PERGAMON

Computers & Geosciences 29 (2003) 291–300

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

Reactive transport in surface sediments. I. Model complexity and software quality

Filip J.R. Meysman[a,*], Jack J. Middelburg[b], Peter M.J. Herman[b], Carlo H.R. Heip[b]

[a] *Marine Biology Department, Ghent University, K.L. Ledeganckstraat 35, Gent 9000, Belgium*
[b] *The Netherlands Institute of Ecology (NIOO-KNAW), Korringaweg 7, Yerseke 4401 NT, The Netherlands*

Received 1 December 2001; received in revised form 22 May 2002; accepted 22 May 2002

Abstract

Analysis of three recent diagenetic model codes (OMEXDIA, CANDI and STEADYSED) revealed that codes have a rigid, static and problem-specific character, leaving little autonomy for the application user. The resulting lack of flexibility and extensibility, and the associated need for ground-level reprogramming, constitutes a major barrier for potential model users. Present codes have apparently passed a critical threshold of code complexity, above which code development becomes time-consuming and expensive using the present procedure-oriented techniques. We have explored the advantages of object-oriented technology and the concept of a problem-solving environment to improve the quality of software for reactive transport modelling. A general blueprint for an object-oriented code for modelling early diagenesis is presented. The MEDIA environment consists of a toolbox of building blocks (element, species and process objects), which can be combined freely by the user to construct new models (without the need for recompilation). An object-oriented database stores current objects and accommodates new user-defined building blocks. Altogether, it is advocated that by improving the software quality, one can substantially lower the threshold for using model codes as an integrated data-analysis tool.

Keywords: Reactive transport modelling; Early diagenesis; Object-oriented design; Problem-solving environment; Software quality assurance

1. Introduction: model complexity and software quality

Ever since the quantitative study of reactive transport in surface sediments emerged, the discipline—known as early diagenetic modelling—has experienced a progressive evolution towards more complex model formulations. Prior to the availability of the necessary computing power, models were restricted to analytical solutions, and by necessity, model development was kept relatively simple (Goldberg and Koide, 1962; Berner, 1964; Guinasso and Schink, 1975; Boudreau and Canfield, 1988, 1993; Aller, 1990; Boudreau, 1991). A second wave of models pioneered a numerical approach, and could handle depth-dependent transport parameters and non-linear kinetics, though still focused on particular processes or zones in the sediment (Gardner and Lerche, 1987,1990; Rabouille and Gaillard, 1991; Blackburn et al., 1994; Dhakar and Burdige, 1996). Recently, a third generation of early diagenetic models emerged, which explicitly include all redox zones in the sediment and which incorporate an extensive species and reaction set (OMEXDIA, Soetaert et al., 1996a; STEADYSED, Van Cappellen and Wang, 1996; CANDI, Boudreau,

*Corresponding author. Present address: The Netherlands Institute of Ecology (NIOO–KNAW), Korringaweg 7, Yerseke 4401 NT, The Netherlands. Tel.: +31-113-577-489; fax: +31-113-573-616.
E-mail address: f.meysman@nioo.knaw.nl (F.J.R. Meysman).

1996). Corresponding model applications (Soetaert et al., 1996b; Wang and Van Cappellen, 1996; Boudreau et al., 1998) illustrated the potential of these complex state-of-the-art numerical model codes. They form an efficient instrument for testing hypotheses and a valuable tool to check the consistency of available biogeochemical datasets.

Other reactive transport disciplines, such as subsurface sediment models (groundwater geochemistry, contaminant hydrology, petroleum engineering) showed a similar evolution towards more sophistication. Hence we can investigate early diagenetic models as a proxy for other reactive transport disciplines. Definitely, more complex models have proven beneficial, grasping progressively more of the biogeochemical reality in the sediment. However, the increase in model complexity also comes at a cost. This evolution has affected all three key components of the modelling process: (1) the *mathematical model* (a simple measure of *model complexity* could be the number of species and reactions incorporated), (2) the *numerical solution* procedure (a simple measure of *solution complexity* could be the number of steps in the algorithm by which the model equations are solved) and (3) the *model code*, which implements both the model equations and the solution algorithm (a simple measure of *code complexity* could be the number of lines or subroutines in the computer code). More complex models require a more sophisticated solution procedure to alleviate their excessive demand for computational resources. These procedures, combined with additional requirements for data processing and output visualization, have in turn increased the complexity of the corresponding computer codes. With the increasing complexity of the model development process, two downsides have become apparent.

Firstly, numerical model development has evolved into a rather specialized discipline with the unfortunate consequence that models themselves have become somewhat "inaccessible" to the non-modelling geochemist. Opposite to simple analytical models, which are part of any scientist's mathematical background, the development of a state-of-the-art numerical model code requires considerable skills in numerical analysis (selecting an efficient solution procedure) and computer programming (developing the actual computer code). A division of labour has emerged between model application developers (the "modelling community") and model application users (the "data community"). At present, the tools developed by the modelling community do not easily transfer to the data community. The fact that most model applications are presented by the actual authors of the codes is a clear indication of this (e.g. Soetaert et al. (1996b), Boudreau et al. (1998) and Wang and Van Cappellen (1996) for the OMEXDIA, CANDI and STEADYSED codes, respectively). The usual data-analysis tool of the geochemist and the microbial ecologist remains a spreadsheet and when publishing datasets, the observational scientist still reverts to back-of-the-envelope calculations and simple analytical models. Therefore, a grand challenge is to bring the computing power of numerical model codes within reach of the non-model–developing geochemist.

As the complexity of modelling applications increased, a second "gap" has become more and more apparent, i.e. that between "software engineers" and "scientific modellers". Until now, the attention of model developers has been primarily focused on the first two components of the modelling process: (1) the actual construction of mathematical models and (2) the selection of efficient numerical solution techniques (Boudreau, 1997 is a basic reference in the area of early diagenesis). The translation of both model and solution method into a computer program has been granted far less attention. Unlike experimental papers — which invariably include a section on material and methods — modelling papers do not provide information on the design and structure of the model code. Briefly stated, software design and software quality assurance (flexibility, extensibility, etc.) are presently not a great concern to reactive transport modellers.

Nevertheless, we believe that proper code design will become vitally important as the complexity of the model applications will continue to increase in the nearby future. A recent personal experience is particularly illustrative at this point. In the framework of the ECOFLAT project (Herman et al., 2001), we aimed at enhancing the pH modelling capacities of the existing generation of early diagenetic models. The initial idea was to avoid unnecessary code development, and hence, our intention was to adapt/extend the source code of an existing model. Therefore, we downloaded the source code of the OMEXDIA (Soetaert et al., 1996a), CANDI (Boudreau, 1996) and STEADYSED (Van Cappellen and Wang, 1996) models and evaluated their source code (Table 1 provides a comparison of these three diagenetic model codes). Unfortunately, it became soon clear that these source codes were rigid, hard to "penetrate" and difficult to adapt for an outsider. Eventually, it turned out to be more efficient to develop a new model code from scratch (the MEDIA environment on which we report later). The "frustration" of not being able to amend existing software is the driving force behind this communication. Yet, this failure cannot be attributed to a coincidental finding of sub-standard programming. Conversely, the fact that all codes suffered the same problem clearly pointed at a structural problem in present reactive transport codes. Rather, one must acknowledge that it is inherently difficult to build a "transferable" code, especially when model formulations and model codes become increasingly complex. Unlike software engineers, in the standard natural sciences curriculum, one only receives a superficial

Table 1
Comparison of three early diagenetic model codes: OMEXDIA (Soetaert et al., 1996a), CANDI (Boudreau, 1996) and STEADYSED (Van Cappellen and Wang, 1996). Differences in model structure, numerical solution and source code are shown

	OMEXDIA	CANDI	STEADYSED
Model structure			
Modeling aim	Benthic-pelagic coupling	Nutrient cycling	Fe–Mn cycling
Chemical species	9	27	29
Irreversible reactions	6	20	23
Reversible reactions	0	8	4
Transport processes	Advection, porewater diffusion, diffusive bioturbation	Advection, porewater diffusion, diffusive bioturbation, bio-irrigation	Advection, porewater diffusion, diffusive bioturbation, bio-irrigation
Diffusion of total species	ODU	ΣCO_2, ΣNH_4,ΣH_2S, ΣPO_4	Not applied
Bioturbation	Intraphase mixing	Intraphase mixing	Interphase mixing
Electron acceptors for organic matter oxidation	O_2, NO_3, ODU	O_2, NO_3, MnO_2, FeOOH, SO_4	O_2, NO_3, MnO_2, FeOOH,SO_4
Explicit formulation of the Mn and Fe redoxcycles	No	Yes	Yes
Precipitation-dissolution	Not applied	FeS, FeS_2	FeS, $MnCO_3$, $FeCO_3$
Dissociation reactions	Not applied	ΣCO_2, ΣNH_4,ΣH_2S, ΣPO_4	ΣCO_2, ΣH_2S
Adsorption reactions	NH_4	ΣNH_4, ΣPO_4	NH_4, Fe, Mn
pH model	Not applied	Via charge balance	Via proton balance
Solution method			
Discretization in space	Finite differences	Finite differences	Finite differences
Steady-state model	Implicit/explicit	Explicit	Implicit
Steady-state solution approach	Newton–Raphson	Method of lines	Thomas algorithm
Linearization necessary	No	No	Yes
Dynamic model available	Yes	Yes	No
Dynamic solution approach	Method of lines	Method of lines	Not applied
Use of a stiff diff. equation solver	VODE	VODE	Not applied
Reversible reactions	Not applied	Charge balance	Local equilibrium assumption
pH calculation	Not applied	Charge balance	Proton condition
Source code			
Programming language	Fortran 77	Fortran 77	Fortran 77
Subroutines	85	15	10
Lines	6561	4479	4410
Average subroutine length (Lines)	77	299	441
CPU time (PII 450 Hz) steady-state run	Seconds	20 min	10–30 min

training in code development, which is then predominantly focused on computational issues. Hence, scientific modellers are basically unfamiliar with the issues on how to build user-friendly and high-quality software systems.

Here, we want to investigate (1) how we can significantly improve the software quality of numerical diagenetic models (and by extension, reactive transport codes) and (2) how this may lower the threshold for using them as an integrated analysis tool. Our ambition is to show that by closing the one gap between "software engineer" and "modellers", and thus by improving the software quality of the codes, the other gap between "modelling" and "non-modelling" scientists can be equally reduced. We will first review the conventional way of numerical code development (Section 2), while subsequently we will discuss the potential of problem-solving environments (PSE) (Section 4) and object-oriented technology (OOT) (Section 5) to enhance the software quality of reactive transport codes.

2. An assessment of conventional model code development

In order to expose the limitations of the current approaches, we have carried out a critical analysis of the OMEXDIA, STEADYSED and CANDI model codes (Table 1). The results are presented here in a general fashion, as to bear relevance to related fields of reactive transport. The analysis follows the three steps that are conventionally recognized in the software development process (Rumbaugh et al., 1991; Martin and Odell, 1992; Booch, 1994; Meyer, 1997; see Fig. 1).

2.1. Requirement analysis

In the present view of model code development, little effort is devoted to the requirement analysis step. Generally, no explicit statement is made of the specifications and tasks the model code eventually should fulfil. Moreover, specifications are generally narrow and rigid, and model codes tend to have a fixed model formulation. The set of species, reactions and transport processes is predetermined at the start of model code development, and as a consequence, the model formulation is rigidly embedded in the model code (as in the case of OMEXDIA, CANDI and STEADYSED, Table 1). This one-to-one relation between model and model code is referred to as the 'one model-one code' approach. Its antagonism is the 'multimodel-one code' approach, where multiple models can be solved within the same application without modification of the underlying source code. Fig. 2 compares both approaches.

The basic problem with the "one model-one code" approach is the *lack of flexibility*, i.e. model codes cannot cope well with change. By fixing the model formulation during code development, one ignores the crucial fact that the scientific question, for which model and model code were initially constructed, can be subject to *evolution*. As science progresses, new mechanisms and processes are discovered, and therefore, reactive transport models should allow an easy update to this newly acquired knowledge. For example, in the context of early diagenesis, new reaction mechanisms have been discovered only recently, such as a new way of pyrite formation (Drobner et al., 1990; Rickard and Luther, 1997), and the disproportionation of thiosulphate (Jørgensen, 1990). Secondly, when the application developer fixes the model formulation at the time of code development, little autonomy remains for the application user. In the case of OMEXDIA, CANDI and STEADYSED codes, the only "modelling freedom" left is to shut down processes by setting the corresponding parameters equal to zero (e.g. zeroing a kinetic

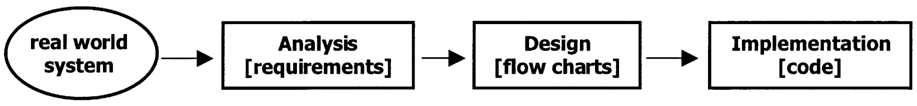

Fig. 1. Three steps in software development process. In analysis phase, important characteristics of problem are recognized and a simplified model is created of real problem to be tackled. In design step, outline of software package is generated based upon resulting understanding of model (typically by means of diagrams and charts). Finally, in implementation step, design is finally programmed into code using some specific programming language.

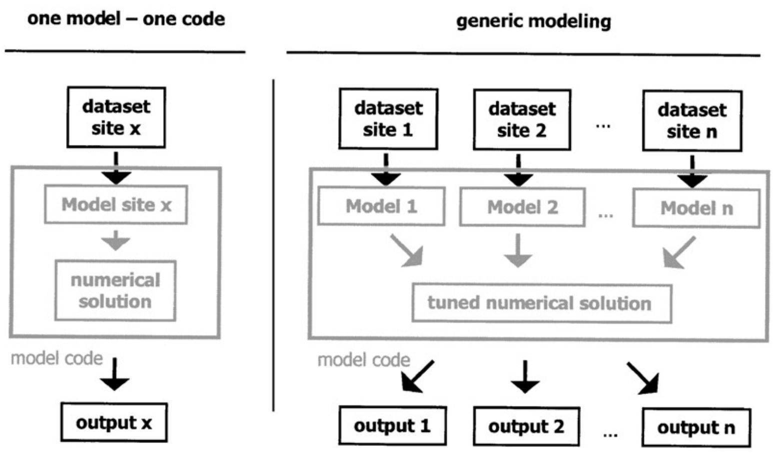

Fig. 2. Comparison of 'one model-one code' approach and the generic 'multimodel-one code' approach. In conventional 'one problem-one code' approach, application developer decides on model formulation (i.e. model formulation is fixed at compile time). Logical improvement would be to let application user decide on model formulation (i.e. model formulation should only be stated at runtime). In 'multimodel-one code' approach, multiple models can be solved within same computer code i.e. without recompilation.

reaction constant). However, such a "simplification" of the embedded model does usually not diminish the computational demand and memory reservation, and thus effectively results in performance degradation.

The lack of flexibility of a model code is almost invariably accompanied by a *lack of extensibility*. A predefined model formulation severely limits the type of problems to be investigated. Applying the same reactive transport code to a different environment almost invariably requires the inclusion of additional reactions and new species. However, this usually invokes a modification of the source code beyond the skills and the background of the average non-modeller. The resulting need for reprogramming and recompilation therefore constitutes a major barrier for potential model users.

The implications of a lack of flexibility and extensibility depend on the level of complexity of the model application. When dealing with simple applications, one can afford to throw them away and replace them with entirely new software rather than attempt to reuse them, repair them, or extend their functionality (Booch, 1994). Oppositely, for complex applications, a large investment of resources is allocated to the development of the software system, and hence, a different view on software is necessary. Such a paradigm shift has been particularly well documented in the area of industrial software development (e.g. financial administration, aviation) as code complexity increased due to the falling costs in computational resources (Cox, 1987). During the 1980s a radical transition took place, which resulted in the abandoning of older programming techniques, such as the procedure-oriented approaches (see below for details), and which ultimately led to the adoption of the object-oriented paradigm (Stroustrup, 1988; Rumbaugh et al., 1991; Coad and Yourdon, 1991; Jacobson et al., 1992).

The current situation in early diagenesis (and in reactive transport modelling) clearly mimics the transition observed in industrial software engineering. Previous generations of diagenetic model codes were only modest in size and required a moderate investment in programming. However, the increasingly large model applications we are developing today (which require months, or even years of developing/debugging/testing) cannot be treated as disposable programs. Consequently, the drawbacks of the "one model-one code" are becoming more and more apparent, and the approach can no longer be sustained. To achieve a broad range of applicability, the flexible adaptation of the species and reaction set must become a crucial part of the model code specifications. The "multimodel-one code" approach promotes such software qualities, as different models are solved without the need for accessing the underlying program code.

2.2. Design

The OMEXDIA, CANDI and STEADYSED codes are typical examples of a code whose structure is determined by the *procedure-oriented approach*. This approach became influential in the 1970s, and is built on both the concepts of structured programming and the method of top-down design (e.g. Yourdon and Constantine, 1979; Dahl et al., 1972). *Structured programming* was directly influenced by the topology of traditional high-order programming languages, such as ALGOL, FORTRAN and COBOL (Booch, 1994). The fundamental unit of decomposition is the task or process, which is coded as a subroutine, and the program takes the shape of a treelike network, in which subprograms perform their work by calling other subprograms at a lower level. *Top-down design* takes advantage of procedural decomposition to reduce the complexity of a given problem. Top-down design directs designers to start with an abstract description of the system's function, and then to refine this view through successive steps, decomposing each subsystem into a small number of simpler subsystems, until a sufficiently low level of abstraction is achieved to allow direct implementation. In the field of industrial software engineering, procedure-oriented techniques have received a significant amount of criticism in recent years (Booch, 1994; Meyer, 1997; Page-Jones, 2000). There is a threshold of code complexity (estimated at $\sim 10^4$ lines), above which procedure-oriented techniques rapidly lose their benefits. In early diagenesis we are presently approaching/passing this critical threshold (see Table 1).

As the code size increases, procedure-oriented design makes software time-consuming to develop and expensive to maintain (e.g. Lientz and Swanson, 1981). Due to the low level of abstraction, rigid and monolithic codes are produced, which are hard to modify or extend (even by the code author). When the application user wants to add components to an existing model, it becomes a daunting task to modify the source code, and sometimes a major re-implementation of the model code is necessary. Consequently, codes suffer from a limited lifespan, as it is often more practical to develop a new code, than to repair or extend the old source code. This is exactly what we experienced in the ECOFLAT project. Rather than modifying the OMEXDIA, CANDI or STEADYSED code, it proved more efficient to develop an entirely new code (the MEDIA code, see later).

2.3. Implementation

Until present day, FORTRAN—one of the oldest high level programming languages—makes up the dominant programming language in the field of diagenetic modelling. As most code developers have received their formal training in Fortran77, most model codes are still implemented in the Fortan77style. The Fortran77 syntax is completely attuned to the procedure-oriented way of structuring programs. Its dense and obsolete programming style, exemplified by the deep nesting of control statements and the ample use of statement labels, enforces the rigidity of the model code and obscures the overall structure. In addition to this, the source code is often poorly documented, which greatly hampers the extension and modification of the code. Even when codes are programmed in Fortran 90/95, the modern features (data-abstraction, information hiding, operator overloading, etc.) of this programming language are not often extensively explored.

3. New approaches to model code design

For large software applications, such as the present-day reactive transport codes, the development cost of the model codes is a dominant factor in the overall cost of the research. The challenge therefore is to develop new concepts, which (1) reduce the need for modification and extension of model codes as much as possible and (2) when modification is unavoidable, enable fast and flexible adaptation. In order to meet these criteria, the transfer of two techniques from the field modern software engineering could prove highly valuable. Here we will discuss two techniques with regard to their potential for reactive transport modelling. (1) The concept of a problem-solving environment PSE enables model construction at a higher level within the model code (Ford and Chatelin, 1987; Gallopoulos et al., 1994). (2) An *object-oriented approach* (OOA) enables the development of modular and extensible code, and thus allows the flexible adaptation of the model code (Booch, 1994; Page-Jones, 2000). Both techniques were implemented in the development of MEDIA (Modelling Early DIAgenesis), a software package for biogeochemical simulations in marine surface sediments.[1] The MEDIA software was developed in the framework of the ECOFLAT project (Herman et al., 2001) and is an attempt to put the recommendations of the previous sections into practice. Details on the model formulation, numerical solution methods, verification and validation within MEDIA are presented in a companion paper (Meysman et al., 2002).

4. Problem-solving environments

Rather than focusing on a single model, a PSE is a computer application that provides all the computational facilities necessary to solve a *target class* of related problems (Ford and Chatelin, 1987; Gallopoulos et al., 1994). These features include flexible model construction, advanced solution methods and ways to easily incorporate novel solution methods. The two central goals of a PSE are (1) to enable more people to solve problems more rapidly, and (2) to provide many people with the possibility to do things they could not do otherwise (Gallopoulos et al., 1994). An important requirement states that a PSE should use the "natural language" of the target class of problems, so users can run them without specialized knowledge of the underlying computer software. The development of PSEs for different kinds of applications is an active and promising research field in computational science (e.g. Akers et al., 1997; Parker et al., 1997; Knox et al., 1997; Houstis et al., 1998; Fujio and Doi, 1998). An overview is given in the reviews of Rice and Boisvert (1996) and Appelbe and Bergmark (1996) and the references therein. Up to present, most PSEs have a broad application domain and are built to solve broad classes of partial differential equations (e.g. Ahlander, 1999).

So, in the situation of a reactive transport, the PSE should use a "geochemical" vocabulary close to the natural language of the geochemist using the model. The aim of the MEDIA-project was to develop a PSE that solves reactive transport equations for chemical species in surface sediments (see Berner, 1980; Boudreau, 1997; Meysman, 2001 for the actual form of these so-called diagenetic equations). Implementing the recommendations from the previous sections, two important criteria were set for MEDIA.

The first criterion was *flexibility*: the application user should be in control of the model formulation. More precisely, it should be possible to assemble a diagenetic model from a toolbox of available model components (species, processes) without the need for writing a single line of new code. Within MEDIA this criterion was met by means of *generic model construction*. Generic model construction basically transfers the control on the model formulation from the application developer to the application user. The basic idea behind it is not to fix models at *compile time*, but to construct them at *run-time* from a set of basic building blocks. As a consequence, model formulation is not fixed a priori by the *program developer*, but is decided upon by the *program user* at run-time. The user then specifies a particular set of basic building blocks, which are subsequently assembled by the computer program into a costumer-tailored model. The modular building blocks are created using a certain template, i.e. they are defined as objects in the true object-oriented sense (see below). Within MEDIA, a number of basic early diagenetic building block types, such as elements, species, transport processes, reactions and parameters, were made available (Fig. 4). The advantage of generic model construction is that it enables model development at a high level.

[1] MEDIA online manual. http://www.nioo.knaw.nl/homepages/meysman/media.htm

Complex numerical models can now be stated and solved without the knowledge of the underlying computer language. Equally, when modifying an existing model, there is no need for accessing or rewriting the program code. This way the time-consuming and strenuous tasks of low-level code writing, debugging and recompilation are avoided.

The second criterion was *extensibility*: the application user should exert control over the model components. Hence, to ensure a large application domain, the application user must have access to a wide number of model buildings blocks. One solution could involve a closed database, where model objects can be selected from a large, fixed set of predefined building blocks. However, to allow real extensibility, an application user must be able to 'bring her/his own building blocks', i.e. it should be possible to extend the modelling environment with new components (e.g. new species and reactions) without the need for rewriting code. The construction of such an *open* database constitutes a challenging software problem, as the number of building blocks to be included is not known a priori. To tackle these issues, an *object-oriented database* was created in MEDIA, which stores all present building blocks and accommodates new user-defined building blocks. Using a template, the model user can add new objects to the database, which are subsequently available for model construction. Fig. 3 compares the modelling procedure of a conventional diagenetic model code and the MEDIA modelling environment.

5. Object-oriented approach

In generic model formulation, a model is regarded as a collection of separate building blocks that can be combined freely. Therefore, an obvious choice is to turn to object-orientation, the paradigm of modern software engineering, which emphasizes the benefits of modular and reusable computer code (Booch, 1994; Meyer, 1997; Page-Jones, 2000). The world of scientific computing has been slow to adopt object-oriented techniques, and is still very much inclined towards the procedure-oriented paradigm. Nonetheless, interest in object-oriented numerics is increasing rapidly (e.g. Arge et al., 1997; Langtangen, 1999). Besides modularity, object-orientation stimulates additional software quality factors, such as reliability, robustness, extensibility and reusability, which are required by modern standards of software quality assurance (Meyer, 1997). Especially for large applications, object-orientation proves beneficial to master the complexity of the developed software

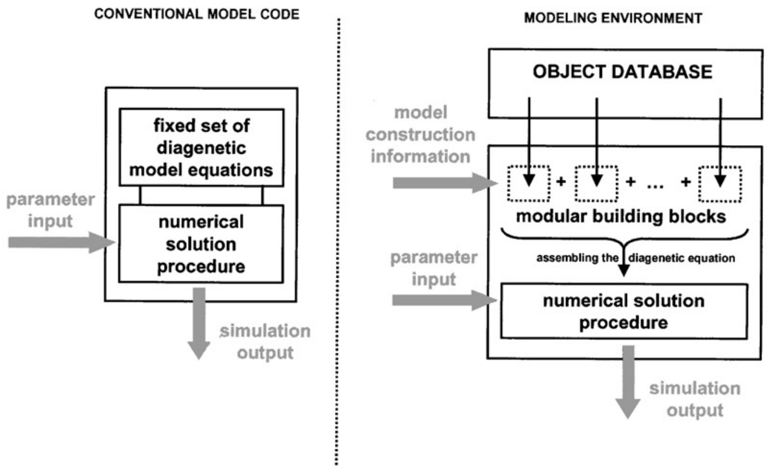
CONVENTIONAL MODEL CODE
fixed set of diagenetic model equations
numerical solution procedure
parameter input
simulation output
MODELING ENVIRONMENT
OBJECT DATABASE
model construction information
+ ... +
modular building blocks
assembling the diagenetic equation
parameter input
numerical solution procedure
simulation output

Fig. 3. Comparison of conventional diagenetic model code and MEDIA modelling environment. Conventional model code, constructed according to 'one model-one code' paradigm, possesses a priori fixed model formulation and thus only allows one type of activity: model execution. Hence, model code only needs one specific type of input: parameter values. Conversely, MEDIA modelling environment allows diagenetic models to be flexibly built from a set of basic building blocks, such as elements, species, transport processes, reactions and parameters. User must provide two types of input information: (1) model construction information which allows MEDIA to assemble diagenetic model from available modular building blocks in object database (2) parameter information which is needed to perform specific simulation. Model construction information itself consists of wish list of diagenetic building blocks (species, reactions, etc.) which the user wants to include into new model. Object database stores all building blocks currently available and can accommodate new user-defined building blocks, which are subsequently available for model construction. Upon input, model construction information is processed by model construction procedure, which is at heart of PSE. Latter procedure (1) generates an appropriate diagenetic model based on model construction information, and (2) delivers model to appropriate numerical model solution procedure. When actual parameter values for specific simulation are fed to solution procedure, final simulation output is obtained.

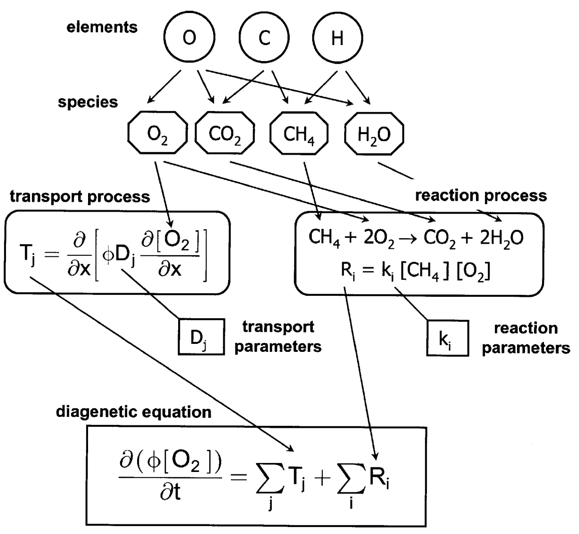
elements
O
C
H
species
O_2
CO_2
CH_4
H_2O
transport process
$T_j = \frac{\partial}{\partial x}\left[\phi D_j \frac{\partial [O_2]}{\partial x}\right]$
reaction process
$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$
$R_i = k_i [CH_4] [O_2]$
D_j
transport parameters
k_i
reaction parameters
diagenetic equation
$\frac{\partial(\phi[O_2])}{\partial t} = \sum_j T_j + \sum_i R_i$

Fig. 4. Overview of different object types / building blocks within MEDIA and connection between several types of objects. A species object is assembled from basic element objects. A transport process is modelled as an operation on a given species object. A reaction object encapsulates multiple species objects associated with that particular chemical reaction. Parameter objects are linked to species objects (e.g. molecular diffusion coefficient), to transport operator (e.g. bioturbation coefficient) and to reaction objects (e.g. a kinetic rate constant). Ultimate object comprises diagenetic "equation" object, which consists of both transport and reaction "process" objects. Reader is referred to Meysman et al. (2002) and MEDIA on-line manual (see footnote 1) for more details on actual implementation of object-oriented principles.

(Booch, 1994). The adaptation of an OOA during the development process can be regarded as one of the major innovative features of the MEDIA modelling environment.

The switch from a procedure-oriented to an OOA requires a major shift in programming practices. Instead of thinking in terms of loops, program steps and procedures, object-orientation allows software to be constructed of objects that have a specified behaviour (Booch, 1994). A computer code is then designed as a collection of autonomous software modules, called objects, and the code developer decides how objects are structured, accessed and modified. Objects of the same type are created using a common blueprint, called a class, which defines both the data structure of an object (the components) as well as the operations possible on these objects (methods). The methods make it possible to access the objects and change the object's properties in the course of program execution. The task of the software developer is to analyse the problem domain, identify the key abstractions as objects and map the events that interact with these objects. The computer program can now be regarded as a "software world", a collection of interacting software objects, which then represents a simplification of the structure and dynamics of the real physical world.

Within MEDIA we have tried to create a "geochemical" software world, consisting of "element", "species", "process", "parameter" and "equation" objects (Fig. 4 explains the hierarchy among these objects). Given its numerical performance, the widespread use in the diagenetic modelling community and the possibility for an object-oriented approach, Fortran90 was preferred as the programming language to develop MEDIA. Following the OOP methodology as outlined in Decyk et al. (1998) and Norton et al. (1998), we used the new Fortran90 capabilities (derived data types, modules, generic interfaces) to emulate typical OOP-concepts as 'class', 'object' and 'method' and to adopt typical OOP-strategies as data encapsulation and inheritance (Carr, 1999). The adoption of OOP required initially a larger investment in code development, but this certainly was already remunerative, and will be so in the longer term. The OO design of MEDIA aided significantly in the management and maintenance of the large code during the development cycle. Secondly, it enabled us to easily rework the MEDIA code during application trials. Finally, the modular object-oriented structure greatly improved the detection of logical modelling errors (i.e. model verification) and enabled intelligent messaging of these errors. Basically, the MEDIA environment checks rigorously whether the model constructed by the MEDIA user is internally consistent. By nature, the object-oriented design invokes a strict bookkeeping of the element objects, and as a consequence, MEDIA performs a rigorous mass balance control and imposes a stringent check on reaction stoichiometry.

6. Conclusion

Although intrinsically powerful applications, it is observed that current diagenetic model codes are far from being used as routine instruments to analyse datasets. Increasing model complexity has effectively reduced the "ease of use" and "applicability" of the software in which reactive transport models are embedded. Codes are rigid due to a procedure-oriented approach and a dense Fortran77 programming style. Concentrating on numerical efficiency, little effort is spent on modern software qualities, such as flexibility, extensibility, robustness, reliability and an intuitive user-interface. As a result, model codes have become increasingly inaccessible to the experimental geochemist, since their operation and adaptation often requires an in-depth understanding of the underlying numerics and programming language. If we are to improve and popularise the use of complex numerical models, we

Table 2
Summary of how concept of problem-solving environment (PSE) and object-oriented technology (OOT) can increase popularity, productivity and possibilities of reactive-transport applications

	PSE	OOT
Popularity	Application users are saved from programming	Other application builders can reuse and extend the modular architecture
Productivity	Model construction without the time-consuming task of hand-coding	Avoids building model codes from ground level
Possibilities	Flexible and extensible model construction	Flexible and extensible adaptation of the model code

clearly need flexible and extensible model codes, which can be easily adjusted to the needs of the model code user. We believe that object-oriented technology and the concept of a problem-solving environment could be valuable tools to achieve this goal in reactive transport model software. Both techniques clearly possess the potential to increase the popularity, the productivity and the possibilities of modelling applications (as summarized in Table 2). They were successfully implemented in the object-oriented problem-solving environment MEDIA (Modelling Early DIAgenesis), which enables the design of costumer-tailored diagenetic models and provides an efficient numerical solution for these models.

Acknowledgements

The ideas presented in this paper have benefited greatly from reviews by D. A. Farrell and an anonymous reviewer, and from fruitful discussions with Bernie Boudreau and Karline Soetaert. This study was financially supported by a grant from the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT) and supported by the European Commission through contract ENV4-CT96-026 (ECO-FLAT). This is publication 2985 of the NIOO-KNAW, Centre for Estuarine and Marine Ecology (CEME), Yerseke, The Netherlands.

References

Ahlander, K., 1999. An object-oriented framework for PDE solvers. Ph.D. Dissertation, Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 423, Uppsala, Sweden, 29 pp.

Akers, R.L., Kant, E., Randall, C.J., Steinberg, S., Young, R.L., 1997. Scinapse: a problem-solving environment for partial differential equations. IEEE Computational Science & Engineering 4 (3), 32–42.

Aller, R.C., 1990. Bioturbation and manganese cycling in hemipelagic sediments. Philosophical Transactions of the Royal Society of London (Series A-Mathematical Physical and Engineering Sciences) 331 (1616), 51–68.

Appelbe, B., Bergmark, D., 1996. Sofware tools for high-performance computing: survey and recommendations. Scientific Programming 5, 239–249.

Arge, E., Bruaset, A.M., Lantangen, H.P., 1997. Object-oriented numerics. In: Daelhen, M., Tveito, A. (Eds.), Numerical Methods and Software Tools in Industrial Mathematics. Birkhauser Press, Boston, MA, pp. 7–26.

Berner, R.A., 1964. An idealized model of dissolved sulfate concentration in recent sediments. Geochimica et Cosmochimica Acta 28, 1497–1503.

Berner, R.A., 1980. Early Diagenesis: A Theoretical Approach. Princeton University Press, Princeton NJ, 241pp.

Blackburn, T.H., Blackburn, N.D., Jensen, K., Risgaardpetersen, N., 1994. Simulation-model of the coupling between nitrification and denitrification in a fresh-water sediment. Applied and Environmental Microbiology 60 (9), 3089–3095.

Booch, G., 1994. Object-oriented Analysis and Design with Applications. The Benjamin Cummings Publishing Company, Inc., San Francisco, CA, 589pp.

Boudreau, B.P., 1991. Modeling the sulfide-oxygen reaction and associated pH gradients in porewaters. Geochimica et Cosmochimica Acta 55 (1), 145–159.

Boudreau, B.P., 1996. A method-of-lines code for carbon and nutrient diagenesis in aquatic sediments. Computers & Geosciences 22 (5), 479–496.

Boudreau, B.P., 1997. Diagenetic Models and Their Implementation. Springer, Berlin, 414pp.

Boudreau, B.P., Canfield, D.E., 1988. A provisional diagenetic model for pH in anoxic porewaters-application to the foam site. Journal of Marine Research 46 (2), 429–455.

Boudreau, B.P., Canfield, D.E., 1993. A comparison of closed-system and open-system models for porewater pH and calcite-saturation state. Geochimica et Cosmochimica Acta 57 (2), 317–334.

Boudreau, B.P., Mucci, A., Sundby, B., Luther, G.W., Silverberg, N., 1998. Comparative diagenesis at three sites on the Canadian continental margin. Journal of Marine Research 56 (6), 1259–1284.

Carr, M., 1999. Using Fortran 90 and object-oriented programming to accelerate code development. IEEE Antennas and Propagation Magazine 41 (6), 85–90.

Coad, P., Yourdon, E., 1991. Object-Oriented Analysis. Yourdon Press, Englewood Cliffs, NJ, 233pp.

Cox, B.J., 1987. Object-Oriented Programming: An Evolutionary Approach. Addison-Wesley, Reading, MA, 274pp.

Dahl, O., Dijkstra, E.W., Hoare, C.A.R., 1972. Structured Programming. Academic Press, London, England, 220pp.

Decyk, V.K., Norton, C.D., Szymanski, B.K., 1998. How to support inheritance and run-time polymorphism in Fortran 90. Computer Physics Communications 115 (1), 9–17.

Dhakar, S.P., Burdige, D.J., 1996. Coupled, non-linear, steady state model for early diagenetic processes in pelagic sediments. American Journal of Science 296 (3), 296–330.

Drobner, E., Huber, H., Wachtershauser, G., Rose, D., Stetter, K.O., 1990. Pyrite formation linked with hydrogen evolution under anaerobic conditions. Nature 346 (6286), 742–744.

Ford, B., Chatelin, F., 1987. Problem Solving Environments for Scientific Computing. North Holland, Amsterdam, The Netherlands, 416pp.

Fujio, H., Doi, S., 1998. Feel: a problem-solving environment for finite element analysis. NEC Research & Development 39 (4), 491–496.

Gallopoulos, E., Houstis, E.N., Rice, J.R., 1994. Computer as thinker/doer: problem solving environments for computational science. IEEE Computational Science & Engineering 1, 11–21.

Gardner, L.R., Lerche, I., 1987. Simulation of sulfate-dependent sulfate reduction using monod kinetics. Mathematical Geology 19 (3), 219–239.

Gardner, L.R., Lerche, I., 1990. Simulation of sulfur diagenesis in anoxic marine-sediments using rickard kinetics for FeS and FeS_2 formation. Computers & Geosciences 16 (4), 441–460.

Goldberg, E.D., Koide, M., 1962. Geochronological studies of deep-sea sediments by the Io/Th method. Geochimica et Cosmochimica Acta 26, 417–450.

Guinasso, N.L., Schink, D.R., 1975. Quantitative estimates of biological mixing rates in abyssal sediments. Journal of Geophysical Research-Oceans and Atmospheres 80 (21), 3032–3043.

Herman, P.M.J., Middelburg, J.J., Heip, C.H.R., 2001. Benthic community structure and sediment processes on an intertidal flat: results from the ecoflat project. Continental Shelf Research 21 (18–19), 2055–2071.

Houstis, E.N., Rice, J.R., Weerawarana, S., Catlin, A.C., Papachiou, P., Wang, K.Y., Gaitatzes, M., 1998. Pellpack: a problem-solving environment for PDE-based applications on multicomputer platforms. ACM Transactions on Mathematical Software 24 (1), 30–73.

Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Wokingham, England, 524pp.

Jorgensen, B.B., 1990. A thiosulfate shunt in the sulfur cycle of marine-sediments. Science 249 (4965), 152–154.

Knox, R.G., Kalb, V.L., Levine, E.R., Kendig, D.J., 1997. A problem-solving workbench for interactive simulation of ecosystems. IEEE Computational Science & Engineering 4 (3), 52–60.

Langtangen, H.P., 1999. Computational Partial Differential Equations, Numerical Methods and Diffpack programming. Lecture Notes in Computational Science and Engineering, Springer, Berlin, 662pp.

Lientz, B.P., Swanson, E.B., 1981. Problems in application software maintenance. ACM 24 (11), 763–769.

Martin, J., Odell, J.J., 1992. Object-Oriented Analysis and Design. Prentice-Hall, Englewood Cliffs, NJ, 412pp.

Meyer, B., 1997. Object-oriented Software Construction. Prentice-Hall, Englewood Cliffs, NJ, 1296pp.

Meysman, F.J.R., 2001. Modelling the influence of ecological interactions on reactive transport processes in sediments. Ph.D. Dissertation, Ghent University, Gent, Belgium, 213pp.

Meysman, F.J.R., Middelburg, J.J., Herman, P.M.J., Heip, C.H.R. 2002. Reactive transport in surface Sediments II. MEDIA: an object-oriented problem-solving environment for early diagenesis. Computers & Geosciences, this issue.

Norton, C.D., Decyk, V., Slottow, J., 1998. Applying Fortran 90 and object-oriented techniques to scientific applications. Lecture notes in computer science. Object-Oriented Technology. 1543, 462–463.

Page-Jones, M., 2000. Fundamentals of Object-Oriented Design in UML. Dorset House Publishing, New York, NY, 458pp.

Parker S, .G., Weinstein, D.W., Johnson, C.R., 1997. The SCIRun computational software steering system. In: Arge, E., Bruaset, A.M., Lantangen, H.P. (Eds.), Modern Software Tools in Scientific Computing. Birkhauser Press, Boston, MA, pp. 203–226.

Rabouille, C., Gaillard, J.F., 1991. Towards the edge-early diagenetic global explanation-a model depicting the early diagenesis of organic-matter, O_2, NO_3, Mn, and PO_4. Geochimica et Cosmochimica Acta 55 (9), 2511–2525.

Rice, J.R., Boisvert, R.F., 1996. From scientific software libraries to problem solving environments. IEEE Computational Science & Engineering 3 (3), 44–53.

Rickard, D., Luther, G.W., 1997. Kinetics of pyrite formation by the H_2S oxidation of iron(II) monosulfide in aqueous solutions between 25 and 125 degrees C: the mechanism. Geochimica et Cosmochimica Acta 61 (1), 135–147.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., 1991. Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, NJ, 500pp.

Soetaert, K., Herman, P.M.J., Middelburg, J.J., 1996a. A model of early diagenetic processes from the shelf to abyssal depths. Geochimica et Cosmochimica Acta 60 (6), 1019–1040.

Soetaert, K., Herman, P.M.J., Middelburg, J.J., 1996b. Dynamic response of deep-sea sediments to seasonal variations: a model. Limnology and Oceanography 41 (8), 1651–1668.

Stroustrup, B., 1988. What is object-oriented programming? IEEE Software 5 (3), 10.

Van Cappellen, P., Wang, Y.F., 1996. Cycling of iron and manganese in surface sediments: a general theory for the coupled transport and reaction of carbon, oxygen, nitrogen, sulfur, iron, and manganese. American Journal of Science 296 (3), 197–243.

Wang, Y.F., Van Cappellen, P., 1996. A multicomponent reactive transport model of early diagenesis: application to redox cycling in coastal marine sediments. Geochimica et Cosmochimica Acta 60 (16), 2993–3014.

Yourdon, E., Constantine, L.L., 1979. Structured Design. Prentice-Hall, Englewood Cliffs, NJ, 473pp.