

Article

A Decentralized Framework for Managing Task Allocation in Distributed Software Engineering

Chetna Gupta ^{1,*} and Varun Gupta ² 

¹ Department of Computer Science & Engineering and Information Technology, Jaypee Institute of Information Technology, Noida 201305, India

² Department of Economics and Business Administration, University of Alcalá, Plaza de la Victoria 2, Alcalá de Henares (Madrid), 28802 Madrid, Spain; varun.iit13@gmail.com

* Correspondence: chetnagupta04@gmail.com

Abstract: In distributed software development, planning and managing fair and transparent task allocation is both critical and challenging. The objective of this paper is to propose a decentralized blockchain-oriented, transparent task allocation framework to improve the quality of the task allocation process. It addresses the concerns of (i) enhancing collaboration, (ii) inhibiting knowledge vaporization, and (iii) reducing documentation problems. The proposed method is a novel two-fold process: First, it identifies and categorizes tasks exhibiting different dependencies and complexities to create equal task clusters based on their dependency type, difficulty, cost, and time. Second, it uses a blockchain-oriented framework to broadcast, check bid validity, allow developers to bid on tasks matching their roles and expertise, evaluate, and announce the winner for task allocation using smart contracts. Results of experimentation, surveys, and interviews with software practitioners conclude that the proposed solution is transparent and effective in allocating tasks (with Cranach's alpha of 0.894) at a low cost of contract execution in a distributed software development environment. Overall, the proposed approach will have a positive and significant impact in industrial settings.

Keywords: blockchain technology; cloud computing; distributed software development; innovation capabilities; knowledge management; process innovation; task allocation



Citation: Gupta, C.; Gupta, V. A Decentralized Framework for Managing Task Allocation in Distributed Software Engineering. *Appl. Sci.* **2021**, *11*, 10633. <https://doi.org/10.3390/app112210633>

Academic Editor: Vito Conforti

Received: 22 October 2021

Accepted: 10 November 2021

Published: 11 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Developing successful projects where teams are geographically distributed puts pressure on project managers and teams to collaborate and coordinate effectively. This makes distributed software development (DSD) adoption both challenging and complex. Despite numerous advantages in overcoming barriers of geographically distributed teams, such as different time zones, reduction in development cost, time of development, etc., some of the barriers arise in planning and managing task allocation, coordination, communication, handling of strategic issues, knowledge management, interpersonal relationships, technical issues, team roles and responsibilities, efficiency, and trust among team members [1–6]. Both academia and industry are exploring solutions to these challenges of distributed software development. A significant challenge in DSD is related to handling task allocation—what, when, to whom, and how—to remote teams. Other major issues are related to problems of collaboration and coordination between distributed team members. In DSD, a lack of direct communication leads to non-trustworthy behavior and knowledge vaporization. Studies have acknowledged that many organizations face the challenge of a lack of proper documentation in both agile as well as traditional software development [7–11]. According to them, much of the knowledge remains in the source code test files, and many times outdated, incomplete, missing, or abstract documents are shared [7–9,12]. Another issue that is dominant in DSD is the knowledge vaporization. Due to the lack of face-to-face communication, much of the knowledge exists in the form

of electronic media such emails and chats, which makes retrieval of this knowledge difficult [7,10,11]. This is due to the fact that teams need to keep themselves updated with emerging software development techniques or strategies.

This paper proposes a solution to one of the challenges of allocating tasks in DSD with improved coordination and communication between team members. These two aspects are closely related to each other because efficient task allocation and appropriate communication and coordination confirm smooth development, exclusive of the need for re-allocation, and by ensuring tasks are allocated to the most suitable developers/sites, cogitating the development cost, proficiency, availability, and other geographical challenges. In the past, these two aspects were handled separately. The handling of task allocation among team members or offshore organizations requires effective and efficient collaboration and coordination measures. Deciding what, when, to whom, and how, is one of the biggest challenges that managers face while allocating tasks of geographically separated teams. In general, the task allocation is handled by project managers based on developers' appraisals, expertise, or history of task handling. This ad-hoc or manual allocation does not obey any systematic approach and makes this process non-transparent to a larger extent. The literature lists methods and frameworks independently to address the complexity of inherent task assignment decisions and knowledge management to support adaptability, flexibility, and the existence of a process or flowcharts based on quantitative multi-criteria methods of decision-making and mathematical aspects [3–6]. Furthermore, the issue of documentation and knowledge vaporization complicates the whole process of knowledge-sharing. In this paper, we argue that both of these aspects are to be considered together for successful software development under the DSD umbrella. The literature presents many studies to handle these challenges but still, there is a need of improvement in a few areas, such as knowledge-sharing combined with task allocation, that is still lacking to some extent in this progressive market-driven and evolving technology-backed industry.

One solution to tackle these issues is the adoption of blockchain technology. More recently, researchers have investigated the use of blockchain technology with cloud services to tackle the significant challenges or drawbacks of cloud services. Blockchain technology is considered the future of the software industry, endeavoring for security and privacy improvements. Blockchain is a distributed ledger that provides central authority with provision to validate and verify the data. It eliminates the data security and privacy concerns [13]. To accelerate cloud computing adoption, by integrating with blockchain technology, organizations can eliminate the concerns of data privacy and security. With this combined technology, data security, service availability, and data/knowledge management in cloud services can be improved.

The presented research is a step towards discovering and adopting a technology triggered by blockchain technology to transform the way task allocation is handled in distributed software development. It is a step towards a new solution leveraging social and economic benefits to the software industry. These benefits could then be transferred back to society in terms of high value and lower cost (due to efficient allocation of tasks to developers with the use of blockchain technology) associated with the software products. Blockchain-centric software development has recently gained researcher focus to define new directions in software development. While it is a promising emerging research area, the research, academia, and industry practitioners are still exploring ways to alter existing models or create new models.

To address the above-mentioned challenges and issues, this paper proposes a novel solution using the benefits of blockchain technology for a transparent task auction and allocation framework—*TaskAuc*. It is a distributed, trustworthy, and secure framework that provides a platform for developers who wish to bid for tasks of their interest. Blockchain-oriented software development is an evolving paradigm that has gained a lot of interest in the last few years. The biggest advantage of using blockchain technology is that it defines new directions to allow effective software development by ensuring productive testing activities, enhancing collaboration within teams, and increasing the use of smart

contracts in software development activities for the various role-based accesses in software repositories. In situations where teams are scattered over large geographic locations, diverse time regions, and possibly conflicting cultures, coordination and agreement on task assignment is more challenging. This makes the whole process of task allocation non-transparent and unjustified to some extent.

This is the first decentralized blockchain-oriented approach to help in minimizing issues of coordination, communication, and lack of team spirit. It is a two-fold process where firstly, various complexities and task dependencies are identified and are then classified into three dependency types exhibiting different dependency relations, creating equal task clusters based on their difficulty, dependency type, cost, and time to support parallel and rapid development. Secondly, a decentralized quantitative framework based on the blockchain model will help in the auction process by broadcasting, bidding process, validity check, and registering choices through bidding for specific tasks matching their roles and expertise. Thereafter, the results of bidding are evaluated, and tasks are allocated to winners using two Ethereum smart contracts. The proposed framework will relieve the difficulty of allocating tasks to developers and will also press on parallel concurrent development by engaging developers (distributed globally) to develop a fully independent module concurrently with several other developers.

The overall impact of acceptance ensures that transparency and minimization of issues of coordination, communication, knowledge vaporization, and document storage are investigated. The main contributions of this research are as follows:

- It uses blockchain technology as a quantitative framework to provide an effective novel solution for task auction and allocation among team members quantitatively and without ambiguity.
- It identifies and categorizes task complexities and dependencies to help project managers create task clusters of equal size based on their complexity and duration to leverage the well-known advantages of parallel workflows to support rapid development.
- It minimizes the challenges and issues faced in DSD, particularly communication, coordination, knowledge sharing, knowledge vaporization, and lack of documentation and team spirit.
- Results (verified and evaluated by implementing it on the Ethereum network) indicate that the proposed method is transparent and effective in allocating tasks at a low cost of contract execution in a distributed environment.

The remainder of the paper is organized as follows: Section 2 discusses the background of distributed software development and Section 3 discusses the proposed approach. Section 4 presents the discussion on its implementation, procedure, and results. Section 5 presents the user study. Finally, the conclusion, limitations, and future work are presented in Section 6.

2. Distributed Software Development

Task allocation is a challenging activity for projects being developed in a distributed scenario. As discussed in the above section, studies in the past have discussed many strategies to bridge challenges of task allocation in DSD. Sutanto et al. [3] conducted a study with 95 projects and analyzed that globally dispersed teams generally require about 2.5 times longer than collocated teams to complete their work due to the lack of task coordination. The literature lists a number of techniques to support task allocation using multi-criteria optimization [4], risk-driven models [5], effort estimation models [6], and the SDLC-based work division technique [14], aiming to minimize overhead costs. In addition to these domains, aiding team building, promoting teamwork, personality composition of team members, how cultural identities influence and foster communication, coordination, and collaboration are also explored [1,2,15]. A large number of these approaches are based on quantitative and mathematical aspects. Bohnerand Mohan [16] clearly state that the human factor is the main ingredient for a successful project. Closely related to the issue of task allocation is the issue of task dependency [17]. A few authors have adopted multi-criteria

approaches for efficient task distribution among team members [18–20]. These approaches use multiple factors in an organization, such as perform context characterization, and rank influencing factors using the verbal decision analysis (VDA) method for evaluation and decision making. Almeida et al. [21] proposed multi-criteria decision analysis for planning and fine-tuning project plans. The model was developed using cognitive mapping and MACBETH (Measuring Attractiveness by a Categorical-Based Evaluation Technique) [22].

In the DSD environment, these dependencies have strong communication as well as coordination requirements. Insufficient interaction among teams causes a lack of team spirit and creates the problem of stakeholder dissatisfaction, leading to rework. To the best of our knowledge, to date, the literature lacks in studies focusing on considering the actual view of the developer, whether (s)he can or is willing to work on a particular project based on their strength, preferences, schedules, and weaknesses. It is generally handled by project leaders/managers, dominated by project managers' appraisal, their biased/unbiased judgment, and the availability of past experiences of team members.

In light of the present work, the work carried out in the past has been categorized into the following four types, listed below. These studies have investigated and tried to solve problems associated with research topics in the context of adopting DSD. Several types of research methods are used in these studies, such as case study, controlled experiment, simulation, and expert panel, using several other research sub-models, such as interviews, observation, questionnaire, analysis of data repository, etc., to produce evaluation research papers, experience papers, solution proposals, opinion papers, philosophical papers, and validation research papers. A summary of these is presented below:

- Issue of collaboration, coordination, and communication: A few researchers have investigated and proposed solutions or strategies to improve and provide solutions for effective collaboration, coordination, and communication required for successful implementation of DSD.
- Issue pertaining to knowledge and resource management: Researchers have proposed new models for knowledge representation and resource management focusing on discussing stochastic models, problems of asynchronous communication, managing effective team process and work, feasibility, project execution, coordination, knowledge-sharing, etc.
- Theoretical basis: The progress in the development of new theories, helping with addressing knowledge gaps and issues of misunderstandings leading to techniques not addressing DSD characteristics, to achieve the desired result.
- Task allocation: One of the most influential factors contributing to the success of DSD, which incorporates all the above-mentioned issues to achieve the desired outcome. It involves allocation of the tasks of a project having dependence between tasks and team members who are geographically distributed.

3. The Proposed *TaskAuc* Framework

The process of task allocation is of substantial importance for the research and industry practitioners. Blockchain is an open and distributed ledger that holds transactional records, ensuring security, transparency, and decentralization. Each block consists of the cryptographic hash of the previous block, a timestamp, and a set of transactions, and a nonce in the block header. All transaction records (current and previous) are stored in the crypto currency. A smart contract, which is a piece of the computer program, is built on top of crypto currencies, allowing users to write and execute contracts on the blockchain.

Smart contract execution needs a certain amount of gas that is to be purchased using Ether. The behavior of a smart contract is determined by the execution flow of the source code. This execution is triggered by a transaction consuming gas that is converted into the amount of Ether, charged as the transaction fee. The outline of the proposed *TaskAuc* approach is sketched in Figure 1. The task auction and allocation is performed in a decentralized manner, where all the communication and coordination is conducted using a blockchain. The project manager (representing the organization) will be responsible

for announcing the tasks through the auction process, checking the validity of bids, bids' evaluation, and winner announcement using smart contracts that do not involve any third-party dependence. The proposed framework models developers as bidders. A bid in this system refers to an offer made by a developer to show his/her willingness to take up the particular task. On the other hand, a bidder is the person (here, developer) who has presented the bid. Project managers or organizations who/which are willing to distribute or allocate tasks are modeled as auctioneers, and developers who are willing to pick tasks of their preference are modeled as bidders. Once the task to be distributed is identified, the project manager broadcasts it with essential information. The developers can then decide and make an informed decision according to their preferences. With the use of the blockchain technology, developers can make the most of their effectiveness by keeping minimum interaction and communication with the project managers.

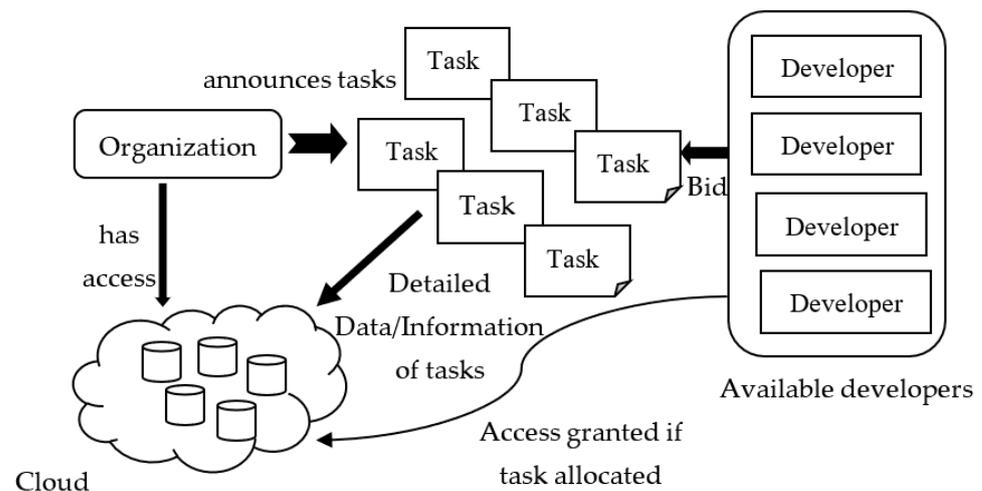


Figure 1. The proposed *TaskAuc* architecture.

3.1. Blockchain Structure

Each block of a blockchain consists of the cryptographic hash of the previous block, a timestamp, and a nonce in the block header. In addition to this, each block of a blockchain contains a brief description of the task and its attributes along with bid content. The attributes of a block are discussed below.

- Duration: Estimated time for completing a specific task which is influenced by factors such as the degree of parallelism in development, training needs, need to develop support infrastructure, etc.
- Task category type: An analysis of relationships (dependency) that is exhibited in the system with other system entities.
- Task: "A task is the smallest unit of work with a well-defined functionality and external interface with other tasks"[23].
- Total number of bids: The total number of interested developers of a particular task.
- Encrypted bidding key: A secret unique key for providing authentication.

The project managers/broadcaster organization authenticity is provided by reputation systems which provide a metric for developers to judge one another and determine the risk of involvement in an exchange with another particular user.

3.2. Task Categorization

Tasks can be complex, domain-specific, or dependent on other tasks. The dependency is represented over time and it indicates that a latter task cannot be processed until its dependencies are fulfilled. The proposed approach captures all possible task dependencies to create clusters of equal sizes and proposes the three categorizations presented below. This will help project managers to create task clusters of equal sizes based on their difficulty,

dependency, cost, and time to leverage the well-known advantages of parallel workflows to support rapid development. The *TaskAuc* implementation has three main phases: auction, bidding, and winner announcement. Registered developers will be allowed to place their valuations as bids for particular tasks. All placed bids are decrypted, analyzed, or evaluated to announce the winner. The winning bids are pushed back to the auction process with the bidder key. We assume that there will be only one auctioneer and multiple bidders. The lowest-cost bids from the range specified in the block structure will be selected and notified. The lowest-cost bid strategy is most optimal for getting the lowest possible cost value results for better optimization of results. In the end, the tasks which are not selected or allocated will be discussed with potential developers until the tasks are allocated to all developers. This will help project managers to create task clusters of equal sizes based on their difficulty, dependency, cost, and time to leverage the well-known advantages of parallel workflows to support rapid development.

- **Definition 1:** Type 1 task dependency—if it is an independent task and cannot be further decomposed into sub-tasks.
- **Definition 2:** Type 2 task dependency—if a task has a possible set of constraints (input, output, any other constraint) that will be accomplished either by decomposing the task into an sub-independent task (similar to type 1 dependency) or into decomposable sub-tasks of type:
 - **Definition 2.1:** Direct task dependency: t_1 cannot be completed until t_2 is completed.
 - **Definition 2.2:** Indirect task dependency: t_2 cannot be completed until t_3 is completed, which imposes a transitive condition for t_1 to complete.
- **Definition 3:** Type 3 task dependency—if a task can be broken down into sub-tasks of type 1 dependency with no constraints and which (1) can be allocated to different teams, i.e., n tasks to n teams, or (2) one or more such sub-tasks can be allocated to the same team, i.e., m tasks to n teams.

4. TaskAuc Procedure and Implementation

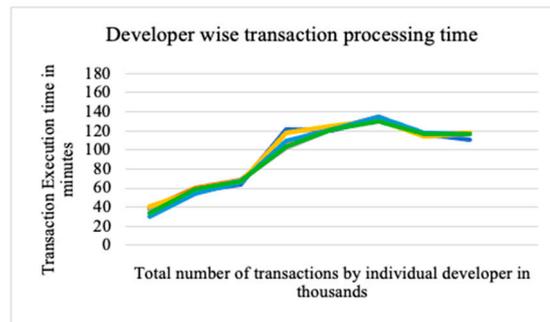
To provide the business logic and centralized authenticity, two Solidity smart contracts are written. The smart contracts are triggered as and when the referring transactions are executed in the code automatically and independently in a prescribed manner on every node of a blockchain network. This will help in maximizing the throughput of the task allocation by allowing individual developers to select specific tasks matching their preferences, expertise, and schedules. The winner will then be provided access to the documentation and information managed over the cloud. The access control permissions and rights to the data are regulated by various policies chosen by the specific software company. The pseudo code is explained in Box 1.

Box 1. Pseudo code.

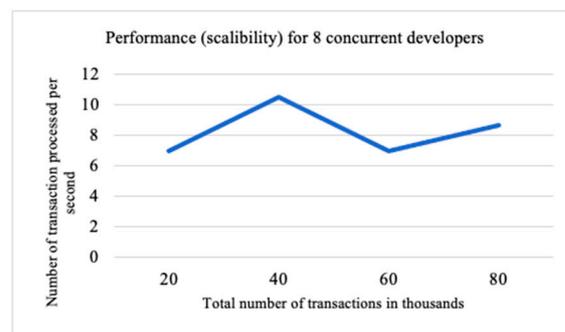
1. For each bidder $b_i \in B$,
2. b_j is the bid submitted by $b_i \in B$, represented as $\text{bid} = \{b_1, b_2, \dots, b_n\}$, indicating the set of all submitted bids.
3. t_x is the task for which the bid is to be placed.
4. For each auction from the auction list:
 - (a) while ($b_{x_i} > 1$) //check if at least one bidder is available for bidding
 - (b) RECEIVE bid b_{x_j} and CHECK and ALLOW single bid;
 - (c) START bidding;
 - (d) COMPUTE best responses bid; //set [0,1] indicating 1 as winner and 0 otherwise
 - (e) PUSH result back on chain
 - (f) ANNOUNCE winner and allocate task;
 - (g) end while.

5. Results and Discussion

Three evaluation parameters—latency time, scalability, and concurrent transaction processing time—are used to test the performance of the *TaskAuc* approach. To address the issue of scalability in the blockchain, the bidding transaction time for up to 8 maximum concurrent developers is recorded. Figure 2a shows the blockchain performance for concurrent bidding transactions and Figure 2b presents the analysis of the transaction processing time for concurrent developers. Additionally, the observation of average latency increase with an increase in the number of transactions was recorded.



(a)



(b)

Figure 2. (a) Scalability performance for concurrent bidding transactions with up to 8 concurrent developers. (b) Transaction processing time for each concurrent developer.

Table 1 presents the cost incurred to set up and execute smart contracts in the Ethereum network with 15 bidders. The cost is in the amount of gas consumed by each operation.

Table 1. Summary of transaction costs.

Operations	Transaction Cost (Gas)	Execution Cost (Gas)	Total Cost (Gas)
Contract deployment	36,162	21,125	57,287
Bidding	24,615	42,802	67,417
Auction End	14,898	4634	19,532
Compute Winner	12,784	1217	14,001

5.1. User Study

5.1.1. Method

The research study was conducted with a total of 64 experts having DSD experience. The participants have diverse experience ranging from less than 5 years (40%), 5 to 10 years (36%), and more than 10 years (24%). The research study participants were located in Europe (36%), Asia (30%), America (14%), the United Kingdom (12%), and Australia (8%).

The participants include 56% males and 44% females. The participants were selected through a series of purposive and snowball sampling (non-probabilistic sampling). The participants in our professional proximity were requested for their participation and their referral for reaching new research study participants. Based on the referrals, the initial list was composed of official communication details of 100 software practitioners. The participants were approached for their voluntary participation by mailing them the questionnaire (refer to Table 2) and the informed consent form.

Table 2. Summary of reliability analysis of survey results.

Questions	Mean	Standard Deviation	Cranach's Alpha
Does the proposed approach provide ease of use?	4.53	1.14	0.894
How satisfied are you with the results of the task assignment?	4.68	0.81	
Would you be willing to use the proposed approach?	4.81	0.88	
Compared to current allocation practices, does the proposed approach make sense?	4.69	1.07	
Is it possible to use the proposed approach in terms of fetching data and processing data?	4.83	1.20	
Can the proposed approach effectively perform task auction and allocation quantitatively and without ambiguity?	4.86	0.91	
Can the proposed approach help in achieving the aim of rapid and parallel concurrent development?	4.82	1.01	

5.1.2. Data Collection

The data about the indicators of latent variables were collected through the online questionnaire (refer to Table 2) through Google forms. Prior to actual data collection, a pilot test was conducted with eight software developers in our professional network. Their feedback resulted in modification in a few portions of the questionnaire, leading to modification and dropping of a few questions. The survey questions were directly mapped to the aim of the presented approach—whether or not the proposed approach is suitable for task allocation. The participants were asked to rate their agreement or disagreement with the indicators of latent variables on a 5-point Likert scale, with 1 representing strong disagreement and 5 representing strong agreement. An additional “do not know” option was added to reduce the noise in the response data. Table 2 presents the survey results. This process helped in attaining in-depth knowledge about the trade-offs in present DSD and the proposed approach. The result shows a positive outcome and that the approach was well-appreciated by the practitioners involved in the study.

5.2. Discussion

Blockchain as a service is a promising solution focusing on ensuring privacy, verifiability, and scalability. Nowadays, many leading organizations are offering BaaS solutions, such as Amazon Managed Blockchain, Azure Blockchain Workbench, HPE Mission Critical Distributed Ledger Technology, HPE Pointnext, IBM Blockchain, Oracle Blockchain Applications Cloud, Samsung Nexledger, SAP Cloud Platform Blockchain, etc. The development challenges such as issues of coordination and communication, knowledge vaporization, and document record-keeping can be reduced. In fact, knowledge-sharing of information and data are much easier and collaboration among team members is enabled. This has motivated us to propose a task allocation approach that integrates blockchain.

The blockchain technology in the distributed task allocation process will provide higher efficiency in terms of cost. As shown in Table 1, the execution cost is very low with the use of blockchain technology. The proposed model allows concurrent bidding. It provides a solution to one of the significant challenges in DSD related to handling task allocation—what, when, to whom, and how—to remote teams. Additionally, it provides

access to developers to choose what they want to work on. This affords a lot of freedom to developers. The survey data collected from across the globe also confirms the usefulness of the proposed solution. Results of Table 2 indicate the positive acceptance of the usefulness of the approach, with 0.894 as the Cronbach's alpha value.

6. Conclusions

This paper presented a novel *TaskAuc* approach for task auction and allocation in a transparent manner using the concept of blockchain technology in distributed software development. The proposed concept addresses the issues of planning and mitigation, including fair task allocation, coordination, and communication among team members and lack of team spirit. The whole process is divided into two phases: In the first phase, task complexities and dependencies are identified and are then classified into three types exhibiting different dependency relations. Next, equal-sized task clusters are formed using task classification information by considering other factors including complexity, cost, and time to leverage the well-known advantages of parallel workflows to support rapid development. This quantitative framework effectively achieves the objective of this paper by ensuring privacy, verifiability, scalability, support for concurrent bidding, evaluation, selection, and winner announcement for task allocation using two smart contracts written in Solidarity. It provides the freedom for developers to choose among alternatives (tasks) best suiting their roles, skills, and expertise. The proposed model also provides authentication to broadcasters and developers by engaging them (distributed geographically) to develop a fully independent module simultaneously with several other developers. Results of the initial experimentation, surveys, and interviews concluded that the proposed method is transparent and effective in achieving rapid development using blockchain technology in a distributed environment quantitatively and without ambiguity.

One of the limitations of the proposed approach is that it does not explicitly consider the factor of labor cost and the current load of developers. We assume that only those developers are bidding who have time to take responsibility. As a part of future work, this work can be extended by incorporating strategies to distribute work equally among developers by considering their workload.

Author Contributions: Conceptualization, C.G.; methodology, C.G.; software, C.G., V.G.; validation, C.G., V.G.; formal analysis, C.G., V.G.; investigation, C.G., V.G.; resources, C.G., V.G.; data curation, C.G., V.G.; writing—original draft preparation, C.G., V.G.; writing—review and editing, C.G., V.G.; visualization, C.G., V.G.; supervision, C.G., V.G.; project administration, C.G., V.G.; funding acquisition, Not applicable. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Nundlall, C.; Nagowah, S.D. Task allocation and coordination in distributed agile software development: A systematic review. *Int. J. Inf. Technol.* **2020**, *13*, 321–330. [[CrossRef](#)]
2. Filho, M.S.; Pinheiro, P.R.; Albuquerque, A.B.; Rodrigues, J.J.P.C. Task Allocation in Distributed Software Development: A Systematic Literature Review. *Complexity* **2018**, *2018*, 6071718. [[CrossRef](#)]
3. Sutanto, J.; Kankanhalli, A.; Tan, B.C.Y. Investigating task coordination in globally dispersed teams: A structural contingency perspective. *ACM Trans. Manage. Inf. Syst.* **2015**, *6*, 1–31. [[CrossRef](#)]
4. Lamersdorf, A.; Munch, J. TAMRI: A tool for supporting task distribution in global software development projects. In Proceedings of the 4th IEEE International Conference on Global Software Engineering (ICGSE 2009), Limerick, Ireland, 13–16 July 2009; pp. 322–327.
5. Lamersdorf, A.; Munch, J.; Fernández-del Viso Torre, A.; Rebate Sanchez, C. A risk-driven model for work allocation in global software development projects. In Proceedings of the 6th IEEE International Conference on Global Software Engineering (ICGSE), Florianopolis, Brazil, 16–19 October 2011; pp. 15–24.
6. Narendra, N.C.; Ponnalagu, K.; Zhou, N.; Gifford, W.M. Towards a Formal Model for Optimal Task-Site Allocation and Effort Estimation in Global Software Development. In Proceedings of the Annual SRII Global Conference (SRII), San Jose, CA, USA, 24–27 July 2012; pp. 470–477. [[CrossRef](#)]

7. Hislop, D.; Bosua, R.; Helms, R. *Knowledge Management in Organizations: A Critical Introduction*; Oxford University Press: Oxford, UK, 2018.
8. Zahedi, M.; Babar, M.A. Knowledge sharing for common understanding of technical specifications through artifactual culture. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, UK, 17–18 May 2014; ACM: New York, NY, USA, 2014; p. 11.
9. Anwar, R.; Rehman, M.; Wang, K.S.; Hashmani, M.A. Systematic Literature Review of Knowledge Sharing Barriers and Facilitators in Global Software Development Organizations Using Concept Maps. *IEEE Access* **2019**, *7*, 24231–24247. [[CrossRef](#)]
10. Borrego, G.; Moran, A.L.; Palacio, R. Preliminary Evaluation of a Tag-Based Knowledge Condensation Tool in Agile and Distributed Teams. In Proceedings of the IEEE 12th International Conference on Global Software Engineering (ICGSE), Buenos Aires, Argentina, 22–23 May 2017; pp. 51–55. [[CrossRef](#)]
11. Borrego, G.; Morán, A.L.; Palacio, R.R.; Vizcaíno, A.; García, F.O. Towards a reduction in architectural knowledge vaporization during agile global software development. *Inf. Softw. Technol.* **2019**, *112*, 68–82. [[CrossRef](#)]
12. Zahedi, M.; Shahin, M.; Babar, M.A. A systematic review of knowledge sharing challenges and practices in global software development. *Int. J. Inf. Manag.* **2016**, *36*, 995–1019. [[CrossRef](#)]
13. Lamba, A.; Singh, S.; Balvinder, S.; Dutta, N.; Rela, S. Mitigating IoT security and privacy challenges using distributed ledger based blockchain (DI-BC) technology. *Int. J. Technol. Res. Eng.* **2017**, *4*, 5687–5692. [[CrossRef](#)]
14. Wickramaarachchi, D.; Lai, R. A method for work distribution in global software development. In Proceedings of the IEEE 3rd International Advance Computing Conference (IACC), Ghaziabad, India, 22–23 February 2013; pp. 1443–1448.
15. Nguyen, H. A new knowledge-based measure for intuitionistic fuzzy sets and its application in multiple attribute group decision making. *Expert Syst. Appl.* **2015**, *42*, 8766–8774. [[CrossRef](#)]
16. Bohner, S.A.; Mohan, S. Model-Based Engineering of Software: Three Productivity Perspectives. In Proceedings of the 33rd Annual IEEE Software Engineering Workshop, Skövde, Sweden, 13–14 October 2009; pp. 35–44. [[CrossRef](#)]
17. Sooraj, P.; Mohapatra, P.K. Modeling the 24 h software development process. *Strat. Outsourcing Int. J.* **2008**, *1*, 122–141. [[CrossRef](#)]
18. Simao Filho, M.; Pinheiro, P.R.; Albuquerque, A.B.; Simao, R.P.; Azevedo, R.S.; Nunes, L.C. A multicriteria approach to support task allocation in projects of distributed software development. *Complexity* **2019**, *2019*, 3926798. [[CrossRef](#)]
19. Simao Filho, M.; Pinheiro, P.R.; Albuquerque, A.B. Task allocation in distributed software development aided by verbal decision analysis. In *Computer Science On-Line Conference*; Springer: Cham, Switzerland, 2016; pp. 127–137.
20. Simao Filho, M.; Pinheiro, P.R.; Albuquerque, A.B. Task assignment to distributed teams based on a qualitative multicriteria approach. In Proceedings of the 12th Iberian Conference on Information Systems and Technologies (CISTI), Lisbon, Portugal, 14–17 June 2017; pp. 1–6.
21. Almeida, L.H.; Albuquerque, A.B.; Pinheiro, P.R. A multi-criteria model for planning and fine-tuning distributed scrum projects. In Proceedings of the 6th IEEE International Conference on Global Software Engineering (ICGSE), Florianopolis, Brazil, 16–19 October 2011.
22. Bana e Costa, C.A.; Sanchez-Lopez, R.; Vansnick, J.C.; De Corte, J.M. Introducción a MACBETH. In *Análisis Multicriterio para la Toma de Decisiones: Métodos y Aplicaciones*; Leyva López, J.C., Ed.; Plaza y Valdés: Mexico City, Mexico, 2011; pp. 233–241.
23. Jalote, P.; Jain, G. Assigning tasks in a 24 h software development model. *J. Syst. Softw.* **2006**, *79*, 904–911. [[CrossRef](#)]