symmetry

MDPI

Article

Symmetry-Based Conflict Detection and Resolution Method towards Web3D-based Collaborative Design

Mingjiu Yu [1], Hongming Cai [2,*], Xiaoming Ma [2] and Lihong Jiang [2]

[1] Department of Industrial Design, Northwestern Polytechnical University, Shaanxi, Xi'an 710072, China; yumingjiu@nwpu.edu.cn

[2] School of Software, Shanghai Jiaotong University, Shanghai 200240, China; xinj2012sjtu@sjtu.edu.cn (X.M.); jiang-lh@cs.sjtu.edu.cn (L.J.)

* Correspondence: hmcai@sjtu.edu.cn; Tel.: +86-21-3420-5153; Fax: +86-21-3420-5145

Academic Editor: Yuhua Luo

Received: 30 March 2016; Accepted: 30 April 2016; Published: 11 May 2016

Abstract: In the process of web3D-based collaborative design, it is necessary to completely prevent operation conflicts among designers due to distributed environments and complex 3D models. Therefore, conflict detection and conflict resolution are of great significance to attain an acceptable result. In order to facilitate effective and smooth design work, a symmetry-based collaborative design framework is proposed using the X3D operation models. Combined considerations cover both models and operations, while different operation strategies are utilized for conflict resolution in web-based collaborative design. The strategy can achieve automatic operation, real-time conflict detection based on dynamically adjustable time, and conflict auto-detection and resolution with designers' customization. A proof-of-concept system is developed for verification. The proposed resolution shows good performance, scalability and interactivity in a case study.

Keywords: web3D Collaborative design; conflict Detection; conflict Resolution; X3D; symmetry

1. Introduction

It is very difficult for one person to accomplish entire work in large-scale projects such as large-sized industrial equipment design, architecture design, aircraft design, *etc.* Therefore, web-based collaborative design provides an effective and visual manner for different designers to work together so as to archive complex task. Compared with traditional client applications, web-based collaborative design has some advantages. The development of cloud-computing has provided web applications with better infrastructure services [1,2]; for instance, web-based collaborative word editing systems like Google docs and Microsoft office web version provide users with more convenience and portability [3,4]. Using WebGL, which has been developed and is supported by new versions of browsers like Mozilla Firefox, Apple Safari, Chrome, and so on, it is possible to display 3D models in browsers without plugins [5,6]. Since X3D conforms to XML structure, supports multi-level operations and can easily be converted to HTML format, we use X3D format models as design resource. As a consequence, building a web-based X3D collaborative design platform is very feasible both for requirements and technology.

However, when more than one designer operates the same 3D model in a large-scale environment, operations conflict may appear. How to detect and resolve conflict is a crisis task for cooperative application. In order to prevent the accumulation of conflicts, every designer needs to be timely informed whether his/her operations conflict with those of others, and the hope is that the conflicts can be resolved by the system automatically and quickly if there is any conflict. Thus, it is necessary to ensure the interactivity and real-time performance of this design platform. In this platform, Conflict detection (CD) [7] and conflict resolution (CR) [8,9] are important [10–14]. This paper studies

how to solve conflicts by editing operations to atomic X3D models or sub-scenarios. Additionally, decomposing task, which satisfies symmetry properties, will help to control cooperative distributed systems smoothly [15], while some evolutionary symmetry algorithms are proposed to search the space of symmetries effectively [16]. These examples prove that symmetry is useful to design and control complex distributed systems.

Therefore, a novel web-based X3D collaborative design framework is proposed, which focus on conflict detection and resolution to the symmetric designers. It can deal with automatic operation delivery, real-time CD and automatic CR. Its performance, scalability and design interactivity are demonstrated by experiments and analysis.

2. Related Work

The current solutions to collaborative design problems can be mainly divided into three types, namely, authority type, operation transformation, and other synchronous or asynchronous ones. These three types can be treated as pessimistic or optimistic according to their attitude towards conflicts (Table 1). Authority type is pessimistic and does not take conflicts into consideration, while the other two are optimistic and allow conflicts to arise and then they resolve them.

Table 1. Conventional solutions to collaborative problems.

Type	Collaboration Mechanism	Problem resolved
Authority Type	Lock Mechanism Traffic Light Mechanism Request Application Mechanism Right Mechanism Floor Mechanism	Conflict Avoidance
Operation Transformation	Sequence Transformation ABT Mechanism	Real-time Conflict Resolution in Word Editing System
	3D Operation Transformation	3D Model Dependency Conflict Resolution
Other Synchronous or Asynchronous	Semi-Synchronous Conflict Awareness Creative conflict resolution	Conflict Detection or Resolution with Designer Participation

For most of the authority mechanisms, e.g., lock mechanism, traffic light mechanism and floor mechanism [17–25], designers must apply for system's permissions before modifying collaborative models, so they can make all the operations execute sequentially and ensure data consistency among all workstations. However, these methods also reduce designer experience, and lead to deadlock.

Operation transformation (OT) is optimistic in that it transforms conflicting operations to enable them to be executed sequentially, which is commonly used in collaborative word editing system (CWES) [6,26–29]. However, this requires complex computations to transform operations to 3D models, and in 3D collaborative design only one type of conflicting operations can be restored, which is totally different from CWES. In addition, we propose a novel OT solution to deal with the dependency conflicts among 3D models [30]. Note that dependency conflicts are similar to the position conflicts in CWES, which cannot resolve the collaborative editing conflicts of 3D models. The rest of the solutions are primarily designers motivated synchronous ones, by which designers submit their operations autonomously [31,32], then the system makes CD based on versions and conducts CR by means of conflict combination [12], automatic conflict resolution with designer customization [33], creative conflict resolution [34], collaborative conflict resolution [35], conflict awareness [36], *etc.* However, if the time interval between two adjacent submissions were too long/short, the possibility of potential/missing conflicts will be very high. Moreover, both operation submissions and conflict resolution require designers' frequent participation, which would disturb the design work pace. Besides, the automatic conflict resolution does not provide detailed configuration parameters [37].

In short, these approaches only focus on operations or models; however, cooperative conflicts could the result of both models and operations. An effective method should cover both elements of models and operations so as to archive web-based cooperative work.

3. A Framework of Conflict Detection and Resolution in Web-based Collaborative Design

We propose a framework to deal with dynamic time related real-time conflicts detect and automatic conflicts resolution with detailed configuration parameters, based on the above analyzed problems. As Figure 1 shows, different designers use 3D interface or browser to operate a local 3D scene, but since they have the same symmetric properties, they usually put forward symmetric requirements to designer with symmetric operation, and the server uses a global scene, which stores all the completed 3D scenes. In the proposed symmetric framework, 3D browser and service-side architecture are presented symmetrically to design, implement and deploy relevant components. With storage-type and data processing-type cloud platforms, we divide the service-side into several modules, being responsible for designer management, real-time CD, automatic CR, version controlling and sending processed operation outcomes to all the connected browsers; while browsers are responsible for performing compatible operations, undoing conflicting operations and updating local version number.

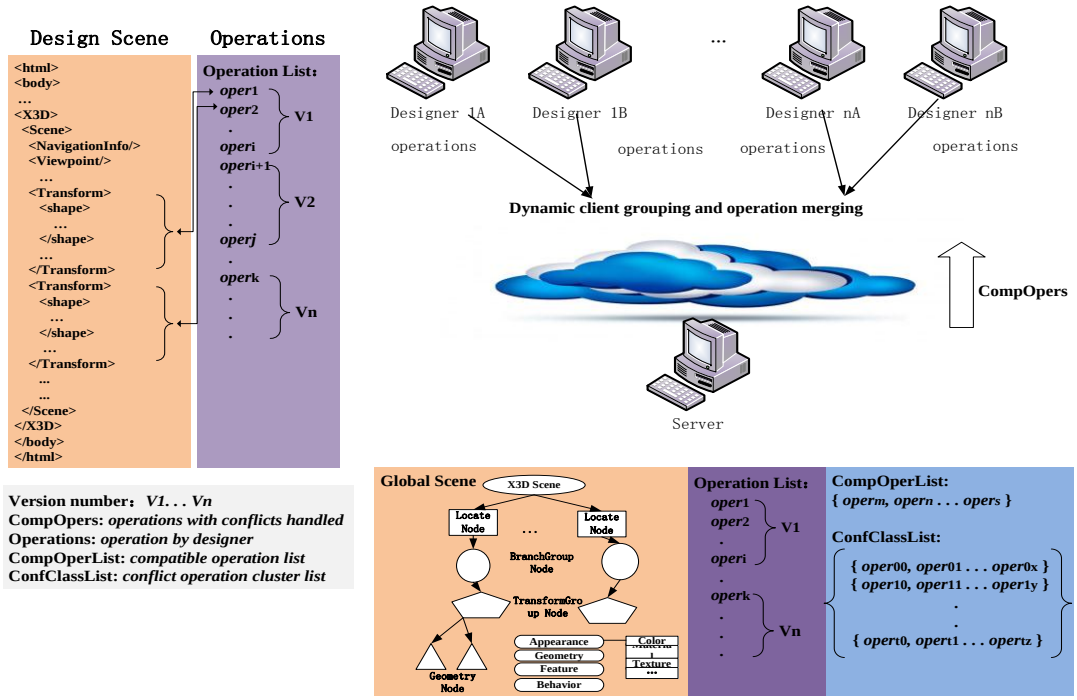

Figure 1. The symmetric framework of cooperative design model based on X3D structure.

Figure 1 shows the data view between browsers and service-side during design process. In order to minimize the data transfer volume between browsers and service-side, and to satisfy the need to undo conflicting operations, we adopt the operation-based mode of data representation compared with state-based and change-based modes [38]. Besides, we use symmetric replica design structure in which each browser side maintains the same history operation lists and version number as service-side, thus designers can edit models without constraints.

4. Basic Conceptions and Conflict Detection

First of all, some basic elements such as design resource and operations should be described, and then conflict detection could be carried out based on operation comparison.

4.1. Design Resource Representation Based on X3D

3D model: We use X3D format models (Figure 1) as design resource for their rich sources and XML schema. First, X3D structure can be converted to HTML structure and edited easily in a manner very similar to editing HTML format files. Second, just like XML, X3D also has extensibility and supports customization [37]. Based on these characteristics, all X3D models in a HTML file can be abstracted into an X3D tree with many nodes, and any updates on X3D models can be reflected to X3D document object model (DOM) tree [37].

Since X3D DOM tree has hierarchical structure, designers can modify all kinds of nodes collaboratively, such as the same nodes or the different nodes, and the parent or child nodes. In this paper, we mainly make CD and CR for operations to atomic models and sub-scenarios.

4.2. Operation Description

We would like to introduce some basic concepts so that our work can be better comprehended, including Operation Type, Concurrent Operations, Causal Operations, Operation Will, and Conflicting Operations.

Operation Type: Considering characteristics of X3D atomic models and sub-scenarios, we come up with X3D design system's operation types based on those in graphic editing system [34]; thus, the five operation types are *Add*, *Delete*, *Modify_Position*, *Modify_Color* and *Modify_Size*. Since some simple models have only one kind of color to be modified, we use Modify_Color to represent this operation type. To minimize the data transfer volume further, we use *JaSON* object with necessary data items (Table 2) as the Operation format. Besides, to provide designers with choices of custom fields depending on their future needs, we make all operation types' data fields scalable. The operation formats are found in Table 3.

Table 2. The description of data fields in operations.

Item	Description
Id	unique ID of an operation
scene_id	unique ID of a scene
user_id	unique ID of a designer
Type	operation type, *Add or Delete*, *etc.*
version_no	version the operation is based on
Params	added or deleted models array
Model	added or deleted model
scene_model_id	unique ID of a model in a scene
Name	model name
Style	model style
Trans	model position
Color	model color
Size	model size
Before	attribute value before modified
After	attribute value after modified

Table 3. Conflict relation between operations when editing the same scene model (except Add).

Operation Type	Add	Delete	Modify Size	Modify Position	Modify Color
Add	Compatible	Compatible	Compatible	Compatible	Compatible
Delete	–	Conflict	Conflict	Conflict	Conflict
Modify_Size	–	–	Conflict	Compatible	Compatible
Modify_Position	–	–	–	Conflict	Compatible
Modify_Color	–	–	–	–	Conflict

Add or Delete: [1 type, version_no, params: [{model:{scene_model_id, name, style, trans, color, size ...] ... }]}.

Modify_Position, Modify_Color and *Modify_Size: [1 type, version_no, params: [{model: {scene_model_id, before, after ...] ...}]}.*

Concurrent operations: They refer to those operations based on the same version [12,13,39].

Causal Operations: The operations that are upon different versions are called *Causal Operations* [12,13,39].

Operation Will: It refers to the model state that a designer wants to achieve by operation [40]. When a designer edits a model, he/she has no idea what others are editing, no matter whether his/her operation is before or after their operations in time. Therefore, we assume they are timeless. However, designers have the right to ignore this feature and suppose concurrent operations are time-related.

Conflicting Operations: When the designers modify the same model for similar attributes [12,13,40], it is often the case that their operations may be conflicting with another one (Table 3). We need to deal with different cases as follows: when designers A and B delete the same model, or modify a specific model for similar attribute to make them remain the same value, or make either of them have the same effect, we have to reserve both of them and regard them as contradictory to each other; this is due to the fact that the corresponding designers may have different authority values which can be used as a basis to make automatic CR. However, when a designer modifies the same attribute of one model repeatedly, we just keep the last one, because this will not influence the CR result.

4.3. Conflict Detection Based on Symmetry Operation Comparison

All conflicting operations can only happen among concurrent operations, so we simply need to conduct CD for concurrent operations. However, when to start and how to make CD are the issues we need to resolve in this section. As mentioned above, we intend to realize real-time CD to balance the calculation loads, so the specific CD process is described as follows:

When the system receives the first operation, it just puts it into the compatible operation list (ComOL); but after receiving the second operation, the system compares it with the first for conflicts since they are concurrent and symmetric, different processing is carried out based on comparison result (Figure 2). For the subsequent operations, the system first compares them with operations in ComOL, then the operations in ConOL one by one.

```
Input: ConOL
Output: CR result list: resOL
foreach operList in ConOL
   foreach oper in operList
      Integer pro1=0, pro2=0, pro3=0, pro4=0;
      If param1.isChecked //compute authority
factor
      pro1= oper.getAuthValue*param1.prop
      if param2.isChecked //compute time order
factor
      pro2= oper.getOrValue*param2.prop
      if param3.isChecked //compute oper type
factor
         pro3= oper.getProp*param3.prop
      if param3.isChecked //compute oper type
factor
         pro4= oper.getComVal*param4.prop
      Integer total=pro1+pro2+pro3+pro4
end
add the operation with biggest total value to resOL
end
return resOL
```

Figure 2. The description of operation comparison process.

5. Multiply Strategies for Conflict Resolution

In this section, we introduce an automatic CR method by describing parameter strategy and time strategy, and then fill the remaining leak.

5.1. Conflict Resolution Method

Then conflict resolution is carried out which involves different factors such as authority, time order and operation types. Different strategies can thus be constructed. Then, a formal description of automatic CR process is given as Figure 3.

```
Input: current JaSon format operation cuOper
Output: ComOL, ConOL.
//make conflict detetion
foreach oper in ComOL
  if cuOper.type==oper.type{
    if Oper.scene_model_id==oper.scene_model_id{
      if cuOper.user_id==oper.user_id{
          delete oper in ComOL
          add cuOper to ComOL
      return ComOL, ConOL.}
        if cuOper.after==oper.after{
         tempList=new List()
          add cuOper&oper to tempList
          add tempList to ConOL
return ComOL, ConOL.
        } }else
  continue
   end
for each operList in ConOL
  for each oper in operList
    as above
end
```

Figure 3. The formal description of automatic CR process

5.2. Parameter Strategy

In conventional CR ways [17], the design work pace is easily disrupted, so it is better to authorize designers to customize the CR parameters and resolve conflicts automatically. In making automatic CR process, the system first calculates priorities for every conflicting operation and then selects the first symmetry type as the CR result.

Taking different designers' technical competences, design experiences, different operation type, and the style of models into consideration, we give authorized designers four CR parameters to use. They express their intention to make CR by checking and assigning different proportions to each parameter. Besides, we also support parameter expansion to satisfy the design needs in the future. Through establishing integrated parameter basis, parameter initialization and calculation mechanisms, authorized designers can create their new parameters to make automatic CR. The four parameters are as follows.

Designer Authority: Every registered designer in our system will be allocated an authority value corresponding to his or her technical competence and design experience. The higher capability and richer experience, the bigger authority value will be assigned to him or her. If the authorized designer selects this parameter and assign a proportion to it, we can compute the influence factor of this parameter by multiplying this designer's authority value and this proportion.

Time Order: The time here refers to the time when the system receives an operation. By checking and setting this parameter, we can deactivate the timelessness of concurrent operations mentioned above, and emphasize the importance of time order when making automatic CR.

Operation Type: Conflicting operations may happen between modifying operations, or modifying operation and deleting operation. Because the models can no longer be edited once deleted, the deleting operations may have greater influences than other types. Based on the type difference designer can assign different priorities to different operation types to show their protection degree for them.

Model Compatibility: Most 3D models have attributes of color, texture, style, *etc.* In this cooperative design framework, for example, symmetric designers A and B collaboratively change a chair's color to red and gray, respectively, which can promote the better operation result. For instance, the gray is more coherent with the desk's color, so we tend to protect the more coherent operation by checking the parameter.

5.3. Time Strategy

ComOL and *ConOL* are the results that can be obtained from CD; and the version number will remain the same if no CR is made. Moreover, the size of *ComOL* and *ConOL* will become bigger and bigger, and CD is made more time-consuming. Thus, the system needs to choose an appropriate time as the trigger point to make automatic CR, and limits the sizes of *ComOL* and *ConOL* to a proper range.

A law needs to be presented before presenting the time strategy.

LAW 1: The real-time performance will get worse if the selected time is too long, which is because the designer does not have a chance to know whether his/her operations have already been sent to service-side in a timely manner or not, which may have conflicts with others. If this is the case, the designer has to perform more conflicting operations because the conflicts cannot be resolved in time. On the other hand, if the time is too short, the system may mistake concurrent operations as causal operations more often, and result in missing some conflicting operations and poor protection of designers' operation will.

It is almost impossible to find a fixed time used for CR due to many factors such as the uncertainties of designers' number, operation habits, network conditions and so forth. For instance, if there are more than two designers, the time should be set longer since the system needs plenty of time to deal with the concurrent operations. Therefore, we assume the time should be dynamically adjustable to solve this problem. Furthermore, we also provide designers an opportunity to customize new time strategy. Next, we introduce the time plan by answering the following three questions:

Question 1: How can you tell if the selected time is too long or short?

Different situations caused by time selection have been introduced by LAW 1. Therefore, our system can tell the time by identifying these two different situations. Here, two indexes are applied to calculate at the end of each version period to indicate them roughly. They are *Avoidable Conflict No*. and *Late Operation No.*

Avoidable Conflict No. refers to the number of conflict operations that could have been avoided if an appropriate time is selected. It is used to represent the situation where the selected time is too long. Conflict operations will accumulate continuously and some unnecessary conflicting operations may appear if the time to make automatic CR is delayed.

Late Operation No. refers to the number of operations that maybe received late by the system, and it is usually used to represent the situation in which the selected time is too short. The server may not start CR timely, and miss quite a lot of concurrent operations if the time is too short; and when CR is accomplished, the version number will be updated, and the system may receive operations with a

smaller version number than the existing one. Nevertheless, whenever the server makes CR, the late operations cannot always be easily avoided since it is difficult to predict designers' operation time. Despite this, the index can help us to know more about the present situation.

Question 2: How do you initialize the time?

We are determined to start CR when the first avoidable conflict operation is detected or the total operation number is equal to the number of designers. We then apply this time interval as the selected time in next version period, with many uncertain factors taken into account before actual design begins.

Question 3: How do you update the time before entering next version period?

We can predict whether the selected time is too long or short by the two indexes, and decide whether to decrease or increase it so as to keep these indexes in the least possible range. We would like to cope with this problem on the basis of our experimental data. Experiments show that it takes an average processing time of two milliseconds for one operation. Suppose there are one hundred designers operating the same scene and each designer sends an operation at the same time, the total time would be two hundred milliseconds. However, in the actual cases, it is almost impossible for all the designers to operate simultaneously, so it often takes more than two hundred milliseconds. Thus, we assume the time is no less than two milliseconds, taking the number of product designers into consideration to ensure all the designers' *operation will*.

Actually, compared with avoidable *Conflict No.*, *late Oper No*. is more difficult to reduce by adjusting the selected time, and this problem is solved in next subsection. However, avoidable conflicts make worse influences than late operations, so we give priority to reducing *avoidable Conflict No.*, and meanwhile reduce *late Oper No*. as far as possible. In order to reduce *avoidable Conflict No.* quickly, when *avoidable Conflict No.* is not equal to zero, we cut the time in half until the time is less than two milliseconds; when *avoidable Conflict No.* is equal to zero, we add the time by 50 until reaching 500 milliseconds. This time is the mean value of the time interval between two operations sent by a designer and is adjustable depending on real situation. We will verify this time strategy by experiments in next section.

5.4. Leak Filling

There still may be a problem even though the time is dynamically adjustable due to the many uncertain factors discussed above; that is, the system may receive operations that are on former versions after the version has been updated. To further ensure designers' operation will, we have to remedy this problem. The solution is as follows.

The system will restore the CR result for one or more version periods (according to different designer settings) after the version number is updated; and the system will compare them with the reserved operations for conflicts individually after receiving the late operations. Then there are two possible results: First, the system will ignore the conflicts and ask the browser ends to handle them if there are conflicts because corresponding browser end has already performed this operation before it is sent to service-side; and if this operation has conflicts with the result operations sent by the system, then the browser end would undo this operation automatically, so there is no need to handle them. Second, if there is no conflict, the system changes the operation's version number to the present one, treating it as concurrent operation and continues processing. By this solution, this problem is well solved and designers' operation will is also well protected.

6. Analysis and Evaluation

In this section, we will verify the framework by case study and analysis.

6.1. System Implementation and Case Study

A web-based cooperative design system is implemented, which acts as an indoor design modules in a real estate enterprise. By means of web-based indoor decorative module, the preferences and

requirement of users are collected for further disposing. The end-users could interact with designer or other users to make their requirement clearly. A scenario is given to verification as follows. Designer A edits a scene model carpet in his or her web page (Figure 4), and the corresponding operation is sent to CD module automatically, then this module makes conflict comparison between this operation and the precious ones on the same version of X3D scene, and puts it into different operation list according to operation comparison result. Meanwhile, the CR module decides whether to start conflict resolution based on dynamic adjustable time, and it automatically resolves conflicts according to the configured parameters, refines the dynamic time and updates scene version; then the designer management service module sends the results to other designer in the cooperation, and the storage-type service module saves the result operations. After receiving the result operations and message tips sent by server, every browser cancels his/her local conflicting operations, performs compatible operations automatically and updates the local version number of scene. Even though the conflicting operations are undone, he or she can still decide whether to remake the conflicting operations on new scene version again.

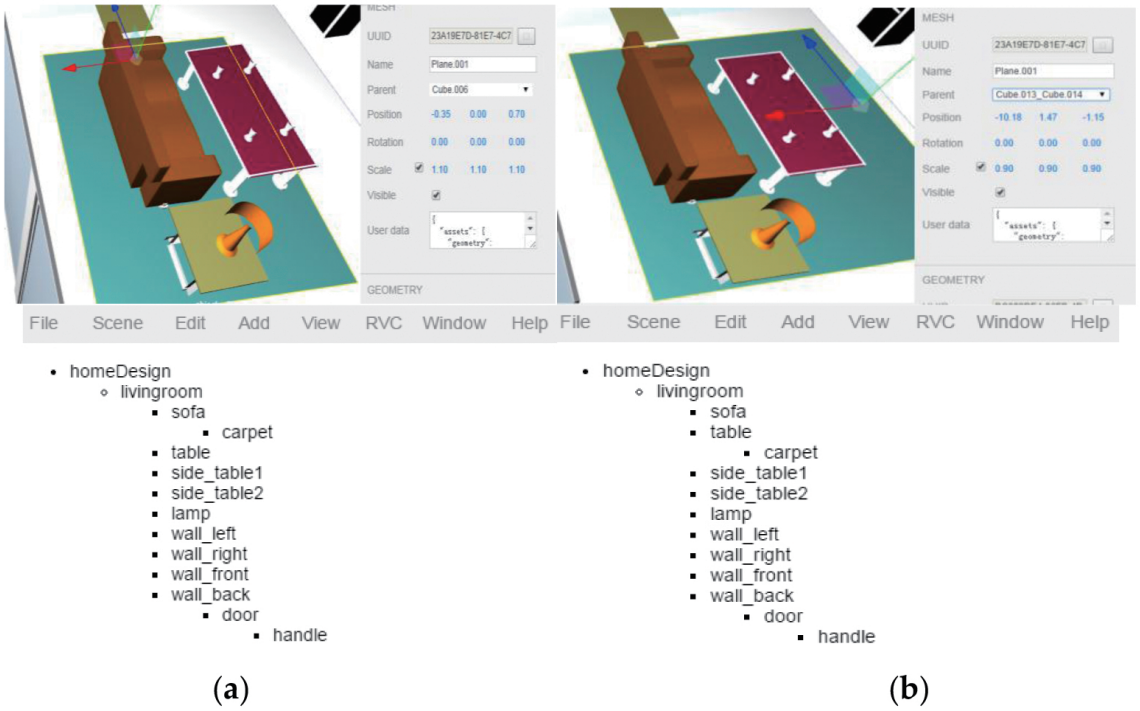

Figure 4. Editing conflict in web-based X3D collaborative design: (**a**) designer A's concept view based on X3D; and (**b**) designer B's concept view based on X3D.

The concept view based on X3D is given in Figure 4, and conflict detection will find a structure conflict. The conflict resolution is based on dynamic mechanism. If Designer A uses related operation to composite Picture with wall-right by position parameter, and Designer B uses Group operation to composite wall-right with Floor, dependency conflicts will happen. Therefore, priority of Designer A, Designer B and the Operation List combined Designer A and Designer B will be calculated by DP mechanism, then the result will return to Designer A and Designer B. Models transformation is limited by only related compatible operations but not all the models in different local scenes (Figure 5).

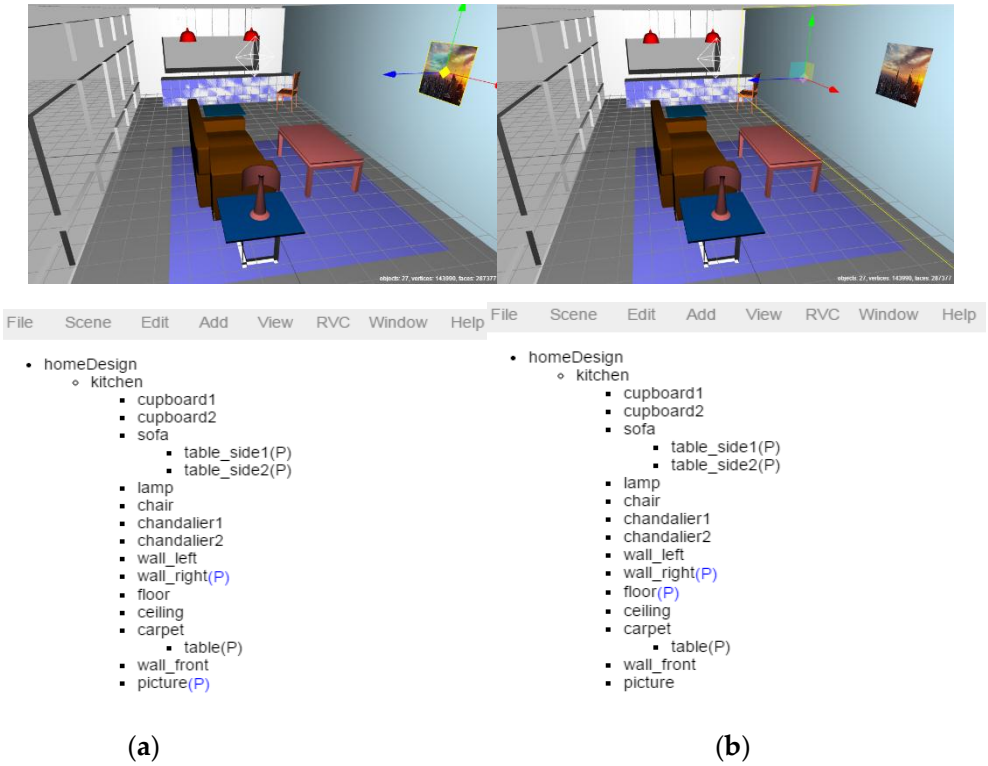

Figure 5. Collaborative dependency conflict: (**a**) concept view of designer A; and (**b**) concept view of designer B.

6.2. Evaluation

As mentioned above, the proposed framework is implemented and run as an instance in collaborative interior design. *Avoidable Conflict No.* and *late Operation No.* are applied to measure protection degree of a designer's *operation will*, and operation processing time is used to demonstrate system responding performance.

We use threads to simulate designers, and the suspended time of threads to simulate the operating time interval of designers or network delay, to make a more real simulation of collaborative design work. In our study, we simulate two groups of designers with a number of fifty and eighty, respectively, and produce operations for each group by combining the five operation types according to the requirements of different scenarios.

Here, we give each thread ten operations, and make each of them send operations at time intervals from 150 to 1150 milliseconds randomly; the number of conflicting operations is also from naught to an enormous value, and we use designer authority parameter to make automatic CR. According to the test scenarios described above, we run the instance and collect running results, and then we organize them into histograms (Figures 6 and 7).

The first scenario is with 50 designers and 10 operations for each of them. These operations have five types, and have conflicts among them (except for *Add*) by modifying the same model's same attribute to different values, so avoidable conflicts will happen; besides, the operations from different designers also have many conflicts within some of them. The selected time is from 100 to infinity; and will be cut in half or added by 50 depending on running results in each version period (Figure 6). The second scenario is also with 50 designers and 10 operations for each of them. However, none of the operations have conflicts with each other (Figure 6).

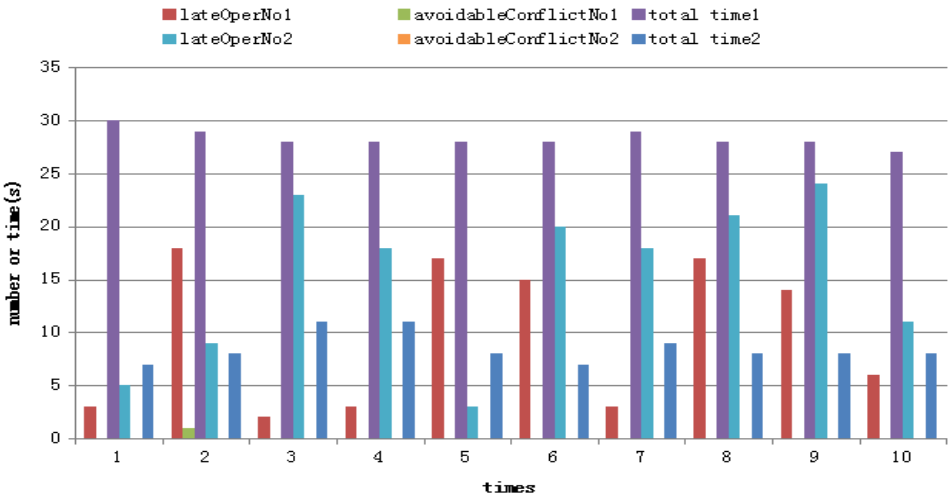

Figure 6. Fifty designers, many conflict operations (1) and no conflict operations (2).

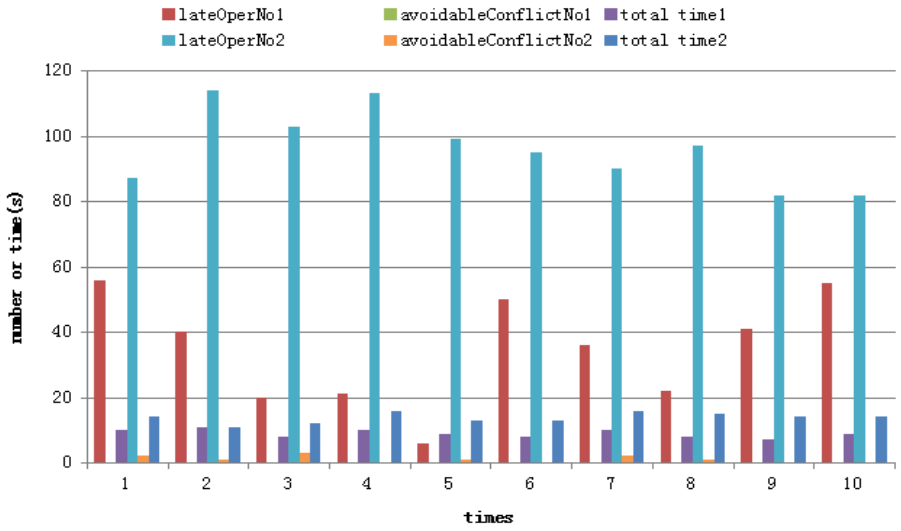

Figure 7. Eighty designers, many conflict operations (2) or no conflict operations (1).

Both the third and the fourth scenario are with 80 designers and 10 operations for each of them, and the selected time is from 200 to 500 milliseconds. However, for the third scenario, none of the operations have conflicts with each other, while for the fourth scenario, most of the operations have conflicts with each other (Figure 7).

According to our experiment results, the avoidable conflict operations are well prevented and the total time cost is acceptable based on designers' operating habits; however, the number of late operations is relatively high because we make the time interval between when the operations are sent so tight as to increase the system's request load. In fact, designers will not operate too frequently, so the number of conflicting operations will not be so great and the hardware environment will run faster too. Besides, most of the late operations can be well handled by the leak filling solution. Therefore, the practical result of our solution will run better.

Moreover, compared with conventional solutions in Table 4, the advantages and disadvantages of our framework are apparent. Among those comparative items, the number of redundant operations in our framework is relatively high; thus, in order to minimize the computational complexity caused by them, we combine the conflicting operations from the same designer once they are received by service-side. Additionally, to ensure system interactivity and operation will, we also require that

designers are online during collaborative design process. The proposed framework in this paper provides designers with a scalable and efficient collaborative design model with high interactivity.

Table 4. The various comparison results of our framework with conventional solutions.

Item	Our Framework	Creative CR	3D Operation Transformation
Resolved Problems	3D Editing CD&CR	2D CD&CR	3D dependency conflict CD&CR
Network Requirement	Online	Online	Online/Offline
Redundant Operation	Much	Little	Little
Collaborative Design Efficiency	High	Low	High
Interactivity & Operation Will	High	Low	Low
Scalability	High	High	Low

6.3. Discussion

6.3.1. Extensibility

Different from the conventional solution that require frequent participation of designers, this collaborative design framework can define data fields of operation and set up new time strategy, which ensures collaboration interactivity and timeliness. The framework also supports the designer to configure the CR parameters and supply the system with automatic CR. For instance, designers can activate a parameter by the following two means: setting its proportion to a nonzero value, or deactivating a parameter by setting its proportion to zero. What is more, they can also create new parameters according to their requirements and result in more choices. The designers in this distributed system are always symmetric to finish the design task cooperatively, which can promote producing the better operation result.

6.3.2. Interactivity and Efficiency

With the interactive help of this design frame, every designer can know the other's operations by selecting an appropriate CR time. The frame also tries to prevent potential conflicting operations, which can save designers much time on resolving those conflicting operations. The case above also verified that automatic CR makes the design work more efficient.

7. Conclusions

We propose a symmetric, efficient, and web-based X3D collaborative design framework, which has realized real-time CD and automatic CR by dynamic adjustable time and configurable parameters. Therefore, the solution can protect designers' operation will, improve the design efficiency and provide designers with better interactivity than conventional solutions. Besides, the framework is revised to be more scalable in operation fields, time strategy, and parameter strategy in order to give designers more choices according to their actual needs. In collaborative design process, we propose an innovative method to analyze and define conflicts to enhance design practice and quality. The scene, 3D model, operation type, and operation attribute are applied as design resources to provide the symmetric designers, and in the process of allocation, the results can be compared under the same design resources and symmetric designers, which is due to the interactivity of symmetric framework.

We also propose a solution to automatically detect and resolve conflicts under the symmetric framework, and designer's capability and experience, differences among operation types, operation characters and comprehensive requirements of particular scenes are taken into consideration in the dynamic and symmetric mechanism. A series of parameter criterions will be set out to solve the conflicts existing in the above-mentioned collaborative design process by setting criterion weights.

In this paper, we discussed the choice of time interval for service-side conflict detection and the regular patterns among time range, undo operation, and potential conflict; however, the proposed solution applies service-side detection and resolution, which cannot avoid the ignorance of some problems and ensure designer's operation will. It is necessary to explore other kinds of factors that may influence the selection of time in future research. We mainly aim at solving conflicts of editing operations to atomic X3D models or sub-scenarios, and did not consider multi-level operations in our study. In fact, due to X3D model's extensibility, multilevel operations are supported. Moreover, the time strategy and parameter strategy are not perfect. We will consider and resolve them in future research.

Acknowledgments: This research is supported by the National Natural Science Foundation of China under No. 61373030 and the 111 Project of China (B13044).

Author Contributions: Hongming Cai and Mingjiu Yu proposed the main idea, and Minghong Cai designed its frame. Mingjiu Yu contributed to finish the main part of the manuscript, and Hongming Cai revised it. The web-based cooperative design system was developed by Xiaoming Ma, and Lihong Jiang is responsible for case study, result analysis and discussion. All of authors have approved the final manuscript.

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

References

1. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
2. Patel, A.; Taghavi, M.; Bakhtiyari, K.; Celestino Júnior, J. An intrusion detection and prevention system in cloud computing: A systematic review. *J. Netw. Comput. Appl.* **2013**, *36*, 25–41. [CrossRef]
3. Spaeth, A.D.; Black, R.S. Google docs as a form of collaborative learning. *J. Chem. Educ.* **2012**, *89*, 1078–1079. [CrossRef]
4. Behles, J. The use of online collaborative writing tools by technical communication practitioners and students. *Tech. Commun.* **2013**, *60*, 28–44.
5. Leung, C.; Salga, A. Enabling webgl. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–27 April 2010.
6. Tarukawa, K.; Inoue, T.; Okada, K. Multi-view is useful for more accurate understanding of object in a virtual soccer field. In Proceedings of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD), Whistler, BC, Canada, 27–29 June 2013.
7. Ma, X.; Cai, H.; Jiang, L. A creative approach to conflict detection in web-based 3D cooperative design. In *Cooperative Design, Visualization, and Engineering*; Springer: Basel, Switzerland, 2014; pp. 261–268.
8. Beunza, J.-J. Conflict resolution techniques applied to interprofessional collaborative practice. *J. Interprof. Care* **2013**, *27*, 110–112. [CrossRef] [PubMed]
9. Guimarães, M.L.; Silva, A.R. Improving early detection of software merge conflicts. In Proceedings of the 34th International Conference on Software Engineering, Zurich, Switzerland, 2–9 June 2012.
10. Koegel, M.; Helming, J.; Seyboth, S. Operation-based conflict detection and resolution. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, Vancouver, BC, Canada, 17 May 2009.
11. Koegel, M.; Herrmannsdoerfer, M.; von Wesendonk, O.; Helming, J. Operation-based conflict detection. In Proceedings of the 1st International Workshop on Model Comparison in Practice, Malaga, Spain, 1 July 2010.
12. Cai, X.; Li, X.; He, F.; Han, S.; Chen, X. Flexible Concurrency control for legacy CAD to construct collaborative CAD environment. *J. Adv. Mech. Des. Syst. Manuf.* **2012**, *6*, 324–339. [CrossRef]
13. Estler, H.C.; Nordio, M.; Furia, C.A.; Meyer, B. Unifying configuration management with merge conflict detection and awareness systems. In Proceedings of the 22nd Australasian Conference on Software Engineering (ASWEC), Melbourne, Australia, 4–7 June 2013.
14. Alonso-Ayuso, A.; Escudero, L.F.; Olaso, P.; Pizarro, C. Conflict avoidance: 0–1 linear models for conflict detection & resolution. *Top* **2011**, *21*, 485–504.
15. Karimadini, M.; Lin, H. Synchronized task decomposition for two cooperative agent. In Proceedings of the IEEE Conference on Robotics Automation and Mechatronics (RAM), Singapore, 28–30 June 2010.

16. Lincke, J.; Krahn, R.; Ingalls, D.; Röder, M.; Hirschfeld, R. The lively partsbin—A cloud-based repository for collaborative development of active web content, System Science (HICSS). In Proceedings of the 45th Hawaii International Conference, Maui, HI, USA, 4–7 January 2012.
17. Jing, S.-X.; He, F.-Z.; Han, S.-H.; Cai, X.-T.; Liu, H.-J. A method for topological entity correspondence in a replicated collaborative CAD system. *Comput. Ind.* **2009**, *60*, 467–475. [CrossRef]
18. Cai, X.T.; Li, W.D.; He, F.Z.; Li, X.X. Customized Encryption of Computer Aided Design Models for Collaboration in Cloud Manufacturing Environment. *J. Manuf. Sci. Eng. Trans. ASME* **2015**, *137*, 1–10. [CrossRef]
19. Baldwin, C.; von Hippel, E. Modeling a paradigm shift: From producer innovation to user and open collaborative innovation. *Organ. Sci.* **2011**, *22*, 1399–1417. [CrossRef]
20. Chellali, A.; Jourdan, F.; Dumas, C. VR4D: An immersive and collaborative experience to improve the interior design process. In Proceedings of the 5th Joint Virtual Reality Conference of EGVE and EuroVR, JVRC, Paris, France, 11–13 December 2013.
21. Juntunen, T.; Kostakos, V.; Perttunen, M.; Ferreira, D. Web tool for traffic engineers: Direct manipulation and visualization of vehicular traffic using google maps. In Proceedings of the 16th International Academic MindTrek Conference, Tampere, Finland, 3–5 October 2012.
22. Sun, H.; Fan, W.; Shen, W.; Xiao, T. Ontology fusion in high-level-architecture-based collaborative engineering environments. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 2–13. [CrossRef]
23. Hepworth, A.; Staves, D.; Hill, L.; Tew, K.; Jensen, C.G.; Red, W.E. Enhancements for improved topological entity identification performance in multi-user CAD. *Comput. Aided Des. Appl.* **2015**, *12*, 1–12. [CrossRef]
24. Turrin, M.; von Buelow, P.; Stouffs, R. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Adv. Eng. Inform.* **2011**, *25*, 656–675. [CrossRef]
25. Kang, J.; Park, J.; Suk, S. Design of a distributed personal information access control scheme for secure integrated payment in NFC. *Symmetry* **2015**, *7*, 935–948. [CrossRef]
26. Li, D.; Li, R. An admissibility-based operational transformation framework for collaborative editing systems. *Comput. Support. Coop. Work (CSCW)* **2010**, *19*, 1–43. [CrossRef]
27. Shao, B.; Li, D.; Gu, N. A sequence transformation algorithm for supporting cooperative work on mobile devices. In Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, Savannah, GA, USA, 6–10 February 2010.
28. Shao, B.; Li, D.; Lu, T.; Gu, N. An operational transformation based synchronization protocol for web 2.0 applications. In Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, Hangzhou, China, 19–23 March 2011.
29. Sun, D.; Sun, C.; Xia, S.; Shen, H. Creative conflict resolution in collaborative editing systems. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, Seattle, WA, USA, 11–15 February 2012.
30. Sun, C.; Xu, D. Operational transformation for dependency conflict resolution in real-time collaborative 3D design systems. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, Seattle, WA, USA, 11–14 February 2012.
31. Oehrn, C.R.; Hanslmayr, S.; Fell, J.; Deuker, L.; Kremers, N.A.; Do Lam, A.T.; Elger, C.E.; Axmacher, N. Neural communication patterns underlying conflict detection, resolution, and adaptation. *J. Neurosci.* **2014**, *34*, 10438–10452. [CrossRef] [PubMed]
32. Trappey, A.J.; Trappey, C.V.; Wu, C.-Y.; Fan, C.Y.; Lin, Y.-L. Intelligent patent recommendation system for innovative design collaboration. *J. Netw. Comput. Appl.* **2013**, *36*, 1441–1450. [CrossRef]
33. Gu, N.; Kim, M.J.; Maher, M.L. Technological advancements in synchronous collaboration: The effect of 3D virtual worlds and tangible user interfaces on architectural design. *Autom. Constr.* **2011**, *20*, 270–278. [CrossRef]
34. Jung, J.J. Computational reputation model based on selecting consensus choices: An empirical study on semantic wiki platform. *Expert Syst. Appl.* **2012**, *39*, 9002–9007. [CrossRef]
35. Brosch, P.; Seidl, M.; Wieland, K.; Wimmer, M.; Langer, P. We can work it out: Collaborative conflict resolution in model versioning. In *Ecscw 2009*; Springer: Vienna, Austria, 2009; pp. 207–214.
36. Ahmed-Nacer, M.; Urso, P.; Balegas, V.; Preguiça, N. Concurrency control and awareness support for multi-synchronous collaborative editing, Collaborative Computing. In Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Austin, TX, USA, 20–23 October 2013.

37. Behr, J.; Eschler, P.; Jung, Y.; Zöllner, M. X3DOM: A DOM-based HTML5/X3D integration model. In Proceedings of the 14th International Conference on 3D Web Technology, Darmstadt, Germany, 16–17 June 2009.
38. Holyoak, V.L.; Red, E.; Jensen, G. Effective collaboration through multi user cax by implementing new methods of product specification and management. *Comput. Aided Des. Appl.* **2014**, *11*, 560–567. [CrossRef]
39. Goldman, M.; Little, G.; Miller, R.C. Real-time collaborative coding in a web ide. In Proceedings of the 24th annual ACM symposium on User interface software and technology, Santa Barbara, CA, USA, 16–19 October 2011.
40. Wang, X.; Love, P.E.D.; Kim, M.J.; Wang, W. Mutual awareness in collaborative design: An augmented reality integrated telepresence system. *Comput. Ind.* **2014**, *65*, 314–324. [CrossRef]