A Framework for the E-R Computational Creativity Model

Rodrigo García[1], Pablo Gervás[2], Raquel Hervás[2], Rafael Pérez y Pérez[1], and Fernando ArÃmbula[1]

[1] Posgrado en Ciencia e Ingenieria de la Computacion, Universidad Nacional Autonoma de Mexico, Mexico
rodrigog@uxmcc2.iimas.unam.mx, rpyp@servidor.unam.mx, arambula@aleph.cinstrum.unam.mx

[2] Departamento de Sistemas Informaticos y Programacion, Universidad Complutense de Madrid, Spain
pgervas@sip.ucm.es, raquelhb@fdi.ucm.es

Abstract. This paper presents an object-oriented framework based on the E-R computational creativity model. It proposes a generic architecture for solving problems that require a certain amount of creativity. The design is based on advanced Software Engineering concepts for object-oriented Framework Design. With the use of the proposed framework, the knowledge of the E-R computational model can be easily extended. This model is important since it tries to diagram the human creativity process when a human activity is done. The framework is described together with two applications under development which implement the framework.

1 Introduction

Humans apply creative solutions across a wide range of problems: music, art, science, literature...If a common plausible model were found of the way humans approach creativity problems in all these fields, it would open possibilities of applying creative mechanisms to problems in a wide range of domains. The Engagement and Reflection model of the creative process developed by Perez y Perez [1] attempted to abstract the way in which the human brain tackles creative composition in the field of storytelling. However, the Engagement and Reflection model is postulated as independent of particular domains. Several efforts are under way to apply it to different tasks - geometry, storytelling, image interpretation... From the point of view of development, it would be extremely interesting if the essence of the computational model, which is common across different applications, could be captured in some kind of abstract and reusable software solution. This paper explores the design of an object oriented framework intended to capture in this way the common functionalities of computational solutions based on the Engagement and Reflection model for addressing creativity problems.

A framework is a set of classes and interfaces closely related in a reusable form design for a family of systems with a strong structural connection (hierarchy of classes, inheritance and composition) and of behavior (model of interaction of the objects) [2]. When an application is implemented based on a framework it is said that it is an *instance* of the framework. This means that the framework's *hotpoints* - places where details

A. Gelbukh and C.A. Reyes-Garcia (Eds.): MICAI 2006, LNAI 4293, pp. 70–80, 2006.
© Springer-Verlag Berlin Heidelberg 2006

and specific fragments of code concerning a particular domain must be provided - are specified to transform it into a concrete application. The framework can be seen as the skeleton that supports the general structure and the hotpoints provide the flexibility required to obtain different applications by different instantiation processes.

The fundamental advantage of frameworks is that they can significantly reduce the development time for particular applications in the selected domain because of design reuse. But there are also disadvantages that have to be considered. By introducing a common structure for applications in a given domain, a framework may effectively restrict the range of alternatives that a designer can consider. This can have unforeseen consequences in terms of restricting the creative freedom of the applications that we are contemplating.

In spite of this disadvantage, a framework can be a good solution for capturing particular methods of approaching problem solving that can be applied accross several domains. In this paper, we work under the assumption that the computer model based on Engagement and Reflective States can be applied to different fields such as story development, image interpretation and graphs generation.

This paper is organized as follows. In section 2 *previous work* related to Design Patterns and Frameworks and the Engagement and Reflection model is presented. Section 3 shows the *design* of the framework architecture and its components. Section 4 describes *three instantiation examples* related to storytelling, image interpretation and graph generation. Section 5 presents a *discussion* about the generalization of the E-R creativity model. Finally, section 6 outlines *conclusions* and future work.

2 Previous Work

To design a framework for Engagement and Reflection computational models of the creative process, relevant work on two different fields must be considered: framework design and the Engagement and Reflection model.

2.1 Design Patterns and Frameworks

The development of a framework requires a significant effort of domain analysis. In order to identify the ingredients that are common across different aplications of a given type, several examples must be analysed carefully. Before building a framework in a given domain one should have a solid understanding of the domain, ideally as result of the experience gained in building prior applications in that domain.

Another important aspect concerning frameworks is the stages of evolution they pass during their lifetime [3]. According to Tracz [4], to acquire sufficient knowledge to identify the reusable essence for building artefacts of a given kind one must have built at least three different examples of such artefacts. This is considered the first stage in the evolution of a framework. The second stage is a *white box framework*: the framework provides a bare structure in which the user will have to introduce actual fragments of code adapted to the particular domain in which he wants the framework to operate. The user has to understand how the different modules in the framework work, and he may have to write software components himself. The third stage is a *black box framework*:

the framework provides a structure and a set of software components - organised as a *component library* - which constitute different alternatives for instantiating modules of the framework. A user may put together an instance of the framework simply by assembling elements from the component library into the framework structure. Later stages in the evolution of a framework gravitate towards obtaining a visual builder interface, to make even easier the process of building applications. However, this refinement in not necessary in most practical applications.

2.2 Engagement and Reflection

The main goal of Engagement and Reflection (E-R) model is to provide a model of the way in which human beings go about the task of applying abstract knowledge to creative tasks. Human beings store an enormous amount of abstract knowledge, refined from a lifetime of experience. This knowledge is used for solving problems. The Engagement and Reflection model is a plausible representation of the process a human being follows when trying to solve a problem that requires the use of abstract knowledge.

The basic unit of representation in the Engagement and Reflection model is an action. An action has a set of preconditions and a set of postconditions.

The model is based in two main processes that form a cycle, Engagement and Reflection. During Engagement we produce a lot of ideas - or instances of some equivalent conceptual material - that help us by acting as cues to solve the problem. At this stage, restrictions such as the fulfillment of the preconditions of an action within a given plan are not evaluated. The generation of these ideas is driven by a set of parameters or constraints that have to be defined. This ensures that the generation process is guided towards a specific goal. In the Reflection stage, the ideas generated during Engagement are evaluated carefully, restrictions such as the fulfillment of preconditions are enforced, and any required modifications are carried out to ensure that the partial result at any given stage is coherent and correct.

The process of solving a problem follows a cycle of transitions between the Engagement and the Reflection states. At each pass through the Engagement state more material is added. At each pass through the Reflection state, the accumulated material is checked for consistency, completed and corrected.

The solution of a problem is a train of well structured actions based on the previous knowledge of solved problems. If we can extract the preconditions and the postconditions of the actions problems it can be reuse for the solution of other problems in the same domain.

The E-R model was conceived for the creative process in writing. In later research efforts, the model has been applied in other fields like the solution of geometric problems, image interpretation problems and strategic games. These last two examples are currently under development.

MEXICA. The E-R model was originally used in MEXICA [1]. MEXICA was designed to study the creative process involved in writing in terms or the cycle of engagement and reflection. MEXICA's stories are represented as sequences of actions. MEXICA has two main processes: the first creates all data structures in memory from information provided by the user. The second, based on such structures and as a result of a cycle

between engagement and reflection, produces new stories. It has the next goals: (1) To produce stories as a result of an interaction between engagement and reflection. (2) To produce material during engagement without the use of problem-solving techniques or predefined story-structures. (3) To produce novel and interesting stories. (4)Allow users to experiment with different parameters that constraint the writing process.

The Geometrician: A Computer Model for Solving Geometry Problems. Based on the creative model E-R, Villaseñor [5] tries to solve geometric problems with the use of rule and compass only. The user defines a text file with a set of solved problems, then the key information from these problems is extracted and it is used as a knowledge base for the system. When a new problem is presented to the program, it tries to find a solution as a result of the interaction between engagement and reflection. During engagement the program looks for actions in memory that could be done in order to solve the problem, and during reflection those actions retrieved are checked before they are executed. The program implements some learning mechanisms and some new characteristics to the basic model (E-R). One of this new characteristics is the capability to solve sub-problems in a recursive way.

Image Interpretation. The problem consists in identifying the correct outline of a prostate in a Transurethral ultrasound image. Nowadays the experience of specialized doctor is necessary to identify this outline. The problem is not the time required to train doctors in this specific task, but the fact that the only way of acquiring this knowledge is during the process of real-life prostate operations. From the data provided by an ultrasound image, the only accurate knowledge about a prostate is conveyed as a white area surrounded by a dark zone. Based on the E-R model cycle of engagement and reflection and a Point Distribution Model (PDM) [6] used by Arambula [7] the program tries to find the most suitable outline of the prostate in Transurethral ultrasound images. As in the previous examples, the program needs a text file containing the solved problems. The first step of the process is to extract a set of characteristics of the image. Some of this characteristics could be the gray scale or the maximum brightness for example. This set of characteristics will help as cues for definition of the context which is the state of affair of the problem. The program then searches for similar contexts in the knowledge base acquired from the set of previously solved problems. Each of those contexts will have an associated set of related actions that contributed to the solution of the problem in the original case. These actions are added to the partial solution of the current problem during the engagement phase, and they are checked for consistency with the current problem later during the reflection stage.

3 Framework Design

The goal is to use software engineering techniques to develop a framework based on the E-R model, reducing development time for applications based on it. At the same time the framework - being a generic computational implementation of the E-R model will extend the knowledge about the human cognitive process the E-R model tries to abstract.

The framework is based on two examples: MEXICA and Image Interpretation. The implementation of the examples as instantiations of the framework is reviewed in Section 4.

For ease of understanding, the structure of the proposed framework is divided in two parts, one related to the core structure of the framework, and another one dealing with the specific features of the E-R model.

3.1 General Structure

The core structure of the framework is based on the architecture proposed in [8] for Natural Language Generation applications. Its general structure can be seen in Figure 1.

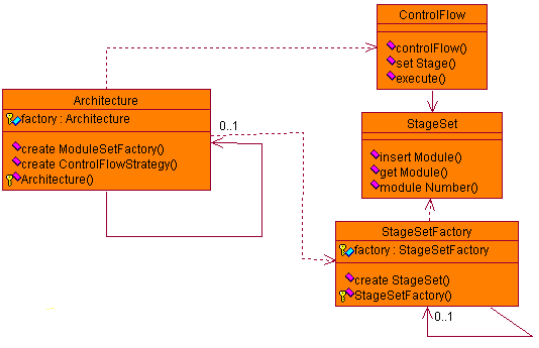

Fig. 1. General structure of the framework

The set of modules or stages involved in the process is stored in a `StageSet` structure. The choice of which modules to use is taken using the abstract class `StageSetFactory`. Its implementations, following the *AbstractFactory* design pattern [9], define specific sets of modules that are stored in `StageSet`.

With regard to the flow of control information, the decision is taken in the abstract class `ControlFlow`, implemented following the *Strategy* design pattern [9]. A `StageSet` is passed as parameter to the constructor of `ControlFlow`, so that the control flow knows which modules the user has decided to use. The goal of `ControlFlow` is to decide the arrangement and execution order of the stages kept in `StageSet`. Decisions as executing a stage more than one time, or deciding if executing it at all, are taken by the `ControlFlow` instantiations. To deal with that, `ControlFlow` has a *nextStage* method that returns the next step to be executed, and an *end* method that becomes "true" when there is no more stages to be executed.

Finally, connection between `ControlFlow` and `StageSetFactory` is found in the abstract class `ArchitectureFactory`, as in the *AbstractFactory* design pattern [9]. Given different set of stages and control flows, the user can decide which is the combination of modules and flow of control information he needs in his application.

3.2 E-R Structure

The E-R structure is the specific piece of the framework in charge to carry out the E-R creativity process. As shown in Figure 2, there are different data structures that

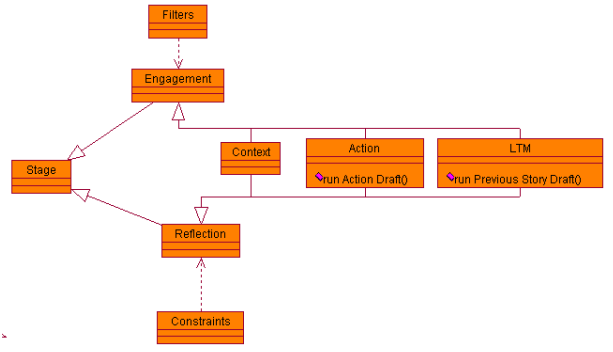

Fig. 2. E-R structure of the framework

interact depending on the actual step of the process. All of them are as a last resort descendants of the abstract class `Stage`, the one that is stored in the `StageSet` of the basic structure.

The E-R model has two main parts: Engagement and Reflection, both of them working with similar data structures. `Engagement` and `Reflection` abstract classes are used to help the user during the implementation of his system to know in which step is the process and to define the content of each class. Both of them have a special method used to query different data structures depending on the class implementation - filters for the Engagement, Constraints for the Reflection. The most important instantiations of the Engagement and Reflection classes are `Context`, `Action` and `LTM`. Depending on the stage of execution some modules will be connected and some others will be ignored.

The correct combination of these components, together with the basic structure in Section 3.1, will result in interesting and useful implementations of the framework.

4 Two Instantiation Examples

In order to show the use and the feasibility of the framework it will be applied to two examples: Image Interpretation and MEXICA.

4.1 Image Interpretation

The aim of the image interpretation problem is to draw as well as posible the form of the prostate in a Transurethral ultrasound images, such as the one in Figure 3. The black circle in the center of the image is the device used to get the ultrasound image. Important clues that may be used for solving the problem can be obtained from the knowledge of the human anatomy. For example, it is known that the rectal conduit is under the prostate. This is reflected in the image as a white region. Prostates are also known to be roughly pear-shaped. In this example all the stages are used, the context,

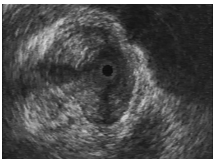

Fig. 3. Prostate Image

the actions, the LTM, the filters and the constraints. They are managed by the *StageSet* and coordinated by the *ControlFlow*.

The first step initiated by *ControlFlow* is to get a good context. This is the process by which the system constructs a general idea of the state of affairs. This is important since in some images it is easier to acquire the prostate form than in others. In order to get a context, the greater possible number of characteristics of the image must be extracted.

This is done by the execution of the action *Extract Characteristics* for example. The response of this action is a set of coordinates (x,y) and a graph as shown in Figures 4 and 5. This can be interpreted as an idea of the prostate form, but the most important is that now there is a **Context** to work with. Once this first step has been carried out, the *ControlFlow* sets in motion the next stage: ***Engagement***.

The Engagement stage checks the actual *Context* against its knowledge base of already solved problems - stored in the file of experiences or LTM - in search for the solved problem whose Context best matches with the Context of the current problem. To achieve this, the *run Previous Draft* method of the *LTM* class must be invoked. This method perfoms the search over the previous examples stored in the *LTM* class. Once a context is retrieved, depending on the filters introduced, an action is executed without checking the preconditions. This action is passed to the *run Action Draft* method of the *Action* class. The only requirement for executing an action is that it must modify the context. This is due to the fact that violation of this requirement may result in the system entering an infinite cycle. This step may be repeated one, two or three times for each example, depending on the requirements of the user.

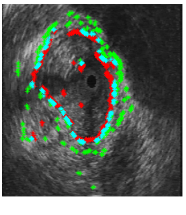

Fig. 4. Prostate Image with Characteristic extraction

Once the Engagement stage has finished, *ControlFlow* shifts control to the ***Reflection*** stage.

For Reflection all the stages are used. The basic idea of this step is to check the actions executed during in Engagement. Thus, all postconditions of each action should

be coherent with the preconditions of its follower, and the result of the executed action should bring the system closer to solving the problem. In order to achieve the first condition, the *run Action Draft method* is executed . Whenever a precondition is not fulfilled, it is explicitly asserted in the correct place in the sequence of actions. It may be possible that more than one precondition is not satisfied, so this step can be recursive until all preconditions are fulfilled. In this step there are also *Constraints* that must be tested. For example, a precondition of an action may be that the graph shown in Figure 5 must have at least two peak to execute the action number 5. Or it could be that the gradient of the line can not be more than 65 degrees. In this figure the **x** axis represents each profile of the prostate (0-359), and the **y** axis means the maximum brightness of each profile (0-255).

To check if the action executed is bringing the system closer to solving the problem, there are different techniques that can be applied. One possibility is to apply the form of the PDM [6] used by Arambula [7].

This whole process is repeated until the user's requirements have been satisfied or until a given value of a certain parameter is reached.

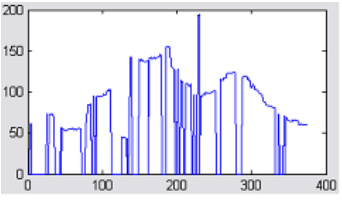

Fig. 5. Prostate Graphic

4.2 MEXICA

MEXICA's goal is to develop a computational model of the creative process of writing in terms of engagement and reflection. The environment of the story is controlled by the previous stories kept in the LTM. The first step in the *ControlFlow* is to form a context. The context here is to set an initial action, an initial scene and the number of characters. To achieve this, the special action *Initial Actions* must be called. One important feature of MEXICA is that the user has the option to manually set these initial data. This provides the means for guiding the output towards desireable results. Once the initial context is built, as in the Image interpretation problem, and depending on the users parameters, *ControlFlow* initiates the Engagement state.

In the Engagement state the *filters, context, action and LTM* modules are involved. The mission here is to retrieve from *LTM* a set of plausible actions to continue the story. As in the Image interpretation problem, the key condition in this step is that the action selected **must** change the state of affairs of the context. Once again the preconditions in this step are not considered when the action selection is made. Those will be considered in the Reflective state. Figure 6 shows a Previous Story file used in MEXICA.

The file includes characters, actions (aggressions, deaths, fights, cures...), scene movement and feelings (hate, love, jealousy...). The selection from the set of actions

Sto ;1	Sto ;2
Eagle_Knight Actor	Prince Went_Texcoco_Lake
Jaguar_Knight Actor	Prince Had_An_Accident
Eagle_Knight Was_In_Love_With Princess	Priest Found_By_Accident Prince
Jaguar_Knight Was_In_Love_With Princess	Priest Realised Prince Had_An_Accident
Princess Was_In_Love_With Warrior	Priest Cured Prince
Eagle_Knight Got_Jealous_Of Warrior	Prince Went_Palace
Eagle_Knight Killed Warrior	Fisherman Mugged Priest
Princess Attacked Eagle_Knight	Prince Realised Fisherman Mugged Priest
Eagle_Knight Wounded Princess	Prince Looked_For_And_Found Fisherman
Jaguar_Knight Attacked Eagle_Knight	Prince Made_Prisioner Fisherman
Jaguar_Knight Fought Eagle_Knight	
Jaguar_Knight Killed Eagle_Knight	
Jaguar_Knight Exiled Jaguar_Knight	

Fig. 6. Previous story file

retrieved is done based on the *filter* parameters. For example, an action can be discarded because it has been used more than twice in the actual story or because it does not modify the context. Next, the *ControlFlow* changes to the Reflection state. Here the actions selected during Engagement are checked according to constraints. Also, preconditions and the continuity of the history are verified. In Engagement the *context*, *action*, *LTM* and *constraints* modules are active.

5 Discussion

From the selected examples a set of possible actions to be undertaken during the process of carrying out the goal can be abstracted, and corresponding sets of preconditions and postconditions can be identified for each action in that set. This allows all three problems to be represented within the general schema that the model requires, being the actions the basic units of representation in the model.

For any particular implementation built using the proposed framework, the search space of possible solutions must be susceptible of being represented as a graph in which the nodes correspond to actions and the edges establish relationships of precondition and postcondition fulfilment between the actions. Such a graph can be seen as a tree like the one shown in Figure 7.

The set of possible complete solutions would then be represented by all possible paths from the root of the tree to one of its leaves. The image captures the fact that in many cases, the specification of the problem already contains explicitly a partial description of the solution - the partial image provided as input in the case of image interpretation; and the initial action, the initial scene and the number of characters in the case of MEXICA. The task of solving the problem corresponds to identifying the missing fragments that will turn this partial description of the solution into a complete solution. In the image, circles bounded by a full line indicate nodes of the solution already described explicitly in the specification of the problem, and circles bounded by a dotted line correspond to actions that must be identified by the system. As example,

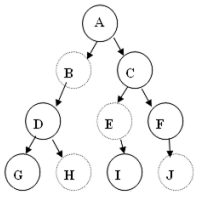

Fig. 7. Graph (tree) structure

a possible solution for a given problem may be the next sequence: A-B-D-H, but at the very beginning there is no idea of a potential solution just the letter A. Later, thinking about the problem, an analogy can be found with some other problem in the past and remembering the steps done for solving some specific problem in the past a clue can be found for solving the current problem. The first decision to be taken in the problem example represented in Figure 7 would correspond to identifying the node labelled with letter B as the next correct step towards a complete solution.

At the heart of the Engagement and Reflection creativity model lies a fundamental idea of the denial of intuition as driving force of human decision making. Under this point of view, when a person has several possibilities at the time of making a judgment, his decision is often related to an event in the past. This event need not be remembered explicitly, as it may be present only subconsciously. If this argument was used as guiding heuristic for a genetic algorithm, the thousands of possible answers that such an algorithm may give rise to might in fact be limited by the events and solutions that have proved succesful in the past. This should in no way be interpreted as a slur on genetic algorithms and the theory of problem solving that underlies them. It is simply a different way to think about the solution of problems.

6 Conclusions and Future Work

The E-R model is a good candidate for developing a reusable framework because it was originally intended as an abstract model of generic intellectual abilities of human beings.

In order to check the utility of the framework two projects are being developed. The first project is related to the implementation of MEXICA, a system that tells stories about the early inhabitants of Mexico. The second project is related to the Image Interpretation problem for Transurethral ultrasound images of the prostate. Both of them are being developed an instantiations of the E-R framework. The goal is to achieve operative implementations with less development effort than would have been required without the framework. The proposed framework contributes to this goal by allowing developers to focus on the key information such as the actions (preconditions and postconditions), filters and constraints.

In addition, when the use of the framework is reliable and efficient, the resulting ease of use may lead to wider adoption of the model and to more basic research on its theoretical underpinnings.

In spite of the abstract nature of the Engagement and Reflection model, and the efforts that have been made to make the framework as reusable as possible by the use of software engineering techniques, the scope of a framework is limited and not all kind of problems can be covered. However, it must be said that in general terms, the framework presented in this paper represents two advantages: it may make implementations of the Engagement and Reflection approach to problem solving faster, and it may serve to extend the use of the model.

References

1. Perez y Perez, R.: A Computer Model of Creativity in Writing. PhD thesis, University of Sussex (1999)
2. Johnson, R., Foote, B.: Designing reusable classes. Journal of object-Oriented Programming **1** (1988) 22–35
3. Johnson, R., Roberts., D.: Evolving frameworks: a pattern language for developing object-oriented frameworks. In: Proceedings of the 3rd Conference on Pattern Languages and Programming, Montecillo, Illinois. (1996)
4. Tracz, W.: In software reuse: Emerging technology. IEEE Computer Society Press (1988) 176–189
5. Acosta Villaseñor, E.: Aplicacion de un modelo en computadora del proceso creativo a la solucion de problemas en geometria. PhD thesis, Universidad Nacional Autonoma de Mexico (2005)
6. Cootes, T., Taylor, C., Cooper, D., Graham, J.: Active shape models-their training and application. Computer Vision and Image Understanding **61** (1995) 38–59
7. Cosio, F., Davies, B.: Automated prostate recognition: a key process for clinically effective robotic prostatectomy. Medical and Biological Engineering and Computing **37** (1999) 236–243
8. Garcia, C., Hervas, R., Gervas, P.: Una arquitectura software para el desarrollo de aplicaciones de generación de lenguaje natural. Sociedad Española para el Procesamiento del Lenguaje Natural **33** (2004) 111
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, USA, First Edition (1995)