Knowledge-Based Assistance for the Development of Drugs

Jürgen Frank, Birgit Rupprecht, and Veit Schmelmer, University of Heidelberg

THE KNOWLEDGE-BASED SYSTEM GSH PROVIDES ASSISTANCE FOR ONE PHASE OF THE DEVELOPMENT PROCESS OF A DRUG. IT IS DESIGNED TO BE USED AND MAINTAINED BY PHARMACISTS IN DAILY ROUTINE WORK. THE SYSTEM CAN SAVE DEVELOPMENT TIME, PRESERVE EXPERT KNOWLEDGE, AND AID THE TEACHING OF THE GALENICAL DEVELOPMENT OF DRUGS.

DEVELOPING A NEW DRUG PRODUCT is a time-consuming and costly affair. From the chemical synthesis of a new substance to the entrance of the product into the market, the process typically takes 12 years and costs $400 million. But, because patent rights last only 20 years, drug developers must cover the lion's share of the cost of developing their products within eight years.

So, as these numbers show, it is highly desirable to reduce R&D costs and save time during the development process. Also, extensive drug quality and safety standards sharply increase documentation efforts—for example, for justifying a drug's composition. Finally, the rapid increase of expert knowledge that must be incorporated into the process, but that is not readily available or easily accessible, aggravates the situation.

This article describes the development of the Galenical Development System Heidelberg (GSH), which aims at giving knowledge-based assistance in one phase of the pharmaceutical development process—namely, the galenical routine development of dosage forms. Galenical development deals with the development of a recipe for a certain drug and its manufacturing technology. The project to develop such as system began in 1990 and is directed by Herbert Stricker, head of the Department of Pharmaceutical Technology and Biopharmaceutics at the University of Heidelberg. It runs in cooperation with the university's Department of Medical Informatics.[1] The GSH system can save pharmaceutical companies considerable time, preserves expert knowledge, and is easy to use for people who are not experts in AI. In addition, the GSH can be used as a training tool for disseminating galenical expertise.

Galenical product formulation

As Figure 1 shows, the development process of a drug product can be divided into several phases: chemical synthesis of a drug substance, chemical and physical characterization of the substance, galenical product formulation for a certain dosage form, preclinical and clinical trials, certification, and product marketing. Galenical product formulation involves adding appropriate excipients and applying procedures to transform a drug substance into a certain dosage form that meets specified properties. Table 1 and Figures 2 and 3 give examples of excipients, procedures, product specifications, and a drug substance. The result of this process is a recipe like the one in Figure 4.

Dosage forms represent different ways of administering and delivering a certain drug substance at its respective target. Well-known dosage forms include aerosols, capsules, granulates, injection solutions, and tablets. In our work, we assume that a cer-

0885-9000/97/$10.00 © 1997 IEEE

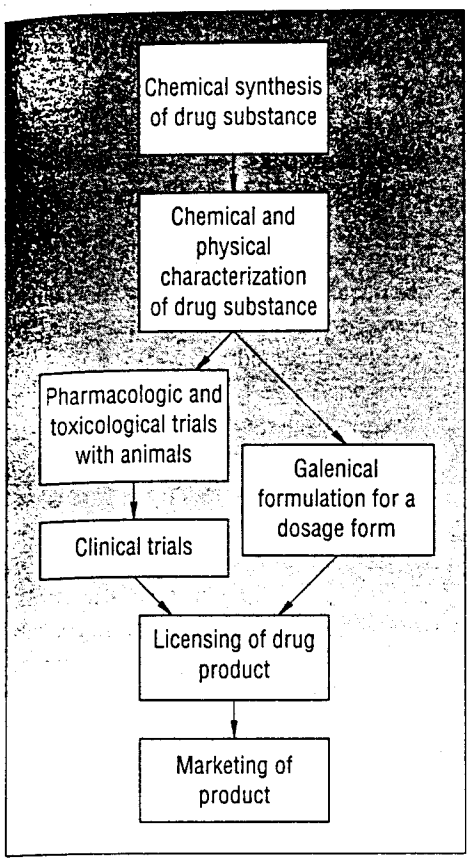

Figure 1. The development process of a drug product.

Table 1. Examples of excipients and procedures used in different dosage forms.

Excipients	Procedures
Ethanol (solubilizer)	Capsule filling machine, intermittent compression filling
Hydrochloric acid, 1 mole (pH corrigent)	Screw mixer, 30 minutes
Lactose (diluent)	Sieving
Magnesium stearate (lubricant)	Steam sterilization
Microcrystalline cellulose (diluent)	Sterilization by filtration
Sodium hydroxide, 1 mole (pH corrigent)	Planetary mixer (3 minutes, 12 rpm, for lubricant)
Sodium lauryl sulfate (wetting agent)	Tumbling mixer (cubic mixer) (15–45 minutes, 12 rpm)
Sodium starch glycolate (disintegrant)	Tumbling mixer (Turbula mixer)

tain dosage form is already determined at the beginning of the galenical development process (see sidebar "Stages in galenical development").

The galenical product formulation process exhibits the typical characteristics of a configuration task (sometimes called *routine design*): Given are a fixed set of elements (*parts*) and the specification of a goal (*constraints*). Then we must choose a subset of the available elements, so that the resulting composition of the selected elements meets the specification. One problem of configuration tasks is the huge number of possible solutions, which is exponential in the size of the set of elements. Another problem is that the elements interact—that is, we cannot choose elements independently. [2,3]

Galenical product formulation is also a weakly structured domain: established methods indicating how to proceed in a certain situation rarely exist. Empirical and heuristic knowledge and the use of qualitative terms prevail over exact models.

Current practice in galenical product formulation

Galenical product formulation is a highly multivariate problem. There are many dependencies between drug substance, excipients, and procedures. Most of them are still insufficiently (or at best only qualitatively) described, which prevents reliable predictions of a composition's properties.

This might be one reason that nowadays galenical development frequently proceeds as follows. To develop the formulation for a new drug substance and a certain dosage form, designers try some standard cookbook recipes and determine their performance: they conduct laboratory experiments and measure the product's relevant properties. These experimental results serve as a starting point for further modifications of the formulations until one recipe has the desired properties. This is a fairly unsystematic procedure and can lead to suboptimal formulations. A product might contain unnecessary components, or the drug product's properties might be close to the given limits, which can cause problems—with long-term stability, for example.

This policy harnesses only a small frac-

Shelf life	5 years
Degradation product (oxidation)	< 0.1%
Homogeneity	< 1% relative standard deviation
Dissolution	> 75%/25 minutes
Capsule size	#1 (0.5 milliliters)
Flow properties (angle of repose)	33° to 45°
Compressibility (Hausner ratio)	1.2 to 2.0

Figure 2. Examples of product specifications for the dosage form *powder mixture in hard gelatine capsule*. The product specifications impose constraints on the admissible drug products.

Particle size (median diameter)	5 micrometers
Angle of repose	43.40°
Bulk density	0.387 g/ml
Tapped density	0.632 g/ml
Solubility (25°C, H_2O)	0.009 mg/ml
Hausner ratio	1.633
Water content	0.2%

Figure 3. Example of some properties of the drug substance Griseofulvin.

Excipients:	/unit
Griseofulvin (drug substance)	150.0 mg
Magnesium stearate	3.5 mg
Microcrystalline cellulose (Avicel pH 102)	99.2 mg
Procedures:	
Capsule filling machine, type 1	
High shear mixer (premix and desagglomeration)	
High shear mixer (main mix)	
Planetary mixer (3 minutes, 12 rpm, for lubricant)	
Packing material: foil (PVC + aluminum)	

Figure 4. Example of a recipe resulting from a run of the GSH with the knowledge base for *powder mixture in hard gelatine capsule*.

Stages in galenical product formulation

We can divide the process of galenical product formulation into three stages (see Figure A):

- During the *preformulation* stage, the developers determine the properties of the drug substance needed for the subsequent stages. These include physico-chemical, technological, and biopharmaceutical properties, and properties characterizing the stability of the drug substance.
- In the *formulation* stage, the developers add excipients and apply procedures to the drug substance to meet the specifications of the final product (see Figure B).
- Finally, *scaling-up* concerns adjusting the formulation as necessary to transfer from a laboratory scale to a production scale.

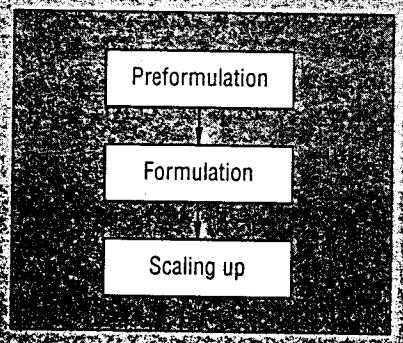

Figure A. The process of galenical product formulation. The preformulation stage determines the properties of the drug substance needed for galenical product formulation. The formulation stage determines the excipients and procedures for production of the dosage form on a lab scale. The scaling-up stage adjusts the formulation results to meet production-scale requirements.

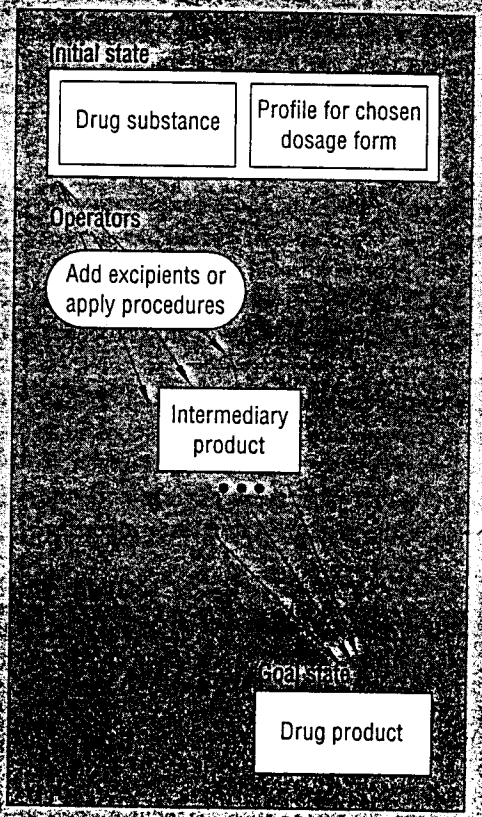

Figure B. Search space for the galenical product formulation problem. The initial state consists of a drug substance (characterized by chemical and physical properties) and a profile for the chosen dosage form (specification of properties the drug product must have). The intermediary product fulfills some of the product specifications. The drug product meets the requirements in the profile.

tion of the existing galenical knowledge. The literature describes many partial, low-order dependencies. However, it is not feasible to keep up with the current state of relevant knowledge—be it due to shortage of time or because the published results are only little pieces of a big puzzle and cannot immediately be used.

Another problem—rooted in galenical formulation being a weakly structured domain—is the empirical, heuristic knowledge of expert technologists. Their knowledge is recorded nowhere. Tapping this source of knowledge would be valuable for pharmaceutical companies preserving the experts' specific knowledge, which is usually lost when they leave the company.

Another problem is that the lack of a clear structure makes it hard for beginners to gain expertise and produce quality results.

Expected gains through the use of a knowledge-based system

As a consequence of the aforementioned drawbacks, our group tried to provide a tool that would help to alleviate some of the problems. In our opinion, giving knowledge-based assistance has the following advantages over the prevailing proceeding:

- a more structured approach to the galenical development makes assumptions explicit, can serve as a base for discussions about how to proceed, and unveils areas of ignorance (lack of or insufficient knowledge);
- valuable knowledge previously residing only in an expert's head can be preserved;
- providing comprehensive, up-to-date knowledge will lead to better utilization of what is known and in consequence to improved formulations;
- the acquisition of expertise for new development staff or students at universities will be facilitated;
- the knowledge-based assistance in the routine development will decrease development time, which will then be available for more complicated problems; and
- this reduction of time will save money.

Project goals

Therefore, the goals of the project are as follows. First, we must provide a model of the galenical product formulation process that applies to all dosage forms. Second, people who are not experts in AI must be able to use the system; they should be able to maintain and extend their knowledge bases without further assistance. This second goal has consequences of importance for the final system:

- The model implemented in the knowledge-based system and its presentation to the user must be as simple as possible.
- The user interface must be easily understandable. This requires using terms familiar to the pharmaceutical technologist.
- Different views on the knowledge base must be possible.
- User interventions must be possible at any time during a development session with the knowledge-based system. Users must be able to enter values measured in the lab that override the values predicted by the system. These adjusted values are then used in further inference steps.
- The system must supply explanations for decisions made during the inference.

These contingencies, prescribed by the intended use of the system, have consequences for the choice of AI techniques. AI has developed very sophisticated methods, which, for example, search a state space very efficiently or which allow integrity checking and control of a knowledge base. In practice, though, many assumptions necessary for the application of these methods do not hold, or they are too complicated to communicate to nonexperts. Therefore, we decided to use relatively simple methods for knowledge representation and inference, because a lack of knowledge in the domain most often poses a

more severe problem than does the use of "inferior" AI methods. (See the "Existing product formulation systems" sidebar.)

Object-oriented analysis model of the domain

In the project's first phase, our group investigated the feasibility of developing a generic model of the galenical development process that would hold for all dosage forms—to identify a core structure of the galenical development. We chose the simple dosage form *intravenous injection solution* as a testbed for the modeling. In parallel, we used a second, more complicated dosage form, *powder mixture in hard gelatine capsule*, to evaluate the model's applicability to other dosage forms.

We started by identifying the domain ontology, describing the common concepts (objects) and their relationships in the domain. This was important for several reasons. It forced the pharmacists working on the project to precisely describe the meaning and use of terms and made previously hidden assumptions explicit. Besides that, it helped the AI researchers understand the domain. The result of this process was an object-oriented analysis model of the domain, which is shown in Figure 5.

We built the object-oriented model using the object-oriented analysis (OOA) methodology of Peter Coad and Edward Yourdon.[4] The advantages of object-oriented models are well-known. They provide an excellent means for structuring a domain, breaking it down into natural parts, and—due to their graphical representation—they facilitate communication between experts and nonexperts. Furthermore, the transition from analysis model to design model and finally to an implementation is fairly easy to make. Changes in the model, especially extensions, are not hard to integrate.

Let us now look closer at the domain model of the galenical product formulation in Figure 5. Excipients and procedures (in the center of the figure) can be grouped to form compound actions. These have at least two components: either two or more excipients or procedures, or a mixture of excipients and procedures. Compound actions are necessary for structuring the domain, because some excipients only make sense in connection with a certain procedure, or vice versa. Moreover, some excipients must be selected in combination—for example, the components of a buffer solution. The three objects—excipient, procedure, and compound action—are all specializations of a generic object called *action*. The link between action and property indicates that we can characterize every action by some properties and their respective values. Attached to a property can be an arbitrary number of formulas used to derive a certain property's value from other properties.

Excipients can be characterized by backbones (the chemical core structure of the excipient's molecule) and functional components (chemically reactive groups). Functional components can be defined at different levels of granularity. Therefore, we allow the definition of *heterarchies* (directed acyclic graphs) of functional components. A backbone itself can be described by its functional components. Backbones, functional components, and actions can describe incompatibilities. Incompatibilities specify adverse interactions between two components that can lead to rejection of a recipe containing them.

The model developed so far represents the domain's *static* structure. It does not make any statement regarding how this is used for solving a given formulation problem.

Design of inference structure

The result of the galenical formulation for a certain drug substance and dosage form—a recipe—contains a subset of all excipients and procedures. This problem of selecting one element from a power set is of exponential complexity in its general, unre-

Existing product formulation systems

The literature describes only a few systems that assist in product formulation. Most systems that directly deal with the galenical development of drug products do not address the whole galenical product formulation process but concentrate on only parts of the process. One system, for example, assists pharmaceutical experts during the preformulation of tablets, proposing compatible excipients and their relative proportions in the final formulation.[1] It does not take into account that machines used for processing the drug substance and the excipients significantly influence the drug product's properties. Should two or more substances, for example, need to be blended to get a homogeneous mixture, different mixers may be utilized. Some are very intensive—that is, they put a lot of stress on the substances—while some are more gentle. If the substance is fragile, those mixers will need to minimize destruction of the drug substance or the excipients.

Other systems help select pharmaceutical powder mixers or identify and remove film-coating defects.[2] Still other systems help in the planning of experiments, statistical analyses, or neural network modeling for gaining insight into dependencies between some parameters.

Innovative materials design is the field of application for the Aladin system, whose domain is the design of aluminum alloys for aerospace applications. Although the system has an elaborate structure with, for example, multispatial reasoning capabilities and meta-planning, an introduction to (routine) practice has not been reported.[3]

The PFES system is a shell tailored for implementing product-formulation problems.[4] It provides facilities to represent the domain knowledge as entities, objects, and their relationships. A hierarchical decomposition of the problem is used with communication between subtasks via a so-called agenda (a kind of blackboard). During a formulation session, the inference engine considers multiple (partial) solutions in parallel. The PFES shell has been successfully used for the development of skin-care products and pharmaceutical tablets. One important drawback of this system is that it has no adequate user interface.

References

1. K.U. Ramani, M.R. Patel, and S.K. Patel, "An Expert System for Drug Preformulation in a Pharmaceutical Company," *Interfaces*, Vol. 22, 1992, pp. 101–108.
2. R.C. Rowe, "Expert Systems in Solid Dosage Development," *Pharmaceutical Industry*, Vol. 55, No. 11, 1993, pp. 1040–1045 (in German).
3. M.D. Rychener, M.L. Farinacci, and I. Hulthage, "ALADIN: An Innovative Materials Design System," in *Artificial Intelligence in Engineering Design, Vol. II, Models of Innovative Design, Reasoning about Physical Systems, and Reasoning about Geometry*, C. Tong and D. Sriram, eds., Academic Press, Boston, 1992, pp. 215–261.
4. B. Skingle, "An Introduction to the PFES Project," *Proc. 10th Int'l Workshop on Expert Systems and Their Applications*, Avignon, France, 1990, pp. 907–922.

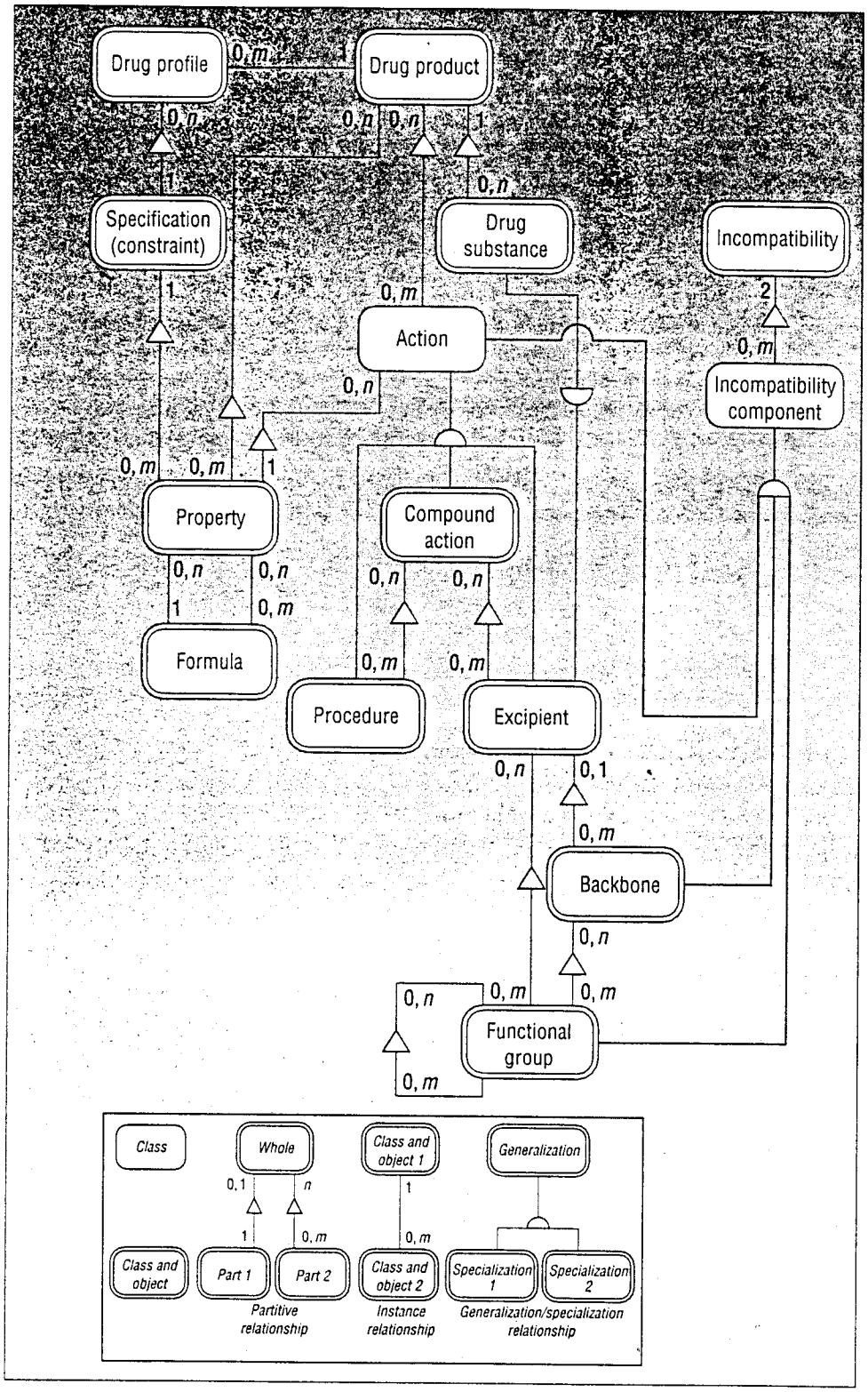

Figure 5. Partial object-oriented analysis model of the domain of pharmaceutical technology. The graph presents the model in a simplified way, leaving out some details. An abstract class has no objects (instances), whereas a class can have objects (instances). In a partitive relationship (between objects), the whole comprises parts 1 and 2, and the numbers represent cardinality restrictions on the number of connections between objects (for instance, the whole has at most one object part 1, which itself belongs to only one object whole). An instance relationship between objects is used like the partitive relationship when the interpretation as a partitive relationship is not adequate. Finally, in a generalization/specialization relationship (between classes), specializations of a class inherit all properties of the superordinate class but might have additional properties.

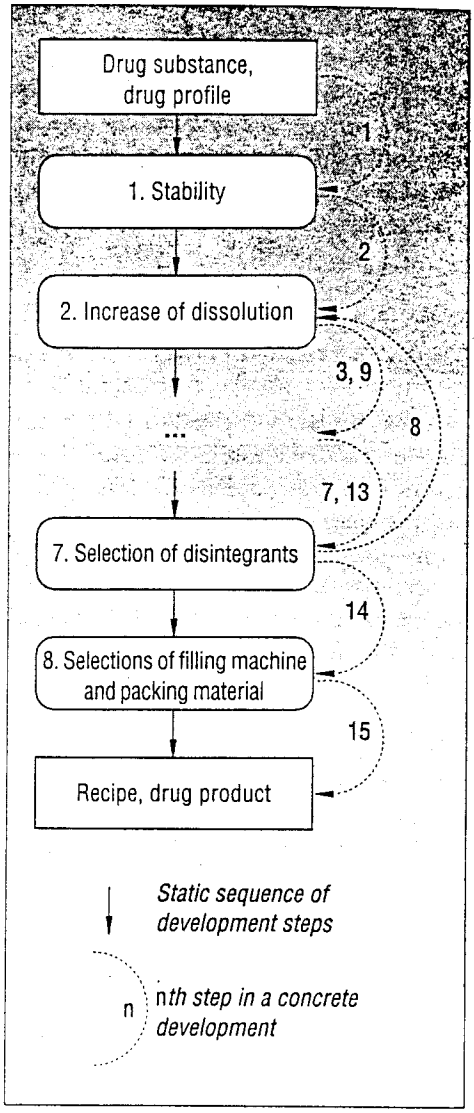

Figure 6. Example of the dynamic development process with knowledge-based backtracking (for the dosage form *powder mixture in hard gelatine capsule*).

stricted form. Therefore, we must use more knowledge from the domain or make reasonable assumptions about the domain to reduce the complexity for a practical inference mechanism.

Decomposing the development process. Because our goals were not only to build a working knowledge-based system but also to provide a structured, systematic approach to the galenical product formulation, it is natural to proceed as follows.

During the development of a dosage form, a pharmacist usually encounters several problems that are typical for this dosage form and that must be solved. Perhaps, for instance, a drug is not sufficiently soluble in water, or a powder mixture does not flow freely enough for use in a capsule filling machine. We therefore decided to conceptually break down the development process into distinct development steps. Every step focuses on one problem and is associated with a subset of the specifications (constraints) for the drug product. A problem is considered solved when its associated specifications are met.

In general, only few of all actions are useful in a certain development step. The knowledge base of our dosage form *IV-injection solution*, for example, contains a development step for the adjustment of the solution's pH value. In this step, we should consider only those excipients that can actually in-

Table 2. Simplified example for the dosage form *powder mixture in hard capsule*: sequence of development steps, corresponding constraints from specifications, and attached groups of actions.

Development step	Constraints defined in specification	Group of actions
1. Check stability	Hygroscopicity, degradation	Unsolvable problems
2. Increase dissolution	Dissolution	Wetting agents
3. Disperse drug	Homogeneity	Premix procedures
4. Select diluents	Bulk density, capsule size	Diluents
5. Select lubricants	Die-wall friction	Lubricants
6. Select mixers	Homogeneity	Mixers
7. Select disintegrants	Disintegration	Disintegrants
8. Select filling machine and packing material	Machine equipment, stability	Capsule filling machines, packing material

fluence the pH, such as buffer solutions, hydrochloric acid, and sodium hydroxide. It would not make sense to consider preservatives or isotonants (which adjust the osmotic activity of a drug solution to the value of the blood plasma) in the context of pH adjustment, because they do not effect the intermediary product in the desired way. Therefore, we associate only those actions with a development step that we consider capable of solving the respective problem. As an example, Table 2 shows the sequence of development steps, the constraints specified for every development step, and the associated groups of actions, for the knowledge base *powder mixture in hard capsule*.

After completing a particular development step, the resulting intermediary product must meet those specifications that are associated with the current development step. However, there are still dependencies between steps. Successfully working through one step by choosing a suitable action might cause a constraint previously satisfied to be violated. So how should the development steps be ordered? Some systems (for example, the PFES) use a dynamic scheduling scheme to activate processes similar to our development steps.

For simplicity, we decided to impose a predefined ordering onto the steps and provide a knowledge-based backtracking mechanism (see Figure 6). For a given dosage form, the most important task the pharmaceutical expert faces is defining the development steps and adequately ordering those steps. The ordering should minimize dependencies between development steps—especially backward influences from later steps on preceding ones.

Our assumptions significantly reduce the complexity of the problem. This is mainly due to the fixed ordering of the development steps, which effectively reduces the maximum depth of the search tree to the number of steps. As a result, the problem is now polynomially complex.

Decomposing a development step. We have described the components constituting the development steps and their linear ordering. We will now explain the internal structure of a single development step. Each step has the same skeletal structure, which needs to be filled with the appropriate knowledge for the respective problem to be solved. Figure 7 shows the complete structure of a development step.

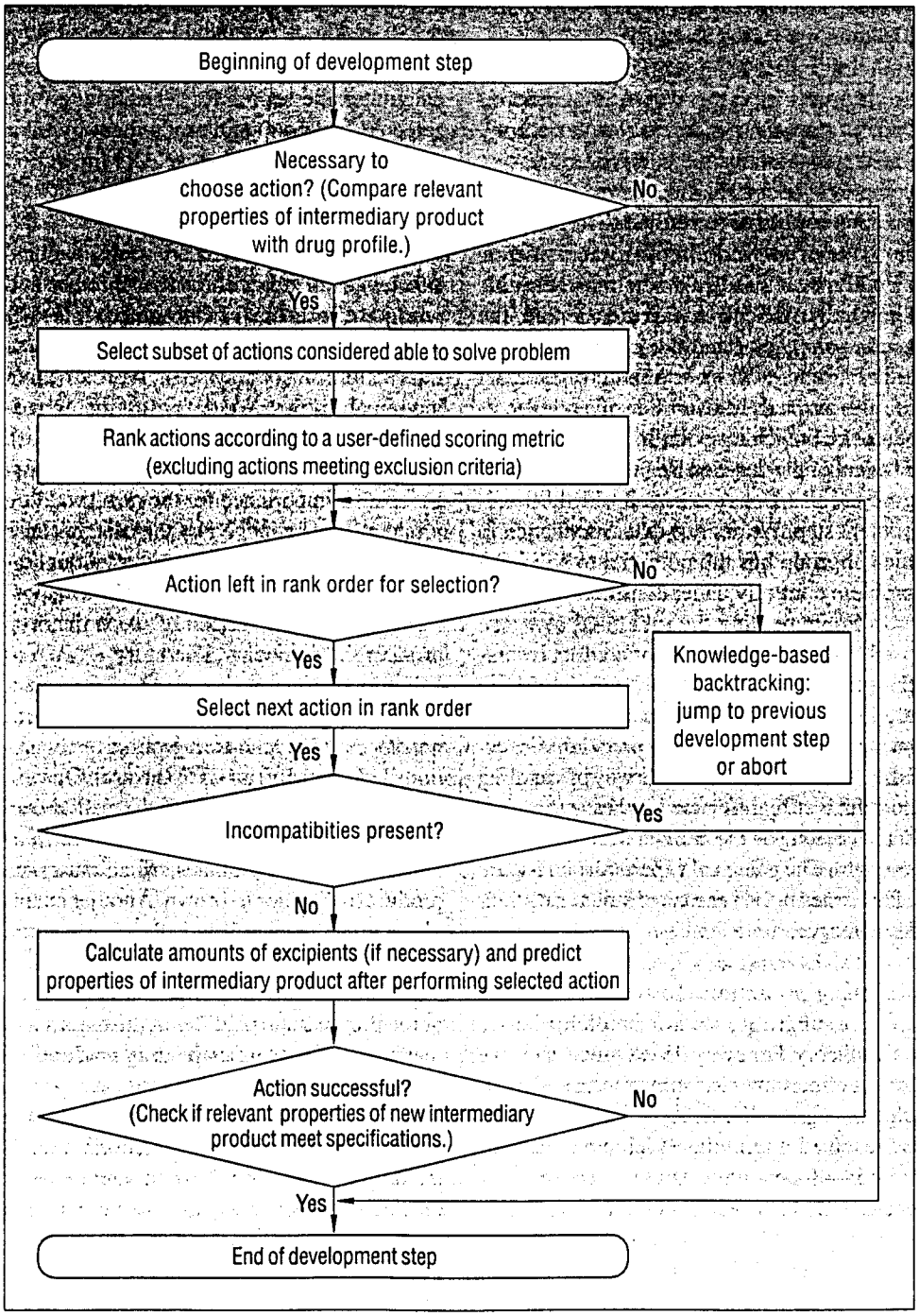

Figure 7. Procedural model of one development step.

Necessity to select an action within a development step. As we explained before, within each development step a certain problem—that is, a subset of all specifications—must be solved without violating any constraints from previous steps. The first decision we face in a development step is whether an action must be selected at all. It could well be that the problem addressed in the current step is not present or has already been solved in preceding steps. The system makes this decision by comparing the predicted properties of the current product with the relevant specifications in the product profile, and jumping to the next development step if none of the specifications are violated.

Subgroups of actions. Within a single development step, the system must select exactly one action—an excipient, a procedure, or a compound action. In most cases, this selection process can be structured to form a decision tree or network. In our system, hierarchically structured rule sets represent this knowledge. Hierarchical means that the action part (right-hand side) of a rule may be either a set of actions or a set of rules. A set of rules as the right-hand side of a rule represents the branching point in a decision tree. The leaf of the decision tree consists of a subgroup of all actions associated with the development step. We assume that every action in the subgroup has the potential to solve the problem currently under consideration.

In implementing the rule interpreter, we used the most simple way of conflict resolution: the rules in a rule set are ordered, and the inference mechanism processes them as such. The first rule with its precondition satisfied fires. To the users, this way of handling conflict is straightforward and easy to understand (as long as the number of rules is small enough). The galenical experts also make use of the ordering as a means of structuring their knowledge.

Selecting an action. To select one action from a subgroup, we use multiattributive value theory. For every development step, we must define some decision criteria and associated weights. We define the constraints to be resolved within the development step as criteria—for example, angle of repose, compressibility, demixing, solubility, and hygroscopicity (for the dosage form *powder mixture in hard gelatine capsule*). Usually, we define additional criteria that do not explicitly occur in the product profile—for example, the physiological compatibility of an excipient, its price, or its availability. Other criteria might refer to constraints that are not crucial to the current development step but that, due to interactions, will probably be influenced by the selection of an action. For every criterion, we must provide a function mapping all values of that criterion into the interval [0, 1]. The value we get after applying the function is a measure of the relative power of a certain action with respect to a single criterion. The weights associated with the criteria indicate their respective importance. All weights must sum to one. The score of an action is the sum of the values of the criteria's functions times the weight associated with the criteria. To select an action, the system calculates the scores for all actions in the selected subgroup and ranks them by their scores. The system then selects the action receiving the highest score.

Incompatibilities. There are some chemical substances or procedures that interact and produce effects that are unwanted. It is necessary to be able to capture those kinds of interactions in our system. We call them incompatibilities. We define an incompatibility as the qualitative interaction between any two actions, backbones, or functional groups. Its importance for the resulting drug product must be judged by the user, because an incompatibility merely makes a qualitative statement that there is some known, possibly harmful, interaction between the drug product's components. Therefore, an incompatibility between a component of the current intermediary product and a new action does not necessarily lead to rejection of the action. This decision is left to the user. One example of an incompatibility is the well-known Maillard reaction (*sugar-browning reaction*) between lactose and amines, which causes the product to discolor to brown. Another example of an incompatibility is the combination of a membrane filter using cellulose acetate or cellulose nitrate that will lead to adsorption of the proteins and peptides at the membrane that are then missing in the drug product.

Calculation of amounts of excipients. We must associate an amount with every excipient in the final recipe. This amount depends on many factors, such as other excipients already contained in the recipe, procedures applied, values of the product profile and values of the intermediary product. We therefore provide the user with a rule-based means for selecting the appropriate formula to calculate an excipient's amount.

Predicting the properties of the (intermediary) drug product. Having selected an action and—in the case of an excipient—calculated its amount in the formulation, we must predict the properties of the intermediary drug product. A rule-based mechanism does this by determining the appropriate function for predicting a property in a certain situation. Our system's rule-based component has only a very restricted working memory. Namely, we only allow that the predicted properties of the new intermediary product (after adding the selected action) be used for predicting other properties. Such properties may play the role of temporary variables. We must demand that there be no cycles in the course of predicting the properties; otherwise, we would run into an infinite regression.

Knowledge-based backtracking. After predicting the properties of the new intermediary product, we have to check whether the specifications in the drug profile are met. At the end of a development step, not only the relevant properties of the current development step, but also all properties focused on in previous development steps, must be within the specifications. (That is, the set of specifications that must be satisfied at the end of a development step is continuously growing while we proceed through the sequence of development steps.) If this is the case, the development step terminates successfully. Otherwise, the chosen action is rejected and the next action tried, according to rank order.

It is possible that no action in the chosen subgroup of actions will lead to successful termination. Should this be the case, we might try to take back previous decisions. We do this via knowledge-based backtracking, using rules to determine the development step to return to. Usually we do not just go back one step in the development process, but rather make use of galenical knowledge to determine why the current step was not successful. We choose a development step that can solve the root of the problem. In Figure 6, we give an example for knowledge-based backtracking taken from the knowledge base for *powder mixture in hard gelatine capsule*. In the development step responsible for the selection of disintegrants, the system encounters a problem with the solubility of the drug substance. Because the developer of the knowledge base has supplied it with the knowledge that solu-

bilizing measures are applied in development step 2, backtracking takes place to this step. When we backtrack to a development step, we retain only the information about the actions taken before that step.

Explanation of the model and the course of inference. One important aspect in the development of a knowledge-based system is the ability of the system to explain its decisions to the user. We have to take into account several aspects of explanation. First, the developer of a certain dosage form constructs a model of the development process—that is, development steps must be identified and ordered. In our system, the designer of a knowledge base must provide a textual explanation for the distinct development steps and their ordering. This is part of the static view of the knowledge base.

Second, the explanation of the results of a development session with the knowledge-based system gives a dynamic view of the knowledge base. It is basically a trace of the steps the system has taken during the inference process. The user can control the granularity of the explanation the system provides. This explanation can be as compact as the plain recipe and the drug product's predicted properties. If needed, however, the system will also generate a very detailed explanation, including information such as the rules activated, the formulas used, or the individual scores of actions in the selected subgroups.

A rationale for the knowledge and decisions is provided through canned text and references to the knowledge sources (for example, original articles, lab experiments, and textbooks). For practical purposes, this seems to be sufficient, due to the system's purposely simple, easy-to-understand structure. Rules in our knowledge-based system always have a clearly defined context, so that they are fairly self-explanatory. In case more information is needed, the most important aid seems to be the original literature. Another advantage to our simple way of providing explanations is that they can easily be handled by the pharmacists maintaining the knowledge base. More sophisticated methods would not be feasible in routine work.

Implementation

We developed the GSH on a PC running under Windows 3.11, using Digitalk's Smalltalk/V for Windows. It needs at least 8 Mbytes of RAM and a 486 PC. We chose Smalltalk as a development tool for several reasons: it is an object-oriented language suited as well for rapid prototyping as for production versions of a program. It comes with a graphical user interface and tools for designing graphical user interfaces. Although it uses a lot of memory and probably does not give the fastest running programs, it is easy to work with. Additionally, during the last years, the tools and Smalltalk compilers available have improved significantly. Because the design of our knowledge base and our inference mechanism is not very complex, we did not have to use sophisticated rule interpreters—for example, with many-to-many pattern-matching mechanisms.

State of the project

Our group is currently working on the knowledge bases for eight dosage forms. We have completed the knowledge bases *aerosols*, *intravenous injection solutions*, *powder mixtures in hard gelatine capsule*, and *tablets (direct-compression)*. The knowledge bases *coated forms*, *granulates*, *lyophylisates*, and *pellets* are in different stages of development.

We presented the GSH to several pharmaceutical companies to explain our approach and encourage external tests with the available knowledge bases. The aim of those activities was mainly to test, in principle, the feasibility of our structured approach. The recipes proposed by the system were well-received by the galenical experts. In December 1996, the GSH was first introduced into practice in a pharmaceutical company. Preliminary results from this field trial are expected in the first quarter of 1997.

Evaluation

Evaluating the GSH is not easy. We must view the system's performance along several axes. First, we have to ask whether the model of the galenical development process is adequate. Intensive discussions and tests have convinced us that in general this is the case for the dosage forms we are currently working on. We do not expect any fundamental changes in the model to be necessary, although several extensions are needed to better accommodate knowledge about more complex dosage forms. The structuring of the domain by introducing the concept of a *development step*, which takes care of a certain problem, has proven to be very useful.

A second dimension with which to evaluate the system would be to compare the predictions of the GSH with the results of lab experiments. This is very time-consuming, and we've only done it with a few drug substances and profiles. Those experiments gave us positive but only casuistic information about the system's performance. Generalizing the results is barely possible; there is no way to completely and systematically evaluate the knowledge base. In our opinion, the GSH knowledge base can be judged only subjectively by its performance in practical use, taking into account the quality of the predictions of the products' properties and whether the GSH actually reduces the time necessary for the galenical development of a dosage form.

The last dimension evaluates how the system compares with the conventional procedure. This is the most difficult point. Differences between recipes are hard to quantify. It is not clear how to measure the *quality* of a recipe and its corresponding drug product. As for now, we leave this judgment to the experts. Our system cannot guarantee *optimality* of recipes. However, what it can do is quickly provide the user with a good initial recipe that can be further refined.

Uncertainty

We have not yet addressed the representation of uncertainty in our system, in this article. Briefly, we attach a value between 0 and 1 to each property value and formula. These values are propagated using the arithmetic minimum rule. We do not use the values for any decisions. They only serve as indicators for the credibility of a fact or a derived value in terms of the weakest element in a chain. The disadvantage of this approach is that the uncertainty inherent in the predictions of intermediary products' properties is not taken into account adequately. In the future, we might extend the representation of uncertain knowledge if the gain seems large enough.

When we make predictions for the values of the properties of an intermediary product, we do not allow qualitative values—that is, statements such as *action X increases the pH of the product*. However, we expect in more complex dosage forms that the knowledge available will become sparse and will not be

expressible in quantitative relationships. This would then call for ways to express qualitative knowledge and to integrate qualitative and quantitative knowledge.

HAVING SUCCESSFULLY ESTABlished a generic model of the galenical development process and implemented knowledge bases for four different dosage forms, we can draw several conclusions from our project. For starters, capturing the terminology of the domain and structuring the domain by means of object-oriented analysis methods, although not a panacea, is very helpful for gaining a clearer view of the subject area. Decomposing the problem-solving process provided a valuable guideline for practical work.

The linear ordering of the development steps might seem a bit too restrictive. But discussions of alternatives—for example, considering several development steps in parallel—suggested that the resulting increase in the model's complexity would not be compensated by the advantages of this approach.

Working together with individuals who are not experts in AI, we learned that it is important not to focus on sophisticated AI methods too much when designing a knowledge-based system (especially if the domain experts are to develop and maintain the knowledge bases, largely unaided by AI experts). Many techniques for modeling knowledge can be great in theory; but, in practice, collecting the detailed information necessary to fill the knowledge base is often impractical. This might be one reason that many knowledge-based systems did not make it out of the lab. In our special case, a lack of knowledge frequently seems to pose a more severe problem than some theoretical deficits due to the use of simple methods.

Of great importance is the need to provide users with capabilities for browsing the knowledge base to explore the knowledge built into the system. This helps users judge the system's decisions, because they can evaluate their own knowledge against the system's knowledge base. Finally, users usually do not care much about the models underlying a system: they want a tool they can use easily and that actually aids them in their daily work.

We have shown that the knowledge model of the domain of pharmaceutical technology is sufficient for creating knowledge bases for several dosage forms. However, as more complex dosage forms are developed, we are considering some extensions to the model. The biggest problem is that the dosage forms we are now working on depend much more on machines and settings of the machines' parameters than did the previous ones. Usually, machines (for example, a fluidized bed granulator) can be controlled by several parameters (such as the rate of air flow and temperature). Changing the settings of these parameters changes the resulting product's properties. The GSH represents machines as procedures, but there is no means for representing the machines' parameters and determining appropriate settings. Therefore, we must assume that the values of parameters are fixed (see the examples in Table 1). It is therefore necessary to extend the model to allow for the representation of machine parameters.

We have been able to reuse parts of knowledge bases for several dosage forms. We can reuse backbones, functional groups, and the definitions of units and conversions between units (not shown in the object-oriented analysis model) in virtually every dosage form. For some dosage forms, we can reuse certain development steps; the same holds for some excipients and, to a lesser extent, procedures.

The results of the development sessions with the GSH are stored in a database but not used by the system any further. During a session, experimental results can replace the values of the product's properties predicted by the system. Should there be significant discrepancies between the system's predictions and the experimental results, this information can be a valuable source for revising the content of the knowledge base.

Acknowledgments

The GSH project described here represents the work of a larger group of people. We wish to express our thanks to the rest of our group: Martin Bultmann, Gordon Detka, Stefan Fuchs, Christoph Gröbel, Reinhold Haux, Frank Lintz, Tilman Rock, Reinhold Rößler, Herbert Stricker, and Stefan Wiegel. Our intensive discussions helped us to reach a better understanding of the project. Thomas Wetter provided his AI expertise in the initial phase.

References

1. H. Stricker et al., "The Galenic Development System Heidelberg/Systematic Development of Dosage Forms," *Pharmaceutical Industry*, Vol. 56, No. 7, 1994, pp. 641–647 (in German).
2. S. Marcus, J. Stout, and J. McDermott, "VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking," *AI Magazine*, Vol. 9, No. 1, Spring 1988, pp. 95–112.
3. J. Mostow, "Toward Better Models of the Design Process," *AI Magazine*, Vol. 6, No. 1, Spring 1985, pp. 44–57.
4. P. Coad and E. Yourdon, *Object-Oriented Analysis*, 2nd ed., Prentice Hall, Upper Saddle River, N.J., 1992.

Jürgen Frank is a doctoral candidate in the Department of Medical Informatics at the University of Heidelberg, and he joined the GSH project in July 1994, refining the GSH model and implementing part of the system. His research interests include knowledge modeling, human-computer interfaces, sensitivity analyses in knowledge-based systems for design problems, and theory and applications of Bayesian networks. He graduated from the University of Heidelberg/Polytechnical School Heilbronn in May 1992 with a diploma in Medical Informatics. Readers can contact Frank at the Department of Medical Informatics, University of Heidelberg, Im Neuenheimer Feld 400, D-69120 Heidelberg, Germany; juergen.frank@urz.uni-heidelberg.de.

Birgit Rupprecht has been working at the University of Leipzig at the Institute of Medical Informatics, Epidemiology and Statistics in the Hospital Information System section since April 1996. She graduated from the University of Heidelberg/Polytechnical School Heilbronn in May 1992 with a diploma in Medical Informatics and received her doctorate from the University of Heidelberg in April 1996. She worked in the GSH project between 1992 and 1996, doing substantial parts of the knowledge modeling and implementation work. Her research interests are conventional and digital optical archiving of medical records, documentation and classification of diagnoses and procedures, models and methods for configuration problem solving, and quality management in health care. Readers can contact Rupprecht at the University of Leipzig, IMISE, Liebigstr.27, D-04103 Leipzig, Germany; birgit@imise.uni-leipzig.de.

Veit Schmelmer is a pharmacist and belonged to the group that started the GSH project and was extensively involved in the modeling process. He developed the knowledge base for powder mixtures in hard gelatine capsules. In July 1995, he received his PhD in pharmaceutics from the University of Heidelberg. He is heading a development laboratory in the pharmaceutical industry. His research interests include the structuring of galenical knowledge and its representation in knowledge-based systems as well as the absorption enhancement of poorly soluble drugs. Readers can contact Schmelmer at Dr. Karl Thomae GmbH, Pharmaceutics Department, Birkendorfer Str. 65, D-88397 Biberach/Riss, Germany; veit.schmelmer@bid.de.