ISSN (e): **2250 – 3005** || *Volume, 07* || *Issue, 06*|| *June – 2017* ||
International Journal of Computational Engineering Research (IJCER)

Categorization of application layer viewpoints in the EAM

[1]rajanikanta Sahu, [2]s B Soumyaranjan,
Gandhi Institute of Excellent Technocrats, Bhubaneswar,India
Sanjay Memorial Institute of Technology, Berhampur, Odisha, India

ABSTRACT
Since the last century companies are getting bigger and big- ger. Thus they have a more complicated structure and are difficult to manage. Enterprise architecture models help to provide better transparency and a clear view for all involved parties. These IT-based systems are good for visualizing business processes of a company. For the representation are among others methods of the software cartography used. In the software cartography there are many different models, which have their own advantages and disadvantages. This paperwillfocusonthecategorizationofthesedifferentsoft-waremappingtechniques.ItalsointroducesviewsfromTO- GAF. The main goal is to match software maps to TOGAF views, such that these views can be illustrated. This shall bedone in order to offer an overview for which use cases a map is patricularly efficient or if there is an other better fitting modelavailable.

INTRODUCTION

Nowadaysthenumberofcompaniesareincreasingreally fast,sodoestheirterncomplexity.Thecomplexityisre-flectedbythehighnumberofinformationsystemsanddif-ferentavailabletechnologies.Evenintodaysmedium-sized companies there is much information to be administrated. EnterpriseArchitectureManagement(EAM)isgettingmore importanttohandleallthebusinessprocessesandcompany- intern structures. Especially from bigger companies it is demanded to provide parts of these informationoutwards, becauseinvestorsorotherstakeholderswanttransparency.
Thereforealltheinformationhasnotonlytobestoredand administrated, but it also has to be visualized. The reason for that is that they, as onlookers can only understand the systemfromaclearoverview.Forthevisualizationofappli-cationlayersusuallysoftwaremapsareused.Therealready existmanytypesofthesesoftwaremapsandeachofthem hasitsownfeatures.Itdependsontheusecaseandtheac-tualstructureofthecompany,tosaywhichsoftwaremapis useful.Inthispaperthedifferenttypesofsoftwaremapswill

.
bepresented and their advantages as well as disadvanages will be discussed. As an example for an EAM framework the popular TOGAF will be presented. There will also be a categorization for these maps in order to give an overview of which software maps can be matched to which view of TOGAF.
The paper has the following structure: The next section will deal with the related work. The third section is devoted to the software cartography. After the software cartography section, software map types will be discussed. Then the TO- GAF and the terms views, viewpoints and stakeholders will be introduced. In the sixth section core views of TOGAF will be presented and they will be matched with specific software map types. At the end, results and possible future workwillbebroachedtoroundupthispaper.

1. RELATED WORK

Therearealreadymanypapersandtheseswhichthema- tizesoftwarecartography,buttheyallhaveadifferentview on this topic. The ones I used to get information from,are moreaboutthesoftwarecartographyitselfandfocusedon the aspect of how to represent information on a software map.

Buckl writes in her paper about how to generate visual- izations of Enterprise Architectures with the help of model transformations. Thereby she mentions software cartogra- phy and techniques to visualize enterprise information con- tent with it.[1] Lankes wrote two papers about software cartography, the first one adressing the visualization of ap- plication envirohnments, and the other one with the title "architecture description of application environments". [5]
[4] Matthes has a short paper about software cartography in general, whereas Wittenburg has a much more in depth going PhD thesis about this topic. [6] In his PhD thesis Wittenburg goes more into detail by explaining in extensive chapters the theoretical background information and man- agement of applicationenvironments.[12]

Therefore I picked the TOGAF as one popular example to examine particular steps of visualizing parts of the EAM. The TOGAF has an excellent documentation, where some key parts of the framework are explained. So, I picked most of the information about TOGAF from the documentation website of The Open Group. [10]

2. SOFTWARE CARTOGRAPHY

Many aspects of the software cartography are adapted from the original principles of cartography. According to the Wikipedia definition cartography is "Combining science, aesthetics, and technique, cartography builds on the premise that reality can be modeled in ways that communicate spa- tial information effectively." By applying the software car- tography in conjunction with EAMs, it deals as a bridge to represent the abstract information about structure and pro- cesses of a company into a clear graphicalview.

The software cartography is not only associated to cartogra- phy, but it has its roots also in the computer science and the economic science. Computer Science and economic science have in common, that both heavily work on project man- agement, which is a central part of software cartography. In computer science, especially software systems engineering with its models for respresenting structures and informa- tion in form of diagrams (like UML), are important. Also the process management from economic science is relevant for software cartography. Finally cartography completesthe

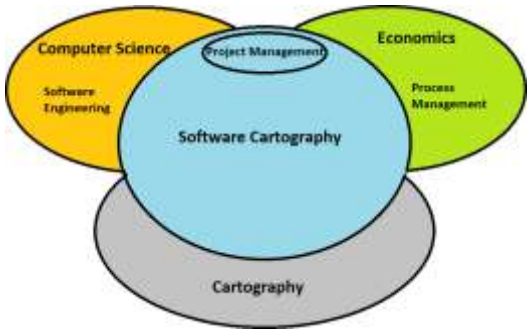

Figure 1: Software Cartography as an Intersection of three Sciences [6]

triple with its methods to create structured maps by utiliz- ingcolours, forms etc. Figure 1 shows that the software cartography indeed fits in as an intersection of all of these three different sciences. [6]

Figure 2: Pyramid of Software Cartography [5]

Each software map has the purpose to give an insight into one part of business processes or parts of the application en- vironment. This shall be achieved by connecting the three important viewing planes of the application environment in order to provide a good representation. These three viewing planesareshowninthepyramidoffigure2.Firstthereisthe ground layer, with information systems, protocols and inter- faces, which all contain important information about how things work in a company. According to this, the ground layer is built all around the question ”How things are done in the company?”. The layer in the center deals with the busi- ness processes and company data models. From an abstract point of view these represent the core actions of a company. Therefore the fitting question to this layer is ”What does the company do?” Last but not least the spire of the pyramid features company goals, strategies, laws as well as business rules. Basically this contains the goals and also reasons for the actions of a company. The content of the spire can be summarized under the question ”Why does the company do something?”. A good designed software map has to have an intuitive representation and it should also connect all of the three shown aspects with each other. [5] [6][4]

3. SOFTWARE MAPTYPES

In computer science there already exist different models to represent data structures and information systems. These are amoung other: the UML diagrams and the Entity Rela- tionship model. It is possible to use these types of software maps also to model parts of an EAM. Nevertheless most of the time the power of these visualization models is not enough to represent the whole EAM or bigger parts of it. But as it will be discussed later, these known models can beused as an addition to the other existing software map types. In the rest of this section, some of the relevant UML diagrams and the four basic software map types will be in- troduced. For some of these map types there will also be an exampleshown.

In general it is hard to fit a whole application environment onto a map. Software cartography makes use of techniques from the cartography itself to visualize information. This is not always as easy as in the cartography due to the fact that methods of cartography are originally designed for ge- ographical maps. Especially the limitation of space is one of the biggest challenges for a software map. In comparison to georgrapahical maps, software maps are not built on a topographical basis, which makes it much more difficult to visualize information. Each software map type has its own way to handle this challenge. One important characteris- tic to distinguish software map types is, if they have a base map for positioning or not. Having a base base map for po- sitioning means, that the position of certain objects on the map has a semantical meaning, e.g. if two objects nearby belong together or are associated in a way. [12] In figure 3 on the next page the classification is shown. With the ex- ception of Graph Layout maps, all other maps are based on the principle of base mappositioning.

UMLDiagrams

The Unified Modeling Language (UML) is collection of diagram types, which allow to visualize software archictec- tures or software structures. Each of these diagram types visualizes a different aspect of the software. There are three

Figure 3: Software Map Types [12]

UML diagram types, I will shortly introduce in this section, since the main focus is on the other software map types.[2]

ActivityDiagrams
Anactivitydiagramgraphicallydescribesasetofactivi- ties for a software. They are considered as statediagrams, thereforetheybeginwithaninitialstateandendonafinal state. There are also states in between. Every state has a namethatdescribesthestateinwhichthesoftwarecanbe. Thestatesareconnectedtogetherbyarrows,whicharethe transitions.Thesetransitionsrepresentthesequentialorder of performed activities. Within an activity diagram there can also be decisions, where it has to be decided in which direction the control flow shall continue. In short, the ac- tivitydiagramsrepresentforasetofactions,everypossible state that the software can be in. They also show if a se- quenceofactivitiesleadstotheintendedresult.Thismakes themsuitableinvisualizingthedataflowinasystem.[11]

SequenceDiagrams
Sequencediagramsareusuallycreatedforonechosensce- nario.Insequencediagramseachinvolvedobjecthasaclass, to which it is assigned to. Every object has its own lifeline belonging to it. These objects perform actions, which can involve one or more objects. An action can be communicat- ing with one or more other objects. But it can also be the case that an object does something on its own. The action inasequencediagramfollowssequentiallyonecontrolflow. Therefore there are no parallel actions allowed in Sequence diagrams. All in all a sequence diagram has the task to represent in a graphical way, how the objects communicate with each other, such that the data flow between software components can be comprehended.[3]

Use CaseDiagrams
ThisUMLdiagramtypefocusesmoreontheuserinter- action with the system. It shows what possibilities a user has,tointeractwiththesystem.Hereforetheusersarecat- egorized, e.g. administrators, customers, etc. All of them can perform different actions. Every action the user can perform, is called a use case. Hereby the technical imple- mentation of the actions does not play a role, because it is not relevant for the user interaction. Figure 4 shows an exemplary use case diagram.[8]

Clustermap
Clustermapshavethepropertythatobjectscanbegrouped together.OnaClustermap,anumberofobjectsoftheorga- nization can be placed nearby, such that together they form a logical unit. Most of the time these logical units are also in a frame. Cluster maps make use of it to representfunc-

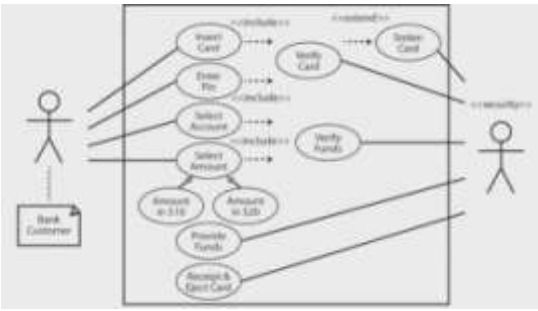

Figure 4: Use Case Diagram [12]

tional units, organization units or even geographical units like e.g. different locations, regions and cities. To hightlight the unity, most of the time the frames are coloured and logi- cal units also have a capture. If one object has to occur more than once within a software map, belonging to two different logical units, then it appears two times on the map, once in each logical unit. An example for a Cluster map can be seen below in figure 5. [1] [6] [12]

Figure 5: Example of a Cluster map [12]

Cartesianmap

A Cartesian map is simply defined by a sofware map that has a base map with an x- and y-axis. The Cartesian map is abstract and only descirbes the layout principle of a map. How the actual map looks like and further attributes are de- fined by the one of the two following map types. The most popular representatives of Cartesian maps are the Process Supporting map and the Time Interval map. Both are de- scribed in the coming two sections. [1] [6][12]

Process Supportingmap

A Process Supporting map focuses on visualizing linear business processes. It is particularly effective to use a Pro- cess Supporting map, when the workload of one business process is distributed onto several organization units. Pro- cess Supporting maps have like every Cartesian map an x- and an y-axis. On the x-axis there are different processes stringed together, whereas on the y-axis the organization or executive units are. Because there could be many differ- ent business processes in a company, the primary process, respectively the one requiring the most organization units decidesaboutthepostionoftheunits.They-axiscanalso

justcontainlocationsofthecompanyorevendifferentprod- ucts the company has to produce. This makes the Process Supporting map very versatile. [1] [4] [6][12]

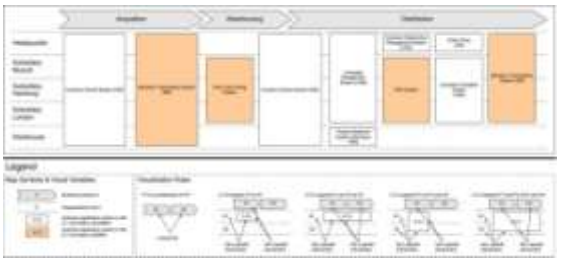

Figure6:ExampleofaProcessSupportingmap[12]

Time Intervalmap

Time Interval maps are very similar to the Process Sup- porting maps. Like in Process Supporting maps the y-axis serves for displaying organizations or executive units of the company. But while on the x-axis of Process Supporting maps the business processes or particular steps of processes are depicted, a Time Interval map uses time for scaling. [6] [12]

Figure 7: Example of a Time Interval map [12]

Graph Layoutmap

In each of the previous presented map types the position of an object on the software map had a semantical mean- ing. Cluster maps use the space as a resource to express that certain organizations or parts of the company belong together. Cartesian maps, like the Process Supporting map and Time Interval maps are also based on positioning, since they operate on a coordinate system. In Graph Layout maps this is not the case. The position of an object does not have a semantical meaning. This is a great freedom for the ar- chitects who are designing a Graph Layout map. With the freedom of positioning they can create a much clearer map, which is more understandable for the viewers. The only dis- advantage of this is, that all the freedom is bought by giving up expressing power for having free positioning on themap. Generally a Graph Layout map utilizes nodes and edges to convey information. So, togehter belonging objects have to be connected via association lines. Appropriate examples for a Graph Layout map are e.g. UML diagrams or Entity- Relationship models. [12][6]

4. VIEWS,VIEWPOINTSANDSTAKEHOLD- ERS INTOGAF

The The Open Group Architecture Framework (TOGAF) is a free popular architecture framework. With the help of this framework it is possible to efficiently build an IT enter- prise architecture for an organization. The architecture is developed by the core of the framework - The Architecture Development Method (ADM). The ADM itself is an itera- tive method to create the whole architecture. Step by step the architecture can be defined, whereby for each step the area which should be covered has to be chosen. Also the breadth of coverage as well as the level of detail haveto be chosen. Below, you can find an image which is showing the inherent Architecture Developmentcycle.

Figure 8: Architecture Development Method (ADM)[7]

Since todays companies have a complex structure, where manypeopleandorganizationsareinvolvedinonecompany, such frameworks are definetly a must. Within a company there are different business processes, associations and or- ganizations. Accordingly the whole architecture of a com- pany cannot be represented in just one software map. The entire architecture of a company is not interesting for ev- ery person which is involved, because different persons or groups within the company have different working environ- ments and so each of them has also a different interest. The different parties in the company are namedstakeholders.

STAKEHOLDER1.Simplyexpressed,astakeholderisaper- sonoranorganizationthatisinterestedinthethingsan- otherpersonororganizationdoes.

Most of the time a stakeholder is affected by the direction a company goes or even able to direct parts of the company himself. This can be people working in the company from "simple workers" up to the managers of a company.Butstakeholders can also be extern people, who have interest in success of the company, like e.g. investors on the capital market.

VIEWS1.Aviewistherepresentationofacontext,e.g. apartoftheorganizationsarchitecture,whichcanbeina formofasoftwaremap.

VIEWPOINTS1.Viewpointsarealwaysreferingtoastake- holder.Viewpointsarethepointofviewofastakeholder whoseesthecompanyfromhisperspective,includingonly theaspectsofthecompanywhicharerelevantforhisactive workingenvironment.

Stakeholders can also be grouped together if they have sim- ilar or same interests, so that they all have a common view- point. To clarify these three important terms views, view- points and stakeholders, the following example should help: An air traffic controller and a pilot work at the same air- port. These two are the stakeholders in our example. Both of them have a view of the system, but neither of them has a view of the whole system, since not every part of the system is relevant for both. In this

example the viewpoints are the persepective of which the pilot sees the system and the per- spective of the air controller on the system. All in all it can be seen that both stakeholders have a subset view of the whole system and that both views differ from each other. The pilot has a air flight view of the system, whereas the air controller has a air space view of the system. The con- text between those three definitions and the software maps is the following: Stakeholders have a viewpoint from which they see the part of the system which is relevant for them - the view. And these views have to be depicted visually, so software maps are used to translate the relevant information for the stakeholder into a clear graphic. The next section is about examples for different stakeholdersacccording to the TOGAF in a company and which of the software maps can be used to visualize them.[7]

5. SOFTWARE MAPS FOR CORE TOGAF VIEWS

In the TOGAF core there are already standard views available. This section will deal with those views and it will provide suggestions on which kind of software maps could be useful to visualize these views.

Business ArchitectureView

At first, there is the Business Architecture View, which focuses at most on the user experience. Also part of the Business Architecture View is the production planning. The biggest issue is a change in the production process, so there have to be different scenarios for the prodution planning. To sum up, the Business Architecture View should be able to represent the production planning and the functionality respectively the usability of the product. In order toarchievethis, at least three types of software maps will be needed: First a Cartesian map (Process Supporting or Time Interval) to model the production process scenarios. The remaining software map types have the task to demonstrate features of the product. To show the coarse functional features the Use Case diagram is a good UML diagram type for it, since it is easy to understand for the user. In addition, Activity Diagrams are a clear way to display all possible outcomes by using or operating theproduct.

Enterprise Security View

TheEnterpriseSecurityViewisdevelopedforsecurityen- gineers,whohavetoensurethatvaluableinformationcannot get in the hands of unauthorized persons or organizations. So they have to be aware of the data exchange and the con- nected system units within the product or company. It is very common that distributed systems are used within com- panies or are even part of a product. Since these systems have different locations Cluster maps are very good to de- pict them. The monitoring of data exchange and the data flow within the application are ideally modeled by UML di- agrams. Activity Diagrams are able to show all possible states in a outcomes, which cases can occur. This is im- portant to know for the security engineers, since they have to be informed about all possible system states. The data exchange between several system units can be modeled by sequencediagrams.

Software Engineering View

For software engineers it is important to have an overview about the data flow and structure. Therefore this infor- mation has to be visualized in a way for them. Software engineers can make use of almost any presented UML type, since UML diagrams were originally designed for software engineers. It often depends on the task of the software en- gineer, but in general every presented UML diagram type is useful for software engineering. Since most of the time software functionality is complex to represent and the posi- tioning is not that important, Graph Layout maps are fitting for thisview.

SystemEngineeringandCommunications EngineeringView

System engineers and communication engineers do not have exact same tasks, but if we want to assign maps to these views, it turns out that we can assign almost the same software maps. System engineers are busy with op- timizing software and hardware interaction as well as de- signing computing models for a distributed computation en- vironment. From the communication engineer's point of view, it is important to understand how the communica- tion within the application is handled and how the system communicates with foreign systems. Therefore an overview of the distributed systems in the network and the commu- nication models itself (e.g. OSI Reference Model or similar) is mandatatory for both views. An exemplary communica- tion model can be seen below in figure 9 on the next page. Forthesemodels,Clustermapsareagoodchoice,sincethey are strong at representing content where the positioning and linking between units are relevant. The communication be- tween system entities can be modeled by the UML sequence diagrams.

Data Flow View
Controlling and monitoring the whole lifecycle of data witihin the system is the task of a database engineer. This includes data storage, data retrieval, data processing, data archiving and also data security. So for this view it is needed to have an overview about the data structures as well as how the data is managed in the system. The ER model provides an optimal model to depict the database structure. The se- curity and communication of data can be monitored from sequence diagrams like mentioned in previous views. As far

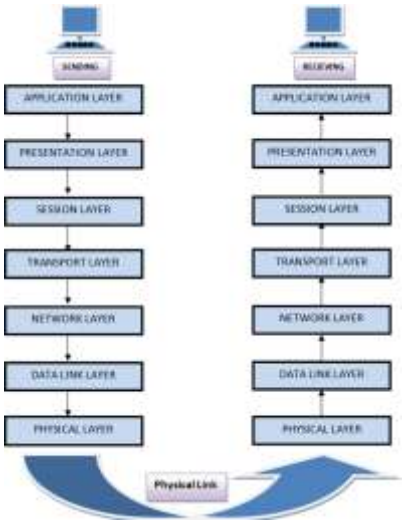

Figure 9: Exemplary Communication Model [9]

as the data processing within the system units goes, Activity diagrams are good to see how actions influence the system state and how the data is affected by this.

Enterprise ManageabilityView
Unlike most of the previous views, the Enterprise Man- ageability View is not related to engineers. Stakeholders of this view are the operations, administration and manage- ment personnel of the system. These personnel has to have the ability to oversee the structure the management within the company, as well as planning on future investments in projects by regarding the available budget. Basically most of the high level decisions are made by these personnel. The general structure of the management, executive organiza- tions and also the different locations of the company are important for this personnel. For modeling the connection betweenthedifferentorganizationsofacompany,oneshould make use of Cluster maps. It makes sense to use them here, because the positioning and grouping of certain elements is really important. Business processes can brilliantly be de- picted in a Cartesian map for the same reasons as mentioned in the Business Architecture Viewabove.

To sum up, we can say that in general the management based views utilize more the Cartesian maps and thetech- nicalviewswhichdealwiththeimplementationoftheprod- uctorsystem,morerelyonUMLandGraphLayoutmaps.
Whatbothoftheseviewcategorieshaveoftenincommon is, that both make use of the Cluster map.[10]

CONCLUSION

The paper began with the goal to first present general information about software cartography and the TOGAF. Thishasbeendonebyfirstexplainingthesoftwarecartogra- phy and introducing the different software map types. After that the TOGAF has been suggested as one popular repre- sentative of the EAM frameworks. With the framework also viewpoints, views and stakeholders have been described. Fi- nally some of the core views of TOGAF and the most im- portant software map types were matched together.Thismatchingshowed for which views, which software maptype can help to constitute the needed information efficiently. As for the future work this procedure can be done in a similar way for more views or for otherframeworks.

REFERENCES

[1] S. Buckl, Alexander, and M. Ernst. Generating visualizations of enterprise architectures using model transformations.2007.

[2] T. Horn. Uml unified modelinglanguage. http://www.torsten-horn.de/techdocs/uml.htm.

[3] IBM. Uml basics: The sequence diagram.http://www.ibm.com/developerworks/rational/library/3101.html.

[4] J. Lankes. Architekturbeschreibung von anwendungslandschaften: Softwarekartographie und ieeestd 1471-2000.2005.

[5] J. Lankes. Softwarekartographie: Systematischedarstellungvon anwendungslandschaften.2005.

[6] F. Matthes. Softwarekartenzurvisualisie rungvon anwendungslandschafte n und ihr en asp ekteneinebestandsaufnahme.2004.

[7] M. Schaefer. Enterprise architecture bebauungsplanungfÃijrinformationssysteme. http://st.inf.tu-dresden.de/files/teaching/ws11/ring/20111123_ Capgemini_Vorlesung_TUDresden.pdf,2011.

[8] B. Schaling. Das use-case-diagramm.http: //www.highscore.de/uml/usecasediagramm.htmll.

[9] Studytonight. Communication model figure.http://www.studytonight.com/computer-networks/images/Figure25.png.

[10] TOGAF. Developing architecture views.http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap31.html,2006.

[11] uml diagrams.org. Activity diagrams.http: //www.uml-diagrams.org/activity-diagrams.html.

[12] A. Wittenburg. SoftwarekartographieModelleund MethodenzursystematischenVisualisierung von Anwendungslandschaften. PhD thesis, July2007.