A Framework for Exploration in the Realm of Collaborative Modeling and Design

EBRAHIM BAGHERI
e.bagheri@unb.ca
Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada

Abstract. The high complexity and diversity of today’s design projects requires the participation of various modelers and designers. The participants can influence the design process by sharing their perspective, expertise and resources. The involvement of various designers is often known as collaborative modeling and design. A collaborative modeling environment can encompass various geographical or organizational boundaries. In this paper, we provide a multidimensional framework to study various aspects of this important issue. The framework addresses both model-oriented and artifact-neutral collaboration models and enumerates their features. The paper introduces and classifies several relevant applications based on the proposed framework and suggest some directions for future work in the domain.

Keywords: Collaborative Modeling, Distributed Design, Coordination, Cooperation

1. Introduction

The modeling and design process of any kind of system naturally commences with the high-level description of the requirements and needs of the participating stakeholders and gradually proceeds towards a finer-grained solution. This process engages with the formalization of abstract coarse concepts as detailed concrete formal representations [1]. In complex systems, design requires opportunistic planning and re-planning in order to adjust to both internal and external parameters. Furthermore, in many cases the initial requirements are incomplete and in occasions ambiguous; therefore, the problem and solution representations are under incremental re-definition and evolution throughout the life-cycle of the system [2]. For these reasons, many researchers consider modeling and design as messy or wicked problems [3]. The translation of high-level needs into a proper concrete form necessitates the exploration of a complex space of alternative solutions [4].

Most traditional and many of the current design processes employ a sequential strategy. In this approach, various designers involved in the process, collaborate through a progression chain, where the output of the activities of one group of designers is the input to the other. This ‘throwing-over-wall’ [5] approach has shown to be weak, fragile, tightly-coupled and often requiring numerous iterations that entails high cost, time consumption and therefore, the inability to examine different design alternatives. A survey conducted in the United States revealed that seventy percent of the people involved in the modeling and design divisions of their company believe that sequential engineering products are insufficient and that they will need collaborative design software [6]. Despite the growing need for collaborative modeling tools, only recently has this domain gained attention. Many
of the developed platforms mainly focus on features such as messaging, approval forms, forums, multi-user editors [7], video-conferencing and workflow tools [8]. The main goal of these artifacts is to enhance communication amongst the participants of the design process. However, communication is only a subset of the constituting issues in collaboration [9].

The requirements engineering community has addressed the issue of collaborative modeling through the introduction of viewpoints. A viewpoint is an encapsulation of partial information about a system’s requirements [10]. Viewpoints facilitate the employment of information coming from different sources or perspectives. They can be used to structure the requirement elicitation and definition process and also to encapsulate different system models [10]. The application of viewpoint-based requirement engineering methods focus on both viewpoint analysis which investigates the consistency and correctness of the models developed within each viewpoint and viewpoint reconciliation that addresses the communication amongst the participating viewpoints [11]. VORD, Preview, Dealscribe, and Synoptic are a few of such models [10].

Similarly, researchers in the CAD community have sensed the need for collaborative technologies. Current CAD problems incorporate considerable amount of complexity and demand significant design knowledge that is unlikely to be found in a single group of designers [12]. Therefore, industrial scale projects frequently assemble teams with various fields of expertise and skill, from different geographical locations and sometimes from various firms to achieve their goals [8]. Based on this setting, collaborative CAD systems require both distribution and collaboration. Distribution will provide physically dispersed teams with the possibility to work together to achieve a common goal, while collaboration coordinates the interactivity of these teams towards the ultimate design goal. Collaborative CAD systems can be organized as either horizontal or hierarchical by nature [6]. A horizontal CAD system focuses on the collocation (physical or virtual) of a design team to carry out a specific complex design problem. In this approach, all members of the design team are working on the same aspect of the problem domain. In contrast, hierarchical CAD systems emphasize on developing proper communication channels between upstream design and downstream manufacturing activities. This would require specific principles and methodologies to link assorted concurrent engineering tools each emphasizing on a different aspect of the problem domain [5].

Analogous to the above classification, Falzon [13] categorizes collaborative design models based on the nature of the shared design goals into distributed design and co-design. In distributed design, each group of modelers has its distinct set of design responsibilities. These responsibilities are defined such that they have the least degree of interdependency so that the design groups share the smallest amount of resources. Collaboration in such situation will only need coordination and management among the design groups that are developing interdependent products. It should be noted that model construction is not a completely hierarchically decomposable activity; therefore, developing totally independent design teams in the distributed design approach is nearly infeasible [14]. DOME [15] is a representative of the distributed design models. The DOME framework asserts that
multi-disciplinary problems can be decomposed into different sub-problems. The existence of such modular sub-problems distributes the overall complexity of design and the responsibility of the designers and facilitates the re-use of the modeling elements [12]. On the other hand, in co-design, design teams contribute to the attainment of the same goals based on their own perception of the design problem. In this model, collaboration would require complex methods for inter-participant negotiation, building common grounds and coordination, inconsistency and conflict identification, and resolution, and much more [3].

Computer-supported cooperative work have traditionally defined various aspects of collaboration through Johansen’s two dimensional time-space matrix [16]. The matrix depicts four spaces namely, face to face interactions (e.g. Tech-meeting rooms), remote interactions (e.g. Desktop sharing conferencing systems), ongoing tasks (e.g. Group bulletin board displays), and coordination (e.g. Asynchronous conferencing bulletin boards). While the classification is quite inclusive with regards to a/synchronoussness and location of activities, it does not explicitly address the participants. Chen et al [17] have extended Johansen’s matrix by adding a third dimension: working groups. This dimension shows whether the participants of the cooperative work are from the same background, company, and if they are working on the same aspects of the problem domain.

In this paper, we investigate various aspects of the collaborative modeling and design process. We focus on two important lines of research namely, model-oriented and artifact-neutral collaboration. In the first class, we gather various techniques from requirement engineering, model management, ontology and database schema engineering and focus on collaboration by centering around the final desired output artifact. The activities and roles involved in the collaborative design process are hence defined by the effect that they will have on the design product. In the second class, we study collaborative modeling in an artifact-neutral setting. In this way, we can investigate the applicability of general purpose argumentation, negotiation, and facilitation methods. We will also investigate models of activity coordination in this class.

2. The Dimensions of the Framework

In a survey on collaborative tools with an emphasis on the user effort required for collaborative software development, Sarma [7] has identified three critical strands of research that crosscut all aspects of collaborative software development. These three strands are communication, artifact management and task management. In this section, we generalize this classification so that it encompasses other fields of collaborative design and modeling such as architectural, and mechanical modeling and design. We make a distinction between collaboration in an artifact-centric setting and a more relaxed collaborative scenario and describe the models under which these two schemas can be collectively employed. The required communication infrastructure for collaboration including the necessary technologies and models are also studied. The framework will further introduce some very interesting aspects that can be employed in collaborative design under related aspects. For each di-
The framework dimensions are namely, Model-oriented Collaboration, Artifact-neutral Collaboration, Communication Infrastructure, and Related Aspects. Figure 1 shows the proposed framework dimensions, and their sub-dimensions.

2.1. Model-oriented Collaboration

In most (if not all) modeling and design projects, the participants are working towards the development of a unique design artifact. The final product artifact, be it an outcome of a mechanical, electrical, software, hardware, or other engineering fields design process should conform to a set of high-level standards. These standards are known as meta-models in software design. Regardless of the target modeling domain, meta-models contain descriptions of the structure and semantics of the domain data. Therefore, through the instantiation of meta-model elements a final design product can be created. The elements of the meta-models can be any entity such as classes, concepts, pipes, wires, or other building block of interest. The work in the field of model engineering has focused on the development of utilities for model construction, transformation, integration, validation and etc [18]. Epsilon [19] and ModelBus [20] are some examples of model management frameworks that provide suitable model manipulation operators. For instance, Epsilon functions over EML, a rule-based language used for merging diverse meta-models
and technologies. Some other frameworks employ various types of reasoning and inference to manipulate model elements [21].

In a collaborative modeling and design setting where the designers are working towards developing a shared conceptual model, not all of the designers are focusing on the same aspect of the problem; therefore, the high-level meta-models that they employ may differ. Differences may also be caused due to the employment of different modeling utilities. In order to be able to create a final unified model, these different products should be transformed into a common meta-model (transformation). After transformation, specific tools for finding the correspondences between the translated models (mapping) are needed so that the models can be integrated (integration). The integration of models developed by different groups of designers will certainly contain discrepancies and possible conflicts that need to be properly dealt with (inconsistency management). The developed designs should also be archived properly (version control) so that the reasoning behind various design decisions can be tracked back and identified later in the design procedure (traceability). This process can be seen in Figure 2.

In the following sub-sections, we will study various approaches and techniques that have been employed for addressing each of the issues in model-oriented collaboration.
2.1.1. **Model Transformation**  The simplest form of model-oriented products are software code. Software code are considered to be models since they conform to a set of syntactic and semantic rules. They are most often transformed into parse trees, which allow syntactic and semantic correctness analysis [21]. Attributed graphs have also been used for representing design artifacts. Here, the only entities that are used are nodes and edges. Nodes and edges have a label, edges have additional source and target node, and they can both carry domain specific constraints. For instance, in a programming language a node can have a type which denotes the type of that certain variable.

Logical formalism have are also appealing formats to serve as an underlying representational models. Mens et al [22] employ first-order logic in the form of Prolog predicates to both manage and manipulate UML models. As an example, ‘node(h,Bank,class, [version(1.0)]).’ declares Bank as a node in a class diagram and ‘cancellation(Money, class).’ would remove a node with label Money and type class from the class diagram. Furthermore, UML models have a basic structure similar to trees in which each element can have a child element within itself. These models are not complete trees due to cross-references; however, they can be modeled as ordered trees. UML models have also been modeled as simple graphs, but the employment of a tree structure is more desirable since algorithms for manipulating trees are more efficient than that of graphs [23].

Ontologies have been transformed into graph-like representations, as well. AnchorPrompt [24] treats an ontology as a graph with classes as nodes and slots as links. In another approach, a translation of an ontology into a graph has been performed using an import filter that understands the basic definitions of OWL/DAML [25]. In the MAFRA [26] framework, the authors translate all input ontologies into the RDF(S) format. The IF-Map [27] framework converts different ontological definitions into Prolog predicates. In all of the above transformations, the main aim is to reduce the overhead of understanding, parsing and manipulating various input models coming from different design groups by only manipulating one single model formalism. Model transformation is also known as normalization. It is important to note that transformations should be behavior and property preserving and no additional information should be added or removed from the source models through this process.

2.1.2. **Model Mapping**  Mapping is a critical operation in various application domains where different underlying representations are employed by the participants. It functions over two or more input models, each consisting of a set of discrete entities and produces as output the interrelationship between the entities of these models [28]. This operation can be interpreted as relating the vocabulary of various models that share the same domain of discourse in such a way that the logical structures as specified by the high-level meta-model are respected. structure (behavior and property) preserving mappings are known as morphisms. Let $M = (S, A)$ represent a model, where $S$ is the signature describing the model vocabulary and $A$ is a set of axioms depicting the intended interpretation of the application vocabulary with respect to the domain of discourse. A total model mapping from
A FRAMEWORK FOR EXPLORATION IN THE COLLABORATIVE ...

$M_1 = (S_1, A_1)$ to $M_2 = (S_2, A_2)$ is a morphism $f : S_1 \rightarrow S_2$ of model signatures, such that all interpretations satisfying $A_2$ must also satisfy $f(A_1)$. In many cases developing a total model mapping is not feasible; therefore, a weaker notion, partial model mapping, is required. A partial model mapping from $M_1 = (S_1, A_1)$ to $M_2 = (S_2, A_2)$ is a total morphism from $M'_1 = (S'_1, A'_1)$ to $M_2 = (S_2, A_2)$, where $M'_1$ is a sub-model of $M_1$ i.e. $S'_1 \subseteq S_1$ and $A'_1 \subseteq A_1$ [25]. A strict derivation of partial model mapping is defined as right (left) outer model mapping which ensures that every element of $M_1$ (or $M_2$) is referenced by the partial morphism function $(f)$ [29].

Based on the definition developed in [28], a model mapping function $(f)$ consists of a discrete set of mapping elements as a 5-tuple $(id, e, e', n, R)$ where $id$ is the unique identifier of the mapping element, $e$ and $e'$ are the source and target elements being mapped, $n$ is a confidence measure in the correctness of the mapping element, and $R$ is the mapping relation holding between $e$ and $e'$ ($R = \{=, \subseteq, \perp, \sqsubset\}$). In most cases mapping functions are reduced to mapping relationships only requiring the model mapping function to specify mapping relations of type equivalence [25]; therefore, model mapping is a collection of binary relations between two or more models.

Some model mapping techniques employ instance data to mine the relationships between different models [30]. This type of model mapping is possible in domains where instance data corresponding to each model exists such as database schemas, but they are not so useful in domains like UML diagram development since instance data cannot be generated for software models. In these domains, schema-level model mapping is required. Schema-level mapping techniques can be classified into element-level and structure-level. Element-level techniques employ the information of each single model entity to find a relevant correspondence. Heuristics such as exact, and partial name matching, levenshtein name matching, entity type comparison, synonym matching, prefix, suffix, and edit distance matching, domain-specific information matching are some of the simplest forms of element matchers [31]. Some more sophisticated language-based techniques employ natural language processing (NLP) methods to identify morphological properties of the entities. Tokenization, lemmatization, and elimination are some of the NLP methods that have been widely employed.

In some more sophisticated techniques for matching graph-based models the location of the entity in the graph has been employed to identify mapping relationships. Bounded path matching takes two paths with links between classes defined by the graph structure. It compares the entities based on their position in the given graph and finds similarities [32]. Furthermore, in super/sub-concept rules, if a correspondence between entity children or parents are found, the similarity can propagate to the entity itself [28]. In the similarity flooding technique, once a relationship has been determined, the similarity will gradually spread to adjacent nodes in the graph. The degree of similarity will however be adjusted based on the degree of entity closeness.

In structure-level model mapping techniques segments of a model are considered as one fragment. The fragments are then compared with each other in order to
find similarities. Fragments are usually selected based on syntactic or semantic correspondences. Structure-level matching is more sophisticated than element-level matching since it involves the analysis of a group of elements as opposed to single entities. To simplify this process, corpus-based models are employed where previous matches are archived in a mapping knowledge base. The knowledge base is incrementally completed as new mappings are made. Using these previous matches, new models are only compared to the structures in the knowledge base. If two fragments match the same fragment in the knowledge base, they are considered to be similar [33]. The corpus-based (AKA re-use oriented, alignment re-use [34]) mapping approach is promising since many of the models developed in the same domain of discourse share many similarities. For instance in e-commerce, structures used for customer information, product cataloguing, invoice management, purchase orders and many others are very similar in different systems and can be easily identified and matched using a corpus-based matcher [29]. Ontology-mediated model matching is also close to corpus-based mapping. In this approach, the models to be mapped are compared against an upper ontology, and correspondences are identified [33]. Finding similarities in models that contain a notion of time or sequence are more complex in nature. Nejati et al [35] have proposed a match heuristic for state chart diagrams based on both static and behavioral properties of these diagrams. In their model, the behavioral matching technique is a derivation of deciding bisimilarity between state machines. Since bisimilarity is a recursive notion it can employed in forward and backward modes. Two states are forward (backward) bisimilar if they can transition to forward (backward) bisimilar states through identical transitions. Some other model mapping techniques such as GLUE [36] which uses probabilistic measures and machine learning techniques for finding correspondences, MAFRA [26] that uses semantic bridge for interrelating two models, and AnchorPrompt [24] that analyzes the paths in the model graph to identify similarities have been proposed in the related literature that show promising performance.

2.1.3. Model Integration Model integration (merge) is involved with the fusion of multiple similar models developed by different modeling groups, in order to obtain a final unified view of all the models. The integrated model should be a replicate-free representative of the initial models; however, not all behavioral and properties of the original models are preserved in the unified model due to possible inconsistencies and conflicts. Merging of several models can be built upon the similarity mappings between the models. Pottinger and Bernstein [37] have proposed a merge operator that produces a unified view over two initial models based on given correspondences. The operator respects several important semantics, namely: element, equality, relationship, similarity, and property preservation, meta-model constraint satisfaction, extraneous item prohibition, and value preference. In a similar vein, Sabetzadeh and Easterbrook [38] have introduced a merge procedure based on structure-preserving maps for depicting model mappings and annotated graphs for addressing incompleteness and inconsistency. In this approach, models are formalized as categories and connectors as functors. Furthermore, In the mapping model based on bisimilarity, models are merged based on the identified mappings. In the
model, similar states and transitions are added to the state diagram, while dissimilar entities are attached to the model annotated with guard conditions. The guard conditions show how conflicting states and transitions have been described by the participants [35].

Model merging has also been described through belief merging operators [39]. In this approach epistemic states have been used to represent participants’ preference ordering over the proposed models. Models are hence merged based on the described model valuations in each participants’ epistemic states. In another attempt partially specified systems are modeled through Modal Transition Systems (MTS) [40]. The merge of these models are achieved based on observational refinement. The authors show that merging consistent models under this framework should result in a minimal common refinement.

Taking a different perspective, some merge algorithms first calculate the difference of the input models and based on the identified differences generate the final merged model. The XDiff algorithm [41] processes all the elements of the structured input documents in both bottom-up and top-down forms and generates a hash key for each entity. The hash keys are then used to find correspondences and differences between the input models [42]. Other models like the tree-to-tree model [42] can be used to identify differences between the models that can take the form of a tree. Similarly, UMLDiff is a domain specific structural differencing algorithm that produces a tree of structural changes which describes the differences between two design models [42]. JDiff is also a differencing algorithm that benefits from a graph structure and employs hammocks for its comparison of the models [43]. In the merging algorithms based on differencing, once the differences are identified, they are checked to see if they do not cause conflicts or inconsistencies (either syntactic or semantic). The differences that are safe are inserted into the final model and the conflict-causing differences are classified and reported for expert inspection and decision making.

Other related work such as FCA-merge which employs formal concept analysis, lattice exploration and the information extracted from within relevant documents related to the designed models can be found in the literature [25]. Text-based merging techniques are also popular methods that merge two or more versions of a textual document. Line-based merging for instance, is one of the most promising models of this approach [21].

2.1.4. Inconsistency Management Inconsistencies and conflicts often arise in the collaborative modeling and group decision making processes [1]. Conflicts may occur between the stated constraints, perceived requirements, resource consumption needs, or even designated priorities of the participants [44]. Discrepancies can delay development, and hence increase the development cost, and failure rate of the process. The maintenance of models containing inconsistency is also difficult; however, inconsistencies can point to existing conflicts between the perception of the participants, specify the aspects of the model that need further exploration, and facilitate the identification of alternative solutions [45]. For these reasons, conflicts and inconsistencies should be handled in a controlled fashion. Currently, many of
the solutions benefit from face-to-face meetings to understand, identify and possibly resolve any inconsistencies [1].

From a model-oriented perspective, conflicts can be categorized into three distinct classes: representation conflicts, domain constraints conflicts, and meta-model constraints conflicts. Representation conflicts occur as a result of describing similar domain concepts with a different syntax. Domain constraints conflicts are caused by a violation of a domain specific limitation, and finally, meta-model constraint conflicts appear when a basic definition in the modeling meta-model is violated. Meta-model constraint conflicts are usually the simplest form of conflicts that can be detected in a fully automated manner [37]. In a distributed requirement engineering setting, conflicts can be classified as either missing, inconsistent, or discordant [46]. A missing requirement conflict emerges when stakeholder A has defined a requirement while stakeholder B has not. Inconsistent requirements are those requirement specifications that although address the same concept, the specifications for the concept are not fully aligned. Discordant requirement specifications are a result of either a different interpretation or dissimilar prioritization of the same specification.

A comprehensive framework for handling and managing inconsistencies should be able to identify the formerly mentioned discrepancies, diagnose and track the roots and causes of the issues, and resolve them through inconsistency resolution policies [45]. For instance, the framework should be able to discover the source, cause, and impact of any inconsistency in the developed models. Based on these information proper resolution strategies and actions should be selected. The complications of industrial models are usually so high that fully automated strategies may not be feasible to design; therefore, resolution actions are mostly classified into changing and non-changing. Changing actions operate on the designed models to automatically (fully or partially) remove the source of inconsistency and resolve the issue, while non-changing actions report the discrepancies to higher level modules or human inspectors [45].

Changing actions are most often restricted to a specific modeling domain, and model dependant features are adopted for resolution. These actions are closely related to the implemented resolution strategy. Edwards [47] introduces four attractive conflict resolution strategies, namely quantum uncertainty strategy, explosion strategy, recursive acceptance strategy, and promotion strategy. In the quantum uncertainty strategy inconsistent entities are allowed to co-exist. All of such entities are labeled as uncertain and a degree of uncertainty is attributed to them. The uncertainty value is updated each time the models change. The explosion strategy computes all the possible courses of action that lead the design to a steady and conflict-free state. This strategy is computationally expensive in cases where too many possible solutions exist. In the recursive acceptance strategy, the designers can iteratively resolve the conflicts through what-if scenarios. The promotion strategy is a conflict avoidance strategy in that it evades conflicts by only permitting safe operations and promotes conflicting operations into safe operations.

Resolution strategies can be also studied based on the attitude of the participating designers [44]. The designers can adopt a constructive/cooperative strategy
where negotiation and education is preferred. On the other hand, they may choose a competitive strategy which contains coercion and competition. If such strategies cannot lead to a safe state, a third-party resolution strategy where outside forces are employed can be implemented. Some of the third-party resolution possibilities are to use of terminological resources such as ontologies or thesauri, to employ prototyping systems, store individual design rationales, relax some of the defined model requirements and constraints, and prioritize the expressions of the participating designers [48].

In the AGORA framework [46], preference matrices are employed to identify any discordances between the different designs. Using these matrices and based on the variances of the values in the rows, columns, and diameters of the matrix, vertical and diagonal variances between the expressions of the participants are identified. Vertical variance depicts the degree of mutual understanding among the participant, while diagonal variance represents the similarity of specification preferences of the participants. In the χbel framework [49] a form of para-consistent logics called Quasi-Boolean logics has been employed to identify the sources of inconsistency. As opposed to using classical logical formalisms, para-consistent logics are utilized since some degree of inconsistency is desired to be tolerated in this framework. χbel incorporates a model checker, χcheck to verify the developed models against the constraints defined in temporal logic. Other models in the literature such as the use of graph partitioning methods to reduce interdependency of model entities to gradually resolve conflicts and inconsistencies can be found [21]. In summary, discrepancy resolution is a difference minimization process through which the model variations are negotiated and an agreed model is gradually developed [42, 45].

2.1.5. Version Control A version control system allows developers to maintain multiple copies of their developed models in an organized manner with the ability to incrementally version and number each of the model editions. These systems provide their users with the possibility to retreat to a previous version of their model, if conflicts arises under a cooperative inconsistency resolution strategy. A version control system is often set up as a centralized repository of models; however, other models such peer-to-peer, open and replicated models of version control systems have also been deployed [50]. Typically, a version control system, regardless of its deployment architecture, provides its users with a basic set of functionality to commit, check-out, merge, update, and differentiate various models.

Depending on their version control mechanism, these systems can be regarded as pessimistic or optimistic [21]. In a pessimistic version control system, all the modelers working on the same model, are obliged to manipulate the same copy of the model and parallel manipulation of the same model is prohibited based on locking methods. This pessimistic strategy endeavors to maintain a consistent version of the model at all times. In the optimistic strategy, each of the modelers manipulating the same model, possess their own personal copy of the model. The optimistic approach is much more flexible, but the models that it maintains may lack consistency in many periods of the project’s time span.
Many version control systems employ delta algorithms to store different versions of a model. State-based delta algorithms store the differences of the various versions of a model with regards to a base model. These differences may be in the form of performed change operations (e.g. deletions, additions, refactorings, etc.) or in the form of actual differences between the models. In forward delta algorithms, the original version of the model is considered as the base model, while in backward deltas the latest version is selected as the base. To reconstruct a version of a model, the stored differences are applied on the base version to create the required version.

In a collaborative design setting, version controlling is essential due to the variety of the models that co-exist and are incrementally changed and updated by various designers.

2.1.6. Issue Traceability

An important characteristic of collaborative processes is that the participants frequently join or leave the course of development and since many of the ideas, selection rationale, reasoning over the produced artifacts of the process are implicitly incorporated into the mental model of each individual participant, there is the chance of loosing many important pieces of valuable information. Such data that portray the motivations behind many of the decisions are often referred to as organizational or collaborative memory [51, 8]. Configuration management change files, formal process models, version control systems, software bug tracking systems, and organizational best practice guideline repositories are some of the means to capture such information.

Most often the sequence of individual steps of each participant is stored in interaction history repositories [52]. Identifying sources of issues and enforcing future policies based on these repositories is a difficult task, since appropriate information need to be dynamically mined from the wealth of available data. Commitment stores are more specific than interaction history repositories and are employed for managing the individual claims that each participant makes. Using these structured information records provides the possibility to prevent denial of claim, contradiction of prior reasoning, and the expression of conflicting statements and models. These information stores are commonly formed using a centralized architecture, while distributed storage of each participants information in a peer-to-peer structure is also possible [53]. The major advantage of storing all of the information in a central location is that traditional reasoning mechanisms that do not support distributed reasoning can also be used on such information stores. However, the downside of this approach is its need for a high capacity information storage.

There have been two main approaches for attributing traceability information directly to one or more of the model entities. In the first approach, traceability information are stored along with the model structure [54]. This approach pollutes the model information with irrelevant data not directly related to the model itself. Furthermore, in a collaborative process, models are gradually merged which will further require traceability information to be merged too. This will cause new issues such as precedence, and distributed reasoning. In the second approach, traceability information are stored based on a unified traceability schema [55, 56]. The schema will hence manage the issues related to indexing, categorizing and reasoning of
the information. Furthermore, slicing traceability information to access its most relevant subset to a given issue is also more easily feasible using a unified schema.

2.2. Artifact-neutral Collaboration

Collaboration among various designers can be enhanced by other means beyond the mere concentration on issues directly related to the target artifact. For instance, monitoring the behavior of the designers, providing them with assistance over technology employment issues, bringing awareness to the designers’ workplace, and harmonizing the activities of the participants are some of the activities that can be undertaken to facilitate collaborative modeling. In this dimension of the framework, we address issues related to the more social aspects of collaboration, such as approaches addressing controlled negotiation and argumentation, strategies for activity coordination, and facilitation, and methods for raising participants’ awareness. Figure 3 depicts the building blocks of the artifact-neutral collaboration dimension of our framework.

2.2.1. Argumentation and Negotiation

Solving messy problems is an argumentative process that involves structural and logical reasoning as well as informal negotiation [57]; therefore, negotiation is required in conditions where process participants possess conflicting interests and intend to maximize their gain through cooperation [52]. Negotiation is a form of collaboration where the participants reach a mutually acceptable solution for the object of negotiation. In this process, each party is trying to ultimately improve the satisfaction rate of their own goals, while also wiling to compromise to satisfy communal goals as well. In contrast to
negotiation models that revolve around the object of negotiation, argumentation approaches allow the participants to exchange additional information about their beliefs, intentions and mental attitudes during the negotiation process in order to both justify their standpoint and influence the other participants [58]. A justification may contain the reason a certain proposal was put forward or why criticisms where made. Justifications are effective in changing the other participants’ region of acceptability [58].

In a negotiation process, various negotiation objects may influence the outcome of each others process. For instance, suppose two designers are debating over the correct model for a certain apparatus of a system. This argumentation should only consist of reasoning about the apparatus in question; however, if one of the designers decides to reason over the correct model by referencing an interdependent device to the apparatus, the final outcome of this negotiation object would be influenced by the other. Threats and promises can be also employed by the participants in the design process to put pressure on or to entice the others. For example, a promise can be made by a group of designers that if their change request is accepted in this phase, they will deliver some of the later models ahead of schedule [52]. A different negotiation strategy is to employ and introduce a predefined design plan and its required resources to justify design choices. This approach is specially useful in collaborative design, since a modeling process is usually a long-term set of interrelated steps. Therefore, current design decisions can be justified by the elicitation of future design plans.

In multi-agent systems, negotiation is formal process in which the interaction protocols and the rules of dialogue are explicitly states. This will allow different vendors of agent systems to be able to design platform independent agents that can collaborate in an open environment. As opposed to multi-agent systems, in a collaborative modeling setting, the participants are often human experts; therefore, a very strict negotiation protocol may hinder an efficient negotiation process. For this reason, some simple negotiation speech acts are usually employed such as propose, accept, and reject [59]. The major issue with this restricted set of axioms is that most of the reasons should be stated in an informal language that cannot be used in later inference procedures. One solution is to use more complex negotiation protocols, but graphically describe them using finite state machines, state transition diagrams, or petri nets. This way the negotiation process can accommodate dynamic logical inference throughout the negotiation procedure. It would also be able to enforce directives such as rules of negotiation participation admission, proposal withdrawal, commitment, and termination [52]. Analogous to issue traceability, negotiations can also be traced and employed for future use by storing the formal negotiation process in interaction history or commitment stores.

A widely used formal argumentation structure has been developed by Kunz and Rittel called Issue-Based Information System (IBIS) [60]. The argumentation model in IBIS revolves around three major concepts, namely: issues, positions, and arguments. Issues are important concerns that are raised by the participants and are subject to debate. In response to each issue, a set of positions are formed that represent possible answers, or solutions to that issue. Arguments are declared by the
participants to rebut or support a given position. More recent extensions of IBIS include gIBIS [61], QuestMAP [62], and HERMES [63]. Based on this structure, the argumentation process can be semi-formally organized. For instance, various types of inconsistent statements can be identified. Inconsistencies in IBIS can be categorized into three classes: idea, argumentation, and position inconsistency [60]. In idea inconsistency, the participant introduces two issues which are inconsistent; whereas in argumentation inconsistency, the participant first argues in favor and then against a given issue. Furthermore, a position inconsistency arises when one supports two contradictory, or incompatible issue-argument pairs.

The three main concepts of IBIS have also been employed and extended in different application areas. HCOME is an ontology engineering methodology which incorporates an extended version of IBIS in its shared space [64]. The set of concepts used in HCOME are Elaboration, Justification, Contrast, Alternative, Example, and Counter Example. In another approach, IBIS has been extended by adding evidential reasoning features to its core elements [57]. The proposed model, combines and evaluates the uncertain positions of the participants using the generalized natural combination rule. Other models for pruning argumentation trees for finding the best explanatory representation [65], and also models for determining the most useful discussion and winner of an argumentation can be found in the literature.

2.2.2. Activity Coordination

In distributed design, each group of participants has its own set of tasks and related goals to pursue that are specific to them. These groups need to share resources and the result of the processes that they perform. On the other hand in a co-design scenario, all the participants share an identical goal and contribute to the attainment of that goal from different perspectives. This is achieved through very constraint direct cooperation. In both of these scenarios, the roles and tasks of all the participants are mutually interdependent leading them to the need for continuous coordination. Being mutually interdependent means that one party relies on the quality and timeliness of the result of the tasks of one or more other participants [2].

Task interdependencies are often understood and explained through the employment of the Activity theory [9]. Activity theory defines an activity as a central unit for understanding human behavior. Engestrom [66] defines an activity as an interdependent system involving a subject that builds the activity object and the group of subjects that are concerned with the state of this object. Cooperation is addressed in Activity theory by the notion of division of labor that shows how the subjects share the work amongst themselves. An activity can be dynamic and evolve based on the new needs of the subjects and the object, and the subjects can also replace the tools that they employ for the realization of the object. In Activity theory, activity harmonization can be categorized in three main gradual levels of sophistication: coordination, cooperation, and co-construction [67].

The interrelationship between the activities performed by each subject are referred to as coupling of work [2]. Coupling depicts the degree and type of communication required by the dependant activities. Based on the extent of interdependency of the related activities, three types of couplings can be identified. Loosely-coupled
activities often consist of the transfer of the input and output data streams at the beginning and end of the activities. The loosely-coupled approach is the simplest type of inter-connection of two activities. Moderately-coupled activities are used to automatically manage the transformation and transmission of the information between two activities. This approach is more sophisticated than the loosely-coupled model in that meta-data based transformations are used to interrelate two incompatible tasks. In tightly-coupled activities, the work require frequent, complicated communication and coordination among the group members with regular feedback loops. One of the main socio-technical approaches to managing task interdependencies is interdependency reduction, such that a tightly-coupled task is transformed into one or more loosely-coupled tasks (task decomposition) [68]. Other solutions for handling task interdependencies are to collocate ambiguous tightly-coupled activities, minimize cross-site communication needs, and employ understandable task (work)-flow or project management tools.

2.2.3. Facilitation

In a face-to-face group decision making process the participants explicitly or implicitly transfer a set of information that enhances the overall performance. These information usually referred to as communicative actions include eye contact, facial expressions, gestures, and body posture. Furthermore, the surrounding environment that can incorporate shared objects, events, constraints and yield similar experiences for the participants is of significant importance. In many distributed collaborative modeling processes the possibility of collocating all the members of the process is very low; therefore, group facilitation is an important aspect of a collaborative modeling and design process. A facilitator either it be a human or a computer agent, should have certain features. A facilitator should be able to communicate well with different design teams consisting of various experts and fields of expertise, listen and observe efficiently, maintain authority for decision making, empower the groups, and separate ideas from the originator [69]. A facilitator should inform the participants about their various options rather than constrain their possibilities.

More informal than a facilitator, a boundary spanner is a person who wanders around different design groups and transfers the required information between the participants [70]. A boundary spanner should be able to translate the acquired information to the target audience in a way that the most benefit is reaped from this process. This is usually an informal role adopted by people with great communication abilities in the design process. Boundary spanners are responsible for providing the basis for two important requirements of the collaborative modeling process: inter-comprehension through cognitive synchronization (construction of a frame of reference), and perspective clarification and convergence. Cognitive synchronization is a pre-requisite to perspective clarification. In cognitive synchronization, the boundary spanner develops a unique understanding of the process intentions and the required final product among all the participants usually prior to the start of the process; however, through perspective clarification, the intent and objective of various modelers are communicated through the boundary spanner to develop a better understanding for reaching consensus. Although facilitation is
a process that is best performed by a human, various models of model recommendation for assisting the modelers, blackboard systems, shared digital environments, desktop sharing applications, video monitoring systems and many others have been developed to assist the human facilitators [71].

2.2.4. Social Issues  Many authors have discussed that designing is more than a cognitive process and should be perceived as a social process [9]. The participants of a design process usually adopt social strategies in their actions [72]. For instance, the participants may compete in order to convince the others to accept their position, taking a win-lose attitude, or practice a collaborative approach, where attention is focused on building common gains which is a win-win strategy. They may also take up the compromise approach which is also a win-win strategy where all the parties divide the gain, or the avoidance tactic where all the parties decide to retreat. In all these cases, the participants are working towards an agreeable equilibrium in the social process. The equilibrium is more easily attained if common grounds have been established.

Détienne argues that there are various factors that affect the construction of common grounds between the participants of a design process [2]. These factors are co-presence where the participants share the same physical location, visibility and audibility in which participants’ presence and behavior are observable by the others, contemporaneity that depicts a scenario where the effects of participants actions are instant, sequentiality showing the order of activities between the design groups, reviewability where participants can audit each others work, and etc. Common ground construction and mutual understanding is also enhanced through awareness [2]. Methodologies enhancing awareness can be classified into three major classes based on the type of awareness that they produce [7]. Task-oriented awareness approaches inform the participants about the state of affair of a specific task, or action. This can be achieved by distributing the participants’ information regarding the condition of an artifact or shared environment of the activity. The dissemination of information can be done according to subscription, or pull-push strategies [7]. Social awareness presents information about the presence and activities of the participants in the shared environment. Collocation awareness provides a (virtual) shared location so that the participants become acquainted with the environment. As an example, in HCOME, the designers are provided with three different spaces, namely: shared, personal, and agreed [65]. The designers initially design their models in their personal spaces, these models are then distributed for argumentation in the shared space, and the accepted models are moved to the agreed space. Other models such as virtual rooms, team rooms, email change notification systems and others can be found in the related literature that enhance awareness and assist the construction of common grounds [51].
2.3. **Communication Infrastructure**

An important aspect of any collaborative activity is the availability of suitable communication facilities and models. In this dimension of the framework, we will briefly review the type of technologies and models that have been employed in collaborative modeling and design environments.

### 2.3.1. Technologies

Computer-aided design requires advanced graphical modeling, and streaming technologies. Some of these technologies address models for delivering and manipulating 3D objects over the network such as VRML, X3D, W3D, and MPEG4. VRML is a standard for representing geometric multi-dimensional elements and scenes, X3D, W3D, and MPEG4 are enhancements that support audio-video media transfer in compressed binary format. Many of these formats store their models using triangular meshes and trimming lines [6]. In a distributed design setting, the models need to be easily transferred between the modelers; therefore, advanced streaming and model compression techniques need to be employed. Mesh simplification, object prioritization, vertex decimation, edge contraction and vertex clustering are some of the techniques that are used to simplify the design models for faster and more convenient communication [6]. Some other approaches employ model differencing algorithms in order to reduce the overhead of continuous model transfer. In these models only the applied changes to the reference model are submitted to the other participants and hence reduces the required communication bandwidth [1].

Other models of communication and coordination in a distributed environment include the use of text, audio and video messaging systems [5]. Through messages the designers can communicate with each other to exchange design ideas. Similar to network protocols, message passing in such contexts needs coordination. The control token mechanism is one of the mechanisms that manages the interaction between the designers. In this approach, only those designers that possess the control token can send messages or edit the shared model. After each action, the token can be automatically or manually transferred to the other participants. Periodic transmission of messages should also be controlled, since too many audio or video messages can hinder communication efficiency. An interesting observation was made by Damian et al [69] in a distributed environment that the slowing down caused as a result of computer-mediated conversation resulted in an enhanced ability of the designers to make proper decisions. This may be a direct result of the extra time that is gained for decision making caused by delay.

The seamless exchange of information can be also enhanced using shared ontologies [65]. Shared ontologies can be used to facilitate common understanding. In some projects, core upper ontologies such as PROTON [73], SUMO [74], and DOLCE [75] are employed to promote knowledge sharing. For example, SUMO is a product of the IEEE standard upper ontology working group that serves as a standard upper ontology which promotes inter-operability, information search and retrieval, and automated inferences [34]. Such core ontologies can be employed by the participants to construct common grounds and mutual understanding.
2.3.2. Models Communication models in collaborative modeling environments are often organized using client-server, web-based, peer-to-peer or agent-based systems. For instance, the Adlib system [76] employs a multi-agent architecture consisting of domain specific agents to design light industrial buildings. The agents employ the monotonic concession protocol in order to negotiate and reach a consensual design. In a different approach, agents are used within the architecture of VisAXSM [31] to automate the mapping of data model specifications. Each of the agents in this architecture incorporates appropriate mapping heuristics that can be used to find model correspondences. The advantage of employing agents in collaborative design is that agents can incorporate decision rationale and heuristics and obtain autonomous control. This would reduce the need for continuous monitoring and coordination of the activities performed in such an environment.

Web-based applications, that commonly use a client-server architecture have also been widely used. DOME [15], for example, is framework that allows designers to build integrated models using local and the need for distributed resources. WWDL is also a web-based application that enables designers to participate in a/synchronous dialogues over the web. Web-based DFX tools and WebCadet are some other examples that employ a web-based deployment strategy [77]. Web-based systems are widely used because they provide world wide accessibility through their web interfaces. Peer-to-peer systems are also interesting in settings where not all of the design resources are available to all of the participants. In this setting, each participant can share its resources and also use the resources that are shared by the other participants. Groove Networks [78] is a peer-to-peer collaborative software development platform that allows digital collaboration, and TOMSCOP [79] is a synchronous peer-to-peer collaboration platform that functions over the JXTA technology. Peer-to-peer systems are more fault-tolerant compared with the other models since various services and resources of the design framework are distributed over all the participants which eliminates the single point of failure issue. The employment of any of these architectures depends greatly on the characteristics of the modeling domain and its type of resource distribution.

2.4. Related Aspects

Collaborative modeling and design shares many aspects with other fields of research such as software development, game theory, cognitive science, and etc. In this dimension of the framework, we briefly introduce some interesting work in other fields that can be employed in collaborative modeling. This review is not exhaustive and is just meant to illustrate such potential.

2.4.1. Model Analysis The field of software development has experienced many different models of quality checking and program testing. These models are employed to find any bugs or defects in software programs. For instance, regression testing [80] is used to identify regression bugs which occur when the software system stops behaving as it previously did, or it does not provide the correct functionality
that it was planned to provide. This kind of analysis can also be used in collabora-
tive modeling where a model is incrementally developed by different participants. 
Regression testing can be employed to identify the aspects of the developed models 
that fail to operate according to the specifications as a cause of a change. The 
change may be performed on the failing segments of the model, or may be carried 
on out on some other segment causing a ripple effect. Ripple effects are common in 
collaborative modeling and design, where one modeler changes a segment of the 
design without understanding the complete effect that it will have on the overall 
model. To understand the effects of change on a model, impact analysis studies 
[81] can be performed. Impact analysis identifies the parts of the model that are 
directly or indirectly affected by a specific change.

The identification of stable model areas is also an interesting venue of research. 
In program stable area analysis, the history of changes made to a software program 
is analyzed to find the most well established and vulnerable segments of the code, 
respectively. Using these information, those segments of the code that are very 
vulnerable to change are tested more thoroughly and receive more attention. Similarly, in collaborative model design, stable model areas can be identified which can 
serve as a core segment of the model that is agreed upon. The frequently changed 
segments of the model can then receive more elaboration and argumentation.

Formal instance generation for model analysis is also useful and attractive in 
model mapping and integration [29]. In such approaches, instance data are gen-
erated based on a given pattern are used to identify the correspondences between 
model entity or structures. Such models are useful for application areas where 
developed models can be instantiated with domain-based instance data. Sam-
ple areas that can benefit from this approach are ontology and database schema 
engineering.

2.4.2. Equilibrium Attainment A collaborate modeling process is informally a 
dialogue game [82] between the participant modelers. Each modeler is interested 
in maximizing its benefit by directing the final model towards its goals and objec-
tives. In this process, the participants can have short or long-term strategies where 
they compromise over some design decisions to gain competitive advantage in other 
situations. The ultimate goal of all the participating designers is to reach a final 
unified model (equilibrium) in which their desires are most satisfied. Game the-
ory principles have been employed in several argumentation and conflict resolution 
strategies [44, 76].

Modelers frequently encounter two common situations in a collaborative modeling 
process that can be explained through game theory. In the first situation, none of 
the designers gains any benefit (attains any of his/her desires in the final product) 
by unilaterally changing his/her local model. This can be caused due to extreme 
conflicts or inconsistencies between the local models of the designers. In this situ-
uation, even if one of the designers changes its model to some extent, still the vast 
degree of conflict is not resolved and therefore, the final unified model will consist 
of discrepancies that is not to the benefit of any of the designers. This condition is 
similar to a Nash equilibrium in game theory [83]. Unlike many problems, a Nash
equilibrium is not desirable in a collaborative modeling setting. The second situation occurs when a final decision should be made between several alternatives. In such a situation, the selection process is an equivalent of a zero-sum game where the gain or loss of one participant is exactly identical to the losses or gains of the other participants. Although loss or gain is not quantitatively measurable in a modeling process, the loss of a possibility to enforce one’s idea can be regarded as a loss in a modeling process. Each step in such a situation is Pareto optimal [84]. The study of game theory principles in the behavior of collaborative modeling participants can reveal interesting facts and can be helpful in designing better collaboration strategies.

2.4.3. Privacy Concerns  In a distributed modeling environment, many of the communications of the participants are transferred through a controlled communication method. Other than the formal communications, video monitoring system, and the facilitation methods observe the behavior of the modelers to enhance communication, raise awareness and communicate user attitudes. In such a setting, a major concern is the correct tradeoff between the collection of legitimate information from users’ interactions and their private information. An incorrect choice can violate the privacy of the participant which will entail less motivation to collaborate in a distributed modeling environment [7]. Even in cases where distributed collaboration is essential modelers may decide to reduce their amount of communication to a minimum that can further influence the outcome of the design process. Some basic models for preserving the privacy of the participants in a distributed environment have been proposed [85], but still many concerns require further attention.

3. Relevant Applications

In this section, we review several applications that directly or indirectly address the issue of collaborative modeling and design. Some of these applications only develop tools that are partially useful in the domain of collaborative modeling and are specific to one of the dimensions of the proposed framework; however, some other works intend to address the collaborative modeling process as a whole. The most important criteria for the selection of a specific application are 1) the model is in the focus of research and is well cited; 2) the scheme has been developed recently; 3) the scheme possesses a distinguishing feature that makes it significantly different compared with the others; and, 4) the scheme is well documented and sufficient amount of literature for understanding its performance exists. Table 1 reviews these application. In each row of the table, a specific application is introduced, a reference to a published documentation is given, the target dimension of the framework that the application addresses is listed, and a brief review of the application is also provided.
4. Concluding Remarks

In this paper, we have introduced a framework for classifying and understanding various aspects of the collaborative modeling and design process. An overview of some prominent sample systems according to the proposed frameworks dimensions has also been provided. The dimensions of the proposed framework can assist model developers choose their desired collaborative modeling systems specifications with appropriate features according to their requirements and needs. The proposed framework consists of four major dimensions that address different orthogonal aspects of this field. The framework distinguishes between model-oriented and artifact-neutral collaboration models and relates to each of these models by studying the relevant aspects of each approach. Models of communication architecture and technology are introduced and several related issues that can have a constructive effect in collaborative modeling have been pointed out.

Although there have been extensive research on the different aspects of collaborative modeling and design, still various open problems exist that need further attention. Since the scope of the introduced issues in this paper is vast, we only reference several open problems that are related to the collaborative modeling process as a whole (and not specific issues regarding the sub-dimensions of the framework). Some of these open problems that need exploration are as follows:

1. The software engineering community has mainly focused on the development of model-oriented collaboration applications. On the contrary, the CAD community has moved towards techniques that center around artifact-neutral approaches. The main challenge is to integrate these two approaches into a unified model, where the co-existence of them can effectively enhance collaboration. This would require automated synchronization and information transfer methods between model development and engineering technologies and the activity coordination tools such that a change in any of these tools is automatically transferred to the others and proper adjustments are made.

2. The outcome of each thread of expert negotiation and argumentation is decisive for future decision making procedures; therefore, formal argumentation and structured negotiation can support future inference and reasoning. The main drawback of such strict formal process is that the human participants are less flexible in expressing their opinions and can even in some cases lead the participants towards less communication and negotiation as a result of this complex procedure. The main research problem is hence to find the best balance in the use of formal restrictions in an efficient, effective, and easy to use argumentation process.
Table 1. Overview of the Selected Applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>Reference</th>
<th>Target Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMMA</td>
<td>[18]</td>
<td>Model Transformation, Merge (weaving), Mapping (projectors)</td>
<td>AMMA is a model engineering platform that provides facilities for model transformation, weaving and projection through ATL, AMW, and ATP, respectively.</td>
</tr>
<tr>
<td>EML</td>
<td>[19]</td>
<td>Model Transformation and Merge</td>
<td>Epsilon Merging Language is a rule-based language that provides tools for merging diverse meta-models. It provides its users with the possibility of defining pluggable algorithms called strategies.</td>
</tr>
<tr>
<td>VisAXSM</td>
<td>[31]</td>
<td>Model Mapping</td>
<td>VisAXSM is a tool that provides domain experts with an extensible set of schema analysis agents that suggest schema element correspondences.</td>
</tr>
<tr>
<td>JDiff</td>
<td>[43]</td>
<td>Model Differencing, and Matching</td>
<td>JDiff is a tool that compares two object-oriented programs and identifies differences and correspondences. For this purpose, the problem is reduced to a hammock matching problem in a graph.</td>
</tr>
<tr>
<td>UMLDiff</td>
<td>[42]</td>
<td>Model Specific Differencing</td>
<td>UMLDiff operates on two UML class models reverse engineered from Java code, and produces a structural change tree depicting the additions, removals, refactorings, and etc.</td>
</tr>
<tr>
<td>Correspondence-based Merge Operator</td>
<td>[37]</td>
<td>Model Merge</td>
<td>The merge operator takes as input two models and some of their inter-model correspondences and generates a duplicate-free union of the two models.</td>
</tr>
<tr>
<td>Bisimilarity-based Merge</td>
<td>[35]</td>
<td>Model (Statechart) Merge</td>
<td>The merging operator employs bisimilarity which is a recursive notion that employs forward and backward actions to match statechart states.</td>
</tr>
<tr>
<td>System</td>
<td>Purpose</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Distributed</td>
<td>Inconsistency Management</td>
<td>Given a set of models and the mappings between them, the model first merges the models and then verifies the derived model against consistency constraints. Three types of consistency checks are allowed, namely compatibility, multiplicity, and reachability.</td>
<td></td>
</tr>
<tr>
<td>Consistency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>χbel</td>
<td>Inconsistency Management</td>
<td>The framework provides means for reasoning over multiple inconsistent state machine models. The viewpoints are annotated with Quasi-Boolean logic values and a model checker called χchek performs an evaluation over the defined properties.</td>
<td></td>
</tr>
<tr>
<td>AGORA</td>
<td>Inconsistency Management</td>
<td>AGORA is goal-oriented discrepancy identification framework. The framework computes two metrics, vertical and diagonal variance, that allow the identification of viewpoint inconsistency.</td>
<td></td>
</tr>
<tr>
<td>Unified Traceability Schema</td>
<td>Issue Traceability</td>
<td>This schema provides a generic meta-model for capturing, storing and reasoning over traceability information.</td>
<td></td>
</tr>
<tr>
<td>HERMES</td>
<td>Argumentation and Negotiation</td>
<td>The system enhances multi-criteria group decision making procedures by extending the concepts in the IBIS framework. The system has been implemented in a medical decision making setting.</td>
<td></td>
</tr>
<tr>
<td>Uncertain IBIS</td>
<td>Argumentation and Negotiation</td>
<td>The IBIS framework has been extended with the ability to handle uncertain argumentations. Argumentation information are aggregated using the Triangular-norm-based Combination rule.</td>
<td></td>
</tr>
<tr>
<td>Framework/Tool</td>
<td>Methodology/Features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCOME [64]</td>
<td>Collaborative (Ontology) Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The HCOME methodology integrates model development and argumentation. It allows the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>development of models in a decentralized setting. It introduces three contexts: shared,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>personal, and agreed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KS-DMME [12]</td>
<td>Collaborative Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The knowledge-server-based distributed module modeling and evaluation framework al-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lows designers to build integrated models using shared resources and collaborate through</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resource sharing. The WebKIDSS is a web-based collaborative design platform that is</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>developed based on KS-DMME.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoCreateOneSpace [87]</td>
<td>Collaborative Modeling and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The software platform allows several modelers to hold online meetings. In the pro-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cess of the meeting, the modelers can mark and annotate two and three dimensional dia-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mgrams. It also provides versioning and history management facilities.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CollabCAD [88]</td>
<td>Collaborative Modeling and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The software employs a client-server architecture where various designers can share,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>view and manipulate each others multi-dimensional models.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebCADET [89]</td>
<td>Collaborative Modeling and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WebCADET is a web-based decision support system designed to support designers during</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the early conceptualization phase.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntegrationSuite [90]</td>
<td>Collaborative Modeling and Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integration suite is collaborative conceptual design framework that employs belief</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>theory to capture the uncertainty in the expressions of multiple experts. Such infor-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mation are employed in the framework to formally guide the participating modelers to-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>wards consensus.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. The type of communication architecture, or the type of information transfer over the communication infrastructure can have a direct effect on the models of activity coordination or argumentation models that can be employed. For instance, if the annotation of model elements in a collaborative process requires the transfer of a great amount of data over the network, the form of collaboration should be different with that of a situation where light-weight models are being developed; therefore, guidelines or adoption strategies need to be devised that allow the selection of the appropriate technologies and models for different conditions.

4. Issue traceability repositories may contain important information regarding the design behavior of each group of participants. These information can be useful in effectively designing inter-participant coordination strategies. For example, traceability repositories can be mined to automatically define or dynamically adjust the participants’ activity interdependencies.

5. As was discussed in the privacy sub-dimension of the framework, a very important aspect of collaborative modeling environments is preserving the privacy of the participants. This requires schemas that maintain an appropriate tradeoff between the important information that need to be stored and participants’ private data. A suitable balance can encourage the participants to wholeheartedly take part in the collaborative modeling process.

Notes

1. $\equiv$: equivalence, $\supseteq$: more general, $\perp$: disjointness, $\cap$: overlapping

References


