Version Control Best Practices für Enterprise Architect

www.sparxsystems.com

Glossar

Baseline (Model Baseline): In Enterprise Architect bezieht sich „Baseline" auf eine Package-Momentaufnahme zu einem bestimmten Zeitpunkt. Der Schnappschuss wird in der Modell-Repository als komprimierte XMI gespeichert und bildet die Grundlage der „Compare and Merge"-Funktionalität von Enterprise Architect.
Check-in: Der Prozess der Einreichung von Änderungen an das Version Control Repository. In Enterprise Architect führen Sie diesen Befehl auf ein Paket aus, das Sie ausgecheckt haben. Dieser aktualisiert dann das Version Control Repository mit Ihren Änderungen und hebt Ihre Editiersperre für das Paket auf.

Check-out: Der Prozess bei dem die neueste Version einer Datei aus der Version Control Repository abgerufen wird. Die Ausführung dieses Befehls in Enterprise Architect überschriebt die ausgewählte Pakete mit der neuesten Version und setzt für Sie eine exklusive Editiersperre.

Commit: So können Sie Ihre Änderungen an die Version Control Repository übergeben ohne Ihre Editiersperre auf die dazugehörige(n) Datei(en) aufheben zu müssen. Dies entspricht in Enterprise Architect der Ausführung des "Put Latest" Kommandos.

DBMS: Database Management System. Um Hosting für die Model Repository zu ermöglichen unterstützt Enterprise Architect einige relationale DBMS-Produkte. Üblicherweise wird ein DBMS verwendet, wenn auf das Model Repository von mehreren Benutzern gleichzeitig zugegriffen wird.

EAP: Enterprise Architect Projekt. Die Abkürzung bezieht sich für gewöhnlich auf die Datei-basierte Model Repository, also "EAP-Datei."

Get Latest: Ein Befehl, das aus Enterprise Architect heraus auf ein ausgewähltes Paket ausgeführt wird. Es aktualisiert das Paket mit den aktuellsten Informationen aus dem Version Control Repository ohne eine Check-out durchzuführen zu müssen. "Get All Latest" aktualisiert alle versionskontrollierten Pakete innerhalb des Projektes.

Model Repository: Enterprise Architect Speichermechanismus für die Modell-Informationen. Wenn mehrere Benutzer gleichzeitig dasselbe Modell bearbeiten, wird das Repository in der Regel DBMS-basiert. In verteilten Umgebungen, in denen Team-Mehrfachzugriff auf das Modell nicht möglich ist, werden üblicherweise stattdessen die lokalen EAP-Dateien eingesetzt.

Enterprise Architect kann Ihr Version Control System für den Informationsaustausch zwischen dem Model Repository und einem Version Control Repository verwenden um Änderungen unter Teammitgliedern zu kommunizieren.

Package: Das primäre organisatorische Konstrukt in Enterprise Architect Modellen, etwa vergleichbar mit einem "Ordner" im Dateisystem. Ein Paket kann ein Modell eines Root-Knotens, eine Ansicht oder ein Sub-Paket sein.

Put Latest: Ein Befehl, der aus Enterprise Architect heraus auf einem ausgewählten Paket, das Sie ausgecheckt haben, ausgeführt wird. Es aktualisiert die Version Control Repository mit Ihren Änderungen auf diesen Paket, unter Beibehaltung seiner "Checked-out"-Status. (Entspricht etwa dem Einchecken eines Pakets, das daraufhin gleich wieder Ausgecheckt wird.)

Version Control Configuration: In Enterprise Architect zeichnet eine Version Control Configuration Verbindungseinstellungen für das Version Control Repository sowie den Pfad zu Ihrer lokalen Arbeitskopie auf.

Version Control Repository: Der Speichermechanismus von dem Version Control System zur Speicherung von Revisionen – insbesondere Modell-Revisionen.

Version Control System: Ein Produkt von Drittanbietern, das Ihrer Modell-Daten-Revisionen verwaltet. Enterprise Architect unterstützt mehrere Version Control Systemen (wie CVS-, Subversion- und SCC-konformen Produkten) und stellt die nötige Benutzeroberfläche zur Verfügung, um Daten zwischen dem Model Repository und dem Version Control Repository verschieben zu können.

Working Copy: Der Datensatz auf Ihrem lokalen Rechner, den Sie von der Version Control Repository abgerufen haben. Enterprise Architect verwendet Ihren Arbeitskopie-Dateien um das Modell und die Version Control Repositories zu aktualisieren.

XMI: XML Meta-Data Interchange. Ein offenes Standard-Dateiformat, das den Austausch von Informationen zwischen Modell-Tools ermöglicht. Enterprise Architect nutzt dieses Dateiformat um Informationen von einem Paket-Modell zu serialisieren, um eine Ablage in der Version Control Repository zu ermöglichen. Wenn die Versionskontrolle auf ein Paket in Enterprise Architect angewendet wird ist eine XMI-Datei für dieses Package (und den darin enthaltenen Elementen) erstellt und dem Version Control Repository beigefügt.

Einleitung

Dieses Dokument erläutert, wie sich Konzepte der Versionskontrolle auf Sparx Systems Enterprise Architect anzuwenden sind und empfiehlt Best-Practice-Methoden für die Versionskontrolle in gemeinsamen sowie verteilten Modell-Umgebungen. Die folgenden Informationen sind für Enterprise Architect Anwender konzipiert, die:

- Wissen wollen, ob die Anwendung der Versionskontrolle auf Ihr Modell von Vorteil sein könnte und welche Alternativen zur Verfügung stehen, oder;
- Best-Practice-Empfehlungen für spezifische Enterprise Architect Einsatzszenarien benötigen, oder;
- bereits Erfahrung mit Version-Control-Modellen haben und verbessern wollen, wie diese verwaltet werden.

Was habe ich von Version-Control-Modelle?

Einige Vorteile der Verwendung von Version Control Systems sind z.B.: die Steigerung des Potenzials für parallele und verteilte Arbeit, verbesserte Verfolgung und Zusammenführung von Änderungen im Zeitablauf, sowie die Automatisierung der Verwaltung vom Revisionsverlauf. Im Folgenden sind einige spezifische Vorteile aufgelistet, die durch die Anwendung der Versionskontrolle auf eine Modellierungsumgebung realisiert werden können. Sie sind besonders relevant, wenn Modelle gemeinsam von mehreren Editoren benutzt werden, die eventuell geografisch getrennt sind:

- Unterstütztung von global verteilter Modell-Bearbeitung durch eine bequeme und effektive Methode Modelle zu vervielfältigen.
- Erleichterung der Zusammenarbeit über mehrere Projekte durch Wiederverwendung von gemeinsamen Modelldaten.
- Verbesserung der Leistung für weit verstreuten Teams über langsame Netzwerke, indem die lokale Speicherung von Modellen ermöglicht wird, wobei nur die Änderungen global übermittelt werden.
- Förderung geordneter Veränderungen gegenüber chaotischer Veränderungen. Hilft Störungen durch die Trennung von "work in progress" und "fertiger" Arbeit zu minimieren.
- Hilfe bei der Automatisierung von Kommunikation innerhalb eines Teams durch den Koordinierten Zugang zu kontrollierten Informationen, wodurch versehentliche Änderungen verhindert werden.
- Hilfe bei der Einhaltung aufeinanderfolgender Revisionen von Work-to-Date mit nützliche Fähigkeiten wie "undo" um falsche Änderungen zu widerrufen oder "roll back" um bis zur letzten "guten" Version einen Fehler rückgängig zu machen oder eine Wiederherstellung nach ungewollter Löschung oder Änderung durchzuführen.
- Hält ein Arbeitsverlauf fest und verfolgt Änderungen an einem Modell, um besser nachvollziehen zu können, "wer was geändert hat, und wann?“

Was heist 'Version Control' in Enterprise Architect?

Abb. 1: Version Control beim Root, View, Package or Sub-Package

Enterprise Architect Version Control gilt für Pakete innerhalb eines Modells. Das Paket ist das primäre organisatorische Konstrukt für UML-Modelle. Auf jedes Paket kann Versionskontrolle angewandt werden – egal ob es der Root-Knoten des Modells, ein View oder ein Sub-Paket ist (siehe Abbildung 1).

Enterprise Architect unterstützt primär zwei Möglichkeiten, die Versionskontrolle auf Pakete in einem Modell anzuwenden:

1. **Model Baselines**: Diese eingebaute Funktion speichert Point-in-Time-Snapshots von einem Paket in der Modell-Repository selbst. Das Modell Baselines Konzept bildet auch die Grundlage der von Enterprise Architects "Vergleichen und Zusammenführen"-Funktionalität.

Bei der Entscheidung über Ansätze zur Versionskontrolle sollte die Verwendung von Model Baselines berücksichtigt werden, insbesondere wenn Revisionsverlauf für Roll-Back-, Zusammenführungs- und Vergleichszwecken erhalten werden soll.

2. **Version Control Integration**: Enterprise Architect unterstützt die Integration von Version Control Systemen von Drittanbietern die es ermöglichen, das Benutzer Paket-Revisionen in ihrem bevorzugten Werkzeug speichern können. Version Control Tools die von Enterprise Architect unterstützt werden sind, unter anderem, Subversion, CVS, Microsoft TFS und SCC-kompatible Tools wie AccuRev und Visual Source Safe.

Für große, geographisch versteute Teams ist es oft notwendig eine eigene Version eines Control-Systems zu verwenden, um breit angelegte Vervielfältigung und Austausch von Modelldaten zu ermöglichen. Dieses Dokument konzentriert sich auf den letztgenannten Ansatz: das Erreichen von Versionskontrolle mit integrierten Drittanbieter-Werkzeugen.

Leser die an verwandten Change Management Features und einfachen Alternativen interessiert sind werden an Anhang B: Built-In-Collaboration und Change-Management-Tools verwiesen.

Abbildung 2 stellt ein High-Level-Schema der Beziehung zwischen Enterprise Architect und dem externen System zur Versionskontrolle dar. Beachten Sie, dass Versionen als XMI-Dateien sowohl gespeichert als auch abgerufen werden. XMI wird verwendet um Paketdaten in serielle Reihenfolge zu bringen, sodass ein Point-in-Time-Snapshot erstellt werden kann. Enterprise Architect sorgt dafür, dass nur ein einziger Benutzer jeweils ein bestimmtes Paket bearbeiten kann. Dieser Ansatz stellt eine "Lock-Modify-Unlock"-Lösung dar, die Revisionskonflikte vermeiden verhilft – dies ist besonders nützlich, da XMI-Dateien binäre Artefakte sind, die man nicht direkt über das Version Control System zusammenführen kann.

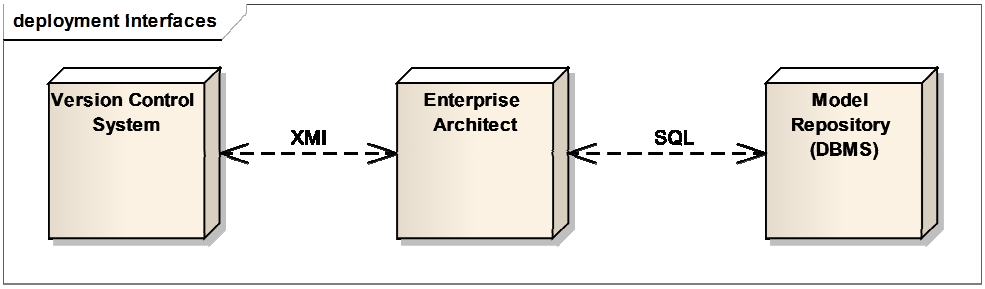

Abbildung 2: Enterprise Architect tauscht Daten mit dem Modell Versionskontrollsysteme mittels XMI-Dateien.

Der Rest dieses Dokuments soll Ihnen dabei helfen, Ihr Einsatzszenario für Enterprise Architect festzustellen und geeignete Versionskontroll-Ansätze vorzuschlagen. Für jeden Ansatz werden auch spezifische Herausforderungen hervorgehoben, welche berücksichtigt werden müssen. Statt ausführliche "how-to"-Schritte zur Einrichtung Ihres Version Control Systems mit Enterprise Architect zu geben, wird auf die relevanten Teile in der Enterprise Architect Bedienungsanleitungs verweisen.

Team -Einsatz: zentralisiert oder verteilt

Der wichtigste Aspekt bei der Entscheidung über die richtige Vorgehensweise für das Version Controlling Ihres Modells ist wie die Modell-Redakteure oder -Autoren verteilt sind:

- *Ist jedes Mitglied des Teams in der gleichen zentralen Lage und über ein Hochgeschwindigkeits-Netzwerk verbunden?*
- *Werden die Editoren eher aus der Ferne selbstständig arbeiten und von einem geteilten Netzwerk für längere Zeit getrennt sein?*
- *Gibt es mehrere Schlüsselstandorte in der Welt wo das Modell geshared und bearbeitet wird?*

Antworten auf diese Fragen bestimmen wie das Enterprise Architect Modell selbst für Sharing zur Verfügung steht[1] und somit wie Versionskontrolle angewendet wird. In den folgenden Abschnitten beschreiben wir wie Versionskontrolle in häufig auftretende Szenarien angewendet werden soll. Nachstehend sind die Szenarien, die wir im Detail betrachten werden, aufgelistet:

1. **Zentralisiertes Team:** Alle Editoren sind über ein Hochgeschwindigkeits-Netzwerk verbunden und können somit an dem gleichen physikalischen Modell in einem Datenbank-Management-System (DBMS) beteiligt sein.

2. **Verteiltes Team:** Vielleicht müssen sie ihren Beitrag am Modell offline leisten und daher eine lokale Kopie des Modells auf der eigenen Maschine erstellen.

3. **Mehrfache Standorte:** Mehrere geografisch verteilte Standorte arbeiten an demselben Modell. Es gibt keine High-Speed-Verbindung zwischen Standorten. Teams an jedem Standort haben gemeinsamen Zugriff auf eine lokale Kopie des Modells.

N.B.: Künftige Versionen dieses Whitepapers werden sich auch mit dem Konzept der Verteilung von "Modell-Bibliotheken" auseinandersetzen: das sind Modelle, die als eigenständige, projektübergreifende Komponenten dienen und von einer beliebigen Anzahl von "Verbrauchern" wiederverwendet werden können.

---

1 Das Whitepaper „Unternehmensweiter Einsatz von Enterprise Architect“ zeigt wie das Modellierungs-Tool in verschiedenen Team-Szenarien einzusetzen ist.

Szenario 1: Zentralisiertes Team

In diesem Szenario sind alle Teammitglieder über einen High-Speed-Netzwerk miteinander verbunden. Sollten mehr als 5 Teammitglieder gleichzeitig Zugriff auf das Modell haben, können wir nicht mehr sicher auf eine einzige EAP-Datei, die sich auf einem Netzlaufwerk befindet, zugreifen. Stattdessen wird empfohlen, mit einen dedizierten DBMS unser Modell zu hosten. Der Vorteil der Verwendung einer gemeinsamen DBMS in dieser Situation (im Gegensatz zu lokalen Kopien des Modells) ist, dass alle Teammitglieder das aktuelle Modell sehen und bearbeiten, ohne eine separate "Get Latest" Synchronisation durchführen zu müssen.

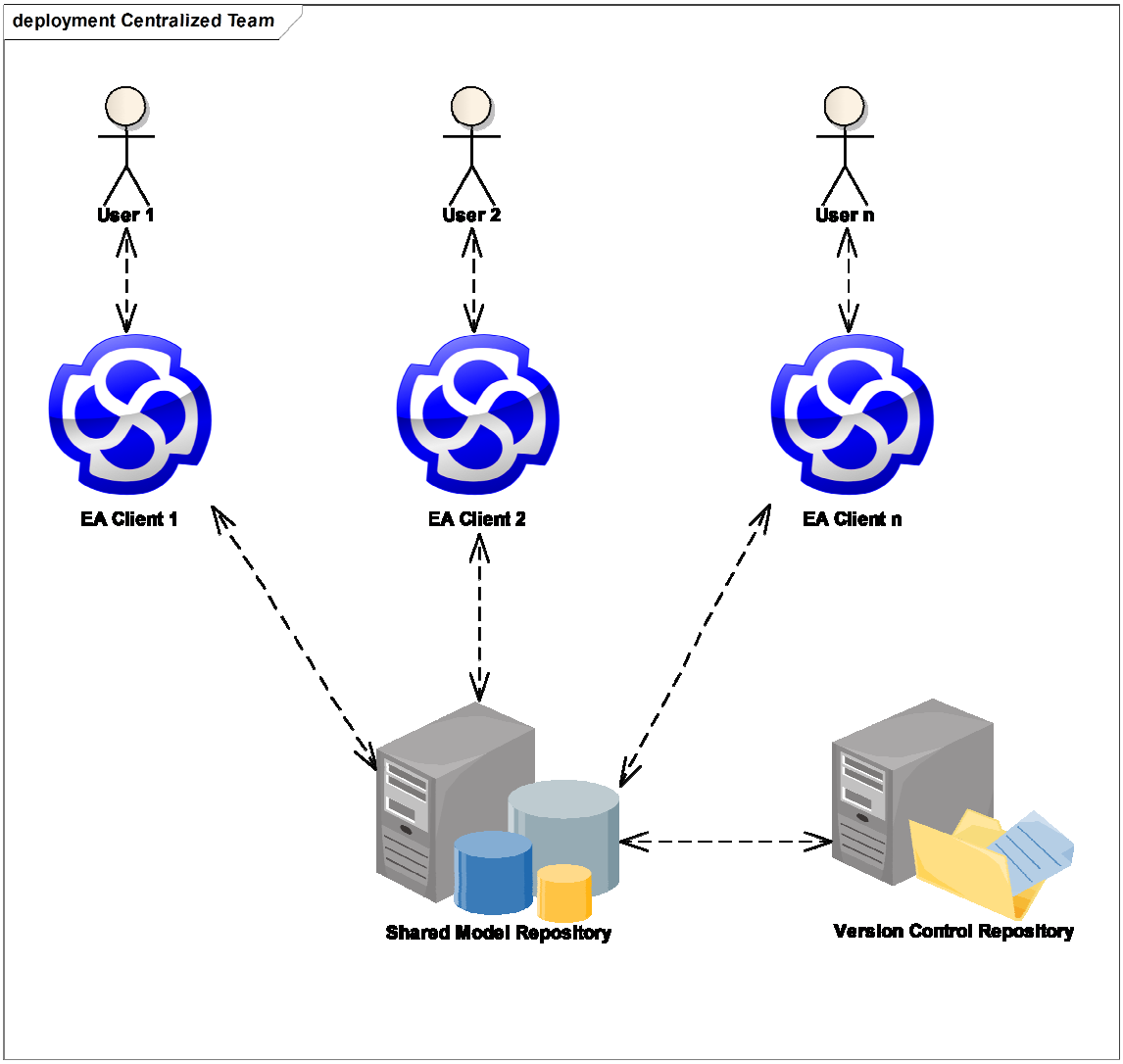

Abbildung 3: ein zentrales Team linkt zu ein gemeinsames DBMS, um das Modell zu bearbeiten.

Deshalb ist die Rolle des Version Control Repositories in dieser Situation nicht einen Mechanismus um die Verteilung der Modelldaten zu ermöglichen – dies ist eine Funktion, die bereits durch das DBMS erreicht wird. Stattdessen hilft das Verson Control Repository Revisionen zu verwalten und das Roll-back von Änderungen zu ermöglichen.

Im Folgenden sind einige Fragen aufgelistet, die häufig bei der Verwaltung von Versionskontrolle in einem solchen Szenario gestellt werden:

- Bei welcher Granularität soll ich Version Control anwenden - Model-Ebene, Paket- oder Unterpaket-Ebene, etc.?
- Wie kann ich verhindern, dass ein Teammitglied die Arbeit eines anderen überschreibt?
- Wenn jemand einen Fehler macht, wie kann der vorherige Zustand des Pakets sicher wiederhergestellt werden?

Antworten auf diese Fragen werden in den folgenden empfohlenen Prozessen und Best Practices zur Verfügung gestellt.

Empfohlener Vorgang für Version Control Modelle

1. DBMS Repository einrichten:
(a) Einrichtung eines dedizierten DBMS-Server der von Enterprise Architect unterstützt wird und für alle Teammitglieder zugänglich ist.
(b) Führen Sie die SQL-Skripte für das Enterprise Architect Datenbankschema aus und übertragen Sie dann alle vorhandenen Starter-Modelle. (Ausführliche Anweisungen zur Verwendung der Skripte sind über den obigen Link verfügbar).
(c) Wahlweise kann User Security auf das Enterprise Architect Modell aktiviert werden. So können Sie kontrollieren wer Zugriff auf bestimmte Funktionen von Enterprise Architect in diesem Modell hat.

2. Version Control Repository einrichten:
(a) Installieren Sie die Server-Komponente Ihres bevorzugten Version Control Anwendung und stellen Sie sicher, dass Team-Mitglieder die entsprechende Client-Software installiert haben.
(b) Erstellen Sie eine Version Control Repository mit einem leeren Projekt das für Enterprise Architect verwendet wird.

3. Version Control Clients einrichten:
(a) Verwenden Sie einen Client-Rechner um eine Arbeitskopie des leeren Projekts auszuchecken und in einen lokalen Ordner abzulegen.
(b) In Enterprise Architect definieren Sie eine Version Control Konfiguration, die auf Dateien der Arbeitskopie zugreift. Der Prozess eine Version Control Konfiguration zu definieren variiert je nach Ihrem Version Control System. Jeder Benutzer, der später auf das DBMS Modell-Repository zum ersten Mal zugreift, wird aufgefordert seine lokale Version Control-Einstellungen zu konfigurieren. Sie werden die Konfiguration, die Sie für dieses Modell erstellt haben, wiederverwenden und den Pfad zu ihrer eigenen Arbeitskopie-Dateien und den Version Control Client als ausführbar festlegen.

4. Pakete unter kontrolle setzten:
(a) Versionskontrolle für einzelne Pakete in Enterprise Architect anwenden. Siehe Anhang C für weitere Details.

Nach Umsetzung der genannten Setups werden Nutzer Pakete in Enterprise Architect unter ihrem Version Control User Account auschecken. Der Checkout-Prozess innerhalb von Enterprise Architect sperrt das Paket für exklusive Bearbeitung durch den Benutzer.

Hier sind “Get All Latest” oder “Get Latest” grundsätzlich nicht notwendig denn das Modell-Repository enthält immer die neuesten Modell Informationen. (Die gespeicherten Revisionen der Version Control Repository stammen aus dem Modell-Repository.)

Best Practice 1: In einem Centralized Team Modell wenden Sie Version Control auf alle Pakete in der Modell-Hierarchie an die sub-Pakete und Root-Knoten enthalten um das Potenzial für paralleles Arbeiten zu maximieren.

Best Practice 2: Verwenden Sie Enterprise Architects rollenbasierte (Benutzer-) Sicherheit um Funktionalität durch Benutzer- und Gruppenberechtigungen zu beschränken und Workflow-Scripting zu ermöglichen. Beispielsweise können Sie eine Security-Gruppe anlegen die über die Berechtigung zur Verwaltung versionskontroller Pakete verfügt und Berechtigungen wie "Configure Version Control" und "Configure Packages" aktivieren.

Durch die gezieltes Hinzufügene administrativer Benutzer zu einer Gruppe können Sie effektiver steuern welche Pakete nach Ihren Vorgaben zur Version Control hinzugefügt (oder entfernt) werden. Darüber hinaus können Sie bestimmen welche Benutzer Pakete auschecken können indem Sie das "Use Version Control" Recht aufheben, damit Benutzer nur mehr Lese-Zugriff auf versionskontrollierte Pakete haben. Sie können auch Zugriffsrechte Ihres Version Control Systems auf bestimmte Ordner einschränken, um Lese-/Schreibzugriff der in diesen Ordnern gespeicherten Paketen zu beschränken.

N.B.: Es wird nicht empfohlen die rollenbasierte Sicherheit zu verwenden um Benutzer-und Gruppensperren auf versionskontrollierte Pakete anzuwenden.

Best Practice 3: Es ist sinnvoll regelmäßig das gesamte Modell zu archivieren - als Backup und für Offlinearbeiten. In Enterprise Architect können Sie ein DBMS Projekt ins EAP-Format übertragen (nutzen Sie die DBMS-to-.EAP Option). Der Modell-Transfer soll zusätzlich zu Ihrem Standard-DBMS Backup-Verfahren durchgeführt werden und nicht als Ersatz.

Best Practice 4: Stets aussagekräftige Kommentare beim Package Check-In (Enterprise Architect fordert Sie auf einen Kommentar über Ihr Version Control System Check-In abzugeben.) Ein Kommentar der sinnvoll die Art der Änderungen beschreibt kann später bei Nachprüfen sehr hilfreich sein. Solche Kommentare sind auch nützliche

Indikatoren des letzten "guten" Zustandes, falls das Paket, aufgrund eines aufgetretenen Fehlers, zu einem früheren Status wiederhergestellt werden muss.

Empfohlene Vorgehensweise für Änderungen zurückzusetzen (Rollback)

1. Finden Sie eine frühere Revision - der letzte "gute" Zustand - des Pakets aus der Versionskontrolle. Verwenden Sie die Package File History in Enterprise Architect um die jeweilige Revision auszuwählen um sie dann in das Modell zu importieren.
2. Erstellen Sie eine Baseline des abgerufenen Pakets.
3. Das Paket in das Modell auschecken. Somit wird das Paket zum Bearbeitung entsperrt - die "Head Revision" des Pakets aus der Versionskontrolle wird damit jedoch abgerufen und importiert.
4. Mit der zuvor erstellten Baseline "Restore to Baseline" ausführen.
5. Das (wiederhergestellte) Paket einchecken.
6. Nach Bedarf können Sie die Basline von Schritt 2 löschen.

Szenario 2: Verteiltes Team und lokale Modelle

In diesem Szenario bearbeiten mehrere Personen das Modell ohne jedoch über ein gemeinsames Model Repository verbunden zu sein. Stattdessen verwaltet jeder Editor ein lokale Kopie des Modells als eine EAP-Datei und aktualisiert regelmäßig ihre Kopie von einem gemeinsamen Version Control Repository. Dieser Ansatz erleichtert umfassende Vervielfältigungen des Modells ohne die Notwendigkeit der Verwaltung eines DBMS.

Empfohlene Verfahren zur Versionskontrolle von Modellen

1. Einrichten eines ersten versionskontrollierten Enterprise Architect Modells:

a) Erstellen Sie ein Version Control Repository.
b) Erstellen Sie einen Eintrag in Ihrem Version Control Repository mit Ihrem Version Control System das zum Speichern Ihrer Enterprise Architect-Paket-Dateien verwendet wird.
c) Erstellen Sie eine Arbeitskopie in einem lokalen Ordner (beispielsweise in CVS und Subversion das Auschecken des im vorherigen Schritts erstellten Eintrag erforderlich).
d) Ein bestimmter Benutzer soll eine Enterprise Architect Projekt (EAP) Datei erstellen und eine Versionskontroll Konfiguration definieren. Die Konfiguration ermöglicht den Zugriff auf die Dateien der Arbeitskopie innerhalb von Enterprise Architect.
e) Wenden Sie die Versionskontrolle auf einzelne Pakete an innerhalb von Enterprise Architect an. Siehe Appendix C für Details.

1. Verteilen Sie das Modell an den Rest des Teams:

a) Sobald das Modell mit angewendeter Version Control erstellt ist, verteilen Sie die EAP-Datei auf den Rest des Teams.
b) Jedes Teammitglied das auf versionskontrollierte Pakete von Enterprise Architect zugreifen soll, muss eine Arbeitskopie wie in Schritt 1c erstellen.
c) Wenn Teammitglieder das Modell zum ersten Mal öffnen, werden sie von Enterprise Architect aufgefordert für alle Version Control Konfigurationen die von dem Modell verwendet werden, eine Definition festzulegen. Geben Sie einfach den Pfad zu den lokalen Arbeitskopie-Dateien und speichern die Definition.
d) Das Modell wird nun an Version Control angeschlossen und einsatzbereit.

Best Practice 5: Wie in einem zentralisierten Team können Sie Parallelarbeiten maximieren indem Sie Versionskontrollen auf alle Pakete in der Modell-Hierarchie anwenden (einschließlich sub-Pakete und Root-Knoten). Auch wenn Sie Cross-Package Abhängigkeiten sorgfältig verwaltet haben (beschrieben im folgenden Abschnitt) ist dies weiterhin angemessen. In einer verteilten Umgebung hat dieser Ansatz einen zusätzlichen Leistungsvorteil, da beim Abgeben und Abrufen der aktuellsten Änderungen jeder Benutzer kleine XMI-Dateien zwischen der Version Control und den Modell Repositories übertragen wird.

Die Bearbeitung kann jedoch durch eine Verringerung der Abhängigkeiten zwischen versionskontrollierten Paketen vereinfacht werden. Um dies zu erreichen müssen Sie bestimmen ob Versionskontrolle nicht in den unteren Ebenen der Modell-Hierarchie anzuwenden ist. Dies ist ein Kompromiss zwischen Verminderung von potenziell fehlenden Cross-Package Abhängigkeiten, Leistung und parallelem Arbeiten.

Best Practice 6: Definieren Sie ein Teammitglied als "Model Manager ', der für die Aufrechterhaltung einer "Master" EAP-Datei verantwortlich ist. Die EAP-Datei wird nicht für die tagtägliche Arbeit eingesetzt werden. Vielmehr ist es ihr Zweck, ein Ausgangspunkt für neue Team-Mitglieder zu sein. Die Kopie des Modells, welches ein neues Team-Mitglied erhält, soll keine Pakete enthalten die bereits von einem anderen Benutzer als ausgecheckt markiert sind. Die "Master" EAP-Datei sollte aus dem Repository der Versionskontrolle mit "Get All Latest" in Enterprise Architect aktualisiert werden. Alle neu erstellten Top-Level-Pakete müssen zu dem Modell mit "Get Package" hinzugefügt werden. Definieren Sie eine Vorgehensweise um Model Manager auf neue Pakete hinzuweisen die zur Versionskontrolle hinzugefügt wurden.

Sie können eine konsistente Schreibweise von Konfigurations IDs auf jeder Kopie erzielen, indem Sie mehrere Kopien einer "Master" EAP-Datei verwenden und es Enterprise Architect ermöglichen den Benutzer zur Eingabe der Definitionen der erforderlichen Version Control Konfigurationen aufzufordern.

Best Practice 7: In einem verteilten Teamumfeld in dem Mitglieder lokale EAP-Dateien bearbeiten, gibt es keine automatische Aktualisierung der rollenbasierten Sicherheitsinformationen von Enterprise Architect, da sie nicht im Repository der Versionskontrolle gespeichert wird. Daher ist es nicht empfehlenswert, rollenbasierte Sicherheit in diesem Szenario anzuwenden.

Cross-Package Abhängigkeiten verwalten

Da Pakete unabhängig voneinander gesteuert werden können, ist es möglich ein Modell (absichtlich oder unabsichtlich) zu "splitten", sodass ein Element an einem Ende einer Beziehung nicht in in der Kopie vorhanden ist, welche Sie gerade bearbeiten. Dies kann beabsichtigt werden um den Umfang der Modell-informationen einzuschranken, der von einem Editor des Modells erforderlich ist. Ein solches Szenario kann jedoch zum Verlust der Modellinformationen führen. Betrachten Sie das Beispielmodell in Abbildung 5.

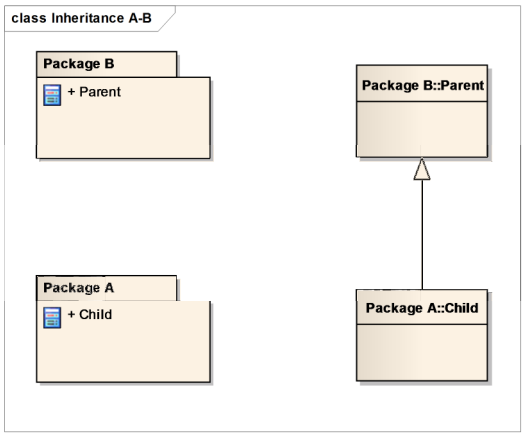

Figure 5: Parent and Child defined in separate Packages

Die Klassen Eltern (Parent) und Kind (Child) werden in abgetrennten, versionskontrollen Paketen definiert und besitzen eine Erbschaftsbeziehung dazwischen. **Kind** ist in **Paket A** definiert, während **Eltern** in **Paket B** definiert ist. Dieses Szenario zeigt ein Beispiel für eine *Cross-Paket Abhängigkeit* zwischen versionskontrollierten Paketen.

Die Modell-Hierarchie ist im Enterprise Architects Project Browser dargestellt (Abbildung 6).

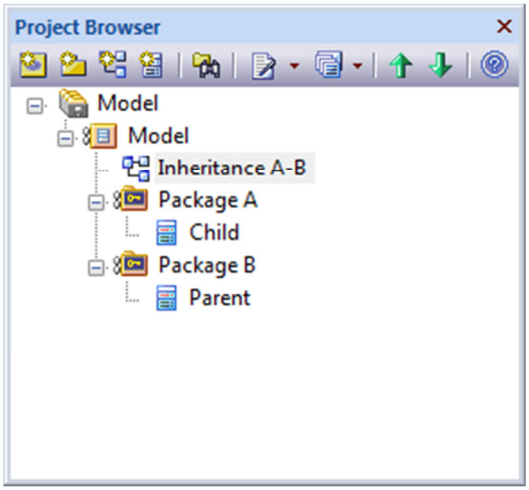

Abbildung 6: In Verbindung stehende Pakete unabhängig voneinander Versionskontrolliert

Wenn beide Pakete im Modell existieren, ist die implizite Abhängigkeit zwischen Paket A und B über die angeschlossenen Elemente gegeben. Umgekehrt, wenn wir nur eines der Pakete in unserem lokalen Modell hätten wäre die Abhängigkeit gebrochen - eines der erforderlichen Elemente würde fehlen. Die unmittelbare Folge wäre, dass alle

auf dieses Verhältnis bezogenen Diagramme unvollständig wären. Weiterhin, falls das Paket eines unvollständigen Modells ausgecheckt und eine neue Revision übergeben wird, dann ist die Beziehung in der resultierenden XMI nicht enthalten, was zu einer semantischen Änderung des Modells führt.

In jedem nicht-trivialen Projekt werden Sie höchstwahrscheinlich Abhängigkeiten zwischen versionskontrollierten Paketen modellieren. Bisher haben wir nur ein Szenario berücktichtigt, wo dies der Fall ist. In den folgenden Abschnitten betrachten wir andere spezifische Modellierungssituationen, die Cross-Paket Abhängigkeiten und empfohlene Prozesse für die sichere Bearbeitung mit einbeziehen.

Best Practice 8: Planen Sie Ihre Paketabhängigkeiten im Voraus und pflegen Sie die bekannte Reihe von Abhängigkeiten. Dies kann mit UML-Paket-Diagrammen erreicht werden und das detaillierte Modellieren von UML Abhängigkeitsbeziehungen dazwischen. Pflegen Sie die Paket-Diagramme in einer Kopie Ihrer "Master" EAP-Datei, wo alle Pakete zur Verfügung stehen. Definieren Sie Modellierungskonventionen und -richtlinien, welche Modellierer von der Einführung ungeplanter Abhängigkeiten abhalten.

Enterprise Architect verfügt über Tools die Ihnen dabei helfen die Beziehung eines bestimmtes Elements zu anderen Elementen im Modell zu identifizieren. Beispiele hierfür sind Traceability Window, Relationships Window und Relationship Matrix. Skripte auf der Sparx Systems Enterprise Architect Community Website, wie das Skript zur Paketabhängigkeit das Ihnen bei der automatischen Erstellung von Diagrammen hilft, können auch hilfreich sein.

Best Practice 9: Arbeiten Sie immer mit dem gesamten Modell. Ein Großteil der Komplexität ,die mit Cross-Paket Abhängigkeiten verbunden ist, kann vermieden werden indem Sie sicherstellen, dass Ihre lokale EAP-Datei eine Kopie des gesamten Modells enthält statt eines Teilmodells. Indem Sie mit einer Kopie der "Master" EAP-Datei beginnen und regelmäßig das "Get All Latest" Kommando ausführen, minimieren Sie das Risiko Änderungen mit fehlenden Abhängigkeiten an das Version Control System zu übergeben. *Es wird empfohlen immer "Get All Latest" vor dem Auschecken eines Pakets auszuführen.*

Empfohlene Verfahren für die sichere Einreichung von Änderungen

Die folgenden Szenarien beschreiben Situationen in denen Modell-Updates mehrere versionskontrollierter Pakete beeinflussen, aufgrund von vorhandenen Abhängigkeiten. Die Abhängigkeit kann explizit sein, wie eine UML-Vererbungsbeziehung, oder implizit, beispielsweise wenn auf einen Classifier aus einem separat gesteuerten Versions Package verwiesen wird, um einen Attribut-Typ anzugeben.

Modellierung von Verbindungen zwischen zwei versionskontrollierten Packages
Angenommen wir haben zwei unabhängig gesteuerte versionskontrollierte Pakete A und B wie in den Abbildungen 5 und 6 modelliert. Nehmen Sie nun an, wir wollen die folgenden Bearbeitungsvorgänge zwischen den Elementen in den beiden Pakete durchführen:

1. Erstellen Sie einen Verbinder aus einem Element in Paket A zu Paket B
2. Ändern Sie das Zielelement des Verbinders auf ein anderes Element
3. Kehren Sie die Verbindungsrichtung um und ändern Sie die Typ- und Modell-Bidirektionalität
4. Den Verbinder löschen

Für jede der oben genannten Modell Bearbeitungen, definieren wir ein empfohlenes Verfahren zur Aktualisierung der entsprechenden Pakete. Mithilfe des empfohlenen Verfahrens wird dafür gesorgt, dass alle anderen Modell-Editoren unsere Updates korrekt empfangen können. Das heißt, die Bearbeitungen die wir in unser lokalen Kopie des Modells sehen, spiegelt sich in den entsprechenden XMI-Dateien wieder, die für die Aktualisierungen von allen anderen Modellen mit dem gleichen Version Control Repository verwendet werden.

1. Erstellen Sie eine Beziehung aus einem Element aus Paket A nach Paket B.
Wir wollen die Situation in Abbildung 7 modellieren. Wir können annehmen, dass die Klasse Child diese Beziehung "besitzt", da die Vererbung die Klasse Parent semantisch nicht verändert. Wir brauchen also nur Paket A auszuchecken um die Änderungen vorzunehmen.

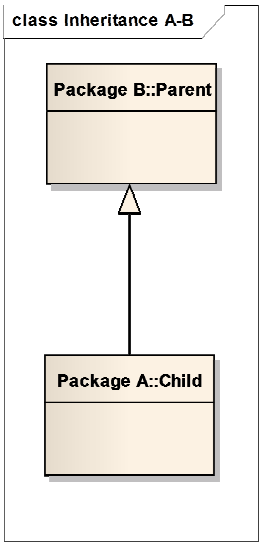

Figure 7: Create Connector

Der Prozess zum Erstellen einer uni-direktionalen Verbindung ist wie folgt:

i. Mit "Get All Latest" stellen Sie sicher, dass Sie über das gesamte Modell verfügen (Best Practice 9)
ii. Package A auschecken
iii. Beziehung erstellen

iv. Package A einchecken

Hinweis: Wenn Sie die Beziehung mit Hilfe eines Diagramms in einem anderen Paket erstellen wollen, muss das Diagramm, zusammen mit Paket A, ausgecheckt und eingeckeckt werden.

2. Ändern Sie das Zielelement des Connectors auf ein anderes Element
Diese Änderung ist im Wesentlichen mit unserer vorherigen Änderung identisch, weil wir weder das ursprüngliche Zielelement noch das neue Zielelement durch Re-Routing des Connectors semantisch verändern. Wir können daher die gleichen Verfahren einsetzen, indem Schritt 3 mir einer Änderung der Zielelement ersetzt wird.

3. Umkehrung der Richtung eines Connectors, Änderungs des Connector Typs und Modell Bidirektionalität
Wenn wir nun das gleiche Modell editieren wollen, um die Veränderungen aus Abbildungen 8-10 zu implementieren, wird eine Änderung herbeigeführt, die die Elemente, sowohl in Paket A als auch B, semantisch verändert. Deshalb müssen wir beide Pakete überprüfen, um die Änderung vorzunehmen und anschließend beide Pakete einchecken um sicher zu gehen, dass die XMI-Dateien in der Version Control Repository unserem aktuellen Modell entsprechen.

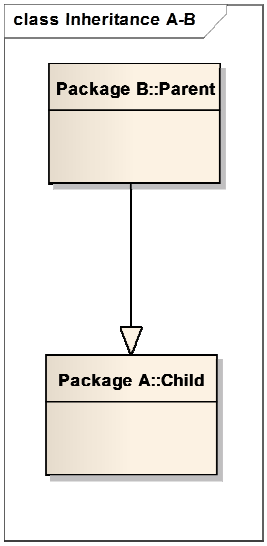

Abbildung 8: Reverse Direction

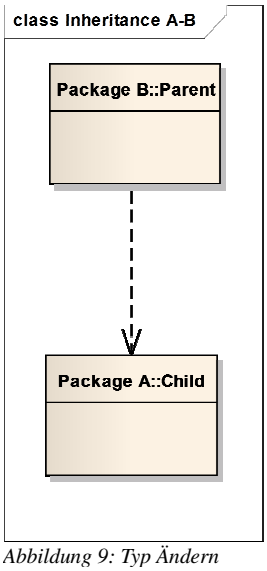

Abbildung 9: Typ Ändern

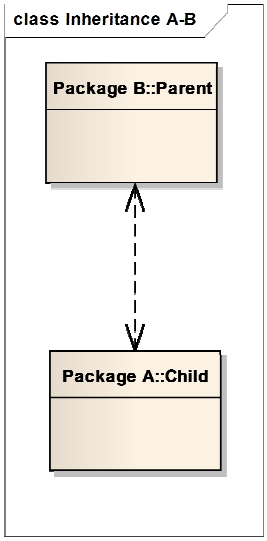

Abbildung 10: Bidirektional machen

Die empfohlene Vorgehensweise zur Durchführung dieser Änderungen ist wie folgt:

i. "Get All Latest" anwenden um sicherzustellen, dass Sie über das gesamte Modell verfügen (Best Practice 9)
ii. Package A und Package B auschecken
iii. Beziehung modifizieren
iv. Package A und Package B einchecken mittels “Check-in Branch” (Best Practice 10)

4. Den Connector löschen
Nehmen wir nun an, wir haben unseren Connector im ursprünglichen Zustand wie in Abbildung 7 modelliert und wir wollen ihn löschen. Derzeit ist es in Enterprise Architect notwendig, dass wir den gleichen Prozess wie oben anwenden – d.h. beide Packages müssen augecheckt werden. Obwohl wir die Parent Klasse semantisch nicht ändern indem wir die Erbschaft Beziehung entfernen, werden Informationen über den Connector in den XMIs von beiden Paketen gespeichert. Um sicherzustellen, dass der Connector nicht zu einem späteren Zeitpunkt über das XMI für Paket B wiederhergestellt wird (z.B., indem "Get Latest" ausführt wird), müssen wir beide Pakete aktualisieren.

Best Practice 10: 'Atomic Commits'. Beim Einchecken einer in sich geschlossenen Änderung, dass mehrere Pakete betrifft, verwenden Sie bitte Enterprise Architects "Check-in Branch".Befehl. Dieser Befehl ermöglicht es Ihnen, alle betroffenen Pakete gleichzeitig zu übergeben - damit wird verhindert, dass andere Editoren nur einen Teil Ihrer Aktualisierung auschecken und somit mögliche Verluste. Es ermöglicht Ihnen auch den gleichen Check-in-Kommentar für alle Pakete und das Change Set logisch zu gruppieren.

Best Practice 11: Kleine, in sich abgeschlossene Änderungen regelmäßig übergeben. Wenn Sie mehrere Pakete über einen längeren Zeitraum ausgecheckt halten, werden Sie wahrscheinlich zahlreiche unabhängige Änderungen

vornehmen, die Zahl an Cross-Paket Abhängigkeiten die durch die Änderungen betroffen sind erhöhen und die mit Rollback Veränderungen verbundene Komplexität steigern.

Modeling Classifier Referenzen zwischen zwei versionskontrollierten Pakete
Gehen wir wieder davon aus, dass wir über zwei unabhängig versionskontrollierte Pakete A und B verfügen. Angenommen, Paket A hat ein Element X und Paket B enthält ein Element Y. Nun wollen wir ein Referenzmodell von Element X auf Element Y als Classifier erstellen. Einige Modellierung-Beispiele in diesem Fall sind:

- Festlegung der Art des Attribut im Element X
- Angeben des Rückgabetyps oder Parameter Typen von einer Operation im Element X
- Angabe der Classifier für Element X, wobei X eine Instanz (oder UML-Objekt) ist

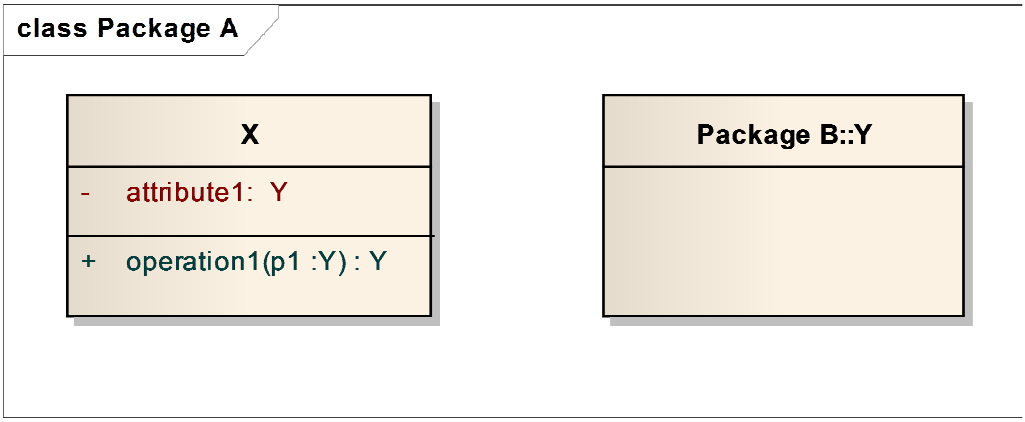

Abbildung 11: Class X bezieht sich auf Class Y als Classifier für Attribute und Operationsparametertypen. Class Y ist in einem separaten versionkontrollierte Paket, Package B.

In solchen Situationen modellieren wir wieder eine Cross-Package Abhängigkeit obwohl keine explizite Beziehung zwischen den Elementen gezogen wird. Die empfohlene Vorgehensweise für die Erstellung oder Aktualisierung dieser "impliziten" Abhängigkeiten ist:

i. "Get All Latest" anwenden um sicherzustellen, dass Sie über das gesamte Modell verfügen (Best Practice 9)
v. Package A und Package B auschecken
vi. Hinzufügen, Aktualisieren oder Löschen der Classifier Referenz(en)
vii. Package A und Package B einchecken mittels “Check-in Branch” (Best Practice 10)

Da wir Paket B oder deren Elemente nicht durch die Einstellung von Classifier Referenzen in einem anderen Package semantisch ändern, ist die folgende Frage sinnvoll: Wieso soll Package B überhaupt ausgecheckt werden? Kurz gesagt, die XMI für beide Pakete enthält die Classifier Referenzinformationen. Wenn wir die XMI nur von einem Pakt aktualisieren würden (über die Check-out/Check-in Prozess), gäbe es widersprüchliche Informationen zwischen den jeweiligen XMI-Dateien. Beim anschließenden Ausführung eines "Get All Latest"-Befehls auf ein Paket dass als Zweites aktualisiert wird, werden Classifier Informationen im ersten Paket überschrieben.

Es ist jedoch nützlich, die Classifier Informationen in beiden XMI-Dateien zu haben. Betrachten wir zum Beispiel das Bestücken eines Modells aus dem Nichts. Wenn wir Paket A erst in ein leeres Modell importieren, Paket B (sowie sein Classifier Y) existiert noch nicht und daher würde eine Referenz von Element X auf Y verloren gehen. Nur durch die Einbeziehung der abhängigen Verweise auf Y in die XMI von Paket B, lösen wir solche "Dangling References" beim späteren Import von Paket B.

Hinweis: Bei künftigen Versionen von Enterprise Architect wird das Auschecken des "abhängigen" Pakets (Paket A) und des "Ziel"-Pakets (Paket B), durch erneutes Scanning der richtigen XMI-Dateien und einem 'Get All Latest', vermieden werden.

Verschieben eines Elements zwischen zwei Versionskontrollierte Pakete
Nun nehmen wir an, wir müssen Element X aus Paket A in Paket B hineintauschen. In diesem Fall ist klar, dass wir das semantische Modell beider Pakete beeinflussen. Der Aktualisierungsvorgang läuft wie folgt:

i. "Get All Latest" anwenden um sicherzustellen, dass Sie über das gesamte Modell verfügen (Best Practice 9)

ii. Package A und Package B auschecken (von Enterprise Architect für diesem Zug erforderlich)
iii. Element X von Package A in Package B schieben
iv. Package A und Package B einchecken mittels “Check-in Branch” (Best Practice 10)

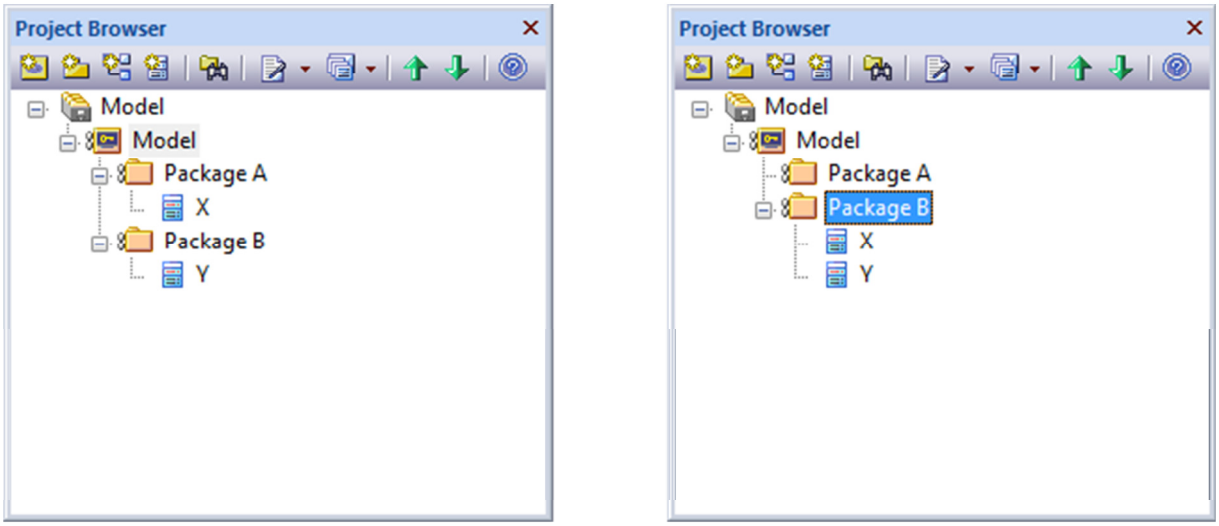

Abbildung 12: Das Verschieben eines Elements ändert semantisch beide Pakete. Wir müssen daher beide Pakete auschecken.

Ein Hinweis zu Sequenz- und Kommunikations-Diagrammen
Bei der Erstellung von Sequenz-Modellen ist es üblich die Classifier, wie Elemente in der Klasse "Domain Model" oder Akteure im "Use Case"-Modell, in verschiedene Pakete aus Sequenz-Diagrammen, die diese Elemente nutzen, zu trennen. Dies ist sinnvoll, da es eine bessere Organisation des Modells ermöglicht. Ein Beispiel für eine Modell-Hierarchie ist in Abbildung 13 dargestellt.

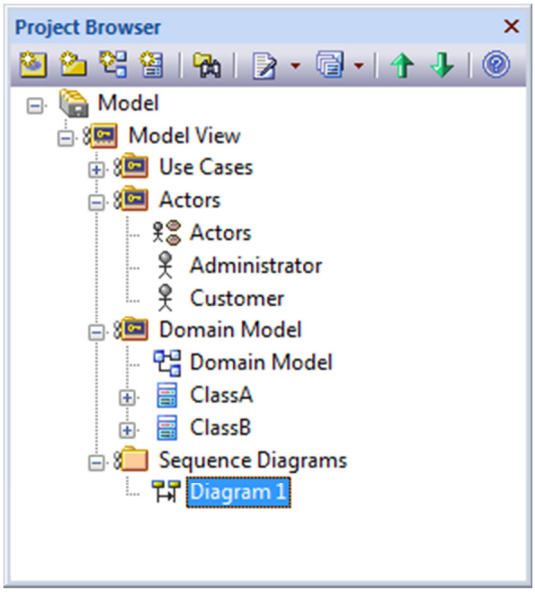

Abbildung 13: Anwendung des 'Sequenz-Diagram' Pakets

Bei der Verwendung von Classifiern aus diesem externen Pakete ist es jedoch am besten Instanzen auf Sequenz-Diagrammen zu erstellen. Dies ist aus UML-Modellierung Perspektive semantisch korrekt und verhindert auch mögliche Verluste von Informationen über die Anschluss-Diagramme beim Roundtrip von Paketen über version control Einchecken/Auschecken. Sequenz-und Kommunikations-Diagramm-Nachrichten werden nur in die XMI des Pakets exportiert, welches das Diagramm enthält. Sind Ihre Instanz Elemente und Diagramme in dem gleichen Paket, werden alle Connector Informationen während des späteren Imports erhalten. Abbildung 14 zeigt den empfohlenen Modellierungsansatz für ein Sequenz-Diagramm das aus dem oben genannten Modell gebaut wurde.

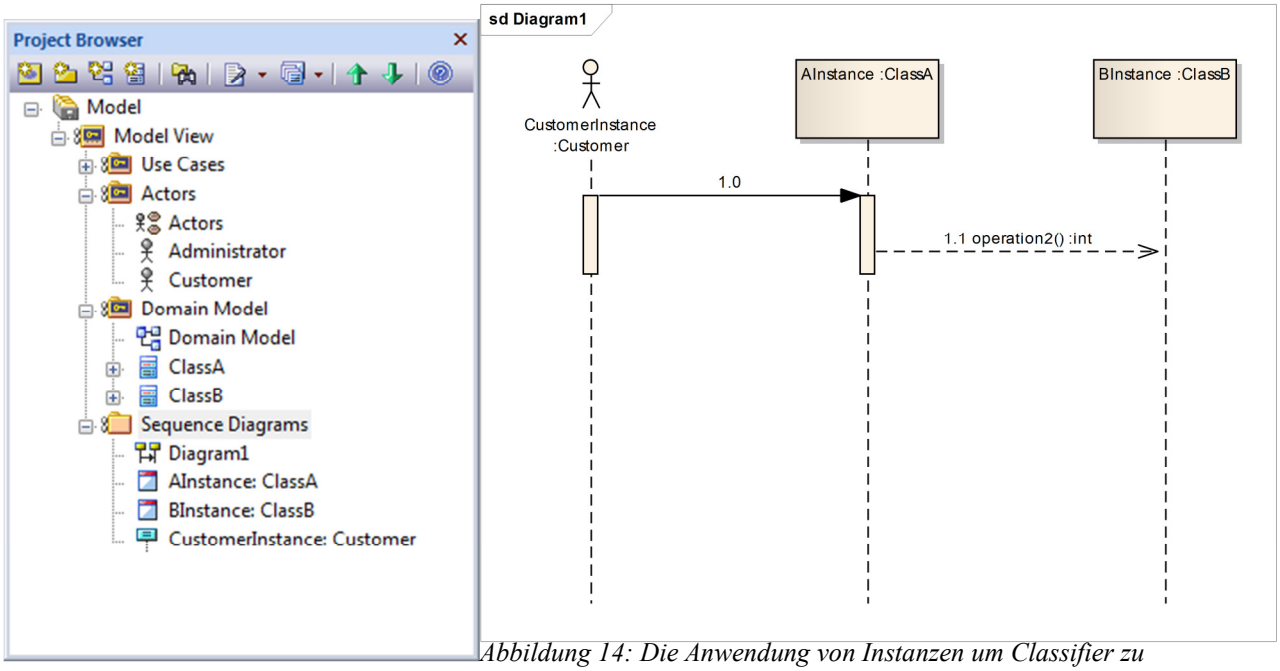

Abbildung 14: Die Anwendung von Instanzen um Classifier zu referenzieren ist semantisch korrekt und hilft Modellintegrität zu bewahren

Best Practice 12: Bei der Erstellung von Sequenz- und Kommunikation- Diagrammen sollten Sie am besten die Instanzen einsetzen, die auf Classifier verweisen. (Verwenden Sie keine Classifier-Elemente direkt.) Dies ist semantisch korrekte und, indem Elemente und Diagramme im gleichen Paket gehalten werden, es wird sichergestellt, dass Nachrichten während des Ein- und Auschecken bewahrt werden.

Empfohlene Verfahren für Rollback (zurücksetzen) von Änderungen

Rollback von Änderungen (oder Fehler zurücksetzen) erfordert das gleiche Verfahren das für die zentralisieten Teams beschrieben wurde.

Nachstehend sind alle Best Practices in diesem Abschnitt in Kurzform aufgelistet:

Best Practice 5: Versionskontrolle auf Low-Level-Pakete anwenden, um das Potenzial für paralleles Arbeiten zu steigern. Gleichen Sie dies mit der entsprechenden Potenzialerhöhung für Abhängigkeiten zwischen versionskontrollierten Paketen aus.

Best Practice 6: Einen "Model Manager" für die Aufrechterhaltung einer 'Master' EAP-Datei beauftragen.

Best Practice 7: Die rollenbasierte Sicherheit ist nicht anzuwenden bei der Verwendung von lokalen EAP -Dateien nicht verwenden um das Modell zu replizieren,.

Best Practice 8: Planen Sie Ihre Paketabhängigkeiten im Voraus und pflegen Sie die bekannten Abhängigkeiten.

Best Practice 9: Immer mit dem gesamten Modell arbeiten. "Get All Latest" vor dem Auschecken jedes Pakets anwenden.

Best Practice 10: 'Atomic Commits'. Verwenden Sie 'Check-in Branch' beim Einchecken einer Änderung die mehrere Pakete betrifft .

Best Practice 11: Kleine, in sich geschlossene Änderungen regelmäßig übergeben.

Best Practice 12: Verwenden Sie Instanzen die auf Classifier verweisen bei Sequenz- und Kommunikations-Diagrammen. Somit werden Instanzen und Diagramme im gleichen Paket beihalten.

Szenario 3: Mehrfache Standorte

Immer häufiger teilen große Konzerne Modell-Informationen über geografisch verteilte Entwicklungsstandorte. Die Herausforderung besteht darin, jeden Standort mit den aktuellsten Modellinformationen auf dem neuesten Stand zu halten. Während DBMS-Ebenen Replikation zwischen Standorten möglicherweise mit Synchronisations-Tools möglich wäre, bietet der Einsatz von versionskontrollierten Paketen eine einfache und effektive Alternative. Die Situation stellt eine Kombination aus den Szenarien 1 und 2 dar. Jeder Standort kann von einem Modell-Repository profitieren, welches auf einem lokalen DBMS gehostet wird oder es können EAP-Dateien von den einzelnen Editoren verwendet werden, wie in Abbildung 15 dargestellt.

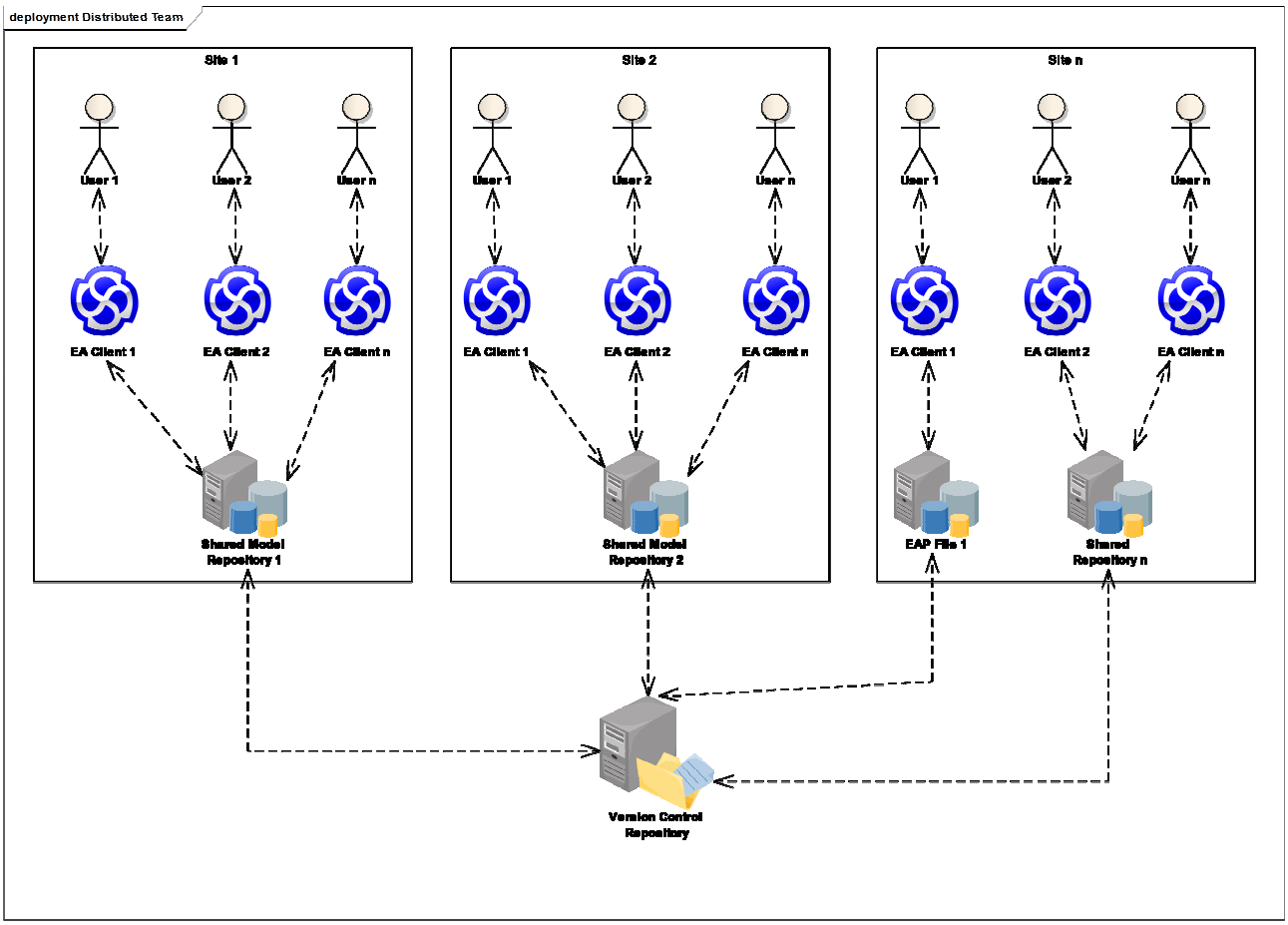

Figure 15: Das Version Control Repository kann die Replikation von Modellen über mehrere Entwicklungsstandorte ermöglichen.

Wenn an mehreren Orten agiert wird, übernimmt jedes gemeinsame Modell-Repository (wie in Szenario 1 beschrieben) eine Rolle, ähnlich wie bei einem lokalen Repository (wie in Szenario 2 beschrieben). Das gleiche Verfahren (wie in Szenarien 1 und 2 beschrieben) gilt für die Einrichtung von Modell-Repositories in diesem Szenario für gemeinsame (DBMS) Modelle bzw. lokale EAP Dateien. Die Verwaltung von Cross-Package-Abhängigkeiten folgt auch den Best Practices wie in Szenario 2 beschrieben.

Berücksichtigt werden muss auch die Verwaltung von Enterprise Architects Reference Data, so dass gemeinsame Projektdefinitionen über jeden Standort geteilt werden können. Eine nähere Beschreibung von Reference Data finden Sie in Anhang A.

Enterprise Architect bietet einen bequemen Mechanismus für die Übertragung des gesamten Modell-Repositories zwischen Standorten (einschließlich Reference Data) über die "Projekt Transfer"-Funktion. Zum Beispiel können Sie das Modell, das im DBMS am Standort 1 erstellt wurde, in eine EAP-Datei und diese an andere Standorte verteilen. Diese Standorte können dann mit der Projekt-Transfer-Funktion eines leeren DBMS-Repositories durch die Übertragung von der EAP-Datei bestückt werden. Das Enterprise Architect Handbuch beschreibt in Detail, wie die Projekt-Transfer-Funktion verwendet wird.

Anhang A: Enterprise Architect Meta-Daten, die nicht in der Version Control Repository gespeichert sind.

Hier setzen wir uns mit Versionsverwaltung von Modell-Paketen und den dazugehörigen Daten auseinander. Es sollte beachtet werden, dass Enterprise Architect Projekte (ob EAP-Datei oder DBMS-Repository) zusätzliche Meta-Daten enthalten, bekannt als Referenzdaten. Diese Daten können verwendet aber nicht direkt innerhalb eines Pakets definiert werden. Beispiele für Enterprise Architects Reference Data sind: Vorlagen für das Generieren von Code und Rich-Text Format (RTF) Dokumentation, Element-Status-Types und Stereotyp-Definitionen.

Wenn solche Meta-Daten verwendet werden, um die Eigenschaften eines Pakets oder seiner Bestandteile zu definieren, werden diese Daten exportiert und zusammen mit dem XMI importiert. Somit bleiben die Daten während einer Paket Aktualisierung mit Versionskontrolle erhalten.

Es kann aber auch vom Nutzen sein die Referenzdaten in der Versionsverwaltung bereitzustellen. Dies ermöglicht es allen zugehörigen Enterprise Architect Modell-Repositories die gleichen Definitionen und Vorlagen wirksam einzusetzen. Sie können Enterprise Architect's integrierte Tools nutzen um Reference Data zu exportieren. Anschließend können Sie die resultierende Datei zu Ihrem Version Control Repository legen. Andere Benutzer können anschließend die Referenzdaten abrufen und sie in ihre Projekte zu importieren. Die Export und Import Prozesse werden im Enterprise Architect Handbuch definiert.

Anhang B: Built-In-Collaboration und Change-Management-Tools

Wir haben in erster Linie an die Verwendung eines Version-Control-Systems von einem Drittanbieter konzentriert, um Informationen für verteilte Teams zu replizieren und Model-Revisionen zu verwalten. Es gibt zahlreiche ergänzende Tools von Enterprise Architect, die einem großen Team das Modellieren erleichtert und ohne fremdes System verwendet werden kann. Einige werden unten beschrieben:

Audit

Diese Fähigkeit ist besonders nützlich wenn mehrere Teammitglieder das gleiche Modell-Repository teilen. Folgende Fragen hilft es zu beantworten: *Wer* änderte *welche* Teile des Modells? *Wann* wurde die Änderung vorgenommen? Was war der *vorherige Zustand*?

Audit-Informationen werden direkt in der Enterprise Architect Model Repository gespeichert und nicht in der Version Control Repository. Die Auditing Fähigkeit bietet ein fortlaufendes Protokoll der Änderungen statt eines Point-in-Time-Snapshots. Audit-Protokolle können in eine Datei exportiert werden und sofortige Änderungen von Protokollen können direkt in Enterprise Architect verglichen werden. Das Enterprise Architect-Benutzerhandbuch bietet weitere Informationen über Auditing.

Baseline Vergleichen und Zusammenführen

Enterprise Architect kann Paket-Versionen direkt in das Modell-Repository als Baselines speichern. Damit können Sie ein Paket mit einem früheren Zustand vergleichen und unerwünschte Änderungen rückgängig machen. Da die Baseline im XMI Format existiert, können Sie auch ein Paket mit jeder XMI-Datei vergleichen, die zuvor aus diesem Paket exportiert wurde. Dies ermöglicht Ihnen gezielt Änderungen, welche von Kollegen auf lokalen Model-Kopien durchgeführt wurden, zusammenzuführen ohne ein Version Control System zu verwenden. Das Enterprise Architect Benutzerhandbuch enthält weitere Informationen über Baselines, vergleichen und zusammenführen.

Controlled Packages

Packages in Enterprise Architect können als "kontrolliert" markiert werden ohne mit einem separate Version Control System verbunden zu werden. Ein "Controlled Package" wird von Enterprise Architect mit einer entsprechenden XMI-Datei erkannt und kann mit dieser auch synchronisiert werden. Während kontrollierte Pakete, im Gegensatz zu Version-Controlled-Paketen, nur begrenzt über Dateiverwaltungs-Befehle verfügen, können sie auch geladen, gespeichert und konfiguriert werden - und zwar komfortabler als das manuelle Durchführen von XMI Import-/Export-Operationen auf jedem Paket.

Role-Based (User) Security

Enterprise Architect's User Security bietet Editoren einen Mechanismus zum Modell-Log-in, welcher zwei wichtige Funktionen erfüllt. Erstens ermöglicht es Unternehmen die Bearbeitungsfunktionen, die für Benutzer verfügbar sind, zu beschränken. Zweitens erlaubt es Pakete und Elemente pro benutzer -und gruppespezifisch zu sperren. Wenn Versionskontrolle in einem Modell verwendet wird ist nur die erste Funktion der User Security ist anwendbar; d.h. Die Beschränkung der Verfügbarkeit von Bearbeitungsfunktionen. Sollte jedoch Versionskontrolle nicht in einem gemeinsamen Modell verwendet werden, spielt User Security bei der Unterstützung kooperativer Modellierungstools eine wichtige Rolle. Durch die Anwendung von Sicherheitssperren können Teammitglieder das gegenseitige Überschreiben von Änderungen anderer vermeiden, sowie ungewollte Änderungen durch Benutzer verhindern, die nicht

als Modell Autoren benannt sind.

Anhang C: Anwendung von Version Control auf Paketen

Basierend auf Ihrem Einsatzszenario und Modellstrukturen über die Sie bereits verfügen, schlagen wir in diesem Anhang Ansätze für die Anwendung der Versionskontrolle von Paketen vor. Wir verweisen auf einige Enterprise Architect Versionskontroll Features, die ausführlicher in der Bedienungsanleitung zur Konfiguration eines versiongesteuerten Pakets erläutert sind.

Prozesse für rückwirkende Anwendung von Versionskontrolle auf bestehenden Pakete

Wie man jedes Paket im Modell versionskontrolliert:

Vielleicht möchten Sie jedes Paket zur Versionskontrolle hinzufügen, um das Potenzial für die parallele Bearbeitung zu maximieren. Sollte das der Fall sein, verwenden Sie einfach den Enterprise Architect 'Add Branch to Version Control'-Befehl beim Modell-Root-Knoten. In diesem Kontext bezieht sich "branch" auf den Unterbaum, welcher bei dem ausgewählten Paket beginnt – wenn Sie also 'Add Branch to Version Control' auf den Root-Knoten anwenden, dann ist das gesamte Projekt betroffen.

Als Ergebnis wird die Versionskontrolle rekursiv auf alle Pakete und deren Sub-Pakete angewendet. Die entsprechenden XMI-Dateien werden automatisch auf Basis der Paket-GUID benannt, welches spätere Umbenennungen von Paketen überlebt. XMI-Dateien für übergeordnete Pakete enthalten nur "Stub" Informationen für untergeordnete Pakete; dabei wird die Größe einer einzelnen Datei reduziert.

Hinweis: Der "Add Branch to Version Control"-Befehl wird Sie zum "Export as Model Branch" auffordern und es wird empfohlen diese Option auszuwählen. Eine Modell Branch-Datei (*. EAB) gibt Ihnen einen bequemen Bezug auf den Unterbaum den Sie exportieren. Es ist eine kleine Datei die in für Menschen lesbaren Begriffen benannt werden kann (im Gegensatz zu einer GUID). Später, wenn Sie oder ein anderes Teammitglied zu einem Modell-Repository von Grund auf neu füllen muss, können Sie dies einfach über die "Import a Model Branch"-Befehl bewirken.

Wie Versionskontrolle selektiv anzuwenden ist:

Anderenfalls, wenn Sie ein verteiltes Team haben, können Sie Cross-Paket Abhängigkeiten zwischen versionskontrollierten Pakete reduzieren. Dies erfordert, dass Sie nicht unabhängige Versionskontrollen von untergeordnete Pakete durchführen. Stattdessen werden untergeordnete Pakete mit dem übergeordneten Paket der XMI-Datei in der Version Control Repository inkludiert. Der Ablauf ist Folgender:

1. Definieren Sie welche Top-Level-Pakete in sich geschlossene Modell Portionen ausreichend darstellen.
2. Für jedes Paket:

i. Bedienen Sie die rechte Maustaste und wählen Sie Konfigurieren, oder verwenden Sie das Tastaturkürzel "Strg + Alt + P" (nützlich, wenn dieselbe Aktion für mehrere Pakete wiederholt werden muss).

ii. Wählen Sie die entsprechende Versionskontroll-Konfiguration, den Standard XMI Dateiname, wenn nötig, anpassen und lassen Sie die übrigen Optionen als Standard.

Verfahren zum Anwendung von Versionskontrolle auf ein neues, leeres Modell, wenn es gerade gebaut wird.

1. Erstellen Sie eine Paket-Skelett-Struktur für das Modell.
2. Pakete zur Versionskontrolle mit einem der folgenden Ansätze hinzufügen:

a) Verwenden Sie den Befehl "Add Branch to Version Control"-Befehl um die Versionskontrolle auf alle Pakete anzuwenden; oder

b) Versionskontrolle auf einzelne Pakete anwenden:

i. Bedienen Sie die Rechte Maustaste und "Configure" auswählen oder verwenden Sie die Tastatur Short-Cut "Strg + Alt + P".

ii. Wählen Sie die entsprechende Versionskontroll-Konfiguration aus, den Standard XMI Dateiname, wenn nötig, anpassen und lassen Sie die übrigen Optionen als Standard.

3. Wenn ein neues Paket in dem Modell hinzugefügt wird, erscheint der "Neues Paket erstellen"-Dialog und stellt Ihnen die Option zur Verfügung, ob es zur Versionskontrolle hinzugefügt werden soll.

Fall Sie in einer verteilten Team-Umgebung arbeiten, sobald das Modell eingerichtet wird:

1. Alle Pakete einchecken und als "Master"-Modell beiseite legen.
2. Verteilen Sie Kopien dieser "Originalkopie" an die Teammitglieder.