AIA 11g: Neuer Meilenstein für die Integration von Systemlandschaften?

Gregor Bublitz
ec4u expert consulting ag
Karlsruhe

Schlüsselworte:

Oracle Application Integration Architecture 11g, Oracle SOA-Suite 11g, Integration, AIA, SOA, ABCS, EBS, kanonisches Datenmodell, MDS, CAVS

Einleitung

In diesem Vortrag geht es um einen Erfahrungsbericht über die Verwendung, des seit Ende April 2010 verfügbaren Application Integration Architecture 11g (AIA) Release von Oracle auf Basis der SOA Suite 11g/Weblogic 11g und ob damit ein neuer Meilenstein für die Integration von Systemlandschaften gesetzt wurde. Nach einer kurzen Einordnung in die Oracle Fusion Middleware und einem Vergleich zwischen AIA und herkömmlicher Integrationsansätze werden die wesentlichen Bestandteile einer AIA 11g Integration (Project Lifecycle Workbench, Entwicklung im JDeveloper der Enterprise Business Services (EBS) und der Application Business Connector Services (ABCS), Meta Data Store (MDS), Composite Application Validation System (CAVS), Harvesting, Bill of Material, Deployment Plan) vorgestellt und anhand von Projekt- und Praxiserfahrungen beleuchtet. Zum Schluss werden Best Practices und Lessons Learned aufgezeigt, die beim Arbeiten mit AIA 11g zu berücksichtigen sind, sowie in einem Fazit die Fragestellung im Titel beantwortet.

Einordung in Oracle Fusion Middleware

Die Abkürzung AIA steht für Application Integration Architecture und beschreibt die Integration von Anwendungen über einen standardisierten Integration Layer. Im Gegensatz zu den klassischen Punkt zu Punkt-Verbindungen, die früher im Rahmen von EAI (Enterprise Application Integration) state-of-the-art waren, steht AIA für eine entkoppelte, schneller implementierbare und besser zu wartende Möglichkeit Systeme zu verbinden.
Im AIA wird jede Applikation „nur“ an den Integration Layer angedockt und kann dann die in der Mitte zur Verfügung stehenden Services nutzen. Dabei interessiert es nicht, welches Backend-System oder welche andere Komponente diesen Service erbringt, denn dies wird durch die Architektur im Integration Layer verborgen.
Generell basiert AIA, wie in der folgenden Abbildung dargestellt, auf dem Oracle Fusion Middleware Stack bzw. der Oracle SOA Suite.

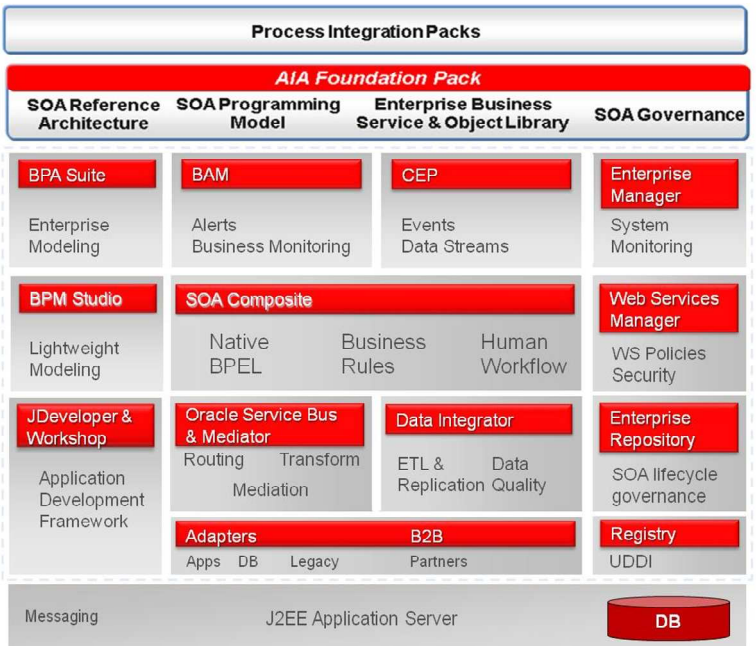

Abb. 1: Oracle Fusion Middleware Stack

Die wichtigsten Elemente der Oracle SOA Suite sind hierbei die Laufzeitkomponenten Mediator und BPEL Process Manager sowie JDeveloper als Entwicklungswerkzeug und Enterprise Manager für das technische Monitoring der Serverlandschaft und des Anwendungsstatus. Für das Real-Time Monitoring von Geschäftsprozessen kann das BAM-Modul (Business Activity Monitoring) eingesetzt werden.
AIA lizenzierbar als AIA Foundation Pack selbst besteht aus einem Referenz- und Programmiermodell für Prozesse, aus kanonischen Geschäftsobjekten und den zugehörigen CRUD-Services sowie aus Werkzeugen und Tools, um den Lebenszyklus der Integrationsszenarien zu verwalten.
Die sogenannten PIPs sind vorgefertigte Prozessabläufe zwischen Oracle und/oder Nicht-Oracle-Anwendungen die out-of-the-box deployed und eingesetzt werden können. Diese wurden mit dem AIA-Paradigma von Oracle entwickelt und werden als Produkt angeboten. Für das aktuelle AIA 11g Release sind die PIPs zurzeit noch nicht verfügbar.

AIA Methodologie

Die wichtigsten Bestandteile einer AIA Integration sind in der Abbildung 2 dargestellt und werden im Folgenden kurz erläutert.

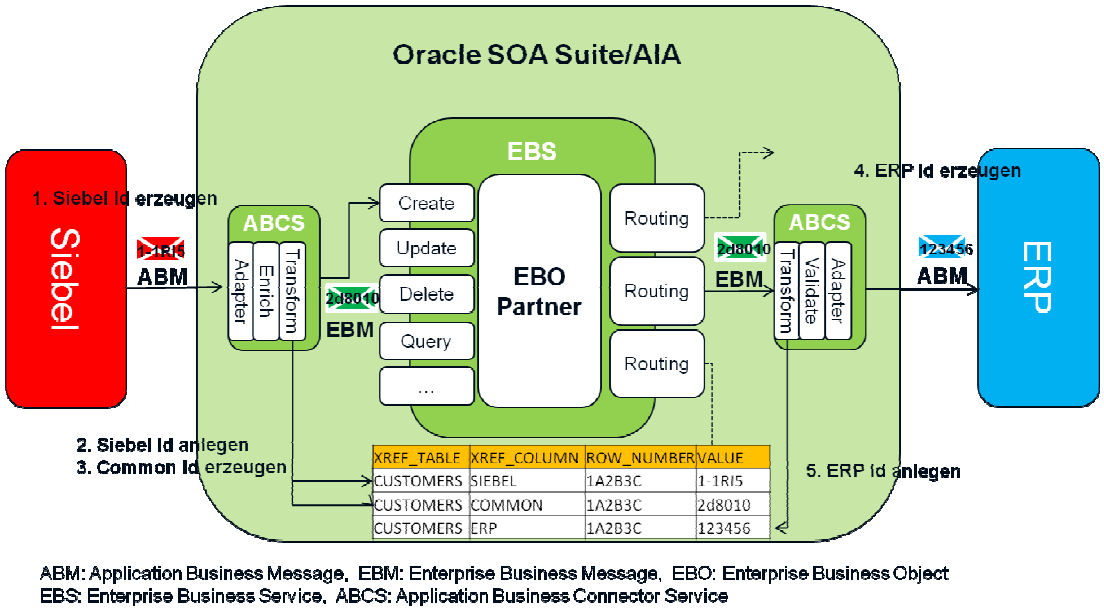

Abb. 2: Schematische Darstellung der Architektur

Aus AIA Sicht wird Siebel in diesem Fall als Requestor Application bezeichnet. Um die Anbindung an AIA zu erstellen, wird eine Application Business Message (ABM) an den Requestor Application Business Connector Service (ABCS) gesendet. Die ABM ist anwendungsspezifisch und wird im ABCS zur Enterprise Business Message (EBM) umgewandelt. Der ABCS ruft eine Methode des Enterprise Business Service (EBS) auf. Der Input im EBS basiert auf dem Enterprise Business Object (EBO), dies ist das kanonische Datenobjekt, welches AIA intern verwendet wird. Der EBS übernimmt das Routing und ruft einen Provider ABCS auf, dabei basiert der Nachrichtenaustausch auf der EBM. Der Provider ABCS wandelt dann wiederrum die EBM in die ABM der Provider Application um und ruft einen Service der Provider Application auf. Somit ist neben der technischen auch eine logische Entkopplung erreicht, da die Applikationen gegenseitig nichts voneinander wissen.
Bei der Transformation von ABM zu EBM bzw. EBM zu ABM bietet AIA unter anderem zwei wichtige Funktionen an:

- Cross Referencing (XREF)
- Domain Value Mapping (DVM)

Der Ablauf für das XREF ist in Abbildung 2 schematisch dargestellt. Es handelt sich dabei um Mapping von Schlüsselwerten (IDs) aus den unterschiedlichen Anwendungen. Die IDs werden mit Hilfe eines XSL-Templates, das im AIA Standard vorhanden ist, in Transformationen gemapped. Der Code sieht dabei wie folgt aus:

```
<corecom:ID>
<xsl:call-template name="lookupOrPopulateXRef">
<xsl:with-param  name="xrefTableName"  select="CUSTOMERPARTY_CONTACTID.xref"
/>
<xsl:with-param name="xrefReferenceColumnName" select="'SEBL_01'" />
<xsl:with-param name="xrefReferenceValue" select="12345" />
<xsl:with-param name="xrefColumnName" select="'COMMON'" />
</xsl:call-template>
</corecom:ID>
```

Die Funktion sucht erst nach der ID und liefert im Erfolgsfall die Common ID. Wenn die ID noch nicht vorhanden ist, wird eine neue Common ID angelegt. Die Common ID wird ausschließlich innerhalb von AIA verwendet.
Eine weitere Funktion ist das Domain Value Mapping (DVM). Hierbei werden unterschiedliche Wertelisten der Applikationen in der Mitte zusammengeführt und kanonisiert. Als Basis dazu dient eine DVM-Datei in der die Werte der einzelnen Anwendungen gespeichert sind. Auch für DVM liefert AIA eine Funktion, bei der die Definitionsdatei, Quellsystem und Wert und das Zielsystem angegeben wird.

```
<xsl:value-of select='dvm:lookupValue("DVM-Datei", "Quellsystem",
"Quellwert“, Zielsystem", "Standardwert")'/>
```

Für beide Funktionen werden von AIA vorgefertigte Dateien für Standardanwendungen mitgeliefert. Im Bedarfsfall können diese Dateien mit neuen Anwendungen ergänzt oder vorhanden Werte für Auswahllisten angepasst werden.

Eine weitere wichtige Komponente von AIA ist das AIAConfigurationProperties.xml File. Dieses wird im MetadataStore gespeichert und zur Laufzeit im Cache gehalten. Die Datei definiert allgemeine Einstellungen wie zum Beispiel für die Benutzerbenachrichtigung im Fehlerfall oder die Servicekonfiguration für AIA Services. Das nachfolgende Listing zeigt eine Servicekonfiguration eines ABCS innerhalb des Files.

```
<ServiceConfiguration
serviceName="{http://xmlns.oracle.com/ABCSImpl/CRM/Core/AIADemoSyncCustomer
PartyListCRMProvABCSImpl/V1}AIADemoSyncCustomerPartyListCRMProvABCSImpl">
<Property name="TRACE.LOG.ENABLED">false</Property>
<Property name="Default.SystemID">AIADEMO_CRM_01</Property>
<Property
name="Routing.AIADemoSyncCustomerPartyListCRMProvAdapter.RouteToCAVS">false
</Property>
<Property
name="Routing.AIADemoSyncCustomerPartyListCRMProvAdapter.AIADEMO_CRM_01.End
pointURI">http://HOSTNAME:8001/soa-
infra/services/default/AIADemoSyncCustomerPartyListCRMProvDBAdapter/AIADemo
SyncCustomerPartyListCRMProvAdapter</Property>
<Property
name="Routing.AIADemoSyncCustomerPartyListCRMProvAdapter.CAVS.EndpointURI">
http://HOSTNAME:8001/AIAValidationSystemServlet/asyncrequestrecipient</Prop
erty>
</ServiceConfiguration>
```

Zunächst wird der Name der Komponente mit Namespace angegeben. Anschließend können verschiedene Properties konfiguriert werden. Dabei ist die Default.SystemID ein wichtiger Parameter, der wiederum in den DVM und XREF-Funktionen verwendet wird. Um das Composite Application Validation System (CAVS) für einen Service zu aktivieren muss RouteToCAVS auf true gesetzt und ein Endpoint angegeben werden. Weiterhin kann hier der konkrete EndpointURI angegeben werden, so dass die Endpunkte unabhängig von der Entwicklung sind.

AIA Entwicklungsprozess

Der AIA Entwicklungsprozess durchläuft unterschiedliche Phasen und wird von unterschiedlichen Rollen ausgeführt. Die nachfolgende Abbildung zeigt die Übersicht des Gesamtprozesses mit den Aktivitäten und den zugehörigen Tools.

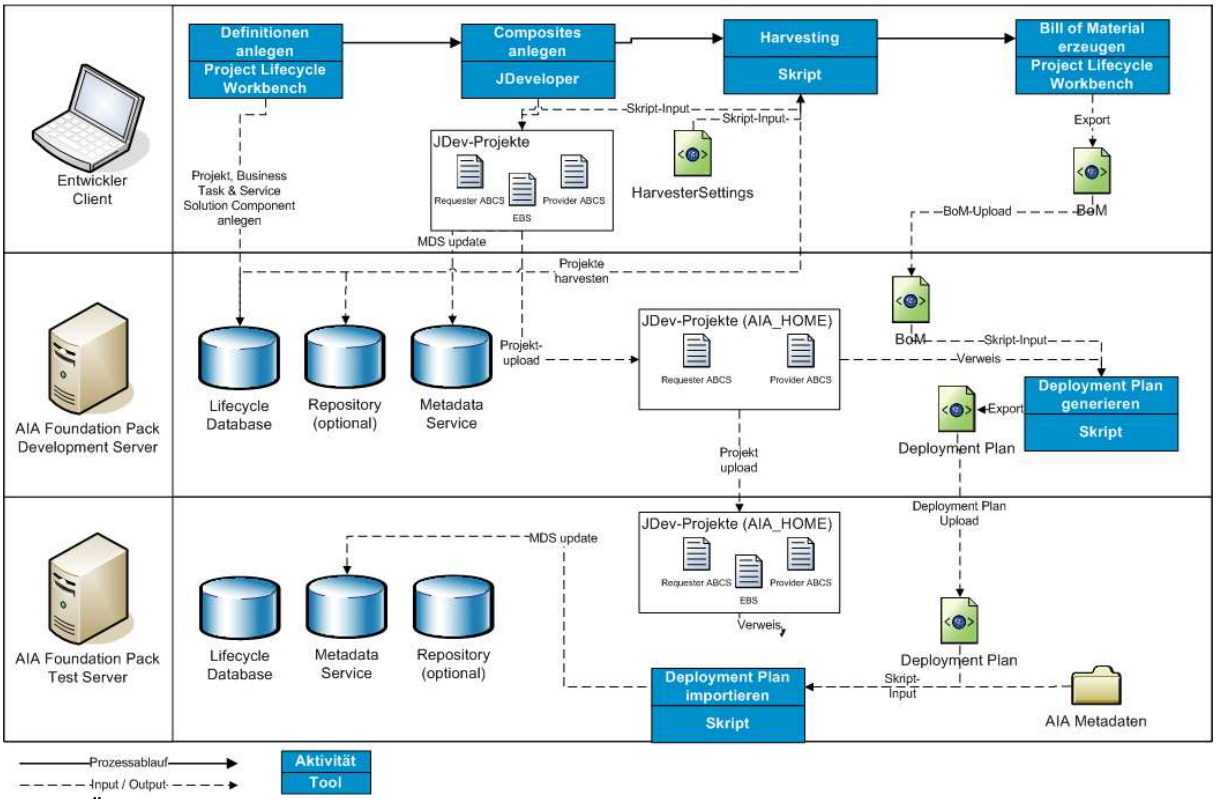

Abb. 3: Übersicht des AIA Entwicklungsprozess mit den wichtigsten Bestandteilen

Zunächst werden Definitionen mit Hilfe der Project Lifecycle Workbench (PLW) angelegt. Diese Definitionen beinhalten Projekt, Business Tasks und die zugehörigen Service Solution Components. Die Basis für die Definitionen wird in der Regel durch Business Analysten in Geschäftsprozessmodellen erstellt und vom technischen Architekten in die PLW in implementierbare Schritte zerlegt. Außerdem wird in der PLW identifiziert, ob es sich um wiederverwendbare oder neue Funktionen handelt.

Als nächstes werden die neuen Service Solution Components im Entwicklungswerkzeug JDeveloper erstellt. Falls es sich hierbei um einen Application Business Connector Service (ABCS) handelt, wird dieser mit Hilfe des Service Constructor erstellt. Der Service Constructor ist ein Plug-in für den Oracle JDeveloper, der auf Informationen der Project Lifecycle Workbench zugreift und danach ein BPEL-Composite automatisiert erstellt. Inkludiert sind dort auch AIA Funktionen wie z.B. Error Handling und die Integration in das Test Framework (CAVS).

Nach abgeschlossener Implementierung der Service Solution Components wird mit Hilfe eines Skripts und Konfigurationsdateien das Harvesting durchgeführt. Dabei werden Informationen, die während der Implementierung entstanden sind, z.B. Annotationen in den Composites, die die Abhängigkeiten zwischen den Artefakten beschreiben, in die PLW zurückgespielt. Optional können diese auch in das Oracle Enterprise Repository geharvested werden.

Als nächstes wird mit Hilfe der PLW pro Projekt die Bill of Material (BoM) erzeugt. Die dafür benötigten Informationen wurden durch das Harvesten in die PLW eingefügt. Die Bill of Material definiert die zugehörigen Komponenten (EBS, ABCS, usw.) eines Integrationsprozesses.

Um die Composites auf verschiedenen Umgebungen bereitzustellen, wird mit Hilfe eines Skripts und der Bill of Material ein Deployment Plan erstellt. Die Entwicklungsartefakte müssen im AIA_HOME-Verzeichnis abgelegt werden, da der der generierte Deployment Plan auf diese verweist.

Im letzten Schritt wird der Deployment Plan verwendet, um die AIA Artefakte und abhängigen Komponente auf weiterführenden Servern (z.B. der Testumgebung) bereitzustellen. Dabei werden umgebungsabhängige Einstellungen (z.B. Endpunkte von Services) in den Komponenten neu konfiguriert und anschließend mit den richtigen Umgebungsparametern deployed.

Best Practices / Lessons Learned

Die folgenden Punkte stellen Best Practices bzw. Lessons Learned dar, die aufgrund der Projekterfahrung gesammelt wurden. Teilweise sind diese von sehr technischer Natur, die nur mit tieferem Wissen verstanden werden können:

- Transformationen sind aufwendig zu testen, da lokale Tests aufgrund den AIA spezifischen Funktionen nur auf dem Server funktionieren, so dass immer ein kompletter Roundtrip mit Deployment auf den Entwicklungsserver und Test im Enterprise Manager vorgenommen werden muss. Dazu kommt noch, dass die Fehlermeldungen im Trace oder in den Logfiles teilweise sehr allgemein gehalten sind und man nicht weiß, wo genau das Problem liegt.
- Das Zusammenspiel von JDeveloper und AIA Service Constructor ist bei den Transformationen noch nicht ideal. Es kann passieren, dass im Design Mode (graphische Möglichkeit ein Mapping von Felder bzw. Strukturen vorzunehmen) Teile der Transformation gelöscht werden bzw. die XLST Versionsnummer geändert wird, die dann bei der Ausführung zu Fehlern führt. Die Einschränkung kann umgangen werden, wenn das Grundgerüst der Transformation kopiert wird und anschließend graphisch bearbeitet wird. Nach der graphischen Bearbeitung fügt man den Code der eigentlichen Transformation an die richtige Stelle des kopierten Grundgerüsts. So kann der Vorteil der graphischen Transformation genutzt werden, ohne den generierten Teil zu verlieren.
- AIA XPath Extension Functions
  Mit dem Service Construtor von AIA für den JDeveloper werden AIA spezifische XPath Funktionen mitgeliefert. Diese können in Transformationen genutzt werden. Nach der Installation des Service Constructor Plug-in im JDeveloper müssen die AIA XPath Extension Functions vorhanden sein. Ob diese vorhanden sind, kann im Expression Builder des JDeveloper geprüft werden.
- CAVS und setDynamicPartnerlinkSequence
  Es ist zu empfehlen während der Erstellung des ABCS mit Hilfe des Service Constructor CAVS zu aktivieren. Dies kann in den Target Service Options geprüft und aktiviert werden, standardmäßig ist die Checkbox aktiviert. Die Auswirkung ist, dass im BPEL Flow die Sequenz „setDynamicPartnerLinkSequence“ angelegt wird. Innerhalb der Sequenz werden Parameter aus der AIAConfigurationProperties.xml gelesen. Dazu zählt ob die Anfrage zu CAVS geroutet werden soll. Falls kein Routing zu CAVS erfolgt, wird mit Hilfe der Sequenz auch der Endpunkt neu definiert, wenn die Einstellung in der AIAConfigurationProperty gesetzt ist.
- NullPointerException in embedded Java “getTargetEndpointLocation”
  Wenn die obengenannte Sequenz im ABCS vorhanden ist, wird die Einstellung Endpoint URI für den dynamischen Endpunkt gelesen. Ist diese Einstellung nicht gesetzt, kann es vorkommen, dass zur Laufzeit eine NullPointerException geworfen wird. Der Grund ist, dass im embedded Java die Variable targetEndpointLocation mit null deklariert wird. Die Deklaration
  ```
  java.lang.String targetEndpointLocation = null;
  ```
  sollte durch
  ```
  java.lang.String targetEndpointLocation = “”;
  ```
  ersetzt werden.
- Refresh des SOA Cache nach MDS Aktualisierung

Änderungen im MetaDataStore werden erst nach Reload des SOA Cache aktiv. Dieser Reload kann über die Homepage des AIA Foundation Pack erfolgen. Auf der Homepage gibt es dafür den Bereich AIA Setup.

- Dateien und Ordner aus MDS löschen
  Das Entfernen von Dateien aus dem MDS Repository ist zurzeit nur mit Hilfe des WebLogic Scripting Tools möglich. Dazu wird die Methode deleteMetadata von WLST verwendet. Wichtig dabei ist, dass WLST aus dem SOA_HOME gestartet wird. Wenn WLST aus dem WL_HOME gestartet wird, ist die Methode nicht bekannt. Das Löschen von ganzen Ordnern im MDS ist aktuell nicht möglich.
- Ordnerstruktur im MDS mit Bezug auf Systemname
  Um zu vermeiden, dass im Deployment Plan der MDS Pfad angepasst werden muss, sollte das System beim Anlegen des ABCS immer genauso benannt werden wie die Ordnerstruktur im MDS. Zum Beispiel als System SYS1 im ABCS Wizard angegeben wird, muss auch im MDS Verzeichnis apps/AIAComponents/ApplicationObjectLibrary/SYS1/ angelegt werden. Das Matching erfolgt dabei Case sensitiv.
- Aktuell gibt es keine Unterstützung bei der Erstellung von eigenen kanonischen Datenmodellen und Services. Diese muss von Hand durchgeführt werden, wobei man bestehende kopiert und entsprechend anpasst. Das ist allerdings sehr fehleranfällig.

Fazit

Mit dem AIA 11g Release hat Oracle ein rundherum gelungenes Produkt auf den Markt gebracht. Dieses profitiert nicht nur allein von der sehr stark weiterentwickelten Technologiebasis SOA Suite 11g und Weblogic 11g Applikationsserver, sondern vor allem durch die konsequente Weiterentwicklung der AIA Konzepte und wesentlich bessere Unterstützung bei der Entwicklung von Integrationsprozessen. Die aktuell vorhandenen Kinderkrankheiten sollten lösbar sein, so dass man wirklich behaupten kann, dass mit AIA 11g ein neuer Meilenstein für die Integration von Systemlandschaften geschaffen wurde.

Kontaktadresse:

Gregor Bublitz
ec4u expert consulting ag
Zur Giesserei 19-27B
D-76227 Karlsruhe

Telefon:	+49 (0) 721-46476 443, +49 (0) 151 19507443
Fax:	+49 (0) 721-46476 4443
E-Mail	gregor.bublitz@ec4u.de
Internet:	www.ec4u.de