AETP/AEFP Documentation for (For Java & Dotnet)

Introduction

We have identified the ideal ratio of AETP vs AEFP on the basis of the historical analysis of applications on-boarded. We have identified various Technology stacks (for e.g. N-tier, client-server etc). Candidate Applications will be tagged into identified Technology stacks. An application will be either in one or other stack, if an application is falling under multiple stacks then it will be tagged into the best matching stack on the basis of code % of particular technology (for E.g. if an N-tier application has mobile technology source code then manually on the basis of code coverage it will be identified if this application will be tagged into N-tier or Mobile Stack.)

Purpose

The purpose of this documentation is to handle High AETP vs AEFP ratio and to provide justification for AETP Objects.

End Users.

AIA

How to Use Documentation

Identified Technology Stacks << Identify Stack>>

1. N-Tier Stack
2. Messaging driven Stack
3. Microservices Stack
4. Mean Stack
5. SOA(REST / SOAP /Webservice) Stack
6. Mobile Stack
7. Client-Server Stack

Recommendation on ratio

- N-Tier Stack: 65 - 35 (On the basis of 15 applications)
- Messaging driven Stack: 45 - 55 (On the basis of 5 applications)
- SOA(REST / SOAP /Webservice) Stack: 50 - 50 (On the basis of 6 applications)
- Microservices Stack: 70 - 30 (Generic Ratio)
- Mobile Stack: 60 - 40 (Generic Ratio)
- Mean Stack: 60 - 40 (Generic Ratio)
- Client-Server Stack: 60 - 40 (Generic Ratio)

Recommendation for General Reasoning - (Attachment for Dataset - Here)

- **These artefacts are valid AETP.**
    - Html5 CSS Source Code fragment & CSS Files can be ignored.
    - If the parent method is not part of the transaction then Framework related artefacts ToString, Hashcode, Equal etc will be part of AETP.
    - Generated Code related to XSD will be part of AETP.
    - Dead Code.
    - All artefacts related to static data screens including console output, Header, Footer will be part of AETP.
    - If artifacts are coming from Designer files (designer.cs or designer.vb) or Reference.cs or Reference.vb files.
    - Auto-generated artefacts from wsdl to java using tools axis2.4- proxy classes.
    - Getter setter which are not part of either model or JPA or pojo classes will be part of AETP.
    - Artefacts of proxy classes generated from wsdl will be part of AETP.
    - Artefacts which are Called Dynamically will always be part of AETP for e.g. Implementation of abstract & factory Design pattern. **[we have to manually check this]**
    - If an implementation class of Interface is missing then methods of that Interface will be in AETP.
    - If Artifacts are doing only String operations than it will be valid AETP. **[we have to manually check this]**
    - In the case there is only a declaration of the artefact and no definition is there or code is commented inside the artefact, then the artefact is valid AETP. **[we have to manually check this]**
    - In case artefact has no source code only return statement is there than artifact is valid AETP. **[we have to manually check this]**
    - Build Script Artifacts which are used to Package/Compile code will be valid AETP. **[we have to manually check this]**
    - If abstract Method does not have any implementing class, then that artefact could be dead code and Valid AETP.
    - Method of Log, Error & Exception related artefacts will be part of AETP.
    - If EJB Methods are coming in AETP then they are valid AETP.

- Artefacts like login, logout, contact, test, cookie, splash, hello world will be valid AETP.

- **These will be the AETP artefacts which should be re-looked for new Entry & Endpoints so that these can be converted into AEFP.**
  - Artefacts which belongs to Data Access Layer, Repository layer, Controller classes, Service Classes & action classes.
  - If implementation class of an Interface is present in source code & Methods of that Interface are coming in AETP then we have to look for valid entry & endpoints.
  - If abstract Method has implementing class, then this artefact we have to look for valid entry and endpoint.

- **These will be the AETP artefacts which can be either in AETP or can be in AEFP depends upon Application Architecture and Frameworks used.**
  - Html5 Javascript Artifacts which are coming from Js file.
  - Artefacts which are constructors.
  - If an Artifacts does not have any callers and call from that artefact is going to valid endpoint then we can look at it from entry point perspective.- **[we have to manually check this]**

STEPS TO USE -:

1) Run the Query attached on <<Local>> database to identify the technology stack of the application. If an application has multiple stacks then manually de cide on the basis of code % which stack is applicable for that application. The output of the query would be like shown below. **Download Sql File**- **Indentify_stack.sql**

---

2) Run the Script file attached which will create a Function on Central Database.[[Replace <<Central>> With Application Central Database & Local Database name with <<LOCAL>>]]. Executing that function will give output as below in screenshot (refer to Output 2.1 , Output 2.2 & Output 2.3. **Download Sql File** - **AETP_PROC_WITH_REMARKS.sql**

Output - 2.1) It will have remarks for the artefacts which are valid AETP (Refer to below fig).

Output -2.2) It will have remarks "Look for possible Entry or Endpoints for following Artifacts ". Scan the source code manually for entry & Endpoints for these artefacts. (Refer to below fig)

Output - 2.3) It will have remarks "These artefacts can be AETP depending upon Application Architecture & Framework used). On the basis of technology or framework used in application decide whether these are AETP or not. (Refer to below fig)