# Domain-Design Driven Approach to E-Meeting Application Development

**Anggoro Ari Nurcahyo, Ade Sukendar**

***Abstract:** Meetings are used as a medium to discuss, negotiate and make a decision on a problem based, except that the function of the meeting is often not going well in the university's faculty of informatics engineering. This happened because the results of the meeting were still scattered in several people and fields, so it was difficult to evaluate the results of the meeting, which eventually resulted in ambiguous meetings with no standardized decrees, rules, procedures that would overshadow every activity within the faculty. as well as the determination of the distribution of assignments specified in the meeting and the incentive burden. In designing this modern application, the approach that is often used is Domain-Driven Design (DDD). Applying and expanding the concepts and tasks of DDD is very challenging because it does not have a description of the software development process and classification in the software development process approach. For this reason, we provide a brief overview of the resource-oriented DDD-based software development process that takes into account the desired application requirements.*

***Keywords:** Meeting; Domain-Driven Design; software development process;*

## I. INTRODUCTION

Documentation is an important activity that conducted in an organization to record every activity passed by an organization which gives the easiness in evaluation activity and decision making for every organizational activity. One of the activities which need to execute documentation is meeting; it is because in that activity there are much information, knowledge, and decisions taken tosustainability of organization. In meeting, several outputs like presentation, text minutes, video/ audio data produced. Participant or meeting member can understand easily a minute of meeting if those data are intended to be extracted mutually of the important points[1]. Information plays the important role in organization, it is possible to management to make decision [2], so every decision will be used finally as a definite legal basis that should be policies, references, rules, standards, procedures, and so on it must be inaugurated and passed depend on the determining of organizational level. If it is not conducted so it will cause an adverse boomerang for management by non-compliance and uncontrolled activities that occur properly or it appears other vulnerabilities.

It is being faced by pasundanuniversity’s engineering faculty so how software is designed can give easiness like conducting a review of any policies made based on the outcome of meeting that has been conducted, the distribution of minutes to be verified against related person and the verified minutes and that become the basic problem is distributing of minutes in several people or field that impacts not to optimum administrative for giving the easiness in doing a review of needs of related parties.

According to Evans[3], DDD approach provides means to represent factual world in architecture, such as by using bounded context which representorganizational unit, and also it identifies and focuses on the core domain. These second characteristics lead to the improvement of software architecture quality [4]. Improving the quality of SW development with involving the soft purposes like reliability, traceability, security and performance, and also other quality attributes from software conception [5].Domain Driven Design offers principles, patterns, activities and examples of how the domain model is built as a core artifact. However, this does not provide a detailed and systematic development process for applying the principles and patterns or classifying them into the field of software engineering. so how this DDD approach can be implemented into the software development process can improve its application. Furthermore, the classification of patterns and principles becomes the concept of software architecture [6], and arranges application components according to the DDD layer and explains the activities needed to build E-meeting applications and discuss both the outcome and limitations.

## II. LITERATURE REVIEW

### Domain-Driven Design

Domain-Driven Design (DDD) is an approach to design and software development; it is introduced more than a decade ago by Eric Evans with clear intention to find more effective way in solving complexity attached from software development. The premise encourages to Evans to create this technique is [3]:

- For some software project, the major focus is must be on domain and domain logic.
- A complex domain design must be depended on model.

Revised Manuscript Received on May 22, 2019.

**Anggoro Ari Nurcahyo,** Informatic Engineering, Pasundan University, Indonesia
**Ade Sukendar,** Informatic Engineering, Pasundan University, Indonesia

The main power of DDD is relied on systematical approach, it offers to Pinpoint and “Crunch” business domain aspect. DDD is revolutionary because it is not depend on technology or increasingly strong service to achieve business objective through software. DDD has two different parts: strategic design and tactical design. Strategic design is the most important things and spin around in the pattern and the practice to analyze domain and design the top-level-architecture of the system. Tactical design is about the outcome implementation of strategic design [7]. To allow doing analysis like thorough analysis of business domain, DDD offers three patterns of analysis as follow: First, Ubiquitous Language, It is a shared team language. It is shared by domain experts and developers [8]. You can understand business language and know in detail about business mechanic. It is true to say that ubiquitous language determines the naming conventions, but the essence of making ubiquitous vocabulary is understanding business and reflecting business process with code. It is more relevant than only making the effective naming convention [7]. Second, Bounded Context, It is business domain area that provides the element of ubiquitous language as the clear and not ambiguous meaning [7]. In single Bounded context is Ubiquitous language that is formulated by team. It is stated between team in within software model. The different teams, sometimes each of them are responsible for Bounded context given, using context mapping to separate strategically Bounded context and understand its integration. In single modeling limitation, team can use a tactical modeling tools which is useful: Aggregate, Entity, Value Object, Service, Domain Event, etc. [8]. Third, Context Mapping, The way of DDD to express top-level-architecture is through the plural bounded context composition which is interconnected to the relationship. Design artifact that expresses scheme is called context map. In other words, context map is diagram that gives a thorough view of system being designed [7].

### Sofware Enginerring Activities

Real software processes are interleaved sequences of technical, collaborative, and managerial activities with the overall goal of specifying, designing, implementing, and testing a software system. The four basic process activities of specification(the requirements elicitation and analysis process), development (the implementation stage of software development is the process of developing an executable system for delivery to the customer), validation, and evolution are organized differently in different development processes. [9].

## III. METHOD

There are two sub-activities that occur during the requirements elicitation and analysis process, first, the information model, as part of the domain model, is created by "crunching knowledge" with domain experts; Second, do discussions with customers and users to design a retirement. Because these two activities are closely related (when discussing prototypes, knowledge of the domain gets deeper, and when finding information models, terms or workflows may change), we combine them into one single activity.

In the book Evans and Vernon provide an overview for understanding needs and domains through modeling [3], [8]. In this stage knowledge of the domain can be obtained through experienced modelers in its domain. This will make it easier to achieve the application development that satisfies all parties.

According to DDD, collaboration with customer was very important to be explored and particularly for domain model. So the first DDD pace and replay were Knowledge Crunching [3]. In line with discussion to customer, developer team did modeling activity and made domain model step by step. Besides that, Ubiquitous language would be made, which was cross-team language. Origination process of domain model was very influenced by exploration and experiment [3], [8].More better if model that was not fully satisfied will be applied, than repairing domain model continuously without betting the implementation [8].

Ubiquitous language was simply glossary from all of terms used by the experts of certain business domain. Every term in the glossary was expected not to be ambiguous and it had a good meaning. Terms, mostly, nouns and verbs, and glossary also cover the correlation between nouns and verbs so it was clear, any verbs (i.e Actions) applied to any nouns (i.e Entity) [7].

The important ideas of DDD was domain binding towards implementation [3]. Domain model was the core artifact to achieve this objective in this domain layer. During analysis stage of computational independent model, information model as part of domain model created. First, this model was separated to be Bounded Context and, second, this Bounded Context was expanded and improved, for example, by implementing design pattern to fulfill application needs. This activity was based on the sample of Evans, Vaughn, and Esposito [3], [7], [8].

To produce Context map that showed the correlation of bounded contexts was needed experience and interaction in explaining information model became bounded context and it was conducted decomposition to developer team division to do each of bounded context.

The next stage was especially conducted by developer team who was responsible for every context limited. The purpose of the next activity was repairing and expanding domain model depended on the qualification of application. Design of UI/ UX was the main source for qualification of application.

Maybe, domain object in information model had been marked by stereotype that showed types such as:aggregate root, value object, andentity or domain event. Even some services were possible to be identified during analysis stage. Object that lost the stereotype must have been improved at first; a must have been added. Then, design of patterns repository, factory and domain service were added depend on necessities.

For example, if there was function which needed to display domain object in UI, a repository was added, or if there was complex aggregate root, factory needs to be added [3]. During the whole of designing process, domain expert and other information source were involved (continuous knowledge crunching). After applying design patterns, domain model was ready to be implemented.

## IV. DISCUSSION AND RESULT

Discussion is conducted in form of analysis and application design based on method explained before by seeing needs of the related parties and knowledge of domain experts.

### Information Model

In the approach described earlier, we need to find out domain knowledge with crunching for the requirements elicitation. The domain expert chosen is the executive in the faculty and conducts interviews to determine the meeting domain from their perspective, and to explore the linkages between concepts, relationships, and constraints that occur in meetings and included in the information model. To know activity plot happened in the meeting, it is explained on figure 1. It uses UML standard notation because it explains only plot depends on domain expert.

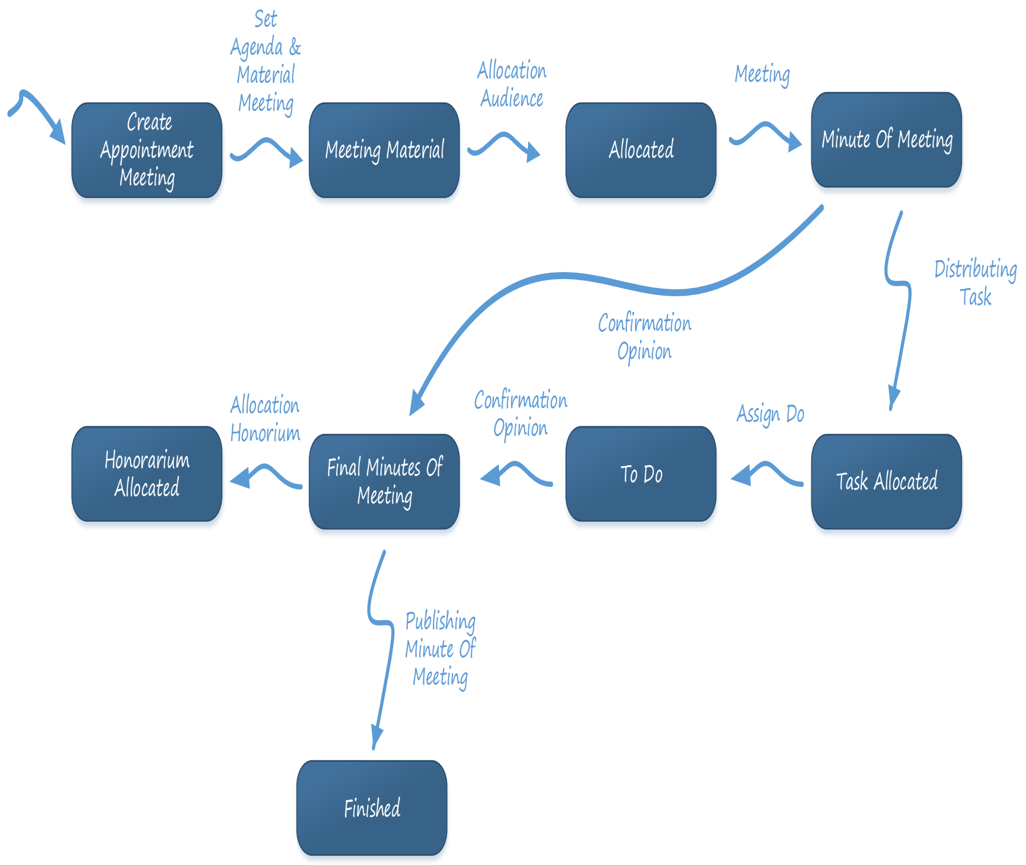

**Fig. 1 Sketch of Meeting Activity**

In discussion result with domain expert so it can be concluded Crunching Knowledge as follows:

1. Type of meeting can be classified as follow:
   a. Routine operational coordination meeting,
   b. Academic meeting (relate to activity of planning, implementing, and learning evaluation and in this case there are some classifications that is specified depends on study program under engineering faculty).
2. Only employee within organization can make invitation and set meeting agenda.
3. Before doing meeting activity so every participant of meeting will be given invitation at first and it is given material attachment related to meeting agenda.
4. Before conducting meeting activity so every participant of meeting can propose additional agenda and material attachment related to meeting.
5. Meeting maker can do approval or cancellation agenda of the meeting member.
6. Each of discussion result or decision happened in the meeting will be documented in a minute of meeting.
7. Member of meeting can do verification of their opinion in discussion of every agenda has been discussed.

8. If there is opinion that needs to be corrected, the opinion will be givens to be proposed the correction to meeting maker and the correction are conducted by meeting maker.
9. If there are tasks determined in the meeting so there is task notification which must do and time dispensation given to the pointed member for task result can be attached in digital media also a place that has been prepared for uploading.
10. Every meeting activity and job allocation can be given budgeting depends on the provision of organizational financial party.

Next, extract to nouns and verbs from the story above to find nouns that will be my main object and not its attributes.

- **Nouns**

Employee, Invitation, Agenda, Comment, Attachment, Cost.

- **Verbs**

Make meeting invitation, See list of meeting member who will be invited, Set meeting agenda, Invite employee who will be invited to meeting, Propose meeting agenda, Attach agenda material, Attach task material, Verify meeting member's opinion in minutes of meeting, Make comment to opinion's correction, Provide task to meeting member, Determine job cost of meeting member, Allocate job for meeting member, See total cost allocated for a job, See job allocated for accomplished.

Based on nouns and verbs that have been defined, then further defining entities, value objects and aggregates as shown in Figure 2, for all nouns will become entities as for additional entities formed because there are constraints on verbs such as on the proposed agenda, noun task and attachment agenda. for value objects can be part of the value object entity, the date range consisting of the start and end dates used in the agenda.

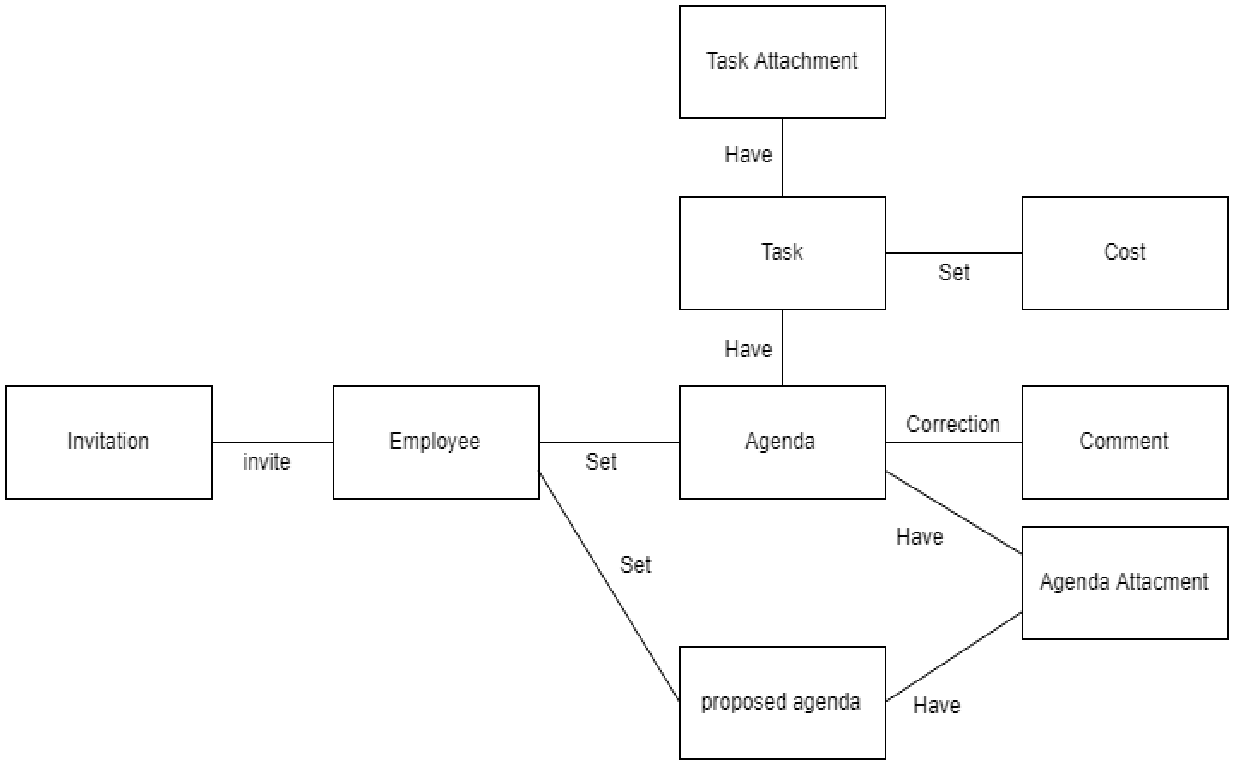

**Fig. 2 Piece of Information Model of E-Meeting**

After getting the information model, this becomes a complex thing to be directly applied to the team, so the information model is grouped to be limited to the focus of each development team. Actually, the boundaries are the main benefit of a bounded context because they 'contain' a specific model, allowing us to focus on it. The main utility of bounded context is that it allows better decoupling, which is the most important trait of maintainability. Basically, grouping together related models while keeping them separated from other models. In this case there are three domains that have been successfully established as shown in Figure 3, First, the invitation domain which focuses on how the mechanism of inviting members to join the meeting from the media used, states attendance until rejecting the invitation offered. Second, the domain agenda focuses on the agenda and the minutes of the meeting are managed until the parking lot even if there are records that need to be corrected, then comments can be given to be repaired and coordinated with the person concerned. in a meeting it is possible to delegate tasks to each member to carry out the decisions that are set, this will be discussed in the third domain, the task domain. The domain implements the time span needed to complete the task, determine the tasks that need to be carried out, carried out by who and what costs are needed in completing the task.

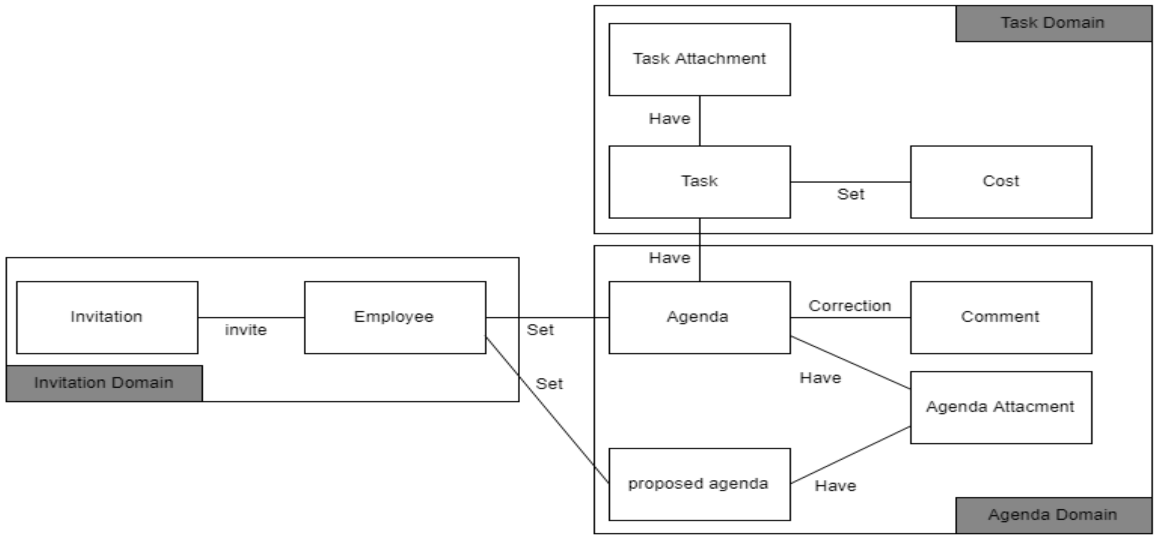

**Fig. 3 The identified domain model pieces**

After that, we start to identify entities, value objects and aggregate that have been classified based on bounded context (see Figure 1, and Figure 2). With diagram of Pieceof Information Model of E-Meeting, we can think to object responsibilities specifically. One of the most common mistakes is emphasizing the responsibility to actor object, in case is meeting. That object has to keep itself and it needs to be covered to communicate directly and you need to communicate with it through the function, as seen on Figure 5.

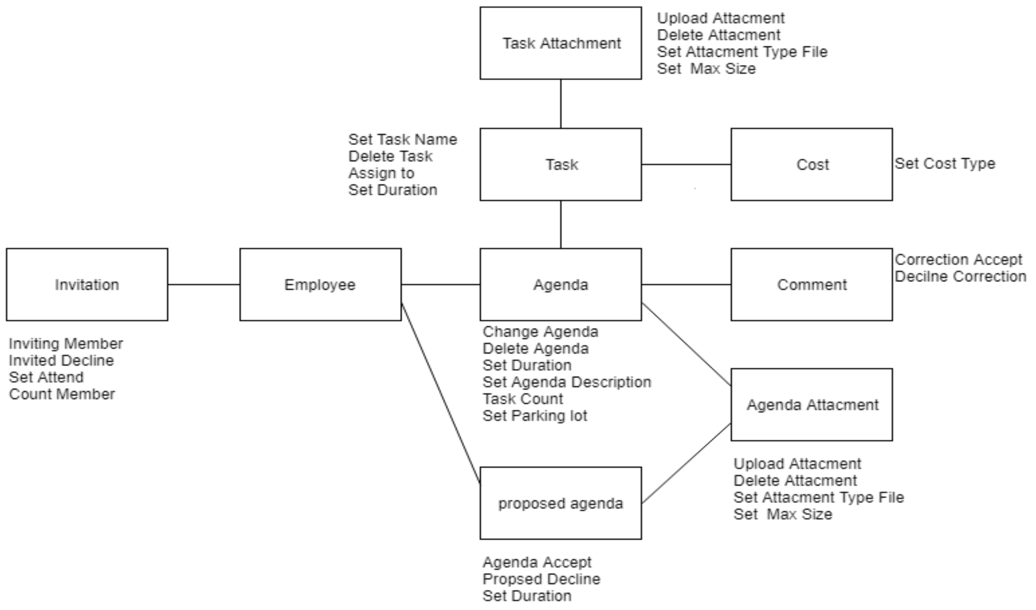

**Fig. 4 Diagram of Object Responsibilities**

**Designing of Prototype**

Besides explaining information model, we start to design prototype and make sketch for Crunching Knowledge that is identified. On Figure 6 we describe an information page from certain meeting. This prototype is used to validate domain knowledge obtained by information page from certain meeting. This prototype is used to validate the obtained domain knowledge.

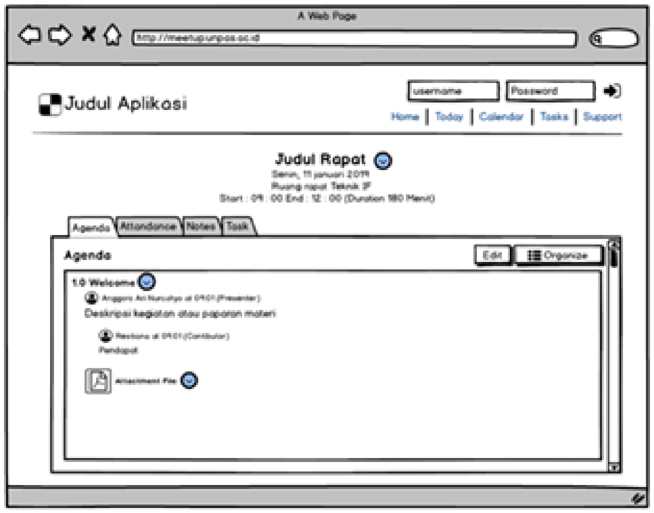
A Web Page
http://meetup.unpas.ac.id
Judul Aplikasi
username
Password
Home | Today | Calendar | Tasks | Support
Judul Rapat
Senin, 11 januari 2019
Ruang rapat Teknik IF
Start : 09 : 00 End : 12 : 00 (Duration 180 Menit)
Agenda
Attendance
Notes
Task
Agenda
Edit
Organize
1.0 Welcome
Anggora Ari Nurcahyo at 09:01 (Presenter)
Deskripsi kegiatan atau paparan materi
Restiana at 09:01 (Contibutor)
Pendapat
Attachment File

Fig. 3 Design mockup for organizer view

## V. CONCLUSION

Domain-Design Driven (DDD) is an approach to design and software development that is intended to find the more effective ways to solve a complexity of software development attached. Although this approach cannot stand by itself in software development so it is combined to software architecture concepts and software development activities.

Activities were not thoroughly discussed but it is limited to design activities that produced to interface design, because DDD is about focus on domain included the concepts, the correlation and their business logic. So it eases to developers to do work division based on the producing domain, and also language uniformity among related parties, domain experts, and developers. It eases communication happened between them, so it will impact to understanding and time for problem solving.

## ACKNOWLEDGEMENTS

I say many thanks to informatics engineering program and engineering faculty which has given facilities during the research, and for all of employees of study program who gives aids in form of power during this research.

## REFERENCES

1. D. Hirashima, M. Tanaka, and Y. Teshigawara, “Development and evaluation of a minutes system focusing on importance in a meeting,” in *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.*, 2004, vol. 2, pp. 293-298 Vol.2.
2. C. Chua and C. Woodward, “Presenting search results of meeting documents,” 2011, pp. 80–87.
3. E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Boston: Addison-Wesley, 2004.
4. E. Landre, H. Wesenberg, and H. Rønneberg, “Architectural improvement by use of strategic level domain-driven design,” 2006, p. 809.
5. B. García, J. C. Dueñas, J. I. Fernández-Villamor, A. Westerski, M. Garijo, and C. A. Iglesias, “ROMULUS: Domain Driven Design and Mashup Oriented Development Based on Open Source Java Metaframework for Pragmatic, Reliable and Secure Web Development,” 2010, pp. 186–189.
6. R. H. Steinegger, P. Giessler, B. Hippchen, and S. Abeck, “Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications to Build Microservice-Based Applications,” *SOFTENG 2017: The Third International Conference on Advances and Trends in Software Engineering*, no. April, pp. 79–87, 2017.
7. D. Esposito, *Modern web development: understanding domains, technologies, and user experience*. Redmond, Washington: Microsoft Press, 2016.
8. V. Vernon, *Implementing domain-driven design*. Upper Saddle River, NJ: Addision-Wesley, 2013.
9. I. Sommerville, *Software engineering*, Tenth edition, Global edition. Boston Columbus Indianapolis New York San Francisco Hoboken Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo: Pearson, 2016.