IADIS International Journal on Computer Science and Information Systems
Vol. 6, No.3, pp. 38-52
ISSN: 1646-3692

COINS: A MODELLING APPROACH FOR RECORDING & REPLAYING COMPLEX INFORMATION

Erqiang Zhou. *School of Computing, Dublin Institute of Technology, Kevin St. Dublin 8, Ireland.*

Bing Wu. *Head of Asian Partnerships, Dublin Institute of Technology, 143-149, Rathmines Rd., Dublin 6. Ireland.*

Yi Ding. *School of Computing, Dublin Institute of Technology, Kevin St. Dublin 8, Ireland.*

Jianfeng Wu. *School of Computing, Dublin Institute of Technology, Kevin St. Dublin 8, Ireland.*

ABSTRACT

A modelling approach *COmplex INformation Specification (COINS)* equipped with a recording and replying mechanism is proposed to facilitate the process of providing a live interactive record of information that can then be queried and replayed. The COINS model consists of three components: 1) an Entity Relationship (ER) model; 2) a rule mechanism which deals with logic constraints among attributes, entities and relationships and; 3) a recorder component which provides support for the idea of replaying complex information. The main function of such a recording and replying mechanism is to record a sequence of information scenes so that this sequence, as a history of the evolving information for the recorder, can be queried and replayed. The puzzle game Sudoku is used as an example in this paper for illustrating the COINS modelling approach.

KEYWORDS

Complex Information, Information Scene, Modelling, Recording Mechanism, Replay.

1. INTRODUCTION

Computerising real-world complex information (e.g. best practices, clinical test protocols, etc.) poses at least three major challenges for information management (Wu, et al., 2007): 1) to support the customisation of best practice so as to suit a specific domain user and task; 2) to keep the changing history of certain piece of information units in order to review their whole

evolution processes and; 3) to facilitate the manipulation, query, and review by replay operation as well as sharing and distribution for these information units.

Experience from our earlier work (Wu and Dube, 2001; Dube, 2004; Mansour, 2008, Wu, et al., 2008) suggests that presentation, analysis and review of information can be significantly simplified and enhanced through a new approach which enables information to be managed, played and replayed in a dynamic and interactive manner on the basis of a well-defined model of domain information.

This paper presents a modelling approach for COmplex INformation Specification (COINS). The original aim of this modelling approach is, by incorporating an Entity Relationship (ER) model (Chen, 1976), to integrate COINS with a mechanism of recording and replaying complex information. Three components are designed in COINS: 1) an ER model which acts as its kernel; 2) a rule mechanism which deals with logic constraints among attributes, entities and relationships and; 3) a recorder component which provides support for the idea of replaying complex information.

The rest of this paper is organised as follows: Section 2 outlines related work. Section 3 presents the COINS approach for recording and replaying complex information. Section 4 illustrates COINS by modelling and specifying the semantics of the puzzle game Sudoku. Section 5 concludes this paper.

2. RELATED WORK

The concept of playing back of stored information has been investigated in a number of application domains. Consequently, a number of approaches and methods of supporting information replay has emerged (Manohar & Prakash, 1994; Brown & Patterson, 2002; Greenhalgh et al., 2000). The idea of replaying the past has been extensively explored in the field of debugging or testing programs (Narayanasamy et al., 2005; Guo et al., 2008; Altekar & Stoica, 2009; Chow et al., 2010).

In addition to debugging program, the idea of replaying the past has been applied to diagnosing more complex information system like database (Galanis et al., 2008). Unlike the simple log approach used by OS, Galanis et al. (2008) adopted a more complex mechanism so as to "record all necessary activity on the DBMS that is required to faithfully reproduce the same activity on a test system". The core idea of the adopted mechanism is to capture all service calls of a client so that these calls can be replayed to another server. One interesting point of this approach is that the testing system can reproduce all the captured events that occurred in the production system. However, the separation between capturing and replaying results in more complex configuration. The fact that the test system is nearly identical to the production system means that users would need more hardware and software resources. Users of such a system could only treat the recoded information just as a media player treats video data. There is no query function provided. Therefore, it is difficult for users to get a piece of information that they are particularly interested in. In summary, Galanis et al. (2008) followed log/viewer way for replaying information scene.

While many efforts have been put into general information modelling and management, Wu and Dube (2001) started the work with a structured language named PLAN, aiming to capture not only structures of data but also the semantics of a specific application domain of clinic test protocols. Based upon the PLAN language, Dube (2004) investigated and

implemented the SpEM framework as well as the TOPS prototype. Mansour (2008) further developed this approach, adopting XML for expressing domain semantics so that the specified semantics can be extended. Mansour & Höpfner (2009) presented a replay support method for rule-based information. This paper continues and extends the previous work by providing an approach for recording and replaying complex information.

3. THE MODELLING APPROACH

The context of the COINS modelling approach is to manage complex information within the paradigm of recording live interactive records of information so that these records can be queried and replayed. The COINS approach focuses on the specification of complex information. Other aspects, such as the customisation and realisation of rules, the interfaces for querying and replaying the recorded information, would all be necessary for systems which incorporate the COINS approach.

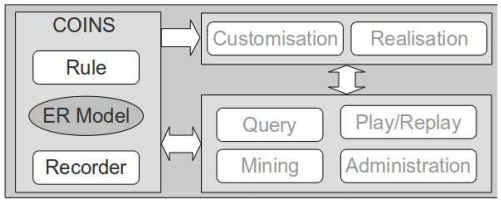


Figure 1. A framework of using COINS

Figure 1 presents a framework for managing complex information with the COINS modelling approach. This paper will only focus on the COINS component and its sub-components. This framework views information systems as systems for recording dynamic and interactive information units. It intends to provide a permanent live interactive record of information which can then be queried, played and replayed. In particular, the replaying mechanism provides abilities to present 1) information in terms of its logic and in a time sequence for a duration; 2) relevant manipulations applied on them, and 3) a whole picture of the evolution of certain piece of information through querying the history of that information.

3.1 The Structure Model

For demonstration purpose, an ER model (Chen, 1976) is employed for the structure modelling part of the COINS. Generally, an ordinary ER model consists of three kinds of basic building blocks: an entity set where each entity represents a real world object or concept; a relationship set where each relationship specifies an association between two entities; an attribute set where each attribute describes a property of an entity or a relationship.

In this paper, the ER model acts as the kernel of COINS, therefore the diagram notations about the ER model in COINS coincide with the well known ones so that the COINS diagram can be easily drawn and understood when users have the knowledge of the ER model. Although the ER model offers a concise way for specifying the structural information, it does not provide ways for describing the logic constraints for attributes or entities, or constraints among attributes and entities. The rule mechanism in COINS is provided for specifying these kinds of constraints.

3.2 Rule Mechanism

Mechanisms for modelling domain rules are often needed in application domains. The COINS approach provides two kinds of rules to tackle this problem. One is the *implicit rule* and the other is the *explicit rule*.

An *implicit rule* is for an attribute and it consists of two parts: *value types* and *value constraints*. A *value type* specifies the type of an attribute (e.g. *integer*, *character* or *string* etc.). A value *constraint* imposes a further restriction upon the specified type. The result of such a constraint is a subset of the value-range of that type. As a modelling approach, COINS does not regulate the format of the constraints.

An *explicit rule* in COINS is identified as a combination of expressions. There are three types of expressions: *logic expressions*, *action expressions* and *scope expressions*. A *logic expression* is represented as a three-tuple (*lexp*, *oper*, *rexp*), where *lexp* represents the left part of expression; *rexp* represents the right part of expression; *oper* represents the operator between *lexp* and *rexp*. The whole logic expression means that the result of "*lexp oper rexp*" should be true. Currently COINS has only one type of *action expression*: the *assignment expression*. The *assignment expression* has a fixed structure (*lexp* ← *rexp*) which means assigning the value of *rexp* to *lexp*. The *scope expression* has two forms. One is (*When*: *logic-expression*) and the other is (∀ element ∈ collection). The first one specifies certain conditions while the second specifies a scope which means for each item (*element*) in the collection (*collection*). The description of the rule mechanism is summarised as follows.

<rules> ::= <implicit rule> | <explicit rule>
| <rules> <implicit rule>
| <rules> <explicit rule>
<implicit rule> ::= {<value type>, <value constraint>}
<explicit rule> ::= <logic exp>
| <logic exp> <explicit rule>
| <scope exp> <action exp>
| <scope exp> <logic exp>
| <scope exp> <explicit rule>
<action exp> ::= <assignment exp>
| <action exp> <assignment exp>
<assignment exp> ::= <lexp>, ←, <rexp>
| (<assignment exp>)
<logic exp> ::= <lexp>, <oper>, <rexp>
| (<logic exp>)
| {<logic exp>}
| not <logic exp>
| <logic exp> and <logic exp>
| <logic exp> or <logic exp>
<scope exp> ::= When: <logic exp>
| ∀ <element> ∈ <collection>
| {<scope exp>}

3.3 Value Recorder and Rule Recorder

The recorder component is used to facilitate the idea of replaying the history of evolving information. This is done by a special mechanism declaring what kind of information needs to be recorded in the COINS model. Targeting at different type of information to be recorded, three kinds of recorders are provided in the COINS model: *value recorders*, *rule recorders*, and *scene recorders* (which focus on a cluster of related entities and relationships and are discussed in Section 3.4).

A value recorder is used to specify an attribute (type) of a certain entity type that users are interested in and therefore want to record its whole changing history. In the diagram notation of COINS, the symbol ® is used to distinguish an attribute attached with a value recorder from normal attributes. Once the symbol ® is marked on an attribute, whenever there is a change in

the value of the marked attribute, following information will be recorded by default: 1) the *ID* of the entity that caused/performed the change; 2) the *old value* of the attribute before the change; 3) the *new value* of the attribute; 4) the *time-stamp* that the change occurs. These items are necessary for replaying the marked attribute.

A rule recorder aims at recording which rules are triggered and what the corresponding values of related attributes are. An *implicit rule* will be triggered if its value constraint is violated by a update. In this circumstance, the recorded information includes 1) the *ID* of the entity the attribute is attached to; 2) the *old value* of the attribute; 3) the *new* value; 4) the *time-stamp*; and 5) the *rule ID*. For an *explicit rule*, if one of its assertions becomes false, the *explicit rule* will be triggered. In this case, the values associated with the *logic expression* as well as the values of the corresponding *scope expression* are the main targets for the rule recorder. In addition, the relative entities' IDs and the time-stamp will also be recorded. In COINS diagram notation, the symbol ® is also used for declaring rule recorders.

(a) Value recorder (b) Value recorder and rule recorder

Figure 2. Examples of declaring value recorder and rule recorder

Figure 2 shows three recorders, in which two of them are value recorders and the other is a rule recorder. For the two value recorders, whenever the visibility of a runway or the duty hour of a pilot within one day changes, the described information units will be recorded as permanent records for the replaying mechanism. As to the rule recorder, if the duty hours of a pilot within one day will exceed 8 hours because of a new assignment of a flight, then the rule R(01) will be triggered, and accordingly, following items will be recorded by the rule recorder: the pilot ID, the rule ID, the original duty hour, the new duty hour, and the time-stamp.

3.4 Scene Recorder

In the ER model, related entities are connected by relationships. At any given time unit, these connected entities and their attributes form a cluster of complex information within which atomic information units hold some logic connections with each other. This paper refers such a cluster of complex information as an ***information scene***. In terms of ER model, an *information scene* is a snapshot of linking-relations among certain entities and relationships at a given time unit. When such entities and relationships are of concrete value/contents for an information scene, it yields an instance of the information scene.

In COINS diagrams, a scene recorder is specified by a diamond with a recorder symbol ® on the top (refer to Figure 4 for example). Rules can also be attached to relationships. In addition to specifying logic connections among entities and attributes, rules can also be used to define how an instance of an information scene is created. In other words, the information units of a scene should be created automatically and the scene in COINS defines how to create these information units.

3.5 Replaying Mechanism

Generally an information/database system responds to queries with **static** information. The ability of such an information/database system to responses to queries on past history with **dynamic** and **evolving** information has not been fully developed. Based on the recording mechanism of the COINS modelling approach, a replaying mechanism providing such ability can be developed.

A simple way for replaying recorded instances of information scenes is to present them based on the time order. That is, the recorded information will be shown with the timestamp that the changes of the targeted information occur. An information unit or certain information units can also be taken as a reference for displaying records of scene instances. In other words, the representation of the replaying history will focus on the evolution of certain information units, therefore its changing history will be shown sequentially, while other scenes whose time-stamps matches the time-stamp of the focused information scene can also be displayed accordingly.

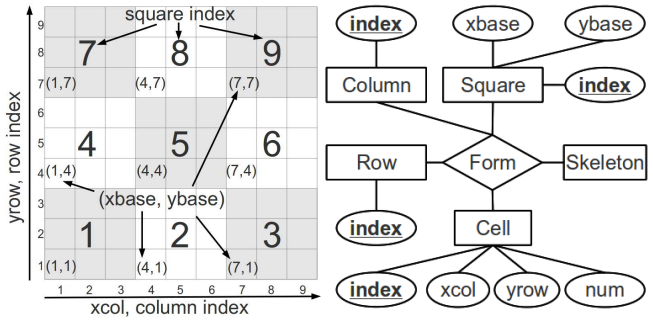
A more complex way to replay recorded information is to display designated scenes within a time interval or more generally, under given conditions. Such an approach would integrate the query mechanism and the visualisation techniques into the replaying process so as to facilitate the examination or inspection of the archived information.

4. APPLICATION TO SUDOKU

4.1 The COINS Model for Sudoku

We start the Sudoku example by presenting its structural information with an ER model. Figure 3(a) gives a classical layout of the Sudoku Game with necessary notations. The concepts of *Row*, *Column*, *Square* and *Cell* are self-evident. The concept *Skeleton* refers to the arrangement of all cells and how each rows, columns and squares are defined. The *Skeleton* of the classic Sudoku Game contains 81 cells, 9 rows, 9 columns and 9 squares. A Sudoku puzzle is a Sudoku skeleton being filled with numbers for its cells in such a way that each of its 81 cells can either be blank, or contain a number within the range of 1 to 9. The key constraint for such Sudoku puzzles is that the number of a cell in 1) each row, 2) each column and, 3) each square must be unique within its corresponding row, column and square, respectively. The purpose the Sudoku Game is to solve a given Sudoku puzzle by filling all its blank cells under such a constraint. A valid Sudoku puzzle must have and only have one solution.

Figure 3(b) presents an ER model for Sudoku based upon those notations. In this model, *Form* is a five-ary relationship that connects the five entities together. It is for structural information between *Skeleton*, *Row*, *Column*, *Square*, and *Cell*. That is, in a classical Sudoku game, the skeleton has 81 cells which are grouped by 9 rows, 9 columns and 9 squares. In the Sudoku model, attributes *index*, *xcol*, *yrow* and *num* shall be an integer between 1 and 9. For a valid Sudoku puzzle, all values of cells either in a row, a column or a square shall remain unique. The rule mechanism in COINS is employed to specify this kind of constraints.



(a) Soduku skeleton (b) The ER model

Figure 3. Sudoku skeleton & the ER model

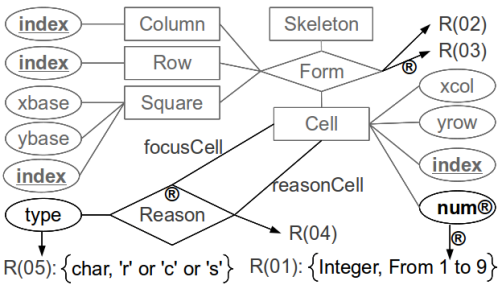


Figure 4. COINS model for Sudoku

4.2 Rule Mechanism in Sudoku

In addition to the ER model, COINS has a rule mechanism for capturing additional semantics. Figure 4 provides a COINS model for the Sudoku game, which is extended from the ER model shown in Figure 3(b). The attribute *num* of *Cell* entity is attached with an *explicit rule* R(01), which requests *num* to be an integer with a value range of 1 to 9. In the COINS model R(n) means a rule and the number *n* is the index of that rule in the whole model. Two rules, R(02) and R(03), are imposed on relationship *Form*. R(02) describes the *part-of* relationships between instances, while R(03) specifies the requirement that all numbers in each block should be unique. These two rules are shown in Figure 5(a) and Figure 5(b) respectively.

$$
\begin{array}{l}
(Cell, \quad isPartof, \quad skeleton) \\
(Row, \quad isPartof, \quad skeleton) \\
(Column, \quad isPartof, \quad skeleton) \\
(Square, \quad isPartof, \quad skeleton) \\
\left\{ \begin{array}{l}
\forall\ cell\ \in\ Cell \\
\left\{ \begin{array}{l}
\forall\ row\ \in\ skeleton.Row \\
\quad When:\ cell.yrow, ==, row.index \\
\quad\quad (cell,\ isPartof,\ row) \\
\forall\ column\ \in\ skeleton.Column \\
\quad When:\ cell.xcol, ==, column.index \\
\quad\quad (cell,\ isPartof,\ column) \\
\forall\ square\ \in\ skeleton.Square \\
\quad When: \left\{ \begin{array}{llll}
cell.xcol, & \geq, & square.xbase & and \\
cell.xcol, & \leq, & square.xbase + 2 & and \\
cell.yrow, & \geq, & square.ybase & and \\
cell.yrow, & \leq, & square.ybase + 2 &
\end{array} \right. \\
\quad\quad (cell,\ isPartof,\ square)
\end{array} \right.
\end{array} \right.
\end{array}
$$

(a) Rule R(02)

$$
\left\{ \begin{array}{l}
\forall\ skeleton \in Skeleton \\
\left\{ \begin{array}{l}
\forall\ row\ \in\ skeleton.Row \\
\quad \forall\ cell\ \in\ row.Cell \\
\quad\quad (cell.num,\ isUniqueto,\ row.Cell.num) \\
\forall\ column\ \in\ skeleton.Column \\
\quad \forall\ cell\ \in\ column.Cell \\
\quad\quad (cell.num,\ isUniqueto,\ column.Cell.num) \\
\forall\ square\ \in\ skeleton.Square \\
\quad \forall\ cell\ \in\ square.Cell \\
\quad\quad (cell.num,\ isUniqueto,\ square.Cell.num)
\end{array} \right.
\end{array} \right.
$$

(b) Rule R(03)

Figure 5. COINS model for rule R(02) & rule R(03)

In R(02), the operator *isPartof* declares a part-of relationship among instances and entities. In any rule, an entity name starting with an upper case represents the set of instances of the entity; an entity name expressed with all lower case letters represents a particular instance of

that entity. Therefore, in the logic expression (*Row*, *isPartof*, *skeleton*), *Row* represents all instances of the entity *Row*; the lower case of entity name *skeleton* refers to an instance of the entity *Skeleton* which is valid within the given scope. In this particular Sudoku example (classic 9x9 Sudoku), the entity *Skeleton* has only one instance. Therefore the word *skeleton* appeared in the expression refers to that unique instance. The whole expression specifies that every instance of the *Row* entity is a part of the only instance of the *Skeleton* entity.

The purpose of R(03) is to specify the feature that the number in each block, should be unique. In the scope of expression (∀ skeleton ∈ Skeleton), three independent explicit rule components are defined for each row, each column and each square. Each component starts with two nested scope expression and a logic expression. Here, *skeleton.Row* in the outer scope expression (∀ *row* ∈ *skeleton.Row*) is the full set of instances of Row belongs to this particular skeleton. Accordingly, the element *row* in the scope expression refers to a row instance. In the inner scope expression (∀ *cell* ∈ *row.Cell*), *row.Cell* refers to the full set of cell instances that belongs to the instance *row*, therefore *cell* in the expression is an element of that set. As to the logic expression, we define that *isUniqueto* is a binary operator. The left operand is an element and the right operand is a set. The semantics of the expression is straightforward: the element should remain unique in the set. In *cell.num*, the dot symbol '.' is used to access the value of the attribute *num* of the instance cell. For *row.Cell.num*, it refers to a collection of attributes which have the name *num* and belong to *row.Cell*.

4.3 Recorders for Sudoku

In the diagram notation of the COINS model, the symbol ® is used to declare which attribute or rule should be recorded. As shown in Figure 3, the attribute *num* of entity *Cell* requires a value recorder. R(01) and R(03) also has rule recorders. The value recorder for *num* indicates that whenever users update the number of a cell, such operations will be recorded. Once the puzzle is solved, the saved records as a whole can be treated as the knowledge for solving the puzzle. The wrong operations that violate the Sudoku rules specified by R(01) and R(03) will also be recorded by corresponding rule recorders. Accordingly, these exceptional operations will be distinguished automatically from the normal ones at the time they occur. Moreover, these exceptional operations can be easily reproduced through the replaying mechanism.

As to the scene recorder of COINS model, an example of how to define a scene is illustrated in Figure 4. In this example, a relationship named as *Reason*, which is attached to entity *Cell*, is defined to describe an information scene that may appear when playing the Sudoku game. This information scene aims to grasp the reasons why only a certain number can be filled into a particular cell and others cannot. That is to say, given a blank cell in Sudoku puzzle, some strategies of the game may determine which numbers the cell can be filled. The scene *Reason* attempts to capture the concrete strategy used in filling the blank cell. In detail, the participated entities *focusCell* and *reasonCell* specify two information units. The *type* attribute defines the third information unit, which is a character and its value can only be ***r***, ***c*** or ***s*** as specified by rule R(5). The meaning of ***r***, ***c*** or ***s*** indicates the reason for an action, which will be explained later in this section.

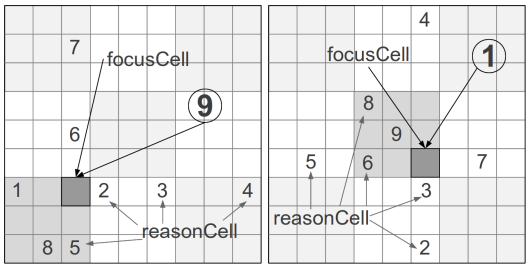


Figure 6. Relations between focusCell & reasonCell

$$
\begin{cases}
\forall\ cell\ \in\ (focusCell \rightarrow Row).Cell \\
\quad When: \begin{cases} cell.index, & !=, & focusCell.index \quad and \\ cell.num, & !=, & Null \end{cases} \\
\quad (reasonCell.index,\ \leftarrow,\ cell.index) \\
\quad (type,\ \leftarrow,\ r)
\end{cases}
$$

$$
\begin{cases}
\forall\ cell\ \in\ (focusCell \rightarrow Column).Cell \\
\quad When: \begin{cases} cell.index, & !=, & focusCell.index \quad and \\ cell.num, & !=, & Null \end{cases} \\
\quad (reasonCell.index,\ \leftarrow,\ cell.index) \\
\quad (type,\ \leftarrow,\ c)
\end{cases}
$$

$$
\begin{cases}
\forall\ cell\ \in\ (focusCell \rightarrow Square).Cell \\
\quad When: \begin{cases} cell.index, & !=, & focusCell.index \quad and \\ cell.num, & !=, & Null \end{cases} \\
\quad (reasonCell.index,\ \leftarrow,\ cell.index) \\
\quad (type,\ \leftarrow,\ s)
\end{cases}
$$

Figure 7. R(04) of the COINS model

In order to illustrate the relations between *focusCell* and *reasonCell*, Figure 6 presents two examples. In Figure 6(a), integers from 1 to 8 have appeared in the row, the column and the square that *focusCell* belongs to; therefore only number 9 can be set to *focusCell*, and integers from 1 to 8 are the reasons why *focusCell* is 9. In Figure 6(b), integers from 2 to 9 have appeared in the corresponding row, column and square, therefore *focusCell* can only be 1. In order to capture such information clusters, R(04) defined in Figure 7 is attached to the relationship *Reason*.

Three independent sub rules are defined for rows, columns and squares respectively in R(04). Each sub rule has two nested scope expressions and two parallel assign expressions. Following explanation takes the first sub rule as an example. In the outer scope expression, (*focusCell* → *Row*) refers to the row instance that *focusCell* belongs to. Therefore (*focusCell* → *Row*).*Cell* then represents a set of cells that belong to the row instance. Accordingly, *cell* appeared in the rule refers to an element of that set. In short, *cell* refers to the cell instance that belongs to the Row instance that includes *focusCell*.

The inner scope expression which starts with 'When' specifies two additional conditions: a *cell* is different from the *focusCell* and, the *cell* is not blank. If these conditions are satisfied, the *cell* can be regarded as one of reasons for putting a value into the *focusCell*. Therefore we assign *cell.index* to *reasonCell.index* and record the type for this reason as ***r*** which means row. Similarly, the characters ***c*** and ***s*** represent column and square respectively. For a given *focusCell*, the corresponding reason cells can be determined after executing R(04). These reasoning cells and the focused cell compose an information cluster; say an information scene which may appear when the value of *focusCell* is changed. The semantics of the scene recorder declared in Figure 4 is that all scenes specified by R(04) will be recorded and thereafter can be replayed.

4.4 Replaying Mechanism in Sudoku

As an example for demonstrating the replaying mechanism in COINS, a prototype system (Zhou, et al., 2011) based on Sudoku is developed. The prototype system satisfies the need for facilitating the querying process by extracting and presenting the recorded information units through a replaying mechanism, which treats a solving process as a sequence of information scenes, just as a media player treats a video as a sequence of images.

Four main panels are designed in the prototype system: 1) a play panel, which is the main area for displaying the recorded information units; 2) an index panel, where each operation is visualised as a shape and the detail information about the operation can be retrieved by interacting with the shape; 3) a control panel, which not only provides buttons for replaying a solving process, but also offers a slider with a drag-able square so as to enhance the replaying experience; 4) a query panel, where natural language queries about the recorded information can be proposed.

Figure 8 shows three screen-shots of the play panel when replaying a solving process. Each screen-shot presents an information scene, which consists of a focus cell (the target information which is marked with a diamond) and several reason cells (the context information which is marked with circles). These information scenes are recorded automatically when users solve puzzles.

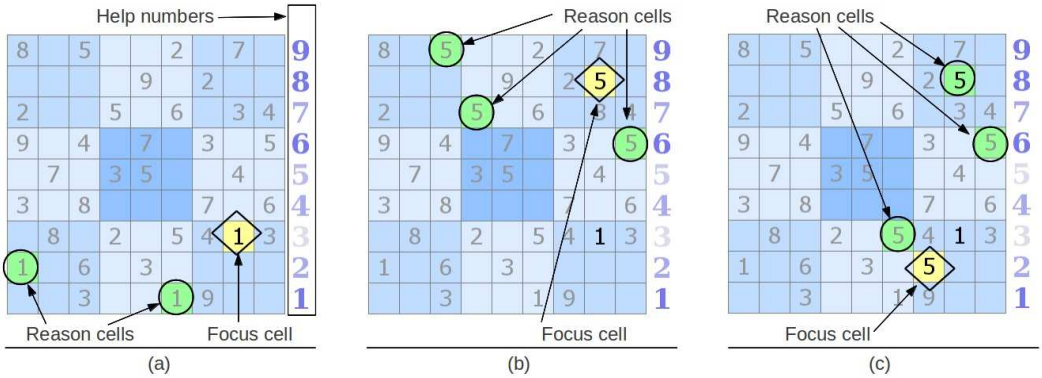


Figure 8. The screen-shots of the play panel, where the first three information scenes are presented

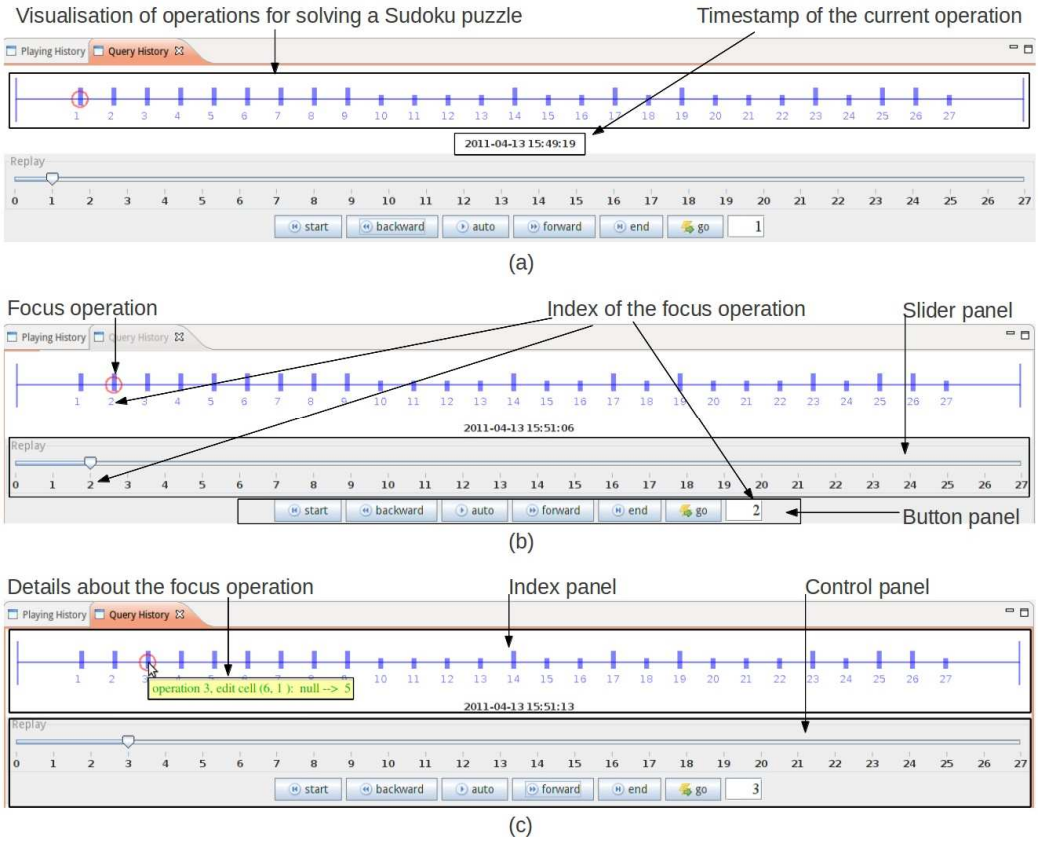


Figure 9. The screen-shots of the index/control panel when replaying the first three information scenes

The help numbers shown on the right of the play panel not only emphasise the row index in the panel, but also present the distribution of the determined numbers in the current puzzle layout. A determined number is a puzzle number, or a number (from 1 to 9) that a user gives to a cell and it does not cause any error under the current puzzle layout. The more the number is determined in a puzzle layout, the lighter the colour of the number on the right will be.

The control panel, which consists of a slider panel and a button panel, is used to control the process of replaying the recorded information scenes. Figure 9 presents the corresponding screen-shots of the index panel and the control panel when displaying the first three information scenes. In detail, Figure 9a is corresponding to Figure 8a; Figure 9b is corresponding to Figure 8b; Figure 9c is corresponding to Figure 8c.

In the index panel, each square represents an operation, which may be a normal operation, or a sub-operation (of a normal operation). Comparing to a normal operation, a sub-operation is represented as a shorter square. A square with a circle represents the focus operation, which means the corresponding puzzle layout of the operation is currently presented in the puzzle panel.

Three ways are provided in the control panel for controlling how the recorded information can be replayed. The tick on the slider can be dragged forward or backward for displaying the solving process. The control buttons can also be used to replay the history step by step

whether manually or automatically. Users can also give an index number so as to show the corresponding recorded information by pressing the "go" button.

When users give orders in the control panel, the corresponding information scenes will be displayed in the puzzle panel. Three kinds of cells are distinguished in a replaying process: the *focus cell* that users updated in the given index; the *error cells* which violates the Sudoku rules; and the *reason cells* that compose a scene defined by rules. The replaying process is dynamically changing with different solving steps. Figure 8 and Figure 9 only show some static images in this process.

4.5 Queries for Replaying Processes

Section 3.5 mentioned that an information unit or certain information units can also be taken as a reference for displaying records of scene instances. In the Sudoku example, such replaying method can be illustrated by giving certain cells that users are interested in, and accordingly, the prototype system will only show the information scene instances that are related to the given cells.

Furthermore, the prototype system will automatically run in "*replay cell*" mode: the functions of buttons in the control panel will change so that only those related scene instances will be displayed. In detail, when receiving the click of the "*start*" ("*end*") button, the prototype system will show the first (last) scene instance that are related to one of the given cells; when receiving the click of the "*forward*" ("*backward*") button, the prototype system will display the next (previous) scene instance.

For example, if a user only wants to view the information scenes related to $cell_{(2,2)}$ and $cell_{(0,7)}$ of a solving process, he can select the process and give command "*replay cell 2 2, 0 7*" to the prototype system. After receiving this command, the prototype system will automatically replay the information scenes that are related to the update of $cell_{(2,2)}$ and $cell_{(0,7)}$. And accordingly, the buttons in the control panel will change to suit the query of information scenes that are related to $cell_{(2,2)}$ and $cell_{(0,7)}$.

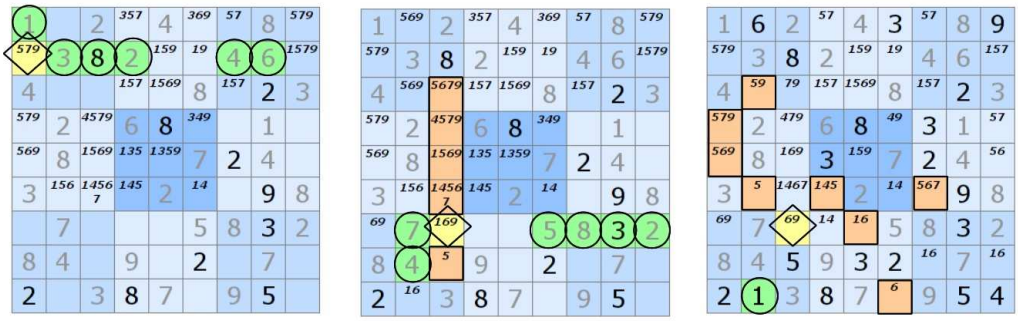


Figure 10. The first three screen-shots of the play panel after executing command "*replay cell 2 2, 0 7*"

Figure 10 and Figure 11 illustrate how this kind of query is executed. Figure 10 presents the first three screen-shots of the play panel when executing the command, and each of them presents a puzzle layout that the value of $cell_{(2,2)}$ or $cell_{(0,7)}$ is updated. These three puzzle layouts are automatically replayed when executing the command. In Figure 10, the *focus cell*,

error cells and *reason cells* are marked with a black diamond, black circles and blank rectangles respectively. Figure 11 presents the corresponding screen shots of the control panel and the index panel when presenting the three puzzle layouts.

In addition to simple queries like "*replay cell 2 2, 0 7*", queries stated with natural language, such as "*count the advanced puzzle which Peter played before the second week in May 2009*" can be directly answered. It is beyond the scope of this paper to fully explain the natural language querying part.

Figure 11. Screen-shots of the control panel and index panel after executing command "*replay cell 2 2, 0 7*"

5. CONCLUSION

Computerising best practice or domain knowledge results in complex information which not only involves structures but also holds logic relations. This paper presented a modelling method which incorporates the idea of recording and replaying information units or scenes so as to provide a permanent live interactive record of information that can be queried, played and replayed.

Although this paper illustrated the COINS approach with the puzzle game Sudoku, the requirements of this approach come from realistic application domains where security and safety are important factors. For example, whenever in airline operation control or in human health care, the authorities need to ensure that all the operations of practitioners are in line with the pre-regulated rules. Thus the practitioners are required to retain the history records of operational activities so that they can be checked and inspected in future. Currently, audio, videos and documents are three main formats of recorded information. Although it is easy to

record a telephone line or a computer screen, it is impractical to retrieve or extract the interactions between users and information systems by using the audio or video files.

The COINS approach addresses this kind of requirements. The rule mechanism, the recording and replaying mechanism in COINS would help to develop computerised systems for these application domains.

As to the future work, further research is needed to design a specification language for the COINS model and to develop an interpreter for such a language. Mechanisms for integrating such an interpreter into the framework also need to be investigated.

ACKNOWLEDGEMENT

The authors would like to thank the China Scholarship Council (CSC) and Dublin Institute of Technology (DIT) for their funding and support.

REFERENCES

Altekar, G. & Stoica, I. (2009). ODR: Output-deterministic Replay for Multicore Debugging. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ACM, New York, NY, USA, pp 193–206.

Brown, A. & Patterson, D. A. (2002). Rewind, Repair, Replay: Three R's to Dependability. In *the 10th ACM SIGOPS European Workshop*, Saint-Emilion, France.

Chen, P. P.-S. (1976). The Entity-Relationship Model–toward A Unified View of Data. *ACM Trans. Database Syst.*, Vol. 1, pp. 9–36.

Chow, J., Lucchetti, D., et al. (2010). Multi-stage Replay with Crosscut. In *Proceedings of the 6th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, ACM, New York, USA, pp13–24.

Dube, K. (2004). A Generic Approach to Supporting the Management of Computerised Clinical Guidelines and Protocols. *Ph. D. thesis*, Dublin Institute of Technology.

Galanis, L., Buranawatanachoke, S., et al. (2008). Oracle database replay. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, Vancouver, Canada, pp. 1159–1170.

Greenhalgh, C., J. Pubrick, et al. (2000). Temporal Links: Recording and Replaying Virtual Environments. In *8th ACM International Conference on Multimedia (ACM MM'00)*, Los Angeles, CA, USA.

Guo, Z., Wang, X., et al. (2008). R2: An Application-level Kernel for Record and Replay. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, Berkeley, CA, USA, pp 193–208.

Manohar, N. R. & Prakash, A. (1994). Replay by Re-execution: A Paradigm for Asynchronous Collaboration via Record and Replay of Interactive Multimedia Sessions. *SIGOIS Bulletin* Vol. 15, No. 2, pp. 32 - 34.

Mansour, E. (2008). A Generic Approach and Framework for Managing Complex Information. *Ph. D. thesis*, Dublin Institute of Technology.

Mansour, E. & Höpfner, H. (2009). Replay the Execution History of Rule-based Information. In *Proceedings of the First International Conference on Advances in Databases, Knowledge, and Data Applications*, Cancun, Mexico, pp 28–35.

Narayanasamy, S., Pokam, G., et al. (2005). Bugnet: Continuously Recording Program Execution for Deterministic Replay Debugging. In *Proceedings of the 32nd annual international symposium on Computer Architecture* , Washington, DC, USA, pp. 284–295.

Wu, B. & Dube, K. (2001). Plan: a Framework and Specification Language with an Event-Condition-Action (ECA) Mechanism for Clinical Test Request Protocols. In *Proceedings of the 34th Hawaii International Conference on System Sciences*.

Wu, B., Mansore, E., et al. (2007). Complex Information Management Using a Framework Supported by ECA Rules in XML. In *the International RuleML Symposium on Rule Interchange and Applications (RuleML-2007)*, pp. 224–231.

Wu, B., Dube, K., et al. (2008). The Motion Picture Paradigm for Managing Information: A framework and approach to supporting the play and re-play of information in computerised information system. In *the 10th International Conference on Enterprise Information Systems (ICEIS 2008)*.

Zhou, E., Wu, B., et al. (2011). An Architecture and a Prototype for Querying and Visualising Recorded Context Information. In *Proceedings of CONTEXT '11: The Seventh International and Interdisciplinary Conference on Modeling and Using Context*. Karlsruhe, Germany, pp. 321—334.