|  |
| --- |
| ArchStudio |
|  |

Изготвили:

Михайл Николов 80527

Ана Иванова 80531

Борис Ненов 80604

Теодор Меродийски 80548

**Contents**

[1 Увод](#h.30j0zll)

1. Въведение във визуализацията на софтуерни архитектури
2. Някои често срещани архитектури
3. Начини за визуалицация на софтуерни архитектури
4. [Триизмерни визуализации](#h.30j0zll)

[2 ArchStudio](#h.1fob9te)

[2.1 Въвдение в ArchStudio](#h.3znysh7)

[2.2 Възможности на средата](#h.2et92p0)

[2.3 Подход за разработка с ArchStudio](#h.tyjcwt)

[3 Компоненти на ArchStudio](#h.1fob9te)

[3.1 Archipelago](#h.3znysh7)

[3.2 ArchEdit](#h.2et92p0)

[3.3 Archlight](#h.tyjcwt)

[4 xADL](#h.3dy6vkm)

[4.1 Основни характеристики на xADL](#h.1t3h5sf)

[4.2 Предимства на xADL пред UML](#h.4d34og8)

[4.3 xArch](#h.2s8eyo1)

[4.4 Използване на xADL в проект](#h.17dp8vu)

[4.5 Начин на работа с xADL](#h.3rdcrjn)

[4.5.1 Apigen](#h.26in1rg)

[4.5.2 Създаване на схема](#h.lnxbz9)

[4.5.3 xArch/xAdl Data binding библиотеки](#h.35nkun2)

[Библиография](#h.1ksv4uv)

1. Увод
   1. 1.1 Въведение в софтуерните архитектури

С нарастването на размера и сложността на софтуерните системи, проблемът с дизайна започва да се простира отвъд алгоритмите и структурите от данни, използвани в процеса на разработка. Изработването на дизайн и спецификации за цялостната структура на софтуерния продукт се превръща във съвсем нов вид проблем.

Един от IEEE стандартите, и по-конкретно - стандартът IEE00, гласи следното: „Архитектурата е базовата организация на една система, включваща се във всеки един неин компонент, във връзките между тези компоненти, както и в средата и принципите определящи нейния дизайн и еволюция във времето.

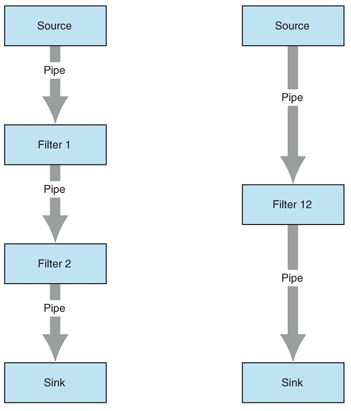
Визуализацията на софтуерни архитектури обикновено се занимава със структурата на софтуера на различни нива на абстракция, но когато става въпрос за истинското описване на архитектура, съществуват много аспекти. Те включват нейната „груба“ организация и глобална контролна структура, както и протоколи за комуникация, синхронизация и достъп до данни, добавянето на функционалност към дизайн елементи, композицията на същите, скалируемостта и изпълнението, както и избора между различни алтернативи на дизайн.

Разделянето на една система на модули улеснява дизайна и разработката на софтуерните системи. Както Дейвид Парнас е казал: „Съществуването на йерархични структури ни подсигурява, че можем да премахнем горните нива на дървото и да почнем ново дърво от главния клон.“

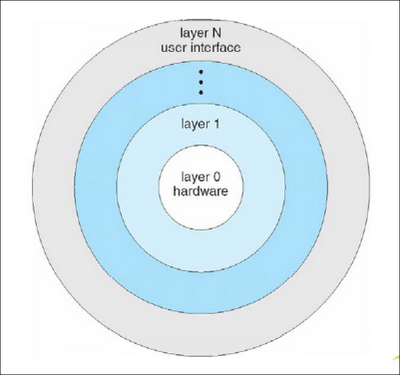
Широко разпространено е в софтуерното инженерство, че когато се прави дизайна на софтуерна архитектури трябва да има малка свързаност между модулите и високо ниво на съгласуваност вътре в самите модули. Докато свързаността представлява „,мярка за силата на асоциацията, основана на връзката между два модула, докато съгласуваността определя „степента на свързаност между елементите на един-единствен модул.“ Но тези представи за свързаност и съгласуваност не взимат предвид посоката на зависимостите. Така в частност, във връзка с обектно-ориентирания дизайн, тези първоначални идеи са разширени и нови принципи, както и свързани с тях метрики са измислени за да насочват дизайна на софтуерните системи.

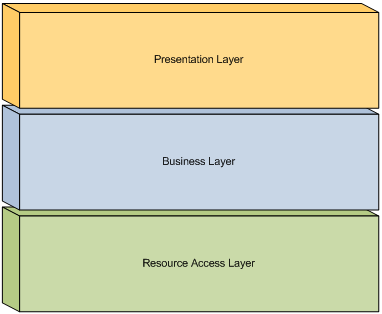
* 1. 1. 2 Някои познати архитектури

*Тръби и филтри –* при този тип архитектура филтрите получават поток от входни данни и произвеждат поток от изходни такива. Тръбите се използват за предаване на изходните данни от един филтър като входни такива за следващия филтър. Infopipes от своя страна разширява метафората за „тръбите“ с буфери, помпи и т.н, за да осъществи дизайна на разпределени потокови приложения.

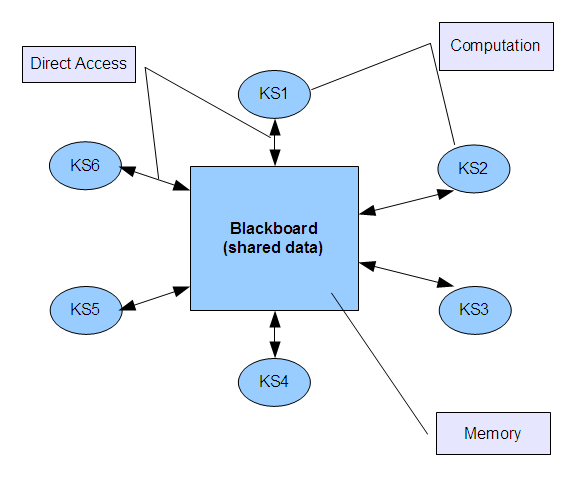


*Слоести системи –* в този случай функционалността на системите е организирана в няколко отделни слоя. В една изцяло слоеста система, функционалността на един слой е имплементирана от функционалността, която е осигурена от слоя точно под него. Много често обаче, в практиката слоевете могат достъпват и някои от слоевете под тях, не само този който е директно под тях. При тази архитектура за визуализация може да се използват и „луковия“ модел и модела, при който слоевете са разположени хоризонтално един над друг (вж. фиг 1.1.2 и 1.1.3). Най-често срещаните слоеве при този архитектурен шаблон са презентационния (или още потребителски интерфейс) слой, приложния (другото му име е слоя за услуги), слоя за бизнес логика или още бизнес слой, както и инфраструктурния слой. Понякога има допълнителен слой – бизнес инфраструктурен слой.





*Черни дъски –* при архитектурния модел на черните дъски, има множество компоненти, които споделят данни чрез черна дъска, като тези компоненти могат да пишат и четат от черната дъската. Така черната дъска съдържа задачи, които трябва да бъдат изчислени, както и резултати от вече завършили задачи.



**1.3 Начини за визуализация на софтуерни архитектури**

По-горе показахме простички диаграми за визуализация на различни архитектурни модели, но едно кратко търсене в Google показва наличието на множество различни начини за представяне на архитектурни диаграми. Повечето от последните използват ad hoc визуализации (тоест са използвани за специфичния проблем, който представят и не могат да бъдат адаптирани за други цели), а също така и семантиката на цветовете, възлите, иконите, линиите и стрелките е честно неясна. За да избегне подобна ситуация, човек може да следва общи правила за семантиката на горе-изброените. Има създадени такива, но когато стане въпрос за големи и сложни системи, създавани от множество разработчици, този подход е неудобен и трудно приложим, а освен това разбирането на архитектурните диаграми там е ключово и стандартизирани графични нотации като UML са решението на проблема.

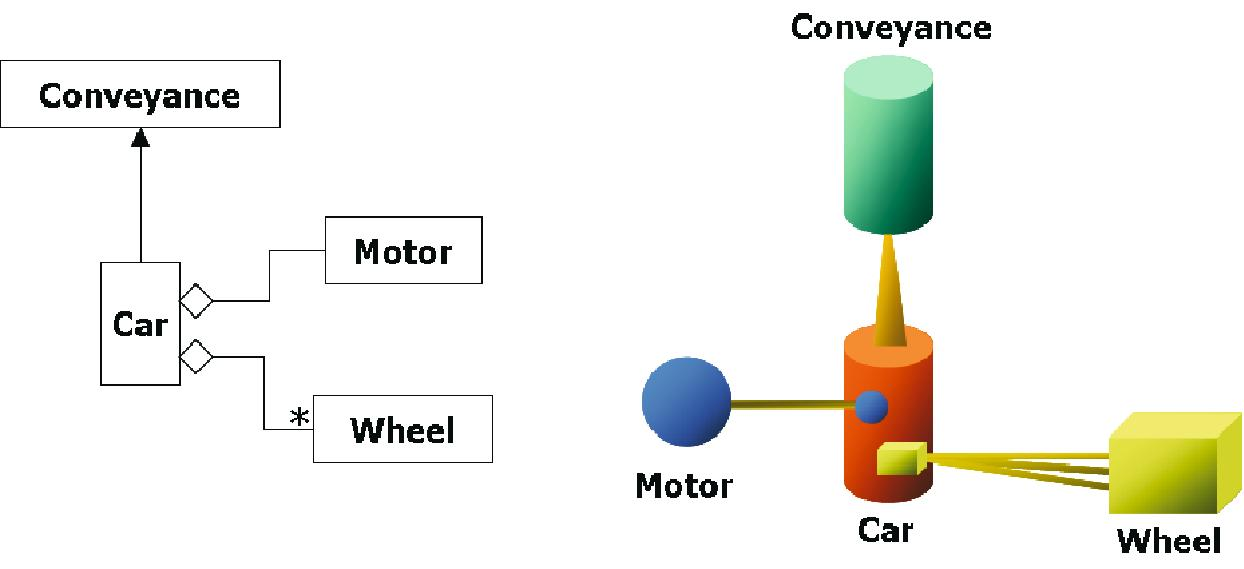
**The Unified Modeling Language**

*UML* e графичен език за визуализиране, специфициране, конструиране и документиране на елементите на една софтуерна система. Той комбинира различни техники – от моделиране на данни, бизнес моделиране, до обектно и компонентно моделиране. Може да бъде използван със всички процеси през цикъла на разработка на софтуера, както и с различни имплементационни технологии.

UML комбинира методите на Booch, Rumbaugh и Jacobsen. Всеки един от тези трима известни софтуерни инженери, понякога наричани “Тримата амигос“ (тримата другари, превод от испански), са разработили различни популярни дизайн методи, съставени от графична нотация и процес за разработка на дизайн. За да приключат така наречената война на методите, те решават да слеят своите методи и през далечната 1997-ма година ре предлагат UML на OMG (The Object Management Group) като стандарт за моделиране на софтуерни системи.

UML осигурява значителен брой различни по тип диаграми, включително класови и обектни диаграми, поведенчески диаграми, диаграми на взаимодействие, имплементационни диаграми и модел-управление диаграми.

UML диаграмите са съставени от малък набор графични примитиви: като цяло те се състоят от текст, кутии, линии и стрелки. Като резултат от това, дизайнерите могат да рисуват лесно UML диаграми на ръка без цветни пишещи средства (химикали, моливи и т.н). Въпреки популярността на този тип диаграми, тяхната чисто визуална ефективност е ниска. В последни потребителски изследвания, геоните(1) били използвани за правенето на диаграми на софтуерни архитектури. Геоните представляват съвкупност от 24 вида примитиви, които са инвариантни спрямо гледната точка триизмерни обекти. Оттук следва, че те са лесни да бъдат разпознати от човек, дори когато са проектирани върху равнина. Няколко експеримента със студенти по информатика показали, че субектите визуално анализират „геонни“ диаграми много по-бързо и с по-голяма точност, а освен това и че могат да си ги спомнят по-бързо в сравнение с еквивалентните UML диаграми. По-долу са показани UML диаграма и еквивалентната и геонна такава, представящи много опростено кола.

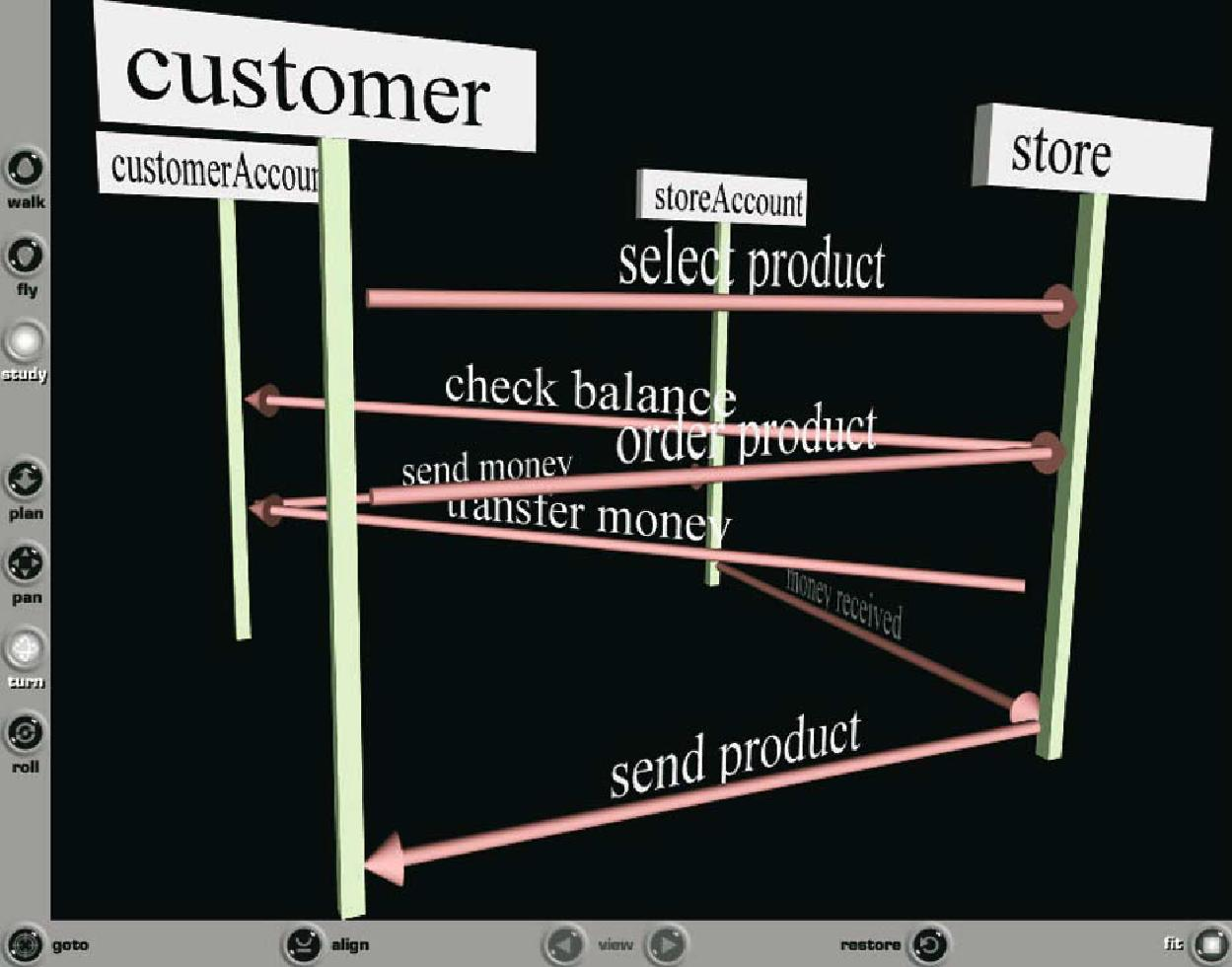


* 2. 1.4 Триизмерни визуализации и софтуерни архитектури

Много спорове са проведени на тема дали е спекулация че триизмерните визуализации са по-добри от двуизмерните такива. Съществува много известен спор между Ware и Frank(2) относно количеството информация, което човек може да възприеме от триизмерна диаграма е цели 3 пъти по-голямо от това, което може да бъде възприето от 2Д диаграма. Мотивирани от тези резултати, както и от напредването на технологиите, много изследователи предлагат триизмерни визуализации на софтуерните архитектури и дори използването на виртуални среди, които позволяват да се разходите буквално из тези архитектури.

Гогола(3) и компания предлагат няколко сценария при които UML диаграмите биха спечелили от триизмерния вид. Те илюстрирали тези сценарии чрез триизмерни UML диаграми, които самите те имплементирали за целта като използвали VMLR(4) (*Virtual Reality Modeling Language*):

* В клас диаграма, важните класове са нарисувани на преден план и поради тази причина са на фокус. Нещо повече, могат да съществуват няколко перспективни изгледа на една и съща диаграма. Във всяка от перспективите, фокусът пада върху различени класове и когато потребителя променя изгледа на перспективата, се преместват с нея и възлите на диаграмата към тяхната нова позиция посредством плавна анимация.
* В обектните диаграми, разнообразни форми могат да бъдат използвани за целта на изобразяването на обекти. Обекти от един и същ клас имат подобни форми.
* За последователни диаграми, анимациите показват съобщенията, изобразени като малки топки, които се движат от подателя до получателя им. Този триизмерен вариант решава проблемът от множеството се пресичащи се и припокриващи се стрелки за едновременно изпратени съобщения. За съжаление обаче проектирането на триизмерните визуализации върху двуизмерната равнина на монитора създава други пресичания и проблеми. По-долу е показана диаграма на последователност.
* Някои диаграми могат да бъдат комбинирани – например една клас диаграма може да бъде показана на заден план, а отпред – свързана с нея диаграма на последователност.



**X3D-UML**

X3D-UML е докторската теза на Paul McIntosh и представлява „Потребителски центриран дизайн, имплементация и оценка на 3D UML използвайки X3D (eXtensible 3D specification) в областта на софтуерната визуализация“.

Чрез X3D-UML диаграмите биват създадени директно от кода на софтуерния продукт. Как се случва това? Кода бива първо преобразуван в XML вид, след което няколко XSLT скрипта извличат от него необходимата информация и от тях създават X3D модели. X3D, както вече споменахме, е eXtensible 3D specification и освен това е XML-базирания наследник на по-горе споменатия VRML.

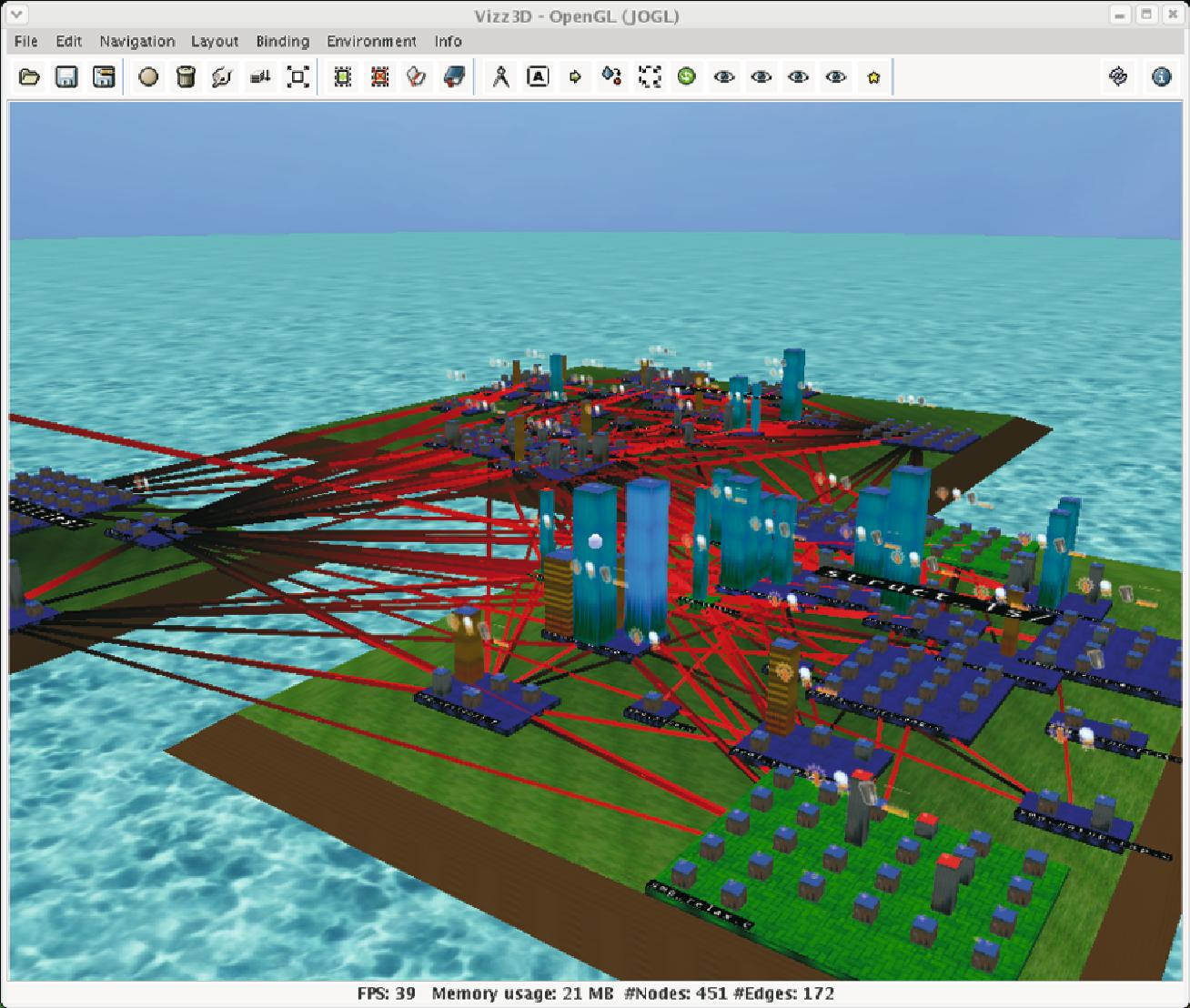
Imsovision е много интересна система, която предлага стереоскопична 3Д визуализация на класовете, техните свойства (properties) и съществуващите между тях зависимости и агрегации във среда на виртуална реалност, наречена пещера. Тъй като пещерата е стая, където визуализацията е задно-проектирана на нейните стени, потребителят може да влезе в стаята и да взаимодеиства със 3D сцената. В Imsovision класовете са представени от платформи, методите са колони, а атрибутите са сфери, поставени отгоре върху платформите. Платформите от подкласове са поставени до техните базови класове.

Много подобен подход има и Software Landscapes на Balzer. Тяхното представяне използва по-реалистични метафори от това на Imsovision – платформите са заменени от острови, а колоните – от небостъргачи.

При Software World, светът представлява софтуерната система като цяло, една страна представя един пакет, докато град – файл, а квартал от град – клас, а методите на класа са представени от сградите в него. Размера на сградата определя броя редове код, съставляващи метода, докато вратите представят неговите параметрите, а прозорците – променливите, декларирани в метода, а цвета на сградата определя дали метода е публичен или не.

Panas и компания предлагат, че динамичната информация също трябва да бъде включена в една такава визуализация например двупосочните извиквания могат да бъдат представени чрез улици, а еднопосочните чрез реки. Информацията се предава чрез лодки или коли между класовете и между пакетите.

В заключение – тризимерните визуализации на софтуерни архитектури са приели буквално метафори от реалния свят и резултантните визуализации са доста привлекателни и могат да помогнат за в бъдеще за предаването на архитектурна информация на неексперти.



1. ArchStudio
   1. Въвдение в ArchStudio

ArchStudio е софтуер с отворен код. Той представлява среда за разботка на системен софтуер, създаден и развит от Института за Софтуерни Проучвания към Калифорнийския Университет в Ървин. Това е среда на интегрирани инструменти за моделиране, визуализиране, анализ и прилагане на софтуерни продукти и системи архитектури.

ArchStudio е софтуер с отворен код и е напълно безплатен (BSD-style лиценз). Праграмата е разработена от Института за софтуерни проучвания (Instituse for Software Research) към Калифорнийския университет „Irvine“. Въпреки, че ArchStudio е научен проект, неговите разаботчици са положили значитилени усулия за налагането на продукта.

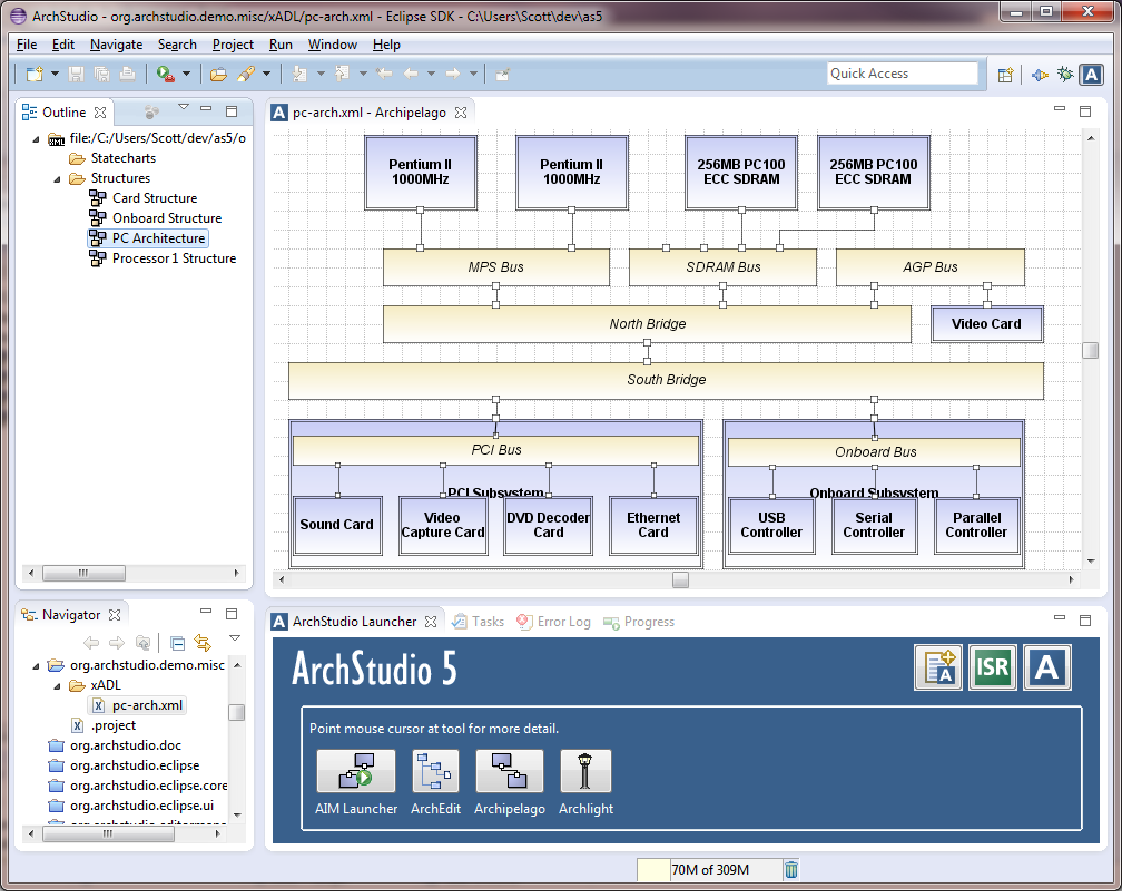
ArchStudio е реализиран на Java 2™ Standard Edition (J2SE) version 5.0, познат също като Java 1.5. Имплементиран е като Eclipse плъгин, поради което работи на всички платформи, поддържани от Eclipse - Windows 2000 и по-нов, MacOS X, и Linux/UNIX.

ArchStudio наподобява на други среди за разработка като Eclipse и VisualStudio, с една основна разлика : ако при Eclipse и VisualStudio разработчиците се фокусират върху програмен код, то в ArchStudio фокуса е върху описание на софтуерни архитектури. Нека дефинираме понятието архитектура като обединение на конструктивни принципи за дадена система. Всеки един проект от каквато и да е област се изразява с резултат в даден контекст. Това означава, че всеки проект има своебразно различен набор от конструктивни принципи, считани за изисквания от заинтересованите от проекта страни.

ArchStudio има две основни роли в изграждането на софтуерни архитектури.

Първо, ArchStudio е среда за разработка на софтуерни архитектури. ArchStudio е с вградена поддръжка за моделиране на йерархичната структура на сложни системи,различни компоненти, конектори и интерфейси, продуктови линии на системи, които са свързани от обща основа, и така нататък.

Второ, ArchStudio е архитектурна meta-modeling среда. ArchStudio се основава на xADL описателен език. xADL позволява на разработчиците да определят и предефинират синтаксиса и семантиката на езика, за да отговарят на техните собствени нужди. ArchStudio предоставя идеална платформа за експерти в областта на софтуерните архитектури, които искат да проучат нови начини за моделиране на архитектурни концепции, без да инвестират необичайно много време в развитието на инфраструктура.

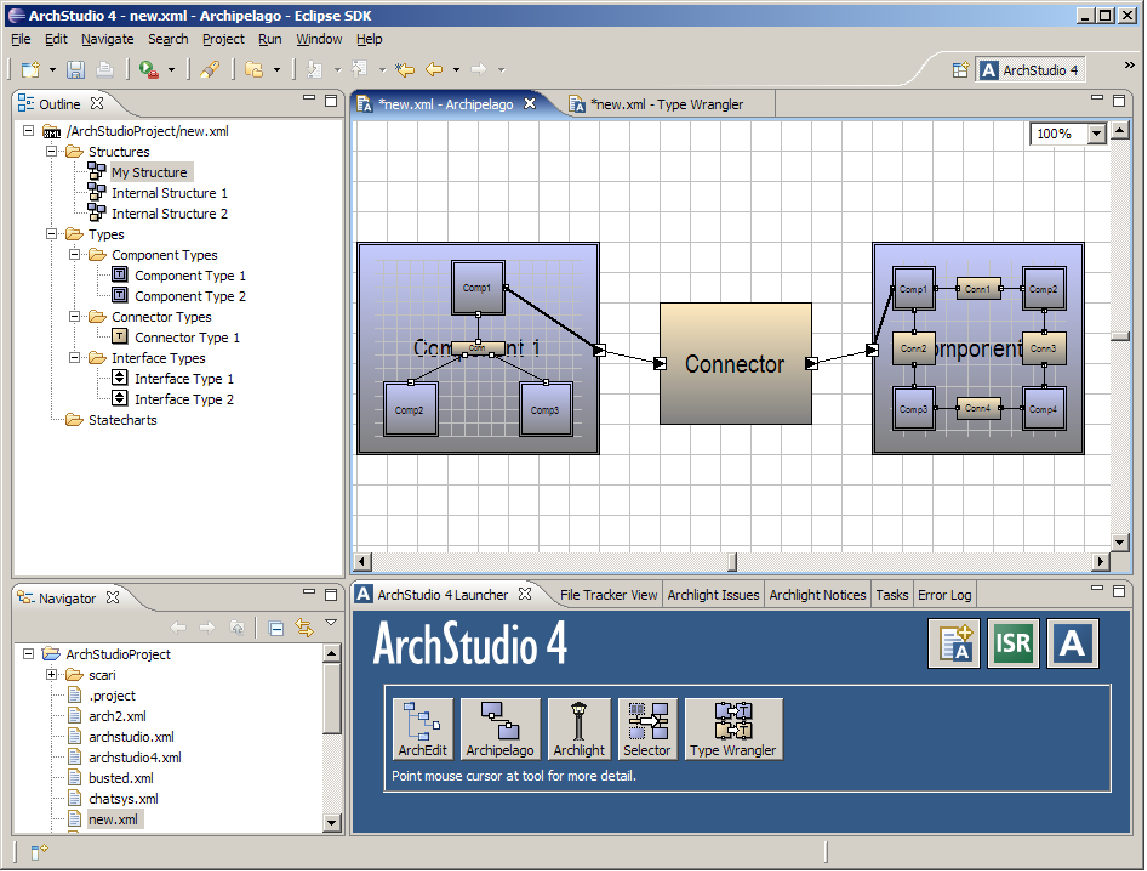


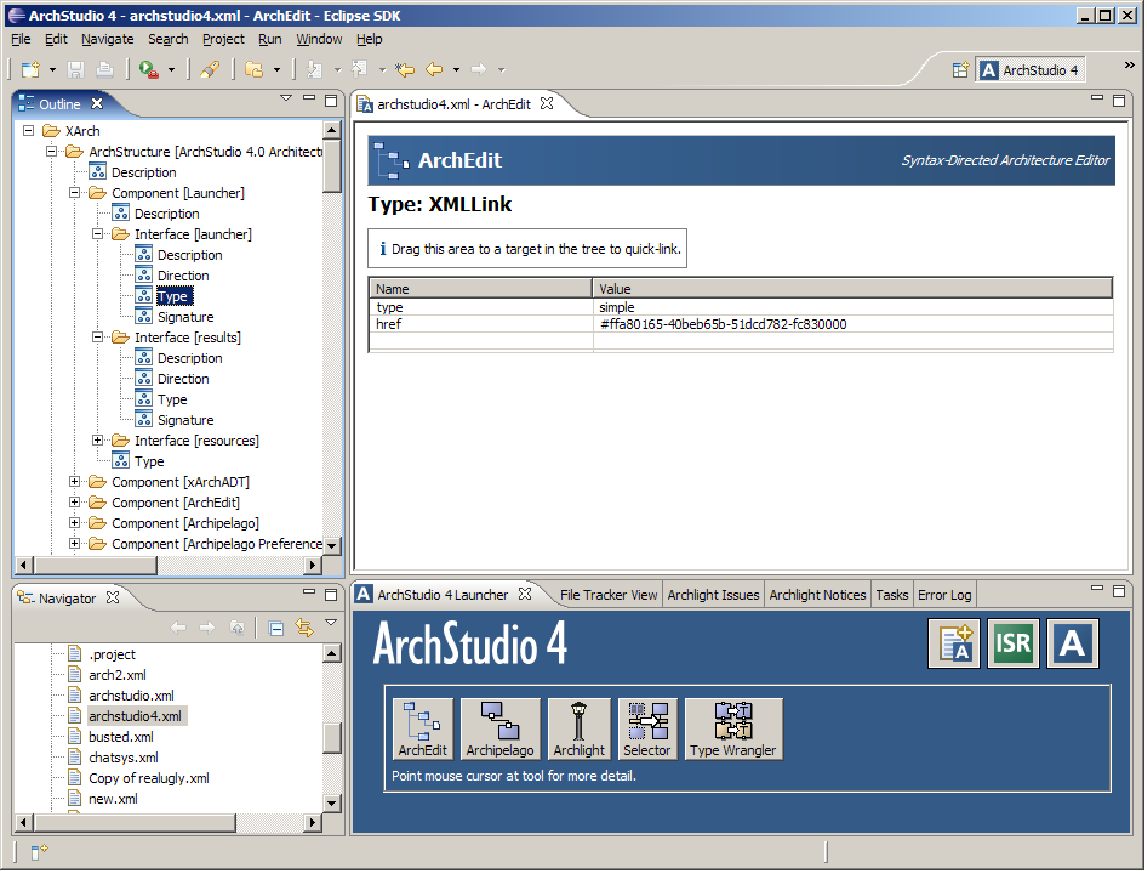
**Figure 1 Тази илюстрация показва йерархична архитектура в графичния редактор.**

* 1. Възможности на средата

ArchStudio създава и манипулира архитектурни описания, изразени в контекста на xADL архитектурно описание (ADL). xADL е първият модулно-разширен архитектурно описателен език. Вместо синтаксиса и семантиката на ArchStudio да са дефинирани монолитно някъде в контектста на средата, ADL разчупва свойствата за моделиране на модули, използващи стандартни XML схеми. ArchStudio притежава вградени инструменти за опериране върху xADL документи, почти по същия начин по който текстов редакор оперира върху текстов документ. Една основна разлика, обаче, е, че инструментариума на ArchStudio интегрира "на живо", което означава, че промяна в някой инструмент е отразена във всички други веднага. Както беше отбелязано по-горе, езика xADL може да бъде надграден от крайните потребители чрез добавянето на нови XML схеми за подпомагане на обхвата или разрешаването на специфични за проекта проблеми.

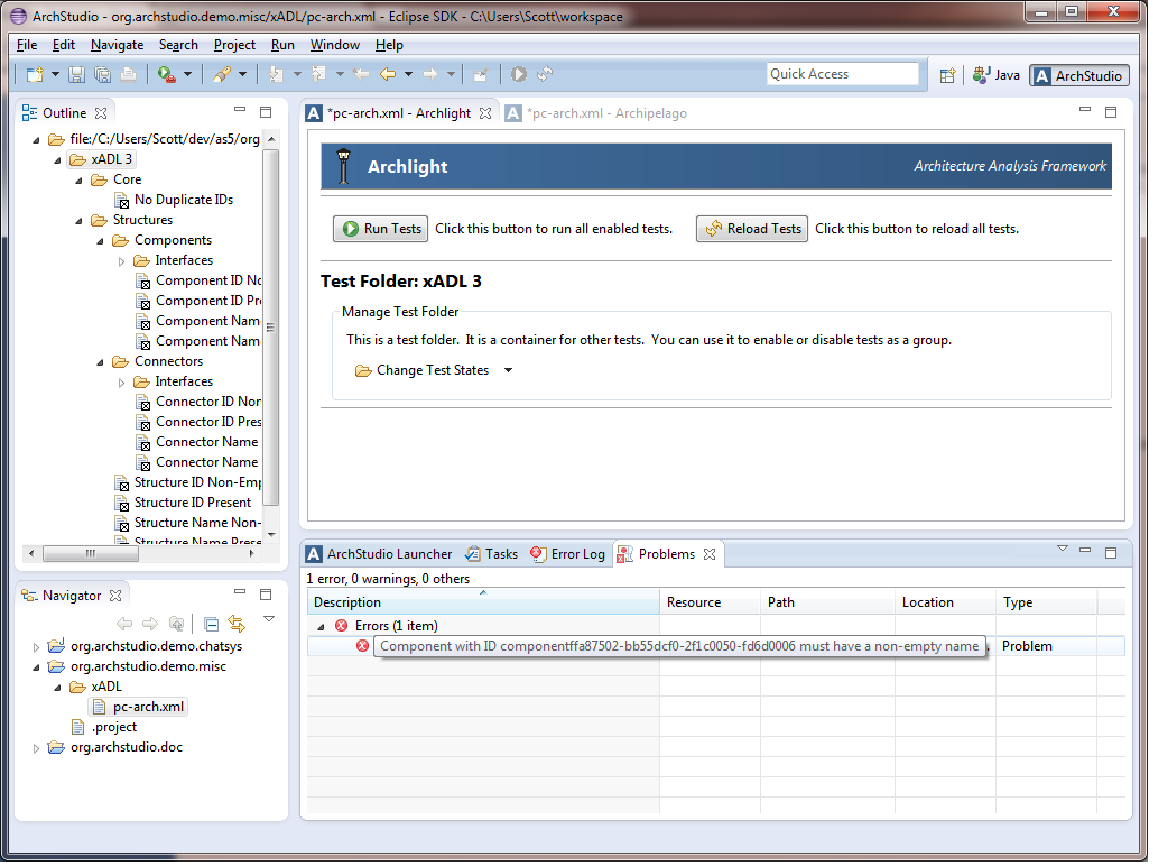
xADL езикът дефинира структурата на архитектурно-описателната информация, но освен това и позволява тази информация да бъде визуализирана и моделирана в процеса на работа. ArchStudio осигурява редица различни начина за визуализация на xADL модели. Графичния редактор на ArchStudio носи наименуванието Archipelаgo. Той осигурява визуализирането на информацията чрез символни графи много подобни на тези от PowerPoint, но за разлика от тях символната визуализация на Archipelago не е само картинки. Те са своебразна проекция на архитектурния модел, която е достъпна за обработка от софтуерния инжинер използващ ArchStudio. ArchStudio включва и други редактори като ArchEdit – синтактично ориентиран редактор и Type Wrangler, който предоставя възможността за персонализация на архитектурни модели.





**Figure 2 Тази илюстрация показва синтактично ориентирания редактор ArchEdit**

Фреймуърка Archlight осигурява начин за автоматично тестване на архитектурни описания по много и различни критерии. Грешките могат да бъдат показани и инспектирани, а потребителите могат да навигират в сайта за проблеми за всеки един редактор с няколко кликвания на мишката. В съответствие с разтегателната природа ArchStudio, всички тестове са осигурени от Archlight плъгините, но е предоставена възможността потребителите сами да добавят и прилагат свои собствени тестове.



* 1. Подход за разработка с ArchStudio

Основен аспект в процеса на разработка с ArchStudio е фактът, че предоставя възможността за надграждане на средата за разработка. Архитектура на ArchStudio е посочена във файл xADL, а от своя страна този xADL файл е част от изпълнението на ArchStudio. Всеки път, когато ArchStudio се стартира на машина, неговата архитектурно описание прави разбор, и информацията в това описание се използва с конкретни примери за да прибави компонентите и конекторите в архитектурата. ArchStudio е софтуер имплементиран в Java 2™ Standard Edition (J2SE) version 5.0 като се осъществява под формата на набор от Eclipse плъгини. Тя би трябвало да работи на всяка платформа, която поддържа Eclipse, включително Windows 2000 или MacOS X, и Linux / UNIX. Въпреки че ArchStudio е изследователски проект, нейните разработчици са положили значителни усилия да развият ипопуляризират средата за разработка на софтуерни архитектури. Предишния рилийз на ArchStudio е самостоятелено Java приложение. Зрелият ArchStudio програмния код е в основата на ArchStudio, но ArchStudio включва редица интересни нови функции, включително значително подобрения потребителски интерфейс и редица други подобрения.

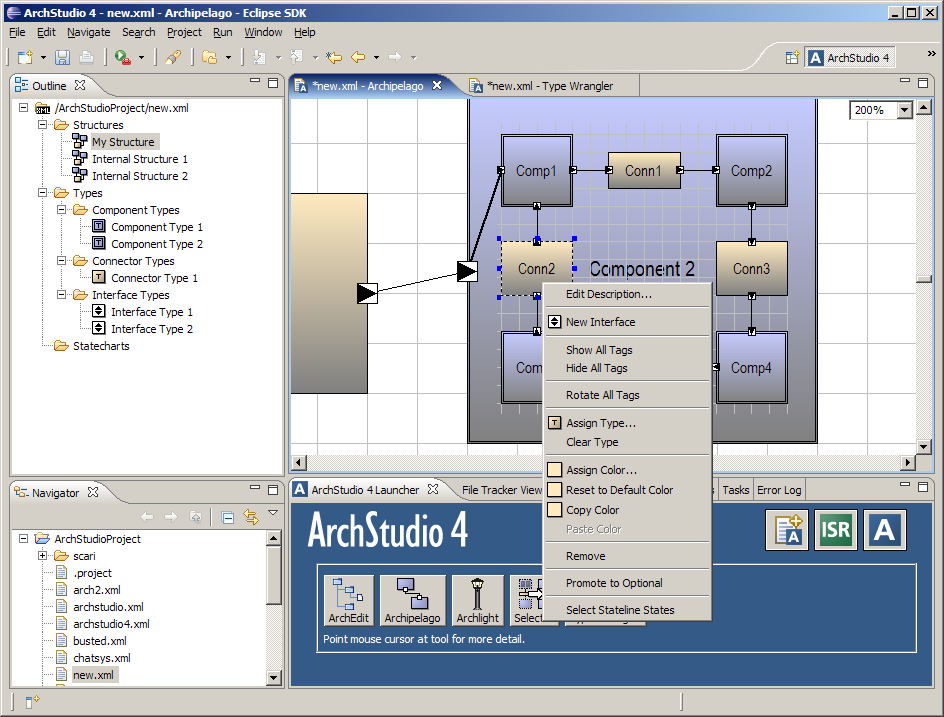
1. Компоненти на ArchStudio
   1. Archipelago

Archipelago е лесен за ползване графичен редактор на ArchStudio. Той се състои от набор от функционални добавки, които осигуряват визуализация и възможност за редактиране за различни xADL модули. Основната функция на Archipelago е графичната. Той представя архитектурата с помощта на „кутий“ и „стрелки“. Потребителският интерфейс е подобен на този на инструменти като Rational Rose, Microsoft Visio и Microsoft PowerPoint. За разлика от някои от тези редактори, Archipelago е зопзнат с основния език за моделиране (xADL) и промените в диаграмите се отразяват незабавно в xADL документа (и обратното е в сила).

Archipelago може да бъде отворен от началния екран на ArchStudio, както всеки друг ArchStudio редактор. Щом редакторът се отвори, се зарежда контекстно базирания потребителски интерфейс. Това означава, че вместо да се използват ленти с инструменти, режим на редактиране или главното меню, основеният метод за редакция е чрез контекстни менюта. Контекстните менюта обикновено се извикват чрез щракване с десен бутон върху елемента, който ще бъде редактиран. Системите, които не поддържат мишки с два бутона като Macintosh, могат да заредят контекстното меню чрез задържане на бутона Ctrl и кликване на желания елемент.

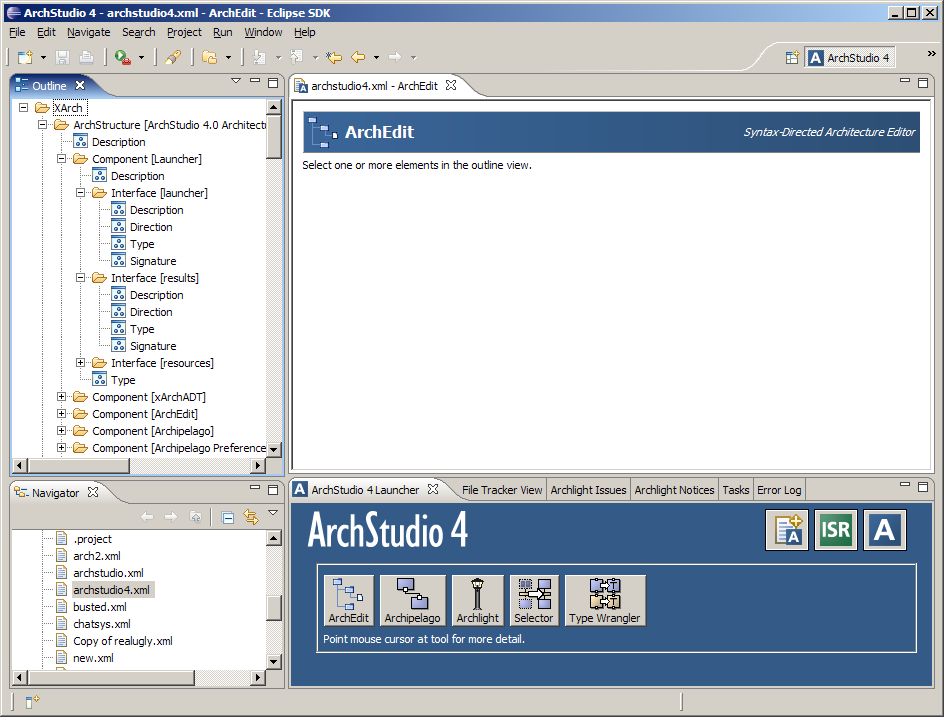
Като цяло, редактирането в Archipelago започва от дървото в Outline View. Кликването с десен бутон върху тези елементи позволява да се манипулира на най-високо ниво. Като цяло, двойно щракване върху един от тези контейнерни елементи ще отвори елемент в прозореца на редактора. Създаването на нови елементи е също толкова лесно. Достатъчно е да се кликне с десен бутон на мишката в празното пространство на прозореца и да се избере елемента, който искаме да създадем.

Графичният интерфейс за редактиране поддържа произволно мащабиране и скролиране. Скролирането се извършва с плъзгачите на прозореца, а мащабирането може да се извърши с колелцето на мишката. Чрез завъртане на колецето напред (надалеч) се увеличава, а чрез завъртане назад – намаля. При натискане на колелцето, курсора на мишката се променя на „ръка“. Докато колелцето е натиснато, движението на мишката ще предвижва и платното.



* 1. ArchEdit

ArchEdit е графичен синтаксис-насочен редактор за xADL документи. Той използва графичен интерфейс базира на дърво, подобен на много XLM редактори или Regedit Microsoft. Подобно на Regedit, ArchEdit е инструмент на ниско ниво за редактиране на архитектурни документи. Той не налага ограничения върху семантичните описания архитектура и е преди всичко полезен за проверка на ниско ниво и редактиране на такива описания. Предимството на този инструмент е, че той автоматично се адаптира към новите схеми на xADL когато езикът се надгражда. Поради тази причина, новите или разширени версии на ArchStudio не изискват обновяване на ArchEdit.

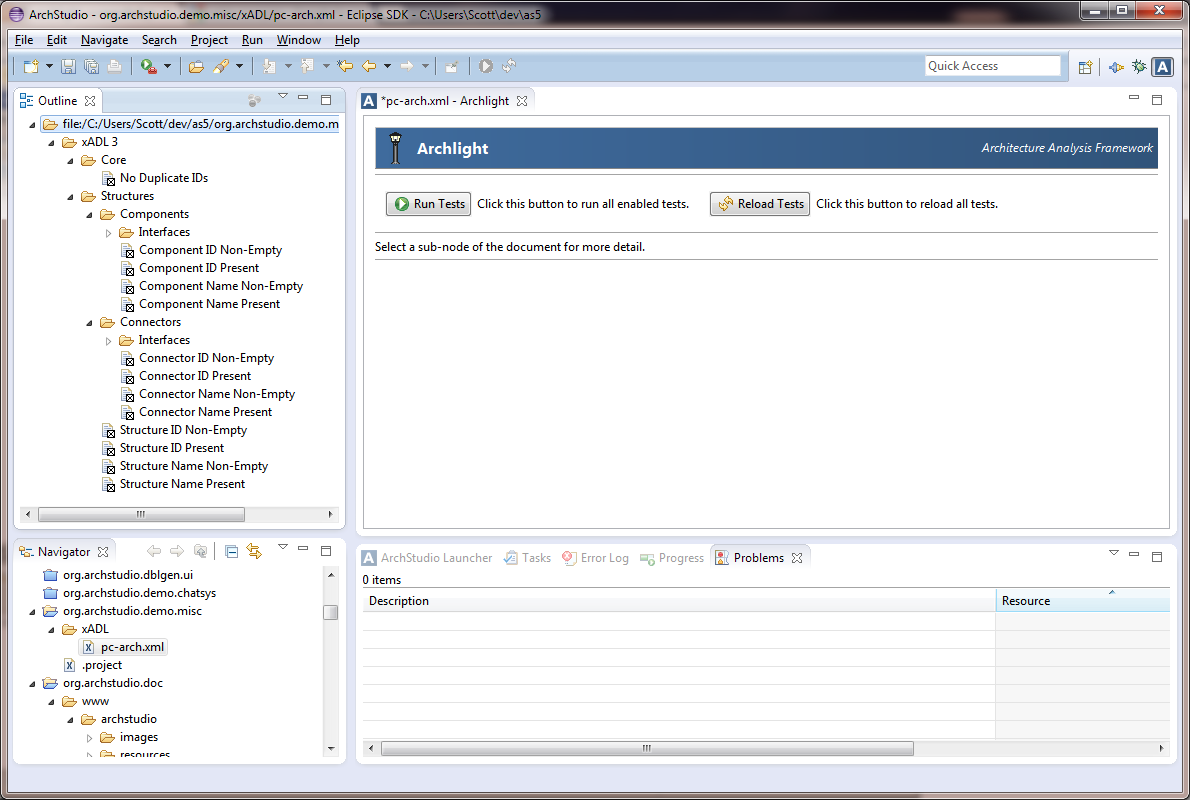


Навигирането в описанието на архитектурата става чрез свиване и разширяване на дървото в левия панел. Когато са избрани възли, атрибутите и елементите, които подлежат на редакция или съдържат данни за архитектура се появяват в десния панел.

ArchEdit може да бъде отворен от началния екран на ArchStudio или от контекстното меню, зареждащо се при клик с десен бутон на мишката върху произволен xADL документ и избиране на ArchEdit от подменюто „Отвори с...“.

* 1. Archlight

Archlight e фреймуърк от компоненти за ArchStudio. Графичният потребителски интерфейс на Archlight е разделен на четири основни зони. Вляво се намира така нареченият Outline. Той съдържа йерархично организиран набор от тестове, който могат да бъдат изпълнени на текущия документ. Edit панела вдясно от него. В него се показва информация за избраните в момента тестове, както и бутони активиране или деактивиране на тези тестове. Долният десен панел съдържа информация за евентуални възникнали проблеми при изпълнение на тестовете.



1. xADL
   1. Основни характеристики на xADL

Софтуерната архитектура помага на хората да разберат по-добре структурата на дадена софтуерна система. В последните години се забелязва огромно увеличение в сложността и обема на системите. Тава прави опесанието на тяпната архитектура по-трудно. За тази цел са създаденимножество методи. Те са наречени Architecture Descriptive languages(ADLs). Много от тях обаче са специализирани за определена архитектура и не са популярни. xADL е създаден за описание на различни типове архитектури. Езикът за описание на архитектури представлява абстрактно описание на софтуерната система използвайки компоненти, конектори, връзки и други. ADL предоставя на потребителите конкретен синтаксис, формална семантика и концептуална рамка. Повечето езици за описание на архитектури са специализирани за определени системи.

ADL - Supporting Domain

Darwin, C2SADL - Dynamic systems

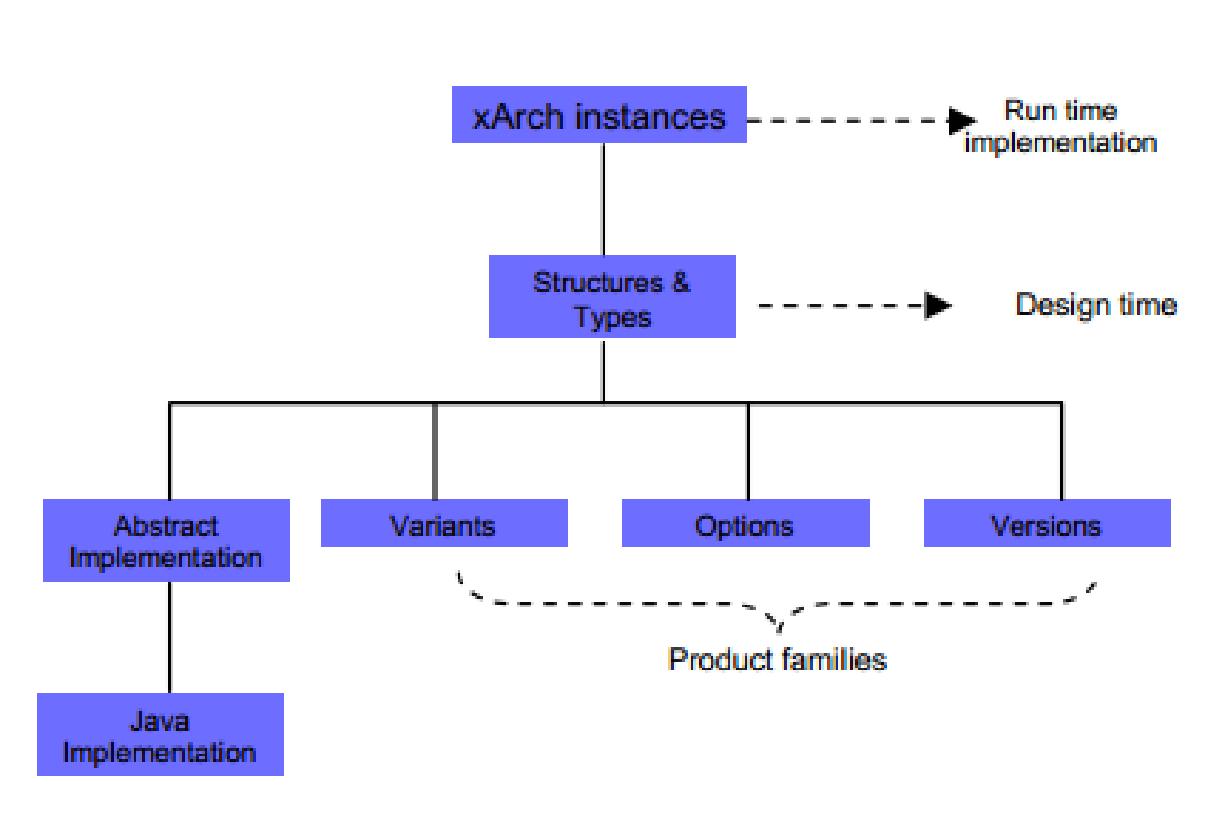
Mae - Configuration management

Koala - Product-line Families

Rapide - Event oriented system

Wright - Behavioral properties

Горните ADL-и са толкова специализирани, че когато се пяви нова архитектура като например мобилните системи, никоя от тях не би могла да се използва. Решение на този проблем е да се създаде език, при който могат да се създават разширения за описание на нови архитектури. xADL е език за описване на архитектурите на софтуерни системи, разработен от Institute for Software Research от University of California, Irvine. Езикът xADL не е обвързан с никой определен архитектурен стил, инструментариум или методология. xADL и инструменти като Apigen и xArch/xADL Data Binding Library могат да бъдат използвани самостоятелно спрямо други среди за разработка. Въпреки това, много инструменти от xADL са интегрирани в ArchStudio Architecture-centric Development Environment. Хората, които се интересуват от xADL би следвало да свалят Archstudio заедно с някои основни xADL инструменти. За разлика от много други ADL-и, xADL е дефиниран като множество от XML схеми. Това дава на xADL безпрецедентни разширимост и гъвкавост, както и основна поддръжка от много от достъпните XML инструменти. Текущото множество от xADL схеми включва поддръжка за моделиране на run-time и design-time елементи на системата, поддръжка на архитектурни типове, сложен конфигурационен мениджмънт като версии, опции и варианти. Включва и product family architectures и architecture diffing. Планирано е да се разшири това множество в най-скоро време. Потребителите на xADL са свободни да избират елементи от това множество и да създават свои разширения, когато им потрябват. XADL също така е и приложение за xArch – основна XML схема, дефинирана от UCI и Carnegie Mellon University.



**Figure 3 Абстрактно представяне на xADL схема**

* 1. Предимства на xADL пред UML

UML диаграмите са трудни за промяна. Те не са лесни и за разширяване или други манипулации. Не можем да моделираме архитектури базирани на продуктови линии. UML използва деаграми на обекти вместо диаграми на компоненти. Основната задача на ADL е да направи някакъв начален анализ на дизайна, но UML няма никакви добри инструменти за анализ. Използването на UML диаграми не могат да генерират код, които е основна част от ADL. UML не може да покрие всички домейни и ако се появи нов такъв не може да се разшири. xADL взима идеи от ACME. Разработен е като модулен език, който лесно да се разширява. Състои се от колекция от модули написани на XML. xADL прави лесни за анализиране софтуерните архитектури. Използвайки го, ние можем да генерираме код за тази архитектура.

* 1. xArch

xArch е стандартна, разширима чрез XML репрезентация на софтуерни архитектури. xArch поддържа стандартни XML нотации за архитектури, които могат да служат за просто представяне на архитектура на софтуерна система, начална точка за по-сложни XML-базирани архитектурни нотации или механизъм за опесание на архитектури. xArch предоставя стандартно основно множество от инкструменти, които да бъдат използвани за описание на системна архитектура. За последните 15 години разработки в областта на софтуерните и системните архитуктури са създали голямо множество от езици за описание на софтуерни архитектури. Съществиват различни нотации за архитектурите. Въпреки всичко, архитектура е обширно панятие и всеки домейн или проект воже да се нуждае от различни методи за нотация. Поради тази причина не е възможно да има единствен най-добър език за опесание на софтуерна архитектура.

* 1. Използване на xADL в проект

Използването на xADL в преокт означава интегрирането му в процеса на разработка. Заради различията между софтуерните системи. За разлика от други нотации, синтаксиса на xADL и инструментите му са предназначени да са максимално гъвкави за удовлетворяване на различните нужди на архитектурите. Езикът xADL може да създава схеми моделиращи проекта. Използвайки Apigen може да се генерира нова data binding библиотека, която да включва нови потребителски разширения. Инструменти базирани на синтакса като ArchEdit на ArchStudio могат да разширяват описанията чрез нови данни в определен формат. Много xADL потребители усещат процеса за разширине като тромав и искат в разширенията да могат да се използват произволни двойки име-стойност. Въпреки че това би улеснило добавянето на нови данни в описанието на архитектурата, това предложение бива отказано от разработчиците на xADL поради редица причини. Позволяването на произволни атрибути означава да се премахне това, което е постигнато от единния стандартизиран език. Синтаксисът на xADL е създаден да не бъде прекалено формален, но губейки формалността му ние ще загубим неща като неконтролируема фрагментация на езика, невъзможност да се валидират свойства, загуба на синтактични инструменти и ше се насърчава encodings-within-encodings и overloading. Неконтролируемата фрагментация на езика би позволила на потребителя да вкарва произволни данни в xADL документи, което на практика означава, че потребителя ще създаде свой xADL диалект, който е недокументиран и несъвместим с други xADL диалекти. XML валидатори базирани на схеми могат да проверяват xADL документа за синтактична коректност, но само ако има схеми. Допълнителните свойства правят неработещи дори най-простите валидатори. Apigen и ArchEdit са примери как ситнактично-ориентираните инструменти могат да подобрят средата за разработка. Въпреки това, тези среди могат да предоставят малко или никаква помощ, ако всичко, което знаят е това, че елементът има свойства състоящи се от имена и стойности. Присъствието на произволни свойства прави много удобно да се съставят някакви структурирани данни. Тази работа обаче е много по-добре да се представи като кодиран XML. Този проблем е виден при xADL дескриптивните елементи като Descriptions, които са overload-нати да имат различни допълнителни семантики. Разбира се нито един език не би бил достатъчно добър за всички нужди и затова xADL може да бъде разширяван със свойства, които са дефинирани от потребителя. Но както е описано по-горе трябва да се внимава с избирането на разширенията, които ще се правят.

* 1. Начин на работа с xADL

UCI са разработили няколко инструмента за работа, които помагат на разработчиците да създават, променят и манипулират xArch документи и схеми. Пример за такива инструменти са Apigen(написан на Java, които генерира Data Binding Library за текущи и нови схеми) и Data Binding Library – Java библиотека за създаване и манипулиране на xArch и xADL документи. Текущият списък с всички Archstudio схеми може да се намери от Navigate | Open Resource → “\*.xsd“. Това ще идентифицира всички xsd схеми в ArchStudio. Важни често срещани схеми са

archlight-3.0.xsd

domain-3.0.xsd

hints-3.0.xsd

implementation-3.0.xsd

statechart-1.0.xsd

structure-3.0.xsd

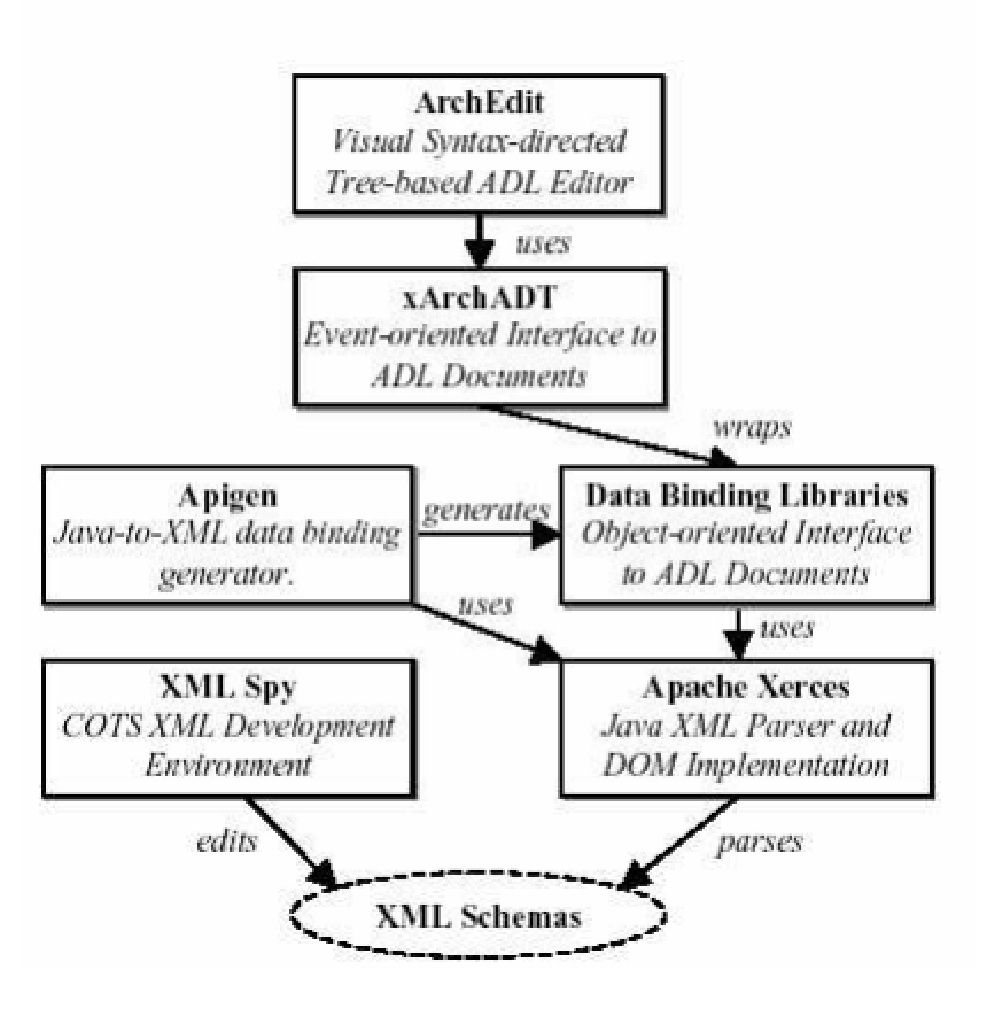
xadlcore-3.0.xsd

xlink.xsd

За създаване на нови схеми се използва Apigen инструментът.

* + 1. Apigen

Apigen е XML data binding генератор за xArch схеми, които автоматично генерира data binding библиотеки за XML схеми. Поддържа голяма част от XML езикът. Apigen е част от Archstudio и е използван да генерира data binding библиотек за xADL схеми. Apigen е част от стандартната инсталация на Archstudio. В Archstudio схемите си съхраняват в отделни Eclipse plugin-и. Apigen автоматично генерира data binding библиотеки за схемите, съхранени в тези plugin-и.



**Figure 4 Връзка между отделните инструменти**

За да се създаде Eclipse plugin трябва да се изполнят следните стъпки:

Избира се File | New | Other ...

"Plug-in Project"

Next

Въвеждаме име ва проекта

Next

В Options, премахваме "Generate an activator, ..."

В Options, премахваме "This plug-in will make contributions to the UI"

Finish

Ако се появи "Open Associated Perspective", избираме Yes

Резултатът ще е нов plugin с определеното име. За да направим генерирането на xADL Data Binding библиотеки трябва да информираме Archstudio-то, че трябва да ги генерира. За да го направим изпълняваме следните стъпки:

Десен бутон на проекта

Избираме Archstudio → Generate xADL 3 Schema DBLs

На излязлото warning съобщение избираме OK. След изпълнение на тези стъпки се появява папка model в проекта. В нея слагаме xADL схемите. Тази папка се сканира за XML схеми и тези файлове автоматично се трансформират в data binding библиотеки в папката src на проекта.

* + 1. Създаване на схема

Всички Archstudio проекти с „xadl3“ в името си имат схема в папката „model“, който може да служи за пример. Ако Archstudio не може да конвертира схема в Data binding библиотека, ще изведе съобщение в Eclipse Problems view-то с детайли за проблема. Това view може да се види от Windows | Show View... менюто. Официалните xADL схеми се подчиняват на следните правила:

За import на Ecore схема:

xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"

Major и minor номерата на версиите се включват в името:

xmlns="http://www.archstudio.org/xadl3/schemas/structure-3.0.xsd"

в декларацията на пакета:

ecore:package="org.archstudio.xadl3.structure\_3\_0"

и в namespace префикса:

ecore:nsPrefix="structure\_3\_0"

Елементите включват core extension point:

<xs:element name="ext" type="core:Extension" minOccurs="0" maxOccurs="unbounded"/>

Линковете показват destination типа:

<xs:element name="innerStructureLink" type="xs:IDREF" ecore:reference="Structure"/>

Ако схемата се трансферира в data binding библиотека, но тази библиотека не се компилира трябва да се вземат впредвид неща като:

Всяка свързана схема се влага в текущата схема използвайки <cs:import ...>

Plugin-ът трябва да разчита на plugin-ите, които съдържат текущата схема.

Например схема structure-3.0 включва в себе си xadlcore-3.0 схемата и неините plugin-и разчитат на org.archstudio.xadl3.xadlcore plugin-а, който съдържа xadlcore-3.0 схемата.

* + 1. xArch/xAdl Data binding библиотеки

xADL data binding библиотеките се съдържат в plugin-ите на техните xADL схеми. Как точно е описано по-горе. Положителната страна на този метод е това, че нова схема може автоматично да се инкорпорира в ArchEdit на ArchStudio-то, когато се компилира с необходимите plugin-и. Data binding библиотеката е изградена на основата на Eclipse Modeling Framework (EMF). Има три начина за използване на тази библиотека- автоматично (в ArchEdit) използвайки Archstudio, чрез разширение на ArchStudio или директно, от приложение. Archstudio автоматично инициализира всички data binding библиотеки. Ако се използва библиотека извън ArchStudio, трябва да се инициализира ръчно. Това става като се нагласи eINSTANCE променливата от генерирания Epackage. Например следният код инициализира библиотека:

org.archstudio.xadl3.structure\_3\_0.Structure\_3\_0Package.eINSTANCE;

1. Библиография

<http://isr.uci.edu/projects/xarchuci/>

<http://isr.uci.edu/projects/xarchuci/tools-apigen.html>

<http://isr.uci.edu/projects/xarchuci/tools-dbl.html>

<http://isr.uci.edu/projects/archstudio/xadl2.html>

Software visualization Stefan Diehl

[Kot00] “xADL – A Better way to Describe Architecture”