Collaborative Processes Management: from Modeling to Enacting

Mamadou Lakhassane Cisse, Hanh Nhi Tran, Samba Diaw, Bernard Coulette, Alassane Bah

Collaborative Processes Management: from Modeling to Enacting

Mamadou Lakhassane Cisse
University of Toulouse Jean Jaures
IRIT Laboratory
Toulouse, France
mamadou.cisse@irit.fr

Hanh Nhi Tran
University Paul Sabatier
IRIT Laboratory
Toulouse, France
Hanh-Nhi.Tran@irit.fr

Samba Diaw
Cheikh Anta Diop University of Dakar
UMMISCO
Dakar, Senegal
samba.diaw@ucad.edu.sn

Bernard Coulette
University of Toulouse Jean Jaures
IRIT Laboratory
Toulouse, France
coulette@univ-tlse2.fr

Alassane Bah
Cheikh Anta Diop University of Dakar
UMMISCO
Dakar, Senegal
alassane.bah@ucad.edu.sn

***Abstract*—Collaborative work is a complex activity that involves several actors thus managing it is complicated. This article addresses some matters of controlling collaborative work in the point of view of process management. The underlying objective is to discuss the issues from modeling to enacting collaborative processes in existing process management systems. To overcome the discussed problems, we are developing a process management framework that integrates directly the concept of collaborative task into both modeling language and enacting mechanism. Moreover, we propose a flexible approach that enables users to choose dynamically a suitable way to enact their collaborative tasks.**

Keywords—collaborative processes, patterns of collaboration, process modeling, processes enacting, process management

I. INTRODUCTION

Nowadays, collaboration and teamwork are becoming a necessity in most of processes, especially in System and Software Engineering to develop complex products. Considering a task is the smallest manageable work-unit of a process, the collaboration can happen as coordination among various tasks to synchronize their progress. It can also happen as cooperation of multiple stakeholders inside a task, so-called collaborative task, to achieve a common goal. In this context, new methodologies and tools to manage collaborative processes may be needed.

Since the last two decades, many researches have been conducted in different communities on various aspects of collaborative works from the way of thinking and working, to the way of modeling, controlling and supporting [1]. This paper examines works in (Business) Process Management field to model and enact processes. We investigate the adequacy of current Process Management Systems (PMSs) in managing emergent collaborative processes. While PMSs provide support to coordinate well-structured processes with non-collaborative tasks, few propose mechanisms to really control collaborative tasks performed by several actors [2]. In this paper, our objective is to identify the open issues from modeling to enacting collaborative processes and suggest some improvements.

Suppose there is a defined process containing collaborative tasks as described in Section 2, we discuss in Section 3 the main issues in modeling and enacting such a process. Then, we attempt to identify the requirements of an efficient framework allowing to model executable collaborative processes and especially to easily deploy the modeled processes in a PMS. Section 4 presents our proposal to introduce flexibility at modeling and enacting time by an approach based on collaborative patterns to allow process actors choosing the most adapted way to perform and control their collaborative activities. Section 5 discusses some related works and Section 6 concludes this paper and outlines our work in-progress.

II. ILLUSTRATING EXAMPLE

In this section, we present a simple process "Review a Document" which will be used to illustrate our observations. This process is composed of two tasks: *Review Document* and *Modify Document*. The first task *Review Document* is performed by the role *Reviewer* to review the submitted document. It can be performed by several reviewers thus it is a collaborative task. The second task *Modify Document* is performed by the *Author* to deal with the annotations made by the different reviewers and modify the original document.

In this example, we consider the following work sequence relations between two tasks:

- *Finish2Start (FS)*: the first task must finish so that the second task can start.
- *Start2Start (SS)*: the second task can start if the first task has started.

Fig. 1 presents the two possible models (in the figure we combined the two work sequence relations – FS or SS, but each one of them consists of a distinct model) of the *Review a Document* process corresponding to the above relation between its two tasks. In these two cases, the process starts with a *Submitted Document* to be reviewed as an input of the *Review Document* task. This task produces an *Annotated Document* artifact which will be used in the second task *Modify Document* to produce a *Revised Document*.

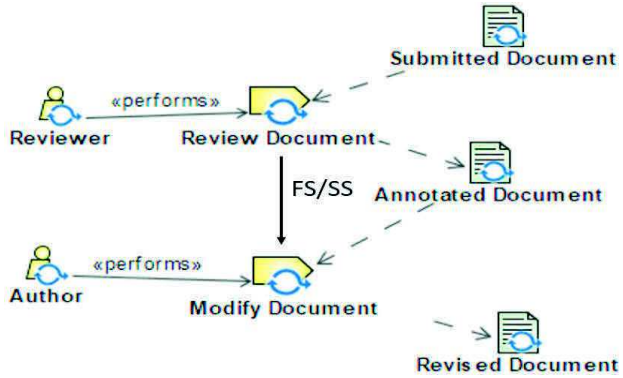

Fig. 1. Review Document showing the two cases Finish2Start (FS) and Start2Start (SS)

In Fig. 1, when the chosen sequencing is *Finish2Start* (FS), before starting, the task *Modify Document* must wait until the task *Review Document* is finished. The *Start2Start* (SS) sequencing means that the *Modify Document* can start as soon as *Review Document* have started.

As our first task can be realized by several actors, collaboration for this process has to be supported at two levels: coordination between the first task and the second task; cooperation among actors of the first task for its completion.

III. Issues of Collaborative Process Management

The most important objective of process management is to provide users a support to control the enactment of the example process, i.e. to coordinate the tasks of the participants to correctly progress the process. For that purpose, the process model must provide enough information to process engine so that it can decide when a task can start or finish.

Based on the illustrating example, this section discusses some issues from modeling to enacting a collaborative process. In this paper, we focus on the coordination and the data exchange of collaborative tasks, the communication between those tasks is out of scope of this work.

A. Ambiguities of collaborative process modeling

When using a PMS to enact a process model, a process instance will be created, i.e. the instances of its tasks will be created. The PMS must have enough information on the task instances and the relations between them to allow a correct control of the running process.

However, most of existing process modeling languages (PMLs) do not provide a clear semantics on how to instantiate a collaborative task. Consequently, from the process model in Fig. 1, there are several possible interpretations to establish the relations among task instances at enacting time. The interpretations can be defined by using the workflow patterns, as proposed in [3], describing various execution scenarios based on different ways to sequence task instances (control-flow patterns) and to exchange data between task instances (data patterns). Those patterns served as inspiration in [4] for Thuan et al. to define some collaboration patterns at modelling, instantiation or execution time.

Following, we use two patterns proposed from [4] to present different cases of the instantiation of the process models from Fig.1:

- *Duplicate in Sequence with Multiple Actors* (aka *sequential pattern*): this pattern is used when a task in the process is enabled after the completion of the previous task (in the same process). This pattern serves to construct a series of consecutive tasks where there is one input and one output and every actor produces a specific part of the final output.
- *Duplicate in Parallel with Multiple Actors* (aka *parallel pattern*): this pattern is used when a set of tasks are executed simultaneously and produce different outputs. It can be, optionally, followed by a merge of the different outputs.

In Fig. 2, we present two cases corresponding to the use of the sequential pattern. Both cases use the sequencing *Finish2Start* (FS) between the tasks *Review Document* and *Modify Document*. *Review Document* is a collaborative task; thus, it has several task instances at enacting time. That implies that every instance of *Review Document* must be linked with the task instance *Modify Document* with the FS sequencing.

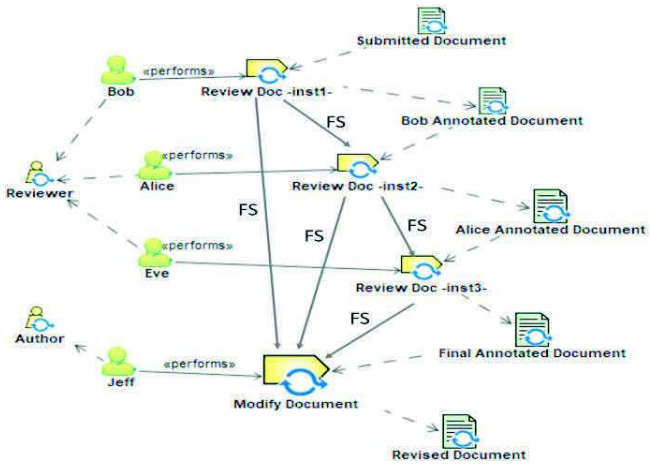

Fig. 2. Sequential case Finish2Start (FS) with FS between the Review Document instances

In the case where the sequencing between the instances of *Review Document* task are set to FS, to start *Review Doc. -inst2-* we must first finish *Review Doc. -inst1-* and so on. The difficulty in the use of this case lays in the fact that there is not much support in the choice of the first instance to execute. Also, what can drive the process manager to start with *Bob* in the first place? This points out the problem of sequencing among reviewers. Starting with *Bob*, what can drive him to start with the first instance instead of the second?

However, when the sequencing between the instances of *Review Document* are set to SS – meaning an instance can start as soon as the preceding one has started – the difficulty lays in the sharing of the different document between instances. Indeed, to start a task instance, the needed input might need to be in a defined state before it is manipulated in the current task instance. Otherwise, in the example of Fig. 2, we might have a situation where two task instances are manipulating the same document, which is not wanted.

Given the sequencing between instances are the only changes that occur in our example, for representation purpose, we are going to leave out the actors and data shared between instances.

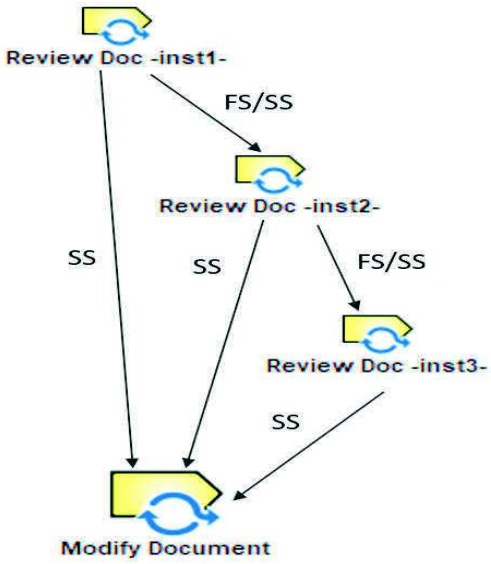

Fig. 3. Sequential case Start2Start (SS) with FS or SS between the Review Document instances

Fig. 3 presents the cases corresponding to the *Start2Start* sequencing between the two tasks of our process. Thus, every instance of *Review Document* is linked to *Modify Document* with SS.

In Fig. 3, with SS sequencing, *Modify Document* can start as soon as all the three instances of *Review Document* have started. However, when we have FS between the instances of *Review Document*, semantically, *Modify Document* cannot start until the last instance of *Review Document* has been started.

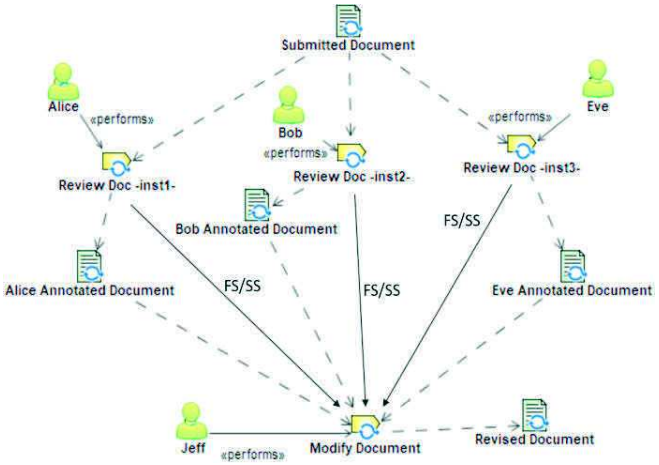

Fig. 4. Parallel case with FS or SS between Review Document instances and Modify Document

Considering Fig. 4, the pattern does not semantically allow to start the *Modify Document* as soon as one instance of *Review Document* is finished (FS) or has started (SS) even though it is a common scenario in real life (even with the *Review Document* example).

If we consider one case (called situation A) where the second task, *Modify Document*, has to be collaborative, thus having multiple instances. In this situation, on both scenarios (FS and SS) from Fig. 2, each instance of *Review Document* would be linked (with FS or SS) to every instance of *Modify Document*. That means *Review Doc. -inst1-* would be linked to two instances or more of *Modify Document*. That situation applied in Fig. 4 implies that every output from each instance of *Review Document* to be taken as an input in each instance of *Modify Document*. This case is represented in Fig. 5. We omitted the work sequences between the two main tasks and also the actors on purpose to better represent the links with inputs and outputs. With regard to this situation, there is no defined shared data between the instances of the second task which makes it difficult to really finish the enactment unless adding a *Merging* task which will change the initial process.

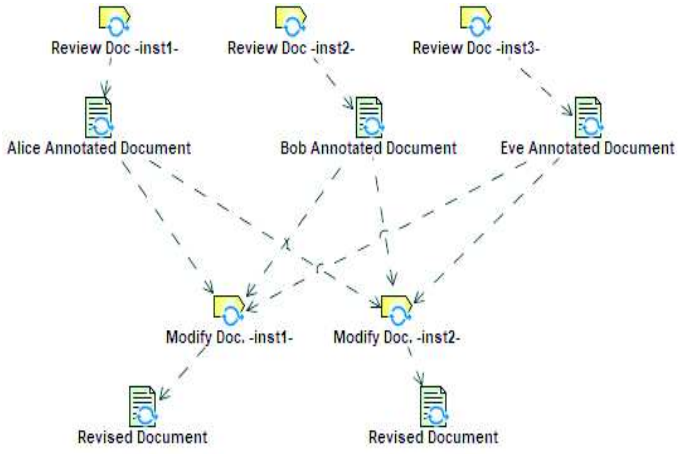

Fig. 5. Representation of Situation A without the work sequences

B. *Inadequacy of collaborative process enacting*

When enacting a process, many collaborative situations as described by the Workflow Patterns [3] can happen. As shown in the evaluation on the website of Workflow Patterns [17], not all of these patterns are supported by existing PMSs, especially the patterns concerning multi-instances and shared data as described in the given example.

One reason of this lack of support is that the implementation of the complex patterns is not easy. It requires the PMS to consider not only the relations between tasks in the model but also the relations of task and data instances emerging at enactment time. When a task executes multiple times, suppose that the relations between its instances are clearly defined, the process engine has to examine all the task's instances to make a decision on a state transition of the task. For example, in the situation of Fig. 3, we cannot start *Modify Document* until the last instance of *Review Document* has been started. This implies a fine-grained control on all of the predecessors of the task to execute (*Modify Document*) but also a control on the links between the instances of the previous task (*Review Document*).

When a multi-instances task with shared data is enacted but not all instances are created at the same time, an issue that can happen is whether the values of data elements are set for all execution instances at the initialization of the multi-instance task or whether they can be fixed after this occurs but prior to the actual invocation of the task instance to which they relate. Another similar issue can arise when we finish a multi-instances task with different outputs produced by its instances. The process engine may deal with a new task to merge the outputs. Considering the situation depicted in Fig. 5, a merge of the *Revised Document* outputs is necessary. This brings up the question of dynamically introducing a new task during execution.

Even if a PMS implements some proposed patterns, it provides a rigid way to support the way that a collaborative task can be enacted, based on how the pattern is implemented. In reality, the way used to enact a collaborative task should be flexibly chosen according to the actual organization of the participants and the nature of the processed data.

IV. TOWARDS A FLEXIBLE PROCESS MANAGEMENT SYSTEM FOR COLLABORATIVE PROCESSES

To address the aforementioned issues, we have proposed an approach to manage collaborative software processes which aim at providing (1) a process modeling language (PML) equipped with special constructs to describe collaborative patterns; (2) an operational semantics enabling execute the selected collaborative patterns; (3) a process management system supporting flexibility by late binding so that users can choose, at enactment time, appropriate collaborative patterns corresponding to their organizational model. Additionally, our approach allows to dynamically add new instances of a running collaborative task, new actors for the execution and part of artifacts and also delete them if necessary.

For supporting our PML execution, we have also proposed a process engine prototype, called CPE (Collaborative Process Engine). Fig. 6 below shows the general architecture of our collaborative process engine prototype. The project manager interacts with the prototype to execute the different possible actions. The component *process models* holds the different models of processes that the prototype needs to execute. They correspond to the process modeling language equipped with special constructs to describe collaborative patterns. Different *listeners* are available inside the process engine. They allow to track the evolution of state of the different task instances that are being enacted.

The process engine can access and store the different task instances through the database management system represented by the *instances store*. The necessary *resources* and *artifacts* for the execution of the task instances are available through external systems such as the *artifacts management system* and the *resources management system.* They are made available for the process engine through copy requests.

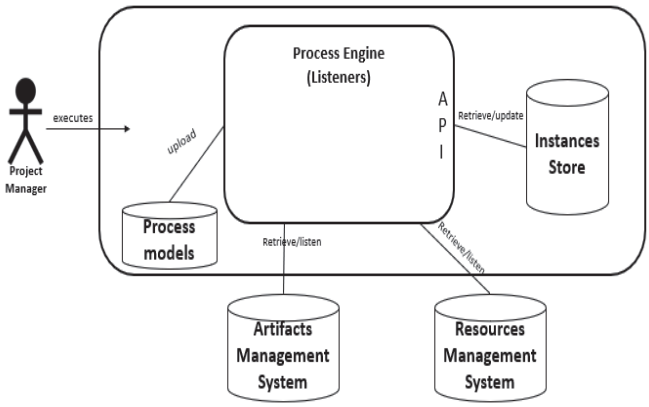

Fig. 6. General architecture of our CPE prototype.

Mainly our prototype allows project teams to upload a process model and then generate all the activities and task instances to be enacted. Given project development is collaborative, the project manager can choose how many instances of every collaborative task he wishes to instantiate. This choice can be dependent on the complexity of the task, the duration of the project and/or other factors organizational-based. He can also choose the collaboration pattern to apply for each collaborative task based on the project characteristics. Our prototype is working with a set of patterns based on parallel or sequential execution at first. The parallel pattern means that the instantiated tasks are executed simultaneously followed by a merge of the different outputs. The sequential pattern means that every task instance must wait for the previous task to be finished (for the Finish2Start – FS work sequence). Those task instances are assigned to actors with inputs and expected outputs. The resource assignment allows

every stakeholder to know his own tasks and the sequencing between the others. Finally, every actor will be able to graphically visualize the states of his task instances and next steps to take.

V. Related Works

In order to provide an efficient support for collaborative works, there are many works in CSCW (Computer-supported Collaborative Work), CE (Collaboration Engineering) and KM (Knowledge Management) addressing the problem of understanding how people collaborate. They have proposed solutions to design collaborative processes, to support sharing, communication and negotiation in collaborative situations. However, these research directions are out of scope of our paper. We discuss in this section only the works dealing with modeling and enacting collaborative processes.

Several works identify the recurrent patterns of collaborative processes. In [3], authors propose a series of workflow patterns for workflows functionalities. References [4][5] propose both collaborative patterns even though [5] only focuses on representing collaborative situations at modeling time and not real situations at enactment time. Other works propose special constructs to model collaborative activities and protocols [6][7]. In [8], authors present an extension of BPMN modeling language including notation to handle occurring tasks in collaborative processes. References [12][13] present a design approach as ThinkLets concepts to achieve repeatable patterns of collaboration. In [14], authors have developed a visual language to model collaboration protocols. That language has been integrated in a model-driven software development method. Some standard process modeling languages as SPEM [15] and BPMN [16] provide also notations to model certain collaborative aspects of processes, such as activities which can have multiple instances at enacting time, messages exchanged among activities, etc. However, the cited modeling solutions focus on communicating collaborative processes, thus produce non-executable process models.

On enacting collaborative processes side, most of BPM and PMS systems provide support for coordinate intra-process activities, some of them providing also inter-processes orchestration. However, the above systems do not propose efficient support to control collaboration at task enactment level, i.e. task instances inside a collaborative task. As discussed in [2], there are CSCW systems providing support for process modeling and execution as [9] and there are BPM systems providing support for collaborative activities as [10][11]. The main lack of these systems is their inability to intertwine collaborative modeling and enacting of processes.

VI. Conclusion and Works In-Progress

Our article mainly addresses issues in collaborative processes management and limits in support by the existing PMSs. In this paper, we underlined scenarios occurring during a collaborative process execution. One of the most important issue we discussed is the lack of clear semantics by existing PMLs on how to instantiate a collaborative task. Thus, our contribution is to propose an approach to manage collaborative processes with a flexible PMS supporting unexpected situations at enactment time and collaborative tasks performed by several actors.

We are now continuing to develop the approach proposed in section 4. Concretely, we are refining our process modeling language to integrate the executable concepts dedicated to collaborative processes. In order to ensure a real cooperation between stakeholders, the operational semantics of such concepts will be defined clearly to provide more control over the transition from a task to another at enacting time. Adopting the model-driven approach, we use a metamodel to define our process modeling language. The operational semantics of the language will be defined using the state machines associated to each executable concept. We hope that the proposed language will allow seizing all the steps in the enactment of collaborative processes.

We are also investigating more control-flow and collaboration patterns from [3] in order to provide more choice in the enactment of a collaborative task. Basically, those patterns will represent the different possible scenarios during the execution of a collaborative process.

References

[1] R. Briggs, G. Kolfschoten, V. Gert-Jan, and D. Douglas, “Defining key concepts for collaboration engineering,” AMCIS 2006 Proceedings, p. 17, 2006.

[2] Hanane Ariouat, Eric Andonoff, Chihab Hanachi. Do Process-based Systems Support Emergent, Collaborative and Flexible Processes? Comparative Analysis of Current Systems. Procedia Computer Science, Elsevier, 2016, vol. 96 (n C), pp. 511-520.

[3] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow patterns,” Distributed and parallel databases, vol. 14, no. 1, pp. 5–51, 2003.

[4] T. T. Vo, B. Coulette, H. N. Tran, and R. Lbath, “Defining and Using Collaboration Patterns for Software Process Development.pdf,” presented at the International Workshop on Cooperative Model Driven Development (CMDD 2015) within the 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2015), Angers, France, 2015, pp. 557–564

[5] Jacques Lonchamp. Process model patterns for collaborative work. 15th IFIP World Computer Congress - Telecooperation'98, Aug 1998, Vienna, Austria, 12 p, 1998.

[6] K. A. Kedji, R. Lbath, B. Coulette, M. Nassar, L. Baresse, and F. Racaru, “Supporting collaborative development using process models: a tooled integration-focused approach. Journal of Software: Evolution and Process, vol. 26, no. 10, pp. 890–909, Oct. 2014.

[7] I. T. Hawryszkiewycz, “A metamodel for modeling collaborative systems,” Journal of Computer Information Systems, vol. 45, no. 3, pp. 63–72, 2005.

[8] P. Antunes, V. Herskovic, S. F. Ochoa, and J. A. Pino, “Modeling Highly Collaborative Processes,” in Proceedings of the IEEE 17th International Conference on Computer Supported Cooperative Work in Design, 2013.

[9] Dustdar S.: Caramba A Process-aware Collaboration System supporting Ad hoc and Collaborative Process in Virtual Teams. Distributed and Parallel Databases, vol. 15, 2004, pp. 45□66.

[10] Charoy F., Guabtni A., Faura M.: A Dynamic Workflow Management System for Coordination of Cooperative Activities. Business Process Management Workshops, Vienna, Austria, September 2006, pp. 205-216.

[11] Mundbrod N., Beuter F., Reichert M.: Supporting Knowledge-intensive Processes through Integrated Task Lifecycle Support. Int. Enterprise Distributed Object Computing Conference Adelaide, September 2015, pp. 19-28.

[12] G.-J. de Vreede and R. Briggs, “Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Tasks,” in 38th Annual Hawaii International Conference on System Sciences, 2005.
[13] R. O. Briggs, G.-J. D. Vreede, and J. F. N. Junior, “Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems,” vol. 19, no. 4, pp. 31–64, 2003.
[14] JesúS Gallardo, Crescencio Bravo, Miguel A. Redondo, and Juan De Lara. 2013. Modeling collaboration protocols for collaborative modeling tools: Experiences and applications. J. Vis. Lang. Comput. 24, 1 (February 2013), 10-23.
[15] S. OMG and O. Notation, “Software & Systems Process Engineering Meta-Model Specification,” OMG Std., Rev, 2008.
[16] Object Management Group (OMG), Business Process Model and Notation (BPMN), Version 2.0, 2011.
[17] Workflow Patterns Website, http://www.workflowpatterns.com