# Analysis part 1

**8 + 2 Word lists: *Adaptor, Bean*, Concurrency, Graphics, GUI, IO, Math, Security, Sound, Xml**

## *Chukwa-0.4.0 ME, NB and HW1 comparison:*

In ***Chukwa-0.4.0*** there are 3 major clusters named ***src***, ***build*** and ***contrib***, all these 3 major clusters are depended or categorized as ***IO*** in **HW1** recovery, whereas in **NB** it is categorized as ***security*** or ***adaptor***, but in **ME** majority of the clusters don't match to any cluster but few are ***adaptors***.

Along with the above mentioned major difference, some other differences are:

a. *AbstractMatricsContext* which is a sub-component of *metric* is clustered as ***graphics*** in **HW1** whereas in **NB** it is ***security*** and in **ME** it is ***Adaptor***.
b. *DataLoadFactory* and *socketDataLoader* sub-components of *Dataloader* are clustered as ***IO*** and ***SQL*** in **HW1** whereas in **NB** they are ***security*** and in **ME** they don't match as I have not chosen ***SQL*** in wordlist.
c. *configuration* component with sub-components named *objects, actions* are clustered as ***graphics*** in **HW1** whereas in **NB** they are clustered as ***math*** and ***security*** and in **ME** they don't match with any chosen wordlist even though ***graphics, Math, Security*** are chosen.

In all 3 recovered architecture's accuracy varies for instance –

a. *QueueToWriterConnector* and *BackfillingLoader* which are sub-components of *Backfilling* was clustered as ***IO*** in **HW1** whereas in **NB** they are ***adaptor*** and in **ME** they don't match, here **HW1** looks more accurate.
b. *Visualisation* sub-components like *heatmap* and *swimlane* are ***graphic*** clustered in **HW1** whereas in **NB** they are ***GUI*** and in **ME** it is ***Adaptor***, here **NB** is more accurate.
c. *Log4J* component is clustered as ***Adaptor*** in **ME** but in **NB** they are ***security*** and in **HW1** they are ***IO***. However, even if we consider this as ***IO*** as in **HW1** but it would be more accurate if they were of no-match to any wordlist.

In **HW1** most of the components are clustered properly like *DataBase* sub-components like *DatabaseWriter* is clustered under ***IO*** and *DatabaseRestServerSetup* is clustered under ***Networking*** whereas in **NB** *DatabaseRestServerSetup* sub-component is categorized as ***sound*** and *dataBaseWriter* as ***security*** and in **ME** *dataBaseWriter*, *DatabaseRestServerSetup* is not categorized to any cluster which is acceptable compared to **NB** which has clustered under ***sound***.

Based on the above observation 3 recovery results are ranked as follows:

**HW1** ranked **1st**, **ME** ranked **2nd** and **NB** ranked **3rd.**

***Chukwa-0.6.0 ME, NB and HW1 comparision***

There are only 2 major clusters in Chukwa-0.6.0 they are ***src*** and ***contrib***, ***build*** is merged with ***src*** as *collectors* are deprecated from 0.6.0 version. In **HW1** majority of the clusters are ***IO*** and ***graphics*** whereas in **NB** it is ***security*** and ***adaptor*** but in **ME** most of them no-match and few are ***adaptor***.

Along with the above mentioned major difference, some other differences are:

a. Sub-components under *test* like *hicc*, *tools*, *database*, *inputtools* and *analysis* are clustered as ***IO*** in **HW1**, whereas in **NB** they are clustered as ***adaptor***, ***security*** or ***IO***. But in **ME** they are either not clustered to any choosen wordlist or they are ***IO***.
b. In **HW1**, *databaseLoader* is clustered as ***SQL*** and *database* subcomponents like *Tablecreator*, *macro* are ***Graphics*** and *DatabaseTrigger* , *Triggerevent* are clustered as ***network*** whereas in **NB** *databaseLoader*, *databasetrigger* and *tablecreator* all are clustered as ***Security*** and in **ME** *databaseLoader* doesn't match with any as I have not chosen SQL data list, *databasetrigger* are clustered as ***Bean***(EJB) and *tablecreator*, *macro* as ***adaptor***.

Based on the above criteria, 3 recovery results are ranked as follows:

**HW1** ranked 1st, **ME** ranked 2nd and **NB** ranked 3rd.

Accuracy of 3 recovered architecture varies as follows – **HW1** is more accurate than ME and NB because *database GUI* components are clustered as ***Graphics*** in **HW1** whereas in **ME** they are ***adaptors*** and in **NB** they are ***security***.

# ANALYSIS PART 2

All these 3 views of recovery result will help us in understanding the system's architecture and to get useful overview of the system, it is evident from the previous analysis that components are categorized to appropriate clusters, also the recovery technique provides human readable files like *dependenies, rsf and contains* text files using them we can figure out:

a. which component is doing what kind of job.
b. what component is dependent on which other components.
c. which component contains what packages.

For example, *Visualisation* component which is basically needed for graphics(UI) to view the data on screen it is clustered under ***graphics***, *DatabaseWriter* is ***IO*** clustered and *DataTrigger* is ***networking*** or ***Bean*** clustered. All these clusters show jobs of the components.

Also rsf files containing *"contain graphics edu.berkeley.chukwa_xtrace.CausalGraph*

*contain graphics edu.berkeley.chukwa_xtrace.XtrExtract,*

*contain graphics org.apache.hadoop.chukwa.ChukwaArchiveKey*" shows packages that are included in the system under different topics. Similarly, there are dependency files which gives us information on system's dependencies.

In terms of their utility, I would like to rank **NB** 1st as winner followed by **HW1** and **ME** because with the given set of wordlist, HW1 gives less categorization like ***graphics, IO, networking, SQL*** whereas in NB it gives little more categorization like ***adaptor, bean, IO, sound, security, xml, math*** but in ME it gives us very few categorization ***adaptor, bean, graphics, IO.***

Relax recovery method gives only dependencies based on class files, whereas it doesn't show data flow, message flow, any ambiguity of interfaces, inter or intra flow dependencies or behavioral transactions etc. Dependency and cluster files gives amazing depiction of the architecture based on class files and it has graphical visualization along with text for human understandability. If we can get Event or Data flow along with graphical visualization this recovery technique will help us understand the architecture more efficiently. From the recovered architecture, it is difficult to find out what kind of connectors or styles are used in this system but to some extent we can say based on dependencies it might have used layered style and a web-portal style interface for displaying data.

**ANALYSIS PART 3**

To understand about the architecture one should be aware of Structure, behavior, Interaction, design, implementation, architectural style used, connectors that are used, Non-functional properties of the system, components etc, In Relax recovery method number of topics/word list is finite, this can be extended to any number but it is tedious to generate wordlists.

From the Relax recovery method we will get structured arrangement of a system's implementation-level artifacts under a set of criteria. If we can integrate this with Event visualization approach to get the Event flow or DataFlow or message flow or ambiguities within the system, it would be helpful for the user to get depth understanding of the architecture. Once we get the data flow or message flow or dependencies among the producer and consumer of the system, we can also identify what kind of connectors that might have been used in the system which in-turn would be easy for a person who has not worked in this system before to understand the architecture and try to create a Domain Specific System Architecture(DSSA) for this system.

Furthermore, we also know that maintenance and evolution of software system is costly, to know about the recent change in the architecture we need to maintain this recovery system by continuous evolution for instance, when something new component or new topic is added to the system, user needs to generate wordlist/topic for that component and re-run the whole architecture recovery with new classifier as in most of the cases only Descriptive architecture is changed not the prescriptive architecture. To overcome this, if it's possible to run the architecture recovery only for specific part of the system and integrate with previously recovered architecture we can reduce the cost of re-run for the whole system.