Hindawi
Mobile Information Systems
Volume 2022, Article ID 5401103, 8 pages
https://doi.org/10.1155/2022/5401103

Research Article

Collaborative Software Engineering Model Dependent on Deep Recursive Least Squares

Jiao Chen and Zhi Gong

College of Computer Science and Engineering, Hunan University of Information Technology, Changsha 410005, Hunan, China

Correspondence should be addressed to Zhi Gong; 15112403011@stumail.sdut.edu.cn

Received 2 November 2021; Accepted 3 January 2022; Published 18 February 2022

Academic Editor: Hye-jin Kim

With the continuous development and maturity of information technology, the open methods and development results of software are enriched constantly. There are uncertain factors that may affect the cost of software development. The control of demand, the determination of the logical framework, and the vagueness and inaccuracy of business logic can all affect the development cycle of software. In this paper, the bit error rate and complexity are calculated by using the related algorithm based on the deep recursive least squares algorithm through the establishment of a collaborative engineering tool evaluation system. Under the same preconditions, the estimation performance of the deep recursive least squares algorithm is compared with that of the ideal channel based on the same foundation, and the deep recursive least squares algorithm is simulated through the simulation curve. In addition, the performance of the algorithm is analyzed. The evaluation criteria of four indexes for perception, synchronization, product, and coordination were put forward. The results of the study indicate that the proposed algorithm is effective and can support the engineering modeling of collaborative software.

1. Introduction

From the stand-alone software that supports a single system to the network software that can support cross-platform systems and from the software that supports PC to the application microprogram that can support the mobile terminal, the continuous maturity and innovation of software have been affected by big data technology, basic underlying framework technology, and smart terminal hardware. But more importantly, there are uncertain factors that may affect the cost of software development. The control of demand, the determination of the logical framework, and the vagueness and inaccuracy of business logic can all affect the development cycle of software [1–4].

For this purpose, experts in the industry have also developed a variety of approaches and means to make the corresponding improvement, such as the application of local area networks or fixed Internet and the use of developer interoperability for the collaborative development of software to support the unification and standardization of software engineering development methods, specifications, and tools with high efficiency. In addition, the changes in the business logic during the development process of software can be clearly covered to carry out planning and execution of the role and activities in time and space effectively in accordance with the actual requirement for changes. In this paper, the collaborative development of software is implemented based on the method of the deep recursive least squares through the establishment of an evaluation system for the collaborative engineering tools.

1.1. Process Model of the Collaborative Design

1.1.1. Basic Definition of Collaborative Design. The so-called collaborative design indicates that in the whole design process, as there are many nodes and parameters, it is necessary to take the control-driven support, data-driven support, and other factors into comprehensive consideration [5–9]. At the same time, since the collaborative design has a certain level of complexity, it is necessary to

understand the complexity clearly in the process of collaborative design of software and distinguish the complexity according to business logic appropriately. Based on the features of the continuous evolution and improvement in the collaborative design process, specific status control is carried out on the tasks to obtain the real-time status.

Based on the practical requirements, practical operations, and other business needs, the following model based on the features of collaborative design is established in this paper.

Definition 1. The network model for the collaborative design process is a labeled network with the relevant state, that is, a combination of eight items as follows.

CoNet = (P, V, T, F, R, Wr, C, θ, L), which meets the following conditions:

(1) P is a limited set of places.

(2) V is a limited dataset. The parameter information used hereunder is defined in the variable dataset.

(3) T is a finite set of changes.

(4) F is a limited flow relationship, FcPxTuTxP.

(5) R stands for the read relationship, and Wr stands for the write relationship. They represent the file read and write of the tasks in the workflow model.

(6) C : P⟶S(D), and $\vartheta(D)$ stands for the set of powers of color D. Among them, the color set D stands for the set of all possible token types in the system, and different colors represent various token types. For p ∈ P, C(p) stands for the set of all possible token colors in place p.

(7) θ stands for the type function of transition, θ: T {form, tool, subflow}. There are three types of changes in the process model proposed, which have specific meaning in the workflow model: the form type task node, the tool type task node, and the subprocess node.

(8) L is the mapping between the place and the transition to the namespace, and L : P∪T)⟶E^*, S^* stands for the set of character strings.

The network is set as the thirteen yuan ancestor. The main influencing elements of the model are divided into subprocesses, tools, and form collections. Among them, analysis tools need to be bound to the related tools to a certain level. In addition, simulation and other tools should be included as well. The form type is the most basic type. The original input data are sent to the subsequent node data through input and output. The subprocesses represent the subnetwork, and the model of the subnetwork is the same as that in the definition of the parent node.

In the model, there is a start place "start," and a terminal place "terminal"; that is, •start = ∅, teminal• = ∅. Based on the definition of the collaborative design model described above, the formal definition of the collaborative design workflow system is introduced as follows.

Definition 2. The network ψ after the collaborative design is set to a thirteen tuple and defined as $\psi = (P, V, T, F, R, Wr, C, \theta, L, K, M, W, \text{status})$, which meets the following conditions:

(1) CoNet = $(P, V, T, F, R, Wr, C, \theta, L)$ is the basic model of the system.

(2) K: P⟶{1, 2,...} is the ctoken capacity function of the workflow model, and the default dtoken capacity is unlimited.

(3) $M{:}p \longrightarrow \varphi(D)$, where $\varphi(D)$ stands for the power set of the multiple set of color D, which is referred to as the flow identifier of the model. In addition, for $\forall p \in P$: $M_c(p) \le K(p)$. $M_c(p)$ stands for the number of ctokens in place p; $M_c(p)$ stands for the number of ctokens in place p.

(4) $W{:}\ F \longrightarrow \varphi(D)$, where $\varphi(D)$ stands for the power set of the multiple set of color D, which indicates that for $f \in F$, W(f) is the multiple token set of the arc. $M_c(f)$ stands for the number of ctokens on the color multiplicity set of W(f); $M_d(f)$ stands for the color multi-sets of dtoken on the color multiplicity set of W(f).

(5) Status: T⟶{0, 1, 2} is referred to as the status function of system changes. The system can be divided into three states: not-ready state, execution state, and submit state.

1.2. Definition and Evaluation of States. According to the definition of international workflow management (WFM), workflow management system should be driven by the formal description of workflow. It is the same as system software engineering in supporting collaborative software development. WFM controls and manages the activity sequence of users through a series of predefined operations or steps in business activities. The basic control modes include sequence, parallel bifurcation, multi-channel selection, synchronization, exclusive selection, and simple combination. Once the workflow is defined in detail in the process oriented management environment, it is difficult to avoid the constraints of collaborative work steps by patterned sequence. In other words, WFM cannot well support the collaborative mode with high dynamics and high flexibility.

1.3. Advantages and Applicable Scenarios of Collaborative Design. Collaborative software engineering is based on computer network and the interoperability and collaborative work of team developers. It includes all software engineering methods and specifications and tools that support the flexible and efficient work of the team, covering all formal and informal communication and coordination requirements in the process of software development, so as to plan, execute, and coordinate all activities and tasks distributed in space and time. Collaborative software engineering includes a series of tasks from requirement analysis to code debugging. Researchers

are now beginning to develop CSE prototype tools for every possible task. In this research field, some new architectures, tools, and viewpoints on CSE have been published and put forward. The representative tools are tools that support real-time software modeling, design, and management. However, development tools are typically based on traditional software engineering tools and technologies. For example, developers can modify the source code, and users can give early warning of problems. Only a few real-time editing and drawing tools exist, and the modification, copying, and merging of models with traditional methods are replaced by fully synchronous file sharing and multi-view support.

1.4. Basic Operations of Collaborative Design. On the basis of the fundamental definitions, this section is mainly about setting the basic operations in the collaborative design model.

(1) $\cdot t = \{x | \forall x \in P, (x,t) \in F\}$, which stand for all the predecessor places in the transition t.

(2) $\overline{\cdot t} = \{(x,t) | (x,t) \in F\}$, which stand for all the precursor flow relationships of the node t.

(3) $t\cdot = \{x | \forall x \in P, (t,x) \in F\}$, which stand for all the subsequent places in the transition t.

(4) $\overline{t\cdot} = \{(t,x) | (t,x) \in F\}$, which stand for all the subsequent flow relationships of the node t.

(5) $\cdot p = \{x | \forall x \in T, (x,p) \in F\}$, which stand for all the precursor changes of the place p.

(6) $p\cdot = \{x | \forall x \in T, (p,x) \in F\}$, which stand for all the subsequent changes of the place p.

(7) t stands for the node set of the subprocess represented by this node, $t \subseteq P \cup T$.

(8) $r(x) = \{t | (x,t) \in R\}$ is referred to as the read set of x.

(9) $w(x) = \{t | (x,t) \in Wr\}$ is referred to as the write set of x.

(10) $tp(x)$ stands for the type of variable x, $x \in V$.

(11) Type $= \bigcup_{x \in V} tp(x)$ is referred to as the type set of the workflow model system.

1.5. Triggering Rules

1.5.1. General Process. In this part, the triggering rules of $\theta(t) = \text{form}$ or $\theta(t) = \text{tool}$ are discussed: the conditions for general process transition t to have the right of concession at M are as follows:

$$\begin{aligned} &\exists p \in \cdot t: M_c(p) \geq W_c(p,t) \wedge T(t) = 0 \\ &\text{if } (\theta(t) = \text{form} \vee \theta(t) = \text{tool}). \end{aligned} \tag{1}$$

At this point, the status of the system is shown in Figure 1.

When Mt>, t transition is changed from the not-ready state to the execution state, and the change in the system is expressed as follows:

$$\begin{aligned} M_c'(p) &= \left\{ \begin{array}{c} M_c(p) - W_c(p,t) \text{If } p \in \cdot t \\ M_c(p) \text{If } p \notin \cdot t \end{array} \right\}, \\ T(t) &= 1, \\ &\text{if } (\theta(t) = \text{form} \vee \theta(t) = \text{tool}). \end{aligned} \tag{2}$$

Formula (2) clarifies the change of the system transition state, from the unready state to the execution state.

When the transition is in the execution state, the transition is waiting for the arrival of the data required. When all the required data are in the predecessor place of the transition, the transition will enter the submit state. The conditions for the conversion of the transition t from the execution state to the submit state at M are shown as follows:

$$\begin{aligned} T(t) &= 1 \wedge \left(\bigcup_{f \in \cdot t} W_d(f) \subseteq \bigcup_{p \in \cdot t} M_d(p) \right), \\ &\text{if } (\theta(t) = \text{form} \vee \theta(t) = \text{tool}), \\ \bigcup_{f \in \cdot t} W_d(f) &= \{a,b\} \bigcup_{p \in \cdot t} M_d(p) = \{a,b,c\}. \end{aligned} \tag{3}$$

Hence, the conditions for conversion to submission are met, and the state of t is converted to the submit state, that is, $T(t) = 2$, as shown in Figure 2.

The transition t is in the submit state, which returns to the not-ready state again after the calculation task is accomplished. The changes in the system are described as follows:

$$\begin{aligned} M_c'(p) &= \left\{ \begin{array}{l} M_c(p) + W_c(p,t) \text{If } p \in \cdot t, \\ M_c(p) \text{If } p \notin \cdot t, \end{array} \right. \\ M_d'(p) &= \left\{ \begin{array}{l} M_d(p) \cup W_d(p,t) \text{If } p \in \cdot t, \\ M_d(p) \text{If } p \notin \cdot t, \end{array} \right. \\ T(t) &= 1. \end{aligned} \tag{4}$$

The status of the system after conversion is shown in Figure 3.

The C3 library in Figure 3 contains a control token and a b-color data token, and the C4 library contains a ctoken, a c-color token, and two d-color dtokens. At this time, the C3 position contains a control token and a b-color data token, and the C4 place contains a ctoken, a c-color token, and two d-color dtokens.

1.5.2. Subprocess. A system structure is set, as shown in Figure 4.

The conditions for the process t to have the right of concession in M are as follows:

$$\begin{aligned} &\exists p \in \cdot t \wedge p \notin t\cdot: M_c(p) \geq W_c(p,t) \wedge T(t) = 0, \\ &\text{if } (\theta(t) = \text{subflow}). \end{aligned} \tag{5}$$

The status of the system is shown in Figure 5.

There is a predecessor place C1 of t that does not fall into t, and the control place contained in C1 is larger than the flow relationship from C1 to t, which needs to consume a control token. In addition, t is in a not-ready state, that is, $T(t) = 0$. At this point, it is referred to as the subprocess t with the right of concession at M, and it is denoted as $M[t>$.

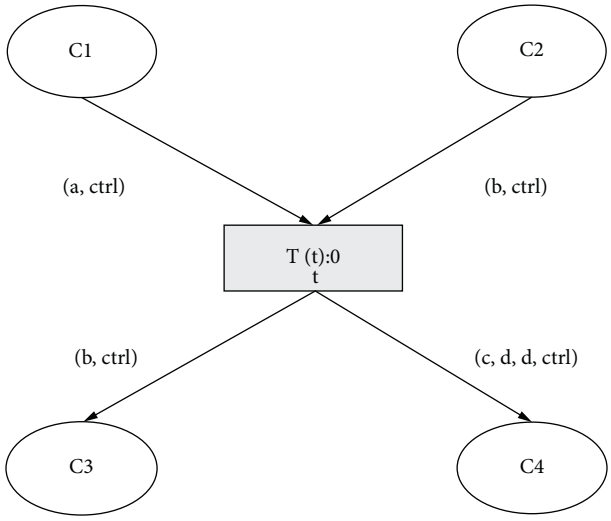

Figure 1: Status of the system at Mt.

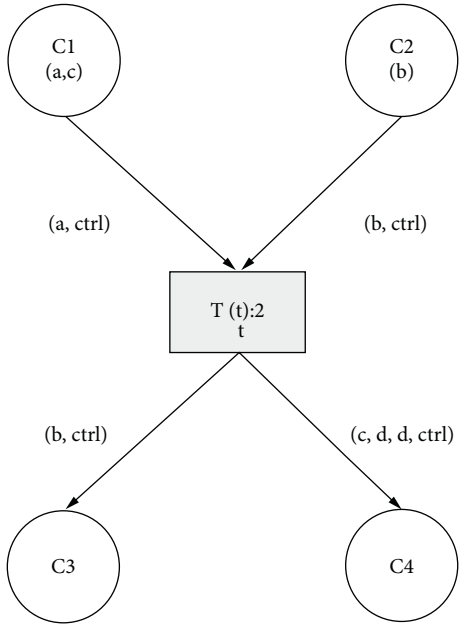

Figure 2: Transition of the system to the submit state.

When $M[t>$, t transitions from the not-ready state to the execution state, and the change in the system is expressed as follows:

$$M_c'(p) = \left\{ M_c(p) - W_c(p,t)\text{If } p \in {}\cdot t \wedge p \notin t\cdot M_c(p) + W_c(t,p)\text{If } p \in t\cdot \cap \dot{t} M_c(p)\text{If } p \notin {}\cdot t \text{ If } p \notin {}\cdot t \cup \left(t\cdot \cap \dot{t}\right)\right\},$$
$$T(t) = 1, \quad \text{If } (\theta(t) = \text{subflow}). \tag{6}$$

When t is in the execution state, upon the arrival of all the data tokens of the precursor place, the data tokens will be transmitted to the start place of the subprocess. The formal representation is shown as follows:

$$M_d'(p) = \left\{ M_d(p) \cup W_d(p,t) \text{If } p \in t\cdot \cap\, t\cdot M_d(p) \text{If } p \notin t\cdot \cap\, \dot{t} \right\},$$
$$\text{If } (\theta(t) = \text{subflow} \wedge T(t) = 1). \tag{7}$$

At this time, a data token "a" and a data token "b" arrive at C1. The change in the system is shown in Figure 6.

When all the data tokens required for the subprocess are already in their predecessor place, they can be converted to the submit state. The conditions for the conversion of t from the execution state to the submit state are as follows:

$$T(1) = 1 \wedge \left(\underset{f \in \cdot t}{\cup} W_d(p) \subseteq \underset{p \in \cdot t}{\cup} M_d(p) \right) \wedge (M_c(p_t) \geq W_c(p_t, t)),$$
$$\text{If } (\theta(t) = \text{subflow}). \tag{8}$$

When t is converted from the execution state to the submit state, if t has completed the processing of the data, it can be further converted from the submit state to the not-ready state and send the control token back to trigger the subsequent nodes.

t is converted from the submit state to the not-ready state, and the change in the system is as follows:

$$M_c'(p) = \left\{ M_c(p) - W_c(p,t) \text{If } p \in \cdot t \wedge p \in t\cdot M_c(p) + W_c(t,p) \text{If } p \in t\cdot \wedge p \notin \dot{t}\, M_c(p) \text{in other circumstances} \right\},$$
$$M_d'(p) = \left\{ M_d(p) \cup W_d(p,t) \text{If } p \in \cdot t \wedge p \notin t\cdot M_d(p) \text{in other circumstances} \right\}, \tag{9}$$
$$T(t) = 0.$$

2. Performance Analysis of the Deep Recursive Least Squares Algorithm

2.1. Bit Error Rate of the Algorithm. When the received signal is 0, the output value obtained after the signal is correlated with the local template is as follows:

$$\begin{aligned} Z_0 &= \int_0^{T_s} [p_R(t) + n(t)] m(t) \mathrm{d}t \\ &= \int_0^{T_s} p_R(t) \widehat{p}_R(t) \mathrm{d}t - \int_0^{T_s} p_R(t) \widehat{p}_R(t - \Delta) \mathrm{d}t + x. \end{aligned} \tag{10}$$

In the above equation,

$$x = \int_0^{T_s} \widehat{p}_R(t) n(t) \mathrm{d}t - \int_0^{T_s} \widehat{p}_R(t - \Delta) n(t) \mathrm{d}t. \tag{11}$$

Its mean value is 0, and the variance can be obtained as follows:

$$\sigma_x^2 = N_0 \left[E_{\widehat{p}_R} - R_{\widehat{p}_R}(\Delta) \right]. \tag{12}$$

$E_{\widehat{p}_R} = \int_0^{T_S} \widehat{p}_R^2(t) \mathrm{d}t$ is the energy carried by $\widehat{p}_R(t)$.

$$R_{\widehat{p}_R}(\tau) = \int_0^{T_s} \widehat{p}_R(t) \widehat{p}_R(t - \tau) \mathrm{d}t. \tag{13}$$

The above equation is the autocorrelation function of $\widehat{p}_R(t)$.

When the transmitted bits are independent of each other and have equal probability, the mean bit error rate (BER) of the receiver is as follows:

$$\begin{aligned} E_{BER} &= \frac{1}{2} \Pr\left(\widehat{b}_j = 1 | b_j = 0\right) + \frac{1}{2} \Pr\left(\widehat{b}_j = 0 | b_j = 1\right) \\ &= \frac{1}{2} \Pr\left(\widehat{b}_j = 1 | b_j = 0\right) = \Pr\left(Z_0 < 0\right) \\ &= \frac{1}{2} erfe \sqrt{\frac{\left(R_{p_R \widehat{p}_R}(0) - R_{p_R \widehat{p}_R}(\Delta) \right)^2}{2N_0 \left(E_{\widehat{p}_R} - R_{\widehat{p}_R}(\Delta) \right)}}, \end{aligned} \tag{14}$$

where $R_{p_R \widehat{p}_R}(\tau) = \int_0^{T_S} p_R(t) \widehat{p}_R(t - \tau)$ stands for the cross-correlation function of $p_R(t)$ and $\widehat{p}_R(t)$.

2.2. Complexity of the FD-RLS Algorithm. The channel estimation is conducted by using the N bit information of the deep recursive least squares [10–13], and the number of sampling points of the signals is set to M. At the same time, the statistical mean, RLS, and DFT of the signals are estimated. The algorithm is used to carry out iteration. In each part of the iterative process, fast transformation is performed through FFT, and at the same time, the RLS algorithm is used to carry out multiplication and 4M addition operations.

FD-RLS still has exponentiation calculation. However, as the pulse signal in the transmission process in the system is

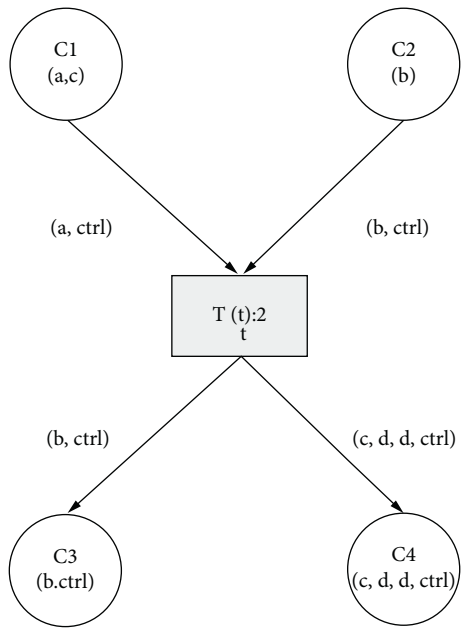

Figure 3: Changes after the system is converted to the submit state.

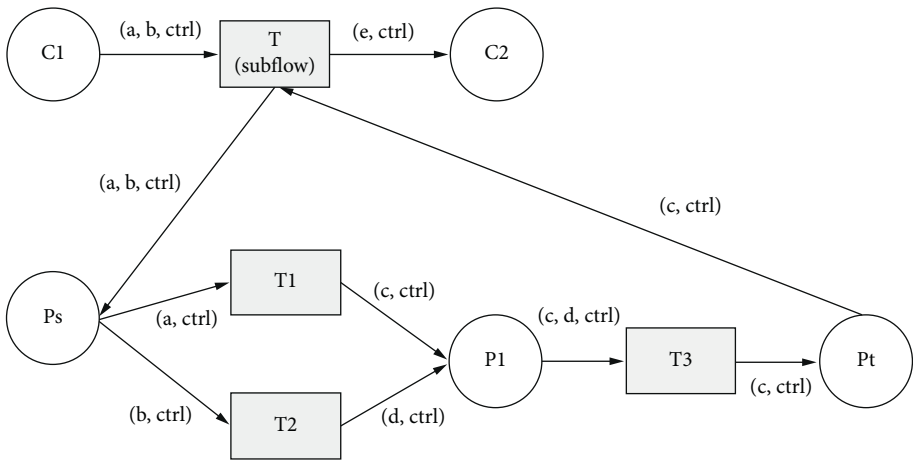

Figure 4: Network system with the subprocesses.

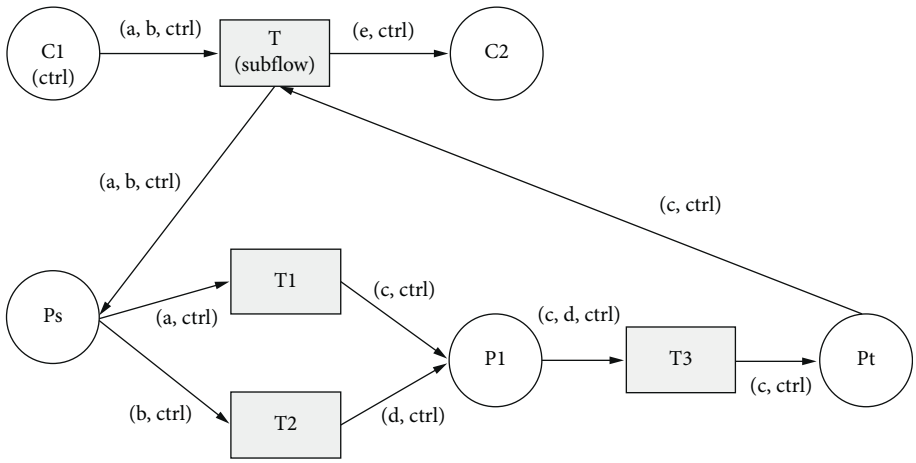

Figure 5: Initial state of the system.

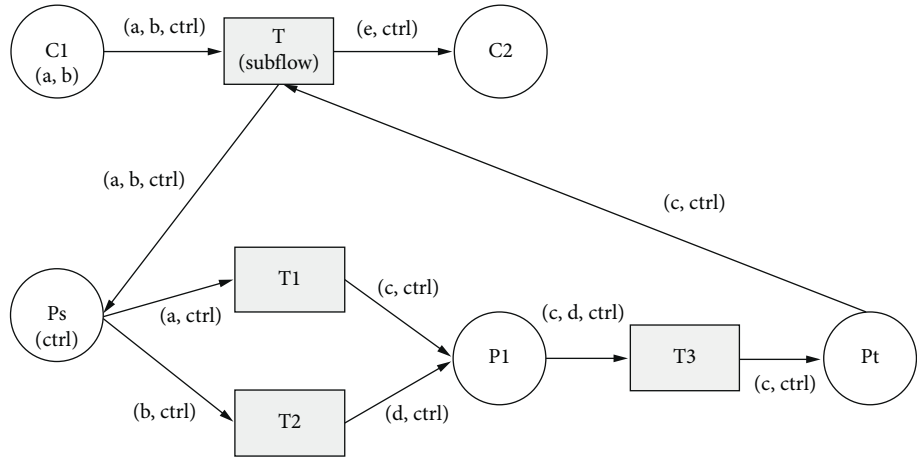

Figure 6: Data generated from the precursor place.

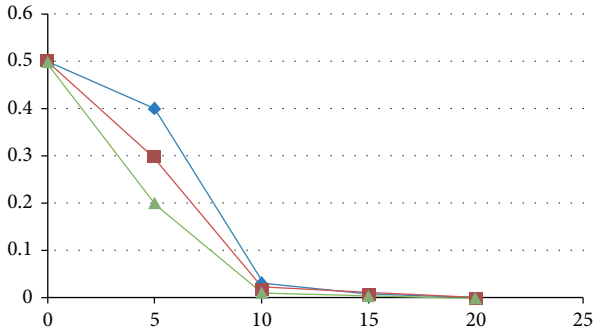

Figure 7: Relationship between the signal-to-noise ratio and the BER under different pulse repetition periods.

relatively weak, it leads to a relatively high sampling frequency for the receiving terminal. At this point, the amount calculated by FD-RLS is less than the calculated amount based on the estimation of the ML channel (that is, the ML algorithm). This suggests that the RLS algorithm has rapid convergence, and the RLS algorithm can achieve the optimization of collaborative design for software by using relatively less recursion, thereby reducing the amount of optimization in the collaborative design of the software.

2.3. Performance Simulation. In this paper, system simulation is carried out on the collaborative design of software, and CM1 is used for the simulation of the channel model to communicate information between service flows [14, 15]. The system parameters are set to 300 (N), 120 ns, and so on, and the duration of the pulse is set to 0.6 ns to ensure timely and rapid response to any requirement. At the same time, it is ensured that the pulse should have sufficient sampling points. In general, f_p = 5GHz is set. For the purpose of making the deep recursive least squares algorithm have the minimum steady-state error, it is necessary to reflect the final algorithm performance index of the collaborative design of software from a number of perspectives.

From Figure 7, it can be seen that the set of experiment 1 has reflected the simulation of the signal-to-noise ratio of the received signal under different collaborative designs of software. The specific results indicate that as the pulse cycle continues to oscillate, the performance of the system is continuously improved. However, the extent of improvement is becoming more and more stable, which has clarified that in the process of software collaborative design, it is necessary to give a response, make modification, and send feedback to software requirements in a timely manner.

In the second experiment, under the same preconditions, the estimation performance of the deep recursive least squares algorithm is compared with that of the ideal channel based on the same foundation, and the deep recursive least squares algorithm is simulated through the simulation curve. From the results, it can be seen that the performance of the deep recursive least squares algorithm is significantly superior to that of the ML algorithm. When the BER is up to the 0.0001 dimension, the actual calculated value of the signal-to-noise ratio is 3 dB less than the ideal value.

The four indexes in the collaborative software design are inherently correlated with each other. However, the ultimate goal is to develop software collaboratively. At this point, the joint decisions, the understanding and sharing of knowledge, and the credibility and reliability of decision making of software design collaborators become more important factors.

3. Conclusions

The complexity of software development has been increasing day by day, the difficulty of design is gradually increased as well, and the process has become more and more complicated. The traditional serial working approach can no longer adapt to the integrated design method at present. In this paper, the application of the related algorithm in the process of collaborative software design is explored based on the deep recursive least squares algorithm through the calculation and analysis of the bit error rate and complexity using the algorithm for the purpose of changing the traditional software design method.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research study was sponsored by Planning Project of Outstanding Youth Project of Scientific Research of Hunan Provincial Department of Education in 2019 (Research and implementation of image digital processing system based on DCT transform domain) (project no. 19B397). The authors are thankful for the support.

References

[1] A. Fichtner, D. P. Vanherwaarden, M. Afanasiev, and S. Simute, "The collaborative seismic earth model: generation 1," *Geophysical Research Letters*, vol. 3, no. 2, pp. 1–9, 2018.

[2] L. Liang, Y. Wei, C. Li, J. Tang, and X. Cao, "Inference with collaborative model for interactive tumor segmentation in medical image sequences," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2796–2809, 2017.

[3] J. Kong, Y. Ding, M. Jiang, and S. Li, "Collaborative model tracking with robust occlusion handling," *IET Image Processing*, vol. 14, no. 9, pp. 1701–1709, 2020.

[4] D. A. Quimby, J. E. Kawatu, K. M. Saul, and L. A. Schamus, "Implementation of a learning collaborative model increases Chlamydia screening at 37 family planning clinics: lessons learned from 3 cohorts," *Sexually Transmitted Diseases*, vol. 48, no. 2, pp. 102–109, 2021.

[5] R. Li, Z. Jiang, C. Ji, and A. Li, S. Yu, An improved risk-benefit collaborative grey target decision model and its application in the decision making of load adjustment schemes," *Energy*, vol. 156, no. 4, pp. 387–400, 2018.

[6] D. Riehle, M. Capraro, D. Kips, and L. Horn, "Inner source in platform-based product engineering," *IEEE Transactions on Software Engineering*, vol. 4, no. 3, pp. 1162–1177, 2016.

[7] V. A. Merchan, E. Esche, S. Fillinger, and G. Tolksdorf, "Computer-aided process and plant development. A review of common software tools and methods and comparison against an integrated collaborative approach," *Chemie Ingenieur Technik*, vol. 88, no. 2, pp. 50–69, 2016.

[8] C. L. Craney, T. Lau, W. T. Nelson, and A. Ghomeshi, "Collaborative middle school science outreach project using the Japanese lesson study model," *Journal of Chemical Education*, vol. 97, no. 5, pp. 89–96, 2020.

[9] P. T. Nguyen-Ha, D. Howrie, K. Crowley, and C. G. Vetterly, "A quality assessment of a collaborative model of a pediatric antimicrobial stewardship program," *Pediatrics*, vol. 137, no. 5, pp. 316–321, 2016.

[10] W. Mu, F. Bénaben, and H. Pingaud, "Collaborative process cartography deduction based on collaborative ontology and model transformation," *Information ences*, vol. 334, pp. 83–102, 2016.

[11] M. Henry, B. Jan, and V. D. H. Andre, "Collaborative modeling in software engineering," *IEEE Software*, vol. 35, no. 6, pp. 20–24, 2018.

[12] P. Nicolaescu, M. Rosenstengel, M. Derntl, R. Klamma, and M. Jarke, "Near real-time collaborative modeling for view-based Web information systems engineering," *Information Systems*, vol. 74, no. 5, pp. 23–39, 2018.

[13] V. Basili, L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "Software engineering research and industry: a symbiotic relationship to foster impact," *IEEE Software*, vol. 4, no. 4, pp. 1–10, 2018.

[14] H. Sharp, Y. Dittrich, and C. Souza, "The role of ethnographic studies in empirical software engineering," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 786–804, 2016.

[15] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 99, pp. 999–1013, 2020.