http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (6): 868-875, 2008
ISSN 1812-5638

Rigorous Development of Fault-Tolerant Transactions for Information Retrieval Systems Using Event-B

[1,2]Hong-Jiang Gao, [4]Hong Sun, [2]Bang-Hai Xu, [5]Ying He and [1,3]Zheng Qin
[1]Department of Computer Science and Technology, Xi'an Jiaotong University,
Xi'an, Shaanxi, People's Republic of China
[2]School of Computer Science and Technology, Ludong University,
Yantai, Shandong, People's Republic of China
[3]School of Information Science and Technology, Tsinghua University, Beijing, People's Republic of China
[4]Department of Computer Engineering, Xinjiang Polytechnical College,
Urumqi, Xinjiang, People's Republic of China
[5]Department of Electrical Engineering, Xinjiang Polytechnical College,
Urumqi, Xinjiang, People's Republic of China

Abstract: The aim of this study is to demonstrate the approach of stepwise development of a distributed transaction mechanism for information retrieval systems. In this study, we formally develop an abstract model of transactions in Event-B for an IR system, in which fault tolerance is provided in the distributed transaction execution. We starts from an abstract system specification and gradually introduce implementation details in a series of correctness-preserving transformations, where complex system properties (such as fault tolerant) could be specified in a structured and rigorous way. During each transformation, the refinement between the abstract specification of the system and its detailed design is verified. Using Event-B, we achieve a high degree of automatic proof via this incremental approach.

Key words: Event-B, formal method, Information Retrieval (IR), refinement, mechanical proof

INTRODUCTION

Over the past few years, Information Retrieval (IR) systems, especially distributed IR systems, have emerged as a new research focus. Due to the dramatically increasing amount of available data, effective and scalable solutions for data organization and search are essential. Distributed solutions naturally provide promising alternatives to standard centralized approaches.

We undertook an Event-B (Abrial and Hallerstede, 2007) development of the intellectualized design system of shape and style in automobile (IDSSSA) (Gao *et al*., 2007) using the Atelier B (http://www.atelierb.eu/ressources/manrefb1.8.6.uk.pdf) and evt2b (http://www.atelierb.eu/resources/evt2b/evt2b_user_manual.pdf) tools for support. Atelier B is an industrial tool of proof obligation generator and proof tool for B. The evt2b tool is intended to translate Event-B models into "classical" B models. The original B method, developed by Abrial (Abrial, 1996; Cansell and Mery, 2003; Cansell, 2006), is a well known approach to the formal specification and development of sequential computer programs. Event-B, inspired by the action system approach of Back (Back, 1990) and other research in the area of distributed systems modeling (Walden and Sere, 1998), is an evolution of B towards understanding and reasoning about systems including reactive, distributed and concurrent systems. As a formalism and method for discrete system modeling, Event-B provides a framework for developing mathematical models of distributed systems by rigorous description of the problem, gradually introducing solutions in refinement steps and verification of solutions by discharge of proof obligations. Recently, tool support has been provided for Event-B specification and proof in the Rodin platform (http://sourceforge.net/projects/rodin-b-sharp/).

As a distributed system, IDSSSA may be involved with several sites in a distributed transaction. A typical distributed transaction (Gray and Reuter, 1993) contains a sequence of data operations which must be processed at all of the participating sites or none of the sites to maintain the integrity of the data. For the sake of avoiding site failures, blocking and even compensation, it needs to ensure consistency and provide fault-tolerance for the distributed transaction execution within IDSSSA.

Corresponding Author: Hong-Jiang Gao, Room 708, West Building 1, No. 28 West Xianning Road, Xi'an, Shaanxi, People's Republic of China Tel: +86-29-82663979

An approach about rigorous design of fault-tolerant transactions in B for a one copy database was proposed by Yadav and Butler (2006). The incremental development of dependable systems was carried out on the B4free and Click'n'Prove tool to generate the proof obligations for refinement and consistency checking. Later, they also developed the Mondex system using Event-B and its associated proof tools with the aim of achieving a high degree of automatic proof (Butler and Yadav, 2008). In this study, inspired by their researches, we formally develop an abstract model of transactions for information retrieval system of IDSSSA in Event-B using Rodin platform. Study refinements break the atomicity of the transaction into several steps following the definition of transaction protocol. Fault tolerance is gradually introduced in transaction models along with stepwise refinements (Rouzaud, 1999; Potet and Rouzaud, 1998; Abrial *et al*., 2005). At any stage a participant can abort a transaction and the retrieved information can be kept in a consistence state after the reconcilement between both participants. The whole development process is also with a high degree of automatic proof in the end.

EVENT-B

Formal methods such as Event-B are based on mathematics and logics and therefore, provide rigor and precision that cannot be achieved by using only natural language. The semantics of an Event-B model is characterized by proof obligations so as to show that a model is sound with respect to some behavioral semantics and to serve to verify properties of the model. Thus, the same proof obligations can be used for very different modeling domains without being constrained to semantics tailored to a particular domain. Event-B is a calculus for modeling that is independent of the various models of computation. Generally, this formal technique consists of the following steps (Rezazadeh and Butler, 2005):

- Rigorous description of abstract problem
- Introduce solutions or details in refinement steps to obtain more concrete specifications
- Verifying that proposed refinements are valid

An abstract specification for a generic system includes two parts. The static part, called the context, defines the reference sets and constants and other related attributes. The dynamic part, called the machine, is defining the variables, their related invariants and events. The machine has access to the context via a SEES relationship. All sets, constants and their properties are defined in the context. The machine contains all of the state variables. The values of the variables are set up using the INITIALISATION clause and values can be changed via the execution of events. Ultimately, we aim to prove properties of the specification and these properties are made explicit using the INVARIANT clause. Event-B is provided with tool support currently in the form of a platform for specification and proof called Rodin. The tool support generates proof obligations which must be discharged to verify that the invariant is maintained.

In order to refine an abstract Event-B specification, it is possible to refine the model and context separately. Refinement of a context consists of adding additional sets, constants or properties.

Events are specialized forms of classical B operations. Refinement of existing events in a model is similar to refinement in the B method: a gluing invariant in the refined model relates its variables to those of the abstract model. Proof obligations are generated to ensure that this invariant is maintained. In Event-B, abstract events can be refined by more than one event. In addition, Event-B allows refinement of a model by adding new events on the proviso that they cannot diverge (i.e., execute forever). This condition ensures that the abstract events can still occur. Since the new events operate on the state variables of the refined model, they must implicitly refine the abstract event skip. Due to the guard strengthening through refinement process, we have to prove that there is always at least one enabled event (no deadlock) and that new events do not introduce livelocks.

ABSTRACT TRANSACTION MODEL

The IDSSSA is very complex as a whole, so we have carefully chosen a subset of the original IDSSSA, called core specification for redevelopment in the context of the Rodin platform. The core specification, giving an idealized view of information retrieval behaviour of the system, is involved in two types of transactions, i.e., a read-only transaction and an atomic update transaction. Through careful use of small refinement steps and appropriate intermediate abstractions, we were able to achieve an impressive degree of automatic proof using Rodin.

System model consists of sets of sites and data objects. The distributed database consists of sets of objects stored at different sites. The data objects are assumed to be stored at a central server site and replicated across all sites. The specification consists of just three state variables as follows:

$$trans \in \mathbb{P}(TRANSACTION)$$
$$transobject \in trans \rightarrow \mathbb{P}1(OBJECT)$$
$$transstatus \in trans \rightarrow TRANSSTATUS$$

The variable trans represents the set of transactions that currently exist in the system. The variable transobject is a total function which maps a transaction to a set of objects. The set transobject(t) represents the set of data objects accessed by a transaction t. The variable transstatus is a total function which maps a transaction to a set of states of transactions. The TRANSSTATUS is an enumerated set that contains values ABORT and ENDED. A transaction is ENDED after its successful completion or ABORT because of failing to finish a transaction. Note that these states, introduced as part of our intermediate abstraction, are not explicit states in the IDSSSA transaction protocol.

In the model, two deferred sets are defined, i.e., TRANSACTION and OBJECT (Fig. 1).

Three significant events of the abstract model are given. The event TransOK represents the successful transaction of information and an update to the related database. The guards given in the WHERE statement ensure that tt is a fresh transaction. The ANY statement set the variable transobjects(tt) to be a non empty set of objects. Transaction failure is represented by sites aborting transactions. The TransFail event models a result

CONTEXT
Retrieval_c0
SETS
TRANSACTION
OBJECT
TRANSSTATUS
CONSTANTS
ABORT
ENDED
AXIOMS
axm1 : TRANSSTATUS = {ABORT, ENDED}
axm2 : ABORT ≠ ENDED
END
MACHINE
Retrieval_m0
SEES
Retrieval_c0
VARIABLES
trans
transobject
transstatus
INVARIANTS
inv1 : trans ∈ ℙ (TRANSACTION)
inv2 : transobject ∈ trans → ℙ1 (OBJECT)
inv3 : transstatus ∈ trans → TRANSSTATUS

Fig. 1: Continued

EVENTS
INITIALISATION ≙
WHICH IS
ordinary
BEGIN
act1 : trans := Ø
act2 : transobject := Ø
act3 : transstatus := Ø
END
TransOK ≙
WHICH IS
ordinary
ANY
tt
objects
WHERE
grd1 : tt ∈ TRANSACTION\trans
grd2 : objects ∈ ℙ1(OBJECT)
THEN
act1 : trans:= trans ∪ {tt}
act2 : transobject(tt) := objects
act3 : transstatus(tt) := ENDED
END
TransFail ≙
WHICH IS
ordinary
ANY
tt
objects
WHERE
grd1 : tt ∈ trans
grd2 : objects ∈ ℙ1 (OBJECT)
THEN
act1 : transstatus(tt) := ABORT
END
Recover ≙
WHICH IS
ordinary
ANY
tt
objects
WHERE
grd1 : tt ∈ trans
grd2 : objects ∈ ℙ1 (OBJECT)
grd3 : transstatus(tt) = ABORT
THEN
act1 : transobject(tt) := objects
act2 : transstatus(tt) := ENDED
END
END

Fig. 1: The initial abstract transaction model

of transaction failure. A site may decide to abort an update transaction due to timeouts or other unpredictable incidents, for example, bad data input or communication link failures. The cause of transaction failure is made more specific in the refined layers. The TransOK and TransFail events should be viewed together are modeling the possible outcome of a transaction which either succeeds or fails. The third event, Recover, is an abstract model of the mechanism whereby the unsuccessful transaction is recovered to the previous states.

REFINEMENT OF THE TRANSACTION MODEL

In the abstract model, we saw that the information object is transferred in a single atomic step, which provides for a clear and immoderately simplified specification of the essence of the protocol. However, real system consists of many transfers engaged in information exchange so multiple parallel protocol runs will be interleaved and many events may occur in the retrieved information.

Next, incrementally add more features to the models through successive refinements and we incrementally achieve a proof that our abstract model is refined by the final detailed system model.

The first refinement: breaking the atomicity of transaction: The main phases of the IDSSSA transaction protocol are illustrated in Fig. 2. The protocol run starts while two participants login the transaction system. After the console terminal emits a startup message to both participants, the subscriber sends a request message StartTrans to the bidder account. On receipt of the StartTrans, the bidder begins to retrieve the required object information from its central database and then answers the starter account with a message CommitTrans. The retrieved information is now in transit. At some stage later, the CommitTrans may be received by the subscriber which proceeds to update its local database and sends an acknowledgment AckTrans to the bidder. Receipt of a CommitTrans by the subscriber denotes successful completion of an object information transfer. For subscriber the protocol run ends after it sends the AckTrans while for bidder the protocol run ends after it receives the AckTrans. Note that the protocol run may fail when it is aborted by either side, which might be caused by a timeout or by a participant quitting their transaction unpredictably from the login terminal.

Owing to the huge gap between previous abstract specification and concrete model involving all the details of the protocol events and states is too large, we introduced an intermediate transaction abstraction first of all to provide sufficient structure to break the atomicity of the retrieved information transfer, which reduced the complexity of reasoning with too much detail.

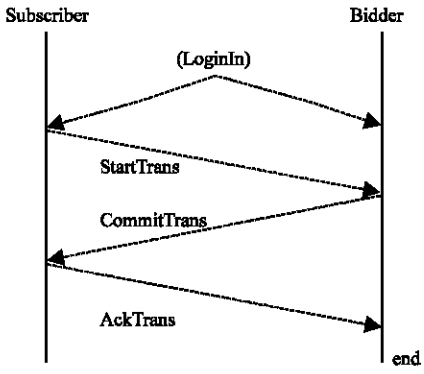

Fig. 2: Overview of IDSSSA transaction protocol

A transaction consists of a source participant and a target participant as well as a data object, so we introduce a record type TRANS modeling transactions according to Evans and Butler (2006), typed as follows:

$$\begin{aligned} \text{TRANS} :: \text{from} &\in \text{TRANSACTION}; \\ \text{to} &\in \text{TRANSACTION}; \\ \text{object} &\in \mathbb{P}1\ (\text{OBJECT}) \end{aligned}$$

According to the transaction protocol, the task of our information retrieval is to retrieve the required information from the central database of the bidder and then write it to the local database of the subscriber. The database is represented by a variable database as a total function from objects to values, i.e., database $\in$ OBJECT $\rightarrow$ VALUE.

For simplicity, we focus on errors caused by sites failing to commit a transaction when one of participants aborts the transaction. Suppose that there only exist two types of transactions, i.e., read-only transactions and update transactions in our transaction model. An individual transaction will involve a set of source objects readset $\subseteq$ OBJECT and a set of target objects writeset $\subseteq$ OBJECT. It will firstly read from a partial projection of the database (pdb) on to readset, denoted by pdb = readset $\lhd$ database where the notation $\lhd$ means domain restriction operation. Afterwards, a subset of readset is picked and the values of the selected objects should be written to the set of objects writeset where writeset $\subseteq$ readset. Thus, a transaction T_i is defined as a read-only transaction if writeset $(T_i) = \emptyset$ whereas a transaction T_i is an update transaction if writeset$(T_i) \neq \emptyset$.

Therefore, here could model an update to a database as function which takes a partial database pdb1

(representing the current values of the objects in readset from the bidder side) and yields a partial database pdb2 (representing the new values of the objects in writeset from the subscriber side) where dom(pdb2) = writeset. The update function update ∈ UPDATE updates the database as follows:

$$database := \leftarrow database\ \ update\ (pdb1)$$

where, the notation ← represents relational override operation.

The variable transeffect is also a total function which maps each transaction to an object update function UPDATE. The invariant t ∈ trans ⇒ ValidUpdate (transeffect(t), transobject(t)) indicate that all updates must be valid.

Based on the several interactive stages in Fig. 2, here introduced 6 events as the refinement of the original abstract transaction events model, depicted in Fig. 3. The event ValidUpdate is used to evaluate whether all updates are valid. The event StartTrans models the creation of a new transaction tt, which is considered as read-only if the ran(transeffect(tt)) is set to an empty set, or it is considered an update transaction if ran(transeffect(tt)) contains at least one UPDATE mapping. The event ReadTrans models commitment of a read-only transaction tt. After reading the objects from the database defined by transeffect(tt), the transaction status is set to PENDING from IDLE. The event CommitWriteTrans models commitment of an update transaction when the abstract database is updated successfully with the effects of the transaction and its status is set to ENDING. The event AbortWriteTran models abort of an update transaction with its status set to ABORTING from PENDING. As for the event RecoverTrans, it is obvious that only the abortion of transaction tt occurring during the event CommitWriteTrans needs to be recovered. After the previous value is re-written successfully to its database object, the transaction status is also set to ENDING.

The whole refinement process is omitted for space reasons, which will be discussed in another paper.

Overview of the refinement chain: Up to now, we have outlined our abstract model of the transaction for IDSSSA and the first refinement. Next, we will present an overview of the remaining refinement chain.

Level 1: This level consists of the abstract model of transactions, which captures the transfer of retrieved information, transaction failure and recovery of the unfinished update transaction in a single atomic event.

ValidUpdate ≙
  WHICH IS
    ordinary
  ANY
    updates
    objects
  WHERE
    grd1 : updates ∈ UPDATE
    grd2 : objects ∈ ℙ1(OBJECT)
    grd3 : dom(updates) = readset ⇸ VALUE
    grd4 : ran(updates) ⊆readset ⇸ VALUE
  THEN
    act1 : valid := TRUE
  END
StartTrans ≙
  WHICH IS
    ordinary
  REFINES
    TransOK
  ANY
    tt
    updates
    objects
    subscriber
    bidder
  WHERE
    grd1 : tt ∈ TRANSACTION \ trans
    grd2 : updates ∈ UPDATE
    grd3 : objects ∈ ℙ1(OBJECT)
    grd4 : valid = TRUE
    grd5 : from(tt) = subscriber
    grd6 : to(tt) = bidder
    grd7 : object(tt) = objects
  THEN
    act1 : trans := trans ∪ {tt}
    act2 : status(tt) := IDLE
    act3 : transobject(tt) := objects
    act4 : transeffect(tt) := updates
  END
ReadTrans ≙
  WHICH IS
    ordinary
  REFINES
    TransOK
  ANY
    tt
    objects
    bidder
  WHERE
    grd1 : tt ∈ trans
    grd2 : status(tt) = IDLE

Fig. 3: Continued

grd3 : ran(transeffect(tt)) = {∅}
grd4 : from(tt) = bidder
grd5 : object(tt) = objects
THEN
act1 : status(tt) := PENDING
act2 : val := transobject(tt) ◁ database
END
CommitWriteTrans ≙
WHICH IS
ordinary
REFINES
TransOK
ANY
tt
pdb
reloaddb
objects
subscriber
WHERE
grd1 : tt ∈ trans
grd2 : pdb = transobject(tt) ◁ database
grd3 : status(tt) = PENDING
grd4 : reloaddb ⊆ OBJECT → VALUE
grd5 : ran(transeffect(tt)) ≠ {∅}
grd6 : transobject(tt) = objects
grd7 : to(tt) = subscriber
THEN
act1 : status(tt) := ENDING
act2 : database := database ⩤ (transeffect(tt))(pdb)
END
AbortWriteTrans ≙
WHICH IS
ordinary
REFINES
TransFail
ANY
tt
objects
subscriber
bidder
WHERE
grd1 : tt ∈ trans
grd2 : status(tt) = PENDING
grd3 : ran(transeffect(tt)) ≠ {∅}
grd4 : transobject(tt) = objects
grd5 : from(tt) = bidder
grd6 : to(tt) = subscriber
THEN
act1 : status(tt) := ABORTING
END

Fig. 3: Continued

RecoverTrans ≙
WHICH IS
ordinary
REFINES
Recover
ANY
tt
objects
bidder
WHERE
grd1 : tt ∈ trans
grd2 : status(tt) = ABORTING
grd3 : ran(transeffect(tt)) ≠ {∅}
grd4 : transobject (tt) = objects
grd5 : from(tt) = bidder
THEN
act1 : status(tt) := ENDING
END

Fig. 3: Operations of the first refinement

Level 2: As outlined in the previous subsection, an elaborate notation of transaction is introduced with end-to-end state (IDLE, PENDING, ABORTING or ENDING). The information transfer process is split into three consecutive events. A transaction t is a read-only transaction if ran(transeffect(t))={∅}. Otherwise it is an update transaction.

Level 3: In this refinement the end-to-end state of transaction, which is redundant notation, is replaced by local protocol states. For example, when the bidder is in the expr (expecting request) state in between receiving a LoginIn message and receiving a StartTrans message, the transaction t is modeled by the condition t∈expr where from(t) = bidder.

Level 4: In order to avoid the access of the state in the other side directly, here introduces explicit messages between the two participants. One participant gains information about the other side only when it receives a message.

Level 5: In the refinement we introduce the notation of participating site failures to retain the updated database remaining in a consistent state in case of presence of site failures.

The system development approach considered is based on Event-B. We used the Rodin tool to generate the proof obligations for refinement and consistency checking. About 62% of proofs were discharged using the automatic prover of the tool and the rest required use of the interactive prover (Table 1).

Table 1: Proof statistics of IDSSSA transaction

Machine	Total POs	Automatic	Manual	Reviewed	Unproved
Abstract model	9	9	0	0	0
1st refinement	26	24	2	0	0
2nd refinement	215	112	72	31	0
3rd refinement	53	39	7	6	1
4th refinement	29	20	4	4	1
Total	332	206	83	41	2

CONCLUSIONS

In this study, here is done two major technical contributions. First, it is defined the abstractions required to model behavior of information retrieval systems. Second, based on these abstractions formally developed a distributed transaction mechanism for fault-tolerant IR systems using Event-B. In the abstract model of transactions, an update transaction is granted to commit atomically updating its local information at commit or none when it aborts. At any stage a participant can abort a transaction and the retrieved information can be kept in a consistence state after the reconcilement between both participants.

In study development of the IDSSSA distributed transactions for information retrieval systems, the Rodin tool helps us discover appropriate gluing invariants to prove the refinements as well as to provide a clear insight to the system. The incremental approach with small refinement step affords convenient facilities for our automatic proof of a refinement step and appropriate abstraction and refinement can also help us to understand the complexity of problem and the correctness of the solutions.

Future work includes the study of the same transaction problem within a role-based open multi-agent system. It also includes the specification and verification of other properties, such as deadlock-free or livelock-free. We also plan to make the model evolve in order to work in a context-aware environment.

ACKNOWLEDGMENTS

The research reported in this study is funded by the Major State Basic Research Development Program of China (973 Program) under Grant No. 2004CB719401, the National Natural Science Foundation of China under Grant No. 60673024 and also supported by Ludong University midlife and youth Natural Science Research project under Grant No. LY20064102. The authors would like to acknowledge gratefully the valuable comments which will help us improve this study.

REFERENCES

Abrial, J.R., 1996. The B-Book: Assigning Programs to Meanings. 1st Edn. Cambridge University Press, New York, USA., ISBN: 0-521-49619-5.

Abrial, J.R., D. Cansell and D. Méry, 2005. Refinement and Reachability in Event-B. In: ZB 2005: Formal Specification and Development in Z and B. Guilford, U.K., H. Treharne, S. King and M. Henson (Eds.). Springer-Verlag, Berlin, pp: 222-241.

Abrial, J.R. and S. Hallerstede, 2007. Refinement, decomposition and instantiation of discrete models: Application to Event-B. Fundam. Inform., 77: 1-28.

Back, R.J., 1990. Refinement calculus, part II: Parallel and reactive programs. Proceedings on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, May 29-June 2, 1989. REX Workshop, Mook, The Netherlands, Springer-Verlag, Berlin, pp: 67-93.

Butler, M. and D. Yadav, 2008. An incremental development of the mondex system in Event-B. Form. Asp. Comput., 20: 61-77.

Cansell, D. and D. Mery, 2003. Foundations of the B-method. Comput. Informatics, 22: 221-256.

Cansell, D., 2006. B-method: The seventeen provers of the world. Lecture Notes Comput. Sci. LNAI, 3600: 142-150.

Evans, N. and M. Butler, 2006. A proposal for records in Event-B. In: FM 2006: Formal Methods, 14th International Symposium on Formal Methods. Hamilton, Canada August 21-27, Springer-Verlag, Berlin, pp: 221-235.

Gao, H.J., Z. Qin, L. Lu, L.P. Shao and X.C. Heng, 2007. Formal specification and proof of Multi-Agent applications using event B. Inform. Technol. J. Pak., 6: 1181-1189.

Gray, J. and A. Reuter, 1993. Transaction Processing: Concepts and Techniques. 1st Edn., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA., ISBN: 1558601902 .

Potet, M.L. and Y. Rouzaud, 1998. Composition and refinement in the B-method. In: B'98: Recent advances in the development and use of the B-method. LNCS, 1393: 46-65.

Rezazadeh, A. and M. Butler, 2005. Some guidelines for formal development of web-based applications in B-Method. In: ZB 2005: Formal Specification and Development in Z and B, Proceedings. 2005 Springer-Verlag Berlin, pp: 472-491.

Rouzaud, Y., 1999. Interpreting the B-Method in the refinement calculus. LNCS, 1708: 411-430.

Walden, R. and K. Sere, 1998. Reasoning about action systems using the B-method. Form. Methods Syst. Des., 13: 5-35.

Yadav, D. and M. Butler, 2006. Rigorous Design of Fault-Tolerant Transactions for Replicated Database Systems Using Event B. In: Rigorous Development of Complex Fault-Tolerant Systems. Butler, M., C. Jones, A. Romanovsky and E. Troubitsyna (Eds.). Springer-Verlag, Berlin, Heidelberg, New York, ISBN-10: 3-540-48265-2, pp: 343-363.