applied sciences

MDPI

*Article*

# An Activity Theory-Based Approach for Context Analysis, Design and Evolution

**Ismael Camargo-Henríquez and Andrés Silva** *,†

ETS de Ingenieros Informáticos, Universidad Politécnica de Madrid-Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain; i.camargo@alumnos.upm.es

* Correspondence: asilva@fi.upm.es

† Group of Biomedical Informatics (GIB), Universidad Politécnica de Madrid (UPM)-Campus de Montengancedo, Boadilla del Monte, 28660 Madrid, Spain.

**Abstract:** This paper presents a new interdisciplinary approach to support context modeling in context-awareness software developments. The premise of this approach relies on the idea that understanding a complex socio-technical ecology, while adapting the software to its behavior and evolution, is a primary challenge to address. Thus, the paper proposes an activity theory-based approach to aid in the conception, design, development, and evolution of emerging context-aware socio-technical ecologies. The concepts and notations used by the proposed approach are illustrated through a proof of concept that demonstrates the essential ideas and their use in real scenarios. Also, the feasibility of this approach is measured empirically through an experiment. Preliminary results show how, for a context-aware software design and development team, the proposal provides a better understanding of context than alternatives and helps to outline context models by establishing relationships and interactions between socio-technical components and by anticipating potential conflicts among them. The key ideas of the proposed approach result in the ability to analyze and model social and technological contexts around perpetually evolving system ecologies as useful representations for understanding operating environments closely tied to human actions, with software as a mediator component.

**Keywords:** application software; context; software; software design; software engineering; software systems



**Citation:** Camargo-Henríquez, I.; Silva, A. An Activity Theory-Based Approach for Context Analysis, Design and Evolution. *Appl. Sci.* **2022**, *12*, 920. https://doi.org/10.3390/app12020920

Academic Editor: Elisa Quintarelli

Received: 8 November 2021
Accepted: 13 January 2022
Published: 17 January 2022



## 1. Introduction

The development of context-aware software systems is plagued by complexity. When addressing software context awareness in analysis and design, many underlying issues arise. These issues are mainly related to the nature of these software systems, especially those closely integrated with components that allow recognition, localization, or adaptation. All these components are also subject to continuous evolution, forcing context-aware software to adapt fluently to these environmental and technological changes in order to keep providing its services (e.g., environmental intelligence, ubiquitous and pervasive computing, intelligent surroundings, etc.).

The problem of how to acquire a precise understanding of the adaptation needs posed by a system, and how to do it efficiently and smartly, is particularly challenging. This problem is directly related to the definition of the requirements, which should include evolvability in synchrony with the system's surrounding environment. However, formalizing and representing those needs, at a proper and workable abstraction level, is an ill-defined and error-prone task, especially when something as fluid as context is involved.

Inadequate modeling or design can give rise to misadaptations that will result in future problems. Several studies and proposals on context-aware software engineering have been published, especially in areas such as requirements gathering [1], software architectures [2],

middleware [3,4], programming [5], and testing [6]. However, these approaches only provide pieces of a bigger puzzle that needs to be solved. The present paper deals with one relevant problem that remains open [7] in the field of context-aware software: *what are the characteristics of a general approach aimed to understand, model, and design contexts?* Also, *how might context modelization facilitate and incorporate continuous evolutions and adaptations?*

On a pragmatic level, an answer to these questions should address the need to establish some balance between the conceptual elaborations required in designing context-aware software and development cycles. The present work is rooted in the idea that designers and developers must deal with the understanding of context as the cornerstone for requirements definition and for adapting future behaviors into the software. The present work is an attempt to address this challenge, by proposing a heuristic-oriented approach rooted in a socio-technical perspective. This solution supports, as one of its contributions, the use of *activity theory* for the tasks of gathering, describing, designing, and modeling contexts. Thus, this paper presents some of the ideas behind the proposal and explains how it contributes to context-aware software development in real-world scenarios.

The paper is structured as follows: this introductory section presents general ideas about the problem addressed and its existing solutions. It also provides a quick overview of activity theory as the main theoretical and instrumental foundation of the current proposal. Section 2 focuses on understanding context complexity from a socio-human perspective, and introduces the reader to later design formalities. Section 3 describes, in practical terms, the use and application of the proposed approach through a proof of concept. Section 4 discusses empirical evidence that sustains the proposed solution, based on a comparison experiment, describing the experimental process, measured elements, applied procedures, results achieved, and potential threats to validity observed. Finally, Section 5 provides some conclusions.

### 1.1. Context: A Challenge for Definition and Modeling

The idea of "*context*" is an elusive concept that has many definitions [8–10] which are often in conflict due to ambiguous or controversial interpretations [11].

While a detailed discussion of the differences among these definitions is outside of the scope of this paper, it is important to note that, behind this lack of clarity, underlies a challenge for modeling and understanding. That is, *how to provide a logical/rational form to a concept with such a broad interpretation? How can a consistent design be created for such a conceptually elusive idea?*

As an answer to these, or similar, questions, several authors have provided solutions aimed to develop efficient modeling strategies capable of addressing contexts and context-based systems. The approaches proposed range from the use of UML to ontologies and mathematical logic [12].

For example, the use of UML and object-oriented methodologies to support context modeling aims to develop high-level functional and conceptual descriptions of the applications and their interactions with the context. There are multiple works associated with this approach [10,13,14].

On the other hand, approaches that support context modeling using mathematical logic [15,16], conceptual modeling languages [17,18], and ontologies [19,20], aim to model the services provided and the rationale behind them.

These approaches, while valuable, do not relate context-aware operations to potential socio-human behaviors, reactions, and actions, as they focus primarily on a description of the software's functionalities and the services it provides.

Moreover, most of these proposals do not simply and practically allow the modeling of contexts from the point of view of certain software dynamics, such as technological changes or self-evolution and adaptation.

In this regard, the vision guiding our approach is rooted on the changing nature of contexts, which implies the need to deal with conditions affecting the scenarios where human, social, and technological elements interact.

Addressing this need is paramount to understanding the deep interrelationship among elements that give shape to complex socio-technical systems and system ecologies (*socio-technical ecologies*).

Thus, this paper is an attempt to describe those socio-technical elements and their interrelationships in order to achieve more descriptive, concise, and practical context models. With this purpose in mind, the present proposal makes use of activity theory as the basis for a multidisciplinary approach aimed at understanding and modeling contexts. The next subsection presents the fundamental ideas related to activity theory.

### *1.2. Redefining Approaches through Activity Theory*

This subsection addresses the following questions: *Why use activity theory as an instrument for context modeling? How is this theory linked to software related issues?* Given the interaction level that technology has achieved and its increasing coupling with the surrounding context, previous context modelling approaches are insufficient, as discussed previously. Thus, the need to provide a helpful instrument for designers and developers is an opportunity for multidisciplinarity and for the application of conceptual tools already proved in other fields of study.

In this regard, activity theory has inspired deep theoretical reflections in a variety of fields such as, for example, psychology [21], education [22], or management [23]. In terms of software research, activity theory has been applied to proposals in human–computer interaction (HCI) to improve development processes based on activity analysis for design rationalization [24]. Additionally, in the field of information systems, activity theory has been used to describe how the development process is carried out [25,26].

Activity theory can be flexibly used in different disciplines due to its technology-agnostic nature and its adaptability to different situations under study. Several researchers have recognized that activity theory is holistically rich for understanding how people do tasks together and, or, with the help of sophisticated tools, even in intricate and dynamic environments [27–29]. These references hint at how helpful activity theory may be for software design and development. In summary, activity theory provides an exceptionally comprehensive basis for understanding complex human, technological, evolutionary, and organizational issues.

### *1.3. Activity Theory: Essentials*

Activity theory was born with the studies of L. Vygotsky and A. Leontiev on historical-cultural Psychology, carried out in the Soviet Union in the 1920s [21]. Activity theory conceptualizes a human activity as the unit of analysis.

In general, people have an informal idea of what an "*activity*" is, but in activity theory, "*human activity*" refers to any action that acquires its form and meaning when it is performed with a purpose, or goal, in mind.

For example, software development is a human activity performed by a team of developers. Each individual developer creates and builds the software by writing, debugging, and executing code. Their goal is to obtain a coherent product that is useful to users and that other developers can understand and improve on.

In classical activity theory, human activity is divided into three components: subject, tool, and object [30]. The subject is the person who pursues a particular goal.

This goal is the object of the activity and represents a prospective outcome that motivates and directs the subject, around which the activity is coordinated and crystallizes in a final form, when completed [31].

The object, then, defines the target, objective, or purpose of the activity. Objects in activity theory are what distinguish one activity from another. In this regard, this object of activity should not be understood as a physical object or a tangible "*thing*". The other classic element in activity theory, the tool, is the mediation mechanism or device through which an action is executed in order to achieve the object.

Tools can be physical (e.g., hammer, pencil, scissors, etc.) or mental (e.g., checklists, procedures, ideas, knowledge, experience, etc.) [32]. Consequently, the tools provide a model based on the experiences of other subjects who made similar activities, which can be improved to make them more useful or efficient.

Eventually, Y. Engeström [33] introduced some modifications to Vygotsky's and Leontiev's original theory, providing additional elements of analysis. The first of these elements is the set of rules or conditions that help determine how subjects can act. Rules are the result of social elaboration and are subject to social conditioning. Rules concern any guidelines, codes, heuristics, or conventions that guide activities and behaviors.

The second element added by Engeström is the division of labor, which accounts for the distribution of actions and operations among a community of coworkers. It can be conceptualized as a hierarchical structure that organizes and supports the activity, balancing the tasks and actions of the activity among different subjects.

Both the rules and the division of labor influence and shape a social plane known as community. Through it, groups of activities and teams of subjects can be analyzed [33].

Finally, there is the outcome element that states what the activity system produces, desired or undesired. It acts as an indicator to measure the degree of achievement of the object of the activity.

For example, the object of a particular participant in a sporting activity, such as a marathon, can be "*participate*", while the outcome can take the form of "*win*", "*lose*", "*complete the run*", etc.

A generic activity system is represented in Figure 1 as a triangle of triangles [31]. In this picture, the subject, the object, and the community are the core elements, shown as a sub-triangle within the bigger triangle. The outer elements (vertexes of the big triangle) act as intermediates between the main core ones; these intermediates are the tools, the tules, and the division of labor.

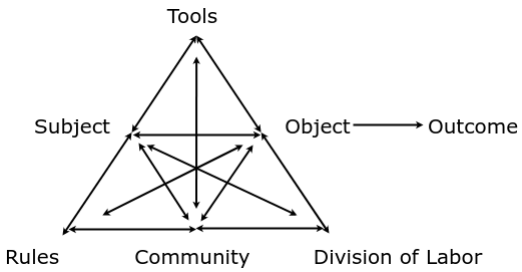


**Figure 1.** Activity System [33].

The tools intermediate between the subject and the object, the Rules intermediate between the subject and the Community and, finally, the Division of Labor mediates between the Community and the object of activity.

This triangle representation (Figure 1) allows a distributed view rooted in the nature of the activities and environments in which they occur. By focusing on the base of the triangle, we find the social elements of community, rules and division of labor. These elements describe how the work is organized towards a common object, and the outcome generated.

Another characteristic of the activity triangle is that it is not static, because changes to any element of the activity system may impact and lead to further changes in other parts of the triangle. The analysis of how activities and their contexts evolve, as proposed in this paper, will be rooted in this fact.

The following section will describe how the essential concepts of activity theory support the understanding and modeling of socio-human contexts, and how technological elements can be integrated into its dynamics.

## 2. Understanding Socio-Human Activities and the Context

Activity theory brings to the scene the relationships between physical actions, individual minds, and groups, all of them interacting while sharing a specific context [34,35].

These relationships arise from exploring the links between thinking, behavior, individual human actions, and collective group practices [36]. In this sense, activity theory provides a technology-agnostic tool, useful for diverse purposes, including its potential use as an instrument for context representation and modeling.

These characteristics of activity theory allow for the "*mapping*" of situations and provide a flexible way to describe methods and procedures performed by the humans involved [35]. This facilitates the study and analysis of different situations and promotes the search for dynamic solutions for the problem at hand thanks to the representation and modeling of the context in which those situations take place.

On the other hand, certain natural situations will arise that disrupt the harmony of a context. These interruptions are also dealt with by activity theory as opportunities that help to improve the solutions in place for the context under study. This section discusses the key ideas for understanding the nature and functioning of a complex socio-technical context from the point of view of activity theory.

### *2.1. Contradictions Leading to Better Designs*

Analysis oriented by activity theory gives key importance to the identification and treatment of "*contradictions*". This has been an important topic in activity theory since its early days and is not completely unrelated to the "*contradiction*" concept in Marxist thought.

This is not surprising, given that activity theory was born in the Soviet Union [31,33]. It must be clarified that "*contradictions*" in Marxism are not logical contradictions, and they should be understood as "*tensions*" among different elements. In the end, contradictions are the driving forces behind transformations [37]. In terms of activity theory, contradictions are defined as "*a disengagement within the elements, between them, between different activities or between different phases of development of a single activity*" [24].

In simple words, contradictions are situations or circumstances that modify the natural interactions between the elements of an activity system (shown in Figure 1) and its orientation towards the object [38], producing unexpected outcomes (positive or negative) as a result.

The important thing is that, in activity theory, contradictions can be leveraged as the "*fuel*" that drives changes. They must not be seen as problems or failures. On the contrary, contradictions provide proof that something must evolve and transform continuously [39]. This positive conception of conflicts and inconsistencies also has a long tradition in the fields of software and requirements engineering [40].

For example, if a design for a contact-tracking mobile application (based on location-awareness technology) is proposed to produce geo-spatial data needed for the analysis of the COVID-19 pandemic, a contradiction arises. On the one hand, for the designers, developers, and researchers, it could be a useful tool to support the visualization of disease escalation in a state of emergency (activity) and to suggest strategic solutions (object).

However, for civil, private, public, or political groups (community), the broad monitoring of citizens can be interpreted as a dangerous violation of fundamental rights to privacy (rules).

Thus, contradictions should lead to an "*evolution*" cycle by forcing all those involved to rethinking their strategies, actions, and decisions. In this way, some hypothetical scenarios could be considered, such as: *can a voluntary and civic use of the application be promoted? Should the design introduce options to activate/deactivate monitoring? Is it possible to limit the number of geo-locations that a user can provide? Can the data acquired be made anonymous to preserve the users' privacy?*

### 2.2. Representing Evolving Ecologies

As discussed, activity theory focuses on general human actions, carried out in a particular context. In consequence, it relies on a set of universal elements constitutive of actions, free of any specific technological implementation. This feature of activity theory allows for its use as a framework for the analysis and representation of the different areas in a domain.

Thus, activity theory supports a general representation of evolutionary ecologies as a map that relates the actions in a socio-human environment and allows for the analysis of its components (rules, tools, community, etc.). This provides an "*input*" that helps to shape the elements that take place in context-aware software systems (algorithms, sensors, networks, etc.).

Hopefully, the ecological dynamics are in harmony with the technological components. However, disruptions and mismatches will take place. These disruptions can be proactively addressed thanks to the idea of "*contradictions*". For this reason, the modeling of the socio-human context is the key that leads to a context-aware design in synchrony with its surrounding reality. As an aid to achieving this, activity theory provides another tool: *the networks of activities*, as described below.

### 2.3. From Activity Systems to Networks of Activities

Activity theory provides resources that can be leveraged in challenging situations like those described in the literature of software engineering. These situations have been studied by (i) Jarke et al., who postulate a shift in historical approaches to the development of "*systems*" [41] and (ii) P. Ralph who, through his sensemaking-coevolution-implementation (SCI) paradigm, has expressed ideas on how products and processes co-evolve, subject to continuous change [42].

These analyses point out the novel dynamics that characterize situations that are unconventional and particularly difficult to explore from a traditional software engineering point of view. In this sense, an activity-centered formalism like activity theory may help to face these challenges due to the fact that "*activities are not static or rigid entities; they are in continuous change and development*" [24], among other things.

The general principle is that, by organizing and analyzing the elements involved in an activity system, it is possible to get an idea of an evolutionary development process that includes the overall context of an activity. In other words, by using activity theory, the contexts related to evolving ecologies can be represented in a general and practical way. *But how is this achieved?*

First, as explained, the primitive unit of analysis in activity theory is the activity system, as it links individual actions to a context (Figure 1). Actions without context (purpose) have no meaning [25]. Thus, the value of activity theory derives from the analysis of the subject, in compliance with the activity and its object.

Also, there is a consideration of other elements, such as the tools, and mediators, such as the rules and community. In any case, for real-world systems, there is a need to take the analysis beyond that of a single activity. There are also complex interactions between activities, which lead to their representation as different triangle elements. This constitutes a network of activities (NoA) [31,33].

To understand how the NoA concept works think, for example, of a car assembly line. In this assembly line, each worker contributes individually to assembly pieces that, when integrated as a whole, give final shape to the car (see Figure 2).

Thus, some workers place doors, wheels, engines, etc. Each one carries out an activity with the aim of achieving a particular objective, has the tools to do it, follows some rules, and knows the details of the specific job. The joint and collaborative work of all workers leads to the collective materialization of a car.

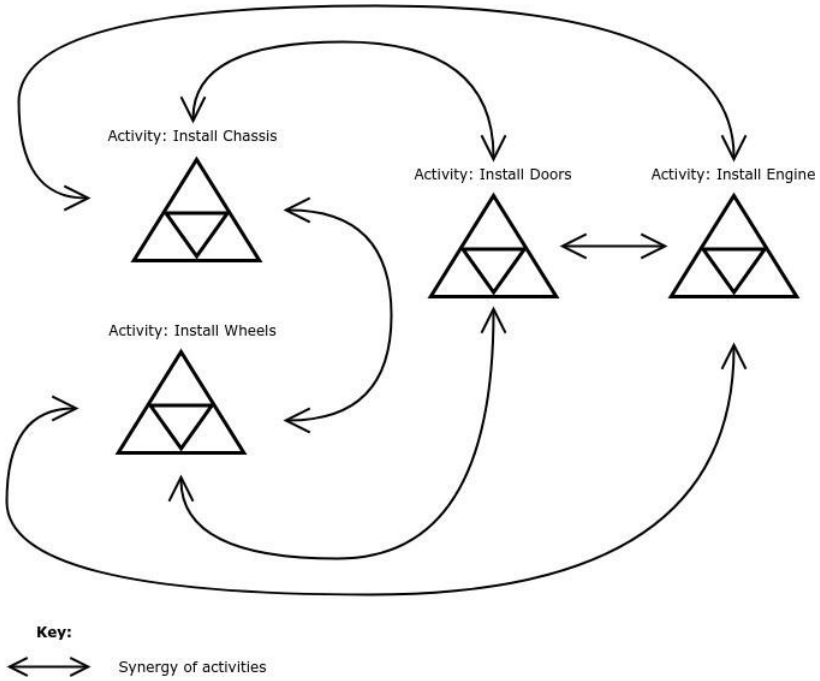


**Figure 2.** Representation of an NoA for a simplified "*ecology*" in a car assembly line.

However simplistic this example looks, the NoA concept has been applied, as shown in the works of Y. Engeström [33], V. Kaptelinin [31], and A. Mursu [25], to real-world complex situations, such as hospital management. In this way, the network of activities (NoA) provides a better model for unifying and organizing the actions, purposes, and objectives of designers, stakeholders and, generally speaking, users.

This is because, through the NoA approach, it is possible to understand the activities of different groups and to carry out a multifaceted analysis of the information and the dynamics shared among activities.

Another important point is that, by using an NoA, it is possible to evaluate contradictions, leading to some fluidity in the design of the requested artifact (software/hardware/social) within its context model.

Thus, as networks are built, measures are established to mediate and drive the evolution of the different components (subjects, tools, rules, etc.) and their interconnections. Consequently, we propose to use NoA's to operationalize, from a practical standpoint, the design and development of tasks. The next subsection presents how activity theory and the use of NoA's provide methodological and practical value for dealing with socio-technical ecologies.

### *2.4. Practical Implications and Uses*

As mentioned above, previous research related to the analysis and representation of contexts has focused on theoretical-specific applications (logic, ontology, mathematics, etc.) that can rarely be handled by real-world practitioners.

Therefore, there is a strong need for a representational schema that facilitates the process of understanding contexts by taking into account relevant factors that drive coevolution within a development process. The presented proposal fulfills this objective because one of its significant contributions is defined in terms of a logical, visual representation of socio-human actions and behaviors expressed using activity theory concepts.

However, the scope of the proposed approach is broader and incorporates other contributions that provide valuable tools for context-aware software projects. For example, NoA-based schemes contribute to strengthening the communication of ideas among those involved, as they provide a "*lingua franca*" for the development teams, designers, users, and stakeholders, which allows for the examination of important issues. Also, NoA's allow

traceability by documenting the development process, its changes, and the resolution of contradictions, etc.

This allows us to achieve other goals in the medium/long term related to software maintenance and knowledge transfer, such as providing an initial context model to other developers. In this regard, the contribution of the NoA concept to context modeling is its ability to map the social, individual, and technological elements that interact within the context under study. Thus, the NoA helps to navigate through these elements, with all their intrinsic complexity. To provide a pragmatic description of the ideas and concepts described above, a proof of concept is presented in Section 3.

The practical implications and contributions of the proposal have been experimentally tested. This has made it possible to estimate the feasibility of designing and modeling contexts through the proposed approach. Further details will be provided in Section 4.

## 3. Putting Everything into Practice

The previous sections were descriptive and theoretical. Now, a practical scenario will be used as a proof of concept. For this purpose, we present a scenario in which a socio-technical ecology is developed for a "*just walk out*" (JWO) store, which is a novel approach to shopping being implemented by companies like Amazon. The idea of a JWO store involves a combination of context-aware technologies (devices, sensors, cameras) and software, to establish "*who took what*" from the store.

In general, its operation is quite simple: a shopper creates and registers an account with the store's owner, providing a credit card and using a compatible smartphone with the official free app installed.

To enter the store, the shopper scans his/her identification code (based on QR) through the app at the store entrance to activate the account and the virtual cart. This "*virtual cart*" acts as a metaphor for the system, supporting the generation of the customer's invoice and the store's inventory control when adding or removing items. Thus, the shopper can pick up items from the shelf and add them to the virtual cart (without needing to scan each item).

For more convenience, the store offers the possibility of placing the purchased items in bags or baskets held by the shopper. If the shopper puts the item back on the shelf, the item is removed from the virtual cart. At the end of the purchase and after leaving the store, the shopper is automatically charged with the full amount of the purchase on his/her account and then debited from his/her credit card, and all this is without any interaction with a cashier.

This proof of concept will be useful for understanding the main ideas of this paper and to show the construction of an NoA for the purpose of representing the context. The analysis will focus on activity evolution, by way of identifying and resolving contradictions. The aim is to illustrate the practical application of the theoretical concepts involved.

### *3.1. Defining a General Scenario*

For reasons of simplicity, this proof of concept is centered on the development of activity systems with few agents, such as the shopper and the JWO-store system, that will be described below.

The store's internal management, and its relationships with suppliers, inventory, etc., have been excluded. Thus, we will not discuss its business structure or its role, represented by store staff, in the general ecology. Furthermore, the flow of interactions that arise between the shopper and the JWO-store system is much more enriching (in socio-technical terms) and interesting (in conceptual terms) than the description and modeling of the store's administrative core. Thus, as mentioned above, the following agents will be considered in detail:

- *Shopper*: a person who purchases and is a regular consumer of one or more products offered by the store. To interact in the store, the shopper must have a credit card associated with a store account and a smartphone with the required app.

- *JWOsStore system*: this is the core system of the store. Essentially, this system detects the entry and movements of the shopper. Also, it automatically activates the tracking of products when they are taken from or returned to the shelves. Finally, it manages payments by sending a receipt with the purchased amount to the shopper's account. It also performs notification tasks to the staff about the inventory status.

The JWO-store system is an active agent within the ecology because is the entity that all shoppers interact with and, as an agent, its activities and actions are similar to the ones performed by a traditional store worker.

### 3.2. Building the NoA

The building of an NoA following the postulates of activity theory involves finding the elements that structure the activity system for each of the subjects involved. For example, let us consider the shopper as an initial subject. As described, the shopper performs a series of tasks related to the activity of purchasing products from the store. These tasks are guided by basic rules such as, for instance, the ownership of a credit card. On the one hand, this rule is required as a precondition for some tasks, also subject to rules, on how to purchase products and pay for them, etc.

On the other hand, the tool that the activity system provides to the shopper is materialized by a software artifact (the smartphone app), which must be complemented by other tools, such as a network of sensors. The shopper's role is very different from the JWO-store system's role. This specialization of the activity systems, between shopper and JWO-store system represents a diversification of tasks, i.e., what the shopper does is not done by the other agent and *vice-versa*. In activity theory language, this is defined as the division of labor.

However, although this apparent "*division*" seems to separate the roles of each one in terms of interaction, the shopper has a relationship with the JWO-store system, so they also constitute a community.

In this way, and using the same general analysis for each participating agent, it is possible to describe the systems (triangles) and how they take part in an NoA. This NoA represents an ecology in which the different activity systems "*live*" and interact. The detailed activities for each particular subject and their corresponding elements are shown in Tables 1 and 2 respectively.

**Table 1.** Key Activity System for the Shopper.

| Element | Description |
| --- | --- |
| **Subject:** | *Shopper* |
| **Object:** | *Get products* |
| **Tools:** | *Smartphone/app* |
| **Rules:** | |
| | R1: *(Must) Have a credit card* |
| | R2: *(Must) Scan the app to get into the store* |
| | R3: *(Can) Pick up items from the shelf* |
| | R4: *(Can) Put items back on the shelf* |
| | R5: *(Must) Pay full purchase amount* |
| **Division of Labor:** | |
| | *Shopper: Purchases items. Pays the bill* |
| | *JWO-store system:* |
| | *Protect shoppers* |
| | *Improve the shopping experience* |
| **Community:** | |
| | *JWO-store system* |
| | *Other shoppers* |
| | *Store staff* |

**Table 2.** Key Activity System for the JWO-Store System.

| Element | Description |
|---|---|
| **Subject:** | *JWO-store system* |
| **Object:** | *Get profit* |
| **Tools:** | *Cameras/sensors/AI/computer vision* |
| **Rules:** | |
| | R1: *(Must) Verify shopper's entry* |
| | R2: *(Must) Activate tracking sensors* |
| | R3: *(Can) Provide product information* |
| | R4: *(Must) Check if an item is picked up from a shelf* |
| | R5: *(Must) Check if an item is put back on a shelf* |
| | R6: *(Must) Calculate purchase amount* |
| | R7: *(Must) Verify the shopper's exit from the store* |
| **Division of Labor:** | |
| | *Shopper: Purchases items. Pays the bill* |
| | *JWO-store system:* |
| | *Check shoppers in and out* |
| | *Calculate purchase amount* |
| | *Report inventory status* |
| **Community:** | |
| | *Other shoppers* |
| | *Store staff* |

Graphically, the activity triangles of the subjects involved can be represented (simplified), as in Figure 3. In the figure, the individual activities connected constitute an initial NoA. The details of the tools, rules, community, and division of Labor have been described in Tables 1 and 2 and have not been included in Figure 3.

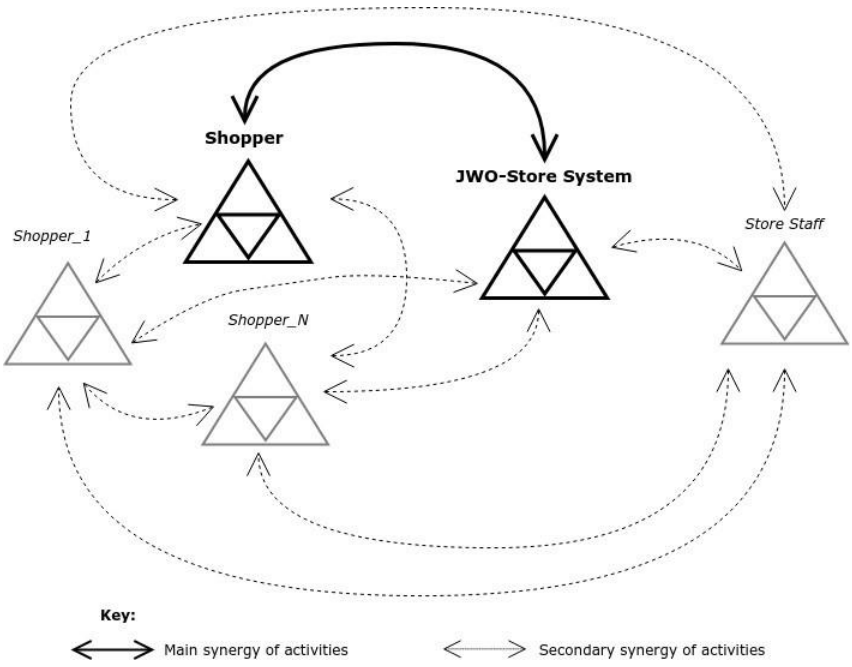


**Figure 3.** Representation of the NoA for the scenario proposed (simplified form).

However, it must be assumed that all the elements are present there and take part, in parallel, in other interactions and interrelations with the elements of other activity systems in the ecology.

### *3.3. Identifying Contradictions and Evolving the NoA's Ecology*

Contradictions, conflicts, and inconsistencies are frequent in any human endeavor, and this is reflected in activity systems. However, as discussed above, activity theory sees contradictions as positive, using them to stimulate and elicit further needs.

Of course, these problems also emerge in the JWO's scenario. It does not require a lot of time for contradictions to emerge, especially those affecting the subjects and some of the rules. However, contradictions also affect other elements, such as the division of labor and the tools, etc.

Figure 4 organizes and expands the descriptions of the rules that were initially established (see Tables 1 and 2). The information in Figure 4 allows, firstly, for the reconstruction of the general guidelines that shape the actions of the agents involved (shopper, JWO-store system) and, secondly, for the understanding and outlining of how contradictions arise and impact the process.

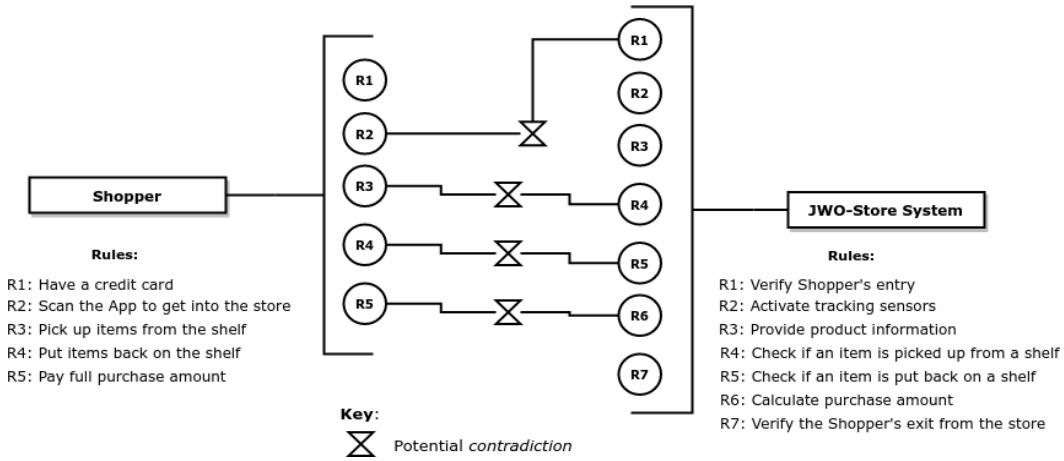


**Figure 4.** Representation of the NoA for the scenario proposed (simplified form).

For example, in Figure 4, the shopper's rules are represented by circles and connectors that are interrelated with the rules of the JWO-store system. As shown, there are rules for each subject which do not lead to contradictions, such as R1 (have a credit card) for the Shopper, or R2 (activate tracking sensors) for the JWO-store system.

Other rules, however, are the source of "*tensions*", which lead to contradictions in the NoA and consequently within the ecology. For instance, in the shopper's activity system, R2 (scan the app to get into the store) interacts with R1 (verify shopper's entry) of the JWO-store system.

Here a simple contradiction may occur that could lead to a change. For example, *could a shopper get in and use his/her code consecutively, allowing access to the store for one or more companions? How should the JWO-store system react to this situation*?

Other elements are a little more complex (see Figure 4). Concerning the dynamics of the store's products, the shopper can add them to his/her purchase (R3) or return them to the shelf (R4). This is in "*symbiosis*" with the JWO-store system and its corresponding rules related to these actions (R4 and R5).

However, this may again generate contradictions. Based on the validity of the rules, the shopper can add or remove as many products as required from/to the shelves, and the JWO-store system will react accordingly, using its tools (sensors, cameras, etc.) to perform the corresponding updates to the shopper's virtual cart.

However, *how should the JWO-store system react if a shopper takes products from the basket/bag from another shopper?* This directly affects the shopper's rule R5 (say full purchase amount) and the R6 (calculate purchase amount) of the JWO-store system.

Thus, scenarios may be raised in the requirements phase for addressing questions such as*: are the tools of the JWO-store system enough, or should it use additional ones? Perhaps, baskets*

*with sensors?* These "*what if?*" scenarios pose questions that trigger reflections about the possible handling of the contradictions in the NoA and, from these questions, they enact the (re) design of "*solutions*" which, after some time, will have to confront other contradictions that will surely arise in the future. This is, after all, a "*sensemaking*" task as defined by P. Ralph [42].

It is relevant to highlight that, in general, contradictions operate as active clues used to understand, guide, and propose solutions to potential disruptive actions carried out by the subject(s) within the activity system or the NoA. However, these contradictions cannot always be predicted the during the design stage.

The appreciation of contradictions can be established in terms of two principles: (i) when predictable, the analysis of contradictions will trigger solutions, but also (ii) when unpredictable, contradictions will drive further evolution as they emerge from empirical reality. The most interesting contradictions are those that surge from real unexpected use within a context. These are those that surprise the designers and developers and generate the most interesting and evolutionary changes.

For instance, YouTube was initially a platform for the publication of videos. Subsequently, its users found ways to extract the audio from the songs in the videos and also ways to upload movies and other copyrighted material, temporarily turning YouTube into a provider of non-legal music, video, and content [43]. The site was forced to co- evolve, given this context, to resolve the conflicts.

### *3.4. From an Evolving Context Model towards Better Design Solutions*

As described so far, the whole process is highly dynamic. At the level of the NoA, even if the objects of each subject are achieved, the process provides feedback as soon as changes in the context take place.

These changes emerge when other subjects are included, other activities are created, new functions are added, designs are changed, or objectives are redefined, etc. This dynamism is stimulated, essentially, by new contradictions that can be better understood through the optics of the NoA and by analyzing the subjects that act together and within their corresponding activities.

Similarly, contradictions will arise unrelated to particular activities within the NoA, but that may affect these activities collectively, such as a change in the basic technologies used. In this way, over time, the software designers and the NoA's participating subjects will be in a constant cycle of production and analysis of contradictions, which lead to questions; these questions lead to resolutions through "*sensemaking*"; then, solution spaces will arise through "*coevolution*" and these solutions will acquire a tangible form through "*implementation*", etc., thus recreating the sensemaking-coevolution-implementation cycle discussed by P. Ralph [42].

In the JWO-store system, for example, this cycle takes place while contextual elements keep changing and, consequently, the harmony of the ecology is altered, affecting the activity systems and the NoA, etc. For example, a policy change, such as selling basic drugs in the store (acetaminophen, ibuprofen, cough suppressants) would automatically disrupt the whole set up through new contradictions; it would force a redefinition of all or part of the operation of the NoA (Figure 3) with implications that range from the social (communities, divisions of labor, rules) to the technical (redesign of the app/system).

Thus, the novel contribution of the NoA concept to context modeling is the possibility it brings of creating a map of the social, individual, and technological elements that interact within the context at hand. Thus, it allows for the summarizing of the complexity of these elements and its constraints. For reasons of space, the examples provided here are too general, but they reflect this core idea.

In conclusion, some practical support has been provided to structure, analyze the operation, and explain the potential use of the proposal. However, to validate and verify the accuracy, acceptance, and usefulness of the approach for developers, a testing exercise was developed.

## 4. Testing Ideas through Experimentation

This section describes an experimental design that was carried out by the authors to estimate the real-world feasibility of an activity theory-based approach for designing and modeling contexts, as part of a context-aware software-centered system. The main objective was to estimate the impact of the proposed approach versus other modeling instruments, for the same setting. More specifically, the aim was to find measurable results for both approaches, for the same case study. In this sense, as a benchmark to compare with, an UML-based approach was chosen.

The choice of using an UML-based approach as a method to contrast the proposed on took into account the fact that UML has a proven tradition in the history of software analysis and development. Additionally, UML is used as a tool to design and model contexts. This has been reported in different implementation cases and exposed in a wide variety of related works [13,44–52].

In this sense, using different UML notations, as suggested by the cited works, it is possible to group elements of different diagrams, in order to express which information is related to a specific context and define it as a model. For example, *use case diagrams* may be used to demarcate a system and its environment, but also *actors* and *components* directly connected by *associations* can represent a system context (if *actors* are represented by rectangles instead of the usual figures).

Another aspect considered was the general knowledge on the subject possessed by the selected individuals of the population under study. That population consisted of software developers in the role of evaluators for the examined approaches. Further details will be provided later.

Regarding the methods used to conduct the research, these were formulated via research questions, leading to two general hypotheses to be considered. The research questions were the following:

**RQ1:** *Can activity theory heuristics and approaches provide a significant impact on how to obtain, analyze, and modeling software contexts?*

This research question is rooted in the existence of gaps that can, and should, be filled by approaches concerned with the gathering, analysis, and modeling of evolving software contexts. There are several measurable attributes that point to a certain inflexibility of traditional instruments with regard to the novelty of needs for gathering software context information. In particular, there are certain considerations associated with social and human factors which often escape classical approaches.

**RQ2:** *Can a series of objective measurements assess the impact of an activity theory-based approach for the analysis and modeling of software contexts?*

This question allows for the investigation of the design of qualitative and/or quantitative indicators, which can be applied to the evaluated approach. Thus, the values of these indicators would lead to the establishment of criteria regarding the impact of this approach for the analysis and modeling of software contexts.

**RQ3:** *Is it possible to compare the measurements obtained from an activity theory-based approach with widely used proposals such as a UML-based approach?*

Indeed, if it is possible to arrive to a series of mechanisms or criteria to measure and evaluate an activity theory-based approach for dealing with software contexts, these could be used to establish objective comparisons with other methods. Such would be the case of UML-based approaches, as well as any other traditional or current instrument that could be subjected to this evaluation and comparison.

These research questions were aimed towards establishing the truth value of the following hypotheses:

**Hypothesis 1 (H1):** *The use of an activity theory-based approach does not significantly influence context modeling for context-aware software development.*

**Hypothesis 2 (H2):** *The use of an activity theory-based approach significantly influences context modeling for context-aware software development.*

In particular, considering that these proposals for gathering and analyzing software contexts are tied to complex socio-technical processes, for their conversion into representational elements and technical specifications we chose an exploratory approach, defining four factors to be measured.

These four factors cover the complexity of the heuristic aspects needed by an approach aimed to obtaining and analyzing software contexts regardless of the particular implementation, procedural structure, etc. Each of these factors is described in detail below.

### *4.1. Measurement Factors and Rating Scales*

To provide an objective comparison between an activity theory-based approach vs. an UML-based approach, four factors of interest were measured. These factors are the result of several instruments designed to understand the complexity associated with the process of acquisition, analysis, design, and modeling of contexts.

Some relevant questions in this regard are, for instance, *what should be measured? how should it be measured*? and, *how should the measurements be compared?* The development of these questions entails an underlying methodological procedure that led to the design of the factors presented.

Initially, the problem of finding what should be measured, particularly in complex environments, such as context-aware software development, can be an unmanageable task if it starts from theoretical methods or from general metrics, such as cost and effort estimation, lines of code (LOC), code complexity, etc. For this paper, a more pragmatic approach was chosen, where twelve specialists (researchers, designers, and developers) were interviewed about which aspects they considered to be the ideal for this type of development.

Thus, a survey was carried out to collect some data from these specialists. The results and answers, according to the criteria and experience of those interviewed, converged in several topics such as the *ability to easily interpret diagrams*, *the richness of details in design schemas*, the *capacity to perform collaborative tasks within the project*, the *possibility of easily introducing changes and outlining solutions to conflicting elements in the project*, among other aspects.

The opinions that pointed towards a regular observable pattern were synthesized in the form of four factors (shown in Table 3). Other criteria, subordinated to the previous general opinions, were organized as sub-items (shown in Table 4).

**Table 3.** Measurement factors to be evaluated.

| Factor | Questions to Research/Reply |
| --- | --- |
| **[A]: Context Richness** | Does the approach represent the context and conceptualize an overview of the problem?<br>Does the approach induce the elicitation of the context? |
| **[B]: Understandability** | Can the approaching express, describe, and be understood by human agents (users, stakeholders, design and development team)? |
| **[C]: Collaboration** | Does the approach allow us to build a chronology and follow-up of the process?<br>Is it possible through the approach to managing changes in context? |
| **[D]: Traceability** | Does the approach produce documentation?<br>What kind of documents does it produce?<br>Is the documentation produced sufficiently communicative, comprehensive, and descriptive? |

**Table 4.** Sub-Items evaluated for each factor.

| Factor | Sub-Item | Related with |
|---|---|---|
| [A] | **A.1:** | Acquisition of inputs and outputs for the analyzed context |
| | **A.2:** | Identification of the human actors/actors involved and their roles in the situation under study |
| | **A.3:** | Addressing of potential restrictions, conflicts, incidents, or risks in the modeled context |
| [B] | **B.1:** | Effective visualization of potential unwanted events |
| | **B.2:** | Defines a clear context scope |
| | **B.3:** | Specifies contexts models clearly to human agents (users, stakeholders, design and development team) |
| | **B.4:** | Use a standard mechanism to express the modeled context unambiguously |
| [C] | **C.1:** | Change management and context problems can be solved efficiently |
| | **C.2:** | The collaboration and interaction between human agents (users, stakeholders, design and development team) is fluid and standard |
| | **C.3:** | The features and functional attributes of the design for the context-aware software are clearly defined through its context model |
| | **C.4:** | There is an evolution and feedback of knowledge that refine the acquisition of the context and allow traceability |
| [D] | **D.1:** | The documentation generated is consistent with the development and design of the model context developed |
| | **D.2:** | Human agents (users, stakeholders, design and development team) can quickly become familiar with the documents, diagrams, and schemes produced and communicate discrepancies, errors, or incidents using them |
| | **D.3:** | The design and development team uses and understands the documentation generated |

In consequence, this procedure provided the factors that will be used in this study for comparing an activity theory-based approach with a UML-based approach. The details of these factors, and how they were used in this study, are described as follows:

*[A] Context Richness:* The approach evaluated must allow for a rich representation of the context, providing enough information to model some essential properties such as, for instance, the profile of the human actors/agents involved, the relationship between the processes performed, and the expected results and objectives, etc.

*[B] Understandability:* This factor evaluates if the applied approach offers mechanisms that gradually lead designers/developers to an efficient elicitation of the intentions in play within a context (what should be done). Also, this topic attempts to measure features that facilitate the statement of purposes (how it should be done) using consistent expressions to describe the functionality and/or rules that allow for the defining of the context and the ability to rationalize several possible scenarios, mainly those unwanted, etc.

*[C] Collaboration:* A key feature to evaluate in the applied approach is how it facilitates teamwork between all those involved, using elements such as the definition of a common vision between users/stakeholders and the design/development team. This also includes the capacity to register and track the context model produced during the entire development process, i.e., its resources to delimit, document, manage changes and control the context modeled, etc.

*[D] Traceability:* This factor evaluates if the applied approach can generate useful artifacts for communication and documentation (diagrams, schemes, reports, etc.). This documentation is intended to be accurate, complete, consistent, and above all, understandable for users, stakeholders and design and development teams, helping to ensure fluid communication. In Table 3, we show the measurement factors described for the experiment and the questions that motivate its application to the approaches selected for evaluation.

The four factors indicated in Table 3 are further divided into the sub-items in Table 4. These sub-items allow for the structuring of the properties that we intend to identify and evaluate in each of the selected approaches. Thus, comparable factors can be obtained.

Regarding the rating scales, it was considered that, in a sample of individuals (evaluators), each one will have a different perspective about each factor (Table 3) and its sub-items (Table 4). Likewise, the evaluator's criteria will be used to evaluate and weigh the approach reviewed (activity theory-based approach or UML-based approach). As the data collected should avoid being biased, a rating scale, within a sample of variable size, will reduce or eliminate the margin of error in the data obtained.

Consequently, a model based on a Likert scale [53] was considered as the most appropriate option. The scale used in this work is summarized in Table 5.

**Table 5.** Scale of values to be evaluated.

| Value | Description | Definition |
|---|---|---|
| 1 | Very Poor | The evaluated sub-item has no significant value |
| 2 | Poor | There is some kind of minimum significant value in the evaluated sub-item |
| 3 | Fair | The evaluated sub-item has a neutral value (neither too much/nor too little) |
| 4 | Good | There is a significantly high value in the evaluated sub-item |
| 5 | Excellent | There is a maximum value in the evaluated sub-item |

To make estimates, each sub-item in Table 4 was weighted based on the scale values in Table 5. The average of the sum for each sub-item evaluated establishes the value total obtained by each factor of Table 3. This procedure is expressed by the following formula:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

where $\overline{x}$: It is the value of a measurement factor according to Table 3, based on the average of each sub-item $x_i$ that corresponds to that factor, $x_i$: Corresponds to the value of each sub-item in a measuring factor. Its value is weighted according to the scale proposed in Table 5, $n$: The number of sub-items in a factor according to Table 4.

### *4.2. Population Overview, Sample Selection, and Data Gathering*

The selected population of the experiment was based on a global group ($\gamma$) of one-hundred-and-twenty (120) software developers with diverse levels of experience (3 to 7 years), who acted as evaluators. This global group was subdivided into two groups: Alpha ($\alpha$) and Beta ($\beta$) of sixty (60) evaluators respectively.

The members of the Alpha ($\alpha$) group were taught the UML-based approach for context modeling, while those in the Beta ($\beta$) group were trained in the use of the activity theory-based approach. The theoretical and practical preparation of both groups ($\alpha$ and $\beta$) was done separately, following a five-session process as shown in Figure 5.

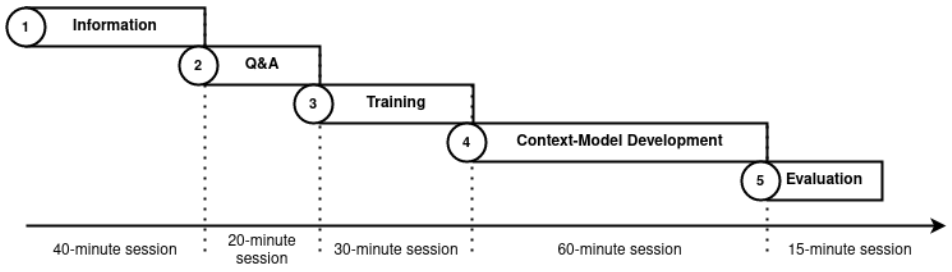


**Figure 5.** Process for training evaluators and data gathering.

First, a 40-min *Information* session took place, in which the evaluators were informed about the objective of the experiment and about the fundamentals of the approach to be evaluated, according to their assigned group (activity theory-based or UML-based).

Afterward, a second session, called Q&A (*Questions and Answers*) was offered to clarify doubts, concepts, and ideas that the evaluators perhaps did not understand in the first session. This second session lasted 20 min. After both sessions, a third 30-min session, called *Training*, was held (see Figure 5). In this session, a brief guided case was developed, using the approach assigned to each group. In the fourth session, a specific problem was presented for the evaluators to (individually) apply the procedure and model the necessary context.

The proposed problem to solve was about the characteristics of a context-aware software system for people with visual disabilities.

The general idea was to design/model a context in which this software helps its user to find important points within a University campus, depending on the location (i.e., cafeterias, libraries, emergency doors, bathrooms, halls, etc.). This fourth session was called *Context Model Development* and was conducted over 60 min.

In the fifth and last session, called Evaluation, the evaluators weighted the approach assigned to their group (the activity theory-based approach or the UML-based approach). For this, the specific sub-items described in Table 4 were scored using the scale in Table 5. The overall result, according to the formula presented in Section 4.1, provided a score for each of the four factors defined in Table 3, obtaining the evaluator's assessment for the reviewed approach.

The evaluation process was designed to be done in 15 min through a survey available through a web platform. In the following subsection, the results are summarized and discussed.

### *4.3. Results and Findings*

For each context modeling approach reviewed by groups $\alpha$ and $\beta$, 840 evaluations were collected. The initial estimation is quite simple: 60 evaluators $\times$ 14 assessment sub-items = 840. Thus, 1680 evaluations were totaled, which is 840 from group $\alpha$ plus 840 from group $\beta$.

As explained above, for each factor rated, its sub-items were weighted according to a value scale between 1 and 5. Thus, each factor evaluated was classified with its corresponding value, according to its category ([A], [B], [C], or [D]). From this classification, overall averages were obtained for each factor. These final averages represent the evaluators' assessment for the examined approach in terms of the factors considered. Thus, these results are summarized and shown in Figure 6.

In this regard, for factors [A] *context richness* and [D] *traceability*, the activity theory-based approach shows exceptional values. Its lower value is identical in factors [B] *understandability* and [C] *collaboration*.

Although a more detailed discussion of this situation will be provided, it is noteworthy that the lowest values identified in factors [B] and [C] in the activity theory-based approach are higher than the maximum values for factors [A], [B], and [C] of the UML-based approach (see Figure 6).

Regarding the UML-based approach, the value of factor [D] is exceptional in that it is the most outstanding; however, it is still far from the value obtained by the activity theory-based approach in the same factor considered.

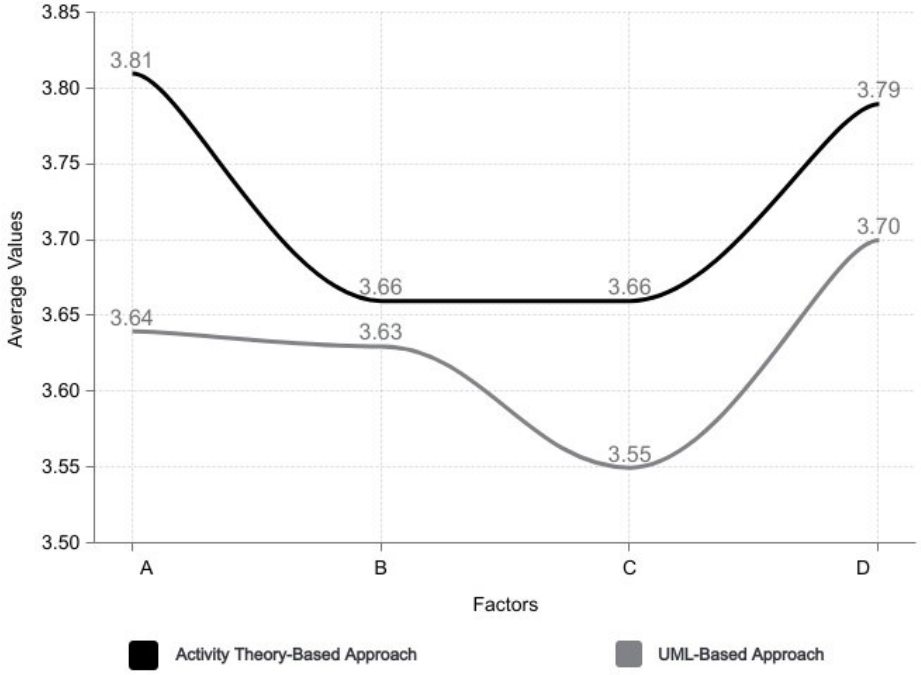


**Figure 6.** Final average values for the reviewed approaches regarding the measurement factors.

### *4.4. Discussion*

As the reader may realize, implementing the activity theory heuristics provides a significant impact on how to obtain, design, and model the context for potential context-aware software applications. This notoriety is related, in our opinion, to the socio-technical character on which it is based, as it is rooted in the theory and conceptualization of the root proposal, which in turn is the main element which differentiates it from other approaches.

In the activity theory-based approach, the development of activity systems provides an understandable and fairly coherent metaphor. In this sense, this metaphor describes the representation of a general context using key elements associated with the activity under analysis.

This property is what makes the context understandable from two points of view: (a) for the developer/designer, who tries to extract its properties, and (b) for the stakeholder/user, who provides, directly or indirectly, the primary information (needs, procedures, etc.). This two-way process provides, for instance, a guide to a common understanding of the context (as described by factor [A]) and facilitates and increases the exchange of information (as proposed by factor [D]).

These affirmations about factors [A] and [D] have been verified in practice, according to the values shown in Figure 6.

Regarding the alternative approach used in the study (UML), there are interesting values for factors [A] and [D]. The results seem to suggest that, in the evaluators' experience, the use of the metaphors provided by the UML-based approach (to represent subjects, actions, tasks, etc.) creates a semantic barrier that slows down the "*fluidity*" and understanding of the produced model and its composing elements. This is strongly related to the degree of comprehension of the context. The resources of the UML-based approach (the UML's diagrams) lead to a lower understanding, in some way, of the subtleties, abstractions, and meanings of the metaphors used (lines, flows, symbols) for the non-specialized human agent.

Concerning factors [B] and [C], when using an activity theory-based approach, both factors had identical values (see Figure 6). These values were still higher than those obtained by the UML-based approach, for the same factors. The assessments suggest that this approach better stimulates the aspects associated with the effectiveness of the task of building, communicating, and transferring technical informational elements.

This is related to the intentions and purposes behind the context model, the procedures for interactions between designers–stakeholders, and changes in management, etc. Even so, the similarity of the values obtained could also suggest that there may be improvements rooted in the mechanisms that guide an activity theory-based approach. In this regard, better ways to articulate the processes of acquisition, modeling, and the design of contexts, such as the automation of some tasks, could be introduced (e.g., "*visual*" construction of the elements of an activity system, or the saving, editing, and modifying of its elements, etc.).

An additional insight provided by the experiment is the feasibility of establishing elements that measure and estimate the impact of the activity theory-based approach. To achieve this, we proposed the four factors to analyze and their corresponding sub-items (Tables 3 and 4). Although it is possible to extend them for other similar studies, we consider that these factors achieve the necessary objectivity required *to demonstrate the feasibility of the activity theory-based approach for context modeling in context-aware software-centric system development*.

### 4.5. An Additional Validation

It could be stated that, although the sampled population can be considered small (120 evaluators), a Chi-square [54] test demonstrates that it has an optimal size for estimating the statistical significance of the experiment [55]. Thus, a Chi-square test was applied to the collected data set to estimate the degree of similarity between observed and expected frequencies on the data.

First, the total number of weightings that evaluators assigned to each sub-item (Table 4) was counted, using the value scale in Table 5. The results of this count were grouped into categories (the very poor and poor values were combined into a "*Weak Values*" category, the fair values remained in the same category, called "*Fair Values*", and the good and excellent values were added together into a category called "*Outstanding Values*". These values became the *observed results*, the results that were measured in the experiment. From these observed results and their corresponding totals, we calculated a set of approximate values which, in terms of the Chi-square test, are called *expected results*. Then, the Chi- square test examined the difference between those expected and observed values (shown in Table 6).

**Table 6.** Values grouping and distribution between observed and expected results for the Chi-square test.

| | **Activity Theory-Based Approach** | | **UML-Based Approach** | |
|---|---|---|---|---|
| | Observed | Expected | Observed | Expected |
| Weak Values | 140 | *148.5* | 157 | *148.5* |
| Fair Values | 186 | *206* | 226 | *206* |
| Outstanding Values | 514 | *485.5* | 457 | *485.5* |
| Total | 840 | | 840 | |

Thus, applying the Chi-square test to the values in Table 6, a *p-frequency* value estimated at 0.016551193 was obtained. This *p-value* means the risk of rejecting a real hypothesis. This result ($p$ = 0.016551193) is lower than the significance level established by the Chi-square test (by default, $p$ = 0.05). From this p, it can be stated that, for the data collected in the experiment, there is a clear association between the observed and expected results.

This allows for the affirmation of the H2 hypothesis, that *the use of an activity theory-based approach significantly influences context modeling for context-aware software development*.

### 4.6. Threats to Validity

In this section, we discuss some elements that may be considered as possible threats to the validity of the experiment. These are:

(a) *The population and the sample*: The results and the repeatability of the experiment may vary depending on the heterogeneity and characteristics of the particular group of evaluators, such as their experience in software projects/development. Also, the number

of members of the selected population and the conditions (motivation, commitment) under which the experiment is performed may affect its results.

(b) *The environment*: The preparation and data gathering process can be significantly influenced by the environment selected and the physical conditions (e.g., meeting space) or virtual (e.g., internet access, computers) needed to bring all the evaluators together.

(c) *The survey*: The web-based survey is not without its problems. For example, a large population with a poor understanding of the process objectives in a dispersed environment, etc., could lead to incomplete data or null values and therefore failures in the data treatment, creating biased results.

## 5. Conclusions

This article is centered on some of the key ideas behind a proposal founded on activity theory, which, among its properties, provides the ability to analyze and model complex socio-technical contexts around perpetually evolving system ecologies. According to this approach, these ecologies become useful representations for understanding operating environments closely tied to human actions which influence, and are influenced by, software behavior. Under this premise, and combined with software-centered technological elements, the understanding and clarification of opaque aspects can be increased, leading to better and well-adapted designs and models, especially in conditions where it is necessary to deal with the meaning of a "*context*" and its changing dynamics.

In this regard, one important topic is "*contradictions*", specifically when they are considered as a mechanism towards adaptive solutions rooted in unusual (but not necessarily unexpected) events that disrupt the "*harmony*" of a socio-technical ecology. These events are usually associated with human and social behaviors which cannot always be predicted or designed beforehand but that, nevertheless, constitute the basis of the evolution of a technical solution.

Thus, under these situations, continuous adjustments are required, as other studies [41,42] have theorized but have not applied in a practical setting.

Additionally, the user acceptability of the proposed activity theory approach was tested through an experiment, showing it to be better adapted to perceptions and usage by developers concerned with the specific factors discussed. In this regard, answers were gathered for the research questions, as follows.

For **RQ1** the values obtained showed a relevant impact between obtaining and analyzing software contexts using an activity theory-based approach when compared with an UML-based approach. As shown in the analysis of the factors in Figure 6, the values obtained by the activity theory approach showed that, in practice, it was regarded as a better mechanism for gathering and analyzing contexts.

Regarding **RQ2**, it is feasible to establish indicators that measure and estimate the impact of an activity theory approach. Our proposal was to formulate the four factors analyzed and their corresponding subitems. Although it is possible to extend these for subsequent research, we consider that they achieved the goals of the present study.

Consequently, **RQ3** was also positively answered, since the formulation of the evaluated factors and their structuring, as well as their weighting through a Likert scale procedure, proved the possibility of carrying out comparisons between activity theory approaches versus alternatives.

Although these results are still primary, they open up a promising path as a starting point for new contributions and inspirations regarding advances in context-aware software development.

Finally, this novel approach, through the concepts provided by activity theory, may lead to potential applications in other related areas of software engineering, such as, for example, requirements engineering, by facilitating the analysis of complex socio-technical ecologies and their evolution. In this regard, it's a potentially useful instrument for situations in which socio-technical complexity must be broken down and reduced to un-

derstandable and analyzable elements (activity systems) that, at first glance, traditional methods do not pay attention to.

Thus, an activity theory approach helps to analyze potential design decisions and solutions through "*contradictions*", in turn helping to describe the underlying elements that configure a context. In addition, by using an activity theory approach, designs can be improved to better reflect the intrinsic fluidity of the socio-technical elements involved, particularly the human ones (behaviors, actions, decisions, objectives, etc.).

**Author Contributions:** Conceptualization, I.C.-H. and A.S.; methodology, I.C.-H.; software, I.C.-H.; validation, A.S. and I.C.-H.; formal analysis, I.C.-H.; investigation, I.C.-H. and A.S.; resources, A.S.; data curation, I.C.-H.; writing—original draft preparation, I.C.-H. and A.S.; writing—review and editing, I.C.-H. and A.S.; visualization, I.C and A.S.; supervision, A.S.; project administration, A.S.; funding acquisition, A.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Acknowledgments:** This work is supported by "*Proyecto colaborativo de integración de datos genómicos (CICLOGEN)*" PI17/01561 funded by the Carlos III Health Institute from the Spanish National Plan for Scientific and Technical Research and Innovation 2017–2020 and the European Regional Development Funds (FEDER).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Choi, J. Context-Driven Requirements Analysis. In *Computational Science and Its Applications—ICCSA 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 739–748. [CrossRef]
2. Choi, J. Software Architecture for Extensible Context-Aware Systems. In Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology, IEEE Computer Society, Daejeon, Korea, 28–30 August 2008; pp. 811–816. [CrossRef]
3. Salber, D.; Dey, A.K.; Abowd, G.D. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh, PE, USA, 15–20 May 1999; pp. 434–441. [CrossRef]
4. Michalakis, K.; Christodoulou, Y.; Caridakis, G.; Voutos, Y.; Mylonas, P. A Context-Aware Middleware for Context Modeling and Reasoning: A Case-Study in Smart Cultural Spaces. *Appl. Sci.* **2021**, *11*, 5770. [CrossRef]
5. Dantas, F.; Batista, T.; Cacho, N. Towards Aspect-Oriented Programming for Context-Aware Systems: A Comparative Study. In Proceedings of the First International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments (SEPCASE '07), IEEE Computer Society, Minneapolis, MN, USA, 20–26 May 2007. [CrossRef]
6. Flores, A.; Augusto, J.C.; Polo, M.; Varea, M. Towards context-aware testing for semantic interoperability on PvC environments. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, The Netherlands, 10–13 October 2004; Volume 2, pp. 1136–1141. [CrossRef]
7. Alegre, U.; Augusto, J.C.; Clark, T. Engineering context-aware systems and applications: A survey. *J. Syst. Softw.* **2016**, *117*, 55–83. [CrossRef]
8. Chen, G.; Kotz, D. *A Survey of Context-Aware Mobile Computing Research*; Dartmouth College: Hanover, NH, USA, 2000. [CrossRef]
9. Dey, A.K. Understanding and Using Context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7. [CrossRef]
10. Strang, T.; Linnhoff-Popien, C.; Frank, K. CoOL: A Context Ontology Language to Enable Contextual Interoperability. In *Distributed Applications and Interoperable Systems*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 236–247. [CrossRef]
11. Rodis, P. On defining and modeling context-awareness. *Int. J. Pervasive Comput. Commun.* **2018**, *14*, 111–123. [CrossRef]
12. Baldauf, M.; Dustdar, S.; Rosenberg, F. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.* **2007**, *2*, 263–277. [CrossRef]
13. Bardram, J.E. The Java Context Awareness Framework (JCAF)—A Service Infrastructure and Programming Framework for Context-Aware Applications. In Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), Munich, Germany, 8–13 May 2005; pp. 98–115. [CrossRef]
14. Henricksen, K.; Indulska, J. Developing context-aware pervasive computing applications: Models and approach. *Pervasive Mob. Comput.* **2006**, *2*, 37–64. [CrossRef]
15. McCarthy, J. Notes on Formalizing Context. In Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, 28 August–3 September 1993; Volume 1, pp. 555–560.

16. Hsieh, F.-S. A Dynamic Context-Aware Workflow Management Scheme for Cyber-Physical Systems Based on Multi-Agent System Architecture. *Appl. Sci.* **2021**, *11*, 2030. [CrossRef]
17. Hoyos, J.R.; García-Molina, J.; Botía, J.A. MLContext: A Context-Modeling Language for Context-Aware Systems. *ECEASST* **2010**, *28*, 1–14. [CrossRef]
18. Hoyos, J.R.; García-Molina, J.; Botía, J.A. A domain-specific language for context modeling in context-aware systems. *J. Syst. Softw.* **2013**, *86*, 2890–2905. [CrossRef]
19. Korpipaa, P.; Mäntyjärvi, J.; Kela, J.; Keranen, H.; Malm, E.-J. Managing context information in mobile devices. *IEEE Pervasive Comput.* **2003**, *2*, 42–51. [CrossRef]
20. Baumgartner, N.; Retschitzegger, W. A survey of upper ontologies for situation awareness. In Proceedings of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering, St. Thomas, USVI, USA, 29 November–1 December 2006; pp. 1–9.
21. Verenikina, I. Cultural-historical psychology and activity theory in everyday practice. *Inf. Syst. Act. Theory* **2001**, *2*, 23–38.
22. Gedera, D.P.; Williams, P.J. *Activity Theory in Education: Research and Practice*; Springer: Berlin/Heidelberg, Germany, 2015.
23. Filippetto, A.; Barbosa, J.; Francisco, R.; Klein, A. A project management model based on an activity theory ontology. In Proceedings of the 2016 XLII Latin American Computing Conference (CLEI), Valparaiso, Chile, 10–14 October 2016; pp. 1–11. [CrossRef]
24. Kuutti, K. Activity theory as a potential framework for human-computer interaction research. In *Context and Consciousness: Activity Theory and Human-Computer Interaction*; Nardi, B.A., Ed.; Massachusetts Institute of Technology: Cambridge, MA, USA, 1996; pp. 17–44.
25. Mursu, Á.; Luukkonen, I.; Toivanen, M.; Korpela, M. Activity Theory in Information Systems Research and Practice: Theoretical Underpinnings for an Information Systems Development Model. *Inf. Res. Int. Electron. J.* **2007**, *12*, 29. Available online: https://eric.ed.gov/?id=EJ1104804 (accessed on 20 August 2020).
26. Wickramasinghe, N.; Luber, S.; Durst, C.; Wiser, F. Development of an activity theory-based framework for the analysis and design of socio-technical systems. *Int. J. Netw. Virtual Organ.* **2020**, *23*, 261. [CrossRef]
27. Crawford, K.; Hasan, H. Demonstrations of the Activity Theory Framework for Research in Information Systems. *Australas. J. Inf. Syst.* **2006**, *13*. [CrossRef]
28. Häkkinen, H.; Korpela, M. A participatory assessment of IS integration needs in maternity clinics using activity theory. *Int. J. Med Inform.* **2007**, *76*, 843–849. [CrossRef]
29. Liaw, S.-S.; Huang, H.-M.; Chen, G.-D. An activity-theoretical approach to investigate learners' factors toward e-learning systems. *Comput. Hum. Behav.* **2007**, *23*, 1906–1920. [CrossRef]
30. Vygotsky, L.S. *Mind in Society: The Development of Higher Psychological Processes*; Harvard University Press: Cambridge, MA, USA, 1980.
31. Kaptelinin, V.; Nardi, B.A. *Acting with Technology: Activity Theory and Interaction Design*; MIT Press: Cambridge, MA, USA, 2006.
32. Kaptelinin, V.; Nardi, B.A.; Macaulay, C. Methods & Tools: The Activity Checklist: A Tool for Representing the 'Space' of Context. *Interactions* **1999**, *6*, 27–39. [CrossRef]
33. Engeström, Y. *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*; Orienta-Konsultit Oy: Helsinki, Finland, 1987.
34. Schatzki, T.R. *Social Practices: A Wittgensteinian Approach to Human Activity and the Social*; Cambridge University Press: Cambridge, UK, 1996.
35. Clemmensen, T.; Kaptelinin, V.; Nardi, B. Making HCI theory work: An analysis of the use of activity theory in HCI research. *Behav. Inf. Technol.* **2016**, *35*, 608–627. [CrossRef]
36. White, L.; Burger, K.; Yearworth, M. Understanding behaviour in problem structuring methods interventions with activity theory. *Eur. J. Oper. Res.* **2016**, *249*, 983–1004. [CrossRef]
37. Sowell, T. *Marxism: Philosophy and Economics*; William Morrow & Co.: New York, NY, USA, 1985.
38. Karanasios, S. Toward a unified view of technology and activity: The contribution of activity theory to information systems research. *Inf. Technol. People* **2018**, *31*, 134–155. [CrossRef]
39. Karanasios, S.; Allen, D. Mobile technology in mobile work: Contradictions and congruencies in activity systems. *Eur. J. Inf. Syst.* **2014**, *23*, 529–542. [CrossRef]
40. Silva, A. Requirements, Domain and Specifications: A Viewpoint-Based Approach to Requirements Engineering. In Proceedings of the 24th International Conference on Software Engineering, ICSE 2002, Orlando, FL, USA, 25–25 May 2002; pp. 94–104. Available online: https://ieeexplore.ieee.org/document/1007959 (accessed on 5 October 2020).
41. Jarke, M.; Loucopoulos, P.; Lyytinen, K.; Mylopoulos, J.; Robinson, W. The brave new world of design requirements. *Inf. Syst.* **2011**, *36*, 992–1008. [CrossRef]
42. Ralph, P. The Sensemaking-Coevolution-Implementation Theory of software design. *Sci. Comput. Program.* **2015**, *101*, 21–41. [CrossRef]
43. Hassanabadi, A. Viacom v. YouTube—All Eyes Blind: The Limits of the DMCA in a Web 2.0 World. *Berkeley Technol. Law J.* **2011**, *26*, 405–440. [CrossRef]

44. Benatallah, B.; Sheng, Q.Z. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In Proceedings of the International Conference on Mobile Business, Sydney, NSW, Australia, 11–13 July 2005; pp. 206–212. [CrossRef]
45. Ayed, D.; Delanote, D.; Berbers, Y. MDD Approach for the Development of Context-Aware Applications. In *Modeling and Using Context*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 15–28. [CrossRef]
46. Simons, C.; Wirtz, G. Modeling context in mobile distributed systems with the UML. *J. Vis. Lang. Comput.* **2007**, *18*, 420–439. [CrossRef]
47. Benselim, M.S.; Seridi-Bouchelaghem, H. Extended UML for the Development of Context-Aware Applications. In *Networked Digital Technologies*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 33–43. [CrossRef]
48. Hsu, I.-C. Extending UML to model Web 2.0-based context-aware applications. *Softw. Pract. Exp.* **2012**, *42*, 1211–1227. [CrossRef]
49. Almutairi, S.; Bella, G.; Abu-Samaha, A. Specifying security requirements of context aware system using UML. In Proceedings of the Seventh International Conference on Digital Information Management (ICDIM 2012), Macau, Macao, 22–24 August 2012; pp. 259–265. [CrossRef]
50. Schefer-Wenzl, S.; Strembeck, M. Modelling Context-Aware RBAC Models for Mobile Business Processes. *Int. J. Wirel. Mob. Comput.* **2013**, *6*, 448–462. [CrossRef]
51. Boudjemline, H.; Touahria, M.; Boubetra, A.; Kaabeche, H. Heavyweight extension to the UML class diagram metamodel for modeling context aware systems in ubiquitous computing. *Int. J. Pervasive Comput. Commun.* **2017**, *13*, 238–251. [CrossRef]
52. Kapitsaki, G.M.; Prezerakos, G.N.; Tselikas, N.D.; Venieris, I.S. Context-Aware Service Engineering: A Survey. *J. Syst. Softw.* **2009**, *82*, 1285–1297. [CrossRef]
53. Likert, R. A technique for the measurement of attitudes. *Arch. Psychol.* **1932**, *22*, 55.
54. Plackett, R.L. Karl Pearson and the Chi-Squared Test. *Int. Stat. Rev./Rev. Int. Stat.* **1983**, *51*, 59–72. [CrossRef]
55. Greenwood, P.E.; Nikulin, M.S. *A Guide to Chi-Squared Testing*; Wiley: Hoboken, NJ, USA, 1996.