iArch仕様書

V0.1

改訂履歴

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | 日付  (改訂者) | 改 訂 内 容 | 改訂理由 | 備考 |
| 0.1 | 2015/02/26 (艾迪) |  |  | 引き継ぐ為 |

1. はじめに
   1. 本書の目的

本書はiArchの既存も機能仕様を説明するために作られている。新参者でもiArchの開発に参加できるよう、想定して、各モジュールの仕様を詳しく説明して、参加できるような型にしたい。よって、次の編集者も出来ればこの方向で内容を追加していくことが望ましい。

実装方法については、艾迪の修士論文の「ツール実装」部分を参考できる。

* 1. 用語定義

|  |  |  |  |
| --- | --- | --- | --- |
| 用語 | 読み | 説明 | 出典 |
| iArch | アイーアーキ | 本書が対象とするツールの名前 | 本書 |
| Archface |  | ドメイン固有言語Archface | 鵜林先生 |
| Archcode |  | Archfaceの言語で書かれたコード | 本書 |
| Archfile |  | Archfaceのコードを保存するファイル | 本書 |
|  |  |  |  |

* 1. 記号定義

|  |  |  |
| --- | --- | --- |
| 記号 | 例 | 説明 |
| オレンジ色の文字 | モデル | 未定義の言葉だが、この後で直ぐ説明する。 |

1. クラスの説明

この部分は各クラスについて説明する。

* 1. DSL部分

一番目のプロジェクトjp.ac.kyushu.iarch.archdslは、Archfaceを定義しているプロジェクト。

Xtextのプロジェクトを作成する時、自動的に

* jp.ac.kyushu.iarch.archdsl
* jp.ac.kyushu.iarch.archdsl.sdk
* jp.ac.kyushu.iarch.archdsl.tests
* jp.ac.kyushu.iarch.archdsl.ui

この四つのプロジェクトを作成する。中に

jp.ac.kyushu.iarch.archdsl/ArchDSL.xtext

はDSL定義するファイル。

このファイル内でDSLを定義すれば、Xtextは自動的にDSLのメタモデル、ソースコード、テストコードなどを生成しれくれる。

Gitで扱う時、自動生成部分が混入すると、実際に実装した部分を探しにくくなるため、.gitignoreファイル内自動生成コード全部追加して無視するようにした。

　Git cloneする時、自動生成部分が欠けているせいで、コンパイラエラーを吐き出される。この場合、jp.ac.kyushu.iarch.archdsl/ArchDSL.xtextを開いて、

run->Generate Language Infrastructureを実行すれば、コンパイラエラーは解消するはず。

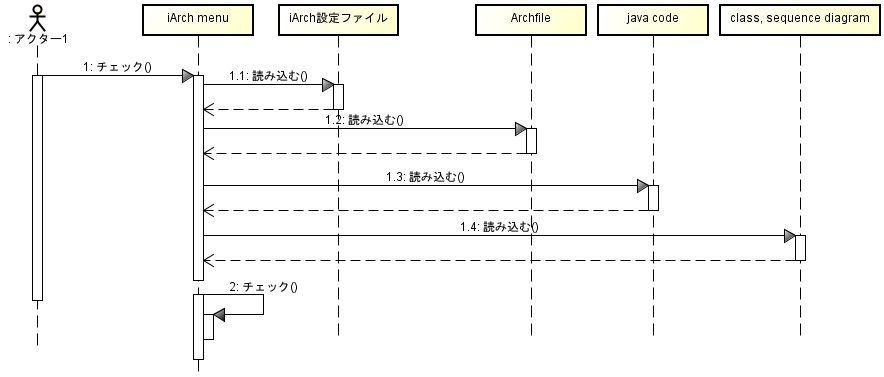
* 1. jp.ac.kyushu.iarch.basefunction

名前の通り、基層機能を実装しているプロジェクト。

機能として

* + 1. **jp.ac.kyushu.iarch.basefunction.reader**
* XMLreader

iArchの設定ファイルを読み込むクラス、イメージは以下のシーケンス図の通り



* ProjectReader

現在操作中のプロジェクトを取得するクラス。このクラスは後期で追加した機能

非常に便利である。例えばiArchメニューから何かを実行する時、いつも「プロジェクトを選択してください」というメッセージが出てくるのだが、このクラスを使えば、解消できる。

　使い方は直接ProjectReader.getProject();を呼び出す。

* ArchModel
* ArchfileからArchcodeを読み込み、モデルに変換してくれるクラス。モデルとは、Xtextで定義したDSLから、自動生成したメタモデルのインスタンス（メタモデルが定義しているモデル）。ArchfileはXMLreaderから取得できる。
  + 1. **jp.ac.kyushu.iarch.basefunction.model**

このパッケージはダイアグラムからビジネスモデルを取り出すクラスである。

ダイアグラムには、ビジネスモデルとダイアグラムモデル２つのモデルがある。例えばclass図エディタ上でクラスAを描いた時、ビジネスモデルとして、class図のメタモデルで定義したumlClass.Classのインスタンス自動的に一個作られる。それと同時に、ダイアグラムモデルとして、GraphitiのContainerShape（長方形を表すクラス）も一個自動的に作られて、更に２つの間で、GraphitiのpictogramLinkも自動的に作られて、両者を結びつけている。pictogramLinkもダイアグラムモデルに保存している。

* + 1. **jp.ac.kyushu.iarch.basefunction.controller**

上のmodelパッケージから取り出したモデルを、編集したい時はこのパッケージを使う。

暗黙的に編集した結果をそのままファイルに保存する。

使い方として、上のmodelパッケージを使わずに、直接controllerからファイル(Resource)を読み込んで、編集メソッドを呼び出す。

以上のパッケージ以外に、iArchが扱っている外部のライブラリも全部、Libフォルダに収納して、管理している。

現在扱っている外部ライブラリ：

* dom4j-1.6.1.jar

iArch設定ファイル（XMLファイル）を解析用

* jaxen-1.1.6.jar

iArch設定ファイル（XMLファイル）を解析用

* slf4j-api-1.7.9.jar

ログ出力

今後、これらのライブラリはMavenで管理できることが望ましい。

* 1. jp.ac.kyushu.iarch.checkplugin

このプロジェクトはiArchの核心にある。

* + 1. jp.ac.kyushu.iarch.checkplugin
* Savehook

earlyStartup()はEclipseを立ち上げる時、最初に実行する部分。

ここはIResourceChangeListenerを追加している。

リソースが変わった時（ファイルが変更された時）、

**public** **void** resourceChanged(IResourceChangeEvent event)

は実行されて、visitorパターンを使って変更したファイルの情報を収集していく。

一定の条件を満たすと、ArchfaceChecker(jp.ac.kyushu.iarch.checkplugin.handlerパッケージ内のクラス)を呼び出して、トレーサビリティチェック及び抽象度メトリクスを計算する。

* + 1. jp.ac.kyushu.iarch.checkplugin.handler
* AbstractionRatioChecker

抽象度メトリクスを計算するクラス

* AbstractionRatioController

抽象度メトリクスの計算結果を（履歴として）CSVファイルに保存したり、表示したりするクラス。

* ArchfaceChecker

自動チェックはここのcheckProject()を呼び出している、全ての始まり。

* ARCSVController

山本が実装したもの。抽象度メトリクスの計算結果をCSVファイルに保存するクラス。

今は必要ない。

* ASTSourceCodeChecker

ASTParserを用いてJavaコードを解析，結果をXML(codeXML.xml)に出力後そのXMLを解析し，JavaコードとArchfaceのチェックを行う．

* CheckArchfaceHandler

もともとトレーサビリティチェックを実行させるためのクラス、自動チェックの実装に伴い、ArchfaceCheckerに変わってしまった。

* CheckARHandler

もともと抽象度メトリクスを計算するためのクラス、自動チェックの実装に伴い、

AbstractionRatioCheckerに変わってしまった。

* CheckASTHandler

もともとASTをチェックするクラス。自動チェックの実装に伴い、ASTSourceCodeCheckerに変わってしまった。

* CheckerWorkSpaceJob

自動チェックを別スレッドで実行させるためのクラス。Singletonパターンを使っているのは、スレッドを立ち上げすぎないするためである。

* ClassDiagramChecker

名前の通り

* codeToArchface

コードからArchcodeの自動生成。

* ConfigurationXMLHandler

iArchを設定するためのクラス。iArchメニューから直接実行できる。

* GenerateArchCode

ダイアグラムからArchcodeを自動生成するクラス。データフロー図からのArchcodeを自動生成する時、このクラスを使っていい。

* ProblemViewManager

トレーサビリティチェック結果をコンパイラエラーとして出すためのクラス

* SelectPoint

Slicingルールの適用、iArchメニューから直接呼び出せる。

* SequenceDiagramChecker

名前の通り

灰色の部分はシャビが担当者となっている。ドキュメント不足かつ改良が望ましい部分である。

* + 1. jp.ac.kyushu.iarch.checkplugin.model
* AbstractionRatio

抽象度メトリクスを計算するためのデータを保存するクラス

* ArchfaceModel

ArchModelクラスと同じ

* + 1. jp.ac.kyushu.iarch.checkplugin.utils
* ProjectSelectionUtils

プロジェクト選んでない時、iArchメニューから何かを実行させて、「Please Select Project.」のメッセージを出すためのクラス。

* + 1. jp.ac.kyushu.iarch.checkplugin.view
* AbstractionRatioViewPart

抽象度メトリクスをUIに表示するためのクラス。

* SelectAllFileDialog

iArchを設定する時に、ファイル一覧をUIに表示させるためのクラス。

以下同じ

* SelectArchfile
* …
  1. jp.ac.kyushu.iarch.model

ダイアグラムのメタモデルと、それの自動生成したソースコードの保存場所。

ファイルを開けない場合は[こちら](https://github.com/templarad/iArch/wiki/Eclipse%E3%81%AE%E8%A8%AD%E5%AE%9A)を参考

* 1. jp.ac.kyushu.iarch.classdiagram

詳細はGraphitiのドキュメントを参照

[API Help (Graphiti Documentation)](https://www.google.co.jp/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCQQFjAB&url=http%3A%2F%2Fhelp.eclipse.org%2Fluna%2Ftopic%2Forg.eclipse.graphiti.doc%2Fjavadoc%2Fhelp-doc.html&ei=gs3uVJykINPv8gWNiYHIBw&usg=AFQjCNEtzvz62fqIXFPb2WdBwwc33uwTdQ&sig2=n5I_7T3xDXj1iVbSG_b5LA&cad=rja)

* 1. jp.ac.kyushu.iarch.sequencediagram

詳細はGraphitiのドキュメントを参照

[API Help (Graphiti Documentation)](https://www.google.co.jp/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCQQFjAB&url=http%3A%2F%2Fhelp.eclipse.org%2Fluna%2Ftopic%2Forg.eclipse.graphiti.doc%2Fjavadoc%2Fhelp-doc.html&ei=gs3uVJykINPv8gWNiYHIBw&usg=AFQjCNEtzvz62fqIXFPb2WdBwwc33uwTdQ&sig2=n5I_7T3xDXj1iVbSG_b5LA&cad=rja)