Mediterranean Smart Cities Conference (MSCC 2024)
2-4 May, 2024, Martil, Morocco

A Sketch of a MDA Approach to Accelerate the Development of User Interfaces in Tourism Web Applications

Lahbib Naimi
GL-ISI Team,
Faculty of Sciences and Techniques of Errachidia, UMI-Meknes, Morocco
l.naimi@edu.umi.ac.ma

El Mahi Bouziane
GL-ISI Team,
Faculty of Sciences and Techniques of Errachidia, UMI-Meknes, Morocco
bouzianeelmahi@gmail.com

Abdeslam Jakimi
GL-ISI Team,
Faculty of Sciences and Techniques of Errachidia, UMI-Meknes, Morocco
ajakimi@yahoo.fr

Lahcen El Bermi
GL-ISI Team,
Faculty of Sciences and Techniques of Errachidia, UMI-Meknes, Morocco
elbermi.lahcen@gmail.com

Abstract—**The rapid growth of the smart tourism sector necessitates the development of efficient and reliable web applications tailored to the demands of modern travelers. However, developing these applications is complex and requires expertise across various technologies, frameworks, and design considerations. This paper introduces an approach that harnesses the power of Model-Driven Architecture (MDA) to accelerate web application development. By generating code from high-level models, we bridge the gap between design and implementation. This approach ensures consistency, reduces manual coding efforts, and enhances maintainability. The heart of the approach lies in creating comprehensive metamodels that captures essential web elements, including navigation between pages, using Garrett's visual vocabulary. By embracing MDA principles, we empower developers to create smart tourism web applications more efficiently, adapt to changing requirements, and deliver exceptional user experiences.**

***Keywords*— Model-driven architecture; Smart tourism; Code generation; Garrett's visual vocabulary; Web developement; User interface design**

I. INTRODUCTION

In recent years, the smart tourism industry has experienced exponential growth, driven by the increasing demand for technologically advanced solutions that cater to the evolving preferences of modern travelers [1]. In this sense, the smart tourism sector is transforming the way travelers experience destinations [2]. From personalized recommendations [3] to interactive maps and augmented reality experiences, smart tourism web applications play a crucial role in engaging users and fostering memorable journeys. However, developing these applications presents a complex challenge. The intricate interplay of diverse technologies, design considerations, and user needs requires expertise across various platforms and frameworks, often leading to lengthy development cycles and inconsistent interfaces.

Here's where Model-Driven Architecture (MDA) [4] emerges as a powerful solution. This paradigm shift in software development focuses on abstracting complexity through high-level models, enabling a more efficient and reliable approach to application creation. By harnessing the power of MDA, we can bridge the gap between design and implementation, unlocking significant benefits for smart tourism web applications.

This paper presents an approach that leverages MDA principles to accelerate the development and enhance the quality of smart tourism web interfaces. By generating code directly from comprehensive metamodels that capture essential web elements, we aim to:

- Reduce manual coding efforts: Automate tedious tasks like UI element generation, freeing developers to focus on core functionalities and business logic.
- Ensure consistency: Guarantee a unified look and feel across the application, enhancing user experience and brand recognition.
- Improve maintainability: Simplify code updates and modifications, allowing developers to adapt to changing requirements with greater ease.

This work aims to empower developers in the smart tourism landscape with an efficient and reliable methodology for creating exceptional user experiences, ultimately contributing to the continued growth and success of the sector.

The remainder of this paper is structured as follows. In Section 2, we will delve into the Background and Related Work, examining existing methodologies and technologies in web development that inform the approach. Section 3 will present the proposed approach. Section 4 discusses the potential and the challenges of this approach. In Section 5, we will conclude the paper and some future work.

II. Background and Related Work

A. Model Driven Architecture (MDA)

Model-Driven Architecture (MDA) [3] represents a paradigm in software engineering that emphasizes the separation of system functionality from its implementation details. At its core, MDA promotes the use of models to represent the essential aspects of a system, which can then be transformed into executable code through automated processes. This approach enables developers to focus on high-level design concerns while streamlining the implementation process. By leveraging MDA principles, developers can enhance the maintainability, scalability, and adaptability of software systems.

B. Smart Tourism

The concept of Smart Tourism [5] encompasses the integration of information and communication technologies to enhance various aspects of the tourism industry, including destination management, visitor experiences, and marketing strategies. With the proliferation of mobile devices and ubiquitous connectivity, Smart Tourism initiatives leverage technologies such as location-based services, augmented reality, and data analytics to provide personalized and immersive experiences for travelers. By harnessing the power of digital technologies, Smart Tourism aims to optimize resource utilization, improve tourist satisfaction, and foster sustainable tourism practices [6].

C. Related Work

The use of MDA the development of user interfaces has been explored in various studies.

This paper [7] presents an approach to integrate accessibility into the adaptation of User Interfaces. The authors propose a model-driven approach that automatically generates accessibility adapted User Interfaces based on MDA principles. This approach is particularly relevant for this work as it demonstrates the potential of MDA in creating user-centric designs.

In this work[8] , the authors describe the application of a development approach, based on MDA, to create a Graphical User Interface (GUI) for the Amazon Integration and Cooperation Project for Modernization of Hydrological Monitoring. This work provides valuable insights into the practical application of MDA in GUI development.

Developing Single Page Applications (SPAs) [9]involves dealing with two main issues: platform compatibility and navigation management. A paper [10] introduces a DSL-based approach that addresses these issues by using a Domain-Specific Language (DSL) [11] to define the SPA components in an abstract and concise way, and then generating code and controlling the navigation of the SPA. The paper follows Garrett's visual vocabulary [12] to specify the elements of the SPA, such as pages, connectors, and transitions. This paper relates to this research because it shows the benefits of using DSL and MDE for web interface development.

III. Proposed Approach

In this section, we present the MDA approach for user interfaces of web application development in smart tourism. The approach consists of four main steps:

- *Step1*: defining the smart tourism domain model and the web application model as subsets of it. The smart tourism domain model is a comprehensive representation of the concepts, relationships, and constraints that are relevant for the smart tourism domain. It covers aspects such as tourist attractions, services, activities, accommodations, ratings, etc. The web application model is a subset of the smart tourism domain model that focuses on the features and functionalities that a specific web application should provide to the users. It defines the web pages, the content, the layout, and the interactions that the web application should support.
- *Step2*: defining the navigation metamodel. The navigation metamodel is a generic model that describes how users can navigate through the web application. It defines the navigation structure, the navigation paths, the navigation rules, and the navigation elements. The navigation metamodel is based on the Garret's visual vocabulary [12], ensuring that the navigation is consistent with the web application features and functionalities.
- *Step3*: defining the implementation metamodels for web elements. The implementation metamodels are specific models that define how the web elements are implemented using web technologies. It includes models for HTML5 elements, CSS stylesheets, JavaScript functions, etc.
- *Step4*: applying model transformations to generate code from models. The model transformations are automated processes that transform models from one level of abstraction to another. They include transformations from the web application model to the navigation metamodel from the navigation metamodel to the implementation metamodels for web elements (model to model transformation) [13], and from the implementation metamodels for web elements to the code (model to text transformation) [14]. The model transformations ensure that the code is generated according to the models and that the code is correct and consistent.

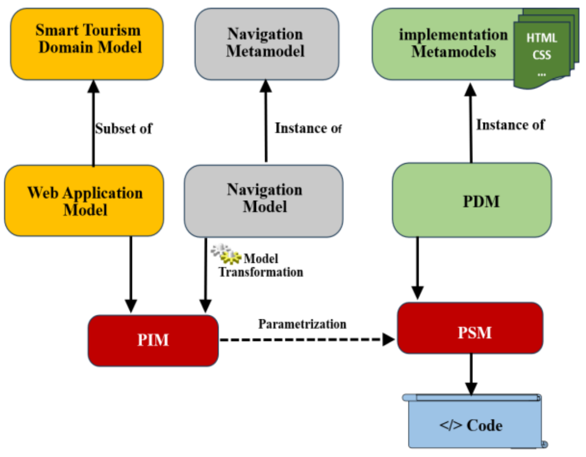

Figure 1: Overview of the approach

Figure 1 shows an overview of the proposed approach and the levels of abstraction that we use. We adopt the standard MDA levels of PIM [4], PSM [4] and PDM [4] :

- *Platform Independent Model (PIM):* This is the highest level of abstraction, where the models are independent of any specific platform or technology. The smart tourism domain model and the web application model are examples of PIMs.
- *Platform Specific Model (PSM):* This is the intermediate level of abstraction, where the models are specific to a certain platform or technology, but still abstract enough to be independent of the implementation details. The navigation metamodel and the implementation metamodels for web elements are examples of PSMs.
- *Platform Description Model (PDM):* This is the lowest level of abstraction, where the models are fully detailed and describe the implementation of the web elements using web technologies.

We also introduce intermediate levels to capture the specific aspects of web applications. We explain each level and its role in the following subsections.

A. *Smart Tourism Domain Model*

The first step of the approach is to define the smart tourism domain model. The Smart Tourism Domain Model is a structured representation that outlines the various components and their relationships within the tourism sector. It integrates information about attractions, accommodations, transportation, and users to facilitate smart tourism.

This diagram comprises numerous entities each equipped with distinctive attributes such as:

- *Attraction*: This entity represents the tourist attractions that can be either physical (such as natural landscapes, monuments, etc.) or cultural (such as festivals, museums, etc.). It has attributes such as accessAcount, id, and isKidAllowed.
- *Location*: This entity represents the geographical location of the attractions, or amenities. It has attributes such as name, description, population, country, city, and accessibleBy.
- *Itinerary*: This entity represents the travel plan of the users, including the departure and arrival locations, dates, and visits.
- *Visit:* This entity represents the visit of a user to an attraction, including the title, start and end dates, and the rating.
- *Transportation*: This entity represents the mode of transportation used by the users, which can be either personal (such as car or bike) or public (such as bus or train).
- *User:* This entity represents the tourist who uses the smart tourism system. It has attributes such as firstName, lastName, email, locationOfBirth, and account.
- *Account:* This entity represents the login information of the user, including the login and password.
- *Address*: This entity represents the postal address of the user, including the street2, city, country, and zip code.
- *Hotel:* This entity represents the hotel where the user can stay. It has attributes such as classification, room, and amenity.
- *Room:* This entity represents the type and price of the room in the hotel.
- *Amenity:* This entity represents the facilities and services provided by the hotel, such as phone and number of beds.
- *Image:* This entity represents the image of the attraction, the hotel, or the user. It has an attribute image that stores the image data.

The entities are connected by lines that indicate the relationships between them, such as one-to-one, one-to-many, or many-to-many. For example, a user can have one account, but can visit many attractions. An attraction can have many images but can be in one location. A hotel can have many rooms but can belong to one classification.

The Smart Tourism Domain Model is an example of a PIM, which is the highest level of abstraction in the MDA approach. It is independent of any specific platform or technology, and it captures the essential concepts and features of the smart tourism domain. It can be used to instantiate other models, such as the Web Application Model and the Navigation Metamodel, which are more specific and detailed. The full version of the proposed smart tourism domain model is available at this GitHub repository [15].

B. *Navigation Metamodel*

The next phase in the approach is to define the navigation metamodel, which captures the structure and behavior of the web application, such as the navigation paths, the user actions, the events, etc. The navigation metamodel consists of various elements and their relationships, which are inspired by Garrett's visual vocabulary [12], a simple and intuitive notation for web design.

The elements of the navigation metamodel are:

- *Area*: This element represents a logical grouping of related content or functionality within the web application. An area can be either a single page or a collection of pages.
- *Flow*: This element represents a sequence of areas that the user can navigate through within the web application, such as a checkout process or a registration form.
- *Connector*: This element represents a connection between two elements, such as a link, a button, or a form submission. A connector has a type attribute that specifies the kind of connector it is, such as a simple connector or a conditional connector.
- *Conditional*: This element represents a condition that determines the outcome of a connector, such as a user input, a system state, or a user preference. A conditional has a conditions attribute that specifies the logic of the condition.

The relationships between the elements of the navigation metamodel are:

- *Instance* of: This relationship indicates that an element is an instance of another element, such as a page being an instance of an area, or a button being an instance of an element. The instance of relationship is represented by a solid line with an open arrowhead pointing to the element that is instantiated.

- *Generates*: This relationship indicates that an element generates another element, such as an iterative area generating a page, or a conditional connector generating a conditional branch. The generates relationship is represented by a dashed line with an open arrowhead pointing to the element that is generated.
- *Continues*: This relationship indicates that an element continues another element, such as a continue from element continuing a flow, or a continue to element continuing a flow. The continues relationship is represented by a solid line with a closed arrowhead pointing to the element that is continued.
- *Connects*: This relationship indicates that an element connects to another element, such as a connector connecting two elements, or a conditional connecting to a connector. The connects relationship is represented by a solid line with no arrowheads.

The navigation metamodel helps to design web applications that are consistent, intuitive, and user-friendly. It allows to define the navigation paths, the user actions, the events, and the outcomes of the web application. It also enables to specify the type and the appearance of the web elements, such as pages, files, buttons, links, forms, etc. The full version of this metamodel and some examples of its applications are available at this GitHub repository [16].

C. Implimentation Metamodel

The diagram in Figure 2 is a part of the implementation metamodel for HTML5 elements, which is the third step of the approach. The implementation metamodels capture the HTML5 elements that are used to create web pages of web applications. We use UML class diagrams to represent the implementation metamodels, as they are consistent with the previous models and allow us to define the attributes and operations of the web elements. We also use attributes and associations with additional information, such as the type, the name, and the conditions of the elements.

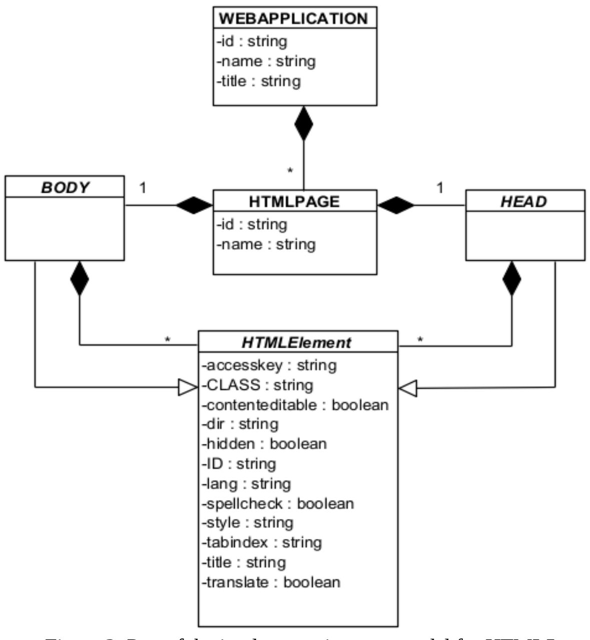

Figure 2: Part of the implementation metamodel for HTML5

The diagram shows five main classes: Application, HTMLElement, BODY, and HEAD. Each class represents a type of web element that has its own attributes and operations. For example, the Application class represents the web application itself, and has attributes such as id, Name, and Title. The Element class represents a generic web element, and has attributes such as id and Name. The HTMLElement class extends the Element class and includes several additional attributes that are common to all HTML elements, such as accesskey, class, contenteditable, dir, hidden, id, lang, spellcheck, style, tabindex, title, and translate. The BODY and HEAD classes extend the HTMLElement class.

The diagram also shows the relationships between the classes, which are represented by lines with different symbols. The relationships indicate how the web elements are related to each other, such as inheritance, containment, or association. For example, there is inheritance relationship between HTMLElement and Element class, meaning that it has all the attributes and operations of the Element class plus its own specific ones. The composition shape from the Application class to the Element class indicates that the Application class contains the Element class, meaning that the web application can have one or more web elements.

The diagram also contains multiplicities of the relationship between classes, for example, the multiplicity of the relationship between HTMLPAGE and head or body indicates that each HTML page has one head and one body element.

This diagram in Figure 2 is only a part of the implementation metamodel for HTML5, which covers the basic web elements such as HTML, BODY, and HEAD. The complete metamodel for HTML5 elements is available at this GitHub repository [17], which includes more classes and relationships for other web elements, such as DIV, SPAN, P, H1, A, IMG, FORM, INPUT, etc.

D. Model Transformations

The fourth step of the approach is to apply model transformations [18] to generate code from models. We use QVT [19], the standard language for model transformations in MDA, to define the transformation rules.

We apply two types of transformations: horizontal [20] and vertical [21]. Horizontal transformations are between models at the same level of abstraction, such as from the web application model to the navigation model. Vertical transformations are between models at different levels of abstraction, such as from the navigation model to the HTML model.

IV. DISCUSSION

In this section, we discuss the potential and the challenges of the approach for smart tourism web development. This MDA approach has four strengths for smart tourism web development:

Efficiency: This work automates HTML code generation from domain models, saving time and effort in creating complex web elements, such as booking forms, maps, and recommendations. This allows developers to focus on improving core features and user experiences, making smart tourism applications more attractive and appealing.

Consistency: This approach ensures consistency in design and code by following the metamodels, building user trust and brand recognition. This is vital in the competitive market of smart tourism, where quality and reliability are key factors.

Maintainability: This research enables easy maintenance and reuse of code by using modular components that can be updated and adapted without affecting other parts. This is important for smart tourism applications, which need to keep up with changing travel trends and user preferences.

Empowerment: This approach empowers developers to concentrate on high-level design by automating tedious tasks. This fosters creativity and innovation, enabling developers to create engaging and personalized user experiences that resonate with travelers. It also allows teams to spend more time on refining and perfecting applications, achieving excellence and improvement.

The MDA approach proposed also faces some challenges that need to be addressed to fully realize its potential for smart tourism web development; in beyond HTML, this approach needs to integrate CSS and JavaScript to achieve more customization and dynamic behavior for smart tourism applications, such as visual appeal and interactivity. This can be done by adding more PDMs for each platform or by integrating with external styling and scripting languages.

V. CONCLUSION

This paper explored the potential of MDA for accelerating the development of user interfaces in smart tourism web applications. The proposed approach offers significant advantages, including reduced development time, improved consistency, enhanced maintainability, and empowered developers. While addressing limitations like limited customization, further enhancements hold the promise of unlocking an even greater impact. By embracing MDA and tailoring it to the specific needs of smart tourism, we can contribute to crafting more engaging, efficient, and personalized user experiences, ultimately shaping the future of the smart tourism sector.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Higher Education, Scientific Research and Innovation, the Digital Development Agency (DDA) and the CNRST of Morocco (Alkhawarizmi/2020/32).

REFERENCES

[1] D. Buhalis, “Marketing the competitive destination of the future,” *Tourism management,* vol. 21, no. 1, pp. 97-116, 2000.

[2] R. Sivarethinamohan, "Exploring the Transformation of Digital Tourism: Trends, Impacts, and Future Prospects." pp. 260-266.

[3] Chandrasekaran, B, Josephson, J.R, Benjamins, and V.R, *Tourism Informatics: Visual Travel Recommender Systems, Social Communities, and User Interface Design*.

[4] J. Miller, and J. Mukerji, *MDA Guide Version 1.0.1,* omg/2003-06-01, Object Management Group, 2003.

[5] D. Buhalis, and A. Amaranggana, "Smart tourism destinations." pp. 553-564.

[6] U. Gretzel, M. Sigala, Z. Xiang, and C. Koo, “Smart tourism: foundations and developments,” *Electronic Markets,* vol. 25, no. 3, pp. 179-188, 2015-08-01, 2015.

[7] L. Zouhaier, Y. B. Hlaoui, and L. J. B. Ayed, "Generating Accessible Multimodal User Interfaces Using MDA-Based Adaptation Approach." pp. 535-540.

[8] J. d. Monte-Mor, E. O. Ferreira, H. F. Campos, A. M. d. Cunha, and L. A. V. Dias, "Applying MDA Approach to Create Graphical User Interfaces." pp. 766-771.

[9] G. Fink, and I. Flatow, "Introducing Single Page Applications," *Pro Single Page Application Development: Using Backbone.js and ASP.NET*, G. Fink and I. Flatow, eds., pp. 3-13, Berkeley, CA: Apress, 2014.

[10] L. Naimi, H. Abdelmalek, and A. Jakimi, “A DSL-based Approach for Code Generation and Navigation Process Management in a Single Page Application,” *Procedia Computer Science,* vol. 231, pp. 299-304, 2024.

[11] S. Cook, “Domain-Specific Modeling and Model-Driven Architecture,” *MDA Journal*, January 2004, 2004.

[12] J. J. Garrett. "A visual vocabulary for describing information architecture and interaction design," http://www.jjg.net/ia/visvocab/.

[13] P. Langer, M. Wimmer, and G. Kappel, "Model-to-Model Transformations By Demonstration." pp. 153-167.

[14] S. Sendall, and W. Kozaczynski, “Model transformation: the heart and soul of model-driven software development,” *IEEE Software,* vol. 20, no. 5, pp. 42-45, 2003.

[15] L. Naimi. "Smart Tourism Domain Model Diagram," February 03, 2024, 2024; https://github.com/lnaimi/Smart-Tourism-Domain-Model.git.

[16] L. Naimi. "Navigation metamodel complete diagram and examples," February 03, 2024, 2024; https://github.com/lnaimi/Navigation-metamodel.git.

[17] L. Naimi. "HTML Metamodel, complete diagram," September 03, 2023, 2023; https://github.com/lnaimi/HTML-Metamodel/blob/main/CompleteHTML5ClassD.jpg.

[18] F. Marschall, and P. Braun, "Model Transformations for the MDA with BOTL." pp. 25-36.

[19] Omg, *MOF QVT Final Adopted Specification*, 2005.

[20] M. Wimmer, and L. Burgueño, "Testing m2t/t2m transformations." pp. 203-219.

[21] J. Oldevik, T. Neple, R. Grønmo, J. Aagedal, and A.-J. Berre, "Toward Standardised Model to Text Transformations," *Model Driven Architecture – Foundations and Applications*, pp. 239-253, 2005.