Analyzing Interaction Flow Modeling Language in Web Development Lifecycle

Karzan Wakil[1,2,3] and Dayang N.A. Jawawi[1]
[1]Universiti Technologi Malaysia Skudai 81310 Johor Malaysia
[2]Sualimani Polytechnic University-Iraq
[3]University of Human Development-Iraq

***Abstract*—Two years ago, the Object Management Group (OMG) adopted a new standard method named Interaction Flow Modeling Language (IFML) for web engineering domain. IFML is designed to express the content, user interaction, and control behavior of the front end of applications. There are number lacks in web engineering methods, because each of them is defined to particular specifications, one of which is the open issue of supporting the whole lifecycle in process development. In this paper, we analyze IFML models in the process development lifecycle to show capability of the method used in the process development. We then make a comparison between IFML and other methods in lifecycle phases. Finally, we add IFML to the web engineering lifecycle's map. It is anticipated that the result of this paper will be to become a guide for developers for using IFML in the development of new applications.**

Keywords*—Interaction Flow Modeling Language; IFML; Web Engineering Methods; Web Development Lifecycle

I. Introduction

Model Driven Web Engineering (MDWE) methods such as WebML [1], UWE [2] or OOHDM [3] have become mature solutions for developing Web Applications. These methods utilize Model Driven Development (MDD) perceptions to acquire advanced web applications ideas into models; hence utilizing such models obtain application automatically. The classic MDWE development process consists of three phases [4]: (1) building a domain model, (2) defining a hypertext model and (3) defining the application's look and feel. The process outcome is a set of models with the capacity to create the ultimate web application via code generation.

As evident in [5], several methods created for the plan of hypermedia systems only partially cover the hypermedia system's lifecycle besides being highly centered on the configuration of these systems, as evident from Fig. 1. Just recently, in 2014, the OMG was able to adopt a novel standard method identified IFML for web domain by Macro Brambilla [6].

There exist several gaps within the field of web engineering methods with one of them being no single method that considers the entire establishment lifecycle thoroughly, with each method having its particular strengths [7], as evident from Fig. 1. As seen in [5], several methods that are created for the design of hypermedia systems only partially cover the hypermedia systems' lifecycle and are highly centered on the configuration of such systems. The web engineering community, several research groups are geared towards sustainable solutions to such variations, with some being solved by merging two methods like RUX-Method and UWE method to support Rich Internet Applications (RIA) [8], while the solution of others was obtained through enhanced methods like UWE metamodels in establishing novel modules of websites [9] although could never have all the gaps completely solved. Subsequent to numerous perfections, Marco Bramilla recommends IFML upon a ten years experience WebRatio and WebML [6], since the preceding researchers had confirmed WebML being among the most accurate methods within web engineering approaches [10-11].

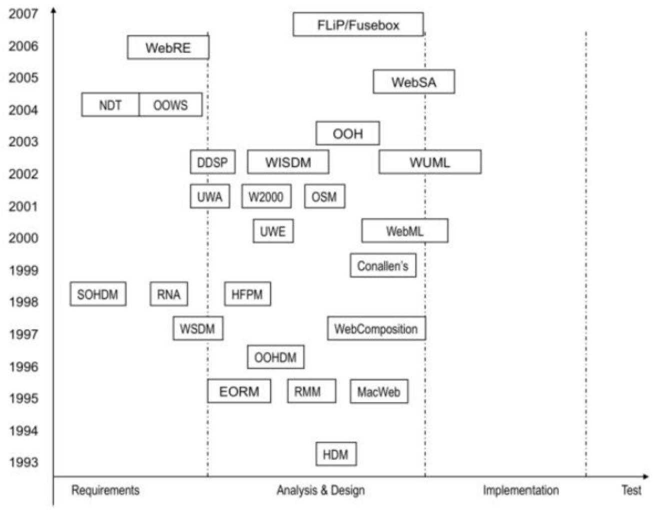

Fig. 1. The evolution and coverage the best-known web development [7]

MDWE [12] offer the tools and methodologies engaged in the structuring and development of various types of web application. The researchers cover various issues by engaging diverse models (presentation, navigation, and data among others), with support from model compilers capable of automatically generating several of the logic codes and Web Pages of the application. The advantages of engaging MDWE are evident from diverse perceptions like software quality, team output or adjustment to improving technologies [13-14]. Of these diverse MDWE methods, it is worth describing the IFML [15], an object management group condition for the establishment of data-intensive utilization hence becoming a key reference within the industry growth [16-17]. Its efficient creation tool, WebRatio, permits the editing and validation of IFML models besides facilitating the development of the final application code for a given technical exploitation platform,

Digital Object Identifier (DOI) : 10.14569/IJACSA.2017.080137

minimizing the time-to-market as well as the development effort for such uses.

IFML was designed after ten years' experience with one of the best methods and managed to solve some gaps in the existing methods. However, our main contribution is finding IFML location among web engineering methods in process development web application phases. In this paper, we will analyze IFML in respect to process development web applications. This will involve demonstrating the capability of IFML to support whole phases of web engineering in a lifecycle to determine IFML's location in the lifecycle map. In future research, we will make a comparison between IFML and the other web engineering methods in a lifecycle.

The paper is organized as follows: Section 2 explains the background work undertaken for the Web Engineering lifecycle and IFML. In section 3 we conduct a web engineering methods analysis to support the lifecycle. In Section 4, we analyze the ability of IFML to support the lifecycle. Section 5 describes the addition of IFML to the lifecycle map and makes a comparison between IFML with other web engineering methods to support lifecycle phases. In section 6 we design case study by using IFML to prove our result in previous sections. In the last section, we present some concluding remarks and suggestions for future research.

II. Related Work

In this section we discuss about web engineering phases in lifecycle and effective the methods in the process development web applications. Also we discuss about the previous work that done by IFML. Optimization of development effort in the Web Engineering domain has been addressed by several works. In [18] the researchers centered on the examination of the effect of engaging a MDWE method concerning customary web developments. The researchers achieved a significant productivity benefit by engaging their model driven approach. Moreover in [19] a detail literature review about MDWE explains that one of the column in this area is process development and agility in lifecycle.

For quite some time, there has been an escalating growth in the various proposed methods, approaches or methods within professional and academic literature as an attempt of handling some particular features of Web development. Of the most significant challenges facing Web-based system design and development include intricate interfaces, navigation, complex maintenance, security concerns, as well as indefinite remote users, although they came up with solutions to problems they equally offer some limitations, with scarcity in cover lifestyle being among them [20-22]. In their study, Lang and Fitzgerald [23] present a comprehensive list of overfly methods and approaches for Web/hypermedia systems development. A depiction and comparative assessment of the renowned Web development methods can be achieved in [24].

An important observation in [20] as noticed from Fig.1 is the varied coverage by methods of the development phases. In the Fig.1, each approach is located in the phase where its main focus lies. Thus, although the UWA Project [25] or WebML [26] give some consideration to requirements definition and implementation, they mainly emphasize the analysis and design phase. As can be seen, the majority of Web development methods are concentrated within the analysis and design phase, with noticeably less focus on the other phases of the life cycle.

We come back to IFML; it has good features for developing web applications, especially rich in interface and can easily support RIA. Macro Brambilla and Piero Fraternal, 2014 [15] explains most concepts IFML within a book. The book explained metamodels, process development web and mobile applications, capability extensions, and so on. Another work is object-oriented analysis and design for developing information systems by using IFML by [27]. In [28-29] used IFML for developing mobile application by WebRatio. But after inventing IFML no work exist in the lifecycle process development; we need to explain this method and present capability in the process development lifecycle.

III. MDWE Life Cycle

In this section we explain current web engineering methods location in lifecycle, and we attempted to present capability the methods in the lifecycle in process development web applications.

In their study [30] offered along-drawn-out lifecycle procedural model for the development of web-based applications within small and medium enterprises. The model comprised of three processes sets, including requirement-development and evolution processes. Predominantly, the significance of post-delivery advancement process to small and medium enterprises is the development and maintenance of quality web applications by engaging the scarce resources and time [30]. Other researchers employed what is commonly identified as mockups (user interface prototypes) as an approach of commencing the modeling process within the framework of an integrated agile MDWE method [31]. As a measure of aiding this method, the present study incorporated a frivolous metamodel that encourages modeling aspects over mockups, creating end users interface as well as creating MDWE models.

Furthermore this study considered a statistical assessment of the two methods (traditional modeling versus mockup based modeling) [31]. In [7], a very excellent combination model has been offered with the objective of covering lifecycle, while suggesting three web engineering approaches: UWE, NDT, and WebML to handle lifecycle as expressed in Fig.2, regardless of this idea being excellent, it is equally intricate in the implementation phase since it requires novel transformation model, besides lack of tool supporting the implementation of this concept.

Model Driven Architecture (MDA) does not only entail modeling, it is unfeasible to anticipate 100% code generation for all computing setbacks, while presently no vendor can practically give a absolute MDA solution. Therefore, increased expectations from MDA would result in a probable failure. Simply, MDA facilitates a method of system design and development approach, engaging several standard tools and notations to acquire interoperability plus reuse among vendors, as well as platform independence. In order to achieve the complete MDA benefits, institutions should not simply incorporate some modeling process within the creation methods; but equally promote the complete software lifecycle development process, from requirements management and analysis, to configuration, creation, execution, deployment, as well as maintenance. Else the complete MDA benefits will be lost [32].

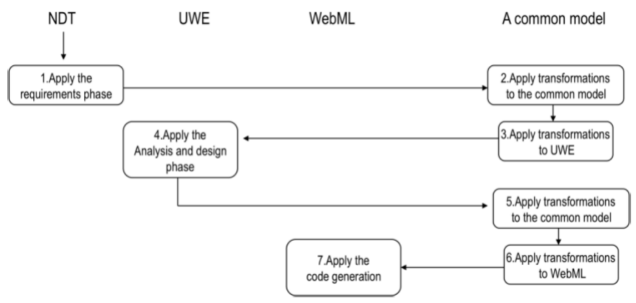
NDT
UWE
WebML
A common model
1.Apply the requirements phase
2.Apply transformations to the common model
3.Apply transformations to UWE
4.Apply the Analysis and design phase
5.Apply transformations to the common model
6.Apply transformations to WebML
7.Apply the code generation

Fig. 2. Use common metamodels to make approaches compatible [7]

The RIAs' development process is founded on the MDA idea. This means, it decouples the system notion by coming up with distinct system models at diverse abstraction levels. subsequently, model transformation is considered during the development lifecycle with standard patterns or rules of transformation. Not only do the models assist in describing the system idea at diverse stages, they too play a role in automated code generation. For the purposes of conforming to MDA, the models we utilized are categorized into three: computation independent, platform specific, and platform specific. There exist tools for developing each model. Fig.1 exemplifies both the MDA compliant process and the system development step. Just as evident form Fig.1, there exists no any method that addresses the whole of lifecycle development in details while each method holds its distinct benefits [33].

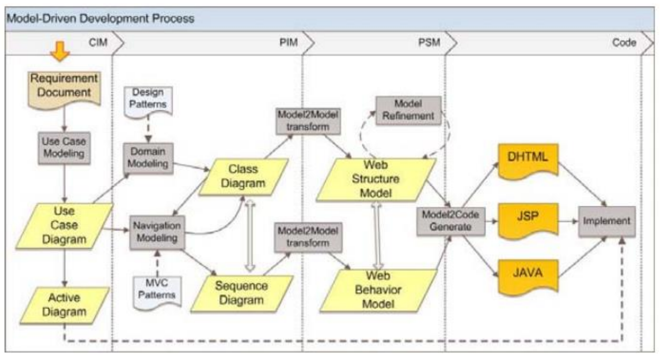

Fig. 3. Model-Driven development process overview [32]

Additional studies by Domingues [34] and Koch [35] have been exemplified in Table 1. The table puts into consideration the phases of the development methods suggested for Pressman [36], and are inclusive of "(i) formulation; (ii) planning; (iii) analysis; (iv) design (architectural, navigational and interface); (v) pages generation; (vi) testing; and (vii) customer assessment." The following notation is used in this table: C, if the method fully fulfils the development stage; P, if the the stage is partially fulfilled; and blank when the method does not deal with the activity.

TABLE I. DEVELOPMENT METHODS PROCESS STAGE [37]

	Formulation	Planning	Analysis	Design			Generation	Testing	Evaluation
				Arch.	*Navig.*	*Interf.*			
HDM				C	C				
RMM				C	C	C	P		
OOHDM			P	C	C	C	P		
HMBS			C	C	C	C	C	P	
UWE			C	C	C	C			
WebML		C	C	C	C	C	C	P	
OO-H			C	C	C	C	P		
W2000			C	C	C	C			
WAE		C	C	C	C	C	P	P	
SWM	P	P	P	P	P	P	P	P	
OOWS			C	C	C	C	C		

Upon offering a quick evaluation of the MDWE lifecycle as evident above, it is worthy concluding the web engineering methods' certain strengths in lifecycle phases and lost assessment phase from all methods. Therefore, the proposed model merges two or three methods to have these setbacks handled. The subsequent chapter analyzes IFML in comparison to other methods so as to exemplify its capacity against other methods.

IV. ANALYZING IFML IN LIFECYCLE

In this section we study centers on the models and components of IFML associated with the lifecycle development process.

A. General Overview

IFML [15] has been confirmed to enhance the platform-independent delineation of Graphical User Interfaces (GUI) among applications accessed or installed on systems like laptops, desktop computers, tablets, mobile phones and PDAs. The key focus is on the application's behavior and structure as observed by the end user. The language used in modeling equally integrates references to the business logic and data influencing the experience of the user. This is attained accordingly by having the domain model objects referenced so as to offer the content presented in the interface as well as the measures capable of being triggered through interface interaction.

B. IFML Artifacts

The IFML language is specified within an official, human-readable OMG specification document, which in turn is accompanied by some technical artifacts:

- The IFML metamodel, specifying the structure and relations between the IFML elements;
- The IFML is UML profile, defining a UML-based syntax for expressing IFML models, through an extension of the concepts of the class, state machine, and composite structure diagrams;
- The IFML visual syntax, offering a graphic notation for expressing IFML models in a concise and intuitive way; and;

- The IFML model serialization and exchange format, for tool portability.

Altogether, these artifacts compose the IFML language specification. Each of them is specified according to the OMG standards:

- The metamodel is defined through the MOF metamodeling language (an equivalent ECORE definition is available too).
- The UML profile is defined according to UML 2.4 profiling rules.
- The visual syntax is defined through Diagram Definition (DD) and Diagram Interchange (DI) OMG standards.
- The model serialization and exchange format is defined based on XMI.

C. Metamodels

Definition of IFML metamodel is done respective of the best methods of language description, incorporating abstraction, modularization, recycle as well as extensibility. There are three packages categorizing the metamodel: Extension package, the core package, as well as data-type package. The core package entails the ideas creating the language interaction infrastructure in terms of interaction flows, Flow Elements as well as the limits. Central package ideas are broadened by actual ideas in the extension package to cover highly precise behaviors. The Data Types package entails the custom data types delineated by IFML. The basic UML metamodel data types are reused by the IFML metamodel, focuses several UML meta classes as the FML meta classes basis, and talking the assumption that a domain model is illustrated with a UML class diagram or an identical representation.

IFML model is considered as the top-level component of the other model components. It entails a domain model, an Interaction Flow Model, as well as View Points. Interaction Flow Model offers the application view of the user, by quoting to the Interaction Flow Model Elements sets, jointly defining a completely functional portion of the system. As an abstract category, Named Element focuses on the Element class (the model's broad class) exemplifying the named elements. For any component, it is easy to specify comments and constraints. Interaction Flow Model Element is considered an abstract category that levels the aspects of an IFM. Per se, its use is not directly associated with the IFML diagrams; rather, it is defined by more particular notions (such as Interaction Flow Element, Interaction Flow). Sequentially, these sub-concepts are abstract, hence the need to be aptly specialized.

D. IFML Development Process

The development of applications defined by interactivity is normally handled with agile techniques, which navigate diverse phases of "problem discovery" / "design refinement" / "implementation." The iteration of the development method derives a partial version or a prototype of the system. Such an augmentable lifecycle is predominantly suitable for contemporary web and mobile uses, with the need of being installed swiftly and alter frequently throughout their lifetime to adjust to user prerequisites. Fig. 4 offers a probable structural development process hence positioning IFML within the activity flow:

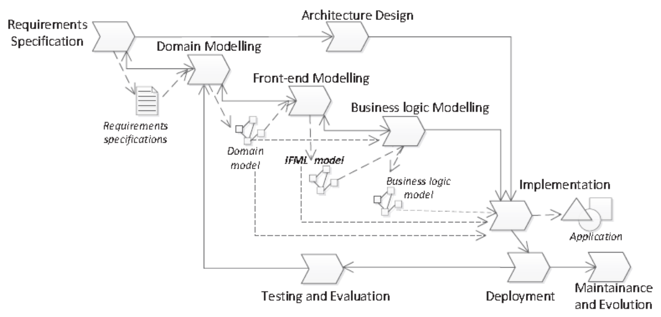

Fig. 4. The role of IFML in the development process of an interactive application

1) Requirements specification: gathers and formalizes the data concerning the application domain as well as the anticipated functions. The input entails a set of business needs promoting the application development as well as the accessible data on the organizational, technical and managerial settings. The result is a practical specifications file entailing:

- The recognition of the user functions plus of the use cases linked with each function;
- A data dictionary of the needed domain notions as well as of their semantic associations; and
- The workflow represented in every application case, showing the interaction of the key actors (the application, the user and perhaps external services) during the implementation of the use case.

Furthermore, nonfunctional needs should equally be delineated, such as scalability, performance, accessibility, maintainability and security. Upon directing the application to the ordinary people, the prerequisites about the feel and look as well as the interfaces' usability take into assumption special prominence among the nonfunctional requirements. User-focused configuration practices that depend on the development of ideal mockups of the practicality operation can be utilized. Such mockups can be applied for the primary validation of the interface notions and later act as the setting for establishing more comprehensive and technical delineations for the front-end modeling stage.

2) Domain modeling: systematizes the key information objects established during conditions delineation into a broad and articulate setting model. Domain modeling delineates the key data sets established during conditions requirement into a domain model, normally a (characteristically visual) depiction of the necessary objects, their qualities and relationships.

3) Front-end modeling: plots the data manipulation and information conveyance functionality proposed by the requirements application conditions into front-end model. The operation of front-end modeling is at the conceptual angle, with IFML coming into play. The developer is at the liberty of utilizing IFML in the specification of front-end organization in a single or several top-level view containers, the internal formation of every view container regarding sub-containers, the constituents forming each view container's content, the events depicted by the components and vie containers, as well as how such events set off business events and revise the interface.

4) Business logic modeling: delineates the business objects and the techniques needed to sustain the established use cases. UML dynamic and static figures are usually used in highlighting the objects interface as well as messages flow. Process-adjusted details (like UML functionality and sequence charts, BPMN process models, and BPEL service orchestrations) offer an efficient method of signifying the workflow across services and objects. The services highlighted in the business logic plan can be oriented in the front-end model to signify the operations to be set off through interface interaction. Being interdependent in nature, front-end, data, and business-logic structure events are performed in an iterative manner. The preference category of Fig.4 is simply indicative. Within some companies, the responsibility could commence at the structure of the front-end while the actions and data objects could be established at a later phase though analysis of the published information as well as the requested operations towards sustaining the interactions.

Architectural structure is the technique of delineating the network, hardware as well as the software elements that compose the architecture whereby the application offers its services to the users. The objective of the architectural structure is to establish the mixture of these components that adequately achieves the application needs as regards to scalability, efficiency, accessibility, security, and all together adhering to the economic and technical project limitations.

5) Implementation: entails the approach of creating the software modules that convert the business logic, data as well as interface design into an application functioning on the opted design. Implementation of data situates the domain model onto a single or several data sources by merging the conceptual-level aspects with the formations of logical data (such as relationships and aspects to relational tables). The execution of business logic generates the software components required to sustain the identified use cases. The execution of individual entities may gain from the adoption of software designs, which systematize the manner in which fine-grain elements are devised and merged into a wider and highly reusable operational units and equally provide for nonfunctional needs like scalability, accessibility, security and competence. Translation of abstract-level View Components and View Containers into the opposite aspects within the considered execution plan is done courtesy of interface accomplishment. It is possible for the View Containers and business objects to interoperate either in the server or client layer.

6) Testing and evaluation: confirms the consistency of the installed application concerning the nonfunctional and functional requirements. The key important aspects for interactive model testing include:

a) Functional trialing: verification of the application behavior regarding the functional requirements. Functional testing is disintegrated into classical events of module examination, system testing and integration testing.

b) Usability Assessment: the nonfunctional prerequisites of accessibility, communication efficiency, and observance to merged usability values are confirmed against the generated front end.

c) Performance assessment: the application's response time and throughput ought to be examined in peak and average workload provisions. There is the need to monitor and examine the insufficient service levels, the usability design, so as to establish and get rid of bottlenecks.

V. Result of IFML Analyses and Adding to Lifecycle Map

After conducting a detailed review of IFML in process development and analyzing existing references, we were able to acquire a full image concerning the IFML lifecycle. Our analysis centered on IFML's need for requirements, but not necessarily supporting it. It is the UML profile that has helped in the design and analysis phase. Also, with the support of WebRatio, visual syntax has been defined through: DD and DI and OMG standards; model serialization; and exchange format which is defined based on XMI. These factors have all helped to fully support the implementation stage. Finally, we can add to the web engineering phases the fact that location between analysis/design and some implementation is the same WebML as shown in Fig.5 because Webratio allowed the implementation after design, but with rich interface and best practice.

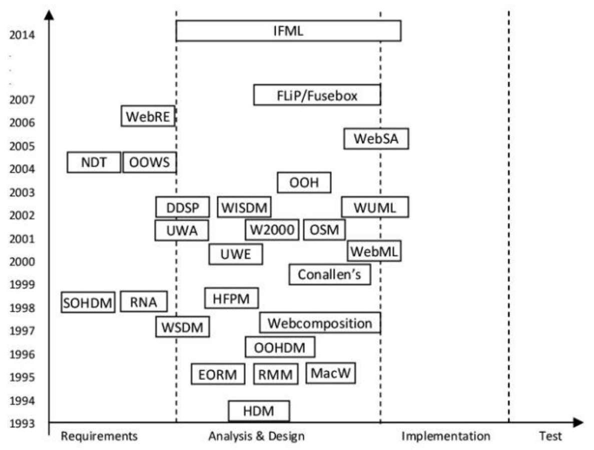

Fig. 5. The evolution and coverage the best-known web development after adding IFML

IFML location between analysis/design and implementation phases, however starting some gathering requirements and test usability but not fully supported.

In order to show the capability of IFML for process development, we need a comparison between IFML and the existing methods. For this case, we updated a comparison proposed by [37] after adding IFML as shown in Table 2. IFML cannot support formulation, planning, and, but can support analysis/design and code generation. This is one of the new terms in IFML that can evaluate the project, as shown in Fig.4.

TABLE II. COMPARISON OF IFML WITH OTHER METHODS IN THE DEVELOPMENT PROCESS STAGE

	Formulation	Planning	Analysis	Design			Generation	Testing	Evaluation
				Arch.	*Navig.*	*Interf.*			
IFML			C	C	C	C	P	P	P
HDM				C	C				
RMM				C	C	C	P		
OOHDM			P	C	C	C	P		
HMBS			C	C	C	C	C	P	
UWE			C	C	C	C			
WebML		C	C	C	C	C	C	P	
OO-H			C	C	C	C	P		
W2000			C	C	C	C			
WAE		C	C	C	C	C	P	P	
SWM	P	P	P	P	P	P	P	P	
OOWS			C	C	C	C	C		

VI. DESIGN CASE STUDY

For showing the capability IFML method to design web applications, we highlighted movie shop in Amazon website as case study as shown in Fig.6.

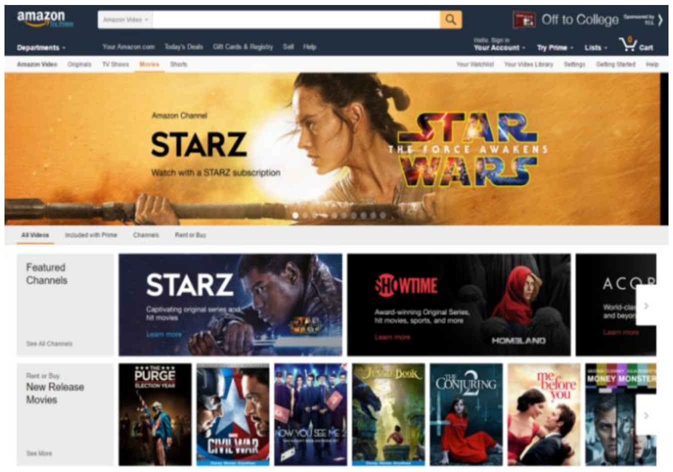

Fig. 6. Amazon Movie Homepage

A. Content Model

In this example, a Customer is assigned a Credit Card that at the beginning is empty. As the user browses through the page and gets information about the Movies available, adds products to the credit cart. The list of Moves selected at the moment by the user, can be consulted at any time, offering the option of pay the current order, empty the car or continue browsing in order to add more Movies, Fig.7 shows Content Model for Amazon Movie by IFML method.

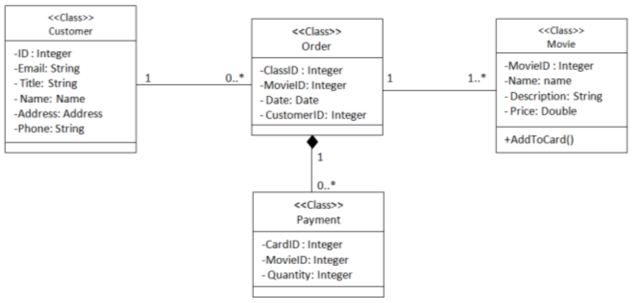

Fig. 7. Content Model for Amazon Movie

B. Process Model

When the customer enters into the website, starts exploring the available Movies. Once he finds a movie of interest, selects it, and the item goes to the credit cart. The user can either keep exploring products in order to add more items to his order, or continue to manage the credit cart by deleting all the Movies, or updating quantities of the selected ones. Once the user is ready to proceed with the payment, performs the checkout. In order to authorize the payment, it's necessary to send the customer information to the bank entity, and wait for the confirmation. This procedure is illustrated in the Fig.8.

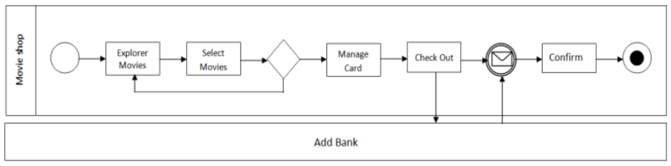

Fig. 8. Process Model of the Amazon Movie]

Fig.9 shows the home page of the Amazon Movie. In this section, the user can select one of the Movies.

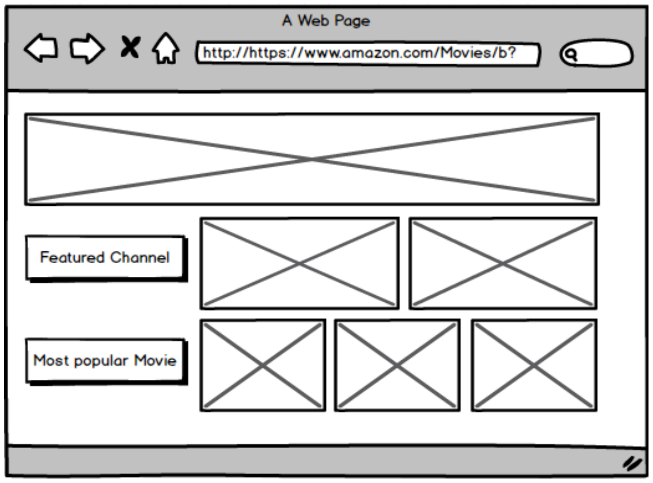

Fig. 9. Amazon Movie Homepage

After selecting a Movie the user can full description of the movie, directly you can buy the movie by adding card, as shown in Fig.10.

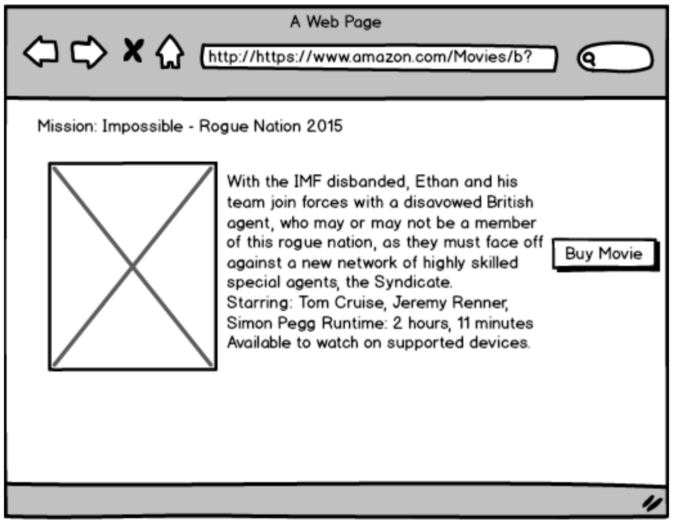
A Web Page
http://https://www.amazon.com/Movies/b?
Mission: Impossible - Rogue Nation 2015
With the IMF disbanded, Ethan and his team join forces with a disavowed British agent, who may or may not be a member of this rogue nation, as they must face off against a new network of highly skilled special agents, the Syndicate.
Starring: Tom Cruise, Jeremy Renner, Simon Pegg Runtime: 2 hours, 11 minutes
Available to watch on supported devices.
Buy Movie

Fig. 10. Details of the selecting Movie

The procedure described in the Fig.9 and Fig.10 is represented in IFML as shown in the Fig.11. Once the user selects a category from MovieCategory a navigation event is produced, and as a result, the details of the Movie showed in MovieDetail.

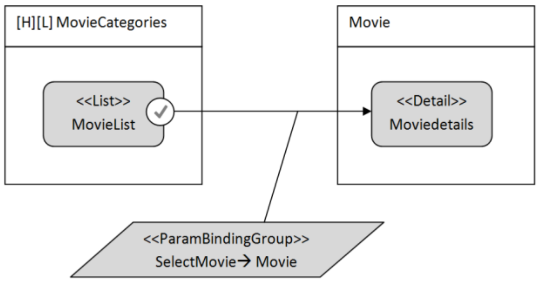

Fig. 11. IFML model corresponding to the exploration of Movie

Fig.12 shows the model fragment that adds a product to the cart, once the user press add button, a modal window appears asking for the quantity of items desired. This value, along with the SelectedMovie are submitted as parameters and represent the input of the add to cart action triggered. Once the action is performed, a confirmation window is displayed.

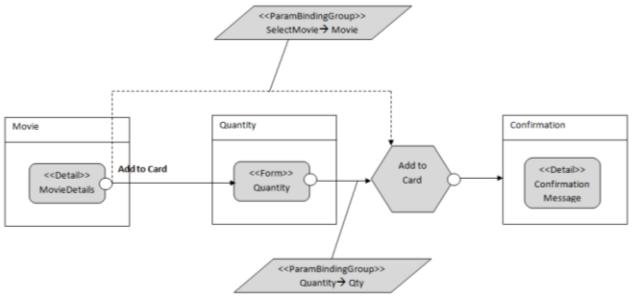

Fig. 12. IFML model corresponding to the add to cart event

When the user chooses the Checkout option, the container Customer Information is displayed. The user must provide his personal information by filling out the form within this container. After the user submits his personal information, the container Payment Information is displayed. In this container the user must provide his bank account details for execute the payment process see Fig.13.

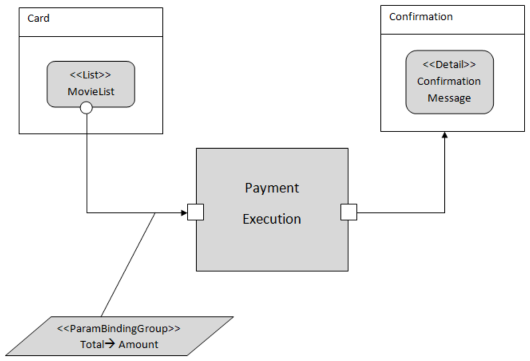

Fig. 13. IFML Module Representation of the Checkout Event

To increase reusability and modularization in the models, designers may decide to cluster homogeneous parts of the model into Modules. For instance, the part of the model that deals with the payment management can be packaged into a specific module. This would simplify the model of the application, as shown in Fig.14.

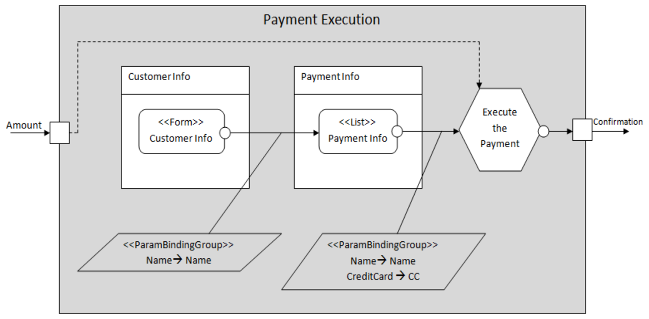

Fig. 14. Inner Process of the Module Payment Execution

After design our case study by IFML model, and showing important interaction in the process buying a move, we can conclude IFML method can fully support analyze/design phase in the web engineering lifecycle. However allowed to generate code generation as semantic implementation, but cannot fully support other phases.

VII. Conclusion and Future Work

In this paper, we analyzed the actions of IFML in the process development life cycle. In addition, we added IFML to the lifecycle map and made a comparison between IFML and the existing methods in order to develop web application process development phases. Results showed that IFML is a good method with best practice but cannot fully support the web development lifecycle. IFML is composed of a UML profile and support rich interface. That is an important point by which to improve IFML to support the lifecycle through combination with other web engineering methods or adding agile methods to improve process development.

We recommend for researcher to extend this research through implement IFML in the different case study, also researcher can combine IFML with other methods to solve weaknesses method. Moreover we recommend making usability and reliability evaluation to present quality this method.

ACKNOWLEDGMENT

We thank Professor Dr. Gustavo Rossi, for his review and useful comments on this article.

REFERENCES

[1] S. Ceri, P. Fraternali, and M. Matera, "Conceptual modeling of data-intensive Web applications," Internet Computing, IEEE, vol. 6, pp. 20-30, 2002.

[2] N. Koch, A. Knapp, G. Zhang, and H. Baumeister, "UML-based web engineering," in Web Engineering: Modelling and Implementing Web Applications, ed: Springer, 2008, pp. 157-191.

[3] G. Rossi and D. Schwabe, "Modeling and implementing web applications with OOHDM," in Web engineering: Modelling and implementing Web applications, ed: Springer, 2008, pp. 109-155.

[4] M. Wimmer, A. Schauerhuber, W. Schwinger, and H. Kargl, "On the integration of web modeling languages: Preliminary results and future challenges," in Workshop on Model-driven Web Engineering (MDWE), held in conjunction with ICWE, Como, Italy, 2007.

[5] N. P. de Koch, "Software engineering for adaptive hypermedia systems," PhD Thesis, Verlag Uni-Druck, Munich, 2001.

[6] IFML. (2016, Interaction Flow Modeling Language. Available: http://www.ifml.org/

[7] G. Aragón, M.-J. Escalona, M. Lang, and J. R. Hilera, "An analysis of model-driven web engineering methodologies," International Journal of Innovative Computing, Information and Control, vol. 9, 2013.

[8] J. C. Preciado, M. Linaje, R. Morales-Chaparro, F. Sanchez-Figueroa, G. Zhang, C. Kroiß, and N. Koch, "Designing rich internet applications combining uwe and rux-method," in Web Engineering, 2008. ICWE'08. Eighth International Conference on, 2008, pp. 148-154.

[9] K. Wakil, A. Safi, and D. Jawawi, "Enhancement of UWE navigation model: Homepage development case study," International Journal of Software Engineering & Its Applications, vol. 8, 2014.

[10] K. Wakil and D. N. Jawawi, "Metamodels Evaluation Of Web Engineering Methodologies To Develop Web Applications," International Journal of Software Engineering & Applications, vol. 5, p. 47, 2014.

[11] K. Wakil, D. N. Jawawi, and A. Safi, "A Comparison of Navigation Model between UWE and WebML: Homepage Development Case Study," International Journal of Information and Education Technology, vol. 5, p. 650, 2015.

[12] N. Koch, S. Meliá-Beigbeder, N. Moreno-Vergara, V. Pelechano-Ferragud, F. Sánchez-Figueroa, and J. Vara-Mesa, "Model-driven web engineering," Upgrade-Novática Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS) IX, vol. 2, pp. 40-45, 2008.

[13] G. Rossi, Ó. Pastor, D. Schwabe, and L. Olsina, Web engineering: modelling and implementing web applications: Springer Science & Business Media, 2007.

[14] P. Vuorimaa, M. Laine, E. Litvinova, and D. Shestakov, "Leveraging declarative languages in web application development," World Wide Web, vol. 19, pp. 519-543, 2016.

[15] M. Brambilla and P. Fraternali, Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML: Morgan Kaufmann, 2014.

[16] S. Casteleyn, I. Garrig'os, and J.-N. Maz'on, "Ten years of Rich Internet Applications: a systematic mapping study, and beyond," ACM Transactions on the Web (TWEB), vol. 8, p. 18, 2014.

[17] G. Toffetti, S. Comai, J. C. Preciado, and M. Linaje, "STATE-OF-THE-ART AD TRE DS I THE SYSTEMATIC DEVELOPME T OF RICH I TER ET APPLICATIO S," Journal of Web Engineering, vol. 10, pp. 070-086, 2011.

[18] Fatolahi and S. S. Somé, "Assessing a Model-Driven Web-Application Engineering Approach," Journal of Software Engineering and Applications, vol. 7, p. 360, 2014.

[19] K. Wakil and D. N. Jawawi, "Model driven web engineering: A systematic mapping study," e-Informatica Software Engineering Journal, vol. 9, pp. 107--142, 2015.

[20] J. A. Hincapié Londoño and J. F. Duitama, "Model-driven web engineering methods: a literature review," Revista Facultad de Ingeniería Universidad de Antioquia, pp. 69-81, 2012.

[21] M. Lang, "A critical review of challenges in hypermedia systems development," in Information Systems Development, ed: Springer, 2005, pp. 277-288.

[22] G. Aragón, M.-J. Escalona, M. Lang, and J. R. Hilera, "An analysis of model-driven web engineering methodologies," 2013.

[23] M. Lang and B. Fitzgerald, "New branches, old roots: a study of methods and techniques in web/hypermedia systems design," 2006.

[24] M. Escalona, J. Torres, M. Mejías, J. Gutiérrez, and D. Villadiego, "The treatment of navigation in web engineering," Advances in Engineering Software, vol. 38, pp. 267-282, 2007.

[25] U. Consortium, "Requirements Elicitation: Model, Notation, and Tool Architecture," ed: Ubiquitous Web Applications Consortium (Deliverable D6), 2001.

[26] S. Ceri, P. Fraternali, and A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites," Computer Networks, vol. 33, pp. 137-157, 2000.

[27] R. S. Wazlawick, Object-oriented analysis and design for information systems: Modeling with UML, OCL, and IFML: Elsevier, 2014.

[28] R. Acerbis, A. Bongio, M. Brambilla, and S. Butti, "Model-Driven Development Based on OMG's IFML with WebRatio Web and Mobile Platform," in International Conference on Web Engineering, 2015, pp. 605-608.

[29] M. Brambilla, A. Mauri, and E. Umuhoza, "Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end," in International Conference on Mobile Web and Information Systems, 2014, pp. 176-191.

[30] W. Huang, R. Li, C. Maple, H.-J. Yang, D. Foskett, and V. Cleaver, "A novel lifecycle model for Web-based application development in small and medium enterprises," International Journal of Automation and Computing, vol. 7, pp. 389-398, 2010.

[31] J. M. Rivero, J. Grigera, G. Rossi, E. R. Luna, F. Montero, and M. Gaedke, "Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering," Information and Software Technology, vol. 56, pp. 670-687, 2014.

[32] N. Moreno, J. R. Romero, and A. Vallecillo, "An overview of model-driven web engineering and the mda," in Web Engineering: Modelling and Implementing Web Applications, ed: Springer, 2008, pp. 353-382.

[33] Y.-C. Huang, C.-C. Wu, and C.-P. Chu, "A new approach for web engineering based on model driven architecture," in International Conference on Management Learning and Business Technology Education, 2011.

[34] Domingues, "Aplicaçoes Web: Definiçao e Análise de Recursos de Teste e Validaçao," Tese de Doutoramento, ICMC/USP, Sao Carlos/SP-Brasil, em andamento, 2005.

[35] N. P. de Koch, "Software Engineering for Adaptive Hypermedia Systems-Reference Model, Modeling Techniques and Development Process," 2001.

[36] D.-C. Opera, M. S.-J. Import, A. B. Girls, and P. P. T. Set, "Software engineering: a practitioner's approach," 2005.

[37] Andr´e Lu´ıs dos Santos Domingues1, Sandro Lopes Bianchini1, Reginaldo R´e1, and and F. C. Ferrari1, "A Comparison Study of Web Development Methods," in Clei'2008 – XXXIV Conferencia Latinoamericana de Inform´atica, 2008