Leveraging Web Analytics for Automatically Generating Mobile Navigation Models

Andrea Salini*, Ivano Malavolta†, Fabrizio Rossi*
*Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy - andrea.salini@student.univaq.it, fabrizio.rossi@univaq.it
†Vrije Universiteit Amsterdam, The Netherlands - i.malavolta@vu.nl

***Abstract*—Today activity on smartphones and tablets accounts for an incredible 60% of the time spent on digital media in the United States. People will rely more and more on mobile devices for performing very different activities like purchasing products, messaging, ordering food, booking holidays, etc. Nowadays, a common technological trait is that browsing even a responsive website on a device can be "uncomfortable" and always more often a dedicated mobile app is introduced from an existing website. However, the navigation design of a mobile app can be totally different from the one of a website due to many factors such as display size, different physical and modal contexts, etc. Therefore, developers cannot simply replicate the navigation design of a website into the corresponding mobile app, but a complete redesign is needed.**

In this paper we present an approach for leveraging web analytics for automatically generating mobile navigation design models. The approach mines usage data of a website, builds a model of the web usage patterns coming from both web and mobile-specific usage sessions, transforms it into a mobile-oriented navigation tree, and generates a mobile-oriented navigation model, usable by interaction designers. Such a generation is totally automated and amounts to solve a variant of the Steiner Tree problem with revenues, budget and hop constraints. The feasibility of the proposed approach has been evaluated on an existing institutional website.

***Index Terms*—Web usage mining; Mobile app design, IFML, Navigation design.**

I. Introduction

Mobile devices are replacing traditional desktop and mobile instruments; people rely on mobile devices for surfing the web, purchasing products, or to be part of a social network; as a result, the mobile applications market now counts more than two millions applications, downloaded billions of times per year [13]. As a clear indicator of this situation, today the total activity on smartphones and tablets accounts for an incredible 60% of the time spent on digital media in the United States [1].

Nowadays, a common technological trait is that browsing even a responsive website on a device can be "uncomfortable" and always more often a dedicated mobile app is introduced from an existing website (e.g., Facebook, Amazon, EBay, YouTube, Wikipedia, to name a few). Along the same lines, websites of governmental agencies must follow specific regulations regarding their information architecture and many services are being provided also via dedicated apps (with less stringent regulations about their information architecture) [1]. However, the structure and the information architecture of a mobile app can be totally different from the one of a website due to many factors, like display size, different physical and modal contexts, etc. Developers cannot just replicate the structure and the information architecture of a website into a corresponding mobile app, but a complete redesign is needed.

The focus of this paper is on the **navigation design** of mobile apps, defined as the structure of a mobile app in terms of its *views* and the *navigation flows* among them. More specifically, in this paper we present an approach that exploits existing data coming from the web usage analytics of already developed websites for generating the mobile navigation model of their corresponding mobile apps. The whole approach is automatic and produces models that (i) can be easily understood and manipulated by interaction designers, and (ii) are specifically tailored to the navigational patterns of the users of the initial websites. Given a website w, our approach (i) mines usage data of w, (ii) builds a graph-based model G_w of the web usage patterns coming from any usage session of w, (iii) analyses and refactors G_w into a navigation tree T_w, representative of the mobile-oriented navigation design of w, and (iv) generates a mobile-oriented navigation model, usable by interaction designers for reasoning on the navigation design of the w companion mobile app. The refactoring step consists in solving a variant of the Steiner Tree problem with revenues, budget and hop constraints, whereas the final navigation model conforms to the Interaction Flow Modeling Language[2] (IFML), an OMG standard for representing the content, user interaction and control behaviour of the front-end of software applications (including mobile apps).

The main **contributions** of this paper are the following:

- definition of an approach for automatically mining raw web usage data and representing it as an abstract navigation graph;
- formalization of an automatic transformation from a generic navigation graph of a website into a mobile-oriented navigation tree;
- definition of a transformation from a mobile-oriented navigation tree into an IFML model, usable by interaction designers;
- implementation of the whole proposed approach exploiting Google Analytics, Python, WordNET, and the IFML

[1]http://www.usa.gov/mobile-apps

[2]http://www.ifml.org

open source editor;

- evidence of the feasibility of the proposed approach by applying it on a real-world institutional website.

The rest of the paper is organized as follows. Section II presents the main concepts at the basis of our work, whereas in Section III we describe the proposed approach. Section IV describes technological aspects of our implemented tool, Section V describes the application of the approach on a real website, and Section VI discusses related work. Section VII closes the paper and discusses future work.

II. Background

A. Web Analytics

The Digital Analytics Association defines web analytics as the measurement, collection, analysis and reporting of internet data for the purposes of understanding and optimizing web usage[3]. Web analytics exploits software and tools to gather data obtained from the study of the visitors' behavior while they navigate the web. The collected data are then analyzed in order to improve the interaction between the user and the web with the aim of increasing the user conversions and, in other words, getting a better profit. So, web analytics is not just a process for measuring web traffic but can be used as a tool for business and market research, and to assess and improve the effectiveness of a website.

Most of web analytics processes down to four essential stages, or steps, namely: collection of data, processing data into information, developing KPI and formulating online strategy. Each of these stages impacts or can impact the stage preceding or following it; in other words they are sequentially connected and not isolated from each other.

The two main categories of web analytics are the *off-site* and the *on-site* web analytics, which is the most common and measure a visitors behavior once on your website. To each of these categories are associated several web analytics software and services used to collect and display data about visiting website users.

In the context of our approach, web analytics is exploited as the main source of information for the automatic generation of mobile-oriented navigation models of a website.

B. Navigation Design of Mobile Applications

Current practices in data-intensive mobile applications development are still plagued by a number of recurring issues and challenges [14]. Even if many of them have a technical nature (e.g., mobile devices fragmentation, multi-device testing, code reuse across platforms), from our experience and interactions with professionals in the field we actually noted that issues and challenges pertaining to the *design* of a mobile application can have a much bigger and disastrous impact on its success [5], [14]. In this context we focus on the navigation design of mobile apps. Navigation design can be defined as the set of screen elements that allow the user to move through the information architecture [6]. More intuitively, navigation design can be seen as the structure of a mobile app in terms of its *views* and the *navigation flow* between them. Navigation design helps interaction designers in reasoning about how information is structured, how the user can participate and understand presented information, and how navigation and access are facilitated [5]. From these reflections it is evident that good navigation design is a key factor of a successful mobile application.

The navigation design has always been a purely creative and manual activity of designers, even when reasoning on the (re-)design of an existing website or companion software. In this context, many designers still tend to *borrow concepts and best practises* about interaction and navigation design of traditional (web) applications. However, working on a mobile application is totally different from working on a desktop program or website: the smaller display, different styles of user interaction, and contextual dependencies have a major impact on the interaction and navigation design for mobile applications, which in turn have a strong influence on the success of mobile applications [14].

In the context of our approach, generated navigation models will support designers in reducing and controlling the design space by automatically obtaining a navigation model from objective navigation patterns mined from existing data (i.e., the web analytics data of the web site).

C. The Interaction Flow Modeling Language (IFML)

As reported in its official book [9], IFML exploits the OMG Model Driven Architecture to support the specification of the front-end of applications, independently of the technological details of their realization. IFML addresses the following concerns of front-end modeling: the composition of the view, the content of the view, the commands, the actions, the effects of interaction and the parameter binding. Figure 1 shows the main entities provided by the IFML language, that are:

- *container*: it defines the elements that make up the interface, that can be either displayed simultaneously or in a mutually exclusive manner;
- *view component*: it defines the data that is displayed by the application to the user and the user input supplied to the application;
- *events*: the supported interaction events;
- *actions*: the business components that can be triggered by events;
- *interaction flows*: the effects that the execution of events or actions have on the state of the interface;
- *parameter bindings*: the exchanged data between interface elements and actions.

The described entities, appropriately combined, form an IFML model; such a model consists of one or more view containers, which can contain view components. A view component denotes the publication content, or interface elements for data entry and can have input and output parameters. The view container and view component elements can be associated with events. The effect of an event is represented by an interaction flow connection or a triggering of an action. An

[3]http://www.digitalanalyticsassociation.org

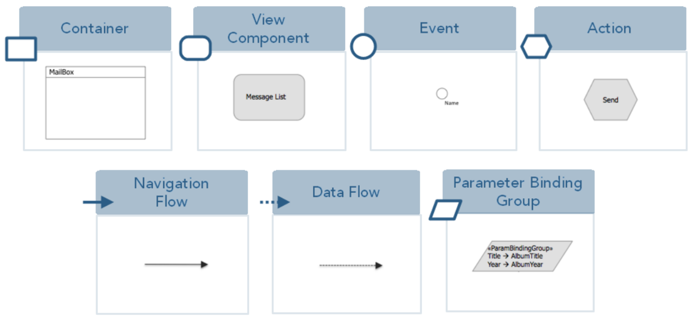


Fig. 1. Main IFML entities

input-output dependency between elements can be specified through parameter bindings associated with navigation flows or through data flows. This combinations of entities are expressed by IFML through a visual modeling language based on the MDA standard.

In the context of our approach, IFML is used as the target modeling language of the whole generation process. Basically, IFML acts as a front-end language to designers, thus allowing us to mask the complexity and the heavy cognitive load that raw analytics data or raw navigation graphs may impose on interaction designers. We chose IFML as target in order to provide to designers models that can be easily understood and for which it is likely that they are already trained (we recall here that IFML is an OMG standard language).

III. The Approach

As shown in Figure 2, our approach is composed of four main stages, each of them taking as input and producing as output different kinds of artifacts. The first and only input artifact to the whole approach is the *web analytics report* containing information about the all the usage sessions of a website w. Intuitively, the web analytics report can be seen as a large table in which each row represents a specific navigation between two pages p_s and p_t of w performed on either a mobile or desktop browser; columns contain information like the name and title of p_s and p_t, the device used for the navigationg, the number of times the navigation has been performed, the average time on p_t, etc. The web analytics report can be easily obtained when using standard web analytics tools such as Google Analytics.

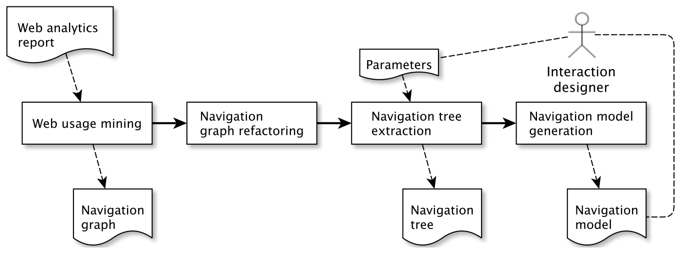


Fig. 2. Overview of the approach

The **web usage mining** stage (detailed in Section III-A) takes as input the web analytics report and generates a *navigation graph G*. G_w is a directed graph where each node represents a single page of w and each arc (u, v) represents a navigation between their corresponding pages in w; each arc has a weight corresponding to the number of navigation transitions performed between the pages represented by its source and target nodes.

The **navigation graph refactoring** stage (detailed in Section III-B) takes as input the navigation graph G_w and refactors and enriches it with additional information in order to ease the subsequent stages. For example, it eliminates redundant nodes, it performs relabeling operations based on the title of the pages represented by the nodes, adds intermediate nodes, etc. Also, each node of G_w has been enriched with specific metrics (e.g., its centrality) in order to provide context about its role within the whole graph.

The **navigation tree extraction** stage (detailed in Section III-C) takes as input the navigation graph G_w and, according to a set of *parameters* provided by the interaction designer, transforms G_w into a *mobile-oriented navigation tree* T_w. The root of the navigation tree is the node representing the homepage of the w. In this context the term "mobile-oriented" refers to the fact that the generated tree satisfies a set of mobile-specific constraints, mainly related to the maximum length that all navigation paths can have, the maximum navigation branches that the user can navigate, etc.

The **navigation model generation** stage (detailed in Section III-D) takes as input the navigation tree T_w and creates a model conforming to IFML. The generated model is meant to be used by interaction designers as a basis for reasoning on the navigation design of the companion mobile app of w; the overall idea is that interaction designers have a working artifact that is specifically tailored to the characteristics and web usage patterns of the users of w, who will likely be representative enough of the potential users of the companion mobile app.

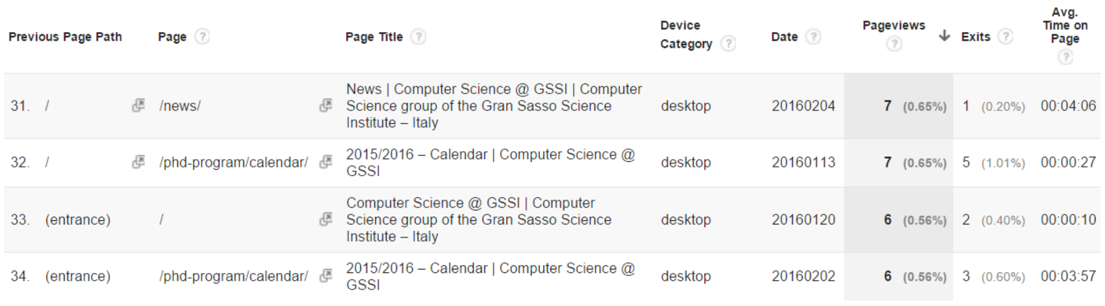
It is important to note that all the intermediate artifacts and complexity of the stages described above are masked to interaction designers, who just have to provide the web analytics report as input and use the automatically generated IFML model as basis for their design activities. Optionally, interaction designers can iterate the generation process by fine-tuning the parameters provided in the navigation tree extraction stage (see Section III-C) in order to obtain the IFML model that better represents the needed navigation patterns.

A. Web Usage Mining

This section details the building process of the navigation graph G_w from navigation data obtained by a website analitycs tool.

Our approach requires a minimal set of navigation data that can be delivered by most popular analytics tools, that is, *Page URI*, *Previous Page URI*, *Page Title*, *Device Category*, *Date*, *Pageviews*, *Avg. Time on Page* and *Exits*. As an example, Figure 3 shows a screenshot from Google Analytics containing all relevant information.

Thus, given a report like the one in Figure 3, a navigation graph $G_w = (N, A)$ is a directed and weighted graph with the



	Previous Page Path	Page	Page Title	Device Category	Date	Pageviews	Exits	Avg. Time on Page
31.	/	/news/	News | Computer Science @ GSSI | Computer Science group of the Gran Sasso Science Institute – Italy	desktop	20160204	7 (0.65%)	1 (0.20%)	00:04:06
32.	/	/phd-program/calendar/	2015/2016 – Calendar | Computer Science @ GSSI	desktop	20160113	7 (0.65%)	5 (1.01%)	00:00:27
33.	(entrance)	/	Computer Science @ GSSI | Computer Science group of the Gran Sasso Science Institute – Italy	desktop	20160120	6 (0.56%)	2 (0.40%)	00:00:10
34.	(entrance)	/phd-program/calendar/	2015/2016 – Calendar | Computer Science @ GSSI	desktop	20160202	6 (0.56%)	3 (0.60%)	00:03:57

Fig. 3. Example of web analytics report (fragment)

following properties:

1) The set N of nodes of G_w contains an element for each webpage of w, plus two special elements, called respectively *(entrance)* and *(exit)*.
2) There is an arc from node u to node v, *arc(u,v)*, if there exists a link from a node (web page) u to a node (web page) v that has been navigated at least once by users.
3) Each arc *arc(u,v)* has weight equals to the number of times that the *link(u,v)* has been navigated by the users;
4) The node *(entrance)* represents all the webpages that do not belong to the website and from which users reached the website, that is, *(entrance)* is connected to all pages that are visited as first in a user session.
5) The node *(exit)* represents the exit from w, that is, the last visited page in a user session is connected to *(exit)*.

Moreover, each node of G_w has several attributes derived from the web analytics report, namely: *Id*, *label*, *pageName*, *visualization*, *averageTime*, *age*. Eventually, this set of attributes can be further extended once the navigation graph has been built.

TABLE I
EXAMPLE OF NODE ATTRIBUTES IN G_w (FRAGMENT)

id	label	pageName	visualization	averageTime
1	/	HOME PAGE	6	10
2	/news/	News	7	246
3	/phd-program/calendar	2015-2016-Calendar	13	124
4	(entrance)	ENTRANCE	-	-
5	(exit)	EXIT	-	-

In what follows, steps needed to build G_w from the report of Figure 3 are detailed:

1) Read the first row of the web analytics report and create in G_w the nodes corresponding to the *Page* and *Previous Page Path* attributes. The label attribute for these node is set to the URI value of the webpage so the two nodes in G_w are */* and */news/*. The other attributes are derived directly from the report. Note that the *pageName* string is not entirely equal to *Page Name* because it contains only the part related to the webpage, and not to the whole website. Note also that the attribute *averageTime* is equal to the value in seconds of *Avg. Time on Page*. After adding them, those two new nodes are linked with an arc with weight equal to 7, corresponding to the *visualization* value in the web analytics report. Finally the leaving traffic from */news/* is directed to the special node *(exit)*.
2) Read the second row and only the node */phd-program/calendar/* with its attributes is added to the graph because the node */* is already in G_w. Two arcs are then added to G_w: one from */* to */phd-program/calendar/* with weight 7 and the other to */phd-program/calendar* with weight 5.
3) Read the third row and add the special node *(entrance)* to G_w. Then the attribute for */* are added; because this node refers to the home page of the website the attribute *pageName* is set to *HOME PAGE*. Finally two arcs are added to G_w.
4) In the fourth row appears for the second time the page */phd-program/calendar/*. So, after adding a new arc from */* to */phd-program/calendar*, the node attributes of *visualization*, *age* and *averageTime* are updated: *visualization* is incremented by 7; the value of the *age* attribute is replaced by the new *Date* value only if the new date is before the old one; *averageTime* is updated by performing an arithmetic average.
5) Continue, until there are no rows to be analyzed.

The resulting navigation graph G_w is showed in Figure 4.

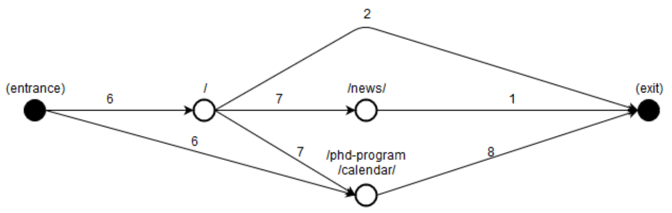


Fig. 4. Navigation graph G_w (fragment)

B. Navigation Graph Refactoring

Once G_w has been fully generated, several measures have been applied on it and new metrics are added as attributes of the nodes: *centrality*, *current-flow closeness centrality* and *communicability centrality* [4]. The *centrality* of a node u is the fraction of nodes it is connected to the *current-flow closeness centrality* is a variant of closeness centrality based on effective resistance between nodes in a network (this metric is also known as information centrality). The *communicability* between pairs of nodes u, v in G_w is the sum of closed walks of different lengths starting at node *u* and ending at node *v*. These metrics will be exploited when estimating how much a node is in a central and significant position within G_w.

This stage consists also of four refactoring operations, that are: *elimination of redundant nodes*, *relabeling*, *predecessor node addition* and *details node addition*. *Elimination of redundant nodes* deletes from G_w all the unnecessary nodes (e.g. isolated nodes or nodes that refer to old non-existing pages). The *relabeling* operation can be applied in those websites that make use of static URLs; more specifically, as shown in Figure5, let */predecessor/* be a node in G_w with an arc from it to a node labeled */predecessor/successor/*. It is plausible that */predecessor/successor/* is hierarchically a successor of */predecessor/*. Then */predecessor/successor/* can be renamed in */successor/*. If there are several */predecessor/successor.../* nodes but no */predecessor/* node, then the *predecessor node addition* operation adds to G_w the node */predecessor/* (see Figure 6). Instead, if this node is in G_w but is not hierarchically a predecessor of any */predecessor/successor.../* node, then *predecessor node addition* make it a predecessor of these nodes (see Figure 7). In both cases the */predecessor/* node attributes have been updated appropriately. Finally, the purpose of the *details node addition* is to identify those nodes which represent pages that contain a specific instance of data (e.g., products in an e-commerce website or profiles of researchers in an academic site). Once those nodes have been identified, they are replaced by a single *details* node (see Figure 8). To identify those nodes we use centrality and visualization values and a semantic analysis made on the *pageName* attribute.

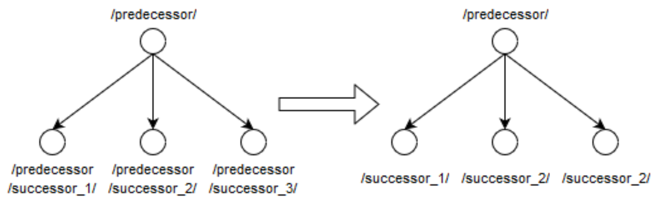

/predecessor/
/predecessor /successor_1/
/predecessor /successor_2/
/predecessor /successor_3/
/predecessor/
/successor_1/
/successor_2/
/successor_2/

Fig. 5. Relabeling operation

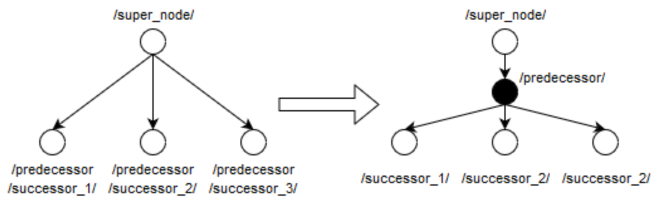

/super_node/
/predecessor /successor_1/
/predecessor /successor_2/
/predecessor /successor_3/
/super_node/
/predecessor/
/successor_1/
/successor_2/
/successor_2/

Fig. 6. Predecessor node addition (1)

C. Mobile-Oriented Navigation Tree Extraction

Given the just refactored navigation graph G_w, the goal of this stage amounts to generate the mobile-oriented navigation

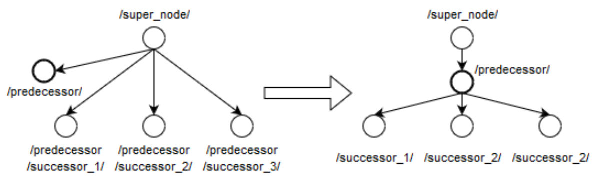


Fig. 7. Predecessor node addition (2)

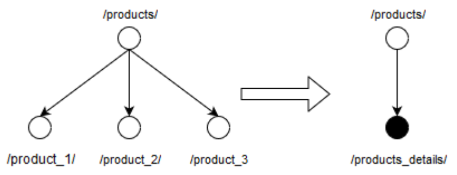


Fig. 8. Details node addition

tree T_w. Roughly speaking, T_w will contain nodes of G_w with high centrality and largely requested by users and it will have reasonable depth and a limited number of children for each node. In order to generate T_w we exploit a particular formulation of the Steiner Tree Problem with Revenues, Budget and Hop Constraints (STPRBH): the Garcia-Gouveia Hop (GG-Hop) formulation [2]. We choose this formulation because it (i) explicitly handles the hop limit (that is, bounds the depth of T_w) and (ii) allows us to easily model extra constraints. Namely, it is possible to model constraints that bound the maximum depth of each node in the final tree and constraints that limit the maximum degree of each node. This gives to interaction designers the opportunity of tuning the parameters so as to obtain trees with different shapes and select the one that best fits with their specific needs. Finally, each node has a profit value, defined as the combination of his centrality and visualization parameters, because the STPRBH generates the tree that maximizes the total profit.

D. Navigation Model Generation

The last stage of our approach is the transformation from the mobile-oriented navigation tree T_w into its corresponding IFML model M_w. Since T_w represents exclusively navigational aspects among pages, without focussing on the specific elements contained into each page, M_w will contain only two IFML concepts, that are*View Containers* and *Navigation Flows*.

The transformation has been realized as a variant of a tree *depth-first search* (DFS). For example, assuming that we want to transform the navigation tree shown in Figure 9, the first step of the transformation is to build an empty IFML model (the final IFML model is depicted in Figure 10); then, the DFS starts from the root of T_w and creates the *root* view container with attribute *name* equal to the string "root" and a unique *id*. Now, the DFS continues by considering *child1* in T_w and generating (i) the *child1* view container in M_w and (ii) the *InteractionFlow* from the *root* view container to the *child1* view container.

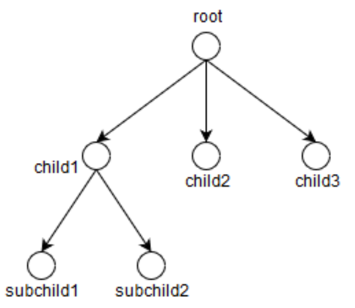


Fig. 9. Predecessor node addition (2).

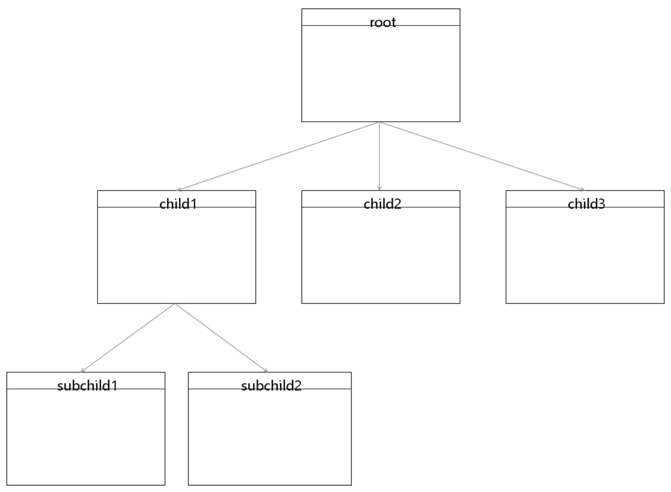


Fig. 10. Details node addition.

The DFS continues and a child of *child1* is searched, suppose *subchild1*. Again a new view container for *subchild1* and the *InteractionFlow* between *child1* and *subchild1* are created in M_w. The DFS continues until all nodes of the input navigation tree T_w have been visited, and thus ported into the output IFML model M_w. In our example, all the nodes in the tree are searched and transformed in this order: *subchild2*, *child2* and *child3*.

It is important to note that the final IFML model is fully generated in automatic and it acts as the basis for the navigation design reasoning performed by interaction designers. Clearly, the generated IFML model can be refined by interaction designers according to additional business-, project- and organization-specific requirements. The main rationale for our approach is that interaction designers do not start their reasoning process from an empty IFML model, rather can work on an already populated IFML model that reflects and is tailored to the navigation patterns of the users base of the initial website w.

IV. Tool Support

The tool implementing our approach is based on a pipeline of Python scripts, which are available here: http://cs.gssi.infn.it/WANDM. The web analytics report is generated from a custom report exploiting the Google Analytics service. The Google Analytics platform allows reports to be exported into Comma Separated Values (CSV) files; those files are the only required input of WANDM . A web analytics report is then scanned row by row by a *Python* script that, with the use of the *NetworkX* library, builds the first navigation graph G_w. NetworkX has been used for (i) the graph manipulations and transformations and (ii) computing several measures and metrics on G_w. The graph refactoring stage is based on visualization and centrality node attributes and on the semantic analysis of page titles carried out by using the Python library for *Wordnet*. Then, the navigation tree extraction has been applied to G_w with the addition of custom contraints and variables; to solve the GG-hop formulation we used *Gurobi*[4], a commercial solver for this kind of problems. The mobile-oriented navigation tree is then transformed into an IFML model via a Python script implementing the depth-first search variant producing the XMI model describing the IFML model. The generated IFML model can be imported and visualized with the open-source IFML editor[5] based on *Eclipse*.

V. Evaluation

In this section we show how we applied WANDM on a real-world website with thousands of views per month. We chose the website of the Computer Science division of the Gran Sasso Science Institute[6] (GSSI), the institution of one of the authors of this paper. This decision is due to the fact that we know the kinds of potential users of the website (mainly researchers and PhD students), making us reasonably confident when assessing the navigation models that WANDM generates from it.

Since the GSSI website is monitored via the Google Analytics engine, we obtained its *web analytics report* by using the standard report export feature of Google Analytics. The report counts 930 rows[7]. The **web usage mining** stage produced the *navigation graph* G_w in Figure 11; this graph has 71 nodes and 236 arcs. Then, we proceeded with the **graph refactoring** stage, resulting in a refactored navigation graph consisting of a significantly lower number of nodes and arcs, that is 36 nodes and 145 arcs. During the **navigation tree extraction** stage we experimented the extraction with some combinations of *max-depth* and *max-degree* values in order to obtain a clean and satisfactory navigation tree. The final *mobile-oriented navigation tree* is shown in Figure 12. This tree has a max-degree value of 4 for the root, and 3 for the nodes in the first level of the tree, whereas the max-depth value is 2.

Finally, with the **navigation model generation** stage we obtained the *IFML model* shown in Figure 13. After a manual inspection of the produced IFML model we identified only one misplaced node: */2015-2016-courses/*, child of the */news/* node. This node is indeed a natural son of */phd-program/*.

[4]http://www.gurobi.com
[5]https://github.com/ifml
[6]http://cs.gssi.infn.it
[7]Date of the last performed export: 20th of November 2015.

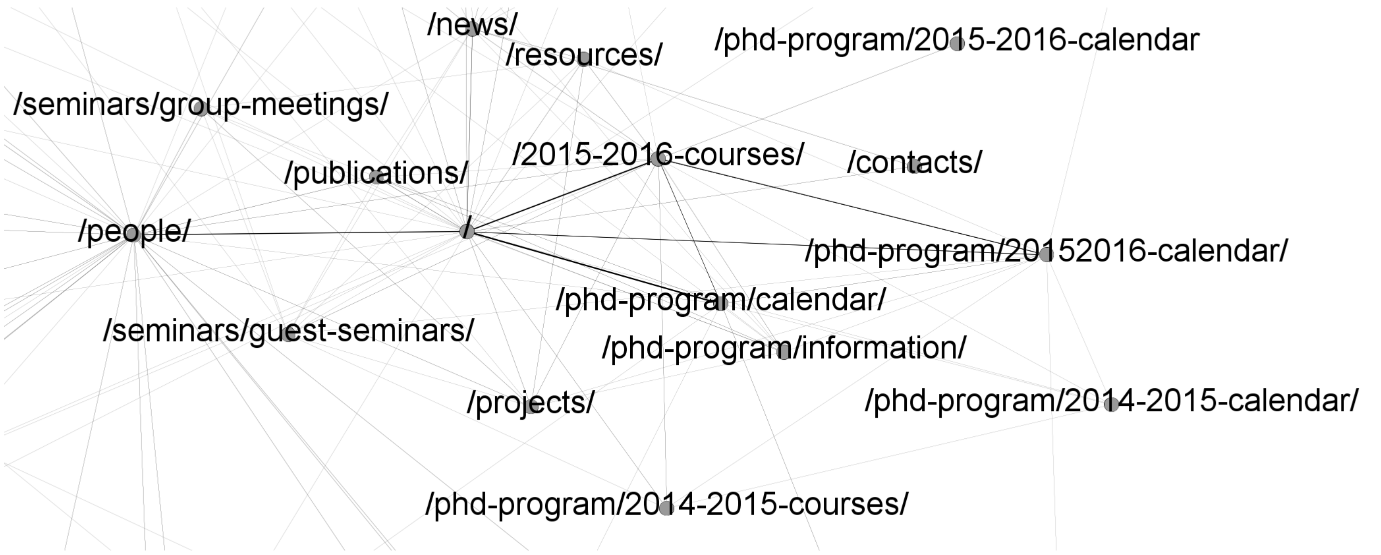

/news/
/resources/
/phd-program/2015-2016-calendar
/seminars/group-meetings/
/publications/
/2015-2016-courses/
/contacts/
/people/
/
/phd-program/20152016-calendar/
/phd-program/calendar/
/seminars/guest-seminars/
/phd-program/information/
/projects/
/phd-program/2014-2015-calendar/
/phd-program/2014-2015-courses/

Fig. 11. GSSI website navigation graph.

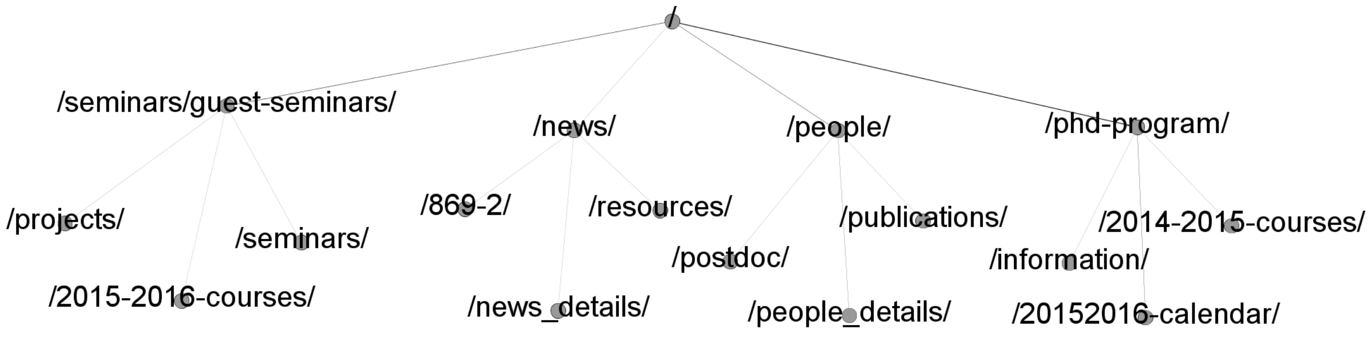


Fig. 12. GSSI website mobile-oriented tree.

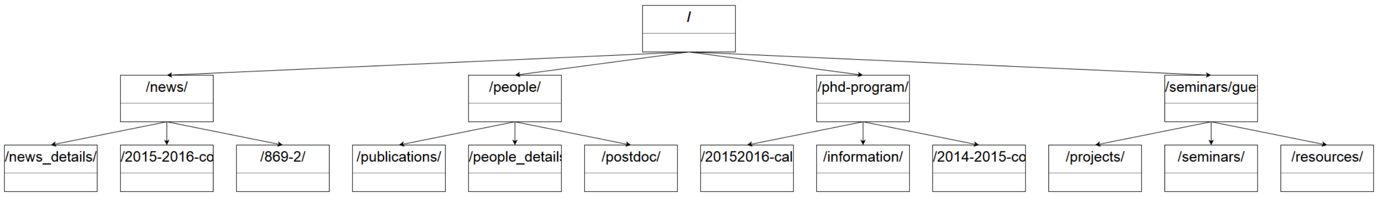


Fig. 13. GSSI website IFML model.

Anyway, this is an acceptable error because it is easy for interaction designers to fix a single navigation flow. It is also interesting to note that the obtained IFML model follows the same structure of the mobile-oriented navigation tree, the transformation to the IFML model has the purpose of providing to interaction designers an artifact to which they should be familiar already (i.e., an IFML model). Clearly, the structure of the obtained IFML model is totally different from the initial navigation graph because of the graph refactoring and navigation tree extraction (see Sections III-B and III-C).

In this paper we stop the evaluation of WANDM to the obtained IFML models because the GSSI mobile app has not been implemented yet, it will be interesting to compare the key navigation steps of the actual implementation of the mobile app with respect to the original website; this part of evaluation is left for future work.

VI. Related Work

We identified two main lines of research that are related to our approach: web usage mining and navigation modeling.

Web Usage Mining (WUM) is the application of data mining techniques to discover usage patterns from web data, in order to understand and better serve the needs of web-based applications. In [7] are described in details the three phases that compose WUM: *preprocessing*, *pattern discovery* and *pattern analysis*. The paper provides also a taxonomy of the work in the WUM area and an up-to-date survey of existing research on the topic. The authors of [12] present a review of the literature containing latest works done in the WUM field with the objective of providing an overview of WUM concepts relevant to the pattern mining phases of the WUM process. In [3] is presented an approach for automating the acquisition of user-interaction requirements in an incremental and reflective way; the proposed solution builds upon inferring a set of probabilistic Markov models of the users navigational behaviors, dynamically extracted from the interaction history. The inferred model is then analyzed to verify quantitative properties by means of probabilistic model checking.

For what concerns *navigation modeling*, [8] proposes to adopt the refactoring technique with the goal of reorganizing and improving the quality of navigation models. This technique was initially proposed to improve the structure of source code without changing its external observable behaviour. The authors adapt the refactoring technique to the navigation models context and present a catalogue of refactorings specific for this particular kind of models. The authors of [11] tackle the validation of the navigational requirements of a web application. A methodology named NRVA is introduced in the paper in order to validate the navigational requirements for correctness, consistency, and completeness. In [15] the concept of throughout-surfing patterns (TSPs) is introduced and then an efficient method for mining the patterns is presented. The research presented in [10] proposes an approach using statecharts to formally model adaptive navigation and show how some specific properties of a navigation model can be verified using existing model-checking tools.

Our proposal differs from other work in the literature because (i) it is fully automated in the sense that there is no need for having some already specified navigational model of the web site being considered; (ii) it build on well known and accepted techniques in the field of graph-based combinatorial optimization (e.g., the STPRBH problem formulation, the use of centrality measures, etc.); (iii) it is the first approach considering mobile-specific constraints, that can be used for fine tuning the navigation model generation; and (iv) it is the only approach generating a model conforming to a standard modeling language (i.e., IFML), allowing interaction designers to directly use the models without the need for long training sessions or advanced technical skills,

VII. Conclusions and Future Work

In this paper we presented an approach that exploits existing data coming from the web usage analytics of already developed websites for generating the mobile navigation model of their corresponding mobile apps. The navigation model of a mobile apps can be seen as the structure of a mobile app in terms of its views and the navigation flows among them. The whole approach is automatic and produces models conforming to the standard IFML language; this allows interaction designers to easily understand and manipulate them; generated models are specifically tailored to the navigational patterns of the users of the initial website.

Future work include (i) the refinement of the graph refactoring and navigation tree stages in order to produce more accurate and informative navigation graphs and tree, respectively, (ii) the execution of an experimental campaign on a series of real-world websites in order to better validate the proposed approach, and (iii) to investigate on the possibility of analysing also the HTML contents of each page of a website in order to generate complete IFML models containing also specific user events, user actions, data flows, etc.

References

[1] Adam Lella, Andrew Lipsman. The U.S. Mobile App Report, 2014. comsCore white paper.

[2] G. L. Alysson M. Costa, Jean-Franois Cordeau. Models and branch-and-cut algorithms for the steiner tree problem with revenues, budget and hop constraints. *Networks*, 53(2):141–159, 2008.

[3] M. S. G. T. Carlo Ghezzi, Mauro Pezz. Mining behavior models from user-intensive web applications. *ICSE*, pages 277–287, 2014.

[4] E. Estrada and N. Hatano. Communicability in complex networks. *Physical Review E*, 77(3):036111, 2008.

[5] B. Fling. *Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps*. O'Reilly Media, Inc., 2009.

[6] J. J. Garrett. *Elements of user experience, the: user-centered design for the web and beyond*. Pearson Education, 2010.

[7] M. D. P.-N. T. Jaideep Srivastava, Robert Cooley. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD*, 1(2):12–23, 2000.

[8] C. G. Jordi Cabot. A catalogue of refactorings for navigation models. *ICWE*, pages 75–85, 2008.

[9] P. F. Marco Brambilla, editor. *Interaction Flow Modeling Language*. The MK/OMG PRESS, 2015.

[10] C. H. Minmin Han. Modeling and verification of adaptive navigation in web applications. *ICWE*, pages 329–336, 2006.

[11] O. P. Pedro Valderas, Vicente Pelechano. A transformational approach to produce web application prototypes from a web requirements model. *IJWET*, 3(1), 2007.

[12] Z. S. A. Rosli Omar, Abu Osman Md Tap. Web usage mining: A review of recent works. *IEEE*, pages 1–5, 2014.

[13] University of Alabama at Birminghams Online Masters in Management Information Systems. The Future of Mobile Application, 2014. http://businessdegrees.uab.edu/resources/infographic/the-future-of-mobile-application/.

[14] A. I. Wasserman. Software Engineering Issues for Mobile Application Development. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, pages 397–400, 2010.

[15] A. J. L. Yao-Te Wang. Mining web navigation patterns with a path traversal graph. *ESA*, 38(6):7112–7122, 2011.