Layered Consistency Management for Advanced Collaborative Compound Document Authoring

Johannes Klein(✉), Jean Botev, and Steffen Rothkugel

Faculty of Science, Technology and Communication,
University of Luxembourg, 1359 Luxembourg, Luxembourg
{johannes.klein,jean.botev,steffen.rothkugel}@uni.lu

Abstract. In distributed collaborative document authoring environments, the preservation of a globally consistent data state is an important factor. However, synchronization conflicts are unavoidable and constitute a serious challenge. Our advanced compound document system provides the basis for a novel consistency management approach, in particular regarding autonomous conflict detection and resolution. Current techniques to achieve and maintain global consistency in distributed environments almost exclusively utilize file-based data structures, thereby limiting the accessibility to supplementary information.

In this paper, we present a layer-based consistency management approach harnessing a fine-granular, graph-based data representation and relational dependencies. We discuss the application of concurrent conflict detection and resolution modules designed to preserve user intent while avoiding workflow interruptions. The combination of an advanced compound document system with autonomous, layer-based consistency management has the potential to notably increase reliability and facilitate the collaborative authoring process.

Keywords: Compound document systems · Document engineering · Distributed authoring · Collaboration · Consistency management · Conflict detection and resolution · Intention preservation

1 Introduction

Global data consistency in a distributed document authoring system is commonly ensured by either utilizing a conflict-free or commutative replicated data type [1], operational transformation algorithms [2], or by employing conflict detection and resolution schemes [3,4]. The latter use optimistic synchronization and eventual consistency [1,5], as conflicts are expected to only occur rarely. This assumption allows for highly responsive implementations with locally issued commands being applied instantaneously. Assuming a conflict-free application, the commands are afterwards distributed to the remote sites. In case of a conflict, an autonomous resolution is aspired. Eventually, the distributed system will reach a consistent state in which possible conflicts have been resolved and no new commands are pending.

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 281–288, 2016.
DOI: 10.1007/978-3-319-28910-6_25

However, for most systems, conflict detection and resolution is more complex and error-prone. In part, this can be attributed to relying on established file-based data structures, which complicate or do not provide access to further information. Particularly approaches based on intention preservation [2,6] require supplementary data as a prerequisite to determine a collaborator's intent. Therefore, we utilize an advanced Compound Document System (aCDS) [7] as a basis for consistency management in a distributed document authoring environment. The aCDS's fine-grained, graph-based structure facilitates storing the data while allowing for an accurate attribution of metadata to elements of arbitrary size and complexity. Moreover, element-centric attributes as well as intra- and inter-document relations, both user-generated or created by the system, are supported. These relations are particularly useful for intention preservation features as they allow for more comprehensive capturing of an edit's circumstances. The modular design of the aCDS furthermore facilitates the maintainability and a straightforward extensibility of the system.

These properties form the basis for the layered consistency management (LCM) approach proposed in this paper. LCM utilizes the native data representation to partition conflict detection and resolution functionality according to the unique structure of the data in the aCDS. Instead of employing the detection and resolution concepts on a block of data, the functionality is distributed among several specialized consistency management modules. These are selected based on the type of edited data and their location in the aCDS graph representation.

Currently, LCM is being integrated into our aCDS. In order to facilitate global information exchange through inter-document relations, a central server connects the distributed collaborators. Furthermore, a master copy of all compound documents is kept server-side, allowing for the application of all commands issued throughout the system and ensuring a consistent document representation. In the following section, we present the aCDS and its graph-based data structure. A discussion of the LCM approach and its close integration with the aCDS's specific representation of data follow in Sect. 3. Related work is reviewed in Sect. 4 before concluding this paper with a brief summary in Sect. 5.

2 Compound Documents

This section introduces the core concepts employed in the aCDS. In particular, we will discuss interdependencies between the underlying data representation and the information managed therein, which is fundamental to our consistency management approach.

2.1 The Advanced Compound Document System

Compound document systems transparently integrate different, application-specific documents or document parts, and have been a recurring subject of research in the past decades [8]. They offer a wide range of exciting possibilities with regard to the creation and management of complex documents. The aCDS developed by our group [7] serves as the basis for our collaborative authoring system. Its combination of fine-grained data representation, inherent modularity and deep integration of relations and attributes make it an ideal candidate for distributed and collaborative document authoring. It allows for in-place editing of various data types and its modularity enables simple extensibility. Harnessing these features for a real-time collaboration environment while maintaining its advanced editing capabilities, responsiveness, and resilience to conflict, is the main objective of this work.

A graph-based structure stores the main data in dedicated nodes and the concomitant metadata in associated nodes and edges. The granularity of the data is, in contrast to file-based approaches, not predefined but directly related to user operations and the layout of the document. This allows for a precise representation of the compound document and its accompanying information. For example, a copy-and-paste command not only makes the elements available in other parts of the document, but it preserves the implicit relations between source and copy for subsequent use. File-based systems generally fail to maintain this relational information beyond the execution of the command.

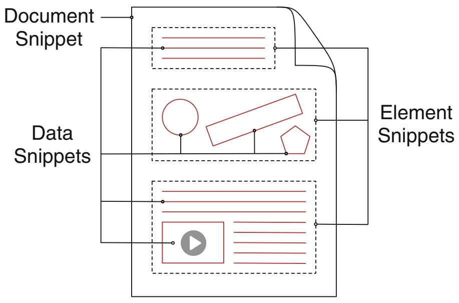

Fig. 1. aCDS data representation

The data representation of the aCDS is schematically depicted in Fig. 1. Every compound document is defined by its Document, Element, and Data Snippets along with their positions in the graph, contents, and relations. Each data type has specialized modules handling its peculiarities and defining its representation. All nodes form a parent-child relationship, therefore enabling straightforward navigation and data retrieval through the graph structure.

Intra- and inter-document relations between the elements of an aCDS are either created explicitly by users invoking commands or implicitly by analyzing their behavior. Similarly, attributes containing additional information can be specified manually by the user or automatically by the system. Manually defined attributes provide valuable information about a user's intentions and document perception, e.g., alternate element versions or source references, which would be otherwise complex to establish. The system itself utilizes attributes to store for instance the last editors or the dataset memberships of an element.

3 Layered Consistency Management

File-based approaches mostly include documents as single entities into the synchronization process, thereby leaving the data's inherent structure unused for a more precise and partitioned procedure. In contrast, and due to the specific properties of our aCDS, we are able to utilize said information in the proposed, layered consistency management (LCM) concept. This enables the analysis of individual elements, supplemented by their relational dependencies, metadata, attributes, and type-specific syntactic and semantic information.

The finer-grained data representation permits the precise localization of conflict-related data, thereby limiting the necessary processing tasks. Moreover, the concurrent application of specialized consistency management modules enables the independent analysis of subsets of an element's data, while also providing the basis for parallel task execution.

Our conflict management is logically partitioned into three distinct layers, directly related to the data representation of the aCDS as depicted in Fig. 1. The first two layers contain structural information and metadata whereas the bottom layer stores the actual data of an element. Document Snippets, which are located on the first layer, include the element's metadata and relational information. Particularly the latter are of great importance regarding the detection and resolution of conflicts involving the main data. On the second layer, Element Snippets store information about the structural representation of the element's graph and serve as an insertion point for other documents. As the structural and visual representation of a document are closely related, this data is utilized for, e.g., conflict analysis involving layout complications. The third layer comprises the actual data stored in Data Snippets with type-specific formats defined by the respective LCM modules.

The distribution of the consistency management logic over specialized modules in combination with the layered data representation allows for reusability and synergy effects. For instance, an elementary module capable of detecting conflicts in XML-based elements is combined with other modules enabling a dedicated analysis of a vector graphics element. Furthermore, the evaluation of one conflict can be limited to a single module and one layer, or, in a more complex scenario, include multiple different layers and modules.

LCM allows for the application of intention preservation techniques in a distributed environment by combining layer-based synchronization with the utilization of supplementary metadata. This reduces the need for manual conflict resolution and keeps workflow interruptions at a minimum [9]. Constant interference can lead to user dissatisfaction, isolation from the collaboration environment and eventually leaving the collaborator excluded from the group. Automatic consistency management is essential as synchronization conflicts are inevitable [10]. Still, manual conflict resolution is unavoidable, e.g., in case of concurrent edits of the same element. LCM enables supporting users involved in the resolution process with comprehensive data about the conflict which is generated utilizing all information available.

3.1 Layer-Based Conflict Detection

Consistency management is a multi-step process beginning with the local detection of synchronization conflicts. A conflict is introduced by the application of either a local or remote command to a data state dissimilar to the one it was initially invoked upon. Various factors, e.g., network latency, add time between invocation and application, thereby increasing the potential for encountering different data states for remotely received commands. Two distinct conflict areas are detected: issues with command application and ambiguous outcomes. They leave the data in a potentially inconsistent state or violate the user's intent. The conflict detection process can be classified into the following scenarios D.1 – D.4:

D.1 – A command has been executed without issues and the data remains consistent. The next command can be safely applied.

D.2 – A command has been executed raising at least one conflict which is limited to a single layer of the data structure. The corresponding conflict detection module analyzes the problem as a basis for the resolution process.

D.3 – A command has been executed raising numerous conflicts on multiple layers of the data structure. The corresponding conflict detection modules concurrently analyze each affected layer before individually providing the gathered information to the relevant resolution modules. The necessity to cumulate the conflict data is assessed during resolution, as it still may be possible all problems are isolated and can be concurrently resolved.

D.4 – A command is able to execute without conflict, but with ambiguous results. To guarantee the selection of a state conforming to global consistency constraints, each possible data state must be analyzed individually. The corresponding modules concurrently gather the related information on the affected layers and provide it to the relevant resolution modules. This information includes the circumstances of the application and all possible resulting data states.

The utilization of the aCDS's data structure facilitates the comprehensive conflict detection and analysis capabilities of LCM and provides the basis for the following, autonomous resolution process.

3.2 Layer-Based Conflict Resolution

In the second LCM phase, concurrently operating conflict detection modules utilize local and global data sources. These sources include information gathered during the detection process as well as data from related, but remote documents. Furthermore, the necessity for user involvement is established. The scenarios discussed in Sect. 3.1 lead to the conflict resolution approaches R.1 – R.4:

R.1 – Successful resolution is facilitated by solely utilizing the conflicting data. Distributed systems relying on a file-based data representation generally apply corresponding approaches.

R.2 – In addition to data experiencing the conflict, resolution requires the utilization of supplementary information still available on site. This includes related information from other local elements or documents and attributes.

R.3 – In contrast to R.2, sufficient information to resolve the conflict is not available locally. Still, relational dependencies indicate the presence of useful data on remote sites which can be accessed over the network. Although the inclusion of remote information is more time-consuming than purely local resolution, user involvement can still be avoided.

R.4 – The system is unable to perform an autonomous conflict resolution. Therefore, manual resolution by the collaborators is unavoidable. All available information, including the output of the detection modules, is processed and comprehensively presented to the users tasked with the solution. The preparation of a precise problem description not only expedites the process and thereby reduces the individual's workflow interruption but also enables proper resolution. These conflict resolution approaches rely heavily on the supplementary information available as part of our aCDS. Each approach involves one or multiple layers of the data structure into the resolution process. Concurrent and linked execution of the layer-specific modules is determined by the individual conflicts' requirements. With the exception of approach R.4, all resolve the corresponding conflict scenarios autonomously.

3.3 Local Concurrency Handling

Eventually, a site within the collaboration environment is already executing a command when it either receives another remote command or a new one is issued locally by the user. In this case, and regardless of any ongoing conflict resolution process, the current command execution is finished before the next is initialized. The sites therefore maintain FIFO queues storing all pending commands, ensuring a globally consistent state for command execution. This is necessary for relational dependencies to be able to provide reliable and up-to-date information.

4 Related Work

As opposed to consistency management approaches in distributed collaboration environments which rely mainly on file-based data structures [11], our LCM employs a more fine-grained representation based on compound documents.

One possible solution is to rule out conflicts by design by utilizing conflict-free replicated data types [1]. This requires further dedicated structures such as tombstones which retain elements for reference even after their deletion, a concept not needed when implementing LCM. Tombstones either result in an unbounded growth of the data structure as, e.g., observable in WOOT [12], or they require the use of garbage collection as applied in the Treedoc [13] implementation. Without tombstones, a linear or even sublinear space complexity with regard to the number of insert operations is achievable, as demonstrated by, e.g., LSEQ [11]. LCM, however, stores supplemental information such as relational dependencies and attributes only during the existence of the corresponding element. Thereby, the need for garbage collection is eliminated while preventing an unbounded growth of the data structure.

Another possible approach is the application of operational transformation as employed by, e.g., dOPT and GOT [2]. Similar to LCM, causality, intention, and global consistency are preserved in the presence of synchronization conflicts. As this is achieved through the conversion of the initial commands according to the requirements of the remote site's data state, transformation needs to be realized via commutative functions. Even for functions treating only character-wise primitives, formal proof of correctness is very difficult and error-prone [14]. In contrast, LCM not only allows for operations of arbitrary complexity but also utilizes the meta-information inherent to these operations. This information is not only applicable to conflicts concerning the local data state but can also be employed for resolution processes on remote sites.

Similar to LCM, the model of eventual consistency [1,5] is employed by these approaches. Therefore, a globally consistent data state is not guaranteed at all times.

5 Conclusion

Consistency management is central to distributed document authoring and a prerequisite for providing the user with a globally synchronized and unintrusive environment. The combination of our advanced Compound Document System and the layered consistency management concept introduced in this paper allow for the comprehensive utilization of relational dependencies and metadata to achieve autonomous consistency management and intention preservation. Specialized functional modules based on the data type enable concurrent conflict detection and resolution, prevent the processing of unrelated information and provide the user with thorough information about a conflict in case manual conflict resolution is required.

The fine-granular data structure inherently supports the utilization of supplementary contextual information as well as of data explicitly and implicitly generated by the use of the collaboration environment. This allows for a responsive, reliable, and extensible system supporting the collaborator's efforts while limiting workflow interruptions through largely autonomous consistency management. We are currently preparing a study to assess further usability and performance aspects of the LCM approach.

References

1. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: Conflict-free replicated data types. In: Défago, X., Petit, F., Villain, V. (eds.) SSS 2011. LNCS, vol. 6976, pp. 386–400. Springer, Heidelberg (2011)
2. Sun, C., and Ellis, C. A.: Operational transformation in real-time group editors: issues, algorithms, and achievements. In: Proceedings of the ACM 1998 Conference on Computer Supported Collaborative Work (CSCW 1998), pp. 59–68, Seattle (1998)
3. Zheng, Y., Shen, H., and Sun, C.: Agile semantic conflict detection in real-time collaborative systems. In: Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), pp. 139–146, Baltimore (2009)
4. Sun, D., Sun., Xia, S., and Shen, H.: Creative conflict resolution in collaborative editing systems. In: Proceedings of the ACM 2012 Conference on Computer Supported Collaborative Work (CSCW 2012), pp. 1411–1420, Seattle (2012)
5. Saito, Y., Shapiro, M.: Optimistic replication. ACM Comput. Surv. **37**(1), 42–81 (2005)
6. Sun, C., Chen, D.: Consistency maintenance in real-time collaborative graphics editing systems. ACM Trans. Comput. Hum. Interact. **9**(1), 1–41 (2002)
7. Kirsch, L., Botev, J., Rothkugel, S.: The snippet platform architecture - dynamic and interactive compound documents. Int. J. Future Comput. Commun. **3**(3), 161–167 (2013)
8. Ter Hofte, G., Van Der Lugt, H.: CoCoDoc: a framework for collaborative compound document editing based on OpenDoc and CORBA. In: Proceedings of the IFIP/IEEE International Conference on Open Distributed Processing and Distributed Platforms, pp. 15–33, Toronto (1997)
9. Hudson, J. M., Christensen, J., Kellogg, W. A., Erickson, T.: "I'd Be Overwhelmed, But It's Just One More Thing To Do:" availability and interruption in research management. In: Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems, pp. 97–104, Minneapolis (2002)
10. Jambon, F.: Error recovery representations in interactive system development. In: Proceedings of the 3rd Annual ERCIM Workshop on "User Interfaces for All", pp. 177–182, Obernai (1997)
11. Nédelec, B., Molli, P., Mostéfaoui, A., Desmontils, E.: LSEQ: an adaptive structure for sequences in distributed collaborative editing. In: Proceedings of the ACM Symposium on Document Engineering 2013 (DocEng 2013), pp. 37–46, Florence (2013)
12. Oster, G., Urso, P., Molli, P., Imine, A.: Data consistency for P2P collaborative editing. In: Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work (CSCW 2006), pp. 259–268, Banff (2006)
13. Preguiça, N.M., Marquès, J.M., Shapiro, M., Letia, M.: A commutative replicated data type for cooperative editing. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), pp. 395–403, Montreal (2009)
14. Li, D., Li, R.: An admissibility-based operational transformation framework for collaborative editing systems. Int. J. Comput. Support. Collaborative Work (CSCW) **19**(1), 1–43 (2010)