DSPECIALISTS
Digitale Audio- und Messsysteme GmbH

Handbuch

EA2Latex

Version: working
6. April 2016

generated from EA Model:
DSPE_EA2Latex\documentation\EA2Latex.eap

DSPE_EA2Latex/documentation/EA2Latex_Handbuch/EA2Latex_Handbuch.pdf

DSPECIALISTS
Digitale Audio- und Messsysteme GmbH
Helmholtzstr. 2-9 L 10587 Berlin

http://www.dspecialists.de

Inhaltsverzeichnis

Version	Release Date	Author	Note
1.0	11.02.2013	DR	Version 1.0 erstellt
1.4	18.06.2013	DR	Version 1.4 erstellt. Um eine Reihe Features ergänzt. Mit Plugin Version 1.4.9 erstellt, mit funktionierenden (externen) Querverweisen
1.4.12	03.03.2014	MG	Abschnitt zu Tabellen hinzugefügt. Erstellung eines Referenzen-Packages überarbeitet. Stereotype und Tagged Values hinzugefügt bzw. ergänzt
1.4.13	24.03.2014	MG	Ergänzt um Abschnitte zur Einbindung von Tabellen / Abbildungen aus externen Dateien (CSV, PNG, JPG, PDF, EPS) und linked Documents.
1.4.13b	03.03.2016	KL	Abschnitte zu Abbildungen und Tabellen klarer strukturiert, Querverweise ergänzt. Architekturbeschreibung in eigenen Abschnitt ausgegliedert. Lizenzverweise hinzugefügt.
1.4.13c	07.04.2016	KL	Bedienhinweise als UseCase neu definert, korrigiert, vervollständigt und die Struktur verbessert.

1 Einleitung

Das EA2Latex Addin dient der Generierung eines Latex-Dokuments aus einem EA (Enterprise Architect) Model und der angeschlossenen Erstellung eines PDF-Dokuments aus den Latex-Quellen. In diesem Handbuch wird beschrieben, wie das Addin zu installieren und zu benutzen ist.

Zunächst wird in einer kurzen Einleitung ein Überblick über die **Architektur**[1] gegeben. Danach werden **Installationshinweise**[2] gegeben und im Anschluss die **Bedienung**[3] beschrieben. Im **Anhang**[4] finden sich spezifische Angaben zur Benutzung von *EA,TaggedValues* zur Steuerung der Ausgabe und zu den EA2Latex-spezifischen Latex-Kommandos.

Das Addin wird lokal auf dem PC installiert, auf dem auch EA installiert ist. Die Latex-Distribution und die Buildumgebung für die Erstellung des PDF-Dokuments aus den Latex-Quellen liegt hingegen auf dem **Build-Server**[5]. Für die Kommunikation mit dem

[1] Siehe Abschnitt 3 Architektur, Seite 6
[2] Siehe Abschnitt 5 Installationshinweise, Seite 9
[3] Siehe Abschnitt 6 Bedienungshinweise, Seite 16
[4] Siehe Abschnitt 7 Anhang, Seite 34
[5] Siehe Abschnitt 5.3 Installation des Buildservers, Seite 10

Server benutzt das Addin PuTTY. Für die Betrachtung des PDF-Dokuments ist ein PDF-Viewer nötig. In der **Konfigurationsdatei**[6] können Systempfade eingestellt werden.

2 Version

Handbuch für EA2latex Addin 1.4.13.0

3 Architektur

Dieser Abschnitt soll einen Überblick über die bei der Dokumentengenerierung beteiligten Komponenten geben.

Der Architektur liegt ein zweistufiger Build-Prozess zugrunde:

1. Das EA2Latex-Addin generiert auf dem Client-PC aus dem EA-Modell den Inhalt eines Latex-Dokuments und kopiert dieses in eine Build-Umgebung.
2. Ein Build-Server generiert aus dem Latex-Dokument basierend auf zentral definierten Templates das finale PDF-Dokument.

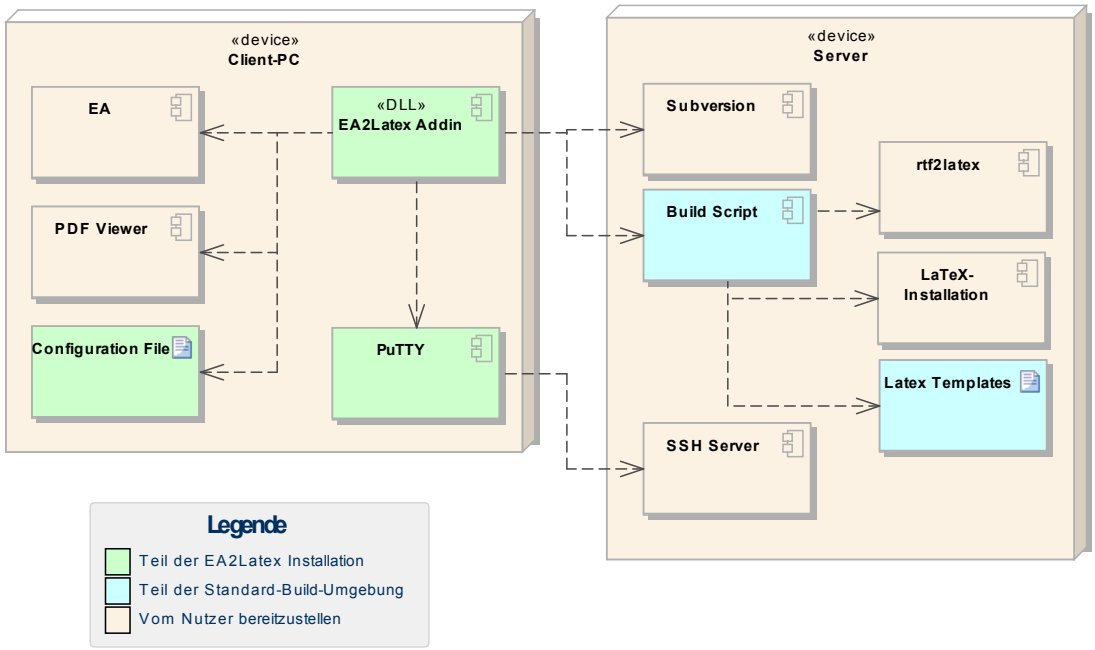


Abbildung 1: Übersicht System Architektur

3.1 Client-PC (Device)

Der Client ist der Arbeitsplatz-PC, auf dem Enterprise-Architect ausgeführt wird.

[6] Siehe Abschnitt 5.5 Konfigurationsdatei, Seite 14

Configuration File

Das EA2Latex-Addin liest Client-spezifische Einstellungen aus der Konfigurationsdatei, u.a. Details zum Build-Server.

EA2Latex Addin (Component)

Das EA2Latex Addin realisiert die Generierung des LaTeX Codes aus dem EA-Modell.

PDF Viewer (Component)

EA2Latex kann einen externen PDF Viewer für die Anzeige des fertigen PDFs benutzen.

PuTTY (Component)

EA2Latex verwendet das externe Tool "PuTTY" bzw. dessen Programme "plink" und "psftp" für den Zugriff auf den Server.

plink (Component)

Mit **plink** wird eine SSH Verbindung zum Server aufgebaut werden, z.B. um das Build-Kommando abzusetzen.

psftp (Component)

Mit **psftp** werden via SFTP die vom Addin generierten Latex- und Bilddateien auf den Server kopiert.

3.2 Server (Device)

Der Server ist üblicherweise ein zentraler Linux-Server, auf dem die endgültige Dokumenten-Generierung ausgeführt wird, kann aber z.B. auch eine Virtuelle Maschine auf dem Client-PC sein.

Die Nutzung eines dedizierten Servers soll die vergleichsweise komplexe LaTeX-Installation von den Clients fernhalten und unternehmensweit identische Build-Regeln für Dokumente (Templates, etc.) garantierten.

Build Script (Component)

Das Build-Script wird vom EA2Latex-Addin nach dem Generieren der Latex-Dateien aufgerufen. Es startet die Dokumentengenerierung basierend auf den hinterlegten LateX-Templates.

Latex Templates

Die Latex-Templates setzen die im generierten Latex Code enthaltenen Style-Kommandos um. Sie bestimmen somit das Look & Feel der Dokumente. Die Wahl der verwendeten Templates erfolgt durch das Build-Script, welches je Dokument konfiguriert werden kann (optional).

LaTeX-Installation (Component)

Auf dem Buildserver muss eine LaTeX-Installation existieren, welche vom Build-Script verwendet werden kann, um aus den vom EA2Latex-Addin generierten TeX-Dateien das finale Dokument zu erzeugen.

Subversion (Component)

Das EA2Latex-Addin kann per Subversion auf Nutzerwunsch verschiedene Dokumentenstände zu revisionieren.

SSH Server (Component)

Der Secure Shell Server ermöglicht den Remote-Zugriff auf die weiteren Server-Tools.

pdfcrop (Component)

Ermöglicht es den überschüssigen, weißen Rand von PDF-Dateien zu entfernen.

pdflatex (Component)

Konvertiert die .tex-Dateien in eine .pdf-Datei.

rtf2latex (Component)

EA2Latex verwendet das open source Tool rtf2latex2e, um rtf-Texte in tex-Dateien zu konvertieren.

4 Lizenzen

Das EA2Latex-Addin wird von DSPECIALISTS unter der GNU Lesser General Public License (LGPL) Version 3 vertrieben.

Die mit dem EA2Latex-Installer ausgelieferten PuTTY-Komponenten plink und psftp stehen unter der MIT-Lizenz.

Für alle Komponenten gilt: Die Software wird ohne Garantie oder Gewährleistung für einen bestimmten Sinn oder Zweck frei zur Verfügung gestellt.

5 Installationshinweise

5.1 Systemanforderungen

Hinweise zu den Systemanforderungen:

- Das Addin wurde bisher nur unter Windows 7 und Windows XP benutzt.
- Adobe Reader 10
- Enterprise Architect 9-12

5.2 Installation des EA2Latex Addins

Im Folgenden wird der Installationsablauf des EA2Latex Addins für die SparxSystems Enterprise Architect Software beschrieben.

1. Ausführen der Datei EA2Latex_Setup.msi
2. Den Haken für das Bestätigen der Lizenzvereinbarung setzen und auf den Install-Button klicken
3. Installationspfad wählen
4. Gegebenenfalls die Windows-Warnung (Änderungen am Computer durch das Programm zulassen) mit "Ja" bestätigen
5. Finish-Button klicken
6. Nun kann der EnterpriseArchitect gestartet werden

Es wird eine Standardkonfiguration im EALatex Ordner der Anwendungsdaten angelegt ("C:/User/<Benutzername>/AppData/Local/EA2Latex"). Außerdem wird der Ordner "graphics" angelegt, in welchem die vom Plugin aus EA exportierten Diagramme liegen.

5.3 Installation des Buildservers

Der Build-Server stellt die Build-Umgebung für die LaTeX-Dokumente bereit. So lange alle Dokumente die gleiche Stilvorgaben verwenden, genügt eine Build-Umgebung für alle Dokumente im Unternehmen. Die Installation des Build-Servers muss dann nur einmal erfolgen.

Wie der Build-Server die Dokumente erzeugt, ist dem EA2Latex Addin prinzipiell egal. Es benötigt nur einen Server, auf dem es via SFTP Dateien ablegen und per SSH das Build-Kommando anstoßen kann. Verzeichnis- und Scriptnamen sind in der **Addin-Konfiguration**[7] einstellbar.

Im EA2Latex Source Repository befindet sich das Standard-Build-Skript "default_build.sh", welches die in der Abbildung 2 ›Installation des Buildservers‹ beschriebene Ordnerstruktur erwartet.

5.3.1 LaTeX-Installation

Für die LaTeX Installation sollten (auf einem Linux-System) folgende Pakete installiert werden:

- texlive
- texlive-lang-german (optional)
- rtf2latex2e 2.2.1
- datatool Package Version 2.2

5.3.2 Subversion-Installation

Wenn die **Revisionerung**[8] der Dokumnte via Subversion (SVN) gewünscht ist, muss auf dem Build-Server Subversion installiert sein.

5.3.3 Standard-Build-Umgebung bereitstellen

In der Default-Ordnerstruktur werden die Verzeichnisse "ea2latex_docs" und "ea2latex_bin" im Nutzerverzeichnis ealatex-Nutzers auf dem Build-Server angelegt. Ist ein anderer Nutzeraccount gewünscht, muss dieser auf allen Clients in die Addin-Konfigurationsdatei eingetragen werden.

[7] Siehe Abschnitt 5.5 Konfigurationsdatei, Seite 14
[8] Siehe Abschnitt 6.5.1 » Revisionsverwaltung, Seite 33

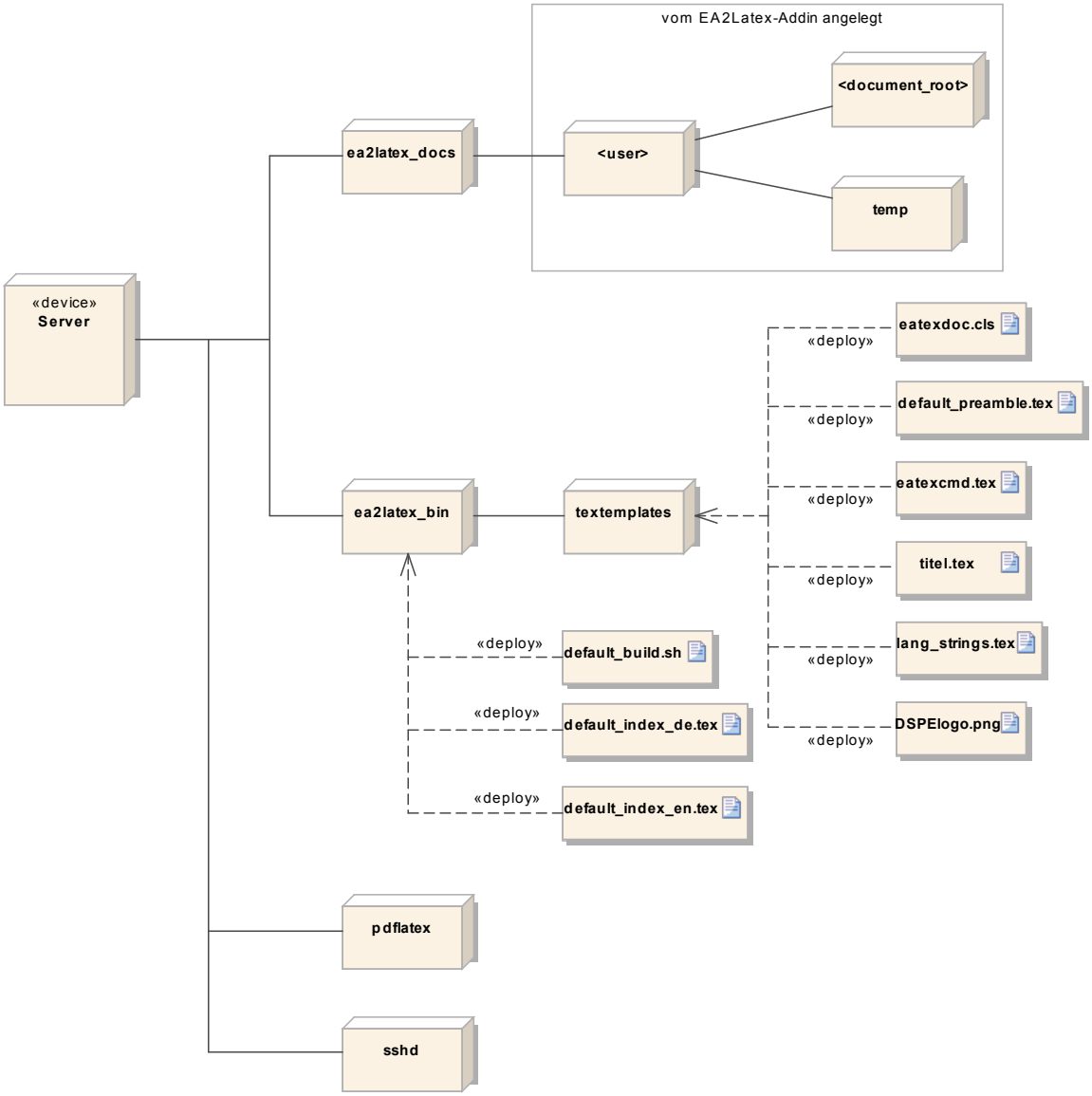


Abbildung 2: Installation des Buildservers

ea2latex_bin (Node)

Dieses Verzeichnis enthält die tex-Templates und das build-Skript. Die Inhalte dieses Verzeichnisses werden im Rahmen der Dokumentengenerierung nicht modifiziert.

Der Inhalt des ea2latex_bin Verzeichnisses kann aus dem SVN-Repository-Pfad "Templates_tex/Default_Spec" ausgecheckt werden. Es muss sichergestellt werden, dass der ealatex-Nutzer Leserechte für alle Dateien und Ausführrechte für das default_build.sh Skript besitzt. Dies sollte gewährleistet sein, wenn die Dateien mit dem ealatex-Nutzer ausgecheckt werden.

default_build.sh

Build Skript

default_preamble.tex

Default Preamble für alle vom Plugin generierten Dokumente.

DSPElogo.png

DSPECIALISTS-Firmenlogo für Titelseite

eatexcmd.tex

EA2Latex-spezifische Latex-Kommandos.

eatexdoc.cls

Klassendefinition.

lang_strings.tex

Sprachabhängige Strings, wie z.B. in Tabellen oder Abbildungen benutzt

titel.tex

Titelseite

ea2latex_docs (Node)

Im ea2latex_docs-Verzeichnis werden die erstellten Dokumente abgelegt.

Die Verzeichnisse "<user>", "<document_root>" und "temp" werden vom EA2Latex-Addin angelegt. Mit der Unterteilung in Benutzer-Verzeichnisse wird sichergestellt, dass sich unterschiedliche Benutzer nicht in die Quere kommen, wenn sie Dokumente generieren. Im "<document_root>"-Verzeichnis landet die Dokumentenstruktur, so wie sie im EA-Model **definiert wurde.**[9] Der Verzeichnisname ist standardmäßig der Name des Packages im

[9] Siehe Abschnitt 6.1.5 » Dokumentenbaum definieren, Seite 18

welchem die *Model Documents* liegen, kann aber über das TaggedValue *ea2tex_document_root* einen definierten Namen erhalten.

Im "temp"-Verzeichnis landen die tex-Sourcen und das PDF-Dokument bei der Ausgabe eines beliebigen Packages.

<user> (Node)

Benutzerverzeichnis. wird bei der Dokumentgenerierung erzeugt.

<document_root> (Node)

Verzeichnis in welchem der Dokumentenbaum, so wie er im EA definiert wurde, angelegt wird.

temp (Node)

Verzeichnis in dem die Dokumente landen, die direkt auf Packages ausgeführt werden.

5.3.4 SSH-Zugang

Damit EA2Latex mit dem Buildserver per SSH und SFTP kommunizieren kann, muss folgendes beachtet werden:

- Auf dem Buildserver muss ein ein User Account angelegt werden (Default: "ealatex")
- Der User muss Schreibzugriff auf das **ea2latex_docs**[10] Verzeichnis haben.
- Der User muss Lesezugriff auf das **ea2latex_bin**[11]-Verzeichnis haben

Hinweis: Das EA2Latex-Addin bietet derzeit noch kein Frontend für die Prüfung/Annahme des Server Keys. Es ist deshalb notwendig, mit JEDEM Client-PC einmalig eine SSH-Verbindung mit PuTTY zu dem Build-Server aufzubauen, um darüber den Server-Key zu akzeptieren. werden, um den Server-Key zu akzeptieren.

5.3.5 Samba-Konfiguration

Für das Öffnen der generierten Dokumente auf dem Build-Server muss ein Zugriff per Dateifreigabe möglich sein.

[10] Siehe Abschnitt 5.3.3 » ea2latex_docs, Seite 12
[11] Siehe Abschnitt 5.3.3 » ea2latex_bin, Seite 11

Auf dem Build-Server muss hierfür ein Samba-Server eingerichtet werden, über welchen die Home-Verzeichnisse freigegeben werden, siehe zum Beispiel unten stehendes Beispiel.

Hinweis: EA2Latex ist derzeit nicht in der Lage, die Netzwerkfreigabe vom Client-PC anzulegen. Es versucht einfach den direkten Zugriff auf die Freigabe. Damit das klappt, sollte eine der folgenden Varianten sicher gestellt sein:

- Der Windows-Nutzer des Client-PCs hat ein Samba-Login auf dem Build-Server. In diesem Fall wird die Netzwerkfreigabe von Windows automatisch geöffnet. Dies ist der empfohlene Weg.
- Der Windows-Nutzer hat die Netzwerkfreigabe zuvor manuell geöffnet und sich mit einem gültigen Build-Server-Nutzer authentifiziert.

```
[home]
path = /home
comment = /home directory
browsable = yes
read only = no
```

Abbildung 3: Beispiel Ausschnitt aus smb.conf

5.4 Deinstallation des EA2Latex Addins

Im Folgenden wird der die Deinstallation des EA2Latex Addins beschrieben.

Das EA2Latex Addin kann über "Systemsteuerung –> Programme deinstallieren" wieder entfernt werden.

5.5 Konfigurationsdatei

Die Konfigurationsdatei definiert die Build-Umgebung für das EA-Addin. Hierzu gehören die benötigten Pfade auf dem Server.

In der Konfigurationsdatei stehen folgende Einträge:

- *server*: Der Name oder IP des Servers auf den die Dateien kopiert werden sollen
- *target_path*: Zielverzeichnis der Dateien auf dem Server
- *build_cmd*: Der Kommandozeilenaufruf für das build-Skript, das die PDF-Erstellung aus den tex-Dateien kapselt

Nach der Addin-Installation kann die Standardkonfiguration angepasst werden. Darüber hinaus ist es auch möglich weitere Umgebungen zu definieren.

5.5.1 Ändern der bestehenden Konfigurationsdatei

Bei der Installation wird die Default-Konfigurationsdatei "ea2latex_default.conf" mit ausgeliefert. Im Normalfall muss diese nur in ea2latex.conf umbenannt werden.

Es ist möglich, dass die voreingestellten Pfade ungültig sind. Das EA2Latex Addin weist durch entsprechende **Fehlermeldungen**[12] darauf hin. In diesem Fall kann über den Button "View Config" der Ordner zu der Konfigurationsdatei geöffnet werden und diese mit einem Editor angepasst werden.

5.5.2 Anlegen einer neuen Konfigurationsdatei

Die neue Konfigurationsdatei muss analog zu der Default-Konfigurationsdatei aufgebaut sein. Dem EA2Latex Addin – bzw. dem "Model Document", das diese Datei verwenden soll – muss nun der Name der neuen Konfigurationsdatei mitgeteilt werden. Dazu wird für das "Model Document" ein neues tagged value mit dem Namen **ea2tex_config_name**[13] angelegt. Im Anwendungsverzeichnis (AppData) wird die Datei <Dateiname>.conf angelegt und entsprechend gefüllt.

Damit wird sichergestellt, dass die Verbindung zum Server erstellt werden kann, auf dem die Ordnerstruktur ausgecheckt wurde; siehe auch den **Abschnitt zur Build-Server-Installation.**[14]

[12] Siehe Abschnitt 7.4 Fehlermeldungen, Seite 44
[13] Siehe Abschnitt 7.2 » Model Document Tagged Values, Seite 38
[14] Siehe Abschnitt 5.3 Installation des Buildservers, Seite 10

6 Bedienungshinweise

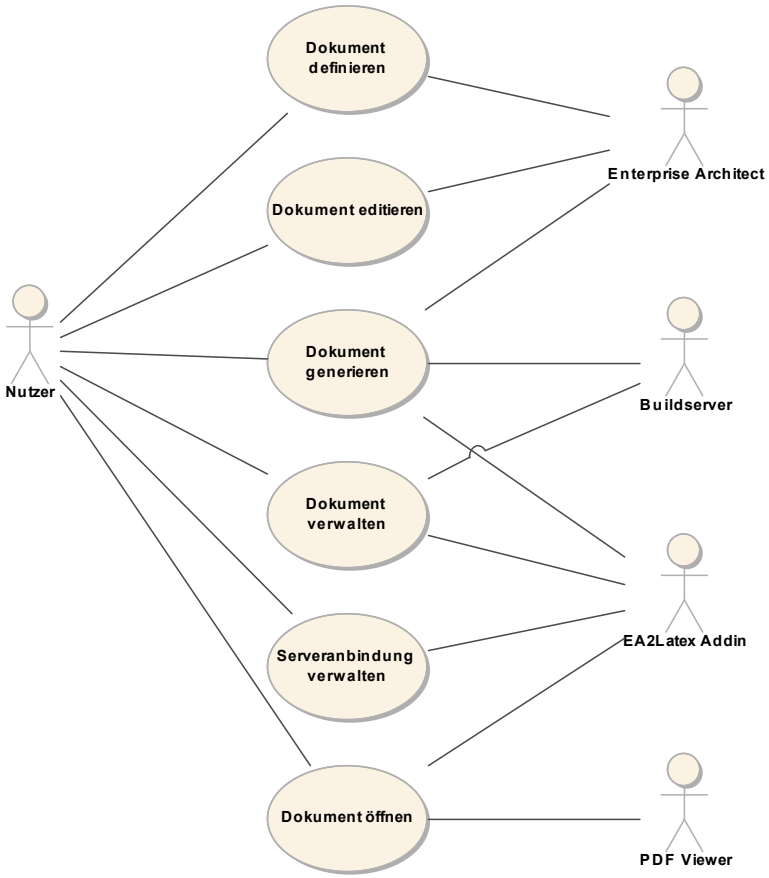


Abbildung 4: Dokumentenerzeugung mit EA2Latex

6.1 Dokument definieren (UseCase)

Es ist möglich, per EA2Latex direkt und ohne weitere Konfiguration **Package-Inhalte auszugeben**[15]. Hierbei wird jedoch der Dokument-Titel direkt aus dem Package-Namen abgeleitet und es sind auch sonst keine weiteren Einstellungen zum Dokument möglich. Dieser Weg wird deshalb vor allem für das schnelle, testweise generieren von **Teilabschnitten eines Dokuments**[16] eingeschlagen.

Die vollständige Definition eines Dokuments wird über das EA-Element *Model Document* ermöglicht.

Verschiedene Spezifikationsdokumente werden über *Model Document* Elemente definiert und üblicherweise innerhalb eines **Dokumentenbaums organisiert**[17].

[15] Siehe Abschnitt 6.3.2 » Dokument aus Package generieren, Seite 31
[16] Siehe Abschnitt 6.2.1.1 » Abschnitte aus Packages erstellen, Seite 21
[17] Siehe Abschnitt 6.1.5 » Dokumentenbaum definieren, Seite 18

Dokumenteigenschaften können über das *Model Document* (genaugenommen über die **TaggedValues des Model Documents**[18]) festgelegt werden.

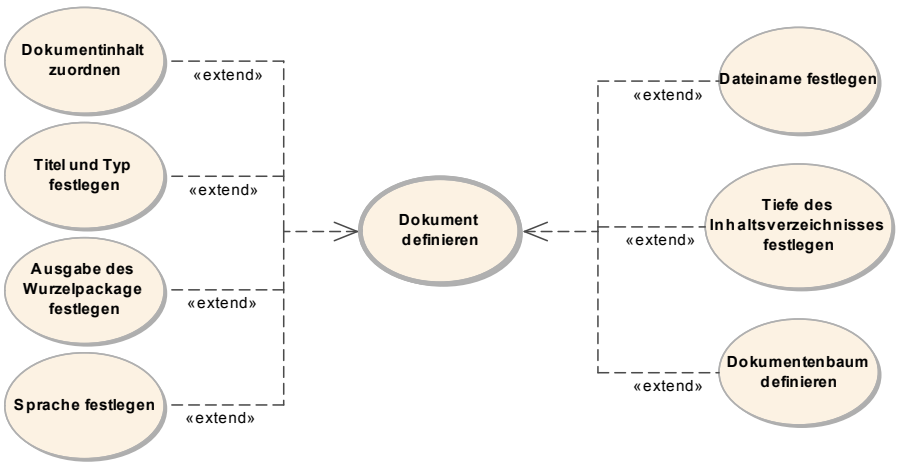


Abbildung 5: Aspekte der Dokumentendefinition

6.1.1 Dokumentinhalt zuordnen (UseCase)

Der Inhalt wird über Zuordnung eines Packages zum Model Document zugewiesen. Dies erreicht man am einfachsten, in dem man ein Model Document Element in einem Diagramm platziert und anschließend das Package, welches den Inhalt liefern soll, aus dem Projekt Browser hinein zieht. Das Package ist dann als Attribut des Model Documents angelegt.

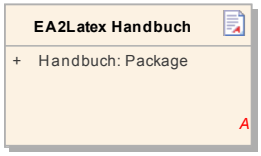


Abbildung 6: Zuordnung Package zu Model Document

6.1.2 Titel und Typ festlegen (UseCase)

Das EA2Latex-Addin unterscheidet zwischen Titel und Typ eines Dokuments.

Der Typ soll die Art des Dokuments beschreiben. Beispiele für Typ sind: "Anforderungsspezifikation" oder "Modulspezifikation". Zu einem Produkt gibt es in der Regel verschiedene solcher Spezifikationsdokumente, die über den Typ unterschieden werden.

Als Titel wird in der Regel der Projekt-, Produktname oder Modulname eingesetzt – je Eingrenzung des Inhalts.

[18] Siehe Abschnitt 7.2 » Model Document Tagged Values, Seite 38

Die Festlegung von Titel und Typ erfolgt über die **Tagged Values**[19] **ea2tex_document_type**[20] und **ea2tex_document_title**[21] des Model Documents.

6.1.3 Ausgabe des Wurzelpackage festlegen (UseCase)

Beim Generieren eines Entwurfs aus einem Packages werden standardmäßig das Package sowie alle Unterpackages ausgegeben. Beim Generieren eines Dokuments aus einem Model Document wird das Wurzelpackage standardmäßig nicht ausgegeben, nur alle Unterpackages.

Über das TaggedValue **ea2tex_skip_root_package**[22] des Wurzel-Packages lässt sich dieses Verhalten explizit steuern.

6.1.4 Sprache festlegen (UseCase)

Einige Textteile werden automatisch generiert (z.B. bei Referenzen: "siehe Abschnitt XY"; oder die Beschriftung von Tabellen und Abbildungen). Das Addin unterstützt dabei derzeit die Sprachen Englisch und Deutsch.

Die Dokumentsprache kann über das TaggedValue **ea2tex_document_lang**[23] des Model Documents festgelegt werden. Wird keine Sprache angegeben, greift die Grundeinstellung der **Server-Buildumgebung**[24]; in der Grundkonfiguration ist das "Deutsch".

6.1.5 Dokumentenbaum definieren (UseCase)

Der Dokumentenbaum bildet das Dokumentenverzeichnis innerhalb des Models ab. Um einen Dokumentenbaum zu definieren legt man (i.d.R auf Model-Ebene) ein Package an. Diesem Package gibt man folgende TaggedValues:

- **ea2tex_document_root**[25]: das Wurzelverzeichnis des Dokumentenbaums im Projektverzeichnis (z.B. für das EA2Latex-Addin: DSPE_EA2Latex/documentation).
- **ea2tex_svn_root**[26]: das Wurzelverzeichnis des Dokumentenbaums im SVN (z.B. für das Plugin: http://svn-server/svn/EA2Latex/documentation)

[19] Siehe Abschnitt 7.2 » Model Document Tagged Values, Seite 38
[20] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_document_type (Attribut), Seite 38
[21] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_document_title (Attribut), Seite 38
[22] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_skip_root_package (Attribut), Seite 39
[23] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_document_lang (Attribut), Seite 38
[24] Siehe Abschnitt 5.3 Installation des Buildservers, Seite 10
[25] Siehe Tabelle 6 Package Tagged Values » ea2tex_document_root (Attribut), Seite 39
[26] Siehe Tabelle 6 Package Tagged Values » ea2tex_svn_root (Attribut), Seite 40

Hinweis: Der Package-Name ist frei wählbar, sollte aber sinnvollerweise "Dokumentverzeichnis", "Document Directory" o.ä. heißen.

Die generierten Dokumente werden entsprechend der im Dokumentenbaum-Package definierten Hierarchie **auf dem Buildserver**[27] generiert.

6.1.6 Dateiname festlegen (UseCase)

Der Dateiname wird standardmäßig automatisch aus Dokumenttitel und Dokumenttyp generiert, es kann aber optional ein individueller Dateiname über das TaggedValue **ea2tex_doc_filename**[28] angegeben werden.

Der Dateiname sollte nicht mehr als 255 Zeichen besitzen und keine reservierten Wörter enthalten, da die Generierung sonst verweigert wird. Sonderzeichen und Umlaute werden durch Unterstriche bzw. durch den Vokal + "e" ersetzt.

6.1.7 Tiefe des Inhaltsverzeichnisses festlegen (UseCase)

Die Tiefe des Inhaltsverzeichnisses ist standardmäßig 3. D.h. es werden im Inhaltsverzeichnis nur die Abschnitte bis zur 3. Unterkapitelebene angezeigt (z.B. 1.3.4 oder 2.5.1). Die Tiefe lässt sich, wenn gewünscht, über das TaggedValue **ea2tex_toc_depth**[29] anpassen.

6.2 Dokument editieren (UseCase)

Die Inhalte eines Dokuments befinden sich unterhalb eines frei definierbaren Packages innerhalb des Models. Die Zuordnung diverser von Dokumenteigenschaften erfolgt über die separate **Dokumentendefinition**[30].

Um sinnvolle Dokumente aus dem Model zu generieren zu können, sollte der Inhalt nach Dokument-Gesichtspunkten **strukturiert werden.**[31] Die Vorgehensweise hängt hier maßgeblich vom Dokumenttyp ab.

Über die Boardmitel von EA hinaus können mittels **Tagged-Values**[32] und **Stereotypes**[33] spezielle Anweisungen an den EA2Latex-Generator gegeben werden, um die Dokumentenausgabe weiter zu beeinflussen.

[27] Siehe Abschnitt 5.3.3 » <document_root>, Seite 13
[28] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_doc_filename (Attribut), Seite 39
[29] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_toc_depth (Attribut), Seite 39
[30] Siehe Abschnitt 6.1 » Dokument definieren, Seite 16
[31] Siehe Abschnitt 6.2.1 » Dokument strukturieren, Seite 20
[32] Siehe Abschnitt 7.2 Tagged Values, Seite 36
[33] Siehe Abschnitt 7.1 Stereotype, Seite 34

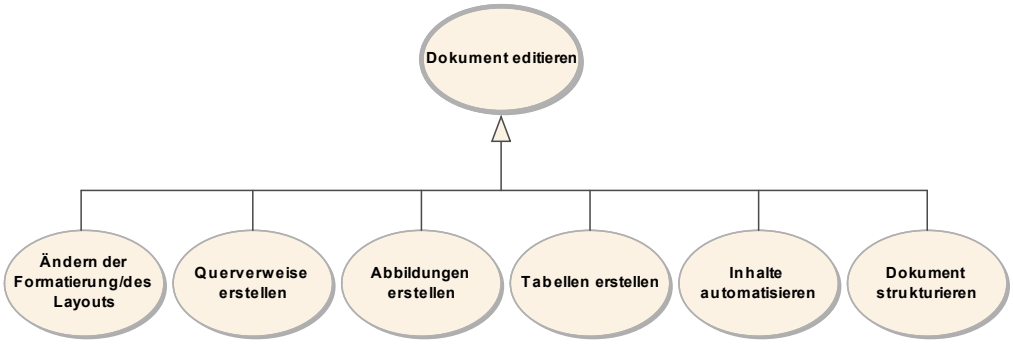


Abbildung 7: Möglichkeiten, ein Dokument zu editieren

6.2.1 Dokument strukturieren (UseCase)

Die Strukturierung von Dokumenten erfolgt über die Definition von Abschnitten.

Aus Model-Sicht definiert jedes Model-Element einen Abschnitt im Dokument, sei es ein Package, ein Use-Case oder ein sonstiges Element.

Abschnitte erhalten als Überschrift den Namen des jeweiligen Elements und darunter den Inhalt des Notes-Feldes. Danach folgen die dem Element untergeordneten Diagramme, Tabellen und weiteren Elemente.

Es gibt Einschränkungen in den Möglichkeiten der Strukturdefinition, welche sich aus der UML-Spezifikation ergeben:

- Nicht-Package-Elemente können nicht jeden anderen Elementtyp enthalten. Die wenigsten Elemente können zum Beispiel Packages enthalten.
- Packages und andere Model-Elemente können nicht beliebig in der Reihenfolge gemischt werden: Packages stehen immer vor allen anderen Elementen.

Aus diesen Einschränkungen heraus ist es sinnvoll, Packages vor allem in den oberen Ebenen der Dokumentenstruktur einzusetzen.

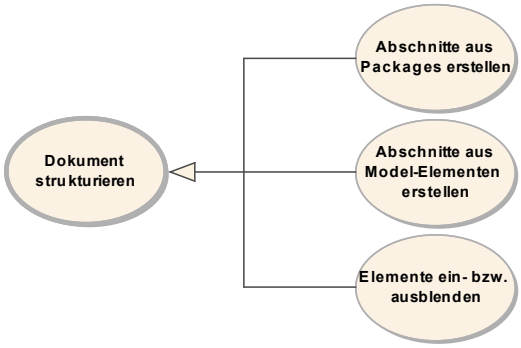


Abbildung 8: Möglichkeiten, ein Dokument zu strukturieren

6.2.1.1 Abschnitte aus Packages erstellen (UseCase)

Packages eignen sich auf Grund ihrer begrenzten Einsetzbarkeit unterhalb anderer Elemente vor allem zur Definition von übergeordneten Dokumentenabschnitten.

Packages geben dem Dokument (und natürlich auch dem Model) somit eine Grundstruktur und dienen auch der Definition von eigentlich Model-fremden Dokumentinhalten wie "Einleitung", "Anhang", und so weiter.

Es ist jedoch auch möglich ein Spezifikationsdokument vollständig über Packages zu strukturieren und Inhalt allein über die Package-Notes zu liefern. Dabei verzichtet man dann aber auf die semantische und formale Aussagekraft der UML-Notation.

Das EA2Latex Addin erzeugt aus Package-Namen in der Standardeinstellung nummerierte Überschriften. Die Nummerierung wird entsprechend der Package-Struktur verschachtelt. Dieses Verhalten kann über das Tagged Value **ea2tex_section_style**[34] explizit gesteuert werden.

6.2.1.2 Abschnitte aus Model-Elementen erstellen (UseCase)

Die Namen von Nicht-Package-Elementen werden von EA2Latex in der Standardeinstellung als nicht-nummerierte Überschriften dargestellt. Die Notes und weitere Inhalte folgen darunter ohne erkennbare Verschachtelung.

Dieses Verhalten kann über das Tagged Value **ea2tex_section_style**[35] explizit gesteuert werden.

6.2.1.3 Elemente ein- bzw. ausblenden (UseCase)

Über das TaggedValue **ea2tex_visibility**[36] lassen sich beliebige Elemente, das heißt Abschnitte des Dokuments, ein- bzw. ausblenden.

In der Default-Einstellung wird die Ausgabe vom Elementtyp abhängig gemacht:

- Requirements werden immer eingeblendet
- Andere Elemente werden nur ausgeblendet, wenn das Notes-Feld leer ist.

Hinweis: Packages lassen sich nicht ausblenden.

[34] Siehe Tabelle 6 Package Tagged Values » ea2tex_section_style (Attribut), Seite 40
[35] Siehe Tabelle 6 Package Tagged Values » ea2tex_section_style (Attribut), Seite 40
[36] Siehe Tabelle 4 Element Tagged Values » ea2tex_visibility (Attribut), Seite 37

6.2.2 Querverweise erstellen (UseCase)

Ein Schwerpunkt des EA2Latex Addins ist die Erstellung von Querverweisen und die Nachverfolgbarkeit (Traceability). Querverweise lassen sich im EA erstellen, indem man im Notes-Feld eines Elements den Text markiert, der einen Querverweis enthalten soll und entweder

- über Rechtsklick->Create->Link To Existing Element, das Ziel des Querverweises auswählt
- auf den Hyperlink-Button in der Menüleiste des Notes-Feldes klickt und das Ziel des Querverweises auswählt.

Das Plugin erstellt automatisch dependency-Verknüpfungen vom Stereotype hyperlink von den Elementen, in denen der Querverweis steht zu dem Zielelement des Verweises. Über diese Verknüpfungen kann der Nutzer im Traceability-Fenster nachvollziehen, ob auf ein Element verwiesen wird und von wo.

Wenn beim Generieren des Elements Verweisziele nicht existieren, erscheint im finalen Dokument der folgende Text: "(Verweis auf nicht existentes Element)" und der **Plausibilitätscheck**[37] schlägt an.

Abbildung 9: Querverweisevarianten

6.2.2.1 Verweise auf interne Dokumentinhalte (UseCase)

Wählt man einen Dokument-internes Verweisziel, so erstellt das Addin automatisch ein entsprechendes Latex-Kommando an der Stelle. Das Layout wird in der **eatexcmd.tex**[38]-Datei festgelegt.

6.2.2.2 Verweise auf externe Dokumentinhalte (UseCase)

Bei einem Verweisziel in einem externen Dokument (das aber auch im EA-Model enthalten ist und generiert werden kann), muss man dem Model Document noch mitgeben, dass es sich auf ein bestimmtes Dokument bezieht. Dafür zieht man eine Dependency-Verknüpfung vom Model Document, das den Querverweis enthält, zu dem Model Document, das das Verweisziel enthält.

[37] Siehe Abschnitt 6.3.3 » Fehlerlog anzeigen, Seite 32
[38] Siehe Abschnitt 5.3.3 » eatexcmd.tex, Seite 12

Das Zieldokument kann darüber auch **in einem Referenzen-Abschnitt aufgelistet werden**[39].

Das Addin kümmert sich automatisch darum, die entsprechenden Latex-Kommandos in den generierten TeX-Dateien unterzubringen, damit der Verweis auf den Abschnitt im externen Dokument aufgelöst werden kann.

6.2.3 Abbildungen erstellen (UseCase)

Abbildungen können aus Model-Diagrammen oder Artefakten ins Dokument eingebunden werden.

Der Name des Diagramms oder Artefakts wird für die Beschriftung im Dokument herangezogen.

Die Notes werden zusätzlich zur Beschriftung im Text ausgegeben. Die Platzierung und Formatierung erfolgt je nach LaTeX-Template, zum Beispiel kursiv oberhalb der Beschriftung.

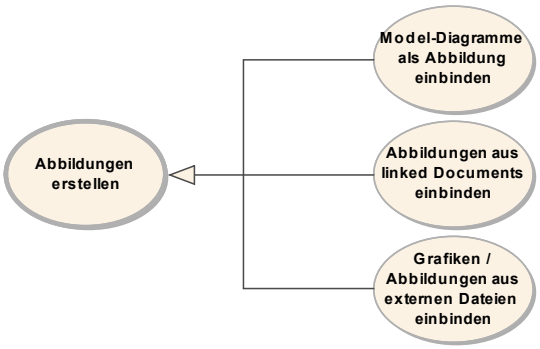


Abbildung 10: Möglichkeiten, Abbildungen zu erstellen

6.2.3.1 Model-Diagramme als Abbildung einbinden (UseCase)

Model-Diagramme werden automatisch als Abbildungen unterhalb des Elements eingefügt, dem sie im Model zugeordnet sind.

Hinweis: Diagramme werden nicht eingebunden, wenn in den Diagram-Properties der Haken bei "Exclude image from Documentation" gesetzt ist.

6.2.3.2 Abbildungen aus linked Documents einbinden (UseCase)

Es ist möglich Abbildungen, die in linked Documents enthalten sind, im Dokument auszugeben. Dazu muss folgendes beachtet werden:

[39] Siehe Abschnitt 6.2.5.2 » Referenzen-Abschnitt generieren, Seite 28

- Es muss ein Artefaktelement erstellt werden.
- Für dieses Artefakt muss in einem linked Document eine Abbildung enthalten sein.
- Der Stereotype des Artefakts ist auf **figure**[40] zu setzen.
- Der Name der Abbildung ist der Name des Artefakts.

Hinweise:

- Die Breite der Abbildung sollte manuell im rtf Dokument angepasst werden, damit eine optimale Darstellung im Dokument gewährleistet ist.
- Notes sind optional
- Folgende Dateiformate für Abbildungen werden unterstützt: *.bmp, *.wmf, *.jpg, *.png, *.gif, *.emf, *.tif.
- Im verlinkten rtf Dokument sollte sich nur die Abbildung befinden.

6.2.3.3 Grafiken / Abbildungen aus externen Dateien einbinden (UseCase)

Es können externe Dokumente vom Typ .jpg, .pdf, .png, .eps direkt im Ausgangsdokument angezeigt werden. Dazu muss folgendes beachtet werden:

- Ein Artefakt muss erstellt werden und das Tagged Value **ea2tex_filename**[41] muss den Dateinamen als Pfadangabe relativ zum Sourceverzeichnis des Dokuments unterhalb der **document_root**[42] auf dem Buildserver beinhalten.
- Der Stereotype des Artefakts ist auf **figure**[43] zu setzen.
- Die Datei wird dann ausgegeben und darunter werden die Artfakt Notes angezeigt.
- Der Name der Abbildung ist der Name des Artefakts.

6.2.4 Tabellen erstellen (UseCase)

Tabellen können auf Basis verschiedener Elementtypen und -eigenschaften erstellt werden.

Der Name des Elements wird für die Beschriftung der Tabelle im Dokument herangezogen.

Die Notes werden zusätzlich zur Beschriftung im Text ausgegeben. Die Platzierung und Formatierung erfolgt je nach LaTeX-Template, zum Beispiel kursiv unterhalb der Beschriftung.

[40] Siehe Tabelle 1 Artifact Stereotypes » figure (Attribut), Seite 36
[41] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_filename (Attribut), Seite 37
[42] Siehe Abschnitt 5.3.3 » <document_root>, Seite 13
[43] Siehe Tabelle 1 Artifact Stereotypes » figure (Attribut), Seite 36

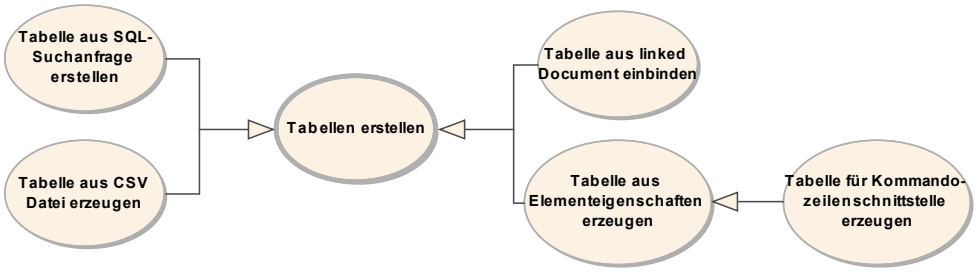


Abbildung 11: Möglichkeiten, Tabellen zu erstellen

6.2.4.1 Tabelle aus Elementeigenschaften erzeugen (UseCase)

EA2Latex erzeugt automatisch Tabellen für die folgenden Eigenschaften eines Elements:

- Methoden
- Attribute
- Runstates
- Requirements

6.2.4.2 Tabelle aus SQL-Suchanfrage erstellen (UseCase)

EA2Latex kann eine (zwei- oder dreispaltige) Tabelle mit den Ergebissen einer SQL-Suchanfrage erstellen. Hierfür muss folgendes beachtet werden:

- Ein Artifact muss verwendet werden,
- Das Artifact muss vom Stereotype **table**[44] sein,
- Die Überschrift der Tabelle ist der Name des Artifacts
- das Tagged Value **ea2tex_sql_query**[45] muss die SQL-Suchanfrage beinhalten.
- zusätzlich kann die Tabelle um 90 grad gedreht werden, dazu muss das Tagged Value **ea2tex_orientation**[46] mit "landscape" verwendet werden. Die Angabe dieses Tagged Value ist optional

6.2.4.3 Tabelle für Kommandozeilenschnittstelle erzeugen (UseCase)

Die Aufrufsyntax einer Kommandozeilenschnittstelle kann tabellarisch dargestellt werden.

[44] Siehe Tabelle 1 Artifact Stereotypes » table (Attribut), Seite 36
[45] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_sql_query (Attribut), Seite 37
[46] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_orientation (Attribut), Seite 37

Für die Spezifikation einer Kommandozeilenschnittstelle muss ein EA-Element vom Typ Interface verwendet werden. Der Stereotyp muss auf **command interface**[47] geändert werden. Damit ist sicher gestellt, dass die Tabellen der Methoden und Attribute einer Kommandzeilenschnittstelle richtig dargestellt werden.

6.2.4.4 Tabelle aus linked Document einbinden (UseCase)

Es ist möglich Tabellen, die in linked Documents enthalten sind, im Dokument auszugeben. Dazu muss folgendes beachtet werden:

- Es muss ein Artefaktelement erstellt werden.
- Für dieses Artefakt muss in einem linked Document eine Tabelle / Abbildung enthalten sein.
- Der Stereotype des Artefakts ist auf **table**[48] zu setzen.
- Der Name der Tabelle ist der Name des Artefakts.
- Die Notes des Artefakts werden zwischen dem Header der Tabelle und der eigentlichen Tabelle ausgegeben

Hinweise:

- Die Breite der Tabelle sollte manuell im rtf Dokument angepasst werden, damit eine optimale Darstellung im Dokument gewährleistet ist.
- Notes sind optional
- Im verlinkten rtf Dokument sollte sich nur die Tabelle befinden.

6.2.4.5 Tabelle aus CSV Datei erzeugen (UseCase)

EA2Latex kann automatisch eine Tabelle aus einer CSV Datei erzeugen. Folgendes ist zu beachten:

- Ein Artefakt vom Stereotype **table**[49] muss erzeugt werden
- Der Name des Artefakts ist der Name der Tabelle
- Die Artfakt Notes werden (falls vorhanden) zwischen dem Tabellenlabel (z.B. Tabelle 1: Daten) und der Tabelle ausgegeben

Es müssen folgende Tagged Values gesetzt werden:

[47] Siehe Tabelle 2 Interface Stereotypes » command interface (Attribut), Seite 36
[48] Siehe Tabelle 1 Artifact Stereotypes » table (Attribut), Seite 36
[49] Siehe Tabelle 1 Artifact Stereotypes » table (Attribut), Seite 36

- **ea2tex_filename**[50]: Dateiname, relativ zum Sourceverzeichnis des Dokuments unterhalb der **document_root**[51] auf dem Buildserver
- **ea2tex_filetype**[52]: Muss "csv" sein
- **ea2tex_csv_delimiter**[53] (optional): Default ist ","

Hinweise:

- Momentan wird die maximale Breite der Tabelle nicht kontrolliert und es finden keine Zeilenumbrüche innerhalb einer Zelle statt, d.h. wenn die Tabelle zu viele Daten enthält, passt die Tabelle nicht auf die Seite.

6.2.5 Inhalte automatisieren (UseCase)

Einzelne Packages/Abschnitte des Dokuments können von EA2Latex mit automatisch generierten Inhalten gefüllt werden. Basis für Inhalte stellen diverse Verknüpfungsinformationen innerhalb des Modells dar.

Abbildung 12: Möglichkeiten, Inhalte zu automatisieren

6.2.5.1 Anforderungsverfolgung generieren (UseCase)

Im Laufe des Softwaredesigns kann es hilfreich sein, sich anzeigen zu lassen, welche Anforderungen bereits von welchen Designkonzepten oder -komponenten realisiert werden und welche Anforderungen noch nicht realisiert werden. EA2Latex bietet zu diesem Zweck eine "Anforderungsverfolgung". Darüber kann der Nutzer sich die realisierten und nicht realisierten Anforderungen tabellarisch anzeigen lassen. Bei den realisierten Anforderungen wird auf die Abschnitte im Dokument verwiesen, die die realisierende Komponente enthalten.

Um die Anforderungsverfolgung zu erstellen sind folgende Schritte notwendig.

1. Package an der Stelle im package-Baum erstellen, wo die Anforderungsverfolgung im Dokument landen soll
2. dem Package das TaggedValue **ea2tex_req_trace**[54] geben

[50] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_filename (Attribut), Seite 37
[51] Siehe Abschnitt 5.3.3 » <document_root>, Seite 13
[52] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_filetype (Attribut), Seite 37
[53] Siehe Tabelle 3 Artifact Tagged Values » ea2tex_csv_delimiter (Attribut), Seite 37
[54] Siehe Tabelle 6 Package Tagged Values » ea2tex_req_trace (Attribut), Seite 39

3. optional den Wert des TaggedValue auf *realizedonly* oder *unrealizedonly* setzen, wenn man nur die realisierten bzw. die nicht realisierten Anforderungen in der Anforderungsverfolgung sehen möchte
4. Dependency Verknüpfungen von diesem Package zu den Packages ziehen, die die zu verfolgenden Anforderungen enthalten. (Am einfachsten durch Erstellen eines Package-Diagrams)

Die Anforderungen müssen über die realized Verknüpfung mit dem(n) realisierenden Element(en) verknüpft werden, damit sie als realisierte Anforderungen erkannt werden.

6.2.5.2 Referenzen-Abschnitt generieren (UseCase)

Über das TaggedValue **ea2tex_references**[55] lässt sich ein Package als Referenzen-Package markieren. In diesen Abschnitt des Dokuments werden dann die Dokument-Referenzen, die über Dependency-Verknüpfungen zu anderen Dokumenten im Modell markiert wurden, generiert. Es wird jeweils ein Titel und ein Dateipfad für das refenzierte Dokument erzeugt.

Es werden Model Documents und Artifact Documents berücksichtigt. Für Model Documents gibt es drei Möglichkeiten für die Bestimmung des Titels, dabei werden alle Möglichkeiten geprüft und nach folgender Hierachie ausgewertet:

1. Aus den Tagged Values **ea2tex_document_title**[56] und ea2tex_document_type wird der Titel erstellt.
2. Der Titel ist gleich dem Tagged Value **ea2tex_doc_filename**[57].
3. Der Titel ist gleich dem Namen des Model Documents.

In allen drei Fällen wird der Pfad automatisch erstellt und der Dateiname ist gleich dem Titel (mit der Endung '.pdf').

Für Artifact Documents wird immer der Name des Artifact Documents als Titel verwendet. Für den Dateipfad gilt folgendes:

- Der Dateiname ist gleich dem Tagged Value **ea2tex_doc_filename**[58] und der Pfad wird automatisch erzeugt.
- Falls unter 'Files' ein Pfad (URL oder Lokale Dateil) existiert, wird dieser als Dateipfad verwendet.
- Falls keiner der beiden genannten Fälle zutrifft wird kein Pfad verwendet (Hilfreich, wenn z.B. ein physisch vorhandes Dokument referenziert werden soll).

[55] Siehe Tabelle 6 Package Tagged Values » ea2tex_references (Attribut), Seite 40
[56] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_document_title (Attribut), Seite 38
[57] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_doc_filename (Attribut), Seite 39
[58] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_doc_filename (Attribut), Seite 39

6.2.6 Ändern der Formatierung/des Layouts (UseCase)

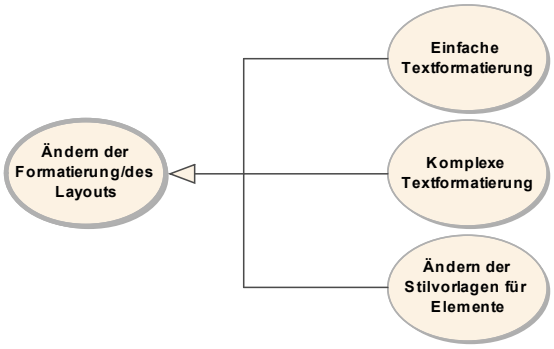


Abbildung 13: Möglichkeiten der Fomatierung

6.2.6.1 Einfache Textformatierung (UseCase)

Einfache Formatierung von Texten kann innerhalb des Notes-Feldes des zu bearbeitenden Elements stattfinden. Unterstützt werden:

- fette oder kursive Hervorhebungen
- Unterstreichungen
- nummerierte einfache Listen
- unnummerierte einfache Listen
- Hochstellen
- Tiefstellen
- Hyperlinks
- Ändern der Schriftfarbe

Komplexere Formatierungen können über linked Documents im RTF-Format **realisiert werden.**[59]

6.2.6.2 Komplexe Textformatierung (UseCase)

Komplexere Formatierungen, die über die **Möglichkeiten des Notes-Feldes**[60] hinaus gehen, können über das Linked Document Feature von Enterprise Architect realisiert werden.

Linked Documents können zum Beispiel über das Kontextmenü eines Elements erstellt werden und erlauben diverse RTF-Formatierungen.

[59] Siehe Abschnitt 6.2.6.2 » Komplexe Textformatierung, Seite 29
[60] Siehe Abschnitt 6.2.6.1 » Einfache Textformatierung, Seite 29

Hinweis: es ist auch Möglich per Copy/Paste Texte aus anderen Anwendungen in ein Linked Document zu kopieren und somit Formatierungen zu ermöglichen, die aus dem EA-Editor nicht direkt zugänglich sind, zum Beispiel Formeln.

Warnung: nicht Möglichkeiten des RTF-Formats werden vom EA2Latex-Buildprozess unterstützt. Eine Einschränkung ergibt sich aus den Fähigkeiten des für die Konvertierung verwendeten **rtf2latex**[61] Programms. Desweiteren ist es nicht möglich Fremdinhalte, z.B. über die OLE-Schnittstelle, einzubinden.

6.2.6.3 Ändern der Stilvorlagen für Elemente (UseCase)

Die Formatierung des Dokuments lässt sich durch Anpassen der **LaTeX-Templates auf dem Server**[62] ändern.

Die von EA2Latex verwendeten LaTeX Kommandos für die verschiedenen Elementtypen sind im **Anhang**[63] beschrieben.

6.3 Dokument generieren (UseCase)

Die Dokumentengenerierung folgt grundsätzlich folgendem Ablauf:

1. Generieren von LaTeX-Code aus gewähltem Teil des EA-Modells.
2. Kopieren des generierten Codes auf den Build-Server
3. Starten des Build-Kommandos auf dem Build-Server zum Erzeugen des finalen Dokuments.

Die Dokumentengenerierung kann über das Kontextmenü des Project Browsers oder einen bereits geöffnetes Dialogfenster des EA2Latex-Addins gestartet werden.

Im Anschluss an die Dokumentengenerierung hat der Nutzer die Möglichkeit, das Dokument direkt zu öffnen **öffnen**[64] oder zu weitergehend zu **verwalten**[65].

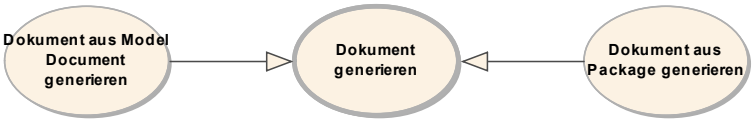
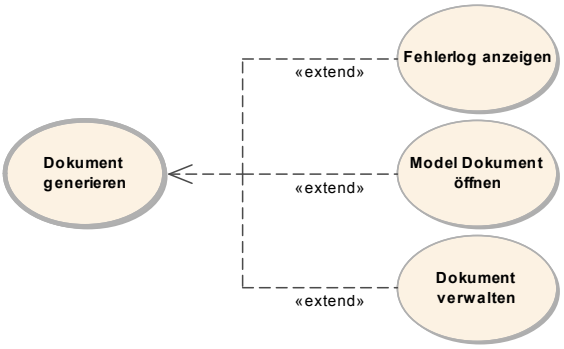


Abbildung 14: Möglichkeiten, ein zu Dokument generieren

[61] Siehe Abschnitt 3.2 » rtf2latex, Seite 8
[62] Siehe Abschnitt 5.3 Installation des Buildservers, Seite 10
[63] Siehe Abschnitt 7.3 LaTeX Kommandos, Seite 40
[64] Siehe Abschnitt 6.4.1 » Model Dokument öffnen, Seite 32
[65] Siehe Abschnitt 6.5 » Dokument verwalten, Seite 33



Nach dem generieren eines Dokuments werden dem Nutzer weitere Schritte angeboten, um den Workflow zu unterstützen.

Abbildung 15: Optionale Schritte nach dem Generieren eins Dokuments

6.3.1 Dokument aus Model Document generieren (UseCase)

Aus einem *Model Document* lässt sich über LaTeX ein PDF Dokument erstellen.

1. Rechtsklick auf das *Model Document*
2. Unter Extensions das Addin EA2Latex wählen
3. Auf "Generieren" klicken
4. Ein Statusbalken zeigt den Fortschritt an
5. Bei Beendigung hat derAnwender die Möglichkeit das PDF-Dokument anzuschauen, indem er auf "Dokument öffnen" klickt

Siehe auch: **Dokument definieren**[66]

6.3.2 Dokument aus Package generieren (UseCase)

Aus einer Package Baumstruktur lässt sich wie folgt ein PDF-Dokument erstellen:

1. Rechtsklick auf das Package
2. Unter Extensions das Addin EA2Latex wählen
3. Ein Statusbalken zeigt den Fortschritt an
4. Bei Beendigung hat derAnwender die Möglichkeit das PDF-Dokument anzuschauen, indem er auf "Dokument öffnen" klickt

Dokumenttitel, -sprache und andere Eigenschaften können für Package-Builds nicht festgelegt werden. Wenn dies gewünscht ist, muss ein Dokument **definiert**[67] und **generiert**[68] werden.

[66] Siehe Abschnitt 6.1 » Dokument definieren, Seite 16
[67] Siehe Abschnitt 6.1 » Dokument definieren, Seite 16
[68] Siehe Abschnitt 6.3.1 » Dokument aus Model Document generieren, Seite 31

6.3.3 Fehlerlog anzeigen (UseCase)

Ein Plausibilitätscheck prüft beim Generieren des Dokuments die Konsistenz des EA Models. Folgende Aspekte werden dabei betrachtet:

- Package-Elemente können nicht Kinder von Nicht-package Elementen sein.
- Hyperlinks müssen immer auf gültige Elemente verweisen

Wenn eine dieser Bedingungen nicht erfüllt ist, wird dies in einer Fehlerdatei vermerkt und am Ende der Generierung wird ein roter Fehlerlog-Button eingeblendet, über den der Nutzer sich das Log anschauen kann.

Im Fehlerlog ist das Package-Element vermerkt, wo der Fehler auftrat.

Das Dokument kann trotz dieser Meldung erzeugt werden. Dabei werden folgende Maßnahmen ergriffen:

- Packages, die Kind eines Nicht-Package-Elements sind, werden nicht ausgegeben.
- Hyperlinks auf nicht existierende Elemente erhalten im Dokument den Hinweis "Verweis auf nicht existentes Element" .

6.4 Dokument öffnen (UseCase)

Dokumente können im Project Browser über das Kontextmenü eines gewählten Packages oder Model Documents geöffnet werden.

Abbildung 16: Möglichkeiten, ein Dokument zu öffnen

6.4.1 Model Dokument öffnen (UseCase)

Aus dem Model generierte Dokumente sind entweder über Package oder ein Model Document definiert.

Diese können über folgende Wege im PDF-Viewer geöffnet werden:

- Im EA2Latex-Addin-Dialogfenster nach dem Generieren des Dokuments;
- Im EA2Latex-Addin-Dialogfenster nach dem Öffnen desselben über den "Verwalten" Kontextmenüeintrag;
- Direkt aus dem Kontextmenü zum gewählten Package oder Model Document über den Eintrag "Öffnen".

Schlägt das Öffnen fehl, weil die Datei z.B. nicht existiert, wird die Benutzeroberfläche gestartet, so dass das Dokument generiert oder ausgecheckt werden kann.

6.4.2 Externes Dokument öffnen (UseCase)

Externe Dokumente können aus dem Model heraus geöffnet werden. Dafür muss ein Model Document angelegt werden und das Tagged Value **ea2tex_doc_filename**[69] mit dem Dateinamen des externen Dokuments gesetzt werden. Das entsprechende Dokument muss natürlich auch an der entsprechenden Stelle im Dokumentenbaum liegen (eingecheckt sein).

6.5 Dokument verwalten (UseCase)

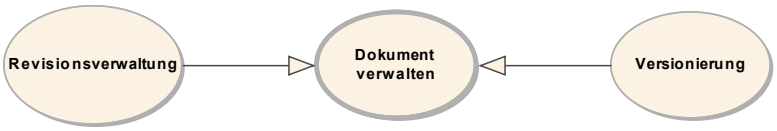


Abbildung 17: Möglichkeiten, ein Dokument zu verwalten

6.5.1 Revisionsverwaltung (UseCase)

Dokumente, die aus *Model Documents* generiert werden, können im zentralen **Repository des Dokumentenbaums**[70] abgelegt werden. Das Addin bietet über entsprechende Schaltflächen auf der Benutzeroberfläche die Möglichkeit Dokumente und Dokumentsourcen auszuchecken, einzuchecken oder hinzuzufügen (*Checkout, Commit, Add*). Beim Hinzufügen und Committen ist es erforderlich die Änderungen im aufpoppenden Dialog-Fenster zu protokollieren.

Um die Änderungen zu betrachten, kann der Nutzer die Diff-Schaltfläche benutzen. Darüber wird das Verzeichnis geöffnet in dem die Dokumentsourcen liegen. Über einen externen SVN-Client (z.B. tortoise SVN) können die Änderungen nun hervorgehoben werden.

Beim Zugriff auf das Repository wird der Benutzername und das -passwort abgefragt. Beide Angaben werden verschlüsselt in Dateien gespeichert, so dass eine einmalige Abfrage genügen sollte. Die Login-Daten können auch **zurückgesetzt**[71] werden.

Hinweis: Die Versionsverwaltung ist nur für Dokumente möglich, die als **Dokument definiert**[72] wurden.

[69] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_doc_filename (Attribut), Seite 39
[70] Siehe Abschnitt 6.1.5 » Dokumentenbaum definieren, Seite 18
[71] Siehe Abschnitt 6.5.3.1 » Login zurücksetzen, Seite 34
[72] Siehe Abschnitt 6.1 » Dokument definieren, Seite 16

6.5.2 Versionierung (UseCase)

Dokumente können versioniert werden. Dafür stellt die Benutzeroberfläche den Button Version erstellen bereit. Dieser taucht nicht auf, wenn ein **SVN-gestütztes Dokument**[73] modifiziert wurde. Ansonsten kann eine versionierte Kopie des Dokuments und der Dokument-Sourcen erstellt werden.

Beim Klicken auf den "Version erstellen"-Button wird der Nutzer nach einer Versionsnummer gefragt, die das Dokument erhalten soll. Durch Bestätigen mit OK wird das Dokument ein letztes Mal mit Versionsangabe auf der Titelseite gebaut, kopiert (samt Sourcen), umbenannt (Dateiname und Source-Verzeichnis erhalte Versionssuffix) und zur Revisionsverwaltung hinzugefügt.

Gleichzeitig wird im Model ein neues Model Document angelegt, das den Dateinamen des soeben versionierten Dokuments als Tagged value **ea2tex_doc_filename**[74] besitzt, so dass das versionierte Dokument aus dem Model heraus **geöffnet**[75] und **verlinkt**[76] werden kann.

6.5.3 Serveranbindung verwalten (UseCase)

6.5.3.1 Login zurücksetzen (UseCase)

Es gibt ein Login für die Server-Verbindung per psftp bzw. ssh und ein Login für die SVN-Revisionsverwaltung. In beiden Fällen werden die Login-Daten verschlüsselt gespeichert und müssen nicht erneut eingegeben werden. Um die Login-Daten zurückzusetzen, muss man über den Button *Konfiguration* in das Einstellungen-Menü wechseln und dort auf *Login zurücksetzen* klicken. bei der nächsten Aktion, die ein Login benötigt, werden die Login-Daten wieder abgefragt.

7 Anhang

7.1 Stereotype

Mit Hilfe des Stereotypes von EA-Elementen wird die Dokumentengenerierung weiter beeinflusst.

[73] Siehe Abschnitt 6.5.1 » Revisionsverwaltung, Seite 33
[74] Siehe Tabelle 5 Model Document Tagged Values » ea2tex_doc_filename (Attribut), Seite 39
[75] Siehe Abschnitt 6.4.2 » Externes Dokument öffnen, Seite 33
[76] Siehe Abschnitt 6.2.2.2 » Verweise auf externe Dokumentinhalte, Seite 22

Artifact Stereotypes (Class)

Stereotype Werte für Artifacts

Tabelle 1: Artifact Stereotypes Attribute

Attribut	Bemerkungen
figure string	Ein Artifact muss vom Stereotype "figure" sein, wenn eine Abbildung aus einem **linked Document**[77] oder einer **externen Datei**[78] ausgegeben werden soll.
table string	Ein Artifact muss vom Stereotype "table" sein, wenn • eine **Tabelle aus einer SQL-Suchanfrage**[79] erzeugt werden soll, oder • eine **Tabelle aus einem linked Document ausgegeben**[80] werden soll, oder • eine **Tabelle aus einer CSV Datei**[81] erzeugt werden soll.

Interface Stereotypes (Class)

Stereotype Werte für Interfaces

Tabelle 2: Interface Stereotypes Attribute

Attribut	Bemerkungen
command interface string	Für Kommandozeilenschnittstellen muss das Interface den Stereotype "command interface" besitzen, damit Befehle und Paramater im Dokument in **Tabellen**[82] richtig dargestellt werden

7.2 Tagged Values

Die Beeinflussung der Dokumentgenerierung aus dem Modell heraus erfolgt über Tagged Values. In diesem Abschnitt werden die EA-Elemente beschrieben, in denen Tagged Values definiert werden können.

Artifact Tagged Values (Class)

Tagged Value Einstellungen für Artefakte

[77] Siehe Abschnitt 6.2.3.2 » Abbildungen aus linked Documents einbinden, Seite 23
[78] Siehe Abschnitt 6.2.3.3 » Grafiken / Abbildungen aus externen Dateien einbinden, Seite 24
[79] Siehe Abschnitt 6.2.4.2 » Tabelle aus SQL-Suchanfrage erstellen, Seite 25
[80] Siehe Abschnitt 6.2.4.4 » Tabelle aus linked Document einbinden, Seite 26
[81] Siehe Abschnitt 6.2.4.5 » Tabelle aus CSV Datei erzeugen, Seite 26
[82] Siehe Abschnitt 6.2.4.3 » Tabelle für Kommandozeilenschnittstelle erzeugen, Seite 25

Tabelle 3: Artifact Tagged Values Attribute

Attribut	Bemerkungen
ea2tex_csv_delimiter String	Wenn eine **Tabelle aus einer CSV Datei**[83] erzeugt werden soll, dann kann mit diesem Tagged Value der Demiliter der CSV Datei angegeben werden. Dieses Tagged Value ist optional. Default: ",".
ea2tex_filename String	Um eine **externe Grafik**[84] vom Dateityp .pdf, .png, .jpg oder .eps in das Ausgabedokument einzubinden oder **eine Tabelle aus einer CSV Datei zu erzeugen**[85], muss dieses Tagged Value gesetzt sein und den Dateinamen als Pfadangabe relativ zum Sourceverzeichnis des Dokuments beinhalten.
ea2tex_filetype String	Wenn eine **Tabelle aus einer CSV Datei**[86] erzeugt werden soll, muss dieses Tagged Value "csv" sein.
ea2tex_orientation String	Wenn eine Tabelle aus einer SQL-Suchanfrage erstellt wird, kann mit diesem Tagged Value die Tabelle um 90 grad gedreht werden. Das Tagged Value muss hierfür den Wert "landscape" besitzen. Wenn dieses Tagged Value nicht gesetzt ist, wird die Tabelle normal ausgegeben.
ea2tex_sql_query String	Mit diesem Tagged Value kann eine Tabelle erzeugt werden. Die Tabelle beeinhaltet die Suchergebnisse einer **SQL-Suchanfrage**[87]. Die Suchanfrage ist in das Tagged Value zu schreiben.

Element Tagged Values (Class)

TaggedValue-Einstellungen beliebiger Elemente.

Tabelle 4: Element Tagged Values Attribute

Attribut	Bemerkungen
ea2tex_visibility String	(optional) Über dieses tagged value lassen sich beliebige Elemente ein- und ausblenden. Mögliche Werte sind: • *always* - Element wird immer im Dokument dargestellt • *never* - Element wird nie im Dokument dargestellt. • *notes* - (default) Element wird dargestellt, wenn das Notes-Feld nicht leer ist. Hinweis: Bei Requirements ist *always* der Default-Wert. Default: notes

[83] Siehe Abschnitt 6.2.4.5 » Tabelle aus CSV Datei erzeugen, Seite 26
[84] Siehe Abschnitt 6.2.3.3 » Grafiken / Abbildungen aus externen Dateien einbinden, Seite 24
[85] Siehe Abschnitt 6.2.4.5 » Tabelle aus CSV Datei erzeugen, Seite 26
[86] Siehe Abschnitt 6.2.4.5 » Tabelle aus CSV Datei erzeugen, Seite 26
[87] Siehe Abschnitt 6.2.4.2 » Tabelle aus SQL-Suchanfrage erstellen, Seite 25

Attribut	Bemerkungen
ea2tex_section_style String	Elemente werden üblicherweise nicht zur Definition von Sections verwendet. Über folgende Einstellungen des ea2tex_section_style Tagged Values kann das Verhalten aber beeinflusst werden: • "tree": das Element wird als (hierarchische) Section behandelt und wird mit dem passenden Latex-Kommando versehen. Damit erhält es in der Standard-Buildumgebung eine Nummer, einen Eintrag im Inhaltsverzeichnis und enthaltene Elemente können weiter untergliederte Nummern erhalten. • "flat" (default): das Element soll nicht als Section behandelt werden. Diese Einstellung wirkt sich rekursiv auf alle enthaltenen Packages aus, so dass es zu einer flachen Darstellung der Elemente kommt.

Model Document Tagged Values (Class)

Model Document Elemente werden für die **Definition von Dokumenten**[88] verwendet.

Tabelle 5: Model Document Tagged Values Attribute

Attribut	Bemerkungen
ea2tex_document_type String	Legt den Typ des Dokuments fest. Der Typ soll die Art des Spezifikationsdokuments beschreiben. Der Zugriff in LaTeX erfolgt über das Kommando \eatexDoctype. Die Standardtemplates schreiben den Typ auf die Titelseite und in die Kopfzeile des Dokuments. Ein Beispiel für eine Typbezeichnung ist: "Anforderungsspezifikation"
ea2tex_document_title String	Legt den Titel des Dokuments fest. Auf diesen wird in LaTeX über das Kommando \eatexTitle zugegriffen. Die Standardtemplates schreiben den Typ auf die Titelseite und in die Kopfzeile des Dokuments. Als Titel wird oft das Produkt angegeben, das in dem Dokument spezifiziert wird.
ea2tex_document_lang englisch,deutsch	Legt die Sprache des Dokuments fest. Die Sprachfestlegung beeinflusst in LaTeX u.a. die Bezeichner für Verzeichnisse, Tabellen, Abbildungen und Querverweise aber auch die Silbentrennung. Default: deutsch

[88] Siehe Abschnitt 6.1 » Dokument definieren, Seite 16

Attribut	Bemerkungen
ea2tex_config_name Dateiname	(optional) Ermöglicht die Spezifikation einer alternativen EA2Latex Konfigurationsdatei, die bei der Erzeugung des Dokuments verwendet wird. Diese wird vom EA2Latex Addin in dessen Konfigurationsordner des **erwartet.**[89]
ea2tex_skip_root_package bool	(optional) Gibt an, ob das Top-Level Package mit in das zu generierende Dokument aufgenommen werden soll oder nicht. Ist defaultmäßig *true*, das Top-Level-Verzeichnis wird also übersprungen. Default: true
ea2tex_toc_depth int	(optional) Gibt die Tiefe des Inhaltsverzeichnisses im Dokument an. Wenn nicht angegeben wird die Tiefe auf 3 begrenzt. Default: 3
ea2tex_doc_filename String	Dateiname des generierten Dokuments. Wenn nicht angegeben wird der Dateiname aus Dokument-Typ und -Titel generiert. Der Dateiname muss angegeben werden, wenn das Model Document ein versioniertes oder ein externes Dokument darstellt.

Package Tagged Values (Class)

Package-Elemente werden in der Regel für die **Definition von Abschnitten**[90] im Dokument verwendet.

Tabelle 6: Package Tagged Values Attribute

Attribut	Bemerkungen
ea2tex_document_root String	Package-Elemente, die die **Dokumentenstruktur**[91] enthalten, können das Tagged Value *ea2tex_document_root* erhalten, worüber der Name des Toplevel-Verzeichnisses gesetzt werden kann. Ein Beispiel für ein Root-Verzeichnis ist: "comos"
ea2tex_req_trace String	Ein Package mit diesem Tagged Value wird für die **Anforderungsverfolgung**[92] benutzt. Indem man den Wert auf *realizedonly* oder *unrealizedonly* setzt, kann man die Anforderungsverfolgung nur auf die realisierten bzw. die nicht realisierten Anforderungen beschränken. Defaultmäßig werden sowohl die realisierten wie auch die nicht realisierten Anforderungen in der Anforderungsverfolgung angegeben.

[89] Siehe Abschnitt 5.5 Konfigurationsdatei, Seite 14
[90] Siehe Abschnitt 6.2.1.1 » Abschnitte aus Packages erstellen, Seite 21
[91] Siehe Abschnitt 6.1.5 » Dokumentenbaum definieren, Seite 18
[92] Siehe Abschnitt 6.2.5.1 » Anforderungsverfolgung generieren, Seite 27

Attribut	Bemerkungen
ea2tex_section_style tree,flat	Packages werden üblicherweise zur Definition von Sections verwendet und gliedern somit das Dokument in einer Hierarchie, je nachdem wie sie verschachtelt sind. Über folgende Einstellungen des ea2tex_section_style Tagged Values kann das Verhalten beeinflusst werden: • "tree" (default): das Package wird als (hierarchische) Section behandelt und wird mit dem passenden **Latex-Kommando**[93] \DSPEsectlvl* versehen. Damit erhält es in der **Standard-Buildumgebung**[94] eine Nummer, einen Eintrag im Inhaltsverzeichnis und enthaltene Packages erhalten weiter untergliederte Nummern. • "flat": das Package soll nicht als Section behandelt werden sondern wie ein einfaches Element dargestellt. Es wird mit dem Latex Kommando \DSPEflatsect versehen. Diese Einstellung wirkt sich rekursiv auf alle enthaltenen Packages aus, so dass es zu einer flachen Darstellung der Packages kommt. Default: tree
ea2tex_svn_root String	Gibt den Pfad des Dokumentationsverzeichnisses (*documentation*) des Projekts im SVN an. Muss angegeben werden, wenn Revisionsverwaltung möglich sein soll.
ea2tex_references bool	(optional) Ist dieses Tagged Value gesetzt, werden in dieses Package automatisch Referenzen generiert, die aus den Abhängigkeiten (Dependency-Verknüpfungen) des zu generierenden Model Documents von anderen Model Documents abgeleitet werden. Default: false
ea2tex_model_path String	Pfad zum EA-Model aus dem das Dokument generiert wird. Dieser Pfad wird auf der Titelseite angegeben.

7.3 LaTeX Kommandos

Im folgenden sind alle vom Addin verwendeten **LaTeX Kommandos**[95] definiert.

TODO: ist zu aktualisieren!

Überschriften

Überschriftebenen 0 bis 10: *\DSPEsectlvlzero* ... *\DSPEsectlvlten.* Erhalten als Argument:

- #1 den Package-Namen

[93] Siehe Abschnitt 7.3 LaTeX Kommandos, Seite 40
[94] Siehe Abschnitt 5.3.3 Standard-Build-Umgebung bereitstellen, Seite 10
[95] Siehe Abschnitt 5.3.3 » eatexcmd.tex, Seite 12

Packages, die nicht als Section interpretiert werden sollen: *DSPEflatsect*. Erhält als Argument:

- #1 den Package-Namen

Package Elements

Default Element: *DSPEelement*. Erhält die Argumente:

- #1 Element-Name
- #2 Element-Notes
- #3 Element-GUID

Artefakt: *DSPEartifact*. Erhält zusätzlich zu den *DSPEelement* Argumenten noch das Argument:

- #4 Der Dateipfad zur vom Artefakt modellierten Datei

Collaboration: *DSPEcollab* (wie Default-Element)

Component: *DSPEcomponent* (wie Default-Element)

Image component: *DSPEimgcomponent*. Erhält die Argumente:

- #1 Element-Name
- #2 Element-Notes

Object: *DSPEobject* (wie Default-Element)

Class: *DSPEclasselem* (wie Default-Element)

Interface: *DSPEinterface* (wie Default-Element)

State: *DSPEstate* (wie Default-Element)

Scenario: *DSPEscenario*. Erhält in der Reihenfolge die Argumente:

- #1 den Scenario-Namen
- #2 die Scenario-Notes
- #3 den Type
- #4 den ObjectType

Scenario-Step: *DSPEscenariostep*. Erhält in der Reihenfolge die Argumente:

- #1 den Step-Namen
- #2 das Level
- #3 die Position
- #4 den State

Scenario Extension: *DSPEscenarioext*. Erhält als Argumente:

- #1 Extension-Name
- #2 Level
- #3 Typ

Requirement: *DSPErequirement*. Erhält als Argumente:

- #1 Requirement Namen
- #2 Requirement Notes
- #3 Stereotype (z.B. functional)
- #4 Status (z.B. proposed)
- #5 Difficulty
- #6 Priority
- #7 Requirement Nummer (z.B. REQ-23)
- #8 Guid

Abbildungen

Abbildung: *DSPEfigure*. Erhält als Argumente:

- #1 Guid
- #2 Diagramm-Name
- #3 scaling. Wird als drittes Argument "scaled" übergeben, dann wurde das Diagramm bereits im EA auf Seitengröße skaliert und es wird die maximale Größe bzw. Breite benutzt. Wenn "fit", dann passt das Diagramm auf die Seite und es wird nur ein konstanter Skalierungsfaktor benutzt, damit die Seitenraender berücksichtigt werden.

Querverweise

Verweis auf eine Abbildung: *DSPEimgref*. Erhält das folgende Argument

- #1 Guid

Verweis auf ein Element: *DSPEelemref.* Erhält die folgenden Argumente

- #1 Typ des Elements auf das verwiesen wird (Package, Element, ...)
- #2 Name des Elements auf das verwiesen wird
- #3 Guid des Elements
- #4 Text der in der Referenz steht

Tabellen

Begin der Table-Umgebung: *DSPEtablebegin*. Erhält als Argumente:

- #1 Tabellentyp
- #2 Caption
- #3 Guid

Ende der Table-Umgebung: *DSPEtableend*. Erhält als Argumente:

- #1 Caption
- #2 Guid
- #3 Stereotype

Referenz auf Tabellen: *DSPEtabref*. Erhält als Argumente:

- #1 TBD
- #2 TBD
- #3 TBD

Klassen-Attribute *DSPEattribute*. Erhält als Argumente:

- #1 Namen
- #2 Typ
- #3 Notes des Klassen-Attributs
- #4 Stereotype
- #5 Default-Wert
- #6 Attribut-Style
- #7 Attribut-Constraints
- #8 Guid des Attributs
- #9 multiplicity (Array/Matrix-Dimensionen)

Klassen-Methode: *DSPEmethod*. Erhält als Argumente:

- #1 Namen
- #2 Rückgabewert
- #3 Parameter und
- #4 Notes der Klassen-Methode
- #5 Guid der Methode

Runstate: *DSPErunstate*. Erhält als Argumente:

- #1 Variablen-Namen
- #2 Operator (=, >=...)
- #3 den Wert

Anforderungsverfolgung

Requirement Trace: *DSPEreqtr.* Erhält als Argumente:

- #1 den Namen der Realisierten Anforderung
- #2 ein *DSPEreqtrref* Kommando

Requirement Trace Referenz: *DSPEreqtrref*. Erhält als Argumente:

- #1 Den Namen des realisierenden Elementes
- #2 die Guid des realisierenden Elementes

Misc

Funktionsbeschreibung (z.B. class::method()): \DSPEfuncdescr. Erhält als Argument:

- #1 die Funktionsbeschreibung

Textfarbe: \DSPEcolor. Erhält als Argumente:

- #1 die Farbe
- #2 den Text

7.4 Fehlermeldungen

In diesem Abschnitt wird eine Liste bekannter Fehlermeldungen gesammelt: **"Fehler beim Konvertieren in Package <Package-Name> (<Fehlermessage>)."**

- Es ist eine Exception beim Durchiterieren durch den Package-Baum im Package <Package-Name> aufgetreten.

"plink returned error code <ExitCode>! PDF creation failed!"

- Die Ausführung des Build-Skripts ist fehlgeschlagen.

"Failed to save Diagram as PDF!"

- Das Speichern eines Diagramms als PDF ist fehlgeschlagen.

"Failed to add new files"

- Beim Hinzufügen von Dateien im Rahamen des Commit ist ein Fehler aufgetreten

"Backgroundworker has been Cancelled!"

- Der BackgroundWorker-Thread wurde aus unkeklärter Ursache vorzeitig abgebrochen.

"Error occurred during <SVN Kommando> (<Exitcode>)! Check LOG for svn stat output!"

- Der fehler mit Fehlercode <Exitcode> ist beim Ausführen des SVN Kommandos <SVN Kommando> aufgetreten.

Abbildungsverzeichnis

Tabellenverzeichnis