# Rapid prototyping of business information systems based on Django framework and Kroki mockup tool

Milorad Filipović*, Tihomir Turzai**, Arpad Fakaš**, Gordana Milosavljević*, Amel Abdo*
* Faculty of Technical Sciences, Novi Sad, Serbia
** Alas D.O.O, Novi Sad, Serbia
mfili@uns.ac.rs, turzait@gmail.com, farkas.arpad91@gmail.com, grist@uns.ac.rs, abdo.amal4@gmail.com

***Abstract*— This paper presents an overview of the integrated framework that provides Python code generation based on the visual specification provided by Kroki mockup tool. Kroki enables end-users to create a specification of the developed information system by sketching a user interface and execute a working prototype based on that specification within seconds. Working prototype evaluation is extremely useful for identifying potential architectural errors that are the result of miscommunication between the parties involved in development, so having a functional prototype available enables such flaws to be corrected on-site, before the projects advance to the implementation phase. In its first iterations, Kroki mockup tool provided adaptive frameworks that generated working Java applications based on the user specification. This paper presents an additional Kroki framework that is used to generate Python web applications based on the Django framework. This additional module is introduced in order to provide more diversity among generated prototypes and in order to foster acceptance of modern web technologies such as HTML5, responsive web design, Twitter Bootstrap and such.**

## I. Introduction

The main motivation of the Kroki project[5][6] is to enhance software requirements elicitation process by narrowing the communication gap between the stakeholders and the development team[3][4]. The main goal is to provide participatory workflow which would allow customers and developers to develop negotiated information system specification within a few collaborative sessions. In the research conducted by the Standish Group[2] that included over 350 US based software companies, the data gathered from over 8000 development project showed that the main reasons for project failure are: the lack of user involvement, requirements incompleteness and changing requirements. The results clearly state that improvement of the requirements quality is crucial. Another research, documented in the Information System Manifesto by James Martin[1], that clearly displays the magnitude of specification flaws impact reports that 64% of errors originate in the software analysis and design phase, despite the fact that the client has formally signed the specification. The same author notes that 95% of project cost is spent on correcting those faults. According to the Agile manifesto, best evaluation is achieved if it is based on something that works. With all this in mind, Kroki tool provides a simple workflow that enables all parties, regardless of their technical skills, to create a specification of information system and generate a working prototype with the click of a button.

Implementation of the latest Django web framework allows developed prototypes to be up to date with the modern web technologies and gives users more realistic insight into look and feel of the final product. The main goal of this part of the research is to replace the old and somewhat outdated web prototyping framework with the one that suits modern web-based business information system requirements. Since the problem of integrating a new framework into Kroki tool has been addressed in our previous publications, the main focus of this research is on the Kroki Django module alone. In order to achieve this, the main body of knowledge and technological solutions has been gathered by implementing the previous Java frameworks, but some platform specificities and challenges that modern web technologies bring also arose and needed to be addressed.

## II. RELATED WORK

In order to implement a Kroki module for prototype execution, existing solutions in the field of rapid web application prototyping based on the user interface mockups are considered. Some of the notable solutions have been summarized here while more comprehensive overview is reported in the full paper.

FileMaker[9] is a professional tool for business application development that has been available for over 30 years. The tool is able to generate a working business application based on the database model and user interface specification. The tool sports a very attractive design and a myriad of predefined application templates in order to make development as easy as possible. Unlike the proposed solution, FileMaker is a commercial tool and it is not intended to be used as a prototyping tool.

SOLoist[10] is Java-based model-driven development framework for rapid business application development. All model constructs and application elements are compliant to the underlying UML profile, called OOIS UML. Currently, it is the only available solution that is based on the UML profile or domain specific language that we have come across, beside KROKI tool. The main difference between the SOLoist and KROKI is that it doesn't fully embrace agile development practices in order to foster end-user participation.

All of the considered solutions can be organized into two main groups. The first are tools used for business application development which aim at automation of the development process by generating as much programming code as possible and leaving the custom logic to be programmed manually. Other tools are used for software prototyping and mainly were able to generate just the basic user interface skeleton without any functionalities. Django module of the Kroki tools is designed to provide a hybrid solution that would be able to produce a functional three-tier Python web information system that would serve for hands-on evaluation of the final product in the early phases.

## III. Kroki Tool

Kroki tool[5][6] provides a streamlined participatory workflow which encourages client involvement in the design phase of software development and tend to reduce time and cost of the overall project. In order to provide more realistic insight into developed project appearance and functionalities, users must be presented with the prototype that suits their expectations and the final product as much as possible. With that in mind, executable prototypes and their corresponding frameworks need to be updated regularly according to current industry trends and standards. Kroki mockup editor can be seen on Figure 1, while the built-in UML editor is displayed on Figure 2.

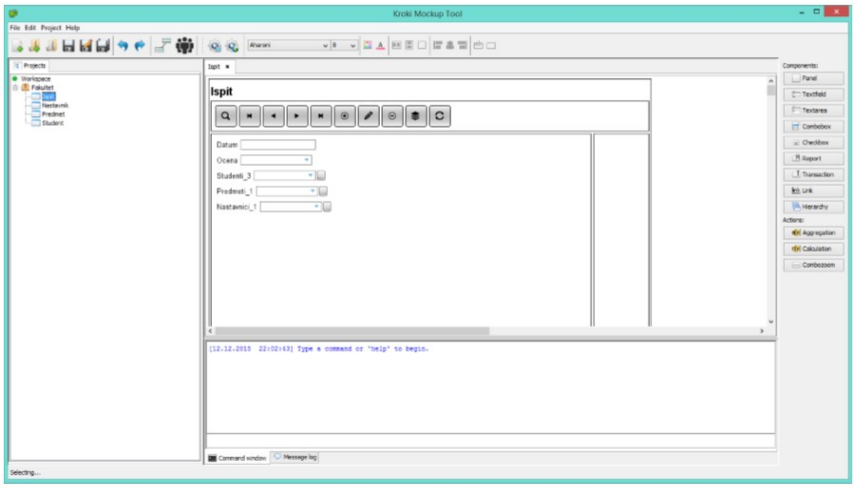
With this in mind, a new adaptive module based on the current state-of-the-art technologies has been developed which further enhances the quality of the prototypes generated from the Kroki specification. The resulting platform for the generated applications is Python programming language and Django web framework which are mature and affirmed technologies for web development. On top of that, a modern responsive web design guidelines have been fostered using the Twitter Bootstrap 3[12] framework and custom responsive jQuery modules. The major benefits of using Django as an underlying engine are its wide acceptance, maturity and widespread community which makes this technology current industry standard. Django provides built-in database management features such as synchronization and migration tools and integrated object-relational mapping which, coupled with the framework extensibility temple engine and HTML forms framework, makes it a great foundation for modern data-driven web applications.

Besides modernizing the Kroki web prototype frameworks, a step has been made towards a more flexible extension of supported languages and platforms for prototype execution. In order to provide the transformation from Kroki generated XML files to target Django project a parser and adapter modules have been developed. This modularization greatly decouples mockup specification modules from the execution frameworks and enables the development of custom generators if the need for additional platforms and language support arises in the future.

## IV. Django framework

Django[11] is a Python web framework aimed at providing a simplified and streamlined experience in modern web applications development. In order to provide this, the framework is based on MVC (Model-View-Controller) design pattern and the ORM (Object-Relational Mapping) techniques. All application entities are described in the *models.py* file, which serves as a basis for database generation. Mapping is executed by running Django *migrat*e task which updates the database according to contents of models file. At any moment, an administration web dashboard is available that provides basic manipulation over the generated entities.

Django URL mapping is specified in the *urls.py* file which maps URL patterns to specific python methods, called views. URL patterns are provided as regular expressions and the frameworks maps the first matched pattern to a corresponding view, or sends a 404 HTTP error code if no match is found. If the matching view has been invoked, its task is to process the client request,



```
[12.12.2015  22:02:43] Type a command or 'help' to begin.
```

Figure 1. Main Kroki window

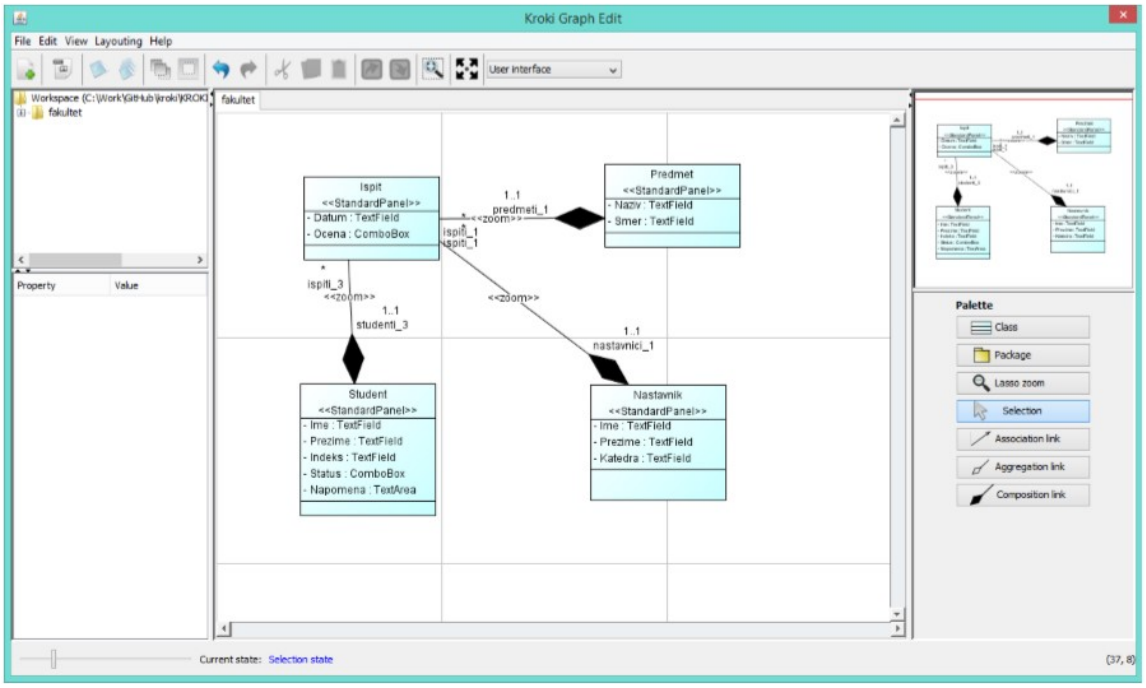


Figure 2. Kroki UML editor

execute specified logic and return

HTTPResponse object to client. Data in this object can be integrated into a file template which can be used to provide a pre-prepared HTML pages. In order to provide this, Django uses its built-in template engine. Beside the basic processing and rendering functions, Django supports template inheritance which greatly fosters template modularity and reusability. This feature makes Django especially suitable for code generation. Django also implements its own HTML forms framework, called *Django Forms*, which enables implementation of a special python controllers that handle HTML forms server and client side functionalities.

## V. Kroki generator extensions

In order to make Kroki generator modules more generic, a whole new abstraction layer has been implemented to support code generation for multiple platforms. The Django generator presented in this paper serves as a proof of this concept. Kroki configuration XML files now serve as an imput for generator unit that can be configured to generate code for multiple platforms. An overview of this solution is presented in Figure 3. The first module in the system is XML Parser which transforms Kroki XML files into format suitable for subsequent processing. Kroki XML files are based on EUIS DSL[7] specification, which means that they specify menu structure, application forms,form relationships and form elements and operations descriptions. The main goal of the parser is to provide all required information so the prototype can be generated using arbitrary generators for various platforms. The idea here is to decouple the input and output modules in order to enable the adapter and generator module to be implemented in different programming languages and platforms that the rest of the system.

Parsed data is then fed to adapter module which is platform-specific, so in this solution a Django adapter is implemented and used. Adapters are used to make all the necessary transformations so the parsed data can suit the provided generator unit. The generation is initiated only if the Adapter has successfully processed the data. Generators, as final units in the system generate source code and configuration files for the generated application. The generator first removes the files from the previous generations and then generates the standard Django

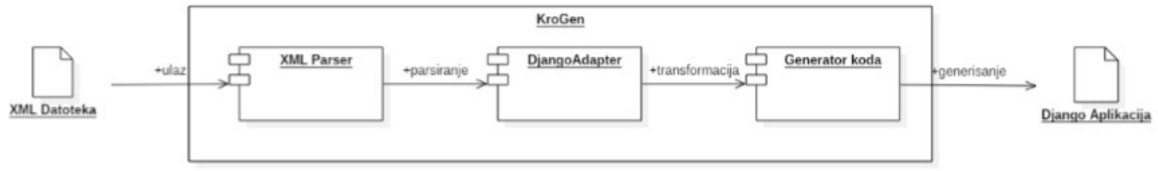


Figure 3. Kroki Django generator modules

directory structure, shown in the Table 1. After the directory structure has been constructed, Django models, forms and url mapping files are then generated. The last part of the prototype that has is generated are HTML templates and static files. For each Kroki standard panel, Django generator generates three HTML pages that provide view, add, and edit functionalities,

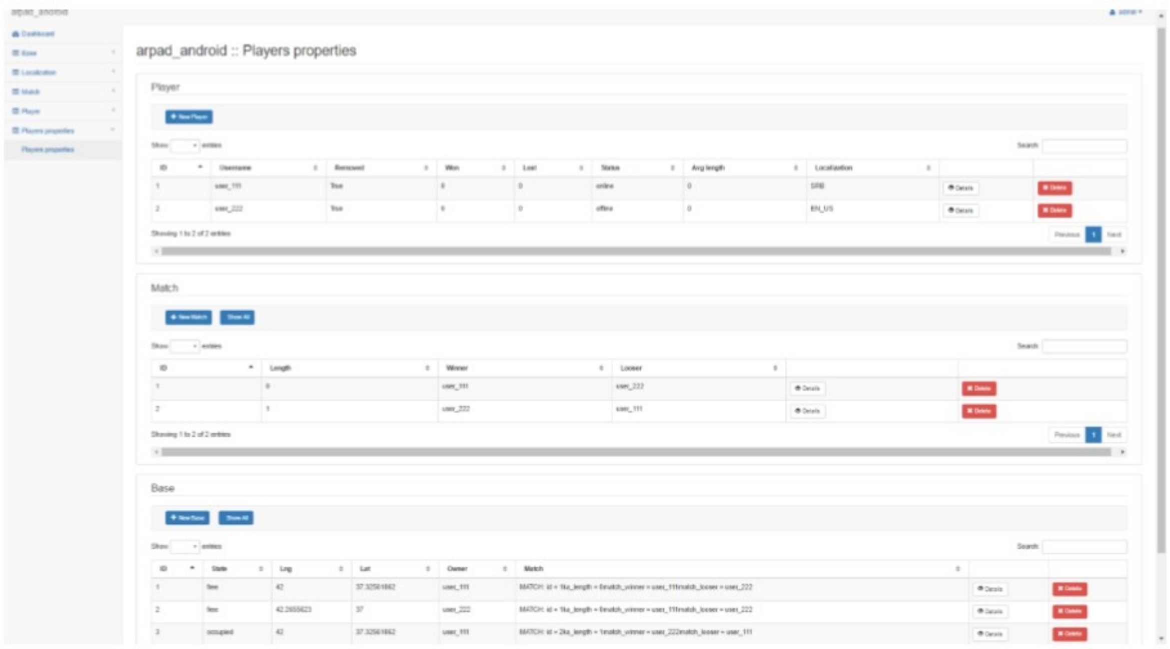
| Directory / File | Description |
|---|---|
| **[projectName**] | Main application module |
| **module** | Project-specific data |
| **static** | Static files |
| manage.py | Django administration tool |
| **[projectName]**/urls.py | Static and generated URL mappings |
| **[projectName]**/settings.py | Main project settings. |
| **[projectName]**/init.py | Database population procedure. |
| **module**/templates | Django HTML templates |
| **module**/templatetags | Additional filters. |
| **[projectName]**/views.py | Django views. |
| **[projectName]**/forms.py | Django forms. |
| **[projectName]**/models.py | Entity models and enumerations. |
| **src** | Java and Groovy sources used in application. |

This way, Kroki is no longer limited to generation of Java prototypes, but can be extended to provide a platform independent information systems.

## A. *Parent-child panels*

As defined in EUIS DSL, Kroki supports specification of parent-child panels which are used to display data from multiple entities on a single page or form. The entities presented in the paren-child panel have to be in hierarchial relationship, and panels have to provide automatic filtering of child entries once the parent entry has been selected. In order to support this, Django generator provides a module that generates HTML pages with parent-child forms that is based on Freemarker and Django templates. First, a set of Freemarker templates is generated based on the parsed data from Kroki XML files. Freemarker templates are then used to produce Django templates which in turn generate final HTML pages. Both Freemarker and Django template engines feature template inheritance which greatly improves code reusability. According to this, a root Django template, called *base.html*, is developed which serves as a basis for all other pages in the application. Base template is not generated, since it is not project-specific, and all other templates inherit it. Besides the overall layout definition, base tamplate includes the navigation template which is generated since it contents depend on actual project that is being developed. An example of generated HTML page with parent-child panel is shown in Figure 4.

Django templates for parent-child forms are generated using the Freemarker template, called *parentChild.ftl*. All Django parent-child templates extend base.html template and define the parent-child panel layout and dependencies.



arpad_android :: Players properties

Player

Show entries Search:

| ID | Username | Removed | Won | Lost | Status | Avg length | Localization | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | user_111 | True | 8 | 0 | online | 0 | SRB | Details | Delete |
| 2 | user_222 | True | 8 | 0 | offline | 0 | EN_US | Details | Delete |

Showing 1 to 2 of 2 entries

Match

Show entries Search:

| ID | Length | Winner | Looser | | |
|---|---|---|---|---|---|
| 1 | 8 | user_111 | user_222 | Details | Delete |
| 2 | 1 | user_222 | user_111 | Details | Delete |

Showing 1 to 2 of 2 entries

Base

Show entries Search:

| ID | State | Lng | Lat | Owner | Match | | |
|---|---|---|---|---|---|---|---|
| 1 | free | 42 | 37.32561862 | user_111 | MATCH: id = 1ka_length = 8match_winner = user_111match_looser = user_222 | Details | Delete |
| 2 | free | 42.2655623 | 37 | user_222 | MATCH: id = 1ka_length = 8match_winner = user_111match_looser = user_222 | Details | Delete |
| 3 | occupied | 42 | 37.32561862 | user_111 | MATCH: id = 2ka_length = 1match_winner = user_222match_looser = user_111 | Details | Delete |

Figure 4. Parent-child panel in generated web application

### B. *Providing responsive HTML pages*

In order to feature a modern and responsive look and feel, jQuery and Bootstrap are used on all generated pages. According to this, all templates are designed with the responsive nature in mind. Event though, the data heavy applications, such as business information systems are not suitable to be presented on mobile screens, a lot of effort has been invested in an attempt to provide a pleasing user experience on various screen sizes and orientations. Figure 5 shows HTML page with main navigation and one panel displayed on an Android mobile device in portrait mode.

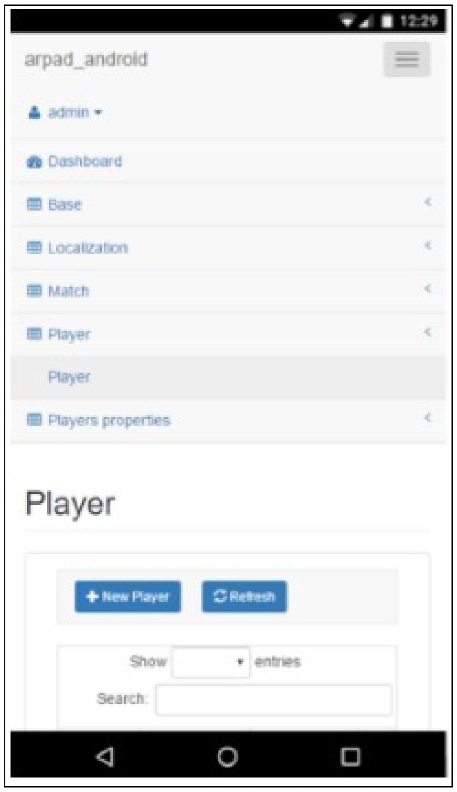


Figure 6. HTML page displayed on mobile device

### C. *Dashboard and search*

Two main things found lacking when evaluating old Kroki prototypes is the search functionality and administrative page. In order to satisfy these requirements, all prototypes generated with the Django generator feature a HTML dashboard that is used to administer the data and provides various statistic information about the system. Current version of the dashboard is displaying only the basic info in the textual form (such as number of entities and entries in the database) but can be enhanced with interactive data such as graphs and maps.

All forms have also been extended with the search functionality. In the previous versions of Kroki prototype, the search functionality is not considered to be an essential part of the prototyping phase, but further evaluation with the clients proved that enterprise users rely heavily on this feature on their day-to-day work. Based on that conclusions, all forms generated by the Django generator feature a search form. In order to satisfy a modern web application development guidelines, the search is fully implement on client side using javascript.

## VI. Conclusions

This paper presented an overview of implementation of generator extension that is used to generate Python Django web applications from the specification made using Kroki tool. The main goal of this extension is to substitute previous Kroki prototyping generators based on somewhat outdated Java Restlet technology and to provide a foundation for plugable generator architecture that would support Multilanguage and multiplatform prototype generation. The implemented generator uses modern Django framework and produces the code that conforms to the proposed standards and best practices in Python and web development. Also, in order to provide a cutting edge user experience and modern look, client-side technologies such as Bootstrap 3 and jQuery javascript library have been utilized.

Further development of Kroki tool based on these improvements would involve incuding advanced and custom front-end features such as additional pop-up dialogs and form validation. Server side of the developed prototype could be also modernized by introduction of microservices.

## VII. References

[1] James Martin: *An Information Systems Manifesto*

[2] Standish Group *2015 Chaos Report*, https://www.infoq.com/articles/standish-chaos-2015

[3] *The Agile Manifesto*, www.agilemanifesto.org , Utah, 2001.

[4] A. Kleppe, J. Warmer, W. Bast, *MDA Explained – The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, 2003

[5] G. Milosavljevic, M. Filipovic, V. Marsenic, D. Pejakovic, I. Dejanovic, Kroki: *A mockup-based tool for participatory development of business applications*. SoMeT (p./pp. 235-242), : IEEE. ISBN: 978-1-4799-0419-8, 2013

[6] Kroki MDE Tool, http://www.kroki-mde.net

[7] B. Perišić, G. Milosavljević, I. Dejanović / B. Milosavljević, *UML Profile for Specifying User Interfaces of Business Applications*, Novi Sad, 2011.

[8] SOLoist Framework - Java Web Framework for Model-driven Development, http://www.soloist4uml.com/

[9] FileMaker Pro - http://www.filemaker.com/

[10] FreeMarker Java Template Engine, http://freemarker.org/

[11] Django web framework, https://www.djangoproject.com/

[12] Twitter Bootstrap, http://getbootstrap.com/

[13] A. Cockburn, J. Highsmith, *Agile Software Development: The Business Of Innovation*, IEEE Computer, pp. 120-122, Sept. 2001.

[14] Restlet framework, https://restlet.com/