Context Based Software Development Knowledge Map

Mohamed E. Fayad, PhD and Charles A. Flood III
Department of Computer Engineering,
Charles W. Davidson College of Engineering,
San Jose State University, San Jose, CA, USA
m.fayad@sjsu.com and charlesFlood3@gmail.com

Context is worth 80 IQ points. Alan Kay [1]

The rich and complex elements existing across the development process that define the ultimate goal of the developers' team could be viewed as their context. This can be viewed across different dimensions and is helpful for a team of developers to remain within a predefined limit. In other words, they can easily recognize their boundaries and are able to view the scope of the software under development. This document clearly defines the dimensions that characterize the work environment of developers in context based software development. It also illustrates the knowledge maps by using enduring business themes, business objects and industrial objects, which create a comprehensive pattern in a software development context. These models and design patterns also provide a new dimension for the developer in a context based development. Furthermore, it gives them a clear perspective to use these design patterns to develop a number of applications on top of it. The developer can use extraction, analysis and availability to gain new knowledge, while using context based software development process. The patterns developed also provide a rich working environment to a particular developer and enables him/her to relate to the task that is being developed. The use case section provides a clear example and of how these enduring business themes and business objects can be mapped to an application leading the way for the user to design similar applications and use cases, to make use of these available design patterns and benefit and be more productive in the process of context based software development.

7.1 Introduction

Context can have different meanings depending on the connotation of usage [2]. There are two different ways of looking at context in the domain of software development. This essentially describes a way which can be very intuitive, and even a person having limited knowledge of context based software development can infer the meaning of it and later use it to develop different applications. In other words, the context is a vague idea and specific to an extent, where a person who does not have enough knowledge, may infer the meaning and understand the concept in its entirety [3, 4]. The other way to look at context is a more specific way of describing it [5, 6, 7]. This can lead us to the important topic that is *Context Based Software Development*. This cannot be generic and vague and number of applications can be built by using a particular context. If we take an example of location in Context [7], we can infer that any information that can be used to uniquely identify, characterize and define the properties of a particular entity can be thought as location.

Hence, in this way we can get a clear idea that an entity is a person, object, group of individuals or a place that is relevant to the interface between a user and its application. It is important to mention that the user's context becomes very important while we define the functional and nonfunctional requirements, especially in the case of Context Based Software Development. Therefore, we should have a very well defined context for developing models and applications in software development.

In fact, it is actually difficult if not impossible, to derive a very general definition or meaning of context. Any context should be very well connected or related to the entity, it should lead to the solution of the problem, it should evolve over time and should have a dynamic process in a very dynamic environment and it should relate to the domain of use and time. The features mentioned will give different problem statements and pose a number of challenges to software developers. Due to these features, it is also becoming incredibly difficult and hard for software developers to have just one dimension and context. Software Development projects have become humungous, incredibly complex and hard to capture as whole. During the development phase, it is difficult for software developers to grasp the context of software development, as they have to read and understand a lot of contextual information that is typically not processed and captured to improve the working environment of software developers.

Thus, the main aim of this Paper is to illustrate the context in software development domain with the help of knowledge maps. These knowledge maps are explained with short and mid-sized templates of applicability, context, unification and scenario. In addition, the work is intended to give a new perspective to developers who are working towards the development of context driven software development [8]. It also presents a context model and gives enough room, idea and knowledge for future research. Finally, it summarizes the entire discussion and gives an overview of the entire topic.

7.2 The Essential Properties of Context-Based Software Development

Context based Software Development displays various important properties including:

- Application:
  Application, in context based software development, refers to the use of Context model and the ultimate goals/aims/objectives behind its use.

- Pattern:
  There are two types of patterns: Dotted and Filled patterns. Dotted Patterns represent explicit relations, whereas Filled patterns represent implicit relations. In addition, common patterns are used to perform search operations.

- Scenario:
  Scenario is where most of the actions take place. Analysis of various scenarios can be used to support the work of developer providing additional knowledge, thereby enhancing the quality of the work environment.

- Context:
  Context represents a complex network of elements across various dimensions, which are not quite limited to the work developed on IDE. It usually takes into account of all dimensions that characterize work environment of the developer.

- Design:

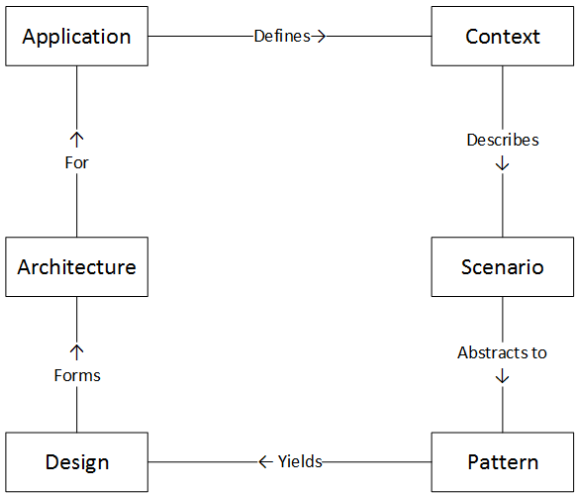
It is designed in such a way that various dimensions can be represented as layered model with four main layers: personal, project, organization and domain.

- Architecture:
  The developer context model has an architecture of layered model. Therefore, by each layer of the proposed context, model will define which context to capture, type of modeling, its representation and the kind of application for that layer.

7.3 Context-Based Software Development Knowledge Map

ILLUSTRATED KNOWLEDGE MAP:

Given the elements mentioned above, context-driven development follows a natural progression from the context to the application. The context is used to describe a scenario, which can then be abstracted into a pattern. This pattern is used to create a design which subsequently yields an architecture for the application.



Knowledge Map is also known as Core Knowledge and it shows how each goal is mapped to its properties in Context Based Software Development that is highlighted below:

- Simplicity:
  Organizations or people (Any Party, Any Actor) involved in software development need to have their components (Any Entity) chosen based on certain conditions (Any Criteria), with the ultimate result (Any Outcome).

- Adaptability:
  Software development team (Any Party) or the software components (Any Entity) need to have an ability to change something (through Any Mechanism) to fit to occurring changes (Any Event) or cope with random unexpected changes (Any Impact).

- Extensibility:
  All the remaining layers (Any Entity) of context model other than the original layers (Personal, Project, Organization, Domain (Any Party)) will be referenced (Any Event) repetitively (through Any Mechanism), as an extension (Any Outcome, Any Level) of the already developed work (Any Criteria).

- Customization:
  Information retrieval facilities needs to be customized (Any Criteria) by the software developers (Any Party) to facilitate the support (Any Event) of reusing (through Any Mechanism) software components (Any Entity).

- Abstraction:
  All the dimensions or layers (Any Entity) modeled in this context based software development are to be abstracted or hidden (Any Event) from the implementation by Development team (Any Party), and therefore cannot be seen (Any Impact, Any Outcome) by a user (Any Actor) directly.

- Applicability:
  Organization or a software development team (Any Party) requests for Applicability, which represents (Any Event) how the context model (Any Entity) is used (Any Event) and the objectives behind its use.

- Unification:
  All dimensions (Any Entity) of context information (Any Context) provided by a working environment of a developer (Any Actor) needs to be integrated (Any Criteria) (through Any Mechanism) for creating a unified context model (Any Outcome).

- Reusability:
  Software components (Any Entity) need (Any Criteria) to be reused (Any Event) by focusing on the information captured (Any Outcome) during the process development (Any Mechanism).

- Configuration:
  Software configuration (Any Outcome) is the task (Any Event) of making tracking of changes in software (Any Entity) and controlling them (through Any Mechanism).

- Personalization:
  For a developer (Any Actor) working on a specific task (Any Event), needs to deal with various resources (Any Criteria) for accomplishing the task (Any Outcome).

7.5 Applicability - Short Pattern Documentation Template

1. **Name: Applicability (EBT)**

Applicability is much-generalized term. This becomes a tool to develop a concrete pattern and it forms a model in any scenario, where one finds applicability in context based software development. This Paper includes a stable and robust design pattern for applicability in context based software development. It is applicable to all the scenarios of context based software development process too

According to Vocabulary.com, applicability is the "usefulness of something for a particular task." For example, books have great applicability for learning and gaining knowledge. In other words, something should be suitable and useful doe accomplishing a task. A book or a journal has more applicability in a library. Likewise, a drilling tool is useful for a carpenter or a mechanic and it cannot be used elsewhere other than the tasks that need it. Hence, the context in which something is used is very critical for greater applicability and effectiveness. In the domain of stable pattern development, the term context denotes the issue of applicability of something in a given context so that it becomes meaningful. Applicability is the main business theme of a Context based stable pattern.

1. **CONTEXT**

A stable pattern designed for Context Based Software Development can be used in various scenarios. The three layers which it is made up of are as follows:

Scenario #1: Personal Layer
In this scenario, a person (AnyParty) plans to develop a new software pertaining to a particular context (AnyEntity). The person wants to develop a new context based software that is designed (AnyMechanism) in a unique (AnyCriteria) manner. When we start a context based software from scratch, then it can be developed (AnyContext) in several stages (AnyStage). Various business rules (AnyRule) applicable to the software are listed and context based software is developed accordingly.

Scenario #2: Project Layer
This scenario corresponds to a startup (Any Actor), where they develop a new model (AnyContext) for their customer. They want it to be very professional (AnyCriteria). They also follow a specific design pattern (AnyMechanism), which can be reused and changed over time (AnyDuration), and it has a specific cost (AnyRule) associated with it for the changes, for developing new features and for improving the existing features along the way.

Scenario #3: Organization Layer
This scenario corresponds to an organization (AnyActor), where they want to redesign (AnyType) and adopt new approach (AnyCriteria) to the way they develop context based software for their internal use. They want fresh talent like new college grads and interns to participate and contribute (AnyMechanism) to this project. The cost (AnyRule) associated with this type of development is estimated and they also figure out the duration (AnyDuration) associated to develop it.

2. PROBLEM

Functional Requirements:

1. Any Actor:
The person, who is a part of the system and interacts with the system, is known as the actor. It can be a creature, hardware, software or any person, who can be termed as an actor. In this context, we can term a person, a startup or the organization, which develops context based software as an actor.

2. Any Party:
We can refer a party as any organization, political party etc. as any party. We can also term any party as a group of individuals having similar ideology and concepts. In this context, we can refer to an organization, which develops context-based software as a party

3. Any Criteria:

A concept or reason on which we take a judgment or decision is known as criteria. We always have a set of criteria before developing a context based software. There are certain requirements of actors like some special business rules and unique features that they want in software, then it can be termed as any criteria.

4. Any Rule:

A set of rules, regulations and policies imposed by a party or actor can be termed as any rule. For Example, when a customer has a unique condition from the organization that the cost of the software should not exceed a particular amount and it should be delivered according to their schedule, then it can be termed as a rule.

5. Any Mechanism:

The process which is used to achieve the desired goals is known as a mechanism. For Example, when we are trying to adopt new methods and design patterns in context based software development or when we are trying to have a new approach to the development process, then it can be termed as any mechanism.

6. Any Stage:

The duration or period in which we take a particular step or activities of context based software development can be termed as a stage. For the development of context-based software, we may have various activities like identifying the right resources, planning the development task, dividing the tasks among different teams, following a certain timeline to achieve desired outcome. These steps should be executed in a series. This can be termed as any stage.

7. Any Duration:

The period or the time it takes to develop a context-based software is known as duration. There is a certain deliverable according to the timeline set. This is known as any duration.

8. Any Type:

Class or group which can be identified in context based software development is known as a type. In this case, the context used for software development forms a legitimate example of Any Type.

9. Any Entity:

The entities and attributes used in context based software development can be termed as an entity.

Non-Functional Requirements:

1. Cost effective:
Context based software development should be very cost effective and should be in the budget of the actor for whom the software is being developed.

2. Creative:
Context based software development should ideally be very creative. It should always set a very high standard and raise the bar, in its creativity, quality of final product or the kind of approach that we take to build the context based software.

3. High Quality:
The factors which are very apparent to the user and which set the final product apart from its counterparts form the quality factor. In terms of context-based development, the software should not crash and should be very responsive to meet this criterion.

4. Feature rich:
Context based software should be available with many features, which are able to meet the needs of the end customer for whom we are developing the software.

3. SOLUTION: APPLICABILITY STABLE ANALYSIS PATTERN

Design Patterns:

Class Diagram for Applicability

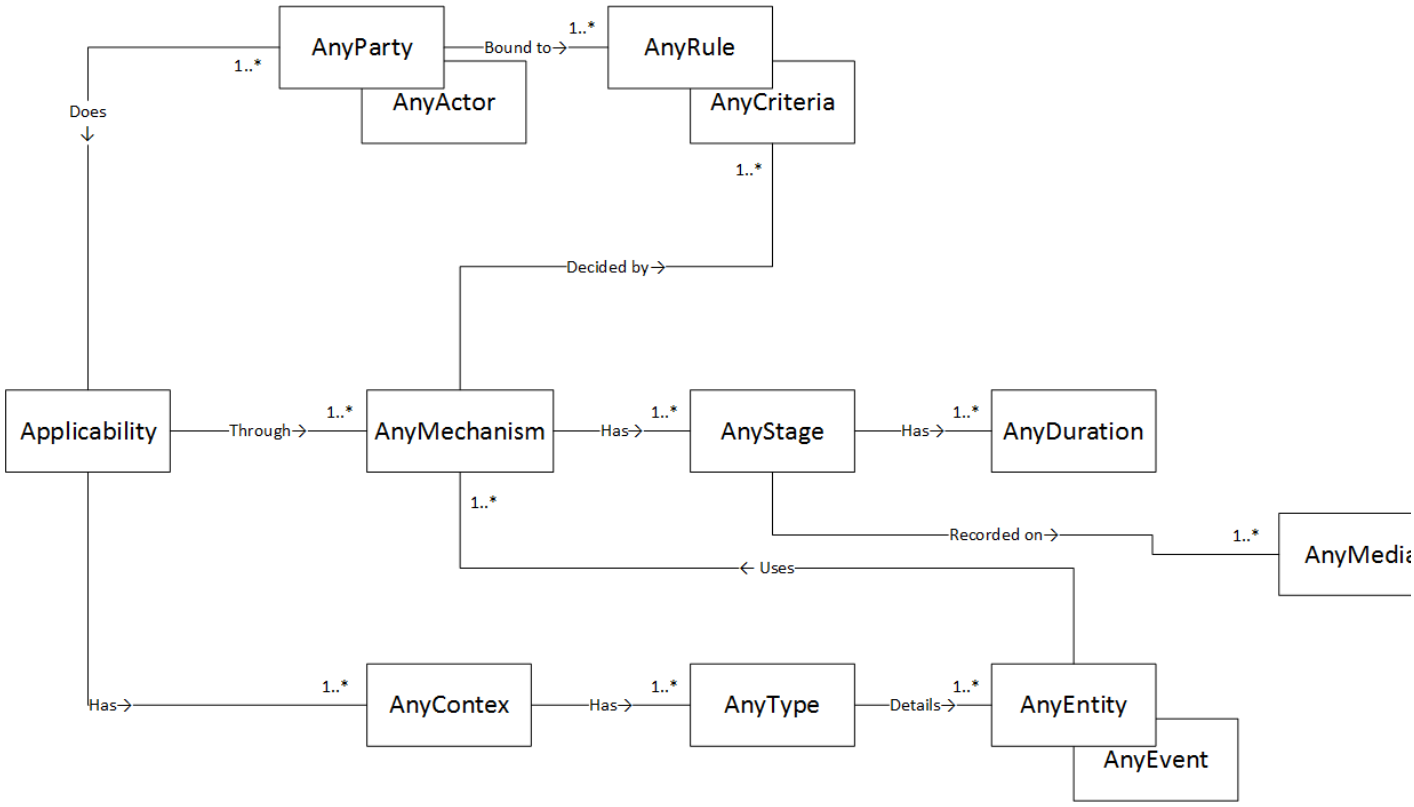


Fig 1. Stable Analysis Patterns for Applicability

Description of Class Diagram:

- Applicability of Context based software development is done by Any Party.
- Moreover applicability is done through any Mechanism.
- AnyCriteria or AnyRule should bind AnyParty or AnyActor.
- Any Mechanism also depends or is influenced by Any Rule or Criteria.
- Any Mechanism also has a duration and a series of steps associated to it.
- All the steps in the process have a stipulated duration or timeline in which they are to be completed.
- Applicability has a particular or specific context.
- Each Context also has type of the software associated to it.
- Context based software development is applicable and is developed for a specific entity

The class participants of Applicability Stable Analysis patterns are as follows:

4. MAPPING FIVE DIFFERENT APPLICATIONS

EBT	BO's	Personal Softwar	Company's inte	Web-Based tool	Native tool-IO	Third Party too
Applicability	Any Party	Student/Person	Company/Orga	Company/Orgar	Company/Orgar	Company/Orga
	Any Criteria	Personal Project	Internal compa	Scalable Project	Specific to one p	Web service
	Any Mechanis	Professional Dev	Productivity of	Available Anytir	Using 100% of a	Ease of use and
	Any Duration	Month	Month	Month	Week	Month
	Any Type	Term Project	App customized need	ERP	Android App	Outsourced too
	Any Entity	Person	Company/Orga	Company/Orgar	Company/Orgar	Company/Orga
	Any Rule	Useful and creati	Cost Effective	Cost Effective	Cost Effective	Replaceable/M
	Any Stage	Planning, Design,	Planning, Desig Testing, Mainte	Planning, Design Testing, Mainte	Planning, Design Testing, Mainte	Planning, Desig Testing, Mainte
	Any Media	Hard drive	Hard drive	Remote Server	SD Card	Hard drive

Context – Short Pattern Documentation Template

1. Name: Context (BO)

The stable pattern designed for Context is a generalized pattern that is applicable to model any scenario that involves the concept of context. This Paper includes a stable design pattern for Context in such a manner that it can be applied to any case, where various contexts are involved.

The enduring business theme for Context is Applicability.

2. CONTEXT:

The stable pattern designed for AnyContext can be used in various scenarios and domains. Following are the five scenarios shortlisted for Applicability.

Scenario #1: Business domain

Web-based systems, aerospace embedded systems, small hand-held instrumentation? (AnyMedia)

Scenario #2: Number of instances

Building one system, a dozen, or millions? (Through AnyMechanism)

Scenario #3: Maturity of organization

How long has the organization (AnyParty) been developing software? How mature are the processes (and the people) (AnyActor) relative to software development?

Scenario #4: Level of innovation

How innovative is the organization? (AnyParty) Creators or early adopters (AnyActor) of new ideas and technologies? Or treading on very traditional grounds?

Scenario #5: Culture

In which culture are the projects immersed? Are both national culture and corporate culture? (AnyType). What are the systems of values, beliefs and behaviors that will impact, support or interplay with the software development practices? (AnyEntity/AnyEvent)

Example: For instance, if the business domain is “aerospace-onboard software”, this will pre-condition several project-context factors: criticality, size, business model, this in turn will make some practices suitable or not, will influence amount and type of documentation.

3. PROBLEM

Functional Requirements:

1. Any Actor

An actor is a person, who interacts well with the system. It can be a person, a software package, a hardware or a creature. In this scenario, actor may be a software developer or an adopter or a consumer of the software package.

2. Any Party

A party is group of individuals having the sharing similar ideas, ideologies and thoughts. Any party refers to any organization, a political party, a team in a software firm etc. In this context of Applicability, any party refers to any Organization or a company.

3. **Any Criteria**

Criteria is something that is used as a reason for making a judgment or decision. There are various criteria need to be checked for proposing a context model in a working environment of software developer.

4. **Any Rule**

Rule is a set of regulations or principles imposed by any party or actor. There are various rules that need to be imposed for context based development of software components by a software developer in a working environment.

5. **Any Context**

Various networks of elements exist across different dimensions that are not limited to the work developed by a software developer on an IDE.

6. **Any Mechanism**

Mechanism is the way of achieving the objectives or goals that are already set. There could be various mechanisms to achieve the unification of system by integrating various dimensions of context model.

7. **Any Type**

Any Type is a particular class, or kind or classification of similar groups. In this scenario, various types could be type of problem to be solved, type of project to be worked on, or type of people included in the team.

8. **Any Entity**

Any Entity is a thing with distinct and independent existence. It is always necessary to know the systems of values, beliefs and behaviors that will impact, support or interplay with the software development practices.

9. Any Media

Any Media in this case could be Web-based systems, aerospace embedded systems, or small hand-held instrumentation.

Non-Functional Requirements:

1. Size:

The overall size of the system under development is by far the greatest factor, as it will drive in turn the size of the team, the number of teams, the needs for communication and coordination between teams, the impact of changes, etc.

2. Stable architecture:

Stable architecture suggests if there is an implicit and de facto architecture already existing in place at the start of the project. Most of the projects are not mature enough to demand a lot of architectural effort. They usually follow commonly accepted patterns in their respective domain.

3. Business model:

It is all about the money flow. It needs to be checked how developing an internal system, a commercial product, a bespoke system on contract for a customer, a component of a large system involves many different parties.

4. Team distribution:

Linked sometimes to the size of the project, how many teams are involved and are not collocated? This increases the need for more explicit communication and coordination of decisions, as well as more stable interfaces between teams, and between the software components that they are responsible for.

5. Rate of change:

Though agile methods are embracing changes, not all domains and system experience a very fast pace of change in their environment. It is important to know how stable is the business environment and how many risk factors exist.

6. Age of system:

Age of system could be the evolution of a large legacy system, bringing in turn many hidden assumptions regarding the architecture, or the creation of a new system with fewer constraints.

7. Criticality:

Criticality in this case, would be the impact on people, if they die or are hurt by the system failure. Documentation needs increase dramatically to satisfy external agencies, who will want to make sure the safety of the public is assured.

8. Governance:

Governance deals with how projects are started, and terminated. Who decides what happens next and when things go wrong. How is success or failure defined and who manages the software project manager?

4. DESIGN/SOLUTION

AnyContext Stable Analysis Pattern:

Pattern structure and Participants:

Class diagram of the pattern (AnyContext):

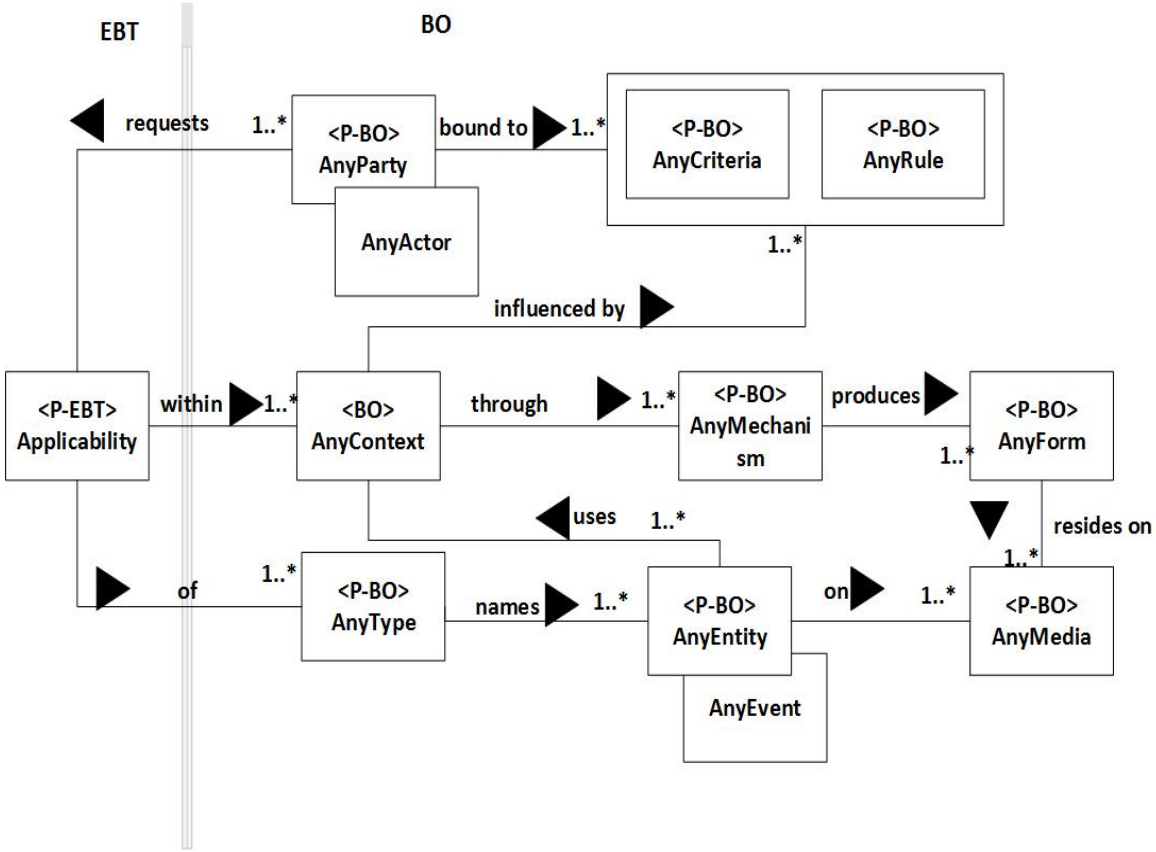


Figure 2: AnyContext Stable pattern analysis

Full description of the class Diagram:

- AnyParty/AnyActor requests for an applicability of AnyContext
- Applicability is within one or more contexts and is of one or more types
- AnyParty/AnyActor are bound to one or more criteria and rules
- AnyContext is influenced by one or more criteria and rules
- Applicability within AnyContext is achieved through one or more mechanisms.
- AnyMechanism produces one or more forms
- AnyForm resides on one or more media
- AnyType names one or more entities or events
- AnyEntity/AnyEvent is on one or more media
- AnyContext is used by one or more entities or events

Participants:

The participants of the AnyContext Stable Analysis Pattern are:

Class:

AnyContext is a BO and Applicability is its enduring business theme, which represents how the context model is used and the objectives behind its use.

Patterns:

- AnyParty: A party is group of individuals having the sharing similar ideas, ideologies and thoughts.
- AnyActor: An actor can be a user, a software developer or a consumer.
- AnyCriteria: is something that is used as a reason for making a judgment or decision.
- AnyRule: is a set of regulations or principles imposed by any party
- AnyType: is a particular kind, class, or group of Applicability.
- AnyEntity/AnyEvent: defines the entities that are subjected to Applicability
- AnyMechanism: is a way of implementation to achieve something and produces forms
- AnyForm: is something that resides on any media.
- AnyMedia: It can be a web-based or embedded systems or small hand-held instrumentation.

5. MAPPING FIVE DIFFERENT APPLICATIONS

EBT	BOs	Business Domain -IOs	Number of Instances -IOs	Organization Maturity -IOs	Innovation Level -IOs	Culture -IOs
Applicability	AnyParty/ AnyActor	Business team, Developers team	Governing Team, Production Team	Organization, Company	Disciplined Agile Team, Technical team	Team Culture, Organization System
	AnyCriteria	Manage budget	Instance Quantity	Duration of development	Length of Organization innovation	Systems Of values, Beliefs
	AnyRule	Certain delivery Date	Large legacy System	Maturity of Process	New ideas and technologies	Best practice
	AnyContext	E-commerce site/ Air traffic control System	Evolution of system/create new one	Process or People in Software	Creators or Early adopters	National & Corporate culture
	AnyMedia	Aerospace /web systems	Web-based	Enterprise, Project	Traditional grounds	Team
	AnyType	In house, Open source	One, dozen, Million	Co-located, Global	Simple, Multiple platforms	Open, Entrenched
	AnyEntity/ AnyEvent	Web/small hand-held instrument	System	Organization	Technology	Team
	AnyMechanism	Adopting process And people	Build Instances	Applying agile at scales	Performing challenges	Agile or lean approach

Conclusions

In this Paper, we have presented an approach of a context based software development. The context model discussed here is based on a layered structure, that considers four major dimensions or layers of the work environment of a software developer (personal, project, organization and domain). Therefore, each layer should be defined by taking into consideration of context capture, modeling, representation and application. The document performs a knowledge map of various recognized enduring business themes and their respective business objects in the context based model of software development. Various common patterns are also mapped across all the EBTs. The network that exist between developers, different tasks and resources are represented in the context model of the developer.

REFERENCES:

[1] http://www.brainyquote.com/quotes/quotes/a/alankay451982.html?src=t_context

[2] De Fina, A., Schiffrin, D., & Bamberg, M. (Eds.). (2006). *Discourse and identity.* Cambridge: Cambridge University Press..

[3] Ghadessy, M. (Ed.). (1999). *Text and context in functional linguistics.* Amsterdam: John Benjamins.

[4] Givón, Talmy. (2005). *Context as Other Minds.* Amsterdam: John Benjamins.

[5] William Labov (1972). *Sociolinguistic patterns*. Philadelphia, PA: University of Pennsylvania Press.

[6] Leckie-Tarry, H. (1995). *Language & context. A functional linguistic theory of register.* London: Pinter Publishers.

[7] Stalnaker, Robert Culp (1999). *Context and content*. Oxford: Oxford University Press.

[8]] M.E. Fayad, Mohamed, Sanchez, Huascar , Hegde, Srikanth, Basia, Anshu, and Vakil, Ashka, Software Patterns, Knowledge Maps, and Domain Analysis. Auerbach Publications (December, 2014), 422 pages.

[9] Fayad, Mohamed and Altman, Adam. "An Introduction to Software Stability." The Communications of the ACM, Vo. 44, No. 9, September 2001, pp 95-98.

[10] Fayad, Mohamed. "Accomplishing Software Stability.*"The* Communications of the ACM, Vo. 45, No. 1, January 2002, pp 95-98.

[11] Fayad, Mohamed "How to Deal with Software Stability." The *Communications of ACM* vol. 45, no. 4, pp. 109-112, Apr. 2002.

[12] Fayad, M.E. Stable Analysis Patterns: Understanding of the Problem, vrlSoft publishing, December 2916.

[13] Fayad, M.E. Stable Design Patterns: The Ultimate Solutions, vrlSoft publishing, December 2916.

Appendix

Knowledge Map of "CBSD"

1. KM Name: Context Based Software Development

2. KM Nickname: None

3. KM Domain/Subject/Topic Description: Context Based Software Development characterizes the work environment of a software developer by considering all dimensions. The term context here implies various circumstances around an area of interest. Useful contextual information can also be extracted by using the project management tools.

4. EBTs/Goals: EBTs of the "Context Based Software Development" and a short description of each EBT [9], 10, 11, 12] in table 1:

Table 1: EBTs of "Context Based Software Development"

EBTs/Goals	Description
Simplicity	Context information needs to represent information ranging fro The simplest structures are easier to build and maintain.
Unification	All dimensions of context information provided by a working e developer needs to be integrated for creating a unified context n

Reusability	Software components needs to be reused by focusing on t captured during the process development.
Adaptability	An ability to change something to fit to occurring changes unexpected changes.
Configuration	Software configuration is the task of making tracking of chan and controlling them.
Applicability	Applicability represents how the context model is used and the behind its use.
Extensibility	All the remaining layers of context model will be referenced ite extent to the developed work already.
Personalization	For a developer working on a specific task, needs to be dealing resources for accomplishing the task.
Customization	Information retrieval facilities are to be customized in order to s reuse of software components.
Abstraction	There are various layers in context model that are hidden/abstra user to handle directly.

5. BOs/Properties: BOs of the "Context Based Software Development" and a short description of each BO [9, 10, 11,13] in table 2:

Table 2: BOs of "CBSD"

BOs/Capabilities	Description
Context	Network of elements across different dimensions that are not li developed on IDE.

Scenario	It is where most of the actions take place, where large amounts information is available.
Application	It refers to the use of Context model and objectives behind its u
Pattern	Dotted and filled patterns represent explicit and implicit relatio Common patterns are used to perform search operations.
Design	Design includes layered model that a typical developer needs to different layers in a working environment.
Architecture	The developer context model has the layered architecture and r integrating all the dimensions.

6. Knowledge Map (Core Knowledge): Each EBT mapped to its BOs of the "CBSD" in table 3:

Table 3: Knowledge Map of "CBSD"

EBTs	**BOs**
Simplicity	AnyParty, AnyMechanism, AnyEntity, AnyEvent, AnyCri AnyOutcome
Unification	AnyParty, AnyCriteria, AnyOutcome, AnyContext, AnySc AnyMechanism, AnyEntity, AnyEvent
Reusability	AnyOutcome, AnyEntity, AnyEvent, AnyMedia, AnyMecl AnyParty, AnyConstraint, AnyCriteria
Adaptability	AnyEntity, AnyEvent, AnyImpact, AnyConsequence, Any AnyCause, AnyMedia, AnyMechanism
Configuration	AnyImpact, AnyOutcome, AnyCriteria, AnyConstraint, An AnyParty, AnyType

Applicability	AnyContext, AnyType, AnyMedia, AnyParty, AnyActor, AnyMechanism, AnyCriteria, AnyEntity, AnyEvent, AnyR
Extensibility	AnyParty, AnyEntity, AnyEvent, AnyMechanism, AnyCri AnyOutcome, AnyLevel
Personalization	AnyParty, AnyActor, AnyMechanism, AnyEntity, AnyEve AnyOutcome, AnyImpact, AnyCriteria
Customization	AnyParty, AnyActor, AnyMechanism, AnyEntity, AnyEve AnyOutcome, AnyImpact, AnyCriteria
Abstraction	AnyParty, AnyCriteria, AnyEntity, AnyEvent, AnyMechar AnyLevel, AnyImpact, AnyOutcome, AnyMedia