WHY SHOULD CHEMO-ENGINEERS BE INTERESTED IN GRAPH GRAMMARS?
A SOLUTION TO THE INCONSISTENCY PROBLEM IN DISTRIBUTED MODELING

Stefan Greiner
Graduiertenkolleg für Informatik und Technik
Rheinisch-Westfälische Technische Hochschule
52056 Aachen/Aix-la-Chapelle/Aquisgrana, (D)
greiner@tkk.rwth-aachen.de

ABSTRACT
In distributed modeling, a group of developers are elaborating some specification in a work-sharing manner. Such specifications and partial models are constructed according to a divide-and-conquer principle. Maintaining the consistency of the evolving sub specifications, generally called documents, is the main meta problem of distributed modeling. The volume of literature on consistency maintenance is vast, especially in the fields of classical software engineering. In this contribution, it is shown how consistent document configurations can be represented by hypergraph and graph grammars. Such models can serve as formal requirements definitions for various kinds of integration tools and consistency management systems. The approach presented here is not at all restricted to software engineering. As our team is busy in a long-term research project on software support for distributed modeling in chemical engineering, the running example of this contribution is taken from this engineering discipline.

KEY WORDS
consistency maintenance, integration tool, specification, coupling, graph schema, graph grammar, modeling

1 INTRODUCTION AND RELATED WORK
Graphs are not only of mathematical interest by themselves, but they also serve as intuitive modeling means in almost every practical application domain. For example, Petri nets [19][20] and Chen diagrams [4] have been such fundamental break-throughs in their times that many nowadays’ graph-based modeling approaches like dynamic task nets [13][30] or UML [9] are still related to them. Other examples of modeling graphs are algorithm flow diagrams, module interconnection diagrams [17], functional decomposition diagrams [6], and so on.

Speaking about modeling, it is important to remember that this can be a really industrial activity with very large documents to be constructed and many people involved. To overcome the complexity, the modeling task is usually shared among the developers who are elaborating different—yet complementary—sub specifications. Step by step, a complex document configuration evolves. Keeping this configuration in a consistent state is a crucial pre-condition of project success. At release time, all relevant sub parts of the specification must properly fit together.

On the other hand, model graphs can be viewed as 'sentences' of special kinds of grammars that have been called 'web grammars' in ancient times and that are now called graph grammars. The principle of polydimensional producing and transforming graphs by graph grammars and graph transformation systems is similar to the production of sentences by non-context-free Chomsky grammars [5] and term rewriting systems [2]. Since the first known report on graph grammars [21], a large volume of literature has been published on that subject. Latest state-of-the-art compilations are [24][25] while tutorial introductions can be found for example, in [1][7]. All of them provide sufficiently expressive lists of survey and specific references.

In distributed modeling, a configuration of mostly graphically denoted specification documents is evolving by the time. Graph grammars and graph transformation systems, on the other hand, describe the evolution of graphs in a formally sound manner. Therefore, it is no surprise that there can be found various approaches to the support of consistent distributed modeling by graph grammatical means and methods. See for example [3][12][14]. As a similar topic one can find graph grammatical schema transformation for consistent database management [14][27].

In [28] the authors describe how to split large graph grammar specification into adequate modules, and they introduce a new module concept to the graph programming language PROGRES which is still a monolithic language in the latest release [29]. This is interesting; because it is the module concept of a specification formalism that makes distributed modeling possible and applicable. Thus: the inconsistency problem—that I claim to be solvable by the means of graph grammars—is now in a strong position to 'strike back' onto this approach. As this problem is closely related to the view update problem of databases, the authors suggest database techniques like active constraints as compensation. My suggestion is not to apply distributed graph grammars for the purpose of solving inconsistency problems emerging from distributed modeling.

The advanced approach of [8] is conceptually similar to the method described in this contribution. There, separate graphic views of a system are maintained by a graph transformation system upon a common reference model. Moreover, a hierarchy of even more abstract meta views is possible. Particularly the graph typing facility of that approach is, however, not very advanced. It could be enforced by an inheritance hierarchy of vertex classes and arc classes as presented later in this paper.

As described in [16], there has been a decade of increasing experience in how to construct consistency maintenance tools employing Pratt's ideas of coupling grammars to each other [22] such that the coupled grammars characterize the space of all legal configurations constituting a whole problem description. In the early beginning, consistency maintaining tools have only been programmed ad hoc. Later one has found that tool specification considerably improved the subsequent tool implementation. But then it was the specifications to be rather ad hoc and not yet as methodically sound as it could have been. The ongoing research of the author is to find a more concise way of pre-specifying consistency maintenance software tools.

Janning has published a specification approach based on static graph schemas [15] that, however, does not employ any graph transformational means. A dynamic specification approach with so called triple graph grammars is known from Lefering [16]. In fact, his 'triple' grammars are more or less pair grammars, while his transformation approach (i) strongly sticks to the language PROGRES—that possibly vanishes some day—and (ii) is restricted to graph productions that may not destroy any vertex.

A sound combination of both the static and dynamic approaches in order to simplify the design of such integrative specifications, as explained by [11] in detail, is the topic of this paper. The new specification method shall be supported by a meta specification tool of which the implementation has already begun. In the following sections it is shown by a toy-size cheminoengineering example how a graph grammar specification of sub model integration dependencies can be developed. The coherence of the grammar coupling definitions will be checked relative to a inter-submodel correspondence schema that serves as static pre-specification of the consistency maintenance problem.

As our group is busy in a long-term research project on software support for distributed modeling in chemical engineering [26], the running example of this paper is taken from that engineering discipline. (One can find several commercial design tools for particular chemino-modeling tasks, but their cooperation and distribution facilities are still quite insufficient.)

2 MOTIVATION BY A SIMPLE EXAMPLE

In Fig.1 one can see a sketch of a bubble tank. Some liquid is pouring into and out of the tank. Bubbles of vapor are pumped into the tank and leave the tank through another valve. Heat radiation is warming up the liquid from outside. The liquid and the vapor bubbles are connected to (and separated of) each other by a thin film.

![Figure 1: bubble tank](image)

Let's further assume that this bubble tank shall be computer-simulated in order to explore its operational properties. For this final purpose (out of scope of this paper), a rough ER-like specification of the bubble tank is constructed at first. This is done in a distributed manner as introduced above. In Fig.2 the structural essentials of the bubble tank are represented by an attributed ER-like graph. This structure graph has the following meaning: The devises liquid and vapor are the main entity nodes. The relation between them is the connection. All rectangular nodes with rounded corners represent the devices by which the devices comminicate with their neighbourhod. The composition cell represents the bubble tank as a whole with all its components.

1 Fig. given by the Dept. of Proc. Eng., RWTH Aachen.
Another view of the bubble tank is described by the sub model graph in Fig.3, the material graph. This data structure is a cut-out of a (big) sub specification of what’s going on with the chemical substances in the bubble tank. The chemo-engineers speak of four single phases containing a mixture of several chemical components that are involved in two homogeneous reactions:

![Diagram of a material graph](image)

### 3 GRAPH SCHEMAS, GRAPH GRAMMARS

With graph schemas one can describe the gestalt of legal sub model graphs to be constructed. Thus, a graph schema can be interpreted as the type of a set of graphs. As graphs are constructed by graph grammars and as the rules of graph grammars are built of graphs themselves, a graph schema does also matter for the graph grammar constructing the graphs described by their common schema.

In the approach presented here, a graph schema is mainly a lattice over disjoint sets of vertex labels and arc labels. The lattice order $\preceq$ represents property inheritance, also called as a "relation" in a less rigorous way of (object-oriented) speaking. Lattice schemas offer the possibility of multiple inheritance modeling and they are, nevertheless, syntactically sound. Moreover, it is fair easy to discover redundancy of inheritance in lattices. The lattice structure is enriched with labeling functions defining the source vertex labels and the destination vertex labels of the arc labels:

**Definition: Graph Schema**

Let $\Sigma = \{ A, T \} \cup S$ with $S \neq \emptyset$ a set of symbols, also called classes. A quadruple $\Theta := (\Sigma, \leq, src, dst)$ is called graph schema over $\Sigma$, if:

- The partial order $\leq$ of $\Sigma \times \Sigma$ defines a lattice with $A$ and $T$ as bottom and top elements.
- $\Theta(\Sigma) := \{ (e \in \Sigma) | \exists r \in S : e \succ r \}$ is the set of final classes of $\Theta$, and $\Theta(S) := \Sigma \setminus \Theta(\Sigma) \setminus \{ A, T \}$ is the set of super classes of $\Theta$.
- $\Theta(\Sigma) = \Theta(S), \Theta$ is a set of vertex labels, $\Delta$ a set of arc labels, and $\forall e \in \Theta, \forall v \in \Delta$, $\forall r \in \Theta: (e \preceq \sigma, \Delta \preceq \sigma) \Rightarrow (e = r)$.
- $\Delta = \Delta \cup \Theta(S)$ and $\forall e \in \Delta \cup \Theta(S)$ are functions that map arc labels to source vertex labels and destination vertex labels, where $\Theta(S) = \Theta(\Sigma) = \{ a \leq \sigma, a \in \Delta \}$.

### 4 CORRESPONDENCES

Please remember that consistency relations have been stated between the sub model graph of Fig.2 and Fig.3 in Fig.4. Now the question arises, if those dependences can be represented employing the according graph schemas and graph grammars. The answer is "yes". But for this purpose, a graph schema and a graph grammar for the other involved sub model must be given, too. They are depicted in Fig.5 and Fig.6 below.

Again, it is possible to show that the graph grammar in Fig.9 is proper with respect to the graph schema given in Fig.7 describing chemical components in a very abstract fashion. The rules $p_0$ and $p_1$ allow the creation of Phase System nodes and Chemical nodes. Rule $p_2$ allows to plug both types of nodes together by the edge "--" as stated by the source and destination functions of the graph schema. Rule $p_3$ is, again, a useful lazy dummy.

So far, a graph schema and a graph grammar for each type of sub model graphs to be integrated have been constructed. In the following, a correspondence schema and a correspondence grammar describing the legal configurations of consistently related sub model graphs are constructed. Such correspondence specifications are

---

3 Fig. given by the Dept. of Proc. Eng., RWTH Aachen

4 In general, the same method may be applied to configurations with more that two involved sub specifications as well.
Please do not mix up these schema correspondences with the consistency relations (shown in Fig.4) of their instance graphs.

Definition: Correspondence Schema
Let $\Sigma_1, \ldots, \Sigma_n$ be graph schemas with alphabets $\Sigma_1, \ldots, \Sigma_n$ and lattice orders $\Sigma_1, \ldots, \Sigma_n$. A set $\mathcal{C}$ is a correspondence schema of size $n$ over $\Sigma_1, \ldots, \Sigma_n$, if $\epsilon = \mathcal{C} \subseteq \Sigma_1 \times \ldots \times \Sigma_n$ and $\epsilon \subseteq \Sigma_1 \times \ldots \times \Sigma_n$ are sets of positive and negative correspondence axioms, and $\{\epsilon_1, \ldots, \epsilon_n\} \in \mathcal{C}$. $\mathcal{C}$ is a correspondence schema if and only if $\epsilon_1 \subseteq \epsilon_2 \subseteq \ldots \epsilon_n$ are consistency rules, one obtains a particular correspondence definition from the general correspondence framework.

Coherent to the following principle, the following construction of the rules from Fig.6 and Fig.8 are defined: $\alpha_0 = (q_0, p_0)$, $\alpha_1 = (q_1, p_1)$, $\alpha_2 = (q_2, p_2)$, $\alpha_3 = (q_3, p_3)$, $\alpha_4 = (q_4, p_4)$, $\alpha_5 = (q_5, p_5)$. The corresponding modeling decisions are based on "insight". They are made by an expert person who is familiar with the possible inconsistency problems of the examined application domain.

Based on these relations it is possible to define the coupled graph grammar of Fig.10 obeying to the correspondence schema shown in Fig.9. Please note that the coupled graph grammar cannot be automatically derived from the correspondence schema, but the correspondence schema is important for reasoning about the adequacy of the coupled grammar.

5 SUMMARY

Difficult engineering tasks, for example the construction of a chemical device model, require a careful specifying to avoid as many construction errors as possible. Such models are usually distributed into several sub models representing different views of the whole problem for clarity reasons. But with this complexity reduction by distributed modeling a meta problem arises, namely: how to keep the sub models coherent to each other in a consistent total configuration.

In this contribution I have shown an example of a new way of specifying sub model integration by a syntactic combination of graph schema and graph grammar coupling. A more elaborated theory of this approach can be found in [11]. The main idea of the method presented here is to test the coherence of a (sub model) graph grammar relative to a given (sub model) graph schema as well as the coherence of correspondence graph grammar relative to a correspondence schema representing the consistency relations between the sub models of a whole model configuration. The main steps, amongst others, of this method towards an integrative specification are the following:

- Explicit exploits for a sufficient understanding of the problem domain. • Construct a graph schema for each sub view of the domain. • Discover integration relations between all involved sub models. • Construct a schema correspondence for these integration relations. • Construct schema-consistent graph grammars for each sub domain. • Construct a coupled graph grammar according to the schema correspondence.

Graph models of practical relevance are considerably
bigger than the intuitive examples given in this contribu-
tion. Therefore, computer support seems necessary for
the construction and maintenance of consistent in-
tegration specifications of realistic size. These integra-
tion specifications serve as basis for the implementa-
tion of interactive consistency control tools that are to be
used by the application experts, in our example: the
chemo-engineer.

Different specifications and implementations of such
consistency control tools are already reported in the lit-
erature on databases and software engineering. But the
specification—if any!—of those integrative tools is still
difficult and tedious. Therefore, a meta tool called 'cor-
respondence editor' that shall support the specifier of
consistency control tools is currently being implemented
by our research group. The main tasks of this meta tool,
amongst other analyses, shall be the following:

- Check, if a graph schema is syntactically correct and
  lattice-shaped.
- Check, if a graph grammar is coherent relative to a
graph schema.
- Compute a list of legal couplings from given schema correspondence.
- Check, if a correspondence grammar is coherent relative to the
  computed list.

In the title of my contribution I have asked the ques-
tion why chemo-engineers should be interested in graph
grammars. Now, at the end of the paper, it should be
obvious that graph grammars can serve as a pow-
erful means to describe the integration problems the
engineer is confronted to, and thus pre-specify the in-
tegration tools to cope with those problems. As there
already are approaches to automatically derive integra-
tion tools out of graph grammar specifications [14] we
may hope to meet some more useful integration tools
on that way some day. Of course, the running exam-
ple from chemical engineering can be replaced by any
other example of ER-like distributed modeling. How-
ever, software support for chemical engineering is a fair
novel research field to be examined by our group.

To avoid misunderstandings I have to stress that the in-
tegration method presented in this contribution is suf-
ficient only for the solution of str uct ur al incon-
sistency problems. Without incorporated knowledge
of chemistry, the integration system cannot prevent the
engineers from producing undesired chemicals by unde-
avored reactions. But supposed that the engineers are
experts of their subject, a structural consistency main-
tenance tool shall be able to increase the efficiency of
their modeling part of work.

One of several interesting open questions is: if an appro-
riate correspondence grammar can automatically—or
at least semi-automatically—be derived if the single
graph grammars, the single graph schemas, and the cor-
respondence schema are given.

ACKNOWLEDGMENTS
Drafts of this paper have been helpfully criticized by
Angela Radatz, Dr. Thomas Noll, and Prof. Dr.
Andy Schürr from different institutes of computer sci-
ence in Aachen and München.

Due to valuable remarks provided by the anonymous
referees of the TAGT'98 workshop I have—hopefully—
been able to improve as well the related work section
as the general understandability of this contribution.

The German Research Community DFG is granting
the author with a scholarship from a special fund for
doctoral students.

REFERENCES
[1] M. Andrés, G. Engels, A. Habel, B. Hoffmann,
H.-J. Kreowski, S. Kuske, A. Schürr, G. Taentzer,
Graph transformations for specification and pro-
gramming. Report 7/96, Fachbereich Mathematik
und Informatik, Universität Bremen, 1996
Springer, 1993
Languages and Tools for the Construction of Cor-
rect Software. LNCS 1009, Springer, Berlin 1995
a unified view of data. ACM Transactions on Data
Base Systems 1/1, pp. 3-36, 1976
[5] N. Chomsky, On certain formal properties of
[6] T. de Marco, Structured analysis and system spec-
[7] H. Ehrlig, M. Kerff, M. Löwe, Tutorial intro-
duction to the algebraic approach of graph gram-
mars based on double and single push-outs. Report
90/21, Fachbereich Informatik, Technische Univer-
sität Berlin, 1990
[8] H. Ehrlig, G. Engels, R. Heckel, G. Taentzer,
A View oriented Approach to System Modeling
based on Graph Transformatian. M. Jazayeri, H. Schauf (Ed.),
ESEC-FSE'97 Joint 5th European Software
Engineering Conference and 9th ACM SIGSOFT
Symposium on the Foundations of Software Engi-
nering. LNCS 1301, pp. 327-343, Springer, Berlin
1998
ley, New York, 1997
Tools Supporting Development Processes. M.
Broy, H. Kramer (Ed.), RTSE'97 Workshop on
Requirements Targeting Software and Sys-
tems Engineering. LNCS 1326, pp.235-256,
Springer, Berlin 1998. Earlier version electron-
ically available from ftp://ftp.informatik.rwth-
aachen.de/pub/reports/index.html
matischen Spezifikation von Integrationswerkzeu-
gen nach Westfahl, Janning, Leftering und Schürr.
Report A10-93-3, Fachgruppe In-
formatik, RWTH Aachen, 1998. Electronically
available from ftp://ftp.informatik.rwth-
aachen.de/pub/reports/index.html
[12] G. Heidenreich, Ein generisches Organisations-
schema der Konfigurationsverwaltung im Software-
Engineering (Doktor Dissertation). Arbeitsberichte
des IMM 29/13, Fakultät für Elektrotechnik und
Elektronik, Albert-Ludwigs-Universität
Erlangen-Nürnberg, 1996
tel, DYNAMITE: Dynamic Task Nets for Software
Process Management. ICSE'96 Proceedings of the 18th International Conference on Software Engi-
[14] J. Jähnke, W. Schürr, A. Zündorf, A design envi-
noment for migrating relational to object-oriented
data base systems. Proceedings of the international
conference on software maintenance 1996, IEEE
Computer Society Press, pp.163-170, 1996
[15] M. Janning, Requirements Engineering and Pro-
grammieren im großen: Integration von Sprachen
und Werkzeugen (Doktor Dissertation). Deutscher
Universitätsverlag, Wiesbaden 1992
[16] M. Leftering, Integrationswerkzeuge in einer
Software-Entwicklungsgebung (Doktor Disser-
tation). Shaker, Aachen 1995
[17] M. Nagl, Softwaretechnik: methodisches Programm-
ieren im großen. Springer, Berlin 1990
[18] M. Nagl (Ed.), Building tightly integrated software
development environments: The IPSEN approach.
[19] C. Petri, Kommunikation mit Automaten (Doktor
Disseration). Schriften des Institutes für instrumen-
telle Mathematik, Bonn 1962
[20] C. Petri, Grundrissichtes zur Beschreibung
üblicher Prozesse. Drittes Kolloquium über Auto-
matentheorie, pp.121-140, Birkhäuser, Basel 1967
International Joint Conference on Artificial
Intelligence, pp.609-619, Washington 1969
[22] T. Pratt, Pair grammars, graph languages and
string-to-graph translations. Journal of Computer
and System Sciences 5, pp.560-595, 1971
Languages with Layered Graph Grammars. Journal of
Visual Languages and Computing 7/3, 1996
and computing by graph transformation, Volume 1
(foundations). World Scientific, Singapore 1997
[25] G. Rosenzweig et al. (Ed.), Handbook of graph gram-
mars and computing by graph transformation, Volume 2
(specifications and programming), to appear 1998
[26] RWTH Forum of Informatics, SFB 476 IM-
PROVE. Informatik-Forschung und Entwicklung
13/2, pp.91-92, Springer, Berlin 1998
[27] G. Santucci, C. Batini, G. di Battista, Multilevel
schema integration. R. Elmasri, V. Koutrasmanjian,
B. Thalheim (Ed.), Entity relationship approach
ER'93, LNCS 823, pp.327-338, Springer, Berlin
1993/94
[28] A. Schürr, A. Winter, Modules and updat-
able Graph Views for Programmed Graph Revisit-
ing Systems. Report A10-97-3, Fachgruppe In-
formatik, RWTH Aachen, 1997. Electronically
available from ftp://ftp.informatik.rwth-
aachen.de/pub/reports/index.html
Version 9.1., 1998 http://www.13.informatik.rwth-
aachen.de/research/progres/index.html
[30] B. Westfechtel, Graph-based models and tools for
managing development processes (Lecturer Habilita-