# A Feature-Based Environment for Digital Games

Victor Sarinho, Antônio Apolinário, Eduardo Almeida



# A Feature-based Environment for Digital Games

Victor T. Sarinho[1], Antônio L. Apolinário Jr.[2] and Eduardo S. Almeida[3]

[1]DEXA – State University of Feira de Santana (UEFS), Feira de Santana, Bahia, Brazil
vsarinho@gmail.com
[2]Federal University of Bahia (UFBA), Salvador, Bahia, Brazil
[3]Reuse in Software Engineering (RiSE), Salvador, Bahia, Brazil
{apolinario, esa}@dcc.ufba.br

**Abstract.** Digital games can be considered as an important software development area in our society. This paper proposes the Object Oriented Feature Modeling (OOFM) usage in the digital game domain. It aims to represent and manipulate distinct game features, defined by NESI and GDS models, in a parameterized and hierarchical way. As a result, a Feature-based Environment for Digital Games (FEnDiGa) is provided, a product line platform able to integrate and adapt represented game features in different types of available game engines.

**Keywords:** Digital game domain, game features, OOFM, FEnDiGa.

## 1 Introduction

Digital games can be considered as an important software development area in our society. Analyzing the digital game domain, it is possible to verify multiple examples of game genres and categories presenting common features among them [1], distinct modeling approaches able to represent digital games [2, 3], and a great influence of game engines during game design and implementation activities [4].

This paper proposes the Object Oriented Feature Modeling (OOFM) [5] usage in the digital games domain. It aims to manage distinct *game features* provided by NESI [2] and GDS [3] models in a specialized OOFM structure, providing the *game logic*, *rules* and *goals* (the G-Factor [4]) in a *parameterized* and *hierarchical* way. As a result, the *Feature-based Environment for Digital Games* (FEnDiGa) is provided, a software structure able to *integrate* and *adapt* represented *game features* in different types of available game engines.

The reminder of this paper presents three game development steps necessary to provide and use the FEnDiGa structure: 1) the variability documentation of digital games in a generic way; 2) the creation of a feature-based environment and process able to work with generic game features; and 3) the configuration of feature game resources in order to provide a final game.

## 2 Step 1: Documenting the Game Variability

An interesting problem in the digital game domain is the representation of game characteristics. There are many types of game representation approaches in the literature presenting a lot of game elements, game behaviors and so on. In a previous work [2, 3], two distinct generic feature models were proposed (Fig. 1): the NESI model, documenting the conceptual variability of a digital game, and the GDS model, documenting the implementation variability of a digital game.

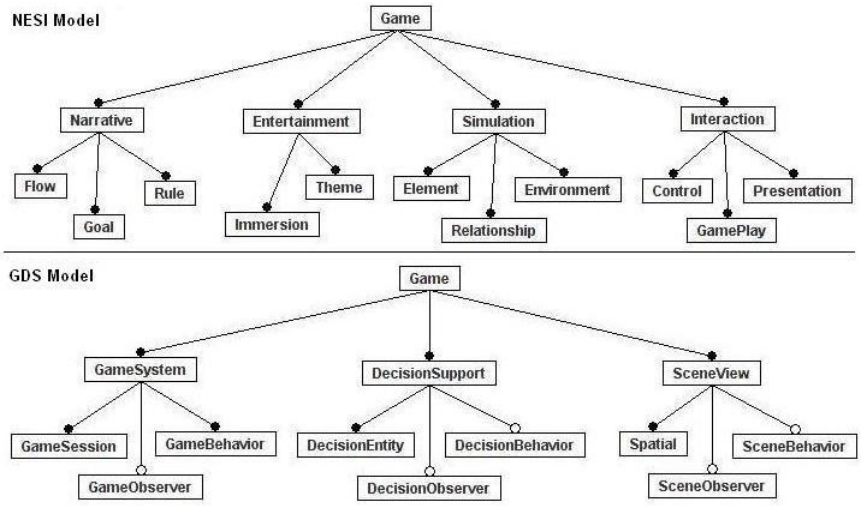


**Fig. 1.** The NESI and GDS models.

The NESI model [2] defines a digital game as a combination of *Narrative*, *Entertainment*, *Simulation* and *Interaction*. A *Narrative* is a *Flow*, a dynamic script trying to achieve *Goals* following defined *Rules* of a game. *Entertainment* is represented by the player *Immersion* during the gameplay, following a *Theme* proposal to integrate the player and the game in the proposed reality. *Simulation* is a combination of *Elements* (resources to play) and respective *Relationships* that happen in a defined *Environment* (spaces to play). The human *Interaction* is represented by *Control* (game inputs) and *Presentation* (game outputs), consolidated by the *GamePlay* that represents a current game execution as a whole. With these main features, other sub-features related to game concepts were defined, providing more than 350 conceptual game features to be configured and manipulated by game designers.

On the other hand, the GDS model [3] defines a digital game as a combination of *GameSystem*, *DecisionSupport* and *SceneView* main features, generating more than 250 related sub-features. The *GameSystem* feature is the main joint point of a game. It is responsible to control the game execution, describing available *GameBehaviors*, *GameContext* and *GameObservers* for a digital game. The *DecisionSupport* feature is an effort to integrate some artificial intelligence strategies used by different digital

games, presenting *DecisionEntities*, the *ContextState* of each *DecisionEntity*, *DecisionObservers* monitoring *DecisionEntities* values, and predefined *DecisionBehaviors* able to read and update *DecisionEntities* and *ContextStates* values during their executions. The *SceneView* feature is a collection of *SceneNodes* distributed by *Spatial* sessions with various *SceneBehaviors* and *SceneObservers* able to perform scene actions in a game. A *SceneNode* represents a collection of information about the scene, describing specific *Location* and *BoudingVolume* for collision detection among other scene nodes, as well as hierarchical information about *AudioNode*, *GraphicNode* and *PhysicsNode* simultaneously.

## 3 Step 2: Creating a Feature-Based Environment

NESI and GDS models describe generic and distinct Feature Models (FMs) and Feature Configurations (FC) for digital games. OOFM provides basic elements (*FeatureType*, *Feature*, and so on) to work with represented FMs and FCs. The *OOFM Framework* [5] is a combination of OOFM elements plus extra classes, interfaces and packages that follows the Model-View-Controller (MVC) architectural pattern [6]. It defines abstract *FeatureState*, *feature observer*, *FeatureBehavior* and *Adapter* structures in order to organize the production of final system for represented FMs and FCs.

*FEnDiGa* is a concrete specialization of the *OOFM Framework* focused on the digital game domain (Fig. 2). It is a game development environment that: *represents game features* based on NESI and GDS models via OOFM resources; *combines game features* using the proposed OOFM Framework structures, and *implements game features* using specialized adapters of desired game engines.

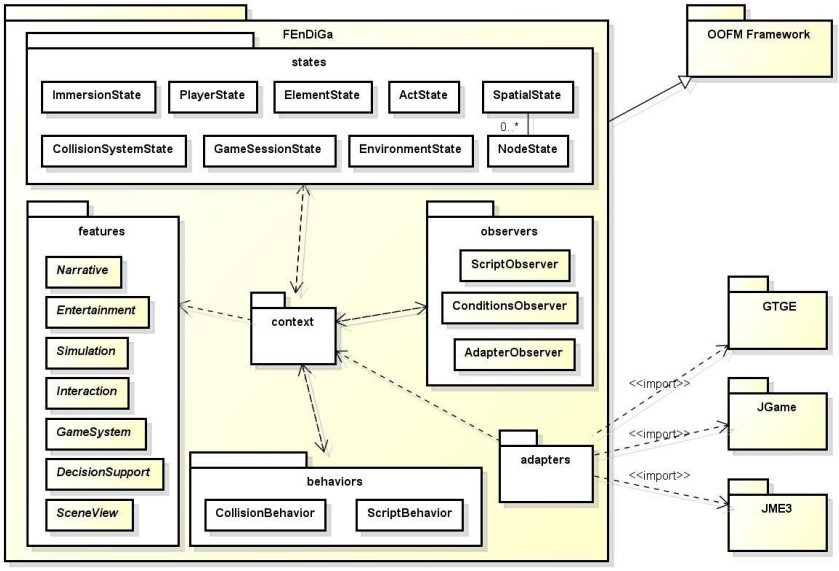


**Fig. 2.** The FEnDiGa architecture.

The *representation of game features* are performed by *FeatureClass* structures [5]. They are created for each main feature of the respective NESI and GDS models (game feature containers), presenting feature methods that prepare each *featureModel* and respective *featureConfigurations* for games.

The *integration of represented game features* can be performed by *FeatureState*, *feature observer* and *FeatureBehavior* specializations. Using a *FeatureState*, different game features (but representing similar characteristics) can be used to model a common state. A good example is the *ElementState* class, which is a combination of *Element*, *Avatar* and *DecisionEntity* game features.

*FeatureState* instances can be monitored by *FeatureStateObserver* instances, which are triggered whenever a *FeatureState* update is applied. They verify collisions among *Element* instances or updates in current values of *Avatar Skills*, for example. When a *feature observer* is triggered, a *FeatureBehavior* instance is performed, providing support for game behaviors described by *game features*, such as *Action*, *Relationship*, *GameBehavior*, *DecisionBehavior* and *SceneBehavior*.

Finally, for the *implementation of game features*, it consists of preparing specific *Adapter* instances to reflect available *features* and *FeatureState* instances in the desired game engine. They are the main interface among game engines procedures and represented *game features* in order to execute the configured game.

Moreover, considering the FEnDiGa development steps, *features* and *states* can be configured by *Game Designers* and *Game Programmers*, *observers* and *behaviors* can be contextualized by *Game Programmers*, and *adapters* can be provided and used by *Game Engine Specialists* and *Game Programmers*, respectively. Fig. 3 illustrates this game development process, describing their game actors and important game development activities.

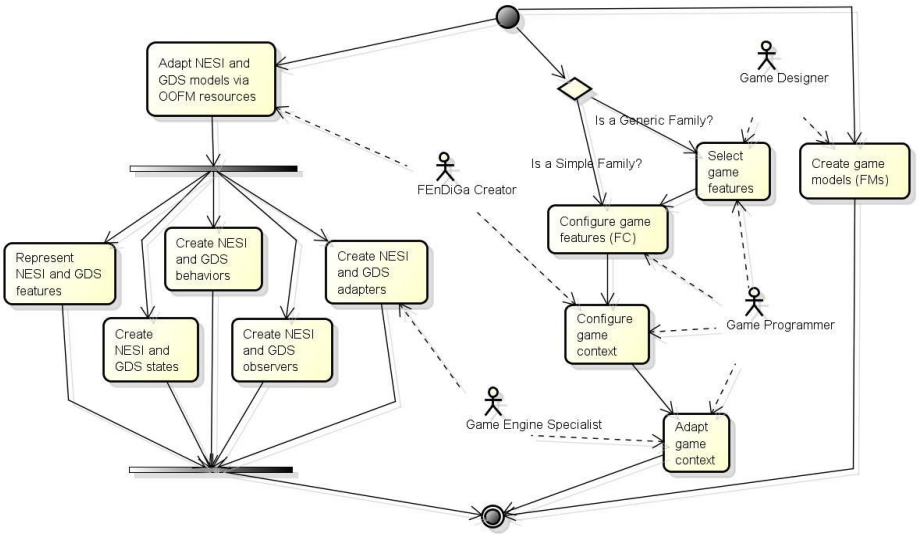


**Fig. 3.** The FEnDiGa development process.

## 4 Step 3: Configuring the *SimplifiedPacman* Game

With FEnDiGa structures ready to be instantiated by the FEnDiGa development process, the next step is the development of a final game (*SimplifiedPacman* in this case) by the configuration and interpretation of adapted *game features*.

Initially, identified *game features* based on NESI and GDS models are instantiated via FEnDiGa by the game developer. Next, several game *states*, such as *pacmanState* and p*illState*, are created to represent and integrate instantiated *features*. *Feature* updates and the notification of *FeatureState* updates are performed by *behavior* instances, which are defined by *ScriptBehavior* (such as *pacmanDieBehavior*) and *FeatureBehavior* specializations (*movePacmanBehavior*, for example). Also, for feature *observer* instances based on condition evaluation (c*ollisionSystemObserver*) and defined *ScriptObserver* (*lifeEndEvent*) results, they evaluate desired game s*tates* and indicate which *behaviors* must be performed, according to *observer* results. Finally, for the adaptation process, *Adapter* interfaces are implemented, such as *SimplifiedPacmanSpriteAdapter* and *SimplifiedPacmanJGObjectAdapter*, in order to adapt available engine resources (such as *Sprite* and *JGObject*) for defined FEnDiGa structures. They are used by engine procedures (such as *init*, *start*, *render*, and so on) in order to associate an engine behavior for each adapted *state* in a game.

As a result, by the proposed configuration of *features*, s*tates*, *behaviors*, *observers* and *adapters*, a complete structure for the *SimplifiedPacman* game, ready to be performed, is provided. Fig. 4 illustrates the *SimplifiedPacman* final architecture, adapted to three distinct game engines (GTGE, JGame and JME3) [7].

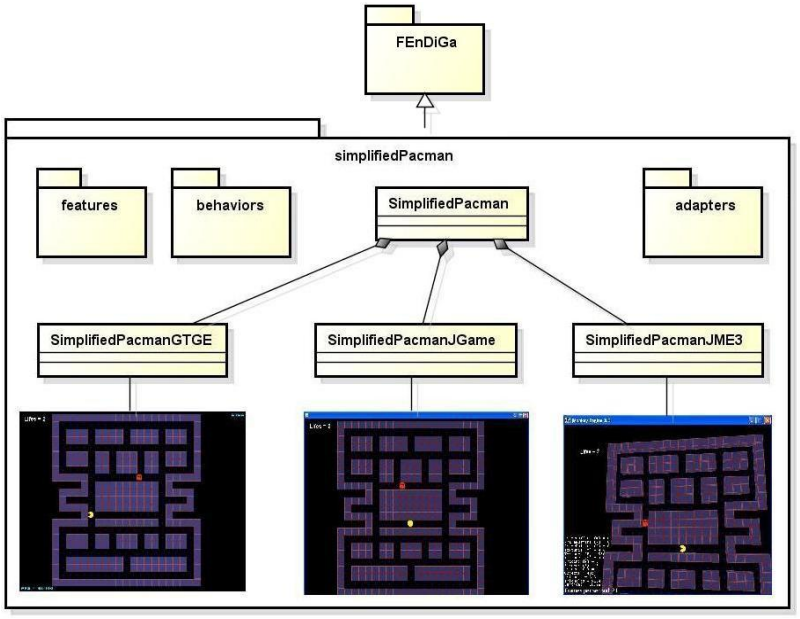


**Fig. 4.** The *SimplifiedPacman* game achitecture.

## 5 Conclusions and Future Work

This paper presented the application of the OOFM technique in the development of digital games represented by NESI and GDS feature models. For that, the FEnDiGa structure and development process were defined, providing a game development environment able to integrate conceptual and implementation aspects of games.

By the FEnDiGa usage, the game development work can be resumed to: instantiate representative *features* of a game; define *states*, *observers* and *behaviors* to represent a game context based on instantiated *features*; and define which *states* will be worked by *adapters* during the game execution in a final game engine.

Several types of combinations among defined FEnDiGa structures can also be performed, allowing as a result a great level of reuse for game categories represented by *features* (the game mixing). In addition, NESI and GDS specializations can be also supported, allowing the production of customized frameworks for games and game sub-domains (FEnDiGa-RPG) according to designer interests.

In other words, by the FEnDiGa usage, the game logic becomes *standardized* (based on NESI and GDS models), *portable* (G-Factor [4] in an independent way via *Adapters* usage) and *reusable* (distinct combination of *game features*) in a new software development approach.

For future work, some extra activities will be performed, such as the evaluation and simplification of NESI and GDS models, the development of FEnDiGa supporting tools, and the FEnDiGa usage in different game development approaches (engines, frameworks), categories (RPG, FPS, Adventure) and complexities (complete games instead of simplified versions, collaborative development, etc.).

## 6 References


1. Wolf, M.: The Medium of the Video Game. University of Texas Press, ISBN: 029279150X, (2002).
2. Sarinho, V. and Apolinário, A.: A Feature Model Proposal for Computer Games Design. In: Proceedings of the VII Brazilian Symposium on Computer Games and Digital Entertainment, Belo Horizonte, p. 54-63, (2008).
3. Sarinho, V. and Apolinário, A.: A Generative Programming Approach for Game Development. In: Proceedings of the VIII Brazilian Symposium on Computer Games and Digital Entertainment, Rio de Janeiro, p. 9-18, (2009).
4. Binsubaih, A. and Maddock, S.: Game Portability Using a Service-Oriented Approach. IJCGT, Volume 2008, Article ID 378485, 7 pages. Hindawi, (2008).
5. Sarinho, V. and Apolinario, A.: Combining feature modeling and Object Oriented concepts to manage the software variability. In: Proceedings of the IEEE IRI 2010, 4-6 Aug. 2010, pp: 344-349, (2010).
6. Krasner, E. and Pope, T.: A cookbook for using the MVC user interface paradigm in Smalltalk-80, J. Object Oriented Program, v. 1, p. 26-49, (1988).
7. List of game engines. From Wikipedia, the free encyclopedia. Available from: `http://en.wikipedia.org/wiki/List_of_game_engines`.