Towards Aligning Multi-Concern Models via NLP

Fatma Başak Aydemir and Fabiano Dalpiaz
Utrecht University
The Netherlands
Email:{f.b.aydemir, f.dalpiaz}@uu.nl

***Abstract*—The design of large-scale complex systems requires their analysis from multiple perspectives, often through the use of requirements models. Diversely located experts with different backgrounds (e.g., safety, security, performance) create such models using different requirements modeling languages. One open challenge is how to align these models such that they cover the same parts of the domain. We propose a technique based on natural language processing (NLP) that analyzes several models included in a project and provides suggestions to modelers based on what is represented in the models that analyze other concerns. Unlike techniques based on meta-model alignment, ours is flexible and language agnostic. We report the results of a focus group session in which experts from the air traffic management domain discussed our approach.**

***Index Terms*—requirements models, natural language processing, alignment, collaborative modeling, model management**

I. Introduction

Large-scale, complex systems such as driver-less vehicles, smart cities, aviation and railway systems are increasingly prevalent in our lives. Their ever-growing complexity poses design challenges for software systems engineers [1].

In requirements engineering (RE), modeling languages are used to capture the relevant parts of the domain to reduce the complexity and preclude the unnecessary details [2]. However, employing a single model or modeling language is insufficient to analyze a *large* system that should be studied from multiple *concerns*.

To tackle largeness, the modeling domain is traditionally divided into sub-domains, such as splitting a plane into engines, wings, passenger cabin and so on, each sub-domain having its own model. To handle the multiple aspects (including quality requirements), such as performance, safety, and security, one model per *concern* is created [3].

We focus on concern-based decomposition, and specifically on the challenge of *model alignment*. The models are aligned when they capture similar if not the same concepts from the domain. This helps ensure that all perspectives are sufficiently analyzed, so to allow well-informed trade-off analysis among the qualities that each concern studies.

Our work is triggered by a real-world case study on the evolution of the European air traffic management (ATM), where experts from different companies and governmental bodies collaborate to design the next-generation European ATM systems that enable the coordination of the many aircrafts that fly throughout Europe every day.

ATM experts typically use enterprise architecture tools to analyze multiple perspectives: security and safety shall be ensured while keeping the costs low and performance high. In this context, it is critical that the models cover the same domain in order to do a healthy trade-off analysis.

The challenges are that (*i.*) experts use different modeling languages; (*ii.*) even when studying the same concern (e.g., security); and (*iii.*) the modelers work from diverse locations and in different time zones. Within the PACAS research project[1], we are studying how to move away from the current way of aligning the models—via expensive face-to-face meetings—toward a Web platform that relies on collaborative modeling and automated reasoning.

Based on the example described above, we set the following research question.

> **RQ:** How to help distributed modelers align the models that cover different concerns in the same domain?

We propose a technique that (*i.*) analyzes the chunks of natural language available in a model, i.e., the element and relationship labels, (*ii.*) compares the identified concepts in a model with those that are only present in other models, (*iii.*) suggests concepts to be modeled based on the missing concepts and a domain ontology, and (*iv.*) learns from modelers' feedback to fine-tune future suggestions.

The technique is flexible and language-agnostic, for it does not rely on meta-model alignment. Our method makes use of a domain ontology to help identify domain-relevant terms that denote important concepts that are expected to appear in the models.

Organization. Section II presents related work. Section III details the proposed technique. Section IV shows feasibility on a realistic example from the ATM domain and reports on the results from a focus group. Section V presents conclusions and future work.

II. Related Work

Several requirements modeling languages have been proposed to capture different concerns of stakeholders. General-purpose modeling languages such as UML, SysML [4], goal models [5], [6], and problem frames [7] have been extended in different ways to focus on specific concerns. Taking security modeling as an example, researchers have proposed UML extensions like UMLsec [8], goal–oriented languages such as STS-ml [9], and extensions of

[1] http://www.pacasproject.eu

problem frames such as abuse frames [10]. Horkoff *et al.*'s systematic literature map reveals that over 100 extensions exist in the goal–oriented world only [11].

Along with these distinct modeling languages, the field of RE has long been concerned with catering different perspectives of stakeholders within a single project [3], [12]. The efforts have been focused on meta-modeling, and viewpoint and view definitions. Fischer *et al.* [13] define a viewpoint as a language which represents a meta-model and a view as an instance of a viewpoint.

Several tools support creating viewpoints and views. Sirius[2] is a framework to create standalone modeling tools. It allows defining viewpoints and views based on Eclipse Modeling and Graphical Modeling Frameworks (EMF[3] and GMF[4]). MetaEdit+ [14] is another standalone tool-set offering collaborative viewpoint and view editing. Recently, Nicolaescu *et al.* [15] presented a web-based collaborative modeling tool that supports viewpoints and views. Although effective when the modeling languages are agreed upon and can be linked via their meta-models, these approaches are not sufficient to promote model alignment when no such agreement exists.

Damian [16] argues that creating and sharing knowledge among stakeholders are still open challenges in global RE. According to Portillo-Rodríguez *et al.* [17], only web-based solutions are seen fit to support global software engineering practices. We build on these premises and offers a concrete solution for model alignment.

III. NLP-based Concept Recommendation

Existing literature on multi-view modeling assumes the existence of a common meta-model from which viewpoints (and then views) are created. Building and maintaining such meta-model costs time and requires effort whenever a new language is used. Our proposal is orthogonal: we rely on NLP to automatically detect the content of the model and to support an aligned co-evolution of models.

A. Overall Solution

Our technique analyzes the chunks of natural language text in conceptual models[5]. There are several sources of text in a conceptual model, including labels that describe elements and relationships, and comments or notes about either on specific elements or the overall model.

By leveraging NLP instead of explicitly mapping all the utilized meta-models, we deliver higher flexibility in terms of the languages that can be utilized: adding a new modeling language does not introduce additional effort. The drawback of our choice is that we lack explicit relationships between the modeling primitives of different languages (e.g., that *tasks* in L1 are related to *goals* in L2).

[2]https://eclipse.org/sirius/
[3]http://www.eclipse.org/modeling/emf/
[4]http://www.eclipse.org/modeling/gmp/
[5]A requirements model is a kind of conceptual model.

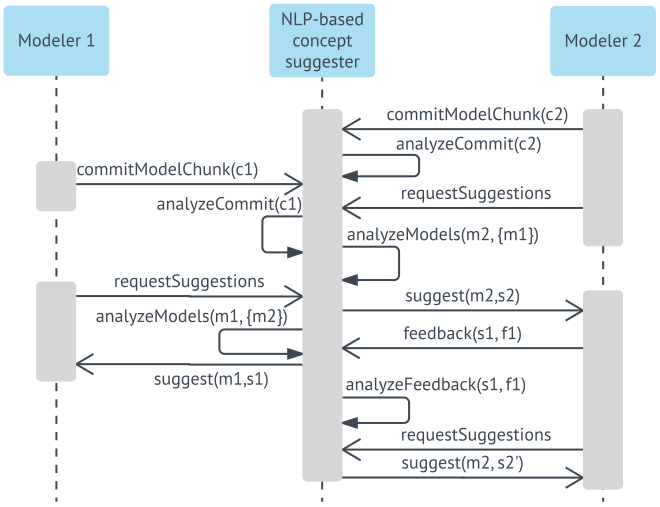

Fig. 1. Interaction diagram between two modelers and the NLP-based concept suggestion service.

Fig. 1 presents an interaction diagram between the NLP-based concept suggestion service and two modelers. The number of modelers can be arbitrary but is kept low for illustration purposes. The modelers inform the service on the content of the models as they model, and the service keeps track of the added, removed, and modified labels. When they need suggestions, the service analyzes the current states of all models in the process, consults a domain ontology, and returns the request with a list of suggestions from the domain ontology.

Input and output formats. To extract the natural text from the models, we need models to be exported in a machine readable format such as JSON[6]. Working with a textual representation of models also allows us to easily implement a *versioning system*. In a similar vein, we use *commits* as basic unit of input to keep track of significant changes rather than real-time changes in the models.

While Fig. 1 presents the interaction among the service and the modelers, Fig. 2 explains the steps followed by the service during these interactions. We provide the details of the three main steps below.

Step 1. When a modeler commits a model, we process the natural language text in the commit. We focus on nouns used in the models because nouns refer to concepts in the domain. We extract each label and comment from the model, identify nouns, calculate noun frequencies, and update the information we have about the model, thus we iteratively build a database of models.

Step 2. This step concerns generating suggestions. First, the nouns that are present in other models but absent in this model are identified by querying the database. Second, the domain ontology is queried to find similar and related concepts to the missing nouns. Finally, the result of this query is sent to the modeler as the list of suggestions.

[6]http://www.json.org/

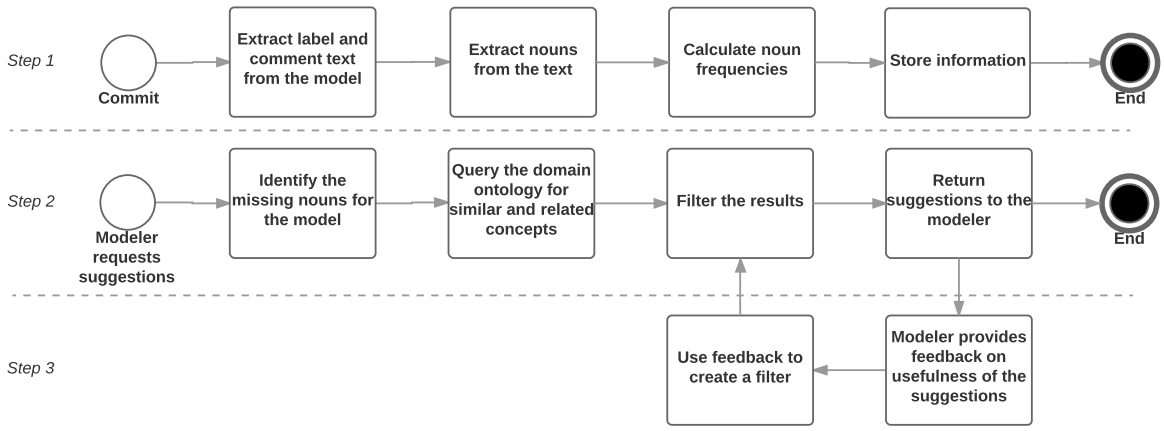

Fig. 2. Aligning multiple models by analyzing natural text in the models and using a domain ontology.

Step 3. Once the modeler receives the suggestions, we ask her feedback on the usefulness of the suggested concepts. The feedback is used in future iterations of the Step 2 to filter out concepts that are found useless.

B. Heuristics

We rely on a combination of several heuristics to discover and filter suggestions to the modeler. In this section we show pros and cons of these heuristics, thereby outlining challenges that should be addressed in future work.

Stemming: Nouns can be both singular or plural. Stemming can be used to unify the different instances so the noun frequencies can be accurately aggregated. However, stemming should be used with care: certain algorithms may result in over-aggregation when multiple nouns are derived from a single stem (e.g., 'aerodrag', 'aerodynamics', and 'aerosol' share the stem 'aero').

Noun phrases: A noun phrase consists of two or more nouns, such as 'flight envelope'. When nouns are individually extracted from labels, noun phrase information is lost. Consequentally, the frequencies of words that commonly occur in noun phrases increase more. For example, having 'flight level' (2 times), 'flight plan' (1 time), and 'flight state' (1 time) phrases in a model increases the frequency of 'flight' more than the other words in the noun phrases. Thus, the domain ontology will be queried for 'flight' which is the most frequent word and not for 'flight level' which is the most frequent noun phrase. NLP libraries still struggle to extract noun phrases from text, which is even a bigger challenge for the text extracted from models due to incomplete sentences or short forms. As a result, we opt for noun extraction rather than noun phrase extraction.

Exact match vs. similarity vs. relatedness: The domain ontology is queried to discover concepts that are similar or related to the nouns that are absent in one model but occur in the others. Many querying methods exist:

- ◇ *Exact match*: The ontology contains an entry that is an exact match of the noun.
- ◇ *Contains*: There is an entry in the ontology that contains the noun, e.g. 'aircraft flight status' for the noun 'aircraft'.
- ◇ *Synonym*: The synonym of the noun is in the ontology.
- ◇ *Similarity*: The ontology contains entries that are similar to the noun. Different NLP libraries implement different measures for similarity. For example, Wordnet offers a metric based on the distance between entries in the is-a hierarchy (hypernymy hierarchy), e.g 'airplane', 'jet', 'zeppelin' for 'aircraft'.
- ◇ *Relatedness*: The ontology contains entries that are related to the noun. Relatedness denotes any relation between concepts, such as antonymy, meronymy, or statistical correlation of occurrence. For example 'fleet', 'cabin', 'cockpit', and 'nose' are related to 'airplane' based on meronymy, however 'pilot' can also be considered related based on statistical correlation.

The precision and recall of the similarity and relatedness metrics have a direct impact on the performance of our approach. NLP tools implement different methods to calculate these scores in a variety of ranges. Discovering the tools that return solutions most similar to human judgment and combining the values returned by these tools to fine-tune the results is an open challenge that requires further experimentation.

Number of suggestions: Too many suggestions may overwhelm the modeler, while too few may not have the intended impact on the modeling process. The number of suggestions can be taken as input from modelers, or can be learned over time based on their feedback. To keep it low, our approach queries the domain ontology only for the nouns with the highest frequency of occurrence.

C. Implementation

We have implemented a web service for NLP-based alignment and integrated it with a collaborative decision making platform that uses models to capture different concerns of European ATM systems. The service exchanges information with an online modeling platform in JSON,

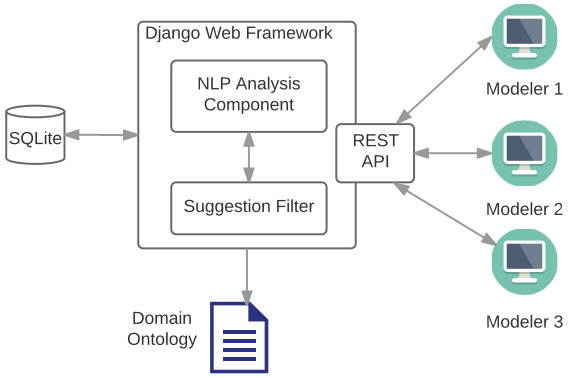

Fig. 3. Architectural view of the concept suggestion service

receiving model information and sending suggestions to the modelers. Modelers request suggestions via their user interface and the suggestions are presented by an avatar in a speech bubble. The service has been implemented using the Django[7] web framework. We use NLTK [18] to tokenize text, extract and stem nouns, and find similarity based on the WordNet lexicon [19]. SpaCy[8] is used to get relatedness scores for concept pairs. The ATM Information Reference Model (AIRM[9]) is used as the domain ontology. Fig. 3 presents the architecture of the service.

IV. Preliminary Evaluation

First, we show *feasibility* of the presented approach via an illustration on a realistic case from the ATM domain (Section IV-A). We then report the results of a focus group discussion with heterogeneous ATM domain experts on the perceived *usefulness* (Section IV-B).

A. Illustration on ATM

We illustrate our service on a small example from the ATM field. Consider two modelers, Sarah and Alice who work on a new solution to manage the European airspace. Sarah decides to build an organizational model and starts by modeling the goals of the supervisor role. The supervisor is responsible for assigning Air Traffic Controllers to sectors. After adding this role, Sarah commits the model in Fig. 4 and switches to another task.

Meanwhile, Alice requests suggestions to have an idea on where to start. She receives the following suggestions from the AIRM ontology based on the organizational model: 'Transferring Unit or Controller', 'Controlled Time Over', 'Sector Configuration Plan', 'Minimum Sector Altitude'.

Based on the suggested concepts, Alice decides focusing on 'Transferring Unit or Controller' and identifies risks associated with it. The resulting fault tree is shown in Fig. 5. When Sarah returns back to modeling, she receives the following suggestions based on the safety model

[7]http://www.djangoproject.com
[8]http://spacy.io/
[9]http://www.airm.aero/

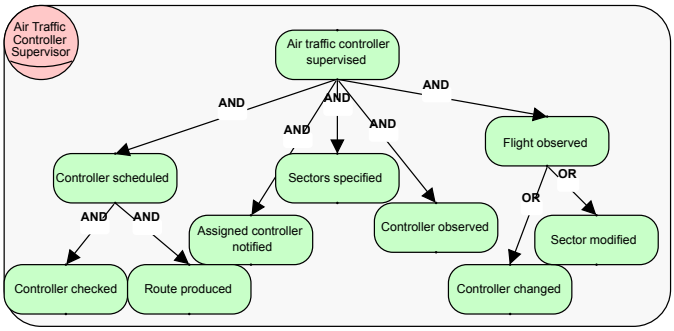

Fig. 4. Goal model of the supervisor role

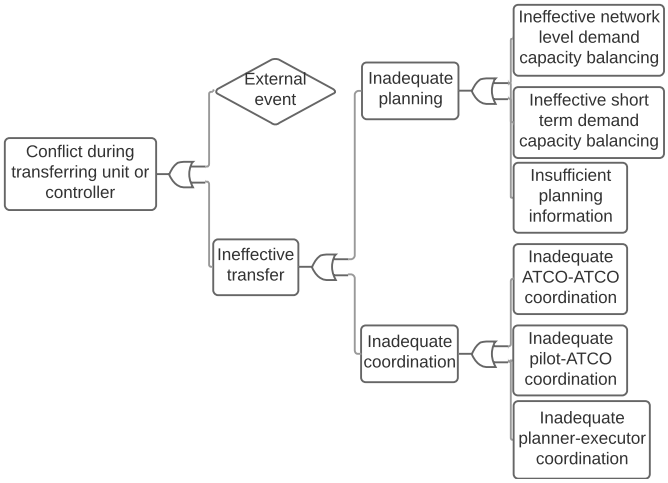

Fig. 5. Fault tree for 'conflict in transferring unit or controller'

presented in Fig. 5: 'Apron Inadequacy', 'Maneuvering Area Inadequacy', 'MCDM Coordination', 'Coordination Message', 'Medium Term Planning'. Now Sarah should decide on how to continue, e.g., by analyzing coordination between controllers. The two modelers keep requesting suggestions during the modeling process to stay aligned.

B. Focus Group Discussion with ATM Experts

Both authors contributed to the design of a focus group study that was moderated by the first author. The selection of the participants was purposive from a pool of ATM experts with different backgrounds such as air traffic control, safety and security, and system engineering. The participants were affiliated with the PACAS project as the advisory board members, and they were aware of our research question (RQ in Section I). The participants were divided into two separate groups of two and three, and two one-hour sessions conducted. The study had been piloted with the members of the PACAS project.

The moderator first introduced the NLP-based concept suggestion service. Then, the participants were presented a scenario with a similar flow to the one presented in Section IV-A. The participants were asked to study the models and the suggested concepts. Next, they filled in a questionnaire, and the group discussion started. The

sessions were (audio) recorded. The material from the sessions and the questionnaire results are available online.[10]

Four participants agreed that the technique helps to improve the quality of models, the other participant disagreed. Three participants agreed that the technique helps to speed up the modeling process for the security model, one participant was neutral and one participant disagreed. Two out of five participants agreed that the technique helps to speed up the modeling process for the safety model, two were neutral and one participant disagreed.

The argument of the most negative participant was that the direction of the suggestions matter in the ATM domain, and certain type of models (e.g., security) do not rely on suggestions from other types of models.

Open-ended questions got mixed replies. Two participants found high level suggestions useful, whereas others leaned toward lower level suggestions (e.g., 'trajectory' and 'trajectory change point', respectively). The subjects preferred to receive suggestions at the beginning (2), during (2), and at the end (1) of the modeling sessions.

V. Conclusions and Future Work

We presented a NLP-based technique to align models that capture different concerns of a domain. Notably, our approach supports alignment among models without mapping their meta-models, thereby gaining in flexibility whenever a new modeling language is employed.

Our aim is to free modelers from the restrictions on modeling languages and frameworks. As long as model information is shared in a portable format such as XML or JSON, the technique can be used to align multiple models. The web-based implementation uses non-restrictive standard HTTP protocols for communication that can be integrated to both web and desktop applications.

The technique received a mostly positive evaluation and raised a healthy discussion during focus group discussion. The original idea was found to be potentially useful and the suggestions on the models provided during session were found to be related to the models.

Our method does not automatically align models, instead it guides human experts on what to model to cover the same domain. Therefore it is unsuitable for projects where one-to-one mapping in between models are required. The technique relies on the performance of the NLP-tools used for implementation and its application is limited to the natural languages for which effective NLP-tools exist.

This paper paves the way for future work. The focus group confirmed the need for different heuristics based on the individual preferences of the modelers. Large-scale experimentation is necessary to reliably assess the validity of our approach and compare it against the performance of meta-model mapping techniques. One interesting direction is to study whether the approach is better suitable for certain types of (requirements) modeling languages. Furthermore, we should study the modeler-service interaction: how many suggestions? how often? how to gather and process feedback from the modeler on the suitability of the suggestions?

[10]https://www.staff.science.uu.nl/~dalpi001/modre-material/

Acknowledgments

The authors would like to thank the PACAS consortium partners. Special thanks go to the advisory board members who joined our focus group session. This work has received funding from the SESAR Joint Undertaking under grant agreement No 699306 under European Union's Horizon 2020 research and innovation programme.

References

[1] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, and R. Paige, "Large-scale complex IT systems," *Communications of the ACM*, vol. 55, no. 7, p. 71, 2012.

[2] S. Greenspan, J. Mylopoulos, and A. Borgida, "On formal requirements modeling languages: RML revisited," in *Proc. of ICSE*, 1994, pp. 135–147.

[3] B. Nuseibeh, J. Kramer, and A. Finkelsteiin, "Viewpoints: Meaningful relationships are difficult!" in *Proc. of ICSE*, 2003, pp. 676–681.

[4] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[5] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp. 978–1005, 2000.

[6] F. Dalpiaz, X. Franch, and J. Horkoff, "iStar 2.0 language guide," *CoRR*, vol. abs/ 1605.07767, 2016.

[7] M. Jackson, *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001.

[8] J. Jürjens, "UMLsec: Extending UML for secure systems development," in *Proc. of UML*, 2002, pp. 412–425.

[9] F. Dalpiaz, E. Paja, and P. Giorgini, *Security Requirements Engineering*. MIT Press, 2016.

[10] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett, "Introducing abuse frames for analysing security requirements," in *Proc. of RE*, 2003, pp. 371–372.

[11] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, J. Mylopoulos, and P. Giorgini, "Goal-oriented requirements engineering: A systematic literature map," in *Proc. of RE*, 2016, pp. 106–115.

[12] G. V. Zemanek, M. A. Jeusfeld, M. Jarke, H. W. Nissen, and H. Huber, "Managing multiple requirements perspectives with metamodels," *IEEE Software*, vol. 13, pp. 37–48, 1996.

[13] K. Fischer, D. Panfilenko, J. Krumeich, M. Born, and P. Desfray, "Viewpoint-based modeling-towards defining the viewpoint concept and implications for supporting modeling tools," in *Proc. of EMISA*, 2012, pp. 123–136.

[14] H.-G. Fill and D. Karagiannis, "On the conceptualisation of modelling methods using the adoxx meta modelling platform," *Enterprise Modelling and Information Systems Architectures*, vol. 8, no. 1, pp. 4–25, 2015.

[15] P. Nicoalescu, M. Rosentengel, M. Derntl, R. Klamma, and M. Jarke, "View-based near real time collaborative modelling for information systems engineering," in *Proc. of CAiSE*, 2016, pp. 3–17.

[16] D. E. Damian, "Stakeholders in global requirements engineering: Lessons learned from practice," *IEEE Software*, vol. 24, 2007.

[17] J. Portillo-Rodríguez, A. Vizcaíno, M. Piattini, and S. Beecham, "Tools used in global software engineering: A systematic mapping review," *Information & Software Technology*, vol. 54, pp. 663–685, 2012.

[18] S. Bird, "NLTK: The natural language toolkit," in *Proc. of ICON*, 2005, pp. 69–72.

[19] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.