Detection of Errors in the Layout Design of Websites for Mobile Devices Based on Capturing User Behaviour

Lukáš Čegan and Petr Filip
University of Pardubice, Pardubice 530 02, Czech Republic
Email: lukas.cegan @upce.cz; petr.filip1@student.upce.cz;

Abstract — Smart mobile touch devices with a small screen such as mobile phones or tablets have become part of everyday life. These devices are taking over conventional personal computers and are increasingly used for accessing web-content. For website developers it is a new challenge which brings issues in the development strategy, because PCs are used differently than mobile touch devices. Adapting web design for every device has become a necessity and a responsive web-design has become standard for every newly developed website. Even if the website development is driven by best practices and developer`s experiences, issues that decrease user experience (UX) on the specific device still exist. This paper presents architecture for capturing user behaviour, followed by a statistical evaluation process which helps to detect errors in the layout design of websites. Information on website use is gathered from real users. Evaluated data is used as feedback for UX-designers who can improve web-design for specific devices and increase user experience.

Index Terms—User Experience, Analytical Tool, Web User Behaviour, HTML5

I. INTRODUCTION

According to the stats[1], the popularity of smart mobile touch devices like smart phones and tablets is growing year by year (see Fig. 1). In the past five years, the worldwide market share of using devices for web content accessing has totally changed. Due to lower prices, mobile devices became more available for more people and in 4Q2016, mobile devices began to dominate the worldwide market share and the trend is still continuous. Because the worldwide market share of these smart devices is still growing, we should pay more attention to them.

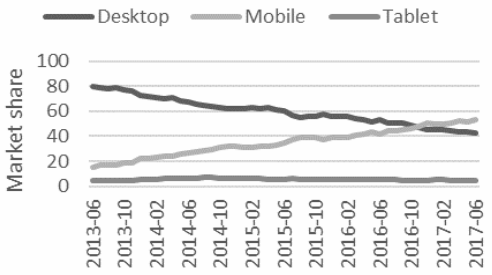

Fig. 1. Chart: Desktop vs Mobile vs Tablet Market Share Worldwide

The web content consumed on a mobile web browser is different from that on a desktop web browser. The main difference between these platforms is the size of the screen and the way of control. While a conventional desktop computer is controlled by a mouse and a keyboard, phones and tablets are controlled by touch gestures.

Nowadays, website development is not a job for only one person but for whole teams. These teams are mainly composed from roles like UX-designer, web-designer, programmer and server administrator. Together they create a chain, which covers the basic needs of modern website development.

Web-designers mostly collaborate with UX-designers and they have to face this situation of web design optimizing, which means updating page style and content for the best user experience on different devices. This is because users use devices with different screen resolution, screen size and web browser. All of these aspects can affect the final look of the web-page and that affects user behaviour and user experience on the device.

In the following list are three options that can be used for dealing with the difference between desktop and touch devices:

- Responsive web-design (RWD) — the key idea is to manage one website which can be used across multiple devices with different screen sizes. To achieve RWD, flexible grid, fluid images and media, and CSS3[2], media queries are used. The web content is adapted according to the rules specified in CSS files [1]. Many free CSS frameworks were developed to make web-development easier. These CSS frameworks are solving problems such as cross-browser incompatibility or RWD. In addition, they contain typography style definitions for HTML elements and CSS helpers classes for more advanced components [2]. For example, the well known CSS frameworks are Bootstrap[3] (created at Twitter) and MaterializeCSS[4]. Differences between them are mainly in the complexity. Some CSS frameworks are more lightweight and

[1] http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201306-201706

[2] Cascading Style Sheets
[3] http://getbootstrap.com/
[4] http://materializecss.com/

it is therefore intended for simpler and smaller websites.

- Native mobile web — can be realized as another website on a sub-domain. An example of this implementation is Facebook which has a website running on http://facebook.com and a native mobile web is available on http://m.facebook.com. This solution can meet specific requirements of mobile devices such as backward compatibility with an older mobile browser, which does not support new HTML5[5] and CSS3 standard. Next is to save data bandwidth because images can have appropriate (lower) resolution, smaller CSS, and optimized JavaScript files.
- Native application — solution is very different in comparison with the previous options. Native application can access the device hardware via an API[6] of OS[7]. On the other hand, development of native application is related to a specific OS and different technologies can be used [3].

Page loading is another aspect of user experience [4]. In the case of mobile devices, consideration of device performance and network connection is needed, because users naturally prefer fast page loading with minimum data bandwidth. That is the reason why CSS files of frameworks are minified and placed to the CDNs[8]. It helps to reduce network latency [5-6].

Page loading is not only about page size but also about code structure of HTML file. Even small change such as placing JavaScript code at the end of the source code file will improve page loading [7].

II. Study Background

This study attempts to explore some of the information that can be gathered by tracking user behaviour and evaluating it. User behaviour tracking can make the UX-designer aware of how, in fact, the mobile web is used. Evaluating of gathered data can point out barriers, bottlenecks and problems. It can also show unexpected user behaviour like content zooming, scrolling or device rotating, which decreases user experience because consuming of the web content must be with minimum effort and without barriers.

For gestures such as click, move and scroll, only one finger (and one hand) is needed. However, during zoom (pinch-to-zoom) action, the user is forced to use two fingers, which means using both hands. This can be understood as increased effort. Content zooming can indicate problems such as low contrast between font and background, small font size, or badly designed elements.

Fig. 2 shows the difference between user behaviour on non-responsive and responsive websites. In the case of responsive web design, the user immediately knows the website layout because common practice is placing the navigation collapsible bar on the top and content underneath.

In the case of non-responsive web design, navigation could be on the top, left or right side. It needs focusing on a thumbnail element and zooming to it. After that, the user does not find the required navigation link and needs additional zoom or move action. This could be followed by a click on the navigation link.

Bad user experience is also presented in Fig. 3 ,which shows different user experience during reading an article using both non-responsive and responsive web designs. The responsive web design is much easier to use for reading on small touch screens [8]. Also, the sum of gestures for non-responsive design is higher than for responsive design.

Another kind of problem is using disabled gestures such as zoom on responsive websites. Typically, it may be images or tables which sometimes the user wants to explore in more detail. Currently available tracking tools cannot detect this problem, and UX-designers cannot reflect the user's needs.

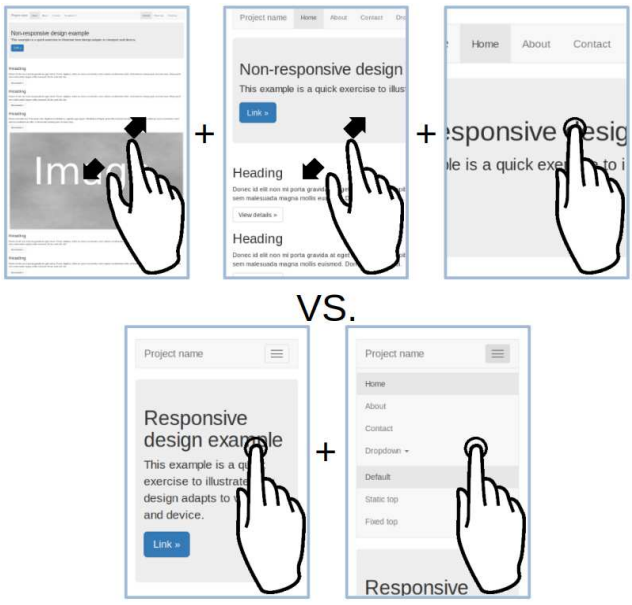

Fig. 2. Non-responsive web design vs. responsive web design: navigation on website

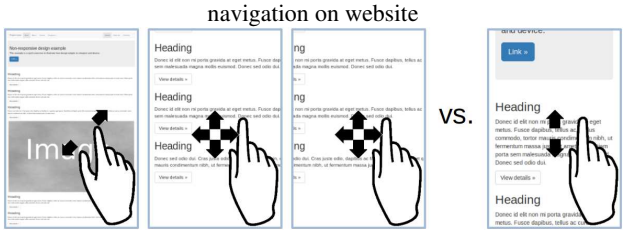

Fig. 3. Non-responsive web design vs. responsive web design: article reading

[5] HyperText Markup Language
[6] Application Programming Interface
[7] Operating System
[8] Content Delivery Network

III. Related Work

A. User Experience

Problem exploring can be generalized. Based on this information, tools and techniques for automated web design adaption can be developed.

One of the tools which helps to make a better user experience is W3Touch. This tool uses a missed links ratio (how often touches miss an intended target to register actions that were intended, but did not get triggered) and zoom level as metrics. From this information, it is possible to make rules which adapt the web interface [9].

Another UX issue is related to cross-browser compatibilities. Web standards such as HTML5 and CSS3 introduce new capabilities, which are not supported in some web-browsers. For sure, these new capabilities could be implemented differently by developers or companies. Both circumstances could lead to different layout representation. This problem occurred more frequently in the past and web-developers had to make many hacks to display the web-pages correctly on the most used web-browsers.

The study aimed at solving this issue by introducing an alignment graph (AG) which is created from DOM and page layout. The AG is used to represent two kinds of relationships between the elements of the layout of a web page. Evaluation is based on a comparison of two AG [10].

Consideration of a different web-browser is not enough, and RWD has to be in mind. Upon concepts associated with the AG was made a study that introduces Responsive Layout Graph and a method that can automatically detect potential layout faults in responsively designed web sites [11].

Another study applies techniques from the field of image processing to analyze the visual representation of a web page, identify presentation failures, and then determine which elements in the HTML source of the page could be responsible for the observed failures [12]. As already mentioned, modern CSS frameworks solve this issue, but more research? is still needed.

Another research is focused on prioritizing the content most relevant to user's preferences. In this work the design and implementation of KLOTSKI is presented — a practical dynamic reprioritization layer that delivers a better user experience [13].

Eye tracking is another practice which can help increase user experience. This technique allows to record places on the web-page where the user is looking [14]. Variety of solutions exist (many of them commercial) but all suffer from one or more of the following:

- high cost,
- custom or invasive hardware
- inaccuracy under real-world conditions.

One of the research from academic environment which tried to overcome this problem is iTracker, a deep convolutional neural network for predicting gaze [14].

It is only a matter of time when a web-camera or a mobile-front-camera will be used as a common part of a device to control such touch gestures.

B. Platform for Remote User Tracking

For the purpose of user tracking, a lot of platforms are available. One of the best known is Google Analytics[9] (GA), which is focused on business needs such as conversion ratios. These ratios are highly connected to advertisement evaluation. Very similar but paid and very advanced tool for tracking users is Kissmetrics[10]. This software provides easy and powerful user interface and allows to create custom metrics easily.

The best known open source representatives from the category of analytical software are Piwik[11] and Open Web Analytics[12] (OWA). Both of them are based on

- PHP[13] — scripting language for web development,
- MySQL[14] — database system,
- JavaScript.

Due to that it is possible to install them on their own server and use it without limitation of the service provider.

None of them provide fully detailed information about users' steps. The granularity is on a page request level, which could be insufficient for the UX-designer since more detailed information is needed. Even though gathered information can provide useful information for the UX-designer, detailed data cannot be analyzed because page layout information is missing.

Unamo[15] is a software for web analysis providing a functionality feature such as session replay that allows users steps to be seen, even on touch devices. But this tool does not provide any automated evaluation, only an individual session replay that could be time-consuming. A similar software is Smartlook[16] ,which provides additional session recording offer heatmaps of clicking and scrolling.

The next kind of UX analytical tools is based on surveys. The UX-designer specifies scenarios, defines representative person profiles, and then a website is served to the tester (survey participant). An example of a system like this is TryMyUI[17].

IV. User Tracking System Architecture

Fig. 4 shows a proposed tracking system architecture with dominant data flow processing. Architecture is divided into the following parts:

[9] https://www.google.com/analytics
[10] https://www.kissmetrics.com/
[11] https://piwik.org
[12] http://www.openwebanalytics.com
[13] http://php.net/
[14] https://www.mysql.com/
[15] https://unamo.com/
[16] https://www.smartlook.com/
[17] http://www.trymyui.com

- Web-client — is represented as the user who uses touch mobile device for web browsing.
- Web-server — contains the whole website source code, and provides web-page when the web-client sends request for page and resources.
- Tracking library — provides JavaScript tracking code and receives all data gathered from client web browser and sends to Tracker repository.
- Tracker repository — keeps data for evaluation systems.
- Evaluator — processes data from Tracker repository. Results are used as feedback for UX-designers.

A. *Tracking Implementation*

For evaluation purposes, it is crucial to obtain device identification which is represented as a device fingerprint. The fingerprint can be computed from device properties like user agent, screen resolution, color depth, etc. Due to computing from many device properties, each fingerprint should be unique but depend on the algorithm used. For lab purposes, Fingerprintjs[18] is used.

Tracking library is JavaScript based. For tracking user action, the method addEventListener()[19] is used, which waits for a user action that invokes a web-browser event. Examples of accepted event types[20] are "touchstart", "touchend", "orientationchange", etc.

Explanation for Fig. 4 (numbers in the following paragraph are related to the figure): Firstly, web-page loading from web-server (1) is needed and then tracking library code (2) is loaded. When a listened web-browser event is invoked, then it is processed by an event specific method. Data is then sent to the Tracker repository (3) where it is saved to the directory whose name corresponds to device fingerprints. Every event is saved separately with a unique identifier to this directory. After that, the Evaluator loads data from the Tracker repository and uses evaluation algorithms (4).

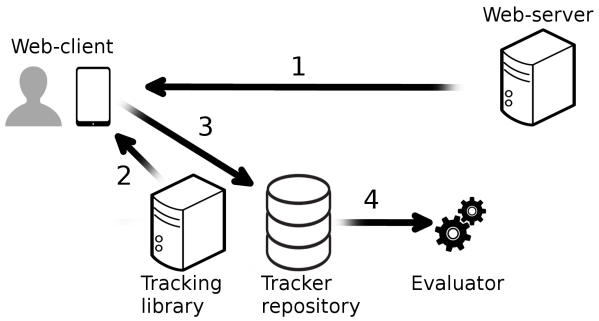

Fig. 4. Tracking system architecture

B. *Captured Data*

Captured data is stored to a JavaScript object which is composed of

- meta-data,
- event specific data,
- view-port snapshot,
- whole page snapshot.

Triggered events are based on user action

- on page loaded — is sent information about user device and about loading performance,
- on touch gesture — it includes touch gestures as zooming, scrolling and tapping,
- on changed device orientation (portrait, landscape).

Every captured event contains different kinds of data. For example, during device orientation change, it is useful to get current coordinates of view-port and zoom level, but in the case of loading performance information, it is not necessary to gather zoom level.

For data evaluation it is necessary to attach basic meta-data such as

- device fingerprint,
- time stamp,
- website identifier,
- current page,
- current page parameters,
- event type — on page load, zoom, tap, device orientation change,
- message number — ordinal number of captured action.

In order to relate captured user action, it is necessary to make a screen-shot of the view-port and the whole rendered page.

Elements which are fully in view-port height are considered, as is shown in Fig. 5.

A more accurate and more complicated solution is to find out how much area of element is in view-port, but it may decrease web performance (dependent on the algorithm).

Information about visible elements in view-port is sent as a collection of full path in DOM[21] which is used in the evaluation process. Example of full path in DOM of visible element in view-port:

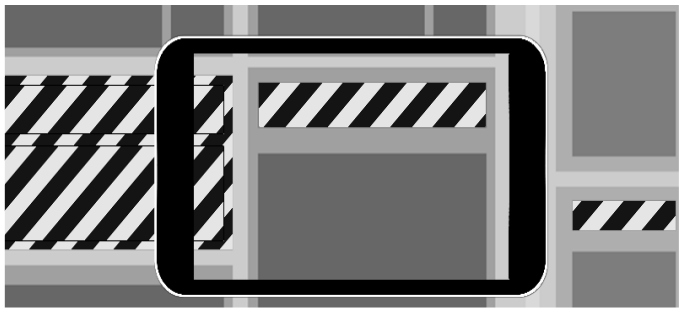

Fig. 5. Elements that are considered for view-port evaluation

[18] https://github.com/Valve/fingerprintjs2

[19] https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener

[20] https://developer.mozilla.org/en-US/docs/Web/Events

[21] Document Object Model

```
body>div#box>table#myTable>tbody>tr:nth-child(2)>td:nth-child(0)
```

The reason why element Id's are not used, is that not every element has one. This way of proposed view-port data representation is inconvenient because of the transfer of large amounts of data, but it is very accurate. To reduce data volume, it is possible to use a data compression library.

Almost similar, are represented DOM elements of a page snapshot, but x and y coordinates, width, height and element background color is added. This information suffices for reverse rendering of a rough page skeleton, which can be seen in Fig. 5. For better network performance, sending a page snapshot can be considered, only if the DOM is changed.

Another way how to implement capturing of visible elements is to use hardware or software emulation. Limitation of both solutions is that it is possible to render only pages without any interaction, and only those which are publicly accessible.

C. *Sending Data*

Communication between Web-client and Tracker repository is provided by XMLHttpRequest[22] API, due to the possibility of sending gathered data. Example of how data is sent via HTTP[23] POST method to the Tracker repository.

```
var c = new XMLHttpRequest();
c.open("POST", "http://srv.com/", true);
c.setRequestHeader("Content-Type",
"application/json");
var jsonData = JSON.stringify(jsObject);
c.send(jsonData);
```

Code example shown sending data in JSON[24] format.

V. EVALUATION

The evaluation of measured data is based on statistical processing using the Pearson's chi-squared test. In the first step, it is necessary to divide the captured data into groups according to the size of the device screen. According to the standard sizes of displays, the following groups are proposed: 240 x 320, 320 x 480, 360 x 640, 480 x 800, 640 x 960, 750 x 1334, 1080 x 1920 (px). Data in each group will be assessed separately. In the second step, there is excluded data from each group which meets the following conditions:

(a) user zoom in content of a page,

(b) the page was oriented on landscape.

In the third step, each page

(i) was split to segments,

(j) according to the height of the group (Fig. 6).

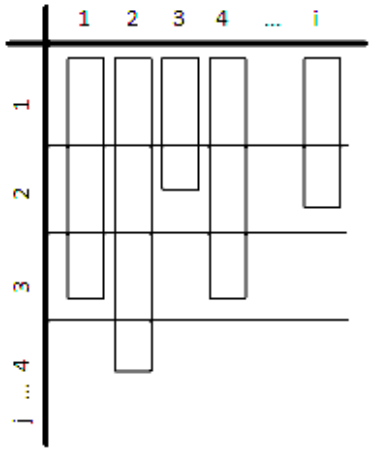

Fig. 6. Segments of web-pages

In the fourth step, there is a test to determine whether the occurrence of rotation in each segment is only a matter of coincidence, or whether the rotation between the segments is the occurrence of a systematic problem. For this reason, a null hypothesis H_0 is constructed and it is accepted or rejected by the Pearson's chi-squared test. An equation (2) is used to calculate the theoretical incidence of rotations, provided that (1).

$$H_0 : p_i = p_j = p_0 \tag{1}$$

$$\sigma_i = n_i \hat{p}_i \tag{2}$$

where:

n_i = total impressions i-th segment,

$\hat{p}_i$ = expected rotation probability of the i-th segment (3)

$$\hat{p}_i = \hat{p}_0 = \frac{\sum f_i}{N} \tag{3}$$

where:

N = total number of hits on all segments,

f_i = empirical frequency of rotation.

Subsequently it is necessary to test the hypothesis H_0 using (4)

$$X^2 = \sum_i^k \frac{(f_i - \sigma)^2}{\sigma_i} \tag{4}$$

where:

k = the total number of segments.

If (5) is valid, then the hypothesis is rejected. Otherwise, the hypothesis is accepted.

$$X^2 > X^2_{k-1}(\alpha) \tag{5}$$

To be able to determine in which segment the users rotate the screen due to errors in page layout, the Tukey's method can be used for multiple comparisons (6).

[22] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpR__equest

[23] HyperText Transfer Protocol

[24] JavaScript Object Notation

$$\left[\hat{p}_i - \hat{p}_j \right] \geq sq_{I,n-I}(\alpha)\sqrt{\frac{1}{2}(\frac{1}{n_i} + \frac{1}{n_j})} \qquad (6)$$

where:

s = standard deviation of $\hat{p}$,

$sq_{I,n-I}(\alpha)$ = critical value of studentizet distribution.

For better graphical representation of defective segments (Fig. 7), the color scheme is proposed where:

- extreme values are red,
- outlying values are orange,
- the rest values are yellow.

VI. Discussion

Unlike the related studies that are based on pages transformation to the graphs or to the images, which are followed by comparing them, a different approach was proposed based on a statistical evaluation which brings a new point of view. In the proposed approach the final web-page appearance or UX-designer intention is not important, but it is aimed at uncovering a bottleneck in the user-experience. Actually, the proposed evaluation method is limited only to the display rotation. Other research should be directed to combine multiple metrics such as pinch-to-zoom, page-scroll and touch moves to the one method. It is also possible to find other overlooked data from sensors such as gyroscope or accelerometer, which could be used for adaption of user interface.

VII. Limitation and Future Work

The possible limitation of the proposed method is that the badly designed elements are not exactly highlighted, only the segments are. The right combination of gathered information about visible elements in the viewport helps to identify the bad element. But every resolution can affect different elements. Experienced UX-developer should immediately recognize the badly designed element in the segment according to the evaluated data.

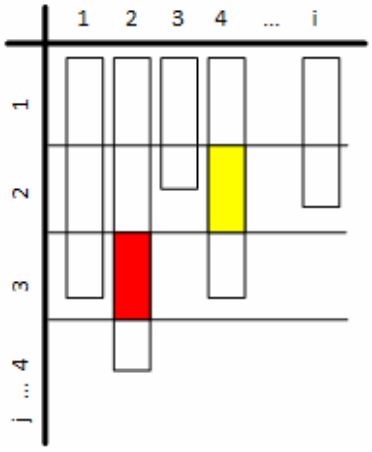

Fig. 7. Example of graphical representation of defective segments

One of the biggest technical issues is how to get a real screen-shot of layout from the device. As is described, the proposal is to get only coordinates, width and height of all elements and according to these attributes, create a page skeleton as a canvas image. For this purpose it is possible to use an existing JavaScript library named HTML2Canvas[25] which provides page rendering based on coordinates and properties of page elements. But techniques of reverse rendering may not be accurate, and that means pages may look different on a real mobile device than a reassembled page render. HTML2Canvas generates images that are not reasonable to transfer because of the image size. Because of that, transferring only coordinates of page element is considered. According to this information the page on the server is rendered.

For mobile users, it is reasonable that data bandwidth should be as low as possible. In the production version of the tracking application, considering appropriate data compression is needed.

The next issue is a fingerprint. If a fingerprint is not unique for all devices, it is not possible to distinguish them. This error affects the final evaluation and will influence its results. For evaluation based on the proposed method, it is not necessary to consider fingerprints.

Every user can have more than one device and according to the fingerprint, it is not possible to identify the user of the device, but only the device itself. This problem can be solved by adding extra information about the user (such as email) from the web application.

Gathered data can be used as a basis for other tools for automated updating of mobile user interface, such as the already mentioned W3Touch or web-page prefetching.

VIII. Conclusions

Smart touch mobile devices are having a bigger impact on everyday life. Nowadays these devices are used for consuming information on the internet more than ever. One of the biggest disciplines is how to provide perfect user experience on any device. That is the new challenge for web developers who have to deal with developing websites for heterogeneous devices, with different screen size, screen resolution, operating system and many more aspects. This paper described architecture for tracking user action such as gestures. Every user action is saved to a data repository and then provided for statistical evaluation. Results can notify the UX-designers of any badly designed element which is needed to update CSS files. Also, the evaluated data shows unexpected user behaviour on specific segment of web-pages.

Acknowledgment

This work is published thanks to the financial support of the Faculty of Electrical Engineering and Informatics,

[25] https://html2canvas.hertzen.com/

University of Pardubice under the grant SGS_2017_025 “Active Monitoring of Web Users Behaviour”.

REFERENCES

[1] C. Sharkie, A. Fisher, *Jump Start Responsive Web Design*, 2013

[2] N. Jain, “Review of different responsive CSS Front-End Frameworks,” In *Journal of Global Research in Computer Science,* 2014, vol. 5, no. 11, pp. 5-10

[3] W. Jobe, “Native Apps Vs. Mobile Web Apps,” *International Journal of Interactive Mobile Technologies* (iJIM), 2013, vol. 7, no. 4, p. 27, doi: 10.3991/ijim.v7i4.3226

[4] E. Bocchi, L. D. Cicco, D. Rossi, “Measuring the Quality of Experience of Web users,” In *ACM SIGCOMM Computer Communication Review*, 2016, vol. 46, no. 4, pp. 8-13, doi:10.1145/3027947.3027949

[5] M. Pathan, R. Buyya, A. Vakali, “Content delivery networks: State of the art, insights, and imperatives,” in *Content Delivery Networks*, 2008, pp. 3-32, doi: 10.1007/978-3-540-77887-5_1

[6] M. Wang, et al., “An overview of Cloud based Content Delivery Networks: Research Dimensions and state-of-the-art,” In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XX: Special Issue on Advanced Techniques for Big Data Management*, 2015, pp. 131-158, doi: 10.1007/978-3-662-46703-9_6

[7] W. Li, et al., “Reordering Webpage Objects for Optimizing Quality-of-Experience,” In *IEEE Access*, 2017, vol. 5, pp. 6626-6635, doi: 10.1109/ACCESS.2017.2689002

[8] N. Yu, J. Kong, “User experience with web browsing on small screens: Experimental investigations of mobile-page interface design and homepage design for news websites,” *Informatics Science,* February 2016, vol. 330, pp. 427-443, doi: 10.1016/j.ins.2015.06.004

[9] M. Nebeling, M. Speicher, M. Norrie, “W3touch: metrics-based web page adaptation for touch,” In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, Paris, 2013, pp. 2311–2320, doi: 10.1145/2470654.2481319

[10] R. S. Choudary, M. R. Prasad, A. Orso, “X-PERT: accurate identification of cross-browser issues in web applications,” In *Proceedings of the 2013 International Conference on Software Engineering*, 2013, IEEE Press, pp. 702–711, doi: 10.1109/ICSE.2013.6606616

[11] T. A. Walsh, P. McMinn, G. M. Kapfhammer, “Automatic Detection of Potential Layout Faults Following Changes to Responsive Web Pages (N),” In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 709-714, doi: 10.1109/ASE.2015.31¨

[12] S. Mahajan, W. G. J. Halfond, “Finding HTML Presentation Failures Using Image Comparison Techniques,” In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, September 2014, pp. 91-96 doi: 10.1145/2642937.2642966

[13] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, V. Sekar, “Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices,” In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’15)*, 2015, CA: USENIX Association, isbn: 978-1-931971-218, pp. 439-453

[14] S. Djamasbi, “Eye Tracking and Web Experience,” *AIS Transactions on Human-Computer Interaction*, 2014, vol. 6, no. 2, pp. 16-31.

[15] K. Krafka, et al., “Eye tracking for everyone,” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2176-2184, doi: 10.1109/CVPR.2016.239

Lukáš Čegan received his M.Sc. and Ph.D. from the Czech Technical University in Prague. He is currently a vice-dean for international development and external relations at the University of Pardubice. He is a member of the research team engaged in the development of mobile navigation systems in transport area. His main research areas are enterprise system integration and development of middleware applications.

Petr Filip received his B.Sc. from Jan Evangelista Purkyně University in Ústí nad Labem. Then he received M.Sc. from the University of Pardubice where he is currently studying a Ph.D. program. His dissertation work is aimed at performance and user experience of web applications.