects incorporated by Geraldi and OliveiraJr (2017). Use the scenario for your perspective described in Document 5 of this instrumentation to inspect this diagram for defect detection.



Figure 1: Use Case Diagram for SPL Mobile Media - Original Version

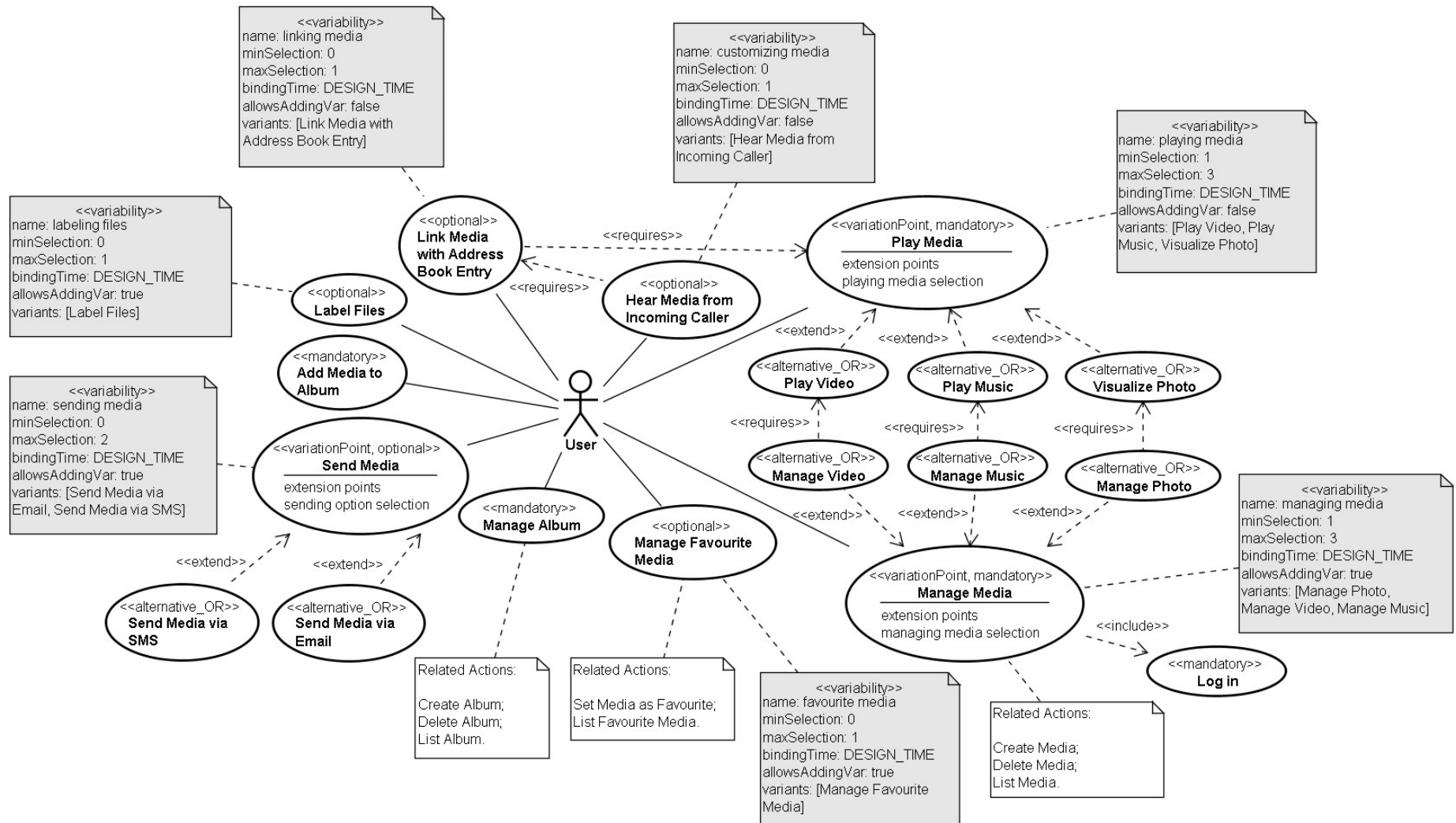
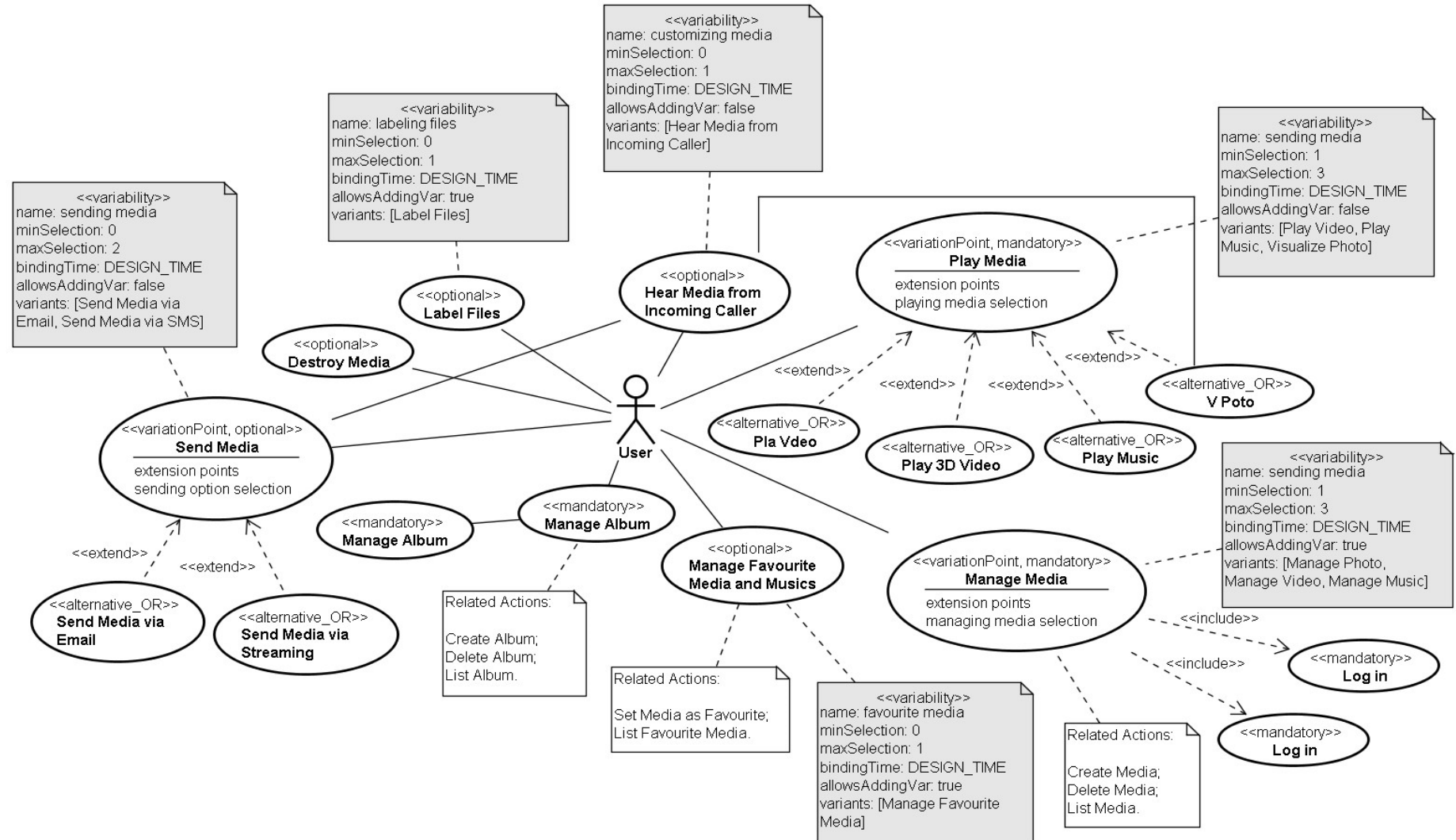




Figure 2: Use Case Diagram for SPL Mobile Media with Embedded Defects - Second Version



3. Resolution for Use case Diagram

The DAM must ensure that the latest version of the diagram is correct in the asset base for it to be used by other perspectives or for other SPL to be derived. Therefore, for this role, there is a need to have two versions of the same diagram: the first version (oracle - original diagram) will serve as a comparison for the DAM to identify defects in the second version of the diagram.

For this example, the use case diagram oracle was considered the correct diagram for the SPL (without defects) in Figure 1. In Figure 2 there is a version of the diagram (second version) with built-in defects that will be used as an example.

The initial DAM scenario technique instruction guides the reader in locating the two diagrams that will be used for inspection (oracle and defective). The diagram with defects will serve as a basis for identifying the defects, as each element defined there will be searched in the oracle diagram to check if they are consistent with each other.

For each element of the diagram, the inspector must read all the questions in the step, and only then proceed to the next element, except when the question directs that the reader can stop the inspection and proceed to the next element, or that they are issues that do not fit the element. For example, a subset of questions for optional elements and the element under inspection is mandatory.

The inspection for the use case diagram by the DAM starts in Step 1. It must be completed for all elements of the diagram (Figure 1), whether it is of the actor or use case type, and only then, proceed with the scenario (Step 2 and 3), as the next step may need information that has already been analyzed previously, such as question 3.1. This question detects omission of elements, by verifying that some elements of the oracle diagram were not found in the diagram with defects

The reader is free to start the inspection for the element of their choice. For this example, the inspection was started from the **LabelFiles** use case, however, after following the entire procedure in Step 1 (reading all the questions in this step), no inconsistencies were found between the diagram and the requirements specification, as , for all questions the answer was negative, that is, no defects were found in the element.

The next element inspected is the **Destroy Media** use case. When the DAM compares this use case of the inspected diagram with the oracle in question 1.1, it will verify that this element cannot be traced between the two versions of the diagram, and since it is not part of any SPL change, a defect. For, there is no functionality compatible with this use case in the oracle diagram.

The defect identified by the perspective will be described in the DIF (Defect Identification Form) as follows (Table 1 - line 1)

- Diagram: use case;
- Question Number: 1.1;
- Element: DestroyMedia;
- Identified Defect: This element cannot be traced back to the original version.

The process will move on to all elements of the use-case diagram with defects and then move on to Step 2. In this step, specific defects related to the UML notation that represent variability are identified.

For this inspection, the example of a defect type characteristic of Step 2 was found for the **Send Media** use case. When analyzing question 2.6 of this step, the reader will be guided to check the meta-attribute **allowAddingVar** (permission to add new variants). However, when comparing the inspected diagram with the oracle it will be verified that the meta-attribute was set to true and in the oracle to false. Therefore, it will be necessary to fill in the defect found in the DIF (Table 1 - line 11):

- Diagram: use case;
- Question Number: 2.6;
- Element: Send Media;
- Identified defect: The value for the allowAddingVar meta-attribute differs from the original version. It is set to false when it should be true.

With all the variability analyzed by Step 2, the inspector can then move on to Step 3, which consists of verifying in the oracle which elements were not identified in the diagram with defects (and were not removed from the SPL through any update described in the version management).

Therefore, it is important that the reader has made a mark on the diagrams they are inspecting, as now those that were not marked on the oracle diagram will be identified as omission defects in the use case diagram.

For this example, the video management use case, **Manage Video**, was not found in the defective diagram, so it is filled out in the DIF form (line 15 - Table 1) as follows:

- Diagram: use case;
- Question Number: 3.2;
- Element: Manage Video
- Identified Defect: The element is not present in the inspected version.

With the inspection completed and the form completed (Table 1), the next step in the DAM scenario is to investigate the **realizes+** and **realizes-** meta-attributes to identify whether it is possible to find the elements defined in these sets in the specified diagram. In the diagram taken as an example for this section, the meta-attributes **realizes+** and **realizes-** were not defined for any element, so the inspection is completed in Step 3.



Table 1: DIF after DAM inspection of the use case diagram

nº	DIAGRAM					QUESTION NUMBER	ELEMENT	IDENTIFIED DEFECT
	FT	UC	CL	CP	SQ			
1		X				1.1	Destroy Media	It is not possible to trace this element to the original version.
2		X				1.5.1	Send Media	There is no relationship between the Send Media and Hear Media from Incoming Caller use cases.
3		X				1.1	Send Media via Streaming	It is not possible to trace this element to the original version.
4		X				1.5.1	Hear Media from Incoming Call	There is no relationship between the HearMedia from Incoming Caller and V Photo use cases.
5		X				1.1	Manage Album	It is not possible to trace this element to the original (duplicate) version.
6		X				1.2	Manage Favourite Media and Musics	The name in the second version does not agree with the first.
7		X				1.1	Pla Vdeo	It is not possible to trace this element to the original version.
8		X				1.2	Play 3D Video	The name in the second version does not agree with the first.
9		X				1.2	V Photo	The name in the second version does not agree with the first.
10		X				1.1	Log In	It is not possible to trace this element to the original (duplicate) version.
11		X				2.6	Send Media	The value of the allowsAddingVar meta-attribute differs from the original version. It is set to false when it should be true.
12		X				2.3	PlayMedia	The name of the variability has already been defined by another variability.
13		X				2.3	ManageMedia	The name of the variability has already been defined by another variability.
14		X				3.1	Link Media with Address Book Entry	The element is not present in the inspected version.
15		X				3.1	Manage Video	The element is not present in the inspected version.
16		X				3.1	ManageMusic	The element is not present in the inspected version.
17		X				3.1	ManagePhoto	The element is not present in the inspected version.

The diagram illustrates a media management system architecture. It features several interfaces and classes, along with their relationships.

Interfaces:

- IAlbumMgr**: Defines methods for album management, including `areThereAlbums()`, `showAlbums()`, `isAlbumUsed(name: String) boolean`, `createAlbum(album: Album) void`, and `deleteAlbum(album: Album) void`.
- IAddMediaAlbum**: Defines the `addMediaAlbum(media: Media, album: Album)` method.
- IAlbumMgt**: Defines methods for album management, including `addAlbum(album: Album)`, `removeAlbum(album: Album)`, `getAlbum(album: Album) Album`, `areThereAlbums()`, `showAlbums()`, `hasAlbumThisMedia(media: Media, album: Album) boolean`, `addMediaAlbum(media: Media, album: Album)`, `isAlbumUsed(name: String) boolean`, and `isAlbumEmpty(album: Album) boolean`.
- IEntryMgt**: Defines methods for entry management, including `getEntry(entry: Entry) Entry`, `areThereEntries() boolean`, `showEntries()`, `hasMediaAssociated(entry: Entry, media: Media) boolean`, and `addMediaEntry(media: Media, entry: Entry)`.
- IUserMgr**: Defines the `getUser(login: String, password: String) void` method.
- IUser**: Defines the `login: String` and `password: String` attributes.
- ILinkMedia**: Defines methods for linking media, including `linkMedia(entry: media: Media, entry: Entry)`, `showEntries()`, `areThereEntries() boolean`, and `isMediaAssociated(media: Media, entry: Entry) boolean`.
- IManageMedia**: Defines methods for managing media, including `createPhoto(photo: Photo)`, `hasMemory(memory: Memory) boolean`, `signifier User() boolean`, `deletePhoto(photo: Photo)`, `isPhotoUsed(photo: Photo)`, `areTherePhotos() boolean`, `showPhotos()`, `deleteMusic(music: Music)`, `createMusic(music: Music)`, `showMusic()`, `areThereMusics() boolean`, `isMusicUsed(music: Music) boolean`, `createVideo(video: Video)`, `deleteVideo(video: Video)`, `areThereVideos() boolean`, and `showVideos()`.
- IPlayMedia**: Defines the `name = "selected media"`, `minSelection = 1`, `maxSelection = 3`, `bindingTime = DESIGN_TIME`, `allowAddingVar = true`, `Variables = (PhotoMgr, MusicMgr, VideoMgr)`, and `realizes = (Managing Media)` properties.
- IHearMedia**: Defines the `name = "selected media"`, `minSelection = 1`, `maxSelection = 3`, `bindingTime = DESIGN_TIME`, `allowAddingVar = true`, `Variables = (PhotoMgr, MusicMgr, VideoMgr)`, and `realizes = (Managing Media)` properties.
- ILabelFiles**: Defines the `name = "selected media"`, `minSelection = 1`, `maxSelection = 3`, `bindingTime = DESIGN_TIME`, `allowAddingVar = true`, `Variables = (PhotoMgr, MusicMgr, VideoMgr)`, and `realizes = (Managing Media)` properties.
- IPhotoMgr**: Defines methods for photo management, including `addPhoto(photo: Photo)`, `removePhoto(photo: Photo) Photo`, `getPhoto(photo: Photo) Photo`, `areTherePhotos() boolean`, and `showPhotos()`.
- IPhotoMgt**: Defines methods for photo management, including `addPhoto(photo: Photo)`, `removePhoto(photo: Photo) Photo`, `getPhoto(photo: Photo) Photo`, `areTherePhotos() boolean`, and `showPhotos()`.
- IVideoMgr**: Defines methods for video management, including `addVideo(video: Video)`, `removeVideo(video: Video) Video`, `getVideo(video: Video) Video`, `areThereVideos() boolean`, and `showVideos()`.
- IVideoMgt**: Defines methods for video management, including `addVideo(video: Video)`, `removeVideo(video: Video) Video`, `getVideo(video: Video) Video`, `areThereVideos() boolean`, and `showVideos()`.
- IMusicMgr**: Defines methods for music management, including `addMusic(music: Music)`, `removeMusic(music: Music) Music`, `getMusic(music: Music) Music`, `areThereMusics() boolean`, `isMusicUsed(music: Music) boolean`, and `showMusic()`.
- IMusicMgt**: Defines methods for music management, including `addMusic(music: Music)`, `removeMusic(music: Music) Music`, `getMusic(music: Music) Music`, `areThereMusics() boolean`, `isMusicUsed(music: Music) boolean`, and `showMusic()`.
- ISenderMgr**: Defines methods for sending media, including `sendMediaSMS(media: Media)`, `getNumber()`, `sendMediaMail(media: Media)`, and `getMail()`.
- ISender**: Defines the `name = "sending media"`, `minSelection = 0`, `maxSelection = 1`, `bindingTime = DESIGN_TIME`, `allowAddingVar = true`, `Variables = (SenderMgr)`, and `realizes = (Sending Media)` properties.
- IFavouriteMgr**: Defines methods for managing favourite media, including `isMediaFavourite(media: Media) boolean`, `getFavourite(media: Media)`, `areThereFavouriteMedia() boolean`, and `showFavouriteMedia()`.
- IFavourite**: Defines the `name = "favourite media"`, `minSelection = 0`, `maxSelection = 1`, `bindingTime = DESIGN_TIME`, `allowAddingVar = true`, `Variables = (FavouriteMgr)`, and `realizes = (Favourite Media)` properties.

Classes:

- Album**: Implements `IAlbumMgr`.
- AlbumMgr**: Implements `IAlbumMgr`.
- Entry**: Implements `IEntryMgt`.
- EntryMgr**: Implements `IEntryMgt`.
- User**: Implements `IUser`.
- UserMgr**: Implements `IUserMgr`.
- Media**: Implements `IManageMedia`.
- MediaMgr**: Implements `IManageMedia`.
- MediaCtrl**: Implements `IManageMedia`.
- Photo**: Implements `IPhotoMgr`.
- PhotoMgr**: Implements `IPhotoMgr`.
- PhotoMgt**: Implements `IPhotoMgt`.
- Video**: Implements `IVideoMgr`.
- VideoMgr**: Implements `IVideoMgr`.
- VideoMgt**: Implements `IVideoMgt`.
- Music**: Implements `IMusicMgr`.
- MusicMgr**: Implements `IMusicMgr`.
- MusicMgt**: Implements `IMusicMgt`.
- SenderMgr**: Implements `ISenderMgr`.
- Sender**: Implements `ISender`.
- FavouriteMgr**: Implements `IFavouriteMgr`.
- Favourite**: Implements `IFavourite`.

Relationships:

- AlbumMgr** and **EntryMgr** implement `IAlbumMgr` and `IEntryMgt` respectively.
- UserMgr** implements `IUserMgr`.
- MediaMgr** and **MediaCtrl** implement `IManageMedia`.
- PhotoMgr** and **PhotoMgt** implement `IPhotoMgr` and `IPhotoMgt` respectively.
- VideoMgr** and **VideoMgt** implement `IVideoMgr` and `IVideoMgt` respectively.
- MusicMgr** and **MusicMgt** implement `IMusicMgr` and `IMusicMgt` respectively.
- SenderMgr** and **Sender** implement `ISenderMgr` and `ISender` respectively.
- FavouriteMgr** and **Favourite** implement `IFavouriteMgr` and `IFavourite` respectively.
- AlbumMgr** and **EntryMgr** implement `IAlbumMgr` and `IEntryMgt` respectively.
- UserMgr** implements `IUserMgr`.
- MediaMgr** and **MediaCtrl** implement `IManageMedia`.
- PhotoMgr** and **PhotoMgt** implement `IPhotoMgr` and `IPhotoMgt` respectively.
- VideoMgr** and **VideoMgt** implement `IVideoMgr` and `IVideoMgt` respectively.
- MusicMgr** and **MusicMgt** implement `IMusicMgr` and `IMusicMgt` respectively.
- SenderMgr** and **Sender** implement `ISenderMgr` and `ISender` respectively.
- FavouriteMgr** and **Favourite** implement `IFavouriteMgr` and `IFavourite` respectively.

4. Traceability between class and use case diagrams

In Step 4 of the scenario from the DAM perspective, the traceability between elements of the diagrams is analyzed. To this end, the sets of **realizes-** and **realizes+** meta-attributes that define the variability collection of lower and higher level models, respectively, that perform the variability are verified.

Before starting the analysis, check which diagrams are traceable. For this example, it has been defined for the class diagram that the set **realizes+** contains elements traceable to the use case diagram and **realizes-** to the component diagram.

For this example we considered the SPL Mobile Media class diagram in Figure 3 and the defective use case diagram in Figure 2. Therefore, the DAM will inspect the class diagram for traceability against the latest version of the use case diagram.

In Step 4 the DAM will check all UML notes that represent the variability in the element associated with it, through the <<variability>> stereotype, to analyze the **realizes+** and **realizes-** meta-attributes as per the scenario instructions. For the example, the class diagram has 3 variability notations: select media, sending media and favorite media.

For the example, we start the inspection by the select media variability of the **Media Mgr** class. This variability is realized in the use case diagram by managing media variability, as reported in the select media **realizes+** set. Question 4.1 directs the reader to look for the variability of the set **realizes+** in the corresponding diagram. But, in the use-case diagram in Figure 2, there is no variability with this name, so it is not possible to track between diagrams. This defect must then be informed in the diagram (Table 2):

- Diagram: class;
- Question Number: 4.1;
- Element: Media Mgr;
- Identified Defect: The variability defined in realizes+ in the class diagram was not found in the use case diagram.

Following the inspection for the managing media variability, in question 4.3 the DAM is instructed to analyze if any element was missing in the realizes+ and realizes- sets. As three of the variabilities were named equally, the traceable variability of the Media Mgr class, managing media, was not found in the use case diagram. By analyzing the two diagrams, the inspector will verify that the variability associated with the Manage Media use case is not traceable to the variability of the Media Mgr class. The DIF (Table 2) will be completed as follows:

- Diagram: class;
- Question Number: 4.3;
- Element: Media Mgr;
- Identified Defect: The use case diagram variability related to the Manage Media use case, named for now as “sending media” was missing from the realizes+ set.

The **favourite media** variability for media management was found in the use case diagram and it realizes the favourite media variability of the **FavouriteMgr** class as specified in the class diagram.

If the inspector had already performed the inspection for the use case diagram, as in the example in the previous Section in Table 1, the inspector would have verified that there is redundancy between the variability names, changing them as necessary. Thus, he would have avoided traceability defects between elements.

Considering that DAM's focus is on traceability between elements and started the inspection in Step 4, it will verify that for the associated variability of the optional class **Sender Mgr**, the set **realizes+** contains the sending media variability. The inspector will find in the use case diagram in Figure 2 three variabilities with this name. However, two of them are associated with the **Manage Media** and **Play Media** use cases. Therefore, following the SPL context, these elements are not traceable and there is a defect in the diagram that must be identified in the DIF form (Table 2) for each of them:

- Diagram: class;
- Question Number: 4.2;
- Element: Manage Media;
- Identified Defect: The variability associated with the Manage Media use case does not realize the variability of the sending media class diagram.

After inspection of the three class diagram variabilities, inspection is completed for this example. The final DIF for Step 4 can be analyzed in Table 2.

Table 2: DIF após inspeção do diagrama de caso de uso pelo DAM no Passo 4

nº	DIAGRAM					QUESTION NUMBER	ELEMENT	IDENTIFIED DEFECT
	FT	UC	CL	CP	SQ			
1			X			4.1	Media Mgr	The variability defined in realizes+ in the class diagram was not found in the use case diagram.
2			X			4.3	Media Mgr	The use case diagram's variability related to the Manage Media use case, named for now as "sending media" was missing from the realizes+ set.
3			X			4.2	Manage Media	The variability associated with the Manage Media use case does not accomplish the variability of the sending media class diagram.
4			X			4.2	Play Media	The variability associated with the Play Media use case does not realize the variability of the sending media class diagram.