



*Facultad
de
Ciencias*

**GENETICS STUDY, INTERACTIONS
MODELING, DEVELOPMENT AND
INTEGRATION OF BIOINFORMATICS
MODULES FOR THE GENETYSIS ®
GENETICS SOFTWARE**

(Estudio de base genética, modelado de interacciones y desarrollo e integración de módulos bioinformáticos para el Software genético Genetysis ®)

**Trabajo de Fin de Grado
para acceder al**

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Raúl Nozal González

Director: Domingo Gómez Pérez

Co-Director: Jose M^a Aznar Oviedo

Septiembre - 2015

Contents

List of Tables	III
List of Figures	III
Listings	IV
License	v
Acknowledgments	IX
Abstract / Resumen	XI
1. Introduction	1
1.1. Motivation	1
1.2. Objective	2
1.3. Internship planning	3
1.4. Report structure	4
2. State of the art	7
2.1. Matrix of combinations	7
2.2. Building system and web development	7
2.3. Phenotype Indicator	8
3. Methods, Materials and used Technologies	9
3.1. Genetics theoretical bases and applications	9
3.2. Software methodologies	10
3.3. General software technologies	10
3.4. Languages and Environments	11
3.5. Building system	12
3.6. Graphical User Interfaces	14
3.7. Web components	14
4. Contents & Results	16
4.1. Matrix of combinations to rules	16
4.2. Werft-projects (building system)	19
4.3. m-forms (Genetysis ® GUI)	27
4.3.1. i18n-components (internationalization)	32
4.3.2. Helpers	35
4.4. Phenotype Indicator	36
4.5. Foreign Function Interface (<i>FFI</i>)	39

5. Conclusions and future work	42
5.1. Conclusions	42
5.2. Future work	43
6. Bibliography	45
7. Appendices	49
7.1. Renders	49
7.1.1. <i>GUI</i> (m-forms)	49
7.2. UML	54
7.2.1. <i>GUI</i> (m-forms)	54
7.2.2. Internationalization	56
7.3. Code	57
7.3.1. Matrix of combinations to rules	57
7.3.2. <i>GUI</i> independent packages	63
7.3.3. <i>i18n-components</i>	74
7.3.4. Helpers	80
7.3.5. Phenotype Indicator	86
7.3.6. Optimizations and <i>Foreign Function Interface</i>	98

List of Tables

1.	Werft tasks, order and dependencies.	21
2.	Comparison between client and refinery development types (using the optimized version).	26

List of Figures

1.	Internships, projects distribution and planning.	3
2.	Contents relationships.	6
3.	Schemes of polymorphisms combinations (matrices to rules).	16
4.	Tree of dependencies and levels of abstraction.	17
5.	Portion of web interface (<code>m-forms</code>) showing the rules visualization and its values.	18
6.	Flowchart showing the stream processes in the <code>babel</code> task.	22
7.	Flowchart showing the stream processes and its subtasks in the <code>refinery</code> task (under <i>Refinery</i> mode).	23
8.	Comparative of tasks and its optimized loading (100 executions, build tasks).	25
9.	Comparative of development modes and its optimized loading (100 executions, build tasks).	26
10.	Some mockups designed and accepted in the first meetings.	27
11.	Overview of the <code>m-forms</code> components system.	28
12.	Overview of <code>content</code> components (<code>m-forms</code> nested level of components).	29
13.	Genetics data administration <i>GUI</i> architecture overview.	30
14.	<code>formCombinator</code> in a demonstration of two complex views (Polymorphisms and Combinations) and with mixed support of <code>il8n-components</code>	31
15.	<code>HelpComponent</code> connected with a set of three different components (<code>m-forms</code> , <code>field</code>).	32
16.	Types of translation mappings (portion of a schema definition).	33
17.	<code>il8n-components</code> Activity diagram (translation).	34
18.	<i>GUI</i> application (over <code>m-forms:field:file-input</code> , but interacting also with <code>m-tooltip</code> independent component and the <code>remove form</code> component).	35
19.	Helpers in <i>Emacs</i> . Left side in <i>Stylus-mode</i> modified. Right side in <i>JavaScript</i> using <code>nav-blocks</code> (from top to bottom with just one key shortcut).	36
20.	Phenotype Indicator screen, using the internal <code>template-phenotype</code> module.	37
21.	<code>template-phenotype</code> on-the-fly error detection, connected to a screen component to show the messages and with syntax transformation.	38
22.	<code>template-phenotype</code> special mode and hinting system. Autocompletion features and automatic keyword guessing.	38
23.	Comparatives between loop algorithms (imperative and functional).	40
24.	<i>Jeff's Greenberg Duff's Device</i> and <i>Nozal's Loop</i> implementation, respectively.	41

25.	<code>m-toastr</code> rendering process during a <i>Refinery</i> session.	49
26.	<code>m-forms</code> during a <i>Refinery</i> session, showing <code>formComponent</code> field (demonstration purposes).	50
27.	<code>m-forms</code> showing <code>formComponent</code> field, panel and container's title (demonstration purposes).	51
28.	<code>HelpBlock</code> showing its capture (static rendering of the component, testing purposes).	52
29.	<code>HelpBlock</code> showing its dynamic rendering (testing purposes).	53
30.	<code>m-forms</code> independent components and pluggable modules.	54
31.	<code>m-forms</code> system, <code>formComponent</code> components (containers and contents).	55
32.	<code>i18n-components</code> Activity diagram (usage overview).	56

Listings

1.	Polymorphism Rules (<code>polymorphism-rules.js</code>).	57
2.	<code>CStr</code> (<code>cstr.js</code>).	63
3.	<code>i18n-components</code> (<code>i18n-components.js</code>).	74
4.	<code>remove button</code> component <code>i18n</code> schema definition (<code>remove-lang.js</code>).	78
5.	<code>remove button</code> component <code>i18n</code> schema definition (<code>remove-lang.js</code>).	79
6.	<code>i18n-components</code> schema bootstrap with <code>yasnipet</code> (<code>lang.snippet</code>).	79
7.	<code>nav-blocks</code> extension (<code>nav-blocks.el</code>).	80
8.	<code>stylus-mode</code> modification (<code>stylus.el</code>).	85
9.	<code>template-phenotype Interpreter</code> (<code>interpreter.js</code>).	86
10.	<code>template-phenotype</code> hinting extension (<code>template-phenotype-hint.js</code>).	95
11.	<code>template-phenotype</code> mode adapter for <i>CodeMirror</i> (<code>template-phenotype-mode.js</code>).	96
12.	<i>Rust</i> "where is" program to locate dependencies (demonstration purposes <code>whereis.rs</code>).	99
13.	<i>JavaScript</i> Jeff Greenberg's <i>Duff's Device</i> (<code>duffs-device.js</code>).	101
14.	<i>JavaScript</i> Nozal's <i>Loop</i> implementation (<code>nozals-device.js</code>).	102

License

The report itself:

GENETICS STUDY, INTERACTIONS MODELING, DEVELOPMENT AND INTEGRATION OF BIOINFORMATICS MODULES FOR THE GENETYSIS ® GENETICS SOFTWARE (Estudio de base genética, modelado de interacciones y desarrollo e integración de módulos bioinformáticos para el Software genético Genetysis ®) by Raúl Nozal is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.



Source codes shown along the report:

GENETICS STUDY, INTERACTIONS MODELING, DEVELOPMENT AND INTEGRATION OF BIOINFORMATICS MODULES FOR THE GENETYSIS ® GENETICS SOFTWARE (Estudio de base genética, modelado de interacciones y desarrollo e integración de módulos bioinformáticos para el Software genético Genetysis ®)

Copyright (C) 2015 Raúl Nozal

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Figures, tables and source code snippets are originally made by the author of this report. All the diagrams, designs and images are under the same licensing as the report itself.

Note: the diagrams and images in most cases are in high-resolution and embedded in the document, being not really prepared for printing purposes due to the limitations of fixed size units (dots). The reader is encouraged to view them with *PDF* readers that support zoom and pan tools (like Okular), to be able to see properly all the details.

The comparatives and graphs are based on the loads, benchmarks and analysis done over the same workstation (computer), although its specifications are not relevant for this Master Thesis' purposes and its conclusions.

“ El hombre que se prepara, tiene media batalla ganada.

(The man who is prepared, it has half the battle won.)

Miguel de Cervantes, 1547 - 1616 ”

Acknowledgments

I would like to thank the people who helped me along the process of making this report, taught me new Science fields like Nutrigenomics and Genetics and complied with reviewing the work I have been involved in for the last year.

I would like to remember the Baigene Bioinformatics staff that led my internship, which provided enough help and information during the most difficult tasks and shared almost every possible knot of knowledge since the very first time I was between them.

Thanks to the whole Baigene enterprise, especially Adrian, Jose and David, addressed all my needs, gave me a comfortable place to work, were flexible about work-hours and deadlines and most importantly, allowed me to work from my home after the second month. A start-up but with great people and really good internal policies.

Special thanks to my director Dr. Domingo Gómez Pérez and my co-director Dr. José M^a Aznar Oviedo, who formally inspected the entire report, were always there to support my duties, carefully reviewed my week work and allowed me to contribute with even more personal solutions for every study we were doing in the bioinformatics and software development departments.

Along the Computer Engineering studies I have been quite happy with everything, not only materials and disciplines, but also with the professors. Thanks to every professor that taught me any lesson. Special thanks go to the professors that were pushing forward, who were learning and updating their knowledge, and taught the best they could. I would like to express my gratitude to Jose Luis Bosque, Mario Aldea Rivas, Michael González Harbour, Pablo Sánchez Barreiro, J. Javier Gutiérrez, Angel Vegas and J. Angel Gregorio Monasterio. Each one with its distinctive features.

Special thanks to my German brother, Peter Külbs, and to my friend María.

Finally, all the efforts, headaches and more than a thousand hours of energy spent in the whole internship that made it possible to fulfill these projects would not have been feasible without my relatives. The recipe for the success is my family, the conditions and environment that they provided me and the understanding of the requirements, variety of fields and amount of complex cognitive tasks.

Abstract

Nowadays, start-ups and big corporations are involved in complex business processes with a wide variety necessities and more often with urge of digital applications to satisfy their clients, suppliers and other companies. These needs and other fields with a demanding R&D component, like Genetics, are present in many businesses combined with specific and competitive tasks. These tasks and the software which is used to fulfill them require World-Wide availability and maximum flexibility to deal with the market and leading research.

The access to genetic computation processes has revolutionized the commercial possibilities of well defined sectors like Sports Nutrition, Nutrigenomics or Population Genetics. There is a long road ahead and several factors to define the whole development infrastructure, the workflow and even the architecture where all the scientific processes lay down. For several times, it has been necessary to redefine the genetic processing algorithms because of the low computational efficiency or even distributing the tasks being aware of the business model and providing an easy application and good usability.

The cross-disciplinary work in different research fields and the advantages of every Computer Engineering discipline contribute to the development of high technology business in a rapid changing market. Because of all that has been mentioned before, this Master Thesis contains several different contributions: analysis and improvement of the current polymorphisms processing, a building system for web projects and business-model-aware support system, software architecture for the client-side genetics-management web application (*front-end*) highlighting the *Graphical User Interface (GUI)*, a specific module where a language is defined (and its interpreter) to set the phenotype indicator based on templates, and finally, the study of tools to connect modules (*Foreign Function Interface, FFI*) between divergent languages and platforms aimed to improve the performance and reutilization of previous modules.

Keywords: Genetic Computation, Nutrigenomics, Population Genetics, web development building system, Graphical User Interface (GUI) architecture, Foreign Function Interface (FFI)

Resumen

Hoy en día desde las empresas más pequeñas hasta las corporaciones más grandes se ven envueltas en complejos procesos de negocio, con diversas necesidades y cada vez con una mayor necesidad de aplicativos digitales con los que satisfacer a clientes, proveedores u otras empresas. Si a estas empresas del sector tecnológico les unimos campos de la ciencia donde hay un fuerte componente de I+D+i, tales como la genética, se consigue un área de mercado con necesidades muy específicas y cambiantes, tales como: disponibilidad a nivel mundial y máxima flexibilidad puesto que existe una relación directa entre el mercado, la investigación y el software utilizado.

El acceso a procesos de computación genéticos, observados y entendidos desde el punto de vista comercial, y aplicado sobre sectores tan concretos como la Nutrición Deportiva, la Nutrigenómica o la Genética de Poblaciones hacen que sea necesario definir una nueva infraestructura de desarrollo y una arquitectura más acorde con esta nueva forma de trabajar. En muchas ocasiones es necesario redefinir algoritmos de procesamiento genético por su bajo rendimiento a nivel computacional o incluso distribuyendo los problemas teniendo en cuenta el modelo de negocio y facilitando su aplicación y utilización.

El trabajo multidisciplinar y la combinación de distintas áreas del conocimiento, unido a una necesidad comercial cambiante y sacándole el máximo partido a las distintas disciplinas de la Ingeniería Informática, contribuye en muchos sectores y se materializa en este trabajo de las siguientes maneras: un análisis y mejora del procesamiento actual de polimorfismos, un sistema de construcción de proyectos web y soporte al desarrollo según el modelo de negocio, una arquitectura software para el aplicativo web de administración de contenidos genéticos desde lado cliente (*front-end*) haciendo hincapié en la interfaz de usuario *GUI* y en un módulo concreto debido a la definición de un lenguaje (y su intérprete) para establecer el indicador de fenotipo basado en plantillas y por último, el estudio de herramientas para la conexión de módulos (*Foreign Function Interface, FFI*) entre lenguajes y plataformas de diversa índole con el objetivo de mejorar el rendimiento y la reutilización de código.

Palabras clave: Computación Genética, Nutrigenómica, Genética de Poblaciones, sistema de construcción para desarrollo web, arquitectura de interfaces gráficos (GUI), Foreign Function Interface (FFI)

1. Introduction

In this chapter it is described a brief introduction to the tasks and objectives presented during the internship in Baigene, S.L. and gathered by this Master Thesis report. Also, it is described the internship planning to understand the report inside a long-term set of projects, and the structure of the document to put in context the work done in this thesis.

1.1. Motivation

Since the appearance of the 90's dot-com companies and the first years of The Internet, the number of business enterprises that do most of their commercial operations inside the Net has been increasing year by year. The technologies and methodologies have been evolving constantly, and languages that were in the early stages, are now in the seventh revision, stables, fully featured and ready for production. One of these examples is JavaScript, the most used language along the World-Wide-Web, running in almost 90% of all the websites (w3techs-js) and in the eighth position in the global spectrum ranking (ieee-spectrum).

Scientific teams from fields like bio-sciences, applying their theoretical research into a commercial product, have been growing around the world. Start-ups like Baigene are slowly flourishing, being aware of the power that the Technologies of Information and Communication, and specifically The Internet has to offer, and creating the perfect mixture of versatile groups: experts of areas like Genomics, Sport, Nutrition or Medicine and DNA applications, with knowledge and market management qualified consultants and technological groups with computer engineers.

The importance of cross-disciplinary work and collaboration in new tasks outside of Computer Engineers' usual knowledge areas, from a purely practical perspective, are starting to provide new tools, methodologies and workflows that increase the application of fields like Genetics. At the same time, being involved in this combination of Sciences, are not only changing the way of thinking and communication between both sides, but also transforming our own working procedures, being more flexible for a so changeable environment. Finally, when all of the technology is at a production stage, it adds big improvements and more variability to the mixture, and the whole company need to *dot their i's and cross their t's*, being completely open to change the way of thinking and aware of the market niche opportunities and necessities. The internship, and therefore this Master Thesis, is centered in being part of the growing process of a start-up and its own developing processes. It is focused on understanding some Genetic theoretical bases and applications, the market necessities and how to adapt to them, but also to improve the current methodologies and software modules, implement genetic algorithms and optimize some genetic applications. Finally, understand the process as a whole, and facilitate the work to other company developers and prepare and design a whole system to produce usable software for The Internet with legacy support for previous modules, and design and implement an architecture to improve the development methodologies of

the Genetics data administration.

1.2. Objective

There are two main groups of objectives relative to this project, the first one focused on the Genetics, the second on the computer engineering application.

Genetics is a broad science and a meaningful understanding requires a Master Degree. A practical approach is to narrow the path and study only the relevant parts related with its applications, but supported by experts and their theoretical bases' lessons. In this thesis, it is focused the following:

- Understanding the core concepts of Genomics and Population Genetics to be able to evaluate specific tasks.
- General overview of the company processes relative to the Genetics application.
- Participate in a specific task of Genetics application, understand context and provide tools and new methodologies that improve the workflow from a TIC point of view.
- Studying the application of values per population, the matrix interconnections and the variability and usage of polymorphisms.
- Being involved in the commercial process, from the bioinformatics department ideas, going through the TIC department until the product consolidation, by studying the main parts and improving the procedures and techniques.

On the other side, the Computer Engineering application has more specific objectives, all taking part or with a direct relationship to the Genetysis ® Genetics Software:

- Understand the core of Genetysis ®, review some pieces and learn how the modules work and how could be connected from the new software to be created.
- Analyze the current web development processes, define the key points and improve the workflow based on the requests from the software development department.
- Study the Genetics data administration software, compact and design the GUI and define a whole set of mockups.
- Define an architecture for the Genetics data administration software based on the previous analysis, present the advantages to the bioinformatics and software development group and implement it.
- Improve the module relative to matrix application from two point of views: commercial and computational.
- Design and implement the process of phenotype indicator definition, with the key point of improving the usability for the bioinformatics group.

1.3. Internship planning

To fully understand the magnitude of the work and the difficulties to pick only some parts and still try to establish them inside a whole, the below figure will help. It shows the relations between every stage of the internships, when started and finished every project and task, and the main points of this Master Thesis.

Those projects and learning periods have been divided along three internships, two of them being completed under the supervision of the *Universidad de Cantabria (Employment Orientation and Information Center, COIE, Unican)*. Also, the last internships have been driven by the self-taught learning period since the Master Thesis proposal was offered by the same company. Almost every project have gone even further because of the previous work.

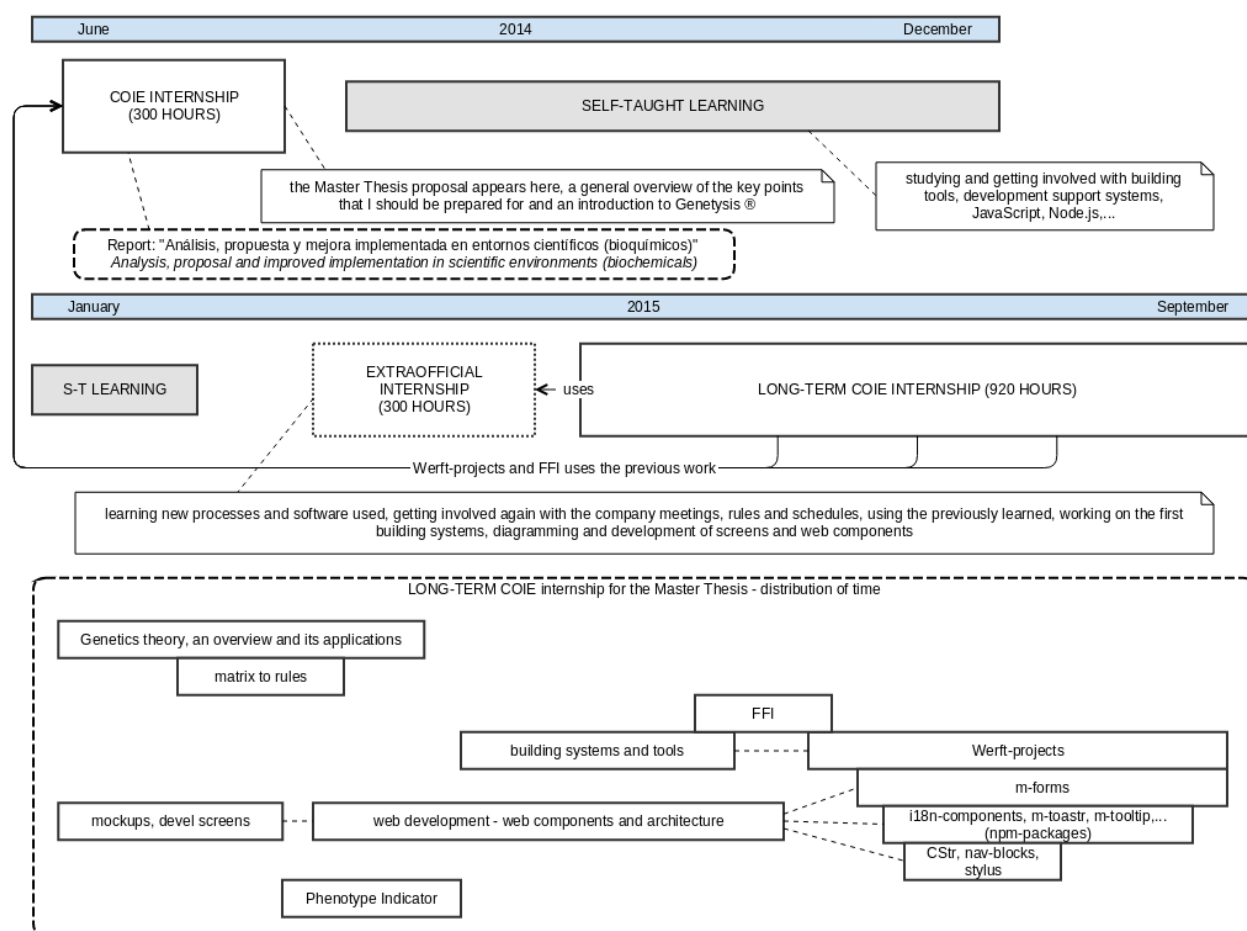


Fig. 1: Internships, projects distribution and planning.

The big blocks are `Werft-projects` and `m-forms`, that are created in a long process of understanding, learning and developing since June 2014. Smaller projects but still connected are `Phenotype Indicator`, `Matrix of combinations to rules` and `FFI`. Also, because

some parts are mentioned in this report, those sections have been localized in the diagram to know when were they created and how much they took.

1.4. Report structure

The thesis is an overview of the main contributions to the most important projects and a summary of the master lines developed in other projects. It has been necessary to remove projects, tasks and sections that were done during the internships, because of the Master Thesis' length restriction. Most of the code is now copyrighted by Baigene, some of the part which has been released license-free for this Master Thesis is so large that only a tiny fraction of it is included here. As a little example, there are more than fifteen thousand lines of *JavaScript* in just `m-forms` and its former designs, via `cloc`.

As it was mentioned in the initial notes but it is important to remark, multiple diagrams and figures have been created along the report to easily review it and facilitate the connection between each part. Because this report is going to be checked digitally, they are embedded with high resolution to be able to zoom to the details and view them properly.

Some of the UML diagrams (and flowcharts) shown are directly taken from the company because they were made to present some processes, document them or define complex sections in a visible and understandable way. They have been modified and adapted to serve the purpose of this report. They are not completely standards-compliant due to the development and prototyping speeds and because of the tools and languages used that extends the limitations of a fully formal UML design (other paradigms, concept approximation).

The Master Thesis report it is structured in seven chapters, all focused on the parts that are presented in the report, and not the whole process, tools or software created during these months. It follows the next structure:

1. Introduction, where the motivation, objectives, internship planning and the report structure is defined.
2. State of the art, it is mentioned, as a general overview, what Baigene uses at the beginning.
3. Then, Methods, Materials and used Technologies, were are listed and explained many tools and techniques used along the projects. Here are mentioned multiple softwares created during the internships and used in the Contents & Results.
4. The Contents & Results, divided in another five parts, each serve a really well defined task, but are all connected as we can see in the below figure. Those parts are:
 - 4.1. Matrix of combinations to rules, the first genetic content, showing the problem, transforming the previous processing algorithm in Baigene, improving it and defining a modular connection to Werft-projects and Foreign Function Interface.

- 4.2. Werft-projects, the fourth and final building system, consolidated and tested with every piece of software shown in the Contents & Results.
- 4.3. m-forms, part of the architecture, web components and mockups made to port Genetysis® and create the genetic content administration software. It describes the help system and briefly some captures of other web components.
 - 4.3.1. The internationalization module, i18n-components, one of the independent packages used in m-forms. It is enough simple to be explained and easily understandable.
 - 4.3.2. Helpers, some support software created during the development of m-forms that has been widely used between the developers of the company. One focused on extending an editor, another to provide an easy transformation between web component views and the last one, to improve the navigation and coding speed in languages like JavaScript.
- 4.4. Phenotype Indicator, also with genetic content, and focused on facilitate the task to the bioinformatics department when defining the Phenotype Indicator. A templating language is defined and implemented using one of the screens of m-forms, using other module named `template-phenotype`.
- 4.5. Foreign Function Interface, the study that allows the connection between Werft-projects, Matrix of combinations to rules and the previous software modules. It explains the ways to connect the new system to languages like C/C++ or Rust.
5. Closing the previous contents, the Conclusions and future work, giving the last notes about the whole Master Thesis, the projects involved in, and future applications.
6. It is included a Bibliography where the most representative references of every section are cited.
7. And finally, the Appendices where are placed the snippets and source codes that are too long to appear inside the document. Also some complementary diagrams to the most important shown along the report.

In the next figure are represented the sections of Contents & Results, which topics and fields are studied and used, the amount of Genetics needed (and communication with bioinformatics department), how related to each other, and an approximation to the Computer Engineering knowledge areas based on the tasks and departments involved with during the previous months.

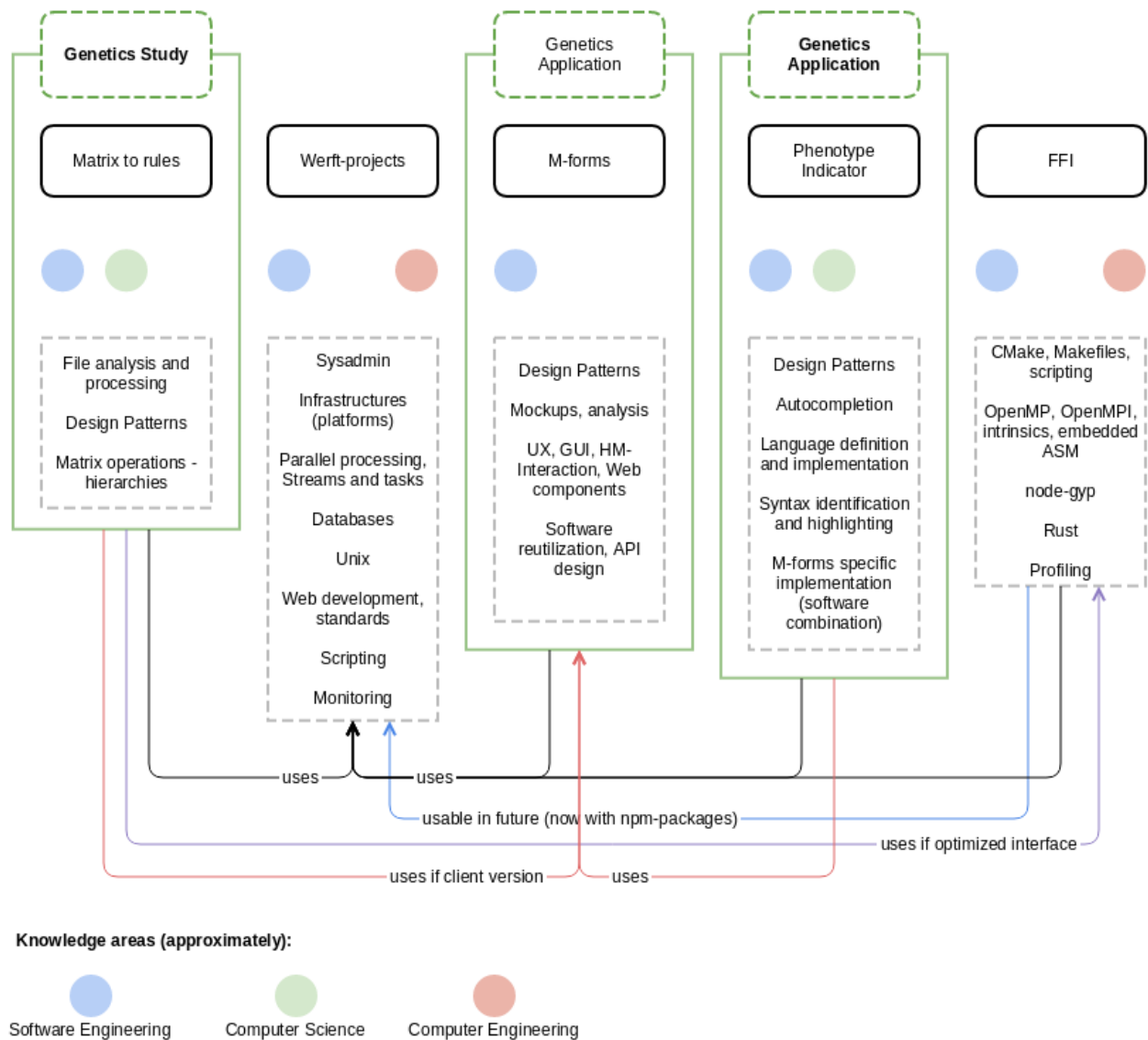


Fig. 2: Contents relationships.

2. State of the art

Under this section are covered the methodologies and technologies that Baigene is using before the Master Thesis' projects started. To be centered in a well defined domain, it is going to be presented per content exposed. The only omitted module in this section is the *FFI*, because there was neither study nor application by itself and it appeared from the Matrix of combinations application.

2.1. Matrix of combinations

The score matrix, where all the polymorphisms are valued and combined with a wide range of processing algorithms, is doing by using a *C/C++* module with thousands of matrices, all being stored and recalculated in case of change. The processing of matrices is doing by querying a MySQL database (in another *branch* the query is established against a *PostGreSQL* instance), obtaining multiple data-sets and then combining its values in unlimited ways, due to the dynamic nature of the combinations.

The definition of new combinations and rules is done using a web interface that communicates directly to the database to set new types of operations, rules and even cloning and modifying matrices. The loading of matrices to the system is doing by using other module that sets multi-dimension arrays and matrices.

The processing speed is fast due to the nature of *C/C++* matrices operations, but the ease of use, adaptation to changes, application of matrix inheritance, and combination of complex multi-domain values are not so good and the server responses are notably delayed.

2.2. Building system and web development

Because there were two main fields inside the web development, client-side (*front-end*) and server-side (*back-end*), the building tools were completely different.

Because the *back-end* was composed of three languages (*C/C++*, *PHP* and *JavaScript*), there were two building systems and multiple tools to facilitate the management of the database, the libraries used and tracked resources. One were focused on building *FastCGI* applications, *C/C++* modules (like the matrices above exposed), and *PHP* scripts to generate the web interfaces. On the other side, the *JavaScript* environment is full of packages using tools like Grunt.js, but also mixing them with *Makefiles* and *Shell* scripts to satisfy all the dependencies, manage the *unit testing* and even the deployment to the web server. The building speeds depends completely in the amount of resources used, but they were not optimized. Some of its tools processed its tasks by dumping the intermediate files to the hard disk and reading after that, slowing the building time. The *unit testing* was applied to every project, as a whole (with tools like gtest and jasmine), instead of splitting in minor parts. In some cases, because of the nature of the libraries and layers of software, even

mixing *unit* and *acceptance tests*. Also, there were complexities when filtering the tests and coverage reports ([15], [16]) from the debugging logs.

Relative to the *front-end* there were tools like the previously mentioned (*Grunt* and *Shell* scripts), but also dependency management software like *Require.js*, *Bower* or whole sets of libraries like *Foundation*, *Bootstrap* or *jQuery*. The amount of resources needed in client-side development extends the hundreds, with complex building rules to satisfy all the necessities of the different departments.

The Genetysis ® software is a software composed of multiple modules, most of them in *C/C++*, but with some interfaces to operate from *PHP*. The main problem is the complex interface to operate with, and how the bioinformatics department need to apply their researches, transform the databases and dump new information every day. The Genetics data administration software is under its early stage of development, with just ideas, sketches and some small implementations using *PHP* and *JavaScript*.

The software development department defined an style guide of programming to follow conventions between the developers, and they used tools like *jscs*, *jshint* and *cpplint* to check and do static analysis to every source code.

The mix of libraries, environments and languages induces to the confusion and complexity, needs lot of maintenance to keep up to date with the latest bugfixes, features and inter-connection adapters, and many times, promotes the replication of code (the same library created in different languages due to the nature of the its application).

Some reasons of using this set is due to the legacy code, speed convenience of *C/C++* modules, well own tested software and fear of change [17], [18].

2.3. Phenotype Indicator

The Phenotype Indicator is a set of descriptions and values dynamically applied to every sample, the nature of its values depends completely on the used genes and phenotype.

The descriptions and values are set using multiple rows specifying the amount of possibilities, and connecting them with a table of descriptions. It was designed with the only purpose of implementing it quickly, because the bioinformatics department need to replicate those descriptions for every new phenotype.

Every description differs from the others in just small portions that have been changed or concatenate strings of results with simple logic operations.

3. Methods, Materials and used Technologies

This chapter describes methodologies, development methods and strategies used to produce or create the software exposed in the Contents & Results. Also, because have been used large amounts of tool-sets and technologies, only the most important are mentioned, although they are grouped in topics to clarify their functions and simplify complex combination of tools. Because most of them are shared between the different projects here exposed, they are not presented per content.

Between all the methods and technologies used in the projects are highlighted those that appear in the Master Thesis report, with the only purpose to be centered in defined tasks and be able to understand its relationships.

3.1. Genetics theoretical bases and applications

To solve the problems exposed in the Matrix of combinations to rules and Phenotype Indicator sections was necessary to understand the core concepts of Genetics Science. With this aim, reviews of scientific and technical books of Genetics ([12], [13]), meetings and talks with the bioinformatics department and a couple of interventions with the Genetysis ® Lead Manager were made for solving the specific questions.

Obtaining the genetic data that were necessary to process in the above applications was done by going through the application of specific Baigene's procedures. In this sense, the DNA genotyping was made in The Sequencing and Genotyping Unit of the University of the Basque Country UPV / EHU by using allele-specific SNPtype assays. The analysis were run using two different ways: 48.48 and 96.96 and Dynamic Arrays™ IFC, in order to address the different possibilities of analysis that Baigene works with. All of them were run using the TaqMan OpenArray and Fluidigm® Biomark HD System platform (Fluidigm Corp., South San Francisco, CA, USA) [14]. The allele-specific SNPtype probes used were designed by Fluidigm platform. Each array was loaded with 94 samples and 96 SNPtype assays or 46 samples and 48 SNPtype assays depending on the type of array used in each case. Samples and assays were mixed and amplified following the default Fluidigm protocol. Fluorescent signals were detected to obtain genotype calls. Finally, resulting data were analyzed using the Fluidigm SNP Genotyping Analysis software. The result output was a thousand lines spreadsheet in a proprietary file format. After the bioinformatics group applied filtering processes the file was ready to be analyzed and processed.

The identification of the necessities of the Genetics data administration software was done by collecting multiple use cases in meetings between the software development and bioinformatics departments. The importance of having a picture of the whole process was as important as understanding the connections between Genetysis® and the software that was going to be created.

Understanding how the Genetics work allow a new set of possibilities when developing, testing or producing new code focused on optimized those processes, even when a high level language was used [5].

3.2. Software methodologies

The software department had implanted some important software methodologies and they were followed in a quite strict way.

Everybody had to follow the programming style guide conventions to be consistent between the different versions of software and people that were in the company during the development stages [2].

Test-Drive-Development (TDD) and a small variant *Behavior-Driven-Development (BDD)* were the roots of building software, techniques that promotes the secure coding because every piece of software should be preceded by its test [3], [4]. There were only specific cases where programming without following *TDD* was allowed: while studying, researching or prototyping.

The hardest pieces of engineering software were under continuous integration, instrumentalized and under high percentages of code coverage, to be sure that every line of code makes sense and there is a *unit test* that satisfies it.

It has been necessary to study and learn software design patterns, not only in web resources but also under language-specific books [11], being aware of the differences and applying higher layers of software development techniques.

Finally, the development process and software life-cycle was following *Agile Techniques* [3], specifically Scrum, where every three weeks to one month there was a meeting with its revisions, demonstrations and reorganizing the team based on how the software was evolving. Support talks and the assistance to conferences were done from time to time to be adapted to this methodologies appropriately.

3.3. General software technologies

Independent of the building software there were some tools that helped during the development processes and was necessary to be comfortable with its technical concepts.

Git and *Subversion* where the main *version control systems* used. *Git* had more popularity and was easy to combine, merge and track remote distributed repositories from the community [4]. In some cases was necessary to clone open source libraries, repair known bugs or add some needed feature for the project in the company and *pull request* to the

owner of the repository [19]. *Subversion* was only used for legacy C/C++ modules and third-party open source libraries.

Based on the previous internship, with even more features and customized, the issue and project tracking software was Jira, and the documentation platform was Confluence. Although many of the tracked Genetysis® modules and development repositories were under their own private intranet with Redmine. These systems facilitate the assignment of tasks, their tracking and resolution.

There were two main editors in the software department, Emacs and WebStorm. Although there are not strict guidelines, the software determined the environment, and developers that were comfortable with *Emacs* needed to move to *WebStorm* in some cases. Some improvements shown in the Helpers section were done to assist and add even more features to the web development process inside the *Emacs* editor [21], [20].

Chapters like Foreign Function Interface were possible with the assistance of tools like *Makefile* and CMake. Also, was determinant to understand the installation and configuration processes from technologies and libraries like OpenMP, OpenMPI and the usage of the compiler intrinsics, allowing to use parallel processing and multi-threading [23], [5], [22].

3.4. Languages and Environments

Due to the amount of languages and environment that were used along the projects here exposed, only the most used are referred.

The dominant language presented in this report and during the previous internships is *JavaScript*, a high level, dynamic, untyped and interpreted programming language standardized in the *ECMAScript* specification [24]. One of the three essential technologies of the World Wide Web alongside *HTML* and *CSS*, also widely used during the web development. The client-side is vanilla *JavaScript*, but the *back-end* is wrapped around Node.js, a *JavaScript* software built over *Chrome's V8 engine* with a non-blocking I/O model and event-driven runtime.

Jade and Stylus template languages where used as a high level abstraction of its counterparts *HTML* and *CSS*, due to the expressiveness, dynamic and robust features like mixins (functions and procedures), inheritance and extension capabilities. There were also ports and usage of styling languages Less and Sass, because some third-party libraries used them and was a convenience to modify small parts instead of rewriting everything. In Werft-projects there are some procedures that uses the Twig template language to be able to process portions of files logically.

Small portions of *SQL* code were necessary to facilitate the task of administering the

`db-adapter` of *Werft-projects*, along with *Bash/Shell* scripts to be able to connect with tools and platforms like *Vagrant* and *Docker* and establish proper environments for the development of projects and its dependencies.

Emacs-Lisp, a dynamic, byte-compiled and functional language, a variant of *Lisp* and created to be the main language of the *Emacs* editor has been used widely to improve the development efficiency and provide more customized features to the day-to-day working process. Sections like *Helpers* show a couple of examples, but during the creation of *Werft-projects* and during the whole creation of *M-forms* and its previous stages more tools and improved modes were built.

C/C++ was used during the evaluation process of *Matrix* of combinations to rules and *Phenotype Indicator*, and when *Werft-projects* needed to be tested against the new *Foreign Function Interface* connector module. The latter section had the most of its usage while combining and configuring the previously explained libraries.

Finally, relative to the *FFI* section, a new systems programming language called *Rust* was tested due to its performance, because of its easy adaptation as a *Node.js* module and due to its efficient *C* bindings that will allow the usage of *C/C++* libraries.

3.5. Building system

Every week new tools and plugins appear in the *Node.js'* *NPM Package Manager* and, therefore, more time is consumed while learning and studying new useful (in some cases) softwares. The evolution of the *npm* packages have been increasing up to the 180000 packages that it stores nowadays. Don't confuse it with the `npm-packages` library built during this Master Thesis, that was called like this because it contains multiple *npm* ports (transformed or adapted to a local repository) and own libraries used by *Werft-projects* and the other projects.

Because the building system, *Werft-projects*, needs more than fifty different direct dependencies (libraries) to be used in its full featured and complete way, the explanations are divided per groups of libraries and its summarized purposes. During the construction of the building system there were necessary another three previous building systems, and previous tools and softwares that achieve their task but have been deprecated or overwhelmed by new plugins and tools are completely discarded from any explanation nor comparison.

Gulp.js is used as the main streaming and building tool, along some *Bash/Shell* scripts and *Makefiles* to support and help more conflicting tasks (like installing *npm* local packages or the `font-factory` management, another library built to help the designers and *front-end* developers to assist their font styles easily and provide a better *GUI*). The main advantage of this *JavaScript* automation tool is that is based on data streams by buffering contents

from one side to another, and achieving faster buildings because of piping (transforming and pushing forward) one task to another.

It has been provided a set of plugins to the stream automation system to add headers (warning developers of avoid editing, setting licenses), linting the source codes and doing static analysis (connection with tools like Tern.js and ESLint, and its importance [2]), providing *unit testing* helpers (splitting processes to help the developers when do *TDD* by sending the debugs to a shell, and the *unit tests* logs to another), source-mapping (debugging) and *TDD/BDD* global features (Mocha.js, its assertion library Chai.js and mocks and adjustable spies from Sinon.js).

Also, because of the web servers' static resources and HTTP caching [1] [7], complete sets of plugins and configurations to be benefit from automated revision of assets, templating languages and source codes.

Other plugins like `gulp-plumber`, `merge2`, `gulp-util` or `through2` are used for multiple purposes: providing error catching of tasks inside the streams, going through reading environment variables to transform completely the whole system based on deployment types, towards the mixing of streams to boost the building speeds.

Relative to the *JavaScript* inside of the building system, it has been built with the last language features in mind. Although the ES2015 spec is defined and the ES7 is under draft [24], most of the browsers and runtimes do not implemented it yet [28] (*Firefox* the most advanced with its 71%). By using "transpiling" tools like Babel.js every *JavaScript* can be coded in ES2015 (and some ES7 features) and be transformed to valid ES5). Also, Sweet.js, a macro template language (to extend the *JavaScript* features), and a set of "minification" tools have been added (`clean-css`, `uglify` or `imagemin`), to reduce the final pack for production deployments.

The chosen web server that support the whole building process, serve the assets, manage the routing and imitate production-type deployments is Express.js. It has been adapted with multiple packages to support features like connection to databases (using another created library `db-adapter` [9]), debugging and logging, a dynamic-paths helper, modified rendering views and facilities to test under security-aware development states (*Secure Sockets Layer*, *SSL* and *CORS*, [27]), [1], [7]).

The creation of library containers like *Stylus-plugins* centered all the web components style definitions, and also define a set of plugins to be attached to the *Stylus* engine, allowing to be usable from the whole system. Also, the most common CSS libraries have been ported and modified the connection points to be more intuitive (in usage and for debugging purposes) for the developers. Also, adapting engines like *Jade* and caching views allows the usage of development and design patterns like Lorem Ipsum, and helping the developers with its stages.

Instead of writing *front-end JavaScript* with old and multiple dependency systems, the new tool supports *Commonjs*, allowing the developers to write browser code like in *Node.js*, facilitating its application and being able to use the whole *npm package system*. Also, tools like *Bower* (client-side libraries) are still usable because of the embedded configuration, but a local library container `npm-packages` was also made to fetch AMD.js-compliant libraries and port easily to the new system (*Commonjs*, taking advantage of its coding features). The chosen module bundler is Webpack, but restricting its usage to only “post-processed” *JavaScript* (ES5 compliant, after the “transpiling” but before the “minification”) because better tools have been created to support the development and building process (CSS, HTML, images and fonts). The modification of the adapter has allowed the definition of dynamic grouping of bundles and new debugging features to help the developers (following the same conventions as in *Stylus-plugins*).

3.6. Graphical User Interfaces

The Graphical Interfaces are one of the key points when manipulating multiple factors and complex systems. The usability and *User Experience (UX)* have been between the learning fields during the development of M-forms and Phenotype Indicator, not only because of the requirements and ease of use for the bioinformatics group, but also because some parts are focused on a commercial product [6], [25].

All the web development process have started with previous analysis of every idea, multiple mockups and wire-framing of designs and surveys to know the users’ voice (in most cases other developers and technical users).

Those analysis defined the web design and the target to achieve when the web components architecture was in its early stage and how the development was going to be with centered in the usability, responsive web documents and promoting the user efficiency when working in scientific tasks.

Most of the time there were multiple ways to achieve one task, but in a easy-to-follow path, and showing the alternatives at any time. As risk measures, in case of failure is possible the degradation of features (*fallbacking*) to support the needed operation (stability and availability centered) [2], [7].

3.7. Web components

M-forms is built based on the concept of software reutilization, modularity, usability (final users and developers) and flexibility.

It is designed over the concept of *Component-based Software Programming*, where every piece is a solid unit with a clear and manipulable interface to the outside system following

a set of specifications, has facilities to operate with it (actions and events, data manipulation and easy adaptation), it works by itself (encapsulated) and it is part of the whole system (*components*) being reusable.

The connected pieces in the design and implementation of the architecture are the *HTML*, *CSS* and *JavaScript*, usually manipulated through the user interaction, defined with Browser-defined events (set in the *Document Object Model*, *DOM*), responses and renderings.

Every component is a conjunction of optional and mandatory parts (not every component has to be visible, therefore could not be defined with a set of *CSS* rules and *HTML* tags), that acts in favor of a greater system and works in predefined ways, usually modeled by the whole architecture (*m-forms*) and its workflow.

The views are orchestrated by a “low-level” rendering *JavaScript* framework called *Mithril.js*, due to its light-weight and robust features, and the most important: the Virtual *DOM* strategies and intelligent auto-redrawing system. It creates a intermediate layer before writing to the *DOM*, being the fastest framework so far [26], and being necessary because the Genetics data administration software needs dozens of layers of abstraction (*GUI components*), complex business logic and data manipulation, and the rendering of thousands of data-sets directly in the browser.

Although all these features increased the website speed and was easy to use in simple applications, a small library *Mithril-Modules* was built over *Mithril.js* as a schema for the web components, with its *submodules*, *models*, *controllers*, *view-models*, *views* and *modules* to facilitate its use. Also, when adapting some designed web components, the applicability was not straightforward, and the *Mithril.js*’ rendering strategies needed to be manipulated due to the complexity of some modules like *m-toastr*, *m-tooltip* and *Draggable*.

4. Contents & Results

4.1. Matrix of combinations to rules

During this section are highlighted the advantages and how the new developed process works. To understand the differences between the old system and the new one, the below figure will help.

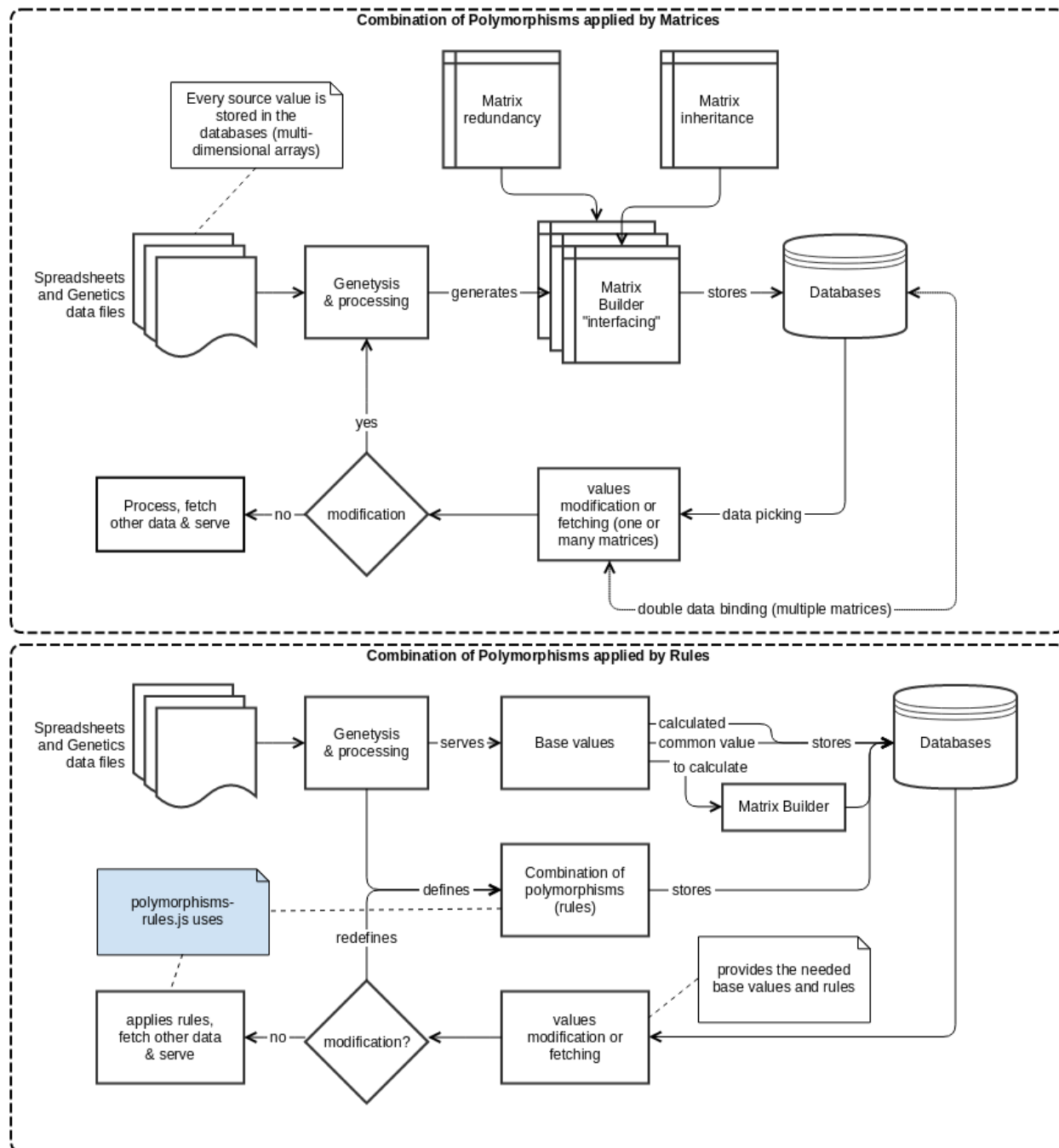


Fig. 3: Schemes of polymorphisms combinations (matrices to rules).

The old system worked by needing the whole set of matrices, its own interactions and combinations to be able to determine a set of results. Also, it worked using the “double data-binding” between the process that fetches matrices and the databases, being multiple fetching-and-processing operations of matrices until the final value is obtained (due to C/C++ processing logic instead being part of the Database Management System, it is needed an undefined number of queries because of the dynamic nature of the interactions).

The matrices were created by a combination of complex inheritance (saving space) and redundancy data-sets (multi-dimensional arrays per sample with a fixed value, obtaining faster readings). The complexity comes from the necessity of applying the same genes to hundreds of phenotypes, each of one has its own combination of polymorphisms. The inheritance tried to combine polymorphisms from different phenotypes, while the matrices redundancy stored every combination in a different matrix (mapping every set of combinations, increasing the calculating speeds when a sample is requested).

In the general picture, taking into account the whole system and not only the application of phenotypes and polymorphisms, the computation complexity and processing times were not satisfactory, due to the tree of dependencies.

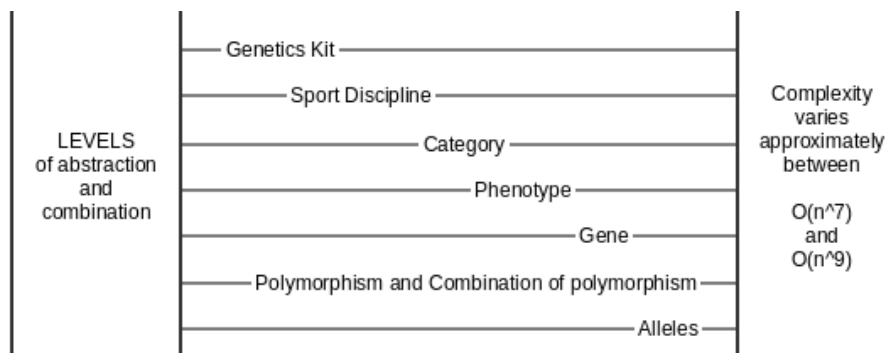


Fig. 4: Tree of dependencies and levels of abstraction.

Each item represents a one or many of the next item (nested calculations), but the last one has an unknown number of possibilities because the R&D bioinformatics department is always working on more complex models. Every phenotype is composed of genes, that itself is composed of one or many polymorphisms (*Single Nucleotide Polimorphisms, SNP*) that, at the same time it is not necessary to be diallelic (2-variant). Some seen values inside the polymorphism level have been between a 3-cell matrix up to more than a million-cell matrix. These so huge differences come from non-linear increasing factors, like number of combinations based on formulas like the genotypes (n) per alleles in the *SNP*: $n \times (n+1) / 2$.

The solution provided is a combination of task reorganization, software redesign and modification of the workflow process.

First, the input data has been divided in two paths (figure 3): base values and rules definition (combinations of polymorphisms). The base values support three type of inputs:

- Common operations and algorithms. Some of the previous matrix definitions were based on Genetics known processes or Baigene's algorithms.
- Complex associations and almost impossible-to-implement algorithms. Values that are not represented by known functions, random values or modifiable associations based on the research.
- A shared common value. In some cases the matrices defined were really simple and were bloating the databases with unnecessary data.

The first two types need the output files created by Genetysis® and processed and filtered by the bioinformatics department. While the first one generates all the vectors, the second one only passes them to the databases. The third one has the advantage of saving space and time.

The variation of associations will only modify its own base value (it does not matter the type). On the other side, the way to provide the variability to the system (new experiments and research) and the configuration of polymorphisms and its combinations is by using the new rules system.

Every rule has a data to operate with (polymorphisms and its genotypes), values to set (text and a numeric value), and the applicability (which values are applied). Also, rules are chained to form combinations of polymorphisms itself. It can be understood as a table: every polymorphism and its defined genotypes is a column (the logical operator AND), and every set of polymorphisms is a row (OR). When there is a match between (at least one row match with all its columns being matched) it is said that the rule is applied over the sample. If a row does not have all the polymorphisms and/or genotypes defined in the phenotype, it is also matched (AND TRUE).

Rules	Values
[BG_0021 (CT CC) BG_0040 (GG)]	<input type="checkbox"/> Incremento de la proporción...
[BG_0021 (TT)]	<input checked="" type="checkbox"/> 6
Rules	Values
[BG_0080 (GT GG) BG_0083 (GG CT) BG_0001 (CC)]	<input checked="" type="checkbox"/> Aumento de la lesionabilidad...
[BG_0021 (TT) BG_0080 (CT CC)]	<input checked="" type="checkbox"/> 2.5

Fig. 5: Portion of web interface (m-forms) showing the rules visualization and its values.

By setting this system, all the possible variations that the bioinformatics department are doing constantly to the databases can be modified easily, reprocessing every possible combinations smoothly and fast.

Second, it has been implemented using the *Chain of Responsibility* (CoR, [29]) software design pattern, due to the loose coupling, adaptability to new changes (modifiable and extensible *handlers*) and how it fits to this solution.

The CoR pattern have been implemented in *JavaScript* and wrapped to be usable for this use case. Also, it has been modified to support extended features like passing messages in the chain or escaping from the chain on different conditions. The extension has been appended to the appendix with the listing 1 containing a example of application and its results.

With this implementation it is possible to share exactly the same code in the server and in the browser. The first one will determine the application of combinations and what are the final values for the sample. The second allows testing multiple values and interacting in a simulation process (a provided canvas of combinations, how many genotypes are used and which rules are applied) while the bioinformatics department is filling the data-sets and establishing the relations.

The storage problems have almost disappear because the only matrices that are now inside the databases are those from complex associations (second type of input), the rest are just rows of rules, base values and formulas that can be generated at runtime. Also, it has been provided a solution to increase the performance by pre-building the matrices and storing its cells in the databases just for the sake of speeding up the processing of analysis with the new system. The new mode is not only flexible, but memory and performance aware.

Finally, the new distribution of data and how this process has been organized, combined with new tools and studies like the Foreign Function Interface section, allows the reutilization of previous modules (C/C++) and fast processing algorithms from the own web interface, even connecting them to the simulation process, not only with the rules, but also with the `MatrixBuilder` and its own generators.

4.2. Werft-projects (building system)

Werft-projects (an analogy to a *shipyard* where boats are built) the fourth trial of generating a building system for the web development and with connections to a multiple tools and platforms like the those studied in previous internships.

It is aimed to help the developers in its process of creating web applications, both client and server-side. The whole compendium of environment configurations, plugins utilization

and development settings have been thought while working on many small and medium projects and after learning the main features of all them (Genetysis ® but also specific modules). The prior development of other building systems like *Orchestrator*, analyzing its caveats and most problematic tasks helped to reduce the development complexities (lot of assistance, really difficult to use and some maintainers were needed). This new building system provides the following features:

- Ease of use. The inner directories, intermediate files and established conventions are transparent to the developer. Special directories (`omit`, `trash`, `shared` and `partial`) defined by a new convention of development improve the whole process.
- Task centered. Multiple combinations of directories, files and data streams are connected to logical tasks.
- Development type modes. Each type of development has its own rules. Providing helpers and task groups allows to focus on the development instead on the building process.
- Emulate production use. Since the very first moment, it has been designed with a shared environment, thinking on a production usage, and just reverting some features (or being transparent) that are not applicable in development mode. This design allows a drastic reduction of deployment problems.
- Convenience in mixed environments. By analyzing multiple projects and its necessities it has been created a whole processes to assist the installation and application of libraries, automatic connection to databases and dependency fetching (`dependency-tree`).
- Pluggable interface (*Werft-modules*). Instead of having a huge building system that “does almost everything”, a pluggable interface to other modules has been attached. Libraries like `font-factory` are completely independent but are connectable to the dependency manager of *Werft-projects*.
- Speed buildings. Instead of increasing the developer load (by using other command-line tools or manual methodologies), the building system itself has all the complexities and the slowest tasks has been optimized.

Although the tasks have an established order of execution, they have been modified afterwards to be even more convenient for the developers, by changing themselves and normalizing the order of execution in case some failed (but the first run will not be so fast as if it is executed in order). These type of design rules have been implemented in many ways, with the only purpose in mind to help during the development process.

Table 1: Werft tasks, order and dependencies.

task name	server	client	client (temp)	dependencies (only in npm-packages)
werft/config	† 1	† 1	† 1	font-factory, db-adapter
werft/deps	† 2	† 2	† 2	dependency-tree
babel	† 3	† 3	* 3	
server	† 4	† 6		
bs	5	7		
jade		† 3		jade-lorem, jade2mithril
styl		‡ 3		stylus-plugins
assets	3	‡ 3		
wp		‡ 4		
rev		† 5		
refinery			† 4	jade-lorem, jade2mithril, stylus-plugins
lint	**	**	**	
test	**	**	**	
clean	**	**	**	

* (if external dependencies) ** (independent) † (mandatory) ‡ (mandatory if css, assets and/or js)

Also, a full set of libraries and modules have been created with the only purpose to satisfy the whole system. There are modules that help during the definition of a *werft* project, modification of templating engines to produce multi-purpose views (*jade2mithril*), dependency fetcher, modifications to help the debugging and testing process and even connectors to platforms like *Vagrant* and assist the initialization of *DNS* servers and databases that the development department usually need.

The advantage of having all previous tools and languages unified multiple. Not only because of the shared code between both sides, like in Matrix of combinations to rules or Phenotype Indicator, but also because the building system it is almost entirely built in *JavaScript*, and the expressiveness and flexibility of this environment can be adjusted easily to new requirements. Also, it is shown in the Foreign Function Interface section, *Werft-projects* allows the usage of other modules built in *Rust* or *C/C++*, although it can only be connected if it is a server-side *Werft-Modules* project. External dependencies (and other package management systems [30]), third-party modules or version control repositories can be nested and transparently exist inside a *Werft-Modules* project, because it was designed to be self-contained and easily transferable.

It is formed by single units, each of one is a *Werft* project. Just a directory with an inner *werft* directory that has a `config.js` implementing a *Werft-Modules* interface. They can be composed of other units, but those units does not need to be inside the parent one. This design allows the reutilization of projects of any size, easy debugging and testing and create hierarchies of abstraction layers based on projects, modules and tools, but all being the same thing, a *Werft-Module*.

The main tasks are shown in the table 1, but to be in a specific domain and understand what a task is, `babel` is taken.

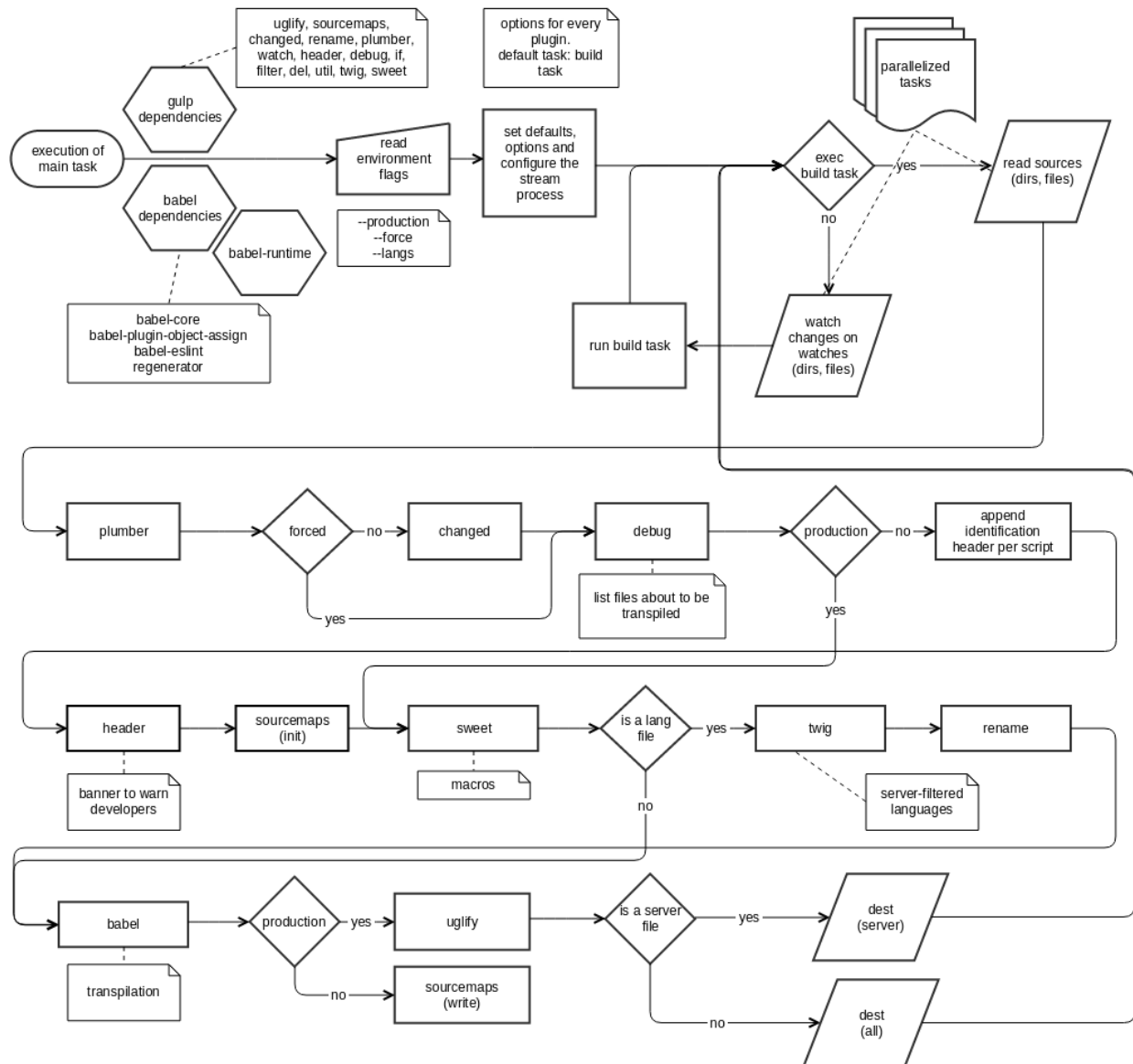


Fig. 6: Flowchart showing the stream processes in the `babel` task.

A task starts with its execution, and it accepts a list of environment variables and shell parameters (*Gulp* is executed from the command-line), being all of them documented to assist the developer in its needs. With the loading, configuring and initialization of dependencies (usually from dozens to hundreds of nested dependencies), the task is started, and two processes start: the watcher and the builder.

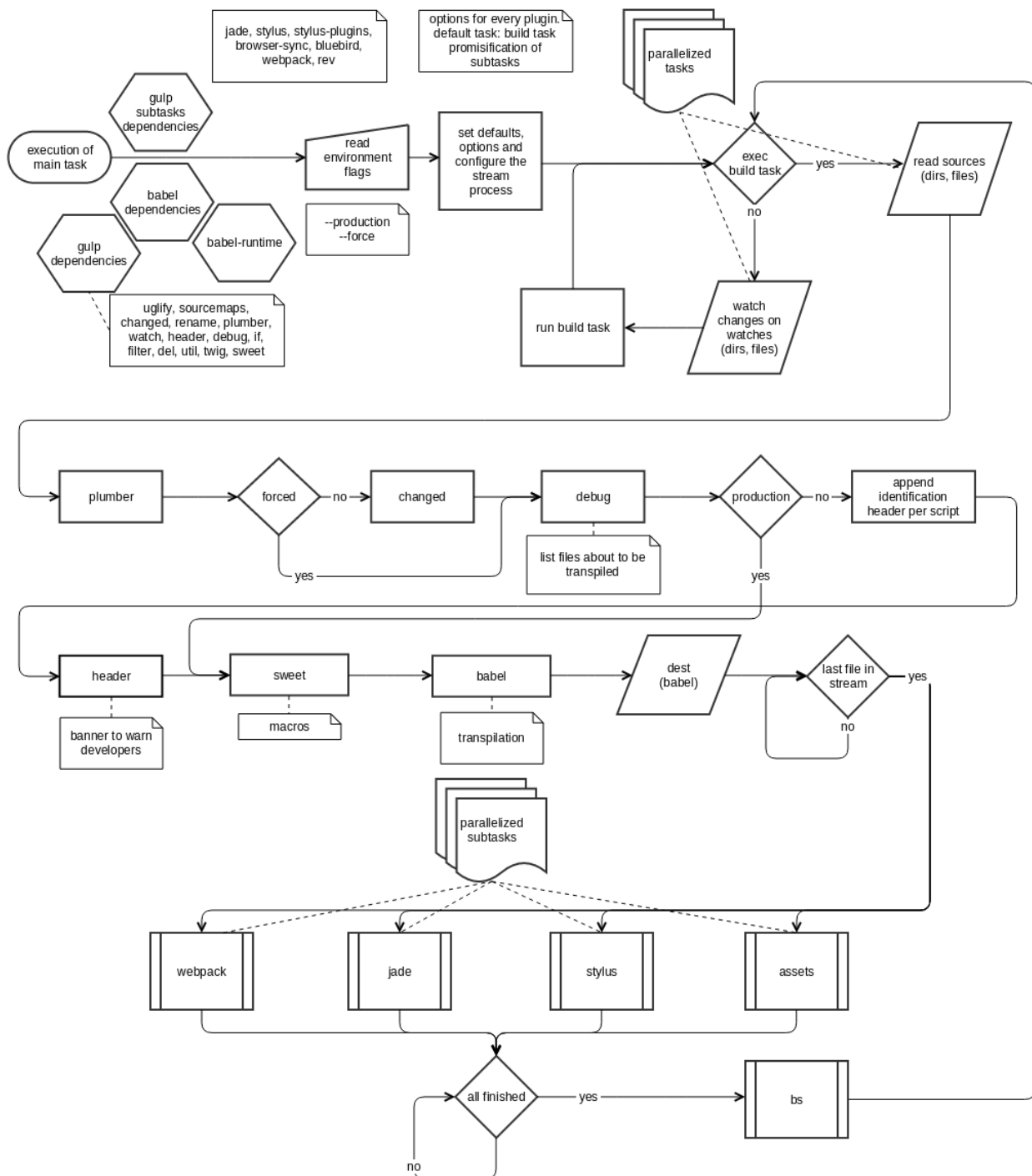


Fig. 7: Flowchart showing the stream processes and its subtasks in the `refinery` task (under *Refinery* mode).

The watcher is doing *polling* to a set of resources that can vary per project, but usually are hundreds of assets, layouts and libraries. Every time the watcher releases a notification of changing, the builder starts (there are some measures to be sure the execution is neither paralyzed nor under recursive execution due to the communication latencies between the file system and the process).

The builder read the sources, being usually a subset of the watcher files (but not necessarily), and push them directly to the initiated stream. The stream has internal processes and triggers to new subtasks, but the global picture includes tasks to control the streaming process, optimize the building of resources that have not changed (but accepting environment parameters to change its behavior dynamically), include debugging information, transforming *Sweet* macros to pure *JavaScript*, transforming internationalization special files to valid *JavaScript* modules to be used by *Werft*, “transpilation” of *ES2015 JavaScript* to be able to use the latest features, change the streaming paths varying of a production building (“minifying” the final builds to save space and execute the code faster) and finally, sending the built resources to paths accordingly of its nature (deployment type).

There are tasks for every purpose a web developer needs, but after working with them while building other projects (like M-forms), it has been designed and implemented a special task that optimizes completely the client-side building process, called *refinery*. It is not only a task, but also a special development mode (*Refinery*), an easily configurable from its *Werft-Modules* interface (just a flag).

All the *Werft-projects* have been revised and modified to support multiple entries (executors), coming from the common tasks, but also from external processes (merging streams and controlling its execution steps). The creation of this mode has decreased the execution times and helped during temporal web development stages, although its limitation is not matter of time but complexity. The more pieces a project has, the less suited is for the *Refinery*. This mode aims to work in *client-side* with just three tasks (instead of the usual nine) and much faster. Real-world examples include moving *Werft* projects to the *Refinery*, working for some time and solve all the issues easily and with speed boosts, and returning it to the previous location and, therefore, working again under *Client* mode.

The last and most important modification of *Werft-projects* is the global optimization for every mode. The *refactoring* and modularization of all the tasks, local dependencies and global *Werft* configurations has promoted the development of a new conjunction of addons, switchable from a new environment flag (now activated by default), that allows the optimum loading of dependencies.

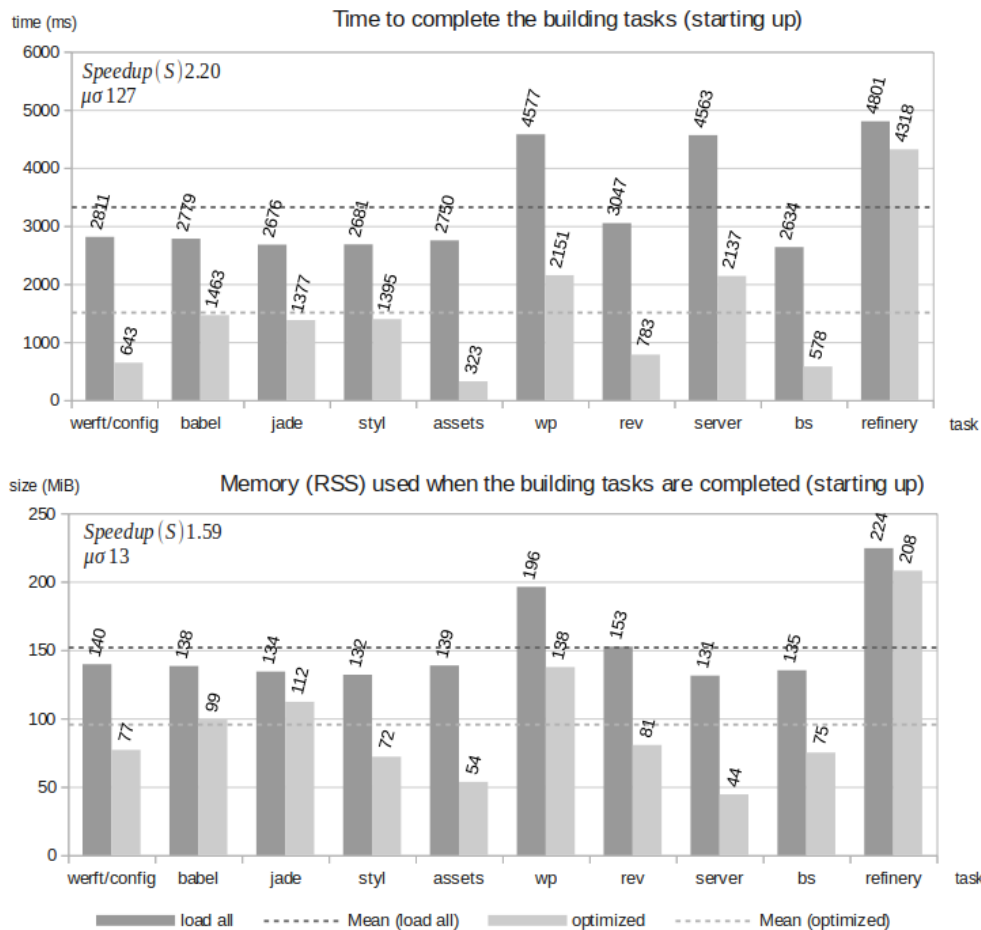


Fig. 8: Comparative of tasks and its optimized loading (100 executions, build tasks).

A dynamic tree of dependencies has been created, and due to the modularization of code, only dependencies of a task are loaded if its tasks are requested. Also, it determines which subtasks are needed so it pre-loads them, instead of loading when the tasks have started (gaining performance, another important issue in every building system).

Every task decreases its execution times and its memory footprint. The Speedups are up to 2.20 and 1.59, respectively. The *Refinery* mode and its task are the slowest and most RAM consuming, but being analyzed as a single unit (seldom a developer uses only independent tasks), and the *development modes* shows even better performance rates (Speedups of 2.31 and 1.65 in average, per type).

The differences between working in *Client* and *Refinery* mode can be appreciated in the next table. Both development modes were using the optimized loader, and still there are average speedups of 2.40 times (time and memory).

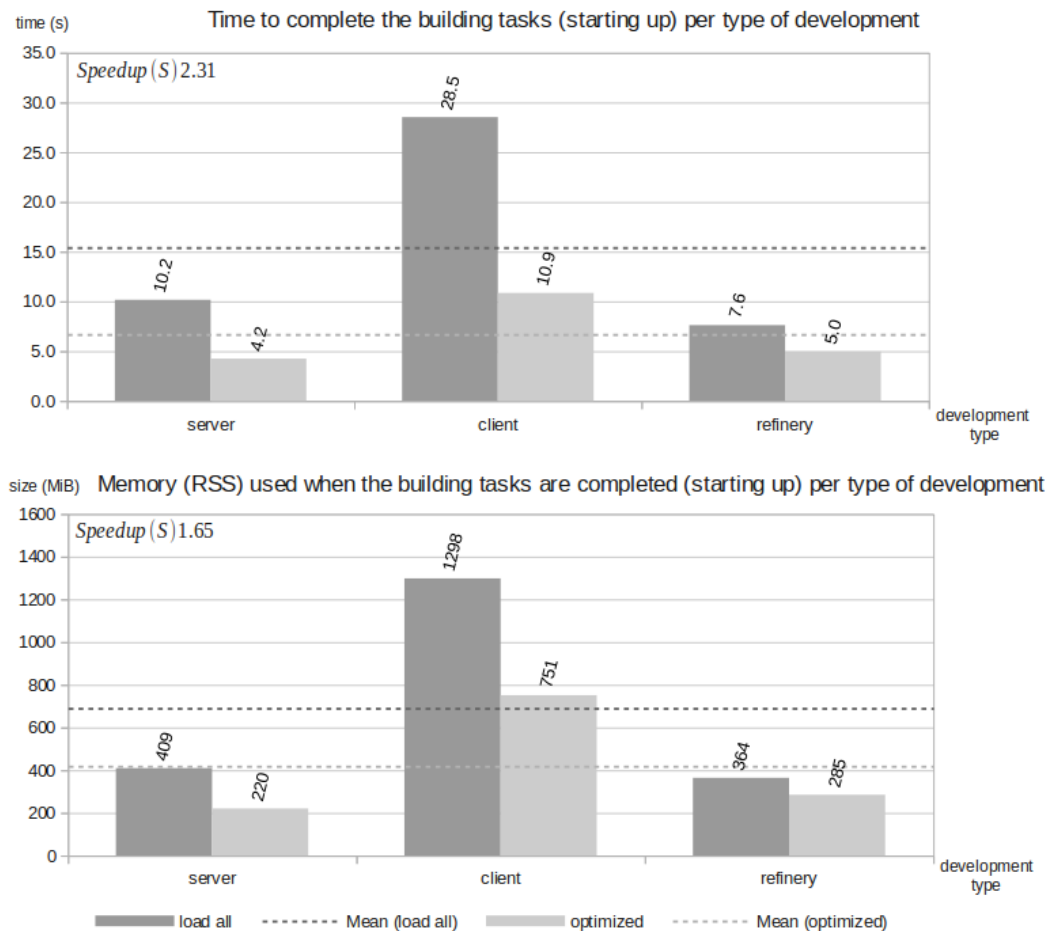


Fig. 9: Comparative of development modes and its optimized loading (100 executions, build tasks).

Table 2: Comparison between client and refinery development types (using the optimized version).

Development type	avg. time (ms)	avg. memory (MiB)	geometric mean
client	10850	751	2855.03
refinery	4961	285	1189.03
Speedup (ref/cli)	2.19	2.64	2.40

With the creation of *Werft-projects* every previous project has been ported to the new system, increasing not only the success rate (less problems while creating the web applications) but also the system loading and development cycles (less building time, more time coding and thinking in the real problem).

The next project have been developed entirely inside *Werft*.

4.3. m-forms (Genetysis ® GUI)

The resulting web component architecture, *m-forms*, started in a process of analyzing the previous Genetysis ® software and the expectations for the new Genetics data administration software. After understanding every stage that the application needed, it was thought (wireframing) and designed through a set of mockups with the only objective to promote the bioinformatics department usability and the unification *Graphical User Interface (GUI)*.

The figure displays eight mockups for the *m-forms* GUI, organized into two rows. The top row includes:

- Fenotipo**: A form with sections for 'Información' (nombre, resaca de tendones y ligamentos), 'Indicador de fenotipo' (valor mínimo, valor máximo, población general), and 'Regla de combinación de polimorfismos'.
- Polimorfismos**: A form showing a list of polymorphisms (e.g., BG_0021, BG_0040) with checkboxes for selection.
- Valores base**: A form with radio buttons for 'valor común base' and 'valor calculado', and a 'Seleccionar fichero adj./ch.' button.
- Combinación de polimorfismos**: A form with a 'Regla Heffner' section and a 'nueva regla' button.

 The bottom row includes:

- Polimorfismo**: A form with an 'Edición' section for 'código' and 'regla', and a 'Genotipos' section for 'CC' and 'CT'.
- Regla de combinación de polimorfismos**: A form with an 'Edición' section for 'regla' and 'Genotipos'.
- Categoría**: A form with an 'Información' section for 'nombre' and 'resaca de tendones y ligamentos', and a 'Fenotipos' section.
- Fenotipos**: A form with an 'Información' section for 'nombre' and 'resaca de tendones y ligamentos', and a 'Fenotipos' section.

 Each mockup contains various input fields, checkboxes, and buttons, demonstrating the flexibility and reusability of the *m-forms* architecture.

Fig. 10: Some mockups designed and accepted in the first meetings.

As can be seen in the previous mockups, most of them are made of defined structures and nested and reusable designs. While thinking the whole architecture, this is one of the targets to achieve: a solid, easy to use and flexible *GUI*.

M-forms was achieved after building some other projects that involved *GUI* development combined with complex business logic. Those experiences changed the way of building web applications, and other two previous web components projects were canceled in favor of *m-forms*.

The architecture is based on reusable components, composed of two types of *Decorators* (a design pattern that attach additional responsibilities to an object dynamically, allows the mutability of components and its flexible adaptation [31]), but modified from a *JavaScript* perspective due to its dynamic bindings. In the top layers have been used *Builder* and *Abstract Factories*, design patterns that helps during the configuration and creation of com-

plex objects, and provides levels of abstraction of families of related or dependent objects without specifying in its early stages the final defined component [32], [33].

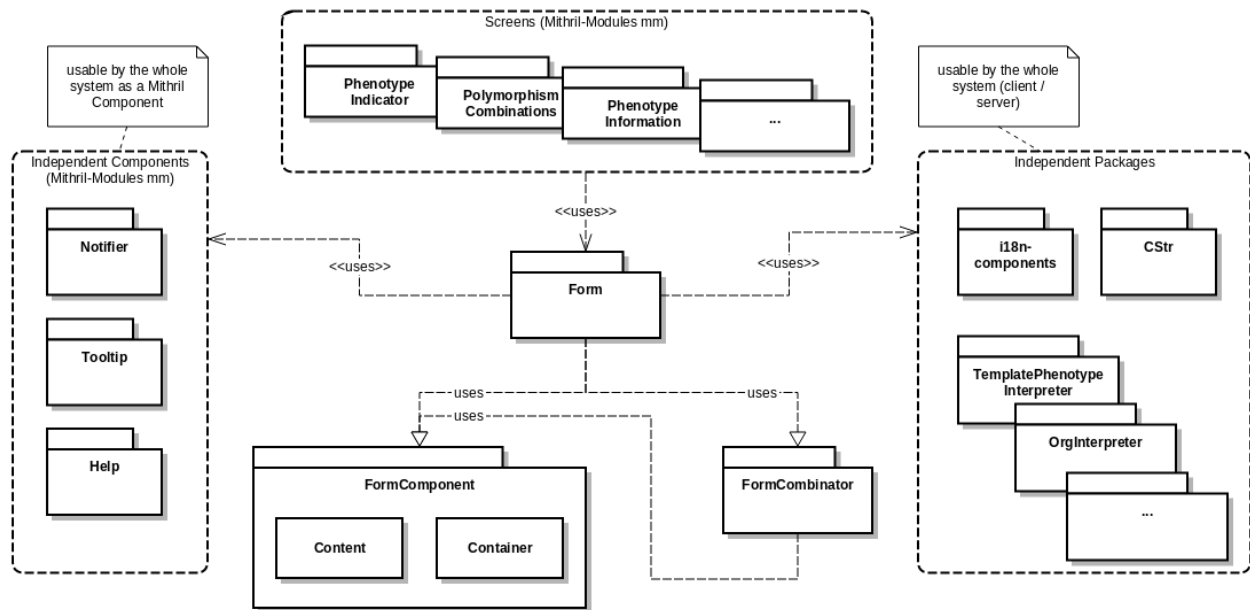


Fig. 11: Overview of the m-forms components system.

The abstraction layers, types of components and containers to build the whole m-forms system has so many pieces that only some parts are represented and the larger diagrams sent to the appendices, like the figures 30 and 31.

Most of the web components does not use the interface `Mithril-Modules` (thin wrapper to operate with `Mithril.js`) but are dependent of inner interfaces `m-forms`, `notifier`, `formComponent` that define the whole system. The components that have been prepared to be used everywhere (not only in m-forms) use the `Mithril-Modules`.

The components have been divided in main constructors (*factories* and *builders*), independent components (attachable to multiple parts of the system and other components, using the `Mithril-Modules` interface), independent packages (usually embedded in other softwares, contained in `npm-packages` and with a thin layer over them to be used in multiple projects without the necessity to be used with m-forms).

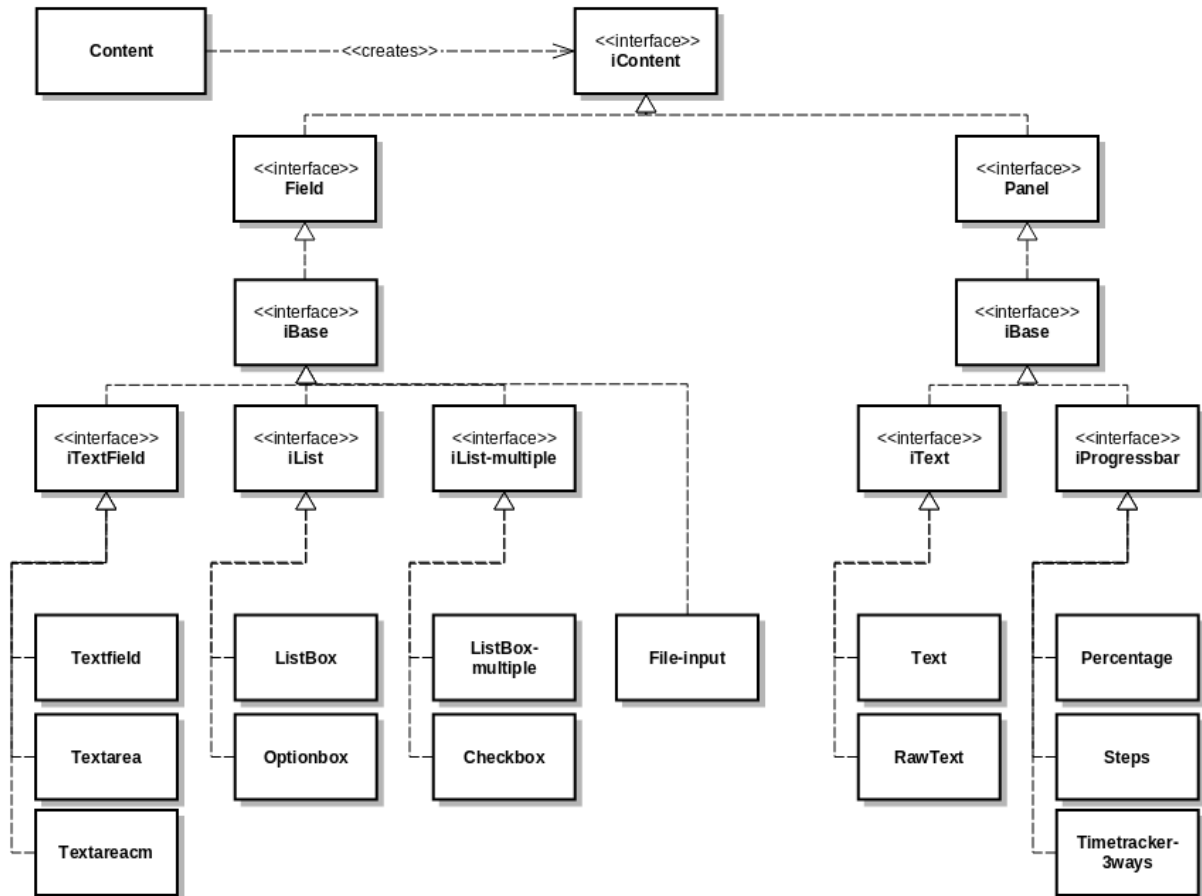


Fig. 12: Overview of content components (m-forms nested level of components).

The main constructor is the *form*, which creates and manipulates a set of minor components to provide an easy interface to the *screens*, which are the top level of forms construction (every mockup can be materialized as a *screen*). The minor components are usually *formComponent* and *formCombinator*, being the combinator a mix of *formComponents*, inner modules and external components (independent or package, figure 11). The *formComponents* have two main types: the *containers* and the *contents*. The first ones usually defines abstract blocks, contexts and in some cases, *GUI* separators and groups of minor components joined logically. The second group is formed by interactuable components, that usually came in another two ways: *panels* (information purposes) and *fields* (every content that an user needs to operate with).

The architecture, its components definitions and how the events are processed are schematized in the next diagram. The *GUI* itself is part of a sum of strategies, combination of technologies and methodologies that establish by itself, an own creational and behavioral framework that is supported by the *Mithril.js* rendering framework.

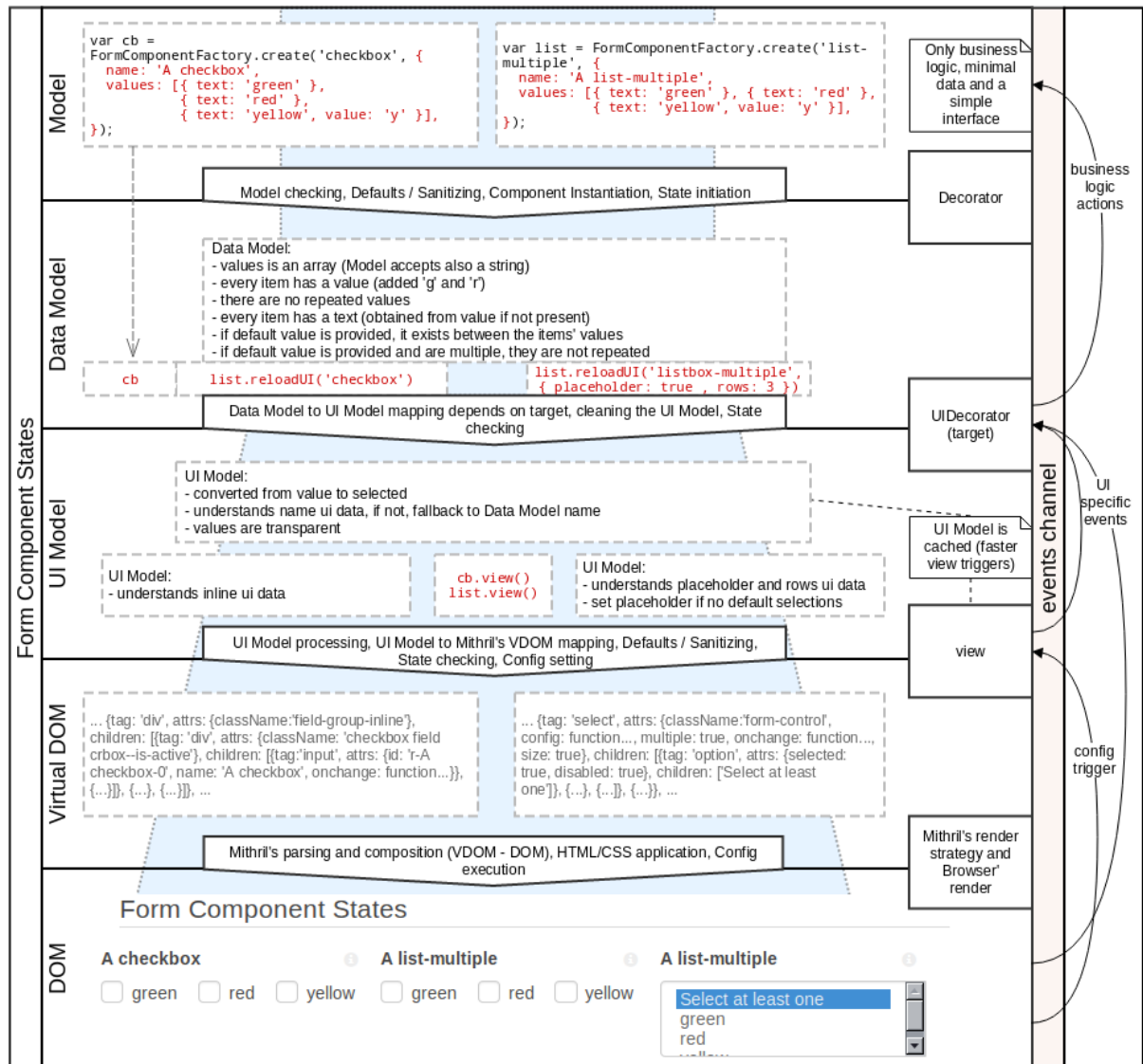


Fig. 13: Genetics data administration GUI architecture overview.

It defines multiple abstractions to help development efforts focus on the business logic (one of the problems present in previous architectures), and by providing an easy interface to manipulate every component, send the data in a simple way, render it transparently and capture the possible events directly in the top-level layer [6], [34].

Libraries that have been used in some components can be shared across the software, as it happens with the `CStr` string buffer module or the internationalization one (explained below). It provides multiple parsing features used in other specific components as the `OrgInterpreter` or `formCombinator` views like the below (interpretation of `Org` features as a extension to plain descriptions to facilitate the researches to do their job appropriately). It has been appended as the listing 2 in the appendices.

Lista de polimorfismos

BG_0021 Este gen codifica uno de los dos tipos de cadenas que forman el colágeno tipo V de las ya conocidas...	 	CC  Menor susceptibilidad de lesión sin contacto	CT  Susceptibilidad media de lesión sin contacto	TT  Mayor susceptibilidad de lesión sin contacto
BG_0067 [indefinido]		GG  [indefinido]	GC  promedio poblacional...	CC  desventaja cuantitativa...

Lista de combinación de polimorfismos





Interacción 2x1+1 Regla definida a partir de la lectura de artículos #2929 y #2913 . <u>Importante: revisar los conceptos subrayados</u>	 	Rules [BG_0021 (CT CC) BG_0040 (GG)] [BG_0021 (TT)]	Values <input type="checkbox"/> Incremento de la proporción... <input checked="" type="checkbox"/> 6
Regla de Sforza Cambio variable ante genes de triple mutación (Sforza). Artículo #1193 . Extraído de http://www.uwyo.edu/dbmcd/molmark/lect06/lect6.html	 	Rules [BG_0080 (GT GG) BG_0083 (GG CT) BG_0001 (CC)] [BG_0021 (TT) BG_0080 (CT CC)]	Values <input checked="" type="checkbox"/> Aumento de la lesionabilidad... <input checked="" type="checkbox"/> 2.5

Fig. 14: formCombinator in a demonstration of two complex views (Polymorphisms and Combinations) and with mixed support of `il8n-components`.

There are multiple new features, but one of them is highlighted: the *Help* system, which have been designed with great adaptability to changes and by supporting direct manipulation of views. This special component is connected with the whole system and its screens, and from the development process, its early definition until the implementation over specific components has been designed to be pluggable and easily testable. The final users (bioinformatics department and new researches) use it, and in case any documentation needs to be changed, even a non-technical person can modify it. Then, the developers can see the differences easily and render it statically in an easy way (this follows the process of static creation, dynamic interaction in a static view and finally, the dynamic view), can be seen in the figure 28 of the appendices.

The pluggable system is attached with simple patterns to other components, and automatically a set of features are activated per form element, like the `button` and `m-tooltip` system to indicate, show or hide (like the user wants) the documentation of every field.

Field.textfield

Base common value for every cell. The value is between the minimum and maximum defined for the phenotype.

$x = \{x \in \mathbb{R} \mid \min \leq x \leq \max\}$

0 0.8 7

Field.optionbox

- ☐ Common value
☐ Table to calculate
☒ Table calculated


Type of insertion when filling the matrix default values.


Common value: the same value to every cell.

Table to calculate: spreadsheet with a table of values and the definition of an operation (procedure of value combination).

Table calculated: spreadsheet with a table of values previously calculated (and combined).

Field.file-input

 Drag & Drop the files here or

select them
 

Spreadsheet with a sheet named exactly like the phenotype and starting from the top-left corner (cell 1:A) with the following format.

Fasting glycemia

RS_CODE					VALUE
BG_0076	BG_0102	BG_0088	BG_0008	BG_0160	
TT	CC	AA	GG	CC	9.216
TT	CC	AA	GG	CT	8.694
TT	CC	AA	GG	TT	8.172
TT	CC	AA	GC	CC	8.01
...
CC	TT	GG	CC	TT	0

Because it is a calculated table, the value does not need to come from any mathematical formula. **RS_CODE** can be an horizontal cell combined with as many cells as polymorphisms are in the phenotype, but also those cells not combined but sharing the same value. **VALUE** can be a vertical cell combined with two cells or those two cells not combined but sharing the same value.

Fig. 15: HelpComponent connected with a set of three different components (m-forms, field).

4.3.1. i18n-components (internationalization)

One of the main independent components created along *m-forms* is the internationalization one, by providing an easy but flexible interface to operate with, and a basic schema definition per component (also known as translation mappings) with a convention of `-lang` appended to the web component filename (without extension) under the same directory.

After working with other languages (*PHP*, *C++*) and its translation tools (*gettext* [35], *mo/po* files, *poeditor*), it served to be aware of difficulties like dependency on third-party tools and procedures, environment agnostic and really focused on the translators. One of the most determinant decisions was to establish a new schema definition for our internationalization. The main points are the necessity to provide language independent features

(eg: plural form depends on the language and modifies different words and/or parts of the word), dynamic and complex rules of translation (business logic, ordinal-dependent translation, etc) and an understandable, modifiable and exchangeable schema (avoiding binary files, allowing custom editors and vocabulary checkers). Also, the adaptation from plain systems (lists of sentences) to this system is trivial, and non technical people understand it just by seeing the English schema definition.

```

title: 'File Uploader',
cancel: 'Cancel',
message: {
  '1': 'Do you like to remove the selected item?',
  'default': 'Do you like to remove the {{number}} selected items?'
},
list: function(num, type){
  var msg = 'Should be listed ';
  if (num === 1){
    msg += 'the selected item?';
  }else{
    msg += `the ${num} selected ${type} items?`;
  }
  return msg;
}

```

Fig. 16: Types of translation mappings (portion of a schema definition).

It is wrapped as an independent package, and connected to *Werft-projects* to support optional building-time features, but designed with the flexibility and server-side efficiency in mind: *Webpack* packs it along with the component so it is attached to every piece of the system, and can be used (if the final developer wants) to allow language switching from client-side, avoiding new requests to the server. Also, it has been connected with *jade2mitrhil* to understand the special functions `_i` and generate wrappers to support the views' configuration with this module, establishing automatically the contexts and following the conventions defined in this package.

This component is designed to allow the maximum flexibility of utilization for every type of component (*m-forms*, *m-tooltip*, *m-toastr*, *Mithril Component Views*), usage in three steps (initialization, context setting and the translation itself), supports a simple but efficient pattern of filtering locales and provides three types of translation mappings.

The diagram of usage overview is appended in the appendices with the figure 32.

Also, the keys can be wrapped inside *literal objects*, and are not limited by nesting levels to define translation groups and organize more complex definitions. The implementation of *i18n-components* is appended to the appendices in the listing 3.

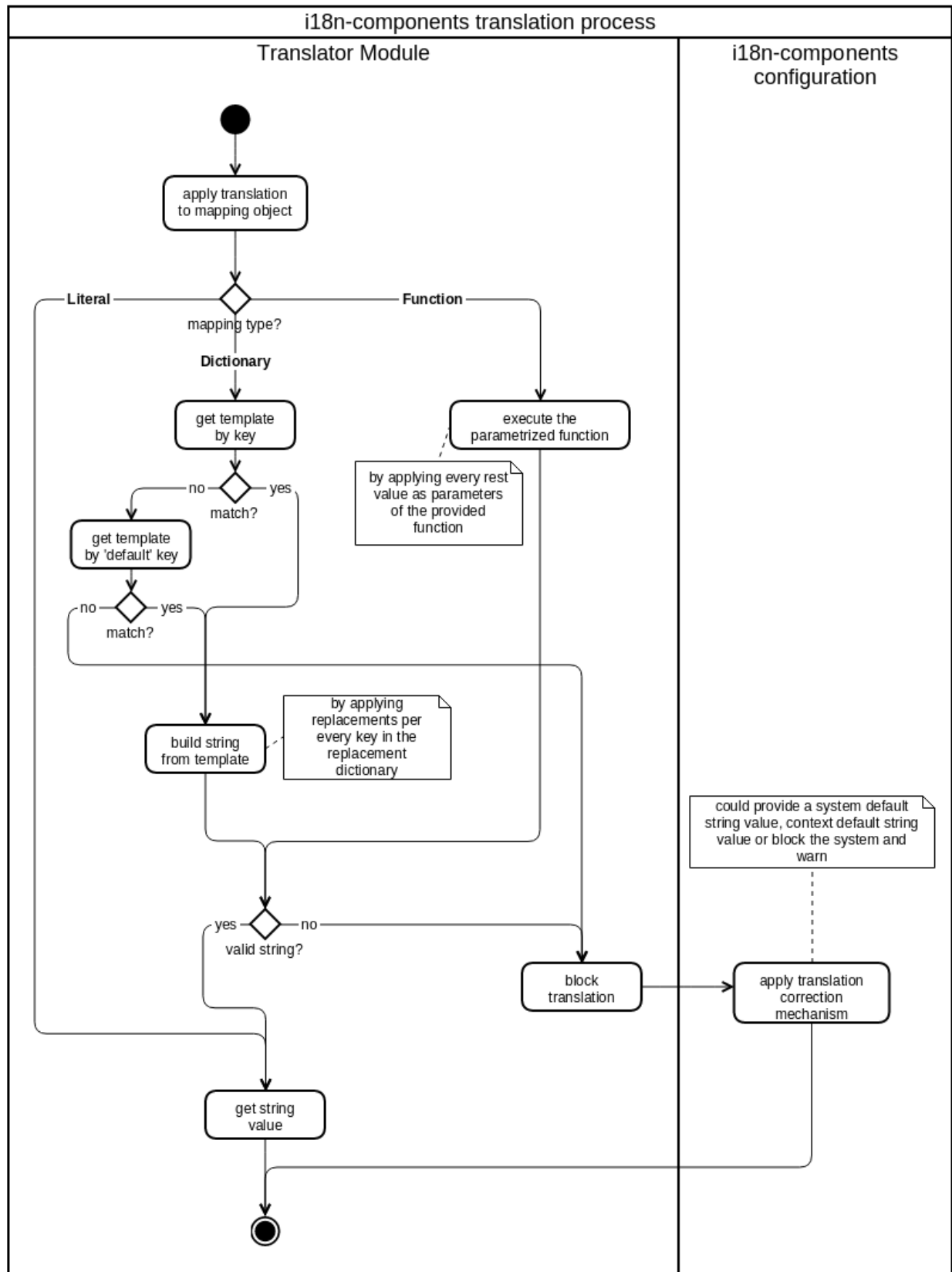


Fig. 17: i18n-components Activity diagram (translation).

The combination of some *Werft-projects* tasks (versatile template languages) with the design of this module is able to build specific bundles of languages saving approximately the ratio between the bundled languages and the available languages defined of space/memory and parsing time (bigger the schema definition, bigger the saving). Those results are measured comparing the activation of this new feature with its previous limited version in some `screens` used in production.

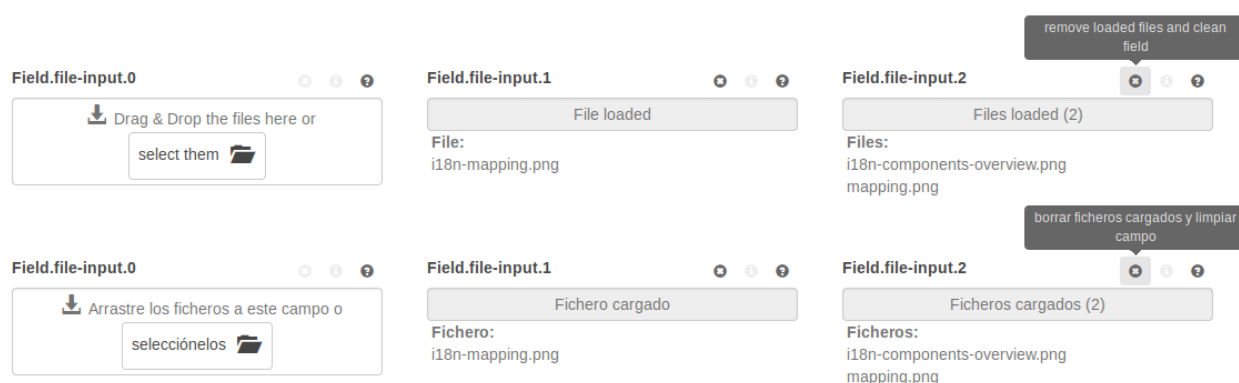


Fig. 18: GUI application (over `m-forms:field:file-input`, but interacting also with `m-tooltip` independent component and the `remove` form component).

Finally, one example of an `i18n-components` application over some web components (`m-forms` and `m-tooltip`) showing multiple UI states, and with literal (tooltip, title) and more complex dynamic (loaded files, file list) mappings. The appendix with the listing 4 has a full featured example (literal, dictionary and function mappings), its application in the listing 5 (parts of the script that uses it) and a *yasnippet* (listing 6) code created to start new schema definitions easily.

The translation module applied over the previous system and its `HelpComponent` can be seen in the figure 29 in the appendices and how it can be used and tested directly in different languages.

4.3.2. Helpers

Some created software to assist the web development process has been added to the listings 7 and 8, in the appendices. Both are source codes to be used with *Emacs*.

The first one is an extension to move and navigate through languages like *JavaScript*, *C* or *Java*, being the unique requirement to have curly braces, brackets or parenthesis in the code. It easily jumps to its pair, and helps during the development of files with high density of code or while configuring *JSON* files. The main advantages are that it does not matter that the code is broken in other parts, it works from the its own “level” (like closures) and starts searching in both ways.

Other part of this extension is the possibility to jump to *JavaScript* keywords, blocks and most important sentences. It provides a customizable interface to modify the *Regular Expressions* in case other developers wanted to extend the program for other languages.

The second one is a modification of the *Stylus* mode to help the designers when choosing and modifying layouts and colors in web applications. The editor will interpret sequences of hexadecimal colors and apply font decorations over them, avoiding to build and test the CSS with every change.



Fig. 19: Helpers in *Emacs*. Left side in *Stylus-mode* modified. Right side in *JavaScript* using *nav-blocks* (from top to bottom with just one key shortcut).

4.4. Phenotype Indicator

The phenotype has a description value that needs to be presented to the final user after being processed some Genetics data-sets (sample of the user) and transformed to be readable as natural language.

The solution provided is a new language completely focused on templating the results description and providing some features like special variables and functions that have been thought to be necessary after understanding a large number of specific results (more than 50 different real-world samples).

The advantages of the new system reduces the amount of text that the bioinformatics department need to write, and provides some help when creating the results to avoid serious commercial issues (customizing results). The redundancy of descriptions and possible paths per sample and phenotype have been drastically reduced, due to the usage of this new pattern and language.

The language have been designed with the flexibility in mind (multiple types of resulting values and combinations, more in the future), but also by providing a set of features for non-technical employees and with the usability as main factor.

Phenotype Indicator

Phenotype

Population (value)

User (offset value)

Template

Estimamos que la valoración realizada en su análisis y siempre refiriéndonos exclusivamente a este fenotipo de { PHENOTYPE } es{ IF-DIFF (un{ DIFF }veces) }{COMP-ALT}favorable que la población en general, desde el punto de vista de la práctica de actividades deportivas que impliquen una gran resistencia.

Messages

Output (case less)

Estimamos que la valoración realizada en su análisis y siempre refiriéndonos exclusivamente a este fenotipo de Glucemia en ayunas es un 1.22 veces menos favorable que la población en general, desde el punto de vista de la práctica de actividades deportivas que impliquen una gran resistencia.

Output (case equal)

Estimamos que la valoración realizada en su análisis y siempre refiriéndonos exclusivamente a este fenotipo de Glucemia en ayunas es igual de favorable que la población en general, desde el punto de vista de la práctica de actividades deportivas que impliquen una gran resistencia.

Output (case greater)

Estimamos que la valoración realizada en su análisis y siempre refiriéndonos exclusivamente a este fenotipo de Glucemia en ayunas es un 1.22 veces más favorable que la población en general, desde el punto de vista de la práctica de actividades deportivas que impliquen una gran resistencia.

Fig. 20: Phenotype Indicator screen, using the internal template-phenotype module.

Some of the usability advantages are: almost plain English variables and one-parameter functions (at least nowadays), function callings via nested operators (just levels) and neither space nor special ASCII character problems (users does not understand these problems).

It has been implemented using the *Interpreter* design pattern, because it is suited when parsing and representing a custom grammar in a hierarchical manner and due to its flexibility with simple languages [38]. This module has been ported to both server and client side, as refactorized as a module with services (figure 9 in the appendices). The above figure shows the simulation of the phenotype indicator under `m-forms` and being packed as a screen.

Template

```
Estimamos que la valoración realizada {} en su análisis y siempre refiriéndonos
exclusivamente a este fenotipo de { PHENOTYPE } es{
  IF-DIFF ( un DIFF-%) }{COMP-ALT}favorable que la población en general, desde
el punto de vista de la práctica de actividades deportivas que impliquen una gran
resistencia.
```

Messages

Unbalanced command separators (open - closed): '{' - '}': -1

Fig. 21: `template-phenotype` on-the-fly error detection, connected to a screen component to show the messages and with syntax transformation.

Also, to improve even more the usability of this module and being used as a web component by itself, it has been created two independent modules after analyzing the *CodeMirror* library. In the appendices can be found the listings 10 and 11. Those provide syntax highlighting and grammar recognition, usage of the services (error notifications and warnings) and smart autocompletion support.

Template

```
Estimamos que la valoración realizada {} en su análisis y siempre refiriéndonos
exclusivamente a este fenotipo de { PHE
  IF-DIFF ( un {DIFF-%}) }{COMP-ALT}f
el punto de vista de la práctica de ac
gran resistencia.
```

Messages

PHENOTYPE
DIFF
DIFF-%
COMP
COMP-ALT
VALUE
IF-DIFF()
IF-EQUAL()

Fig. 22: `template-phenotype` special mode and hinting system. Autocompletion features and automatic keyword guessing.

4.5. Foreign Function Interface (FFI)

The usage of previously built modules and libraries in languages like C/C++ has been possible because of wrappers (bindings) and connectors between those libraries and *Node.js* [37].

There are some complexities when writing libraries for *Node.js*, because it is using *Chrome's V8* and it needs its own stack of compiling tools and development procedures. It is built in C++, but own structures, macros and intrinsic difficulties related with the usage of the high performance evented I/O library [43].

The combination of high level language speed developments with the performance of low level languages can be achieved due to the connection between both libraries. The wrappers are thin layer to established with the low level language, that accesses complex compiler features like *OpenMP*, *OpenMPI* or *SIMD* operations (parallel-vector operations).

There are solutions like *Native Abstractions for Node.js (NaN)* that assist and creates a thin layer of abstraction when developing C++ for *Node.js* [39]. Although in some cases, when the task is getting complex (mixture of modes, events and external libraries like the above mentioned), it is better to return to pure *V8's C++*.

On the other side, small modules can be found to establish a quick connection with dynamic libraries using pure *JavaScript*, like *Foreign Function Interface* [40]. There are small overheads due its flexibility (dynamic function calls), and data that is passed and the execution code must be really optimized (the more data that needs to be processed in an independent way, the less overhead).

One of the advantages of having the *Chrome's V8* in the backside, is due to the support for complete different tool-sets and profiling softwares, like *Intel's VTune Amplifier* [42]. To be able to analyze the *JavaScript* code (*JIT* executor, [41]) from a C++ application (and also when building *Node.js* native modules like above) is necessary to optimize and configure appropriately the `node-gyp` building software, but the *Node.js* modified binary allows to inspect the internal *JavaScript* and optimize the most critic functions.

Related to the optimization, its techniques and applications it has been necessary to analyze multiple snippets and how the previous modules work (Genetysis®). Depending on the module, the load and utilization sometimes was not necessary to call other libraries or create native modules, but only optimize the *JavaScript* code. This procedures were complex, and determining how a high level language works, having a *JIT* compiler inside the engine (*V8*) is almost impossible, but with effort and resources can be improvements depending on the source code [11], [10].

One of the most interesting tiny portions of code created during this research is the optimization of loops based on caching techniques and procedures like *Duff's Device*, applied to *JavaScript* as has been shown in many places like Jeff Greenberg *Duff's Device* and [10]. Having studied this techniques even more and after a whole set of ideas have been applied over those algorithms, the final creation was even better than the publicly available *fast algorithms*, as demonstrated the benchmarks. Also, it was adapted to be usable by dynamic functions, like the battle-tested *Lodash* (former *Underscore*) library, but improving its performance as can be seen in the next figure.

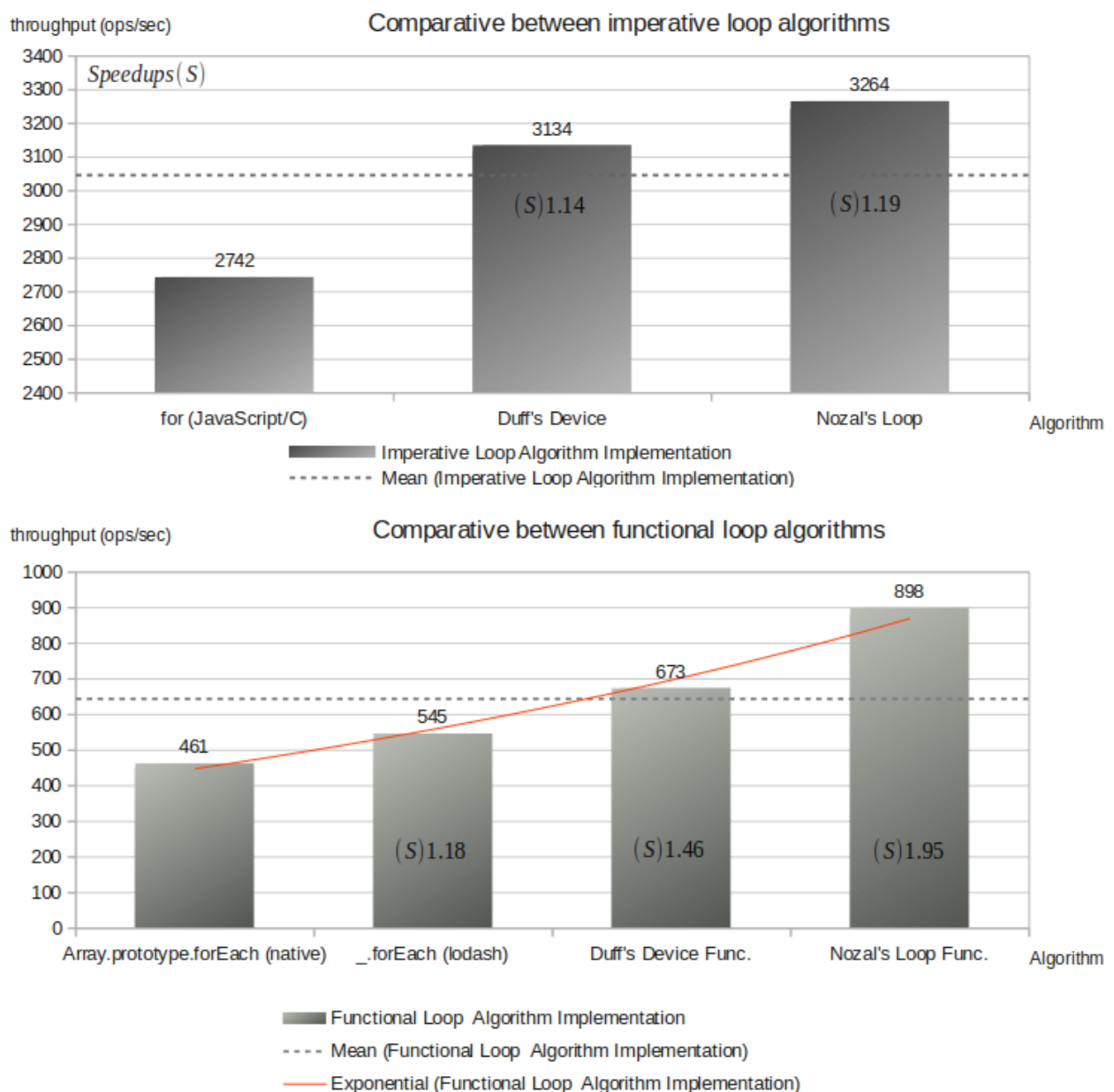


Fig. 23: Comparatives between loop algorithms (imperative and functional).

The *Duff's Device* implementation (and its adaptation to functional programming created just to compare it in benchmarks), and the *Nozal's Loop algorithms* have been added to the appendices in the listings 13 and 14, but here are shown its functional versions inside *Emacs*:

```

/*
 * Adaptation to its functional form
 * @author Jeff Greenberg (imperative)
 */
function duffsDeviceFn(arr, fn){
  var sum = 0;
  var a = arr;
  var l = a.length;
  var i = l % 8;
  while(i){
    fn(a[--i]);
  }
  i = Math.floor(l / 8);
  while(i--){
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
    fn(a[--l]);
  }
  return sum;
}

37 /*
38  * Functional form. Speedup (S) of 1.95
39  * @author Raúl Nozal
40  */
41 !function nozalsLoopFn(arr, fn){
42   var a = arr;
43   var l = a.length;
44   var lmod = l >> 2;
45   var li = lmod << 2;
46   var i, k;
47   ! for (i=0, k=0; i<lmod; i++) {
48     var a0 = a[k++],
49     !   a1 = a[k++],
50     !   a2 = a[k++],
51     !   a3 = a[k++];
52     fn(a0); fn(a1); fn(a2); fn(a3);
53   }
54   var rest = l - li;
55   switch(rest){
56     ! case 3: fn(a[li++]);
57     ! case 2: fn(a[li++]);
58     case 1: fn(a[li++]);
59   }
60 }

```

Fig. 24: *Jeff's Greenberg Duff's Device* and *Nozal's Loop* implementation, respectively.

Finally, and to show the flexibility of *FFI*, it has been created an experimental dynamic library in *Rust* (demonstration purposes), used successfully from *Node.js*. These adaptations and calls in most cases outperform the *JavaScript* analogue [44]. The written and tested code is in the listing 12 in the appendices.

These improvements have given more tools to the software department since so many new techniques and connections are possible. They can choose if do high-load processing in *C/C++* or *Rust*, optimize the matrices processing with the new *Nozal's Loop* or extend the previous web architecture and views' composition with the functional version (used intensively) or definitely choose a combination of optimizations after profiling the hot-spots and getting the best of each procedure by a balance of stability, maintainability and performance.

5. Conclusions and future work

In this section are described the conclusions obtained after describing the results of the Master Thesis and the future work that can be done after studying these advances.

5.1. Conclusions

The importance of getting involved in the processes of a company and its scientific knowledge areas is that it allows the understanding of how they work, what are the most conflicting points and what should be improved in favor of the process itself or the employees.

These improvements were not possible without the study of the Genetics Theoretical bases and getting involved with the bioinformatics department and its meetings. Also, the commercial point of view and how the software department analyzes every step in the Biosciences processing changed completely the way of work, from the very beginning of the idea, going through the mockups and accepted designs, towards the final implementation and its quality tests, with the end users.

Relative to the Genetysis ® processing, it is important to mark how changing the processing workflow, its stages and the organization of the data-sets and optimizing its storage improved the whole system. Databases reduced their load, and the data were divided and redefined. Understanding its paths and how it can be shared between many applications can promote the simulations, reusability of code and avoid failures in the system.

The building systems, usually seen as a side-needed tool and not really focusing on its development, serves not only for helping the success rates when creating new applications, but also reducing its costs (time, complexity, maintenance). Software departments can improve its development speeds, follow conventions easily and be completely centered in the application itself but with a boost of last technologies and methodologies, all being self-assisted in the background by the whole system.

Changing workflows and tools are not always painless, but when the performance-enhancing low-level code and its previous libraries can be reused in new high-level systems with just a small overhead, promotes the usage of these patterns and the improvements that can be achieved after profiling applications. Sharing code between client and server-side, and being able to increase the performance in web servers is one of the keys to follow this methodologies and use this advantages.

One of the advantages of having an iteration process of learning and inspecting the necessities, developing and designing the ideas, reviewing them by experts and finally, implementing them in a gradual way, allows the creation of architectures like the Genetics data administration software. Being in contact with the different departments and having in mind key features (usability, reutilization, flexibility) during the whole process are the

most importance lessons from these contributions.

Finally, having experiences in other Science fields, learning dozens of hundreds of new tools and libraries, feeling like a single explorer in front of more than a dozen of different languages and workflows, and working with professionals for more than one thousand hours is the happiest and most difficult challenge environment that anyone could be involved in.

Definitely the company has changed the workflows, tools and systems, but both bioinformatics and software departments have improved its activity and production quality from the very first time of establishment. The initial internship objectives have been completely achieved and even more studies, libraries and applications have been given to the employees than the initial requests.

5.2. Future work

Some of the improvements and future work that can be applied over its different projects depend completely on the company rhythms but, as an overview, these are some of its next works.

The building system is under continuous testing by the software department, and the developers (system's end users) will contribute to adapt new tasks to the system or add new *Werft* modules. A task for automatic formatting of code mixing the linting process will help to correct almost automatically every source code and follow the department conventions. Also, *Werft-projects* can be extended to support a *command-line* small application to manage its own *Werft* modules and projects, and being connected with tools like *git*, even modifying its pre-commit *hooks* and connect them to *Werft* tasks like linting and testing, to be sure that the repositories have only tested and reviewed code.

The *GUI* architecture and its design can be extended to support more components and features like abstract groups of events that trigger multiple components at the same time, develop an extension to debug appropriately *Mithril-Modules* components when using its *observers* from a component perspective, generate more layers of abstraction over the forms, allowing to adapt the architecture to new bigger containers like navigators, side-bars or even media types (videos, audios, maps). Also, profiling the architecture can promote the optimization of its parts and how the engine creates the component, from the model to the rendered element. By refactoring and extended its core feature, it could be adapted to support other low-level frameworks that render to the virtual *DOM*.

Related to the Genetics data administration software and two of its applications here exposed: matrix of polymorphisms combinations and phenotype indicator, both are under continuous improvement due to the R&D department, but the algorithms to build matrices were not studied and probably can be analyzed and apply parallel computation. Also,

the base values calculation from matrix operations can be extended easily with the interface provided, and the advantages of saving data storage can be benchmarked against pure storing, determining which technique is better.

The evolution of external interfaces and module adaption from different languages determines the way a project can extend its performance. By studying and being up to date with new *FFIs* and libraries that expose parallel computing can provide new speed boosts to the system. Connection to languages like *erlang* and its concurrent structures can improve the current web servers while reusing previous libraries and its shared modules.

6. Bibliography

- [1] James F. Kurose and Keith W. Ross. *Computer Networking. A Top-Down Approach* (Sixth Edition, Pearson, 2013). The Web and HTTP (caching, cookies). *Pages 124-142.*
- [2] Steve McConnell. *Code Complete. A practical handbook of software construction* (Second Edition, Microsoft Press, 2012). Code Improvements, System Considerations and Software Craftmanship. *Pages 463-853.*
- [3] Robert C. Martin. *Clean Code. A Handbook of Agile Software Craftmanship* (Prentice Hall, 2013). Meaningful Names, Functions, Comments, Objects and Data Structures and Unit Tests. *Pages 17-73, 93-101, 121-133.*
- [4] Robert C. Martin. *The Clean Coder. A Code of Conduct for Professional Programmers* (Prentice Hall, 2013). Chapters 1-14. *Pages 7-204.*
- [5] Jon Bentley. *Programming Pearls* (Second Edition, Addison-Wesley, 2012). Preliminaries and Performance (Data Structures Programs and Squeezing the space) *Pages 1-29, 99-108.*
- [6] Steve Krug. *Don't make me think* (Second Edition, New Riders, 2006). *Pages 10-179.*
- [7] Charles P. Pleege. *Security in Computing. Fourth Edition* (Fourth Edition, Prentice Hall, 2006). Program Security, Elementary Cryptography, Security in Networks
- [8] Michael K. Glass, Yann Le Scouarnec, Elizabeth Naramore, Gary Mailer, Jeremy Stolz and Jason Gerner. *Beginning PHP, Apache, MySQL Web Development* (Fourth Edition, Wrox, 2004). Validating User Input. User Logins, Profiles and Personalization. Building a Content Management System. *Pages 213-235, 335-433.*
- [9] Carlos Álvarez Martín y Pablo González Pérez. *Hardening de servidores GNU/Linux* (0xWORD, 2013). Fortificación de un entorno LAMP, Logging. *Pages 211-232, 261-271.*
- [10] Nicholas C. Zakas. *High Performance JavaScript (Build Faster Web Application Interfaces)* (O'Reilly Media, 2010). Data Access, Algorithms and Flow Control and Strings and Regular Expressions. *Pages 15-33, 61-106.*
- [11] Addy Osmani. *JavaScript Design Patterns (A JavaScript and JQuery Developer's Guide)* (O'Reilly Media, 2012). JavaScript Patterns, MV* Patterns and Modern Modular JavaScript Design Patterns *Pages 21-64, 79-120, 141-143, 148-151.*
- [12] Benjamin Lewin. *Genes IX* (Jones & Bartlett Learning, Ninth Edition, 2007). Genes Are DNA, Genome Sequences and Gene Numbers. *Pages 1-21, 76-95.*
- [13] Robert L. Nussbaum, Roderick R. McInnes and Huntignton F. Willard. *Thompson & Thompson Genetics in Medicine* (Saunders, Sixth Edition, 2012). Human Genetic Diversity: Mutation and Polymorphism, Genetic Variation in Populations.

- [14] Wang J, Lin M, Crenshaw A et al. *High-throughput single nucleotide polymorphism genotyping using nanofluidic dynamic arrays* (BMC Genomics, 2008).10:561
- [15] Andrew Glober. In pursuit of code quality: Don't be fooled by the coverage report (IBM).
<http://www.ibm.com/developerworks/library/j-cq01316/>
- [16] Ben Lewis. Measuring Client-Side JavaScript Test Coverage With Istanbul.
<https://blog.engineyard.com/2015/measuring-clientside-javascript-test-coverage-with-istanbul>
- [17] Carlos Oliveira. Why C++ will not die.
<http://coliveira.net/software/why-c-will-not-die/>
- [18] Lauren Orsini. Why do some old programming languages never die?.
<http://readwrite.com/2014/09/01/programming-language-coding-lifetime>
- [19] Git Foundation. Git Manual.
<http://www.git-scm.com>
- [20] Emacs Wiki Community. Emacs Wiki Organization.
<http://www.emacs-wiki.org>
- [21] Xah Lee. ErgoEmacs - Emacs-Lisp Documentation.
<http://www.ergoemacs.org>
- [22] Intel. Intel Intrinsics Guide (reference tool).
<https://software.intel.com/sites/landingpage/IntrinsicsGuide/>
- [23] Vivek N. Waghmare, Sandip V. Kendre and Sanket G. Chordiya. *Performance Analysis of Matrix-Vector Multiplication in Hybrid (MPI + OpenMP)* (Sandip Institute of Tech. & Research Centre, Nashik, India, 2011).International Journal of Computer Applications. Vol. 22 No. 5
- [24] ECMA Committee. ECMAScript Specifications (Editions) - MDN Language Resources.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources
- [25] Shari Thurow. 4 Things Online Marketers Should Know About User Experience (UX).
<http://marketingland.com/4-things-seos-know-user-experience-ux-73200>
- [26] Leo Horie. Mithril.js Benchmarks.
<https://lhorie.github.io/mithril/benchmarks.html>

- [27] W3C (MIT, ERCIM, Keio, Beihang). Cross-Origin Resource Sharing.
<http://www.w3.org/TR/cors/>
- [28] “kangax”. ECMAScript ES6 Compatibility Table.
<https://kangax.github.io/compat-table/es6/>
- [29] Alexander Shvets, Gerhard Frey and Marina Pavlova.. Chain of Responsibility (Behavioral Design Pattern).
https://sourcemaking.com/design_patterns/chain_of_responsibility
- [30] Biicode Community. Biicode: C++ dependency manager.
<https://www.biicode.com/>
- [31] Alexander Shvets, Gerhard Frey and Marina Pavlova.. Decorator (Structural Design Pattern).
https://sourcemaking.com/design_patterns/decorator
- [32] Alexander Shvets, Gerhard Frey and Marina Pavlova.. Builder (Creational Design Pattern).
https://sourcemaking.com/design_patterns/builder
- [33] Alexander Shvets, Gerhard Frey and Marina Pavlova.. Abstract Factory (Creational Design Pattern).
https://sourcemaking.com/design_patterns/builder
- [34] Heinrich Apfelmus. FRP - Three principles for GUI elements with bidirectional data flow.
<http://apfelmus.nfshost.com/blog/2012/03/29-frp-three-principles-bidirectional-gui.html>
- [35] Daiki Ueno (Free Software Foundation). GNU Gettext.
<https://www.gnu.org/software/gettext/gettext.html>
- [36] Sensio Labs Documentation Team. Twig. The flexible, fast and secure template engine for PHP (documentation).
<http://twig.sensiolabs.org/documentation>
- [37] Redis Development Group. hiredis-node (C bindings).
<https://github.com/redis/hiredis-node/blob/master/src/reader.cc>
- [38] Alexander Shvets, Gerhard Frey and Marina Pavlova.. Interpreter (Behavioral Design Pattern).
https://sourcemaking.com/design_patterns/interpreter
- [39] io.js Addon API Working Group. Native Abstractions for Node.js.
<https://github.com/nodejs/nan>

[40] Nathan Rajlich. Node.js Foreign Function Interface.

<https://github.com/node-ffi/node-ffi>

[41] Google. V8 JavaScript Engine.

<https://code.google.com/p/v8/>

[42] Intel. Intel VTune Amplifier 2016.

<https://software.intel.com/en-us/intel-vtune-amplifier-xe/try-buy>

[43] Nikhil Marathe. An Introduction to libuv.

<https://nikhilm.github.io/uvbook/introduction.html>

[44] Gergely Nemeth. How to Use Rust with Node.js When Performance Matters (micro).

<https://blog.risingstack.com/how-to-use-rust-with-node-when-performance-matters/>

7. Appendices

In this chapter are shown some of the contents recorded in the CD-ROM that is given as attachment with the report. They are appended to the document to be easily viewed (hyperlinks, syntax highlighting) when those contents are mentioned, improving the navigation and the review.

The CD has multiple source codes, images (rendered elements and diagrams), and the whole report building system with its *Org* files (in case anyone wants to extract and reuse texts easily) and the report itself in *PDF*.

7.1. Renders

7.1.1. GUI (m-forms)

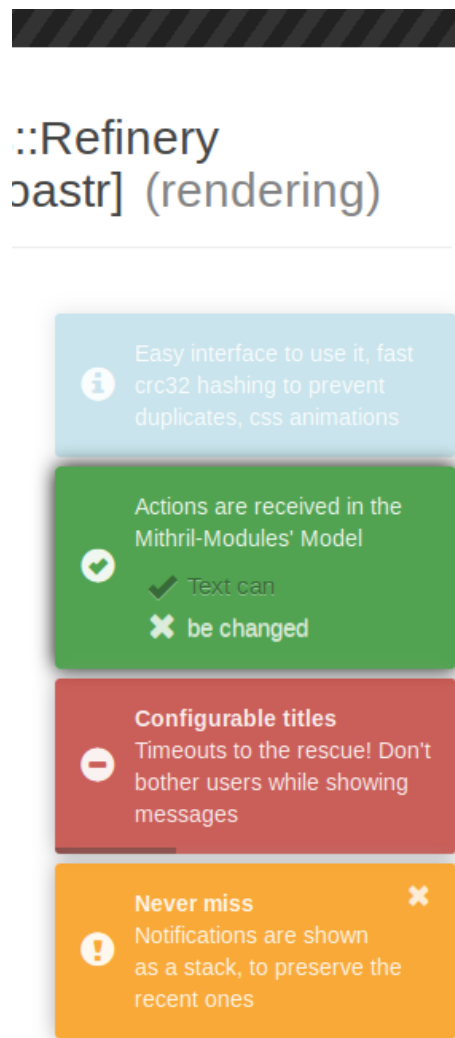


Fig. 25: m-toastr rendering process during a *Refinery* session.

werft-projects::Refinery [products/m-forms] (rendering)

Field.text ⓘ ⓘ

Field.text2 ⓘ ⓘ

Field.textarea ⓘ ⓘ

Algo un poco más largo

Field.textareacm ⓘ

Un texto cualquiera
Con algún salto de línea {DIFF} y
* Título1
** Título2
Algo más por /aquí/ y =sh bin=

Esta es la ayuda del componente superior (otra)

Field.textareacm ⓘ

1 * Título1
2 ** Título2
3 Algo más por /aquí/ y =sh bin=

Field.textareacm ⓘ ⓘ

Estimamos que la valoración realizada en su análisis y siempre refiriéndonos exclusivamente a este fenotipo de { PHENOTYPE } es{ IF-DIFF (un{ DIFF }veces) }{COMP-ALT}favorable que la

Field.listbox ⓘ ⓘ

Seleccione un color

Field.listbox2 ⓘ ⓘ

verde

Field.optionbox ⓘ ⓘ

☐ verde ☐ amarillo ☐ rojo

Field.optionbox2 ⓘ ⓘ

☐ verde ☐ amarillo ☒ rojo

Fig. 26: m-forms during a *Refinery* session, showing formComponent field (demonstration purposes).

Field.listbox-multiple

Seleccione tantos como quiera

- verde
- amarillo
- rojo

Field.listbox-multiple2

- verde
- amarillo
- rojo

Field.file-input

test

Drag & Drop the files here or select them

Field.file-input2

Files loaded (2)

Files:

- imagen.png
- notes.txt

Panel.steps

test

A B C D E F G H I

Panel.perc

Percentage

72%

Panel.timetracker

Estimated

Remaining

Panel.timetracker-3ways

Estimated

1h 40m

Remaining

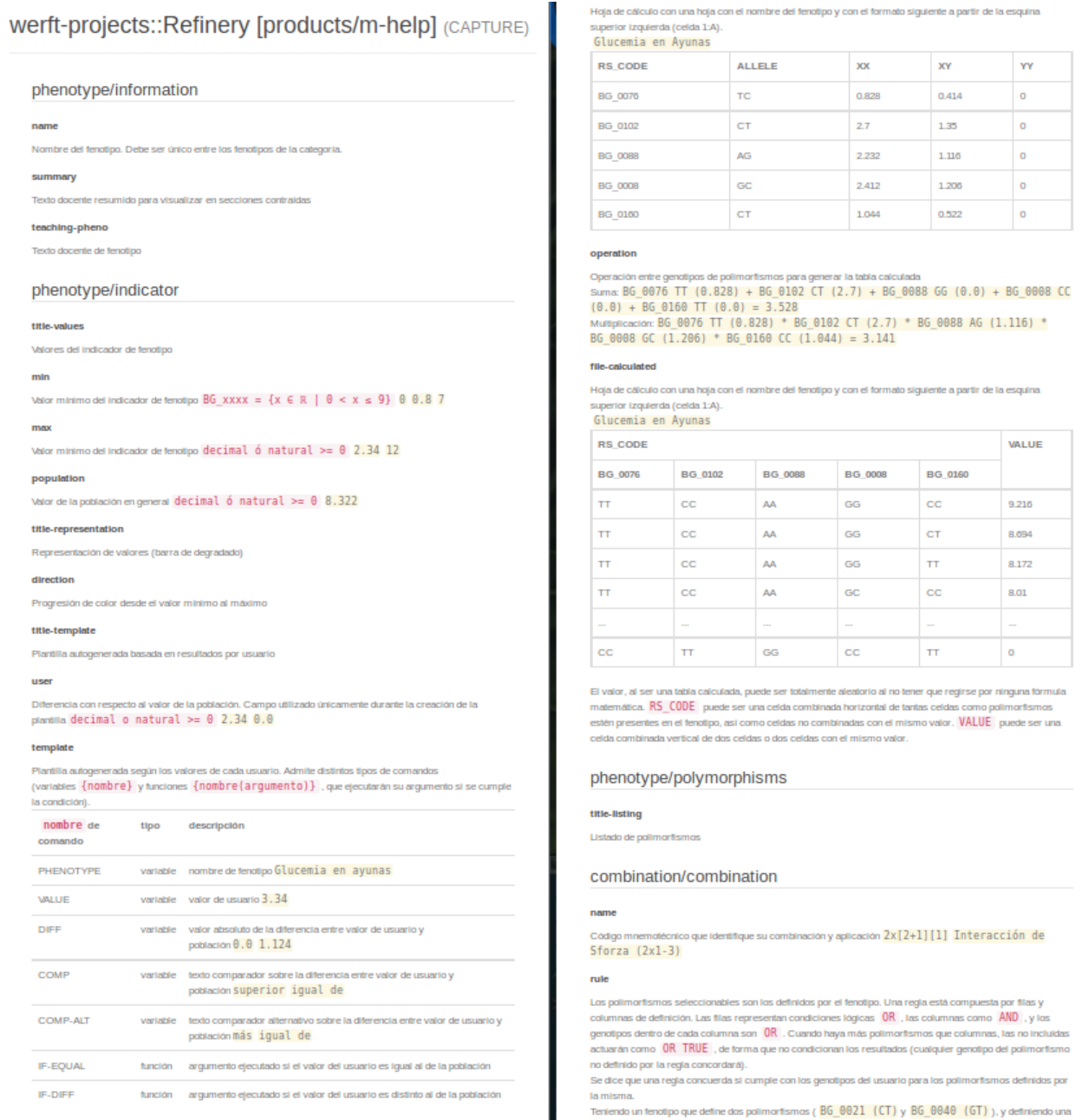
1h 10m

Logged

30m

Container.group0

Fig. 27: m-forms showing formComponent field, panel and container's title (demonstration purposes).



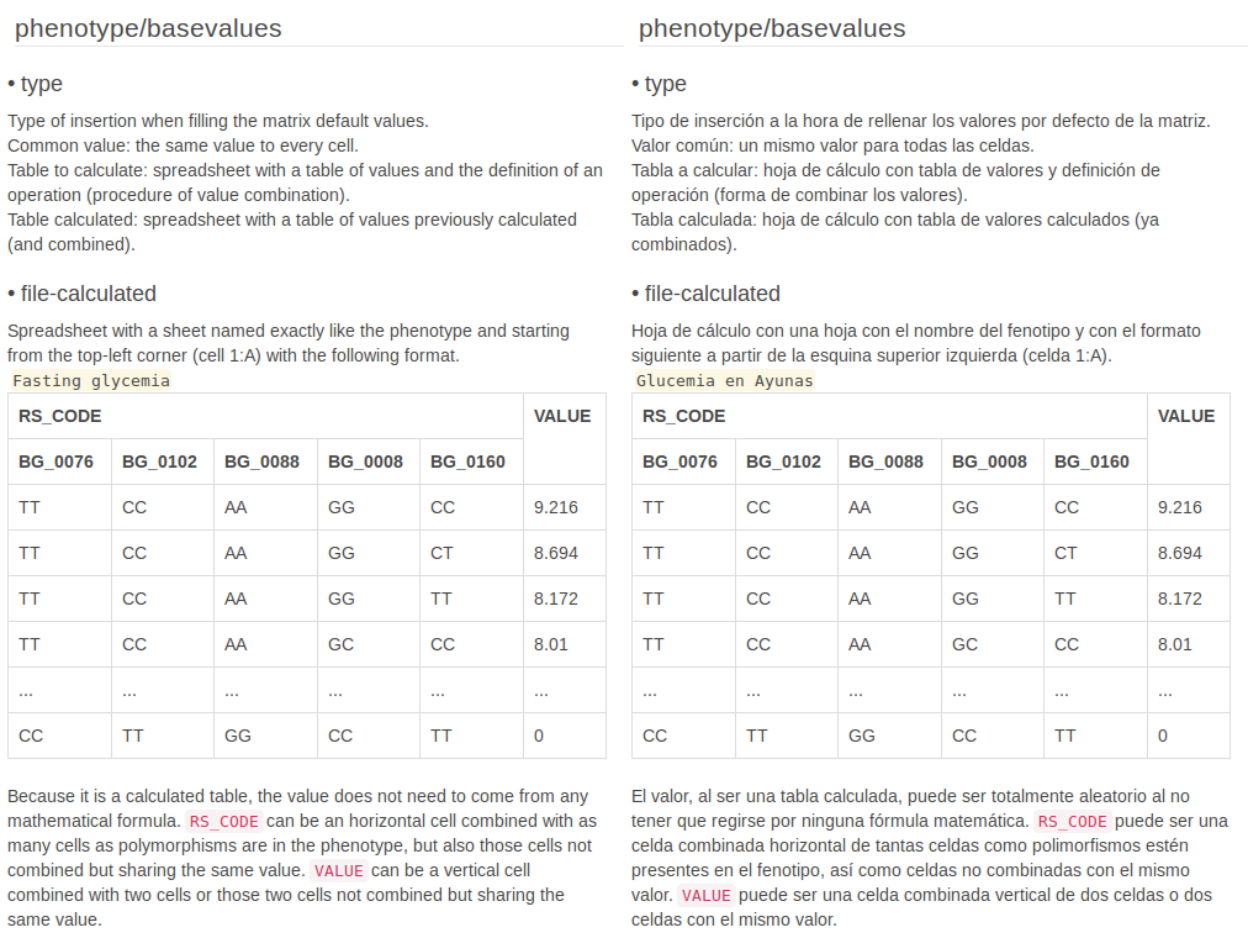


Fig. 29: HelpBlock showing its dynamic rendering (testing purposes).

Page 54 of 104

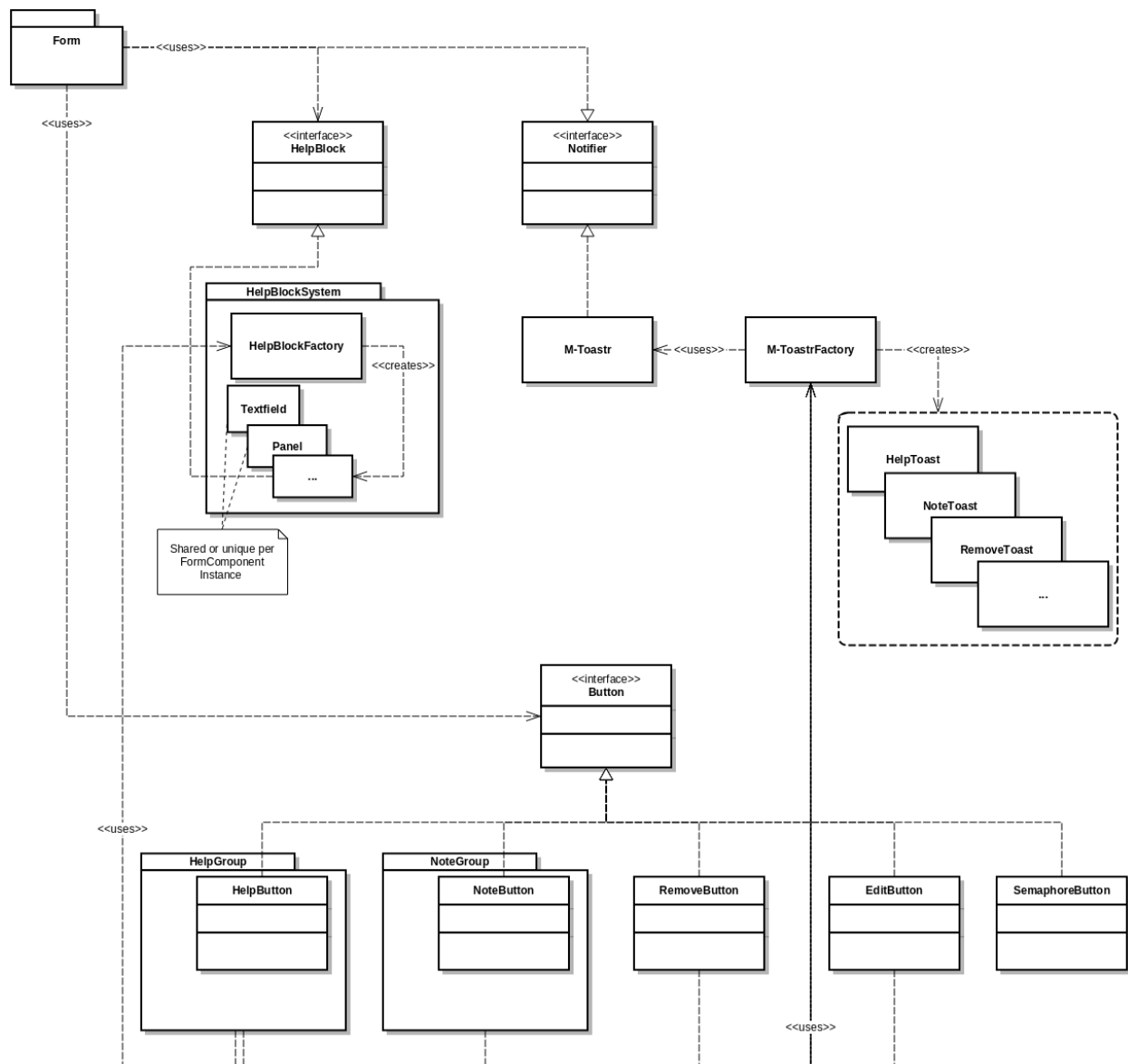


Fig. 30: m-forms independent components and pluggable modules.

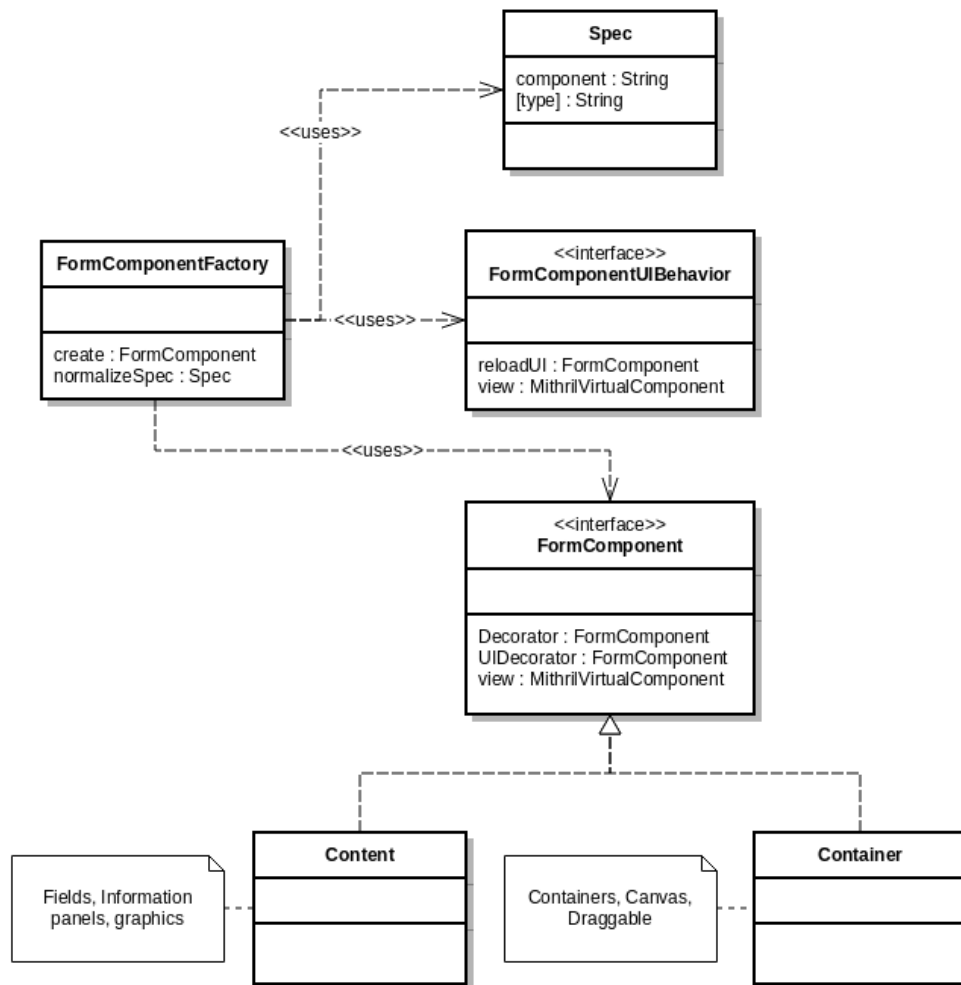


Fig. 31: m-forms system, formComponent components (containers and contents).

7.2.2. Internationalization

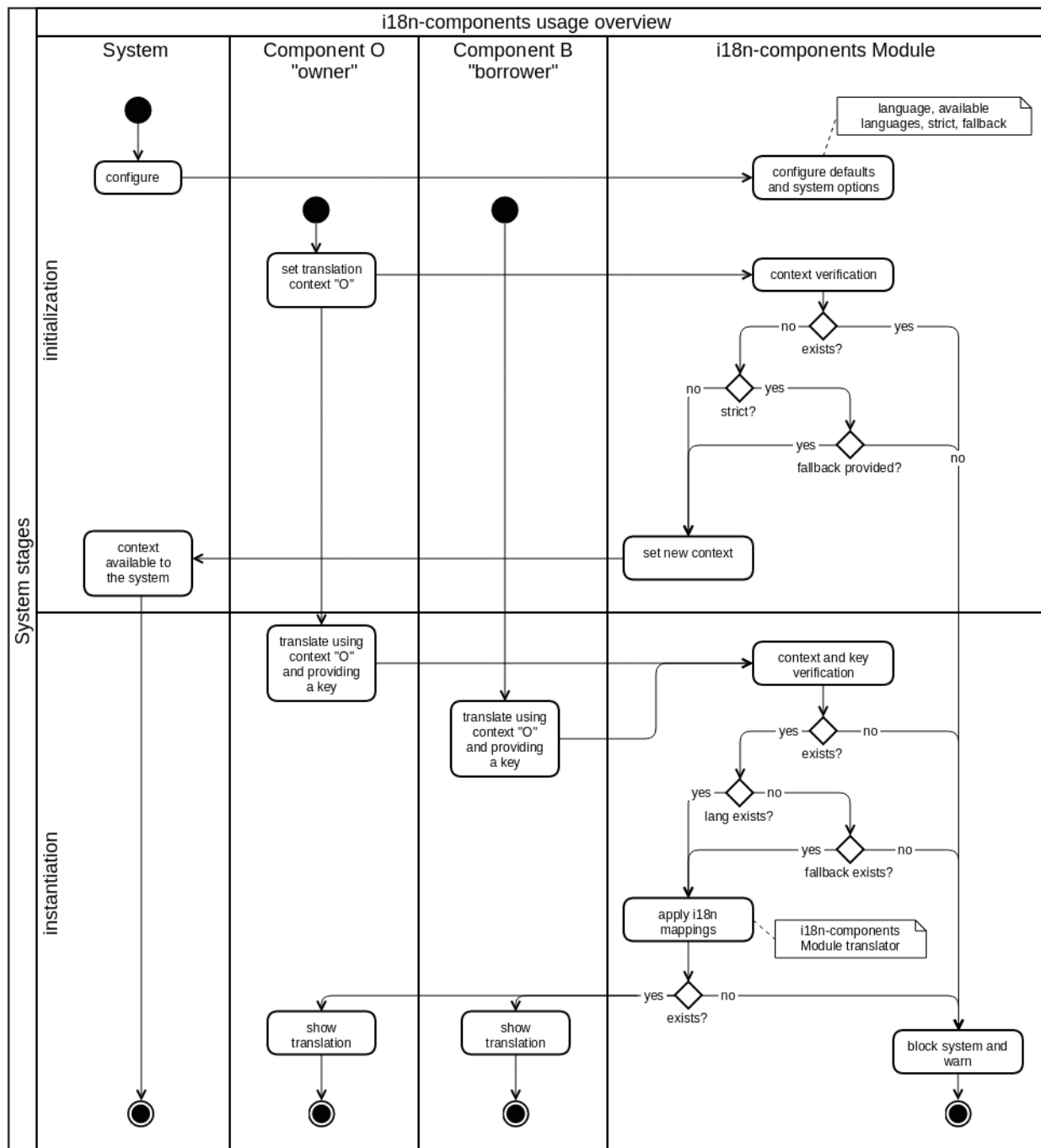


Fig. 32: i18n-components Activity diagram (usage overview).

7.3. Code

7.3.1. Matrix of combinations to rules

Listing 1: Polymorphism Rules (polymorphism-rules.js).

```

0  /**
1   * Behavioral Design Pattern Chain of Responsibility (loose coupling)
2   * default opts {
3   *   passAlways : true (to go through every handler)
4   * }
5   * @param {function} handler : handler function to call every `handle`
6   * @param {function} config : Handler configurator function
7   * @param {object} opts : default options
8   */
9  function Handler(handler, config, opts){
10     if (config){
11         config.bind(this)();
12     }
13     this.handler = handler || function(){};
14     this.opts = {
15         passAlways: opts && opts.passAlways ? opts.passAlways : true
16     };
17 }
18 Handler.prototype.setOpts = function(opts){
19     this.opts = {
20         passAlways: opts && typeof opts.passAlways === 'boolean' ? ↵
21             ↵opts.passAlways : true
22     };
23 /**
24  * Dynamic handle function (any arguments are processed by the ↵
25     ↵HandlerInstance.handler)
26  * Problem agnostic
27  * It will stop going through the handlers when passAlways is false and the ↵
28     ↵current value too.
29  * @returns {object} with current and any properties.
30  */
31 Handler.prototype.handle = function(){
32     var handledObj = {
33         current: false,
34         any: false
35     };
36     handledObj.current = this.handler.apply(this, arguments);
37     handledObj.any = (typeof handledObj.current === 'boolean') ? ↵
38         ↵handledObj.current || handledObj.any : false;
39     var next = this.next;
40     if (next){
41         if (!handledObj.current || this.opts.passAlways){
42             var handledObjRcv = next.handle.apply(next, arguments);

```

```

41         handledObj.current = handledObjRcv.current;
42         handledObj.any = handledObjRcv.any || handledObj.any;
43     }
44 }
45
46 return {
47     current: (typeof handledObj.current === 'boolean') ? ↵
48         ↵handledObj.current : false,
49     any: (typeof handledObj.any === 'boolean') ? handledObj.any : false
50 };
51
52 /**
53  * Establish the next handler of the chain
54  * @param {function} handler
55  */
56 Handler.prototype.setNext = function(handler){
57     this.next = handler;
58 };
59
60
61 /* Example of usage */
62
63 var mutationsInPhenotype = ["BG_0021", "BG_0040"];
64
65 /**
66  * Wrapper around the responses of the Handler
67  * to match the requested behavior for the Polymorphisms Rules
68  * @param {object} data : text and value properties (rule content)
69  * @param {array} rows : rows with objects, each of one with the ↵
70     ↵INTERNAL_CODE/CODIGO_INTERNO (SNP) as a property and its applied ↵
71     ↵genotypes as values.
72  * @param {object} genotypes : genotypes to be tested against (sample to ↵
73     ↵process the rules)
74  * @param {object} resolution : the applied results (combinations), with its ↵
75     ↵weights and values
76  * @returns true if the current rule has been applied with this sample
77  */
78 function processRule (data, rows, genotypes, resolution /* rw */){
79     var done = false;
80
81     for(var ri in rows){
82         var row = rows[ri];
83         var ruleWeight = 0;
84
85         var ANDmatches = true;
86         for(var mi in mutationsInPhenotype){
87             var mutationName = mutationsInPhenotype[mi];
88             var reqGenotypeName = genotypes[mutationName];
89             var mutation = row[mutationName];
90             if (reqGenotypeName && mutation){

```

```

87         ruleWeight++;
88         ANDmatches = ANDmatches && mutation[reqGenotypeName];
89     }
90 }
91
92     if (ANDmatches){
93         if (ruleWeight > 0){
94             done = true;
95             for(var el in data){
96                 var eldata = data[el];
97                 var resdata = resolution[el];
98                 if (eldata.active && ruleWeight >= resdata.weight ){
99                     resdata.value = eldata.value;
100                     resdata.weight = ruleWeight;
101                 }
102             }
103         }
104     }
105 }
106
107     return done;
108 }
109 }
110
111 /* Defined Rules */
112
113 var r1 = new Handler(function(genotypes, resolution){
114
115     var done = false;
116     var rows = [
117         {
118             BG_0021: { "CT": true, "CC": true}, // each prop in ↗
119                 ↘(INTERNAL_CODE/CODIGO_INTERNO) is OR
119             BG_0040: { "GG": true } // each prop in ROW is AND
120         },
121         {
122             BG_0021: { "TT": true }
123             // if has no prop in ROW, means TRUE
124         }
125     ];
126
127     var data = {
128         text: {
129             active: false,
130             value: ""
131         },
132         value: {
133             active: true,
134             value: 6.0
135         }
136     };

```

```
137
138     return processRule(data, rows, genotypes, resolution);
139
140 });
141
142 var r2 = new Handler(function(genotypes, resolution){
143
144     var done = false;
145     var rows = [
146         {
147             BG_0021: { "CC": true },
148             BG_0040: { "GT": true }
149         }
150     ];
151
152     var data = {
153         text: {
154             active: true,
155             value: "This rule has a text. It is specific for the CC/GT."
156         },
157         value: {
158             active: true,
159             value: 3.3
160         }
161     };
162
163     return processRule(data, rows, genotypes, resolution);
164
165 });
166
167
168 var r3 = new Handler(function(genotypes, resolution){
169
170     var done = false;
171     var rows = [
172         {
173             BG_0021: { "TT": true },
174             BG_0040: { "GG": true }
175         }
176     ];
177
178     var data = {
179         text: {
180             active: true,
181             value: "Text to be shown before the genes section (TT/GG)."
182         },
183         value: {
184             active: false,
185             value: 2.5
186         }
187     };
188 }
```

```

188
189     return processRule(data, rows, genotypes, resolution);
190
191 });
192
193 var r4 = new Handler(function(genotypes, resolution){
194
195     var done = false;
196     var rows = [
197         {
198             BG_0021: { "CC": true },
199             BG_0040: { "GT": true }
200         }
201     ];
202
203     var data = {
204         text: {
205             active: false,
206             value: "Another text, this for CC/GT."
207         },
208         value: {
209             active: true,
210             value: 4.4
211         }
212     };
213
214     return processRule(data, rows, genotypes, resolution);
215
216 });
217
218
219
220 /* the orchestrator */
221
222 var phenotypeProcessor = {
223
224     processRequest: function(genotypes){
225         var ruleChain = phenotypeProcessor.buildRules([r1, r2, r3, r4]);
226
227         var resolution = {
228             text: {
229                 value: null,
230                 weight: 0
231             },
232             value: {
233                 value: null,
234                 weight: 0
235             }
236         };
237
238         var handled = ruleChain.handle(genotypes, resolution);

```

```
239         // if passAlways, the last handle send the returned boolean from the ↗
           ↳last rule.
240
241         return resolution;
242     },
243     buildRules: function(rules){
244         var ruleChain = rules[0];
245         var lastRule = rules.length - 1;
246         for(var i = 0; i < lastRule; i++) {
247             rules[i].setNext(rules[i+1]);
248         }
249
250         return ruleChain;
251     }
252 };
253
254
255
256 /* example main executor */
257
258 var executor = function(){
259     var genotypes, resolution;
260
261     genotypes = {
262         "BG_0021": "CC",
263         "BG_0040": "GT"
264     };
265
266     resolution = phenotypeProcessor.processRequest(genotypes);
267     console.log(resolution);
268
269     genotypes = {
270         "BG_0021": "TT",
271         "BG_0040": "GT"
272     };
273
274     resolution = phenotypeProcessor.processRequest(genotypes);
275     console.log(resolution);
276 };
277
278 executor();
279
280 /*
281 {
282     text: { value: 'This rule has a text. It is specific for the CC/GT.', ↗
           ↳weight: 2 },
283     value: { value: 4.4, weight: 2 }
284 }
285 {
286     text: { value: null, weight: 0 },
287     value: { value: 6, weight: 1 }
```

```
288 | }  
289 | */
```

7.3.2. GUI independent packages

Listing 2: CStr (cstr.js).

```
0 function CStrPortion(portion){  
1     if (typeof portion !== 'object'){  
2         portion = {text: portion};  
3     }  
4     this.portion = portion;  
5     // this.type  
6 }  
7 CStrPortion.prototype.isAnyOf = function(types){  
8     var t = this.recognize();  
9     if (typeof types !== 'object'){  
10         types = [types];  
11     }  
12     return types.indexOf(t) !== -1;  
13 };  
14 CStrPortion.prototype.accessor = function(param){  
15     this.recognize();  
16     var ret;  
17     if (param === 'self'){  
18         ret = this;  
19     }else if (param === 'type'){  
20         ret = this.type;  
21     }else{  
22         ret = this.portion[param];  
23     }  
24     return ret;  
25 };  
26 CStrPortion.prototype.recognize = function(){  
27     var type = this.type;  
28     if (type == null){  
29         var p = this.portion;  
30         if (p.text != null){  
31             type = 'text';  
32         }else if (p.newline != null){  
33             type = 'newline';  
34         }else if (p.header != null){  
35             type = 'header';  
36         }else if (p.style != null){  
37             type = 'textstyle';  
38         }else if (p.comment != null){  
39             type = 'comment';  
40         }else if (p.url != null){ // p.name not mandatory
```

```
41         type = 'link';
42     }else{
43         throw new Error(`CStrPortion has an invalid type: ${p}`);
44     }
45     this.type = type;
46 }
47 return type;
48 };
49 CStrPortion.prototype.toString = function(){
50     var ret;
51     var p = this.portion;
52     var t = this.recognize();
53     var opt;
54     switch(this.type){
55     case 'text':
56         ret = p.text;
57         break;
58     case 'newline':
59         ret = '\n';
60         break;
61     case 'header':
62         ret = p.header + ' ' + p.value;
63         break;
64     case 'comment':
65         ret = p.comment + ' ' + p.value;
66         break;
67     case 'textstyle':
68         var char = CStrPortion.prototype.textstyle.recognize(p.style);
69         if (char){
70             ret = char + p.value + char;
71         }else{
72             ret = p.value;
73         }
74         break;
75     case 'link':
76         opt = p.name;
77         if (opt != null){
78             opt = '[' + opt + ']';
79         }else{
80             opt = '';
81         }
82         ret = '[' + p.url + ']' + opt;
83         break;
84     }
85     return ret;
86 };
87
88
89 /* CStrPortionProcesses (begin) */
90 CStrPortion.prototype.newline = function(cParts){
91     var v;
```



```

92     if (this.isAnyOf('text')){
93         v = this.portion.text;
94     }
95
96     if (v && v.indexOf('\n') !== -1){
97         var parts = v.split('\n');
98         var i, leni = parts.length,
99         ei;
100
101         var hasContent;
102         for (i=0; i<leni; i++){
103             ei = parts[i];
104
105             if (hasContent){
106                 cParts.push({'newline': true});
107             }else{
108                 hasContent = true;
109             }
110             var notEmpty = ei.length;
111             if (notEmpty){
112                 cParts.push({text: ei});
113             }
114         }
115     }
116 };
117
118 var reStyle = {
119     code: /^[\\s\\(\\[\\{]=([\\^=]+)=$/ ,
120     codeG: /[\\s\\(\\[\\{]=([\\^=]+)=/g,
121     bold: /^[\\s\\(\\[\\{]\\*(\\^[\\*]+)\\*$/ ,
122     boldG: /[\\s\\(\\[\\{]\\*(\\^[\\*]+)\\*/g,
123     underline: /^[\\s\\(\\[\\{]_([\\^_]+)_$/ ,
124     underlineG: /[\\s\\(\\[\\{]_([\\^_]+)_/g,
125     italic: /^[\\s\\(\\[\\{]\\/(\\^[\\^/]+)\\/$/ ,
126     italicG: /[\\s\\(\\[\\{]\\/(\\^[\\^/]+)\\//g,
127 };
128 CStrPortion.prototype.textstyle = function(cParts, subtype){ // code, italic, ↵
129     ↵bold, underline
130     var v;
131     if (this.isAnyOf('text')){
132         v = this.portion.text;
133     }
134
135     var char = CStrPortion.prototype.textstyle.recognize(subtype);
136
137     if (v && char && v.indexOf(char) !== -1){ // faster
138
139         re = reStyle[subtype];
140         reG = reStyle[subtype + 'G'];
141
142         if (re && reG){

```

```

142         var matches = v.match(reG);
143         var re, reG;
144         if (matches){
145
146             var styles = [];
147             var m;
148             var i, leni, ei;
149             for (i=0, leni=matches.length; i<leni; i++){
150                 ei = matches[i];
151                 m = ei.match(re);
152                 if (m){
153                     var all = m[0];
154                     v = v.replace(all.substr(1), '\0');
155                     var text = m[1];
156                     var stPart = {style: subtype, value: text};
157                     styles.push(stPart);
158                 }
159             }
160
161             var texts = v.split('\0');
162             var i, leni, ei;
163             for (i=0, leni=texts.length; i<leni; i++){
164                 ei = texts[i];
165                 var notEmpty = ei.length;
166                 if (notEmpty){
167                     cParts.push({text: ei});
168                 }
169                 var style = styles[i];
170                 if (style){
171                     cParts.push(style);
172                 }
173             }
174         }
175     }
176 }
177 };
178 CStrPortion.prototype.textstyle.recognize = function(style){
179     var char;
180     switch(style){
181     case "code": char = '='; break;
182     case "italic": char = '/'; break;
183     case "bold": char = '*'; break;
184     case "underline": char = '_'; break;
185     }
186     return char;
187 };
188
189 var reOrgHeader = /^(\s+)\s+(.+)$/;
190 CStrPortion.prototype.header = function(cParts){
191     var v;
192     if (this.isAnyOf('text')){

```

```

193     v = this.portion.text;
194 }
195
196 if (v && v.indexOf('*') === 0){ // first char should be *
197     var m = v.match(reOrgHeader);
198     if (m){
199
200         var header = m[1];
201         var value = m[2];
202         cParts.push({header: header, value: value});
203     }
204 }
205 };
206
207 var reOrgComment = /(?:(#\[^\ ]+)|(#) )?(.+)$/;
208 CStrPortion.prototype.comment = function(cParts){
209     var v;
210     if (this.isAnyOf('text')){
211         v = this.portion.text;
212     }
213
214     if (v && v.indexOf('#') !== -1){
215         var m = v.match(reOrgComment);
216         if (m){
217             var comment = m[1] || m[2]; // 1 special, 2 normal
218             var value = m[3];
219             cParts.push({text: v.substr(0, m.index)});
220             cParts.push({comment: comment, value: value});
221         }
222     }
223 };
224
225
226 var reOrgLink = /^\[([^\]]*)(?:\[([^\]]*)\]|)\]\$/;
227 var reOrgLinkG = /\[([^\]]*)(?:\[([^\]]*)\]|)\]\//g;
228 CStrPortion.prototype.link = function(cParts){
229     var v;
230     if (this.isAnyOf('text')){
231         v = this.portion.text;
232     }
233
234     if (v && v.indexOf('[') !== -1){
235         var matches = v.match(reOrgLinkG);
236         if (matches){
237
238             var urls = [];
239             var m;
240             var i, leni, ei;
241             for (i=0, leni=matches.length; i<leni; i++){
242                 ei = matches[i];
243                 m = ei.match(reOrgLink);

```

```

244         if (m){
245             v = v.replace(m[0], '\0');
246             var url = m[1];
247             var name = m[2] || url;
248             var urlPart = {url: url};
249             if (url !== name){
250                 urlPart.name = name;
251             }
252             urls.push(urlPart);
253         }
254     }
255
256     var texts = v.split('\0');
257     var i, leni, ei;
258     for (i=0, leni=texts.length; i<leni; i++){
259         ei = texts[i];
260         var notEmpty = ei.length;
261         if (notEmpty){
262             cParts.push({text: ei});
263         }
264         var url = urls[i];
265         if (url){
266             cParts.push(url);
267         }
268     }
269
270     }
271 }
272 };
273
274 CStrPortion.prototype.trim = function(cParts, subtype){
275     var v;
276     if (this.isAnyOf('text')){
277         v = this.portion.text;
278     }
279
280     if (v){
281         var befLen = v.length;
282         switch (subtype) {
283             case "left":
284                 v = v.replace(/^ */, '');
285                 break;
286             case "right":
287                 v = v.replace(/ *$/, '');
288                 break;
289             default:
290                 v = v.replace(/ *$/, '').replace(/^ */, '');
291         }
292         if (befLen !== v.length){ // changed
293             cParts.push({text: v});
294         }

```

```

295     }
296 };
297
298 CStrPortion.prototype.transform = function(cParts, subtype){
299     var v;
300     if (this.isAnyOf('text')){
301         v = this.portion.text;
302     }
303
304     if (v){
305         var befLen = v.length;
306         if (v.indexOf(' ') !== -1){ // changed
307
308             switch (subtype) {
309                 case "entity-spaces":
310                     v = v.replace(/ /g, '\u00a0');
311                     break;
312             }
313
314             cParts.push({text: v});
315         }
316     }
317 };
318 /* CStrPortionProcesses (end) */
319
320
321
322 function CStrBuffer(value){
323     this.value = [{text: value}];
324     this.i = 0;
325 }
326 CStrBuffer.prototype.filter = function(type){
327     var f;
328     if (type === 'clean-comments-newlines'){
329         var cStrPs = []; // saved
330         var saved;
331         var withComment;
332         f = function(acc, strs){
333             var type = acc('type');
334             var ret;
335             var save;
336             var dump;
337             if ((type === 'newline' ||
338                 (type === 'text' && acc('text').length === 0))){
339                 save = true;
340             }else if (type === 'comment'){
341                 withComment = true;
342             }else{
343                 ret = true;
344                 dump = true;
345             }

```

```

346
347         if (save){
348             cStrPs.push(acc('self'));
349             saved = true;
350         }
351         if (dump && saved){
352             f.dump(strs);
353             cStrPs = [];
354             saved = false;
355             withComment = false;
356         }
357         return ret;
358     };
359     f.dump = function(strs){
360         if (withComment){
361             if (withComment){
362                 var rep = new CStrPortion({newline: true});
363                 strs.push(rep.toString());
364             }else{
365                 cStrPs.forEach(function(el){ // insert the saved ones
366                     strs.push(el.toString());
367                 });
368             }
369         }
370     };
371 }
372 }
373 return f;
374 };
375 CStrBuffer.prototype.toString = function(filter){
376     var v = this.value;
377     var strs = [];
378     filter = CStrBuffer.prototype.filter(filter);
379     var f;
380     var fres;
381     var i, leni, ei;
382     for (i=0, leni=v.length; i<leni; i++){
383         ei = v[i];
384         var cStrP = new CStrPortion(ei);
385         if (!filter){
386             strs.push(cStrP.toString());
387         }else{
388             fres = filter((param) => cStrP.accessor(param),
389                 strs);
390             if (fres){
391                 strs.push(cStrP.toString());
392             }
393         }
394     }
395     if (filter){ // last checking of filters
396         filter.dump(strs);

```

```

397     }
398     return strs.join('');
399 };
400 CStrBuffer.prototype.insert = function(val, pos, rep){
401     if (!pos){ pos = 0; }
402     if (!rep){ rep = 0; }
403     var l = val.length;
404     var v = this.value,
405         i = this.i;
406     if (l){ // only arrays
407         var first;
408         for (var j=0; j<l; j++){
409             if (first){
410                 rep = 0;
411             }else{
412                 first = true; // we just remove the parent
413             }
414             var CStrP = new CStrPortion(val[j]); // convert it
415             v.splice(i + pos, rep, CStrP.portion);
416             pos++;
417         }
418     }
419 };
420 CStrBuffer.prototype.prepend = function(val){
421     this.insert(val, 0, 0);
422 };
423 CStrBuffer.prototype.replace = function(val){
424     this.insert(val, 0, 1);
425 };
426 CStrBuffer.prototype.append = function(val){
427     this.insert(val, 1, 0);
428 };
429 CStrBuffer.prototype.checkBounds = function(from, to){
430     var vl = this.value.length;
431     if ((from < 0 || from > vl) && (to < from || to > vl)){
432         throw new Error(`CStrBuffer checkBounds error (from - to): ${from} - ↯
433             ↳${to}`);
434     }
435 };
436 CStrBuffer.prototype.run = function(from, to, op){
437     var v = this.value,
438         vlen = v.length;
439     if (from != null){
440         if (to != null){
441             to = vlen;
442         }
443         this.checkBounds(from, to);
444     }else{
445         from = 0;
446         to = vlen;
447     }

```

```

447     var i, leni, e;
448     for (i=from, leni=to; i<leni; i++){
449         e = v[i];
450         this.i = i;
451         var displaced = op.call(this, e);
452         if (displaced){ // != 0
453             leni += displaced;
454             i += displaced;
455         }
456     }
457 };
458 CStrBuffer.prototype.replacer = function(type, subtype, from, to){
459     switch(type){
460     case "newline":
461     case "link":
462     case "header":
463     case "comment":
464     case "textstyle":
465     case "trim":
466     case "transform":
467         break;
468     default:
469         throw new Error(`CStrBuffer replacer does not support the type: ↵
470             ↵${type}`);
471     }
472     this.run(from, to, function(e){ // this
473         var disp = 0;
474         var cStrP = new CStrPortion(e);
475         var cParts = [];
476         cStrP[type](cParts, subtype);
477         var len = cParts.length;
478         if (len){ // 3 to add
479             this.changed = true;
480             this.replace(cParts); // determinant
481             disp = len - 1; // 2 displaced
482         }
483         return disp;
484     });
485 };
486
487 function CStr(value){
488     var value,
489         withFilter;
490
491     var b = this.buf = new CStrBuffer(value);
492
493     this.toString = function(filter){
494         if (b.changed || filter !== withFilter){
495             if (filter){
496                 value = b.toString(filter);

```



```

497         withFilter = filter;
498     }else{
499         value = b.toString();
500         withFilter = null;
501     }
502     b.changed = false;
503 }
504 return value;
505 };
506 }
507 CStr.prototype.push = function(str, type){
508     if (typeof str !== 'object'){
509         str = [str];
510     }
511     var b = this.buf;
512     if (type === 'right'){
513         b.i = b.value.length - 1;
514         b.append(str);
515     }else{
516         b.i = 0;
517         b.prepend(str);
518     }
519     b.changed = true;
520 };
521 CStr.prototype.prepend = function(str){
522     this.push(str, 'left');
523 };
524 CStr.prototype.append = function(str){
525     this.push(str, 'right');
526 };
527 CStr.prototype.buffer = function(type, subtype){
528     var b = this.buf;
529
530     var op;
531     switch(type){
532     case '\n':
533     case 'newline':
534     case 'newlines':
535         b.replacer('newline');
536         break;
537     case 'links':
538     case 'link':
539     case 'URI':
540         b.replacer('link');
541         break;
542     case 'header':
543     case 'head':
544     case '*':
545         b.replacer('newline'); // triggers
546         b.replacer('header');
547         break;

```

```

548     case 'comment':
549     case '#':
550         b.replacer('newline'); // triggers
551         b.replacer('comment');
552         break;
553     case 'textstyle':
554     case 'style':
555         b.replacer('textstyle', subtype);
556         break;
557     case 'trim':
558         b.replacer('trim', subtype);
559         break;
560     case 'transform':
561         b.replacer('transform', subtype);
562         break;
563     }
564 };
565
566 CStr.prototype.map = function(fn){
567     if (typeof fn !== 'function'){
568         throw new Error('CStr map expects a mapper function');
569     }
570     var buf = this.buf;
571     buf = buf.value;
572     var mapped;
573     if (buf && buf.length){
574         mapped = [];
575         var i, leni, ei;
576         for (i=0, leni=buf.length; i<leni; i++){
577             ei = buf[i];
578
579             var cStrP = new CStrPortion(ei);
580             var el = fn( (param) => cStrP.accessor(param) );
581             if (el !== null){
582                 mapped.push(el);
583             }
584         }
585     }
586     return mapped;
587 };
588
589 module.exports = CStr;

```

7.3.3. i18n-components

Listing 3: i18n-components (i18n-components.js).

```
0 | var lang = 'en',
```

```

1      fallback = 'en',
2      strict = true; // check in every context if fallback exists
3
4      const available = ['en', 'es'];
5
6      const ctxs = {
7
8      };
9
10     /**
11     * Translate using the context name, with a
12     *
13     * @param {} ctxkey : context name
14     * @param {} key : key of translation (dot notation string)
15     * @param {} rest : any other parameter to pass (function|dictionary mappings ↵
16     *                  ↵only)
17     * @returns {} the string translate
18     * @throws {} Error when the context does not exist, the key does not exist, ↵
19     *                  ↵does not have fallback, is not in the availables
20     */
21     function _i(ctxkey, key, ...rest){
22         var ctx = ctxs[ctxkey];
23         if (!ctx){
24             throw new Error(`'${ctxkey}' is not a loaded context.`);
25         }
26         var lmap = ctx[lang];
27         var msg;
28         var err;
29
30         function index(obj, i){ return obj[i]; }
31         function dotpick(obj, dotstr){ return dotstr.split('.').reduce(index, ↵
32             ↵obj); }
33         if (lmap){
34             msg = dotpick(lmap, key);
35         }
36         if ((!lmap || !msg) && lang !== fallback){
37             if (fallback === false){
38                 throw new Error(`'${key}' is not in the '${ctx.name}' context. ↵
39                     ↵Language not provided: '${lang}'.`);
40             }
41             lmap = ctx[fallback];
42             if (lmap){
43                 msg = dotpick(lmap, key);
44             }
45         }
46         if (!lmap || !msg){
47             throw new Error(`'${key}' is not in the '${ctx.name}' context. ↵
48                 ↵Languages not provided: nor '${lang}' neither '${fallback}' ↵
49                 ↵fallback.`);
50         }
51     }
52 }

```

```

46     if (typeof msg === 'function'){
47         msg = msg(...rest);
48     }
49
50     if (typeof msg === 'object'){
51         var tempkey = rest[0];
52
53         var template = msg[tempkey];
54         if (!template){
55             template = msg.default;
56         }
57         if (template){
58
59             var vars = rest[1];
60             for(var k in vars){
61                 var re = new RegExp('{{ *' + k + ' *}}', 'g');
62                 template = template.replace(re, vars[k]);
63             }
64             msg = template;
65         }
66     }
67
68     return msg;
69 }
70
71 /**
72  * Check if the keys provided in the schema definition are valid
73  * @param {} a | {key1: '', key2: '', ...}
74  * @throws {} Error when a key is not valid
75  */
76 function checkI18nKeys(a){
77     for(var k in a){
78         if (a.hasOwnProperty(k) && k.indexOf('.') !== -1){
79             throw new Error(`${k} is an invalid i18n dictionary key ('.' is ↯
              ↯not supported)`);
80         }
81         var v = a[k];
82         if (typeof v === 'object'){
83             checkI18nKeys(v);
84         }
85     }
86 }
87
88 /**
89  * Set a new context for the web component
90  * @param {} context | {name: 'unique', en: {}, es: {}, ...}
91  * @throws {} Error when context has no name, when it exists or when it does ↯
              ↯not have fallback
92  */
93 function setContext(context){
94     var name = context.name;

```

```

95     if (typeof name !== 'string'){
96         throw new Error(`Context needs a valid string 'name' property to be ↵
            ↵identified.`);
97     }
98     if (strict && fallback !== false && context[fallback] == null){
99         throw new Error(`Context needs at least the '${fallback}' fallback ↵
            ↵definitions.`);
100    }
101    var ctx = ctxs[name];
102    if (ctx){
103        throw new Error(`'${name}' is a previously loaded context. Choose ↵
            ↵another context name.`);
104    }
105
106    checkI18nKeys(context);
107
108    ctxs[name] = context;
109 }
110
111 /**
112  * Set the initialization configuration
113  * @param {} data | {lang: 'es', fallback: true, strict: false}
114  */
115 function setConfig(data){
116
117     var l = data.lang;
118     if (l && available.indexOf(l) !== -1){
119         lang = l;
120     }
121
122     var fall = data.fallback;
123     if (fall != null){
124         fallback = fall;
125     }
126
127     var str = data.strict;
128     if (str != null){
129         strict = str;
130     }
131 }
132
133 module.exports = {
134     _i,
135     setConfig,
136     setContext
137 };

```

Listing 4: remove button component i18n schema definition (remove-lang.js).

```

0  module.exports = {
1      name: 'remove',
2      /* {% if en %} */
3      en: {
4          message: {
5              '1': 'Do you like to remove the loaded file?',
6              'default': 'Do you like to remove the {{number}} loaded files?'
7          },
8          messagef: function(num){
9              var number,
10             s;
11             if (num > 1){
12                 number = ' ' + num;
13                 s = 's';
14             }else{
15                 number = '';
16                 s = '';
17             }
18             var msg = `Do you like to remove the${number} loaded file${s}?`;
19             return msg;
20         },
21         title: 'File Uploader',
22         remove: 'Remove',
23         cancel: 'Cancel',
24         tooltip: 'remove loaded files and clean field'
25     },
26     /* {% endif %} */
27     /* {% if es %} */
28     es: {
29         message: {
30             '1': '¿Desea eliminar el fichero cargado?',
31             'default': '¿Desea eliminar los {{number}} ficheros cargados?'
32         },
33         messagef: function(num){
34             var msg = '¿Desea eliminar ';
35             if (num === 1){
36                 msg += 'el fichero cargado?';
37             }else{
38                 msg += `los ${num} ficheros cargados`;
39             }
40             return msg;
41         },
42         title: 'Cargador de ficheros',
43         remove: 'Borrar',
44         cancel: 'Cancelar',
45         tooltip: 'borrar ficheros cargados y limpiar campo'
46     }
47     /* {% endif %} */
48 };

```

Listing 5: remove button component i18n schema definition (remove-lang.js).

```

0  const i18n_CONTEXT = 'remove';
1  // ...
2  fn: () => {
3      var num = fS.getFileNames().length;
4      toastr({
5          name: dataModel.name,
6          //messagef: _i(i18n_CONTEXT, 'message', num), // function
7          message: _i(i18n_CONTEXT, 'message', num, {number: num}), // dictionary
8          title: _i(i18n_CONTEXT, 'title'), // literal
9          type: 'warning',
10         menu: [
11             {
12                 text: _i(i18n_CONTEXT, 'remove'),
13                 fn: () => fS.clean()
14             }, {
15                 text: _i(i18n_CONTEXT, 'cancel')
16             }
17         ]
18     });
19 }
20 // ...
21 attr['data-original-title'] = _i(i18n_CONTEXT, 'tooltip');

```

Listing 6: i18n-components schema bootstrap with yasnippet (lang.snippet).

```

0  # -*- mode: snippet -*-
1  # name: lang
2  # key: lang
3  # contributor: Raúl Nozal <nozalr@baigene.com>
4  # expand-env: ((yas-indent-line 'fixed))
5  # --
6  module.exports = {
7      name: '${1:unique-name}',
8      /* {% if en %} */
9      en: {
10         $0
11     },
12     /* {% endif %} */
13     /* {% if es %} */
14     es: {
15
16     }
17     /* {% endif %} */
18 };

```

7.3.4. Helpers

Listing 7: nav-blocks extension (nav-blocks.el).

```
0  ;;; nav-blocks.el --- Navigation through blocks of source code and pairs, fast↵
   ↳ movements
1
2  ;; Copyright (C) 2015  Raúl Nozal
3
4  ;; Author: Raúl Nozal
5  ;; Maintainer: not-yet
6  ;; URL: not-yet
7  ;; Version: 0.1.0
8  ;; Keywords: nav-blocks, navigation, blocks, movement, move, jump, goto, ↵
   ↳ javascript, js
9  ;; Package-Requires: ((emacs "24.1") (dash "2.10") (s "1.9"))
10
11 ;; This program is free software; you can redistribute it and/or modify
12 ;; it under the terms of the GNU General Public License as published by
13 ;; the Free Software Foundation, either version 3 of the License, or
14 ;; (at your option) any later version.
15
16 ;; This program is distributed in the hope that it will be useful,
17 ;; but WITHOUT ANY WARRANTY; without even the implied warranty of
18 ;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
19 ;; GNU General Public License for more details.
20
21 ;; You should have received a copy of the GNU General Public License
22 ;; along with this program. If not, see <http://www.gnu.org/licenses/>.
23
24 ;;; Example of configuration
25
26 ;;;; Nozal
27 ;; (load-file-sure-append "nav-blocks.elc" "config/modes/js")
28 ;; (require 'nav-blocks)
29
30 ;;;; alternative
31 ;; (add-hook 'js2-mode-hook
32 ;;          (lambda ()
33 ;;            (flycheck-mode t)))
34
35 ;;;;; movement
36 ;; (define-key js2-mode-map (kbd "M-f") 'nav-block-goto-pair)
37 ;; (define-key json-mode-map (kbd "M-f") 'nav-block-goto-pair)
38
39 ;; (define-key js2-mode-map (kbd "M-p") 'nav-block-goto-backward)
40 ;; (define-key js2-mode-map (kbd "M-n") 'nav-block-goto-forward)
41
```



```

42 ;;; Code
43
44 (require 'dash)
45 (require 's)
46
47
48 (defgroup nav-block nil
49   "Navigation through blocks (JavaScript-focused)."
50   :group 'nav-block)
51
52 (defvar nav-block--re-keywords-js "\\(function\\|if\\|else\\|for\\|do\\|while\\|
53   ↪\\|switch\\|case\\|default\\|try\\|catch\\|finally\\)")
54
55
56 ;;; custom vars
57
58 (defcustom nav-block-re-keywords nav-block--re-keywords-js
59   "Keywords to match when using `nav--block-find-' or its wrappers
60   `nav-block-goto-backward' and `nav-block-goto-forward'. By default JavaScript ↵
61   ↪ 'keywords'."
62   :type '(choice (const :tag "JavaScript" nav-block--re-keywords-js)
63     (const :tag "Implement more..." nav-block--re-keywords-js))
64   :package-version '(nav-blocks . "0.1.0"))
65
66
67
68
69 ;;; interactive
70
71 (defun nav-block-goto-backward (point)
72   "Dependent on `font-lock-keyword-face' to match the keywords.
73   Search backward from the current cursor point."
74   (interactive
75     (list (point))))
76   (nav--block-find- point 'backward))
77
78
79 (defun nav-block-goto-forward (point)
80   "Dependent on `font-lock-keyword-face' to match the keywords.
81   Search forward from the current cursor point."
82   (interactive
83     (list (point))))
84   (nav--block-find- point 'forward))
85
86
87 (defun nav-block-goto-pair (point &optional affinity)
88   "nav-block function to jump between pair blocks
89   like [], {} or (). Useful for languages like Emacs-Lisp, JavaScript,
90   C/C++, Java, etc."

```

```

91
92 Note: even if your code is not a valid JavaScript it will work because
93 it searches like in 'rings' from the nested one to the outer one.
94 Be careful with comments that content odd open/close pairs in between
95 your 'ring' of search.
96
97 Under `interactive' mode the default length to find a open/close pair
98 from the cursor point is 5 chars. In case of conflict (same length),
99 prevaleces the left one (searching forward).
100 (interactive
101   (list (point) 5)) ;; 5 chars before/after to find a { or }
102
103 (let* ((chars-open '(?\\{ ?\\[ ?\\(\\))
104        (chars-close '(?\\} ?\\] ?\\)\\))
105        (save-pt point)
106        (init-pt point)
107        (down-dir t)
108        (from-char (let* ((min-p (- save-pt (if affinity affinity 5)))
109                          (max-p (+ save-pt (if affinity affinity 5)))
110                          (all-pairs (-concat chars-open chars-close))
111                          (f-bw (+ save-pt 1))
112                          (f-fw (progn
113                                (setq max-p (1+ max-p)) ;; search-forward needs one more
114                                (1- save-pt)))
115                          (bw-pt-pre (save-excursion
116                                      (cadr (--min-by (let* ((off (car it))
117                                                            (offo (car other))
118                                                            (voff (if off off 16))
119                                                            (voffo (if offo offo 16)))
120                                                                (> voff voffo))
121                                      (-map (lambda(char) (progn (goto-char f-bw)
122                                                                    (let* ((ps (search-backward (string
123                                                                    ↪ char) min-p t))
124                                                                    (ps-offset (if ps
125                                                                    ↪ (- init-pt ps)
126                                                                    ↪ nil)))
127                                                                    (list ps-offset ps char))))
128                                      all-pairs))))))
129                          (bw-pt bw-pt-pre)
130                          (fw-pt-pre (save-excursion
131                                      (cadr (--min-by (let* ((off (car it))
132                                                            (offo (car other))
133                                                            (voff (if off off 16))
134                                                            (voffo (if offo offo 16)))
135                                                                (> voff voffo))
136                                      (-map (lambda(char) (progn (goto-char f-fw)
137                                                                    (let* ((ps (search-forward (string
138                                                                    ↪ char) max-p t))
139                                                                    (ps-offset (if ps
139                                                                    ↪ (- ps init-pt)
140                                                                    ↪ nil)))
139

```

```

140                                     (list ps-offset ps char)))
141                                     all-pairs))))))
142      (fw-pt (and fw-pt-pre
143                (1- fw-pt-pre)))
144      (char-pt (cond ((and bw-pt fw-pt) (if (< (abs (- init-pt bw-pt))
145        ↳ (abs (- fw-pt init-pt))) bw-pt fw-pt)) ;; on conflict, ↗
146        ↳go forward
147        (bw-pt bw-pt)
148        (fw-pt fw-pt)
149        (t nil)))
150      (char (if char-pt
151                (progn
152                  (setq init-pt char-pt)
153                  (char-after char-pt))
154                nil)))
155      (if from-char
156        (let* ((msg nil)
157              (to-char (or (let* ((i (-elem-index from-char chars-open))
158                                (ch (when i
159                                      (nth i chars-close))))
160                            (if ch
161                              (progn
162                                (setq down-dir t)
163                                ch)
164                              nil))
165              (let* ((i (-elem-index from-char chars-close))
166                    (ch (when i
167                          (nth i chars-open))))
168                (if ch
169                  (progn
170                    (setq down-dir nil)
171                    ch)
172                  nil))
173              (setq msg (concat "Only supported '" (s-join "' ' (-map '↗
174        ↳string '(40 41))) "'."))))
175        (not-found t)
176        (pt-max (if down-dir
177                  (point-max)
178                  (point-min)))
179        (open-num 1)
180        (place (unless msg
181                  (goto-char save-pt) ;; in case it reaches the top/bottom of ↗
182                  ↳buffer
183                  (save-excursion
184                    (goto-char init-pt)
185                    (while not-found
186                      (if down-dir
187                        (forward-char)
188                        (backward-char))

```

```

187         (let ((pt-n (char-after (point))))
188             (cond ((eq pt-n to-char) (setq open-num (1- open-num)))
189                   ((eq pt-n from-char) (setq open-num (1+ open-num))))
190             (cond ((eq open-num 0) (setq not-found nil))
191                   ((eq pt-n pt-max) (setq not-found nil))))))
192         (point))))))
193 ;; it is not possible to have { as first char (only json)
194 (cond (msg (error msg))
195       ((eq open-num 0) (goto-char place))))
196 (goto-char save-pt)))
197
198
199
200 ;;; private
201
202 (defun nav--block-find- (point &optional direction)
203   "Jump to the next (or previous) special keyword block. Fast navigation ↗
204   ↳through
205   the source code. JavaScript centered, but works for C/C++, C#, Java or even ↗
206   ↳Python.
207   Dependent on `font-lock-keyword-face' to match the keywords (`js2-mode')."
208   (let* ((save-pt point)
209          (not-found t)
210          (fn-search (if (eq direction 'forward)
211                        #'search-forward-regexp
212                        #'search-backward-regexp))
213          (fn-mv (if (eq direction 'forward)
214                    #'forward-char
215                    #'backward-char))
216          (re-keywords nav-block-re-keywords)
217          (pt-found (save-excursion
218                     (let ((place nil))
219                       (while not-found
220                         (let* ((pt-pre (funcall fn-search re-keywords nil t))
221                              (pt-offset (when pt-pre
222                                           (if (eq direction 'forward)
223                                               (- (length (match-string 1)))
224                                               0)))
223                             (pt-beg (and pt-pre
224                                           (+ pt-pre pt-offset)))
225                             (next (if (eq save-pt pt-beg)
226                                       t
227                                       nil))
228                             (pt-font (and pt-beg
229                                           (get-text-property pt-beg 'font-lock-face))))
229                         (unless next
230                           (if pt-pre
231                             (when (and pt-font
232                                           (eq pt-font 'font-lock-keyword-face))
233                               (progn

```

```

236             (setq not-found nil)
237             (setq place pt-beg)))
238         (progn
239             (setq not-found nil)
240             (setq place nil))))))
241         place))))
242     (if pt-found
243       (goto-char pt-found)
244       (goto-char save-pt))))
245
246
247 (provide 'nav-blocks)

```

Listing 8: stylus-mode modification (stylus.el).

```

0  ;; Copyright (C) 2015  Raúl Nozal
1
2  ;; Author: Raúl Nozal
3  ;; (not a package
4
5  ;; This program is free software; you can redistribute it and/or modify
6  ;; it under the terms of the GNU General Public License as published by
7  ;; the Free Software Foundation, either version 3 of the License, or
8  ;; (at your option) any later version.
9
10 ;; This program is distributed in the hope that it will be useful,
11 ;; but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13 ;; GNU General Public License for more details.
14
15 ;; You should have received a copy of the GNU General Public License
16 ;; along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 ;;; Code
19
20 (require 'stylus-mode)
21
22 ;; redefinition of stylus-mode' `font-lock-keywords' to support colors
23 ;; adapted to understand #XYZ and #XXYYZZ hexa units
24 (setq stylus-font-lock-keywords
25   `(
26     (,"^[ {2,}]+[a-z0-9_:\-]+[ ]" 0 font-lock-variable-name-face)
27     (,"\\(:?:?\\(root\\|nth-child\\|nth-last-child\\|nth-of-type\\|nth-last-of- ↵
        ↵type\\|first-child\\|last-child\\|first-of-type\\|last-of-type\\|only ↵
        ↵-child\\|only-of-type\\|empty\\|link\\|visited\\|active\\|hover\\| ↵
        ↵focus\\|target\\|lang\\|enabled\\|disabled\\|checked\\|not\\|\\)\\)*" . ↵
        ↵font-lock-type-face) ;; pseudoSelectors
28     (,(concat "[^_]?\\(<\\(" stylus-colours "\\)\\>[^_]?")
29       0 font-lock-constant-face)

```

```

30      (,(concat "[^_]?\\<\\(" stylus-keywords "\\)\\>[^_]?" )
31      0 font-lock-keyword-face)
32      ("\\(#[abcdef[:digit:]]\\{6\\}\\)" .
33      (0 (put-text-property
34      (match-beginning 0)
35      (match-end 0)
36      'face (list :background (match-string-no-properties 0)))))
37      ("\\(#[abcdef[:digit:]]\\{3\\}\\)[^abcdef[:digit:]]" .
38      (0 (put-text-property
39      (match-beginning 1) ;;1
40      (match-end 1) ;;1
41      'face (list :background (match-string-no-properties 1))));;1
42
43      (,"\\([.0-9]+?\\(em\\|ex\\|px\\|mm\\|cm\\|in\\|pt\\|pc\\|deg\\|rad\\|grad\\|
44      ↵ms\\|s\\|Hz\\|kHz\\|rem\\|%\\)\\b\\)" 0 font-lock-constant-face)
45      (,"\\b[0-9]+\\b" 0 font-lock-constant-face)
46      (,"\\.\\.w[a-zA-Z0-9\\-]+\" 0 font-lock-type-face) ; class names
47      (,"$\\w+\" 0 font-lock-variable-name-face)
48      (,"@\\w[a-zA-Z0-9\\-]+\" 0 font-lock-preprocessor-face) ; directives and ↵
49      ↵backreferences
50    ))

```

7.3.5. Phenotype Indicator

Listing 9: template-phenotype Interpreter (interpreter.js).

```

0
1  module.exports = function(config){
2    var i18n = config.i18n,
3    _i = i18n._i,
4    i18n_CONTEXT = 'interpreter';
5
6    i18n.setContext(require('./interpreter-lang'));
7
8    var PhenotypeData = function(values, options){
9      this.population = (typeof values.population === 'number') ? ↵
10     ↵values.population : 0;
11      this.min = (typeof values.min === 'number') ? values.min : 0;
12      this.max = (typeof values.max === 'number') ? values.max : 0;
13      this.phenotype = (typeof values.phenotype === 'string') ? ↵
14     ↵values.phenotype : '';
15      this.options = options || {};
16
17      var langCache = {
18        inferior: _i(i18n_CONTEXT, 'inferior'),
19        equal: _i(i18n_CONTEXT, 'equal'),
20        superior: _i(i18n_CONTEXT, 'superior'),
21        less: _i(i18n_CONTEXT, 'less'),

```

```

20         "equal to": _i(i18n_CONTEXT, "equal to"),
21         more: _i(i18n_CONTEXT, 'more'),
22     };
23     if (!this.options.diff){
24         this.options.diff = function( diff ){
25             return diff.toFixed(2);
26         };
27     }
28     if (!this.options['diff-%']){
29         this.options['diff-%'] = function( diff ){
30             return diff.toFixed(2) + '%';
31         };
32     }
33     if (!this.options.comp){
34         this.options.comp = function( diff ){
35             var comp = langCache.inferior;
36             if (diff === 0){
37                 comp = langCache.equal;
38             }else if (diff > 0){
39                 comp = langCache.superior;
40             }
41             return comp;
42         };
43     };
44     if (!this.options['comp-alt']){
45         this.options['comp-alt'] = function( diff ){ // we know it uses ↵
46             ↵"favorable"
47             var comp = langCache.less;
48             if (diff === 0){
49                 comp = langCache['equal to'];
50             }else if (diff > 0){
51                 comp = langCache.more;
52             }
53             return comp;
54         };
55     };
56
57     var UserData = function(phenotypeData, userValue){
58         this.phenotypeData = phenotypeData;
59         this.user = (typeof userValue === 'number') ? userValue : 0;
60     };
61     UserData.prototype.setUserValue = function(userValue){
62         this.user = (typeof userValue === 'number') ? userValue : this.user;
63     };
64     UserData.prototype.precalc = function(){
65         var diff = (this.user - this.phenotypeData.population);
66         this.diff = (typeof diff === 'number') ? diff : null;
67     };
68
69

```

```

70     var templatePhenotypeCommands = {
71     var: {
72         PHENOTYPE: function(data){
73             var phenotype;
74             if (typeof data.phenotypeData.phenotype === 'string'){
75                 phenotype = data.phenotypeData.phenotype;
76             }else{
77                 phenotype = '';
78             }
79             return phenotype;
80         },
81         DIFF: function(data){
82             var diff;
83             if (typeof data.diff === 'number'){
84                 diff = data.diff;
85             }else{
86                 diff = data.user - data.population;
87             }
88             diff = Math.abs(diff) || 0;
89             return data.phenotypeData.options.diff( diff );
90         },
91         'DIFF-%': function(data){
92             var diff;
93             if (typeof data.diff === 'number'){
94                 diff = data.diff;
95             }else{
96                 diff = data.user - data.population;
97             }
98             diff = Math.abs(diff) || 0;
99             return data.phenotypeData.options['diff-%']( diff );
100         },
101         COMP: function(data){
102             var diff;
103             if (typeof data.diff === 'number'){
104                 diff = data.diff;
105             }else{
106                 diff = data.user - data.phenotypeData.population;
107                 diff = ( typeof diff === 'number' ) ? diff : 0;
108             }
109             return data.phenotypeData.options.comp( diff );
110         },
111         'COMP-ALT': function(data){
112             var diff;
113             if (typeof data.diff === 'number'){
114                 diff = data.diff;
115             }else{
116                 diff = data.user - data.phenotypeData.population;
117                 diff = ( typeof diff === 'number' ) ? diff : 0;
118             }
119             return data.phenotypeData.options['comp-alt']( diff );
120         },

```



```

121     VALUE: function(data){
122     return data.user || 0;
123     }
124 },
125     fn: {
126     'IF-DIFF': function(data){
127         return data.diff !== 0;
128     },
129     'IF-EQUAL': function(data){
130         return data.diff === 0;
131     }
132     }
133 };
134
135
136 /*
137  * Type: behavioral
138  * Pattern: interpreter
139  * Use cases:
140  * - loose coupling a bit between objects/classes (each a different ↵
141     ↵expression)
142  * - simple grammar
143  * - interprets expressions based on a context
144  * - not efficient but easily modifiable/extensible
145  */
146 var templatePhenotype = {};
147 (function(templatePhenotype){
148
149     var utils = {
150         trimSpaces: function( string ){
151             return (typeof string === 'string') ? ↵
152                 ↵string.replace(/^\s*/, '').replace(/\s*$/, '') : null;
153             // support spaces before and after the command
154         }
155     };
156
157     function InterpreterContext (input){
158         this.err = null;
159
160         this.complete = {
161             input: input,
162             parts: []
163         };
164         this.partial = {
165             input: input,
166             parts: this.complete.parts
167         };
168     }
169
170     function CommandExpression( cmds ){

```

```

170
171     if (!cmds){
172         cmds = {};
173         for (var type in templatePhenotypeCommands){
174             for (var name in templatePhenotypeCommands[type]){
175                 cmds[name] = type;
176             }
177         }
178     }
179     this.cmds = cmds;
180 }
181 CommandExpression.prototype.interpret = function( ctx ){
182     var matched = false;
183     if (ctx && typeof ctx.partial.input === 'string'){ //null or undefined
184
185         var s = ctx.partial.input;
186         var sLen = s.length;
187
188         var open = 1, // 0 is '{'
189             close = -1,
190             balance = 0;
191
192         var ctx_partials = [];
193
194         var guessed = 'v';
195
196         // maybe function
197         var parOpen = s.indexOf('(');
198         var parClose = s.lastIndexOf(')');
199         var cmdname;
200
201         var part = { type: 'cmd' };
202         var isSubtype = 'fn';
203
204         if (parOpen && parOpen !== -1 && parClose !== -1 && parOpen < parClose){
205             cmdname = utils.trimSpaces( s.substring(1, parOpen) );
206             if (this.cmds[cmdname] === isSubtype){
207
208                 var innerStr = s.substring(parOpen + 1, parClose);
209                 var innerExp = new DecomposerExpression();
210                 // process
211                 part.subtype = isSubtype;
212                 part.value = cmdname;
213                 part.parts = [];
214                 part.lengths = [ s.substring(0, parOpen + 1).length,
215                               s.substring(parClose, sLen).length ];
216
217                 var currentParts = ctx.partial.parts;
218
219                 ctx.partial.parts.push( part );
220

```

```

221         ctx.partial.parts = part.parts;
222
223         ctx.partial.input = innerStr;
224         innerExp.interpret( ctx );
225
226         ctx.partial.parts = currentParts;
227
228         matched = true;
229     }
230 }else{
231     isSubtype = 'var';
232     cmdname = utils.trimSpaces( s.substring(1, sLen - 1) );
233     if (this.cmds[cmdname] === isSubtype){
234         // process
235         part.subtype = isSubtype;
236         part.value = cmdname;
237         part.lengths = [ s.substring(0, sLen).length ];
238
239         ctx.partial.parts.push( part );
240         matched = true;
241     }
242 }
243
244 }
245 return matched;
246 };
247
248 function PlaintextExpression( ){
249 }
250 PlaintextExpression.prototype.interpret = function( ctx ){
251     var matched = false;
252     if (ctx && typeof ctx.partial.input === 'string' && ctx.partial.input ↵
253         ↵ !== ''){ //null or undefined
254         var part = { type: 'plain',
255             value: ctx.partial.input,
256             lengths: [ ctx.partial.input.length ] };
257
258         ctx.partial.parts.push( part );
259         matched = true;
260     }
261     return matched;
262 };
263
264 function DecomposerExpression( ){
265     this.exps = [ // order matters, first match "exists"
266         new CommandExpression(),
267         new PlaintextExpression()
268     ];
269 }
270 DecomposerExpression.prototype.interpret = function( ctx ){
271     var matched = false;

```

```

271     if (ctx && typeof ctx.partial.input === 'string' && ctx.partial.input ↵
        ↵ !== ''){
272     var s = ctx.partial.input;
273     var sLen = s.length;
274
275     var open = 0,
276         close = -1,
277         balance = 0;
278
279     var ctx_partials = [];
280
281     for (var i=0; i<sLen; i++){
282         var char = s[i],
283             from, to;
284         switch (char){
285             case '{':
286                 if (balance === 0){
287                     open = i;
288                     from = close + 1;
289                     if (from < open){
290                         ctx_partials.push( s.substring(from, open) );
291                     }
292                 }
293                 balance++;
294                 break;
295             case '}':
296                 balance--;
297                 if (balance === 0){
298                     close = i;
299                     to = close + 1;
300                     ctx_partials.push( s.substring(open, to) );
301                 }
302                 break;
303             }
304         }
305         from = close + 1;
306         if (sLen > from){
307             ctx_partials.push( s.substring(from) );
308         }
309         if (balance !== 0){
310             ctx.err = "Unbalanced command separators (open - closed): '{' - ↵
        ↵ '}' : " + balance;
311         }else{
312             var l = ctx_partials.length;
313             for (var i = 0; i < l; i++){
314                 var part = ctx_partials[i];
315                 if (part){ // && part.length){
316                     var done = false;
317
318                     var exps = this.exps,
319                         nExps = exps.length;

```

```

320
321         ctx.partial.input = part;
322
323         for (var j=0; j<nExps; j++){
324             done = done || this.exps[j].interpret( ctx );
325         }
326
327     }
328 }
329     matched = true; // is a string not empty (at minimum divided in ↵
        ↵one part)
330 }
331 }
332     return matched;
333 };
334
335
336 function buildExpression( object, data ){
337
338     function spacePunctuation( string ){
339         if (typeof string === 'string'){
340             string = string.replace(/([^\d][,\.])([^\d])/g, '$1 $2'); // not ↵
        ↵in a number '2.3', '13,22'
341         }
342         return string;
343     }
344
345     var o = "";
346     if (object && object.parts){
347         var len = object.parts.length;
348         for (var i=0; i<len; i++){
349
350             var part = object.parts[i];
351             var spacer;
352             switch (part.type){
353                 case 'plain':
354                     o += spacePunctuation( part.value ) + ' ';
355                     break;
356                 case 'cmd':
357                     var response = ↵
        ↵templatePhenotypeCommands[part.subtype][part.value](data);
358                     if (typeof response === 'boolean'){
359                         if (response){
360                             var currentParts = object.parts;
361                             object.parts = part.parts;
362                             o += spacePunctuation( buildExpression( object, data ) ) + ' ';
363                             object.parts = currentParts;
364                         }
365                     }else{
366                         o += spacePunctuation( response ) + ' ';
367                     }

```

```

368         }
369     }
370 }
371
372 }
373     return o;
374 }
375
376 templatePhenotype.exp = {
377     build: function( object, data ){
378         var built = buildExpression( object, data );
379         var matchSpaces = built.match(/\s{2,}/g),
380             nSpaces = (matchSpaces) ? matchSpaces.length : 0;
381         for (var i=0; i<nSpaces; i++){
382             if (matchSpaces[i].indexOf('\n') !== -1){ // we remove \t
383                 built = built.replace(matchSpaces[i], '\n');
384             }else if (matchSpaces[i].indexOf('\t') !== -1){
385                 built = built.replace(matchSpaces[i], '\t');
386             }else{ // just spaces
387                 built = built.replace(matchSpaces[i], ' ');
388             }
389         }
390         // Spanish grammar: ' , ' -> ',,'; ' .' -> ' .'
391         built = built.replace(/ +,/g, ','); // not \s
392         built = built.replace(/ +\./g, '.'); // not \s (\t, \n)
393         return built.trim(); // first and last char.
394     },
395     parse: function( string ){
396         var ctx = new InterpreterContext(string);
397         var exp = new DecomposerExpression();
398         exp.interpret(ctx);
399         if (ctx.err){
400             throw new Error(ctx.err);
401         }
402         return ctx.complete;
403     }
404 };
405
406 })(templatePhenotype);
407
408
409 module.exports = exports = {
410     PhenotypeData: PhenotypeData,
411     UserData: UserData,
412     templatePhenotypeCommands: templatePhenotypeCommands,
413     templatePhenotype: templatePhenotype
414 };
415
416 };

```

Listing 10: template-phenotype hinting extension (template-phenotype-hint .js).

```

0  var CodeMirror = require('codemirror');
1  var interpreter = require('./interpreter');
2
3  var RANGE = 20;
4
5  var availableCommands = [];
6
7  for (var type in interpreter.templatePhenotypeCommands){
8      for (var name in interpreter.templatePhenotypeCommands[type]){
9          availableCommands.push( name + ((type === 'fn') ? '()' : ' ' ));
10         }
11     }
12
13     CodeMirror.registerHelper("hint", "phenotype-template", function(editor, ↵
        options) {
14         var cur = editor.getCursor(), curLine = editor.getLine(cur.line);
15
16         var line = cur.line;
17
18         var start = cur.ch,
19             end = cur.ch;
20
21         var matchKeyword = curLine.substring(0, end).match(/.*{\s*([^\s]*)$/);
22         var ret;
23         if (matchKeyword){
24             var mK = matchKeyword[1].replace('(', '\\(').replace(')', '\\)');
25
26             var re = new RegExp("\\s*" + mK + "(.*)");
27
28             var map = availableCommands.map(function(keyword){
29                 var m = keyword.match(re);
30                 return m ? m[1] : undefined;
31             }).filter(function(el){
32                 return el != null; // null or undefined
33                 // https://dorey.github.io/JavaScript-Equality-Table/
34             }).slice(0, RANGE);
35
36             var posStart = CodeMirror.Pos(line, start);
37             var posEnd = CodeMirror.Pos(line, end);
38
39             ret = {list: map, from: posStart, to: posEnd};
40         }
41         return ret;
42     });
43
44     module.exports = exports = CodeMirror;

```

Listing 11: template-phenotype mode adapter for *CodeMirror* (template-phenotype-mode.js).

```

0 var CodeMirror = require('codemirror');
1 var interpreter = require('./interpreter');
2
3 CodeMirror.defineMode("phenotype-template", function(config) {
4
5     var flushedPosition = 0;
6     var flushed = true;
7
8     // pos means number of chars up to the char to be analyzed
9     function getStyleOfPartAtPos(parts, pos){
10     var left = pos;
11     for (var pi in parts){
12         var part = parts[pi];
13
14         var fnSearchable = true;
15         for (var si in part.lengths){
16             var len = part.lengths[si]; // length
17
18             left -= len;
19             if (left <= 0){ // is inside this
20                 if (part.subtype === 'fn'){
21                     return 'fn';
22                 }else if(part.subtype === 'var'){
23                     return 'var';
24                 }else{
25                     return 'plain';
26                 }
27             }else if (part.subtype === 'fn' && fnSearchable){ // can be nested
28                 var inner = getStyleOfPartAtPos( part.parts, left ); // recursive
29                 switch (typeof inner){
30                     case 'string':
31                         return inner;
32                         break;
33                     case 'number':
34                         left = inner;
35                         break;
36                 }
37                 fnSearchable = false;
38             }
39         }
40     }
41     return left;
42 }
43
44 var stored = {
45     string: null,
46     obj: null,
47     firstChar: null
48 };

```



```

49
50     var instance = {
51     editor: null,
52     logger: config.logger
53     };
54     instance.fetch = function(){
55     if (!instance.editor || !instance.editor.getValue){
56         var editor = (config && config.getInstance) ? config.getInstance() : ↵
            ↵null;
57         if (editor){
58             instance.editor = editor.self;
59         }
60     }
61     };
62
63     return {
64     token: function(stream, state) {
65         var reparse = false,
66             currentContent;
67         if (instance.editor && instance.editor.getValue){
68             currentContent = instance.editor.getValue();
69         if (currentContent !== stored.string){
70             reparse = true;
71         }
72         }else{
73             instance.fetch();
74         }
75
76         if (!instance.logger){
77             instance.logger = config.logger;
78         }
79
80         var logger = instance.logger;
81
82         try{
83         if (reparse){ // cached
84             stored.obj = interpreter.templatePhenotype.exp.parse( ↵
                ↵currentContent );
85             stored.string = currentContent;
86         }
87
88         logger && logger.flush();
89
90         }catch(e){
91             logger && logger.write( e.message );
92             stored.obj = null;
93         }
94
95         var token_name,
96             char = stream.peek();
97

```

```

98     if (stored.obj && char){
99     var OFFSET_TO_CHAR = 1;
100    flushedPosition = stream.pos + OFFSET_TO_CHAR; // 1 -> 0; 2 -> 1
101
102    var streamOffset = currentContent.indexOf(stream.string);
103
104    var braces = {'{':true, '}':true};
105    var parens = {'(':true, ')':true};
106    if (char in braces){
107        braces = true;
108    }else{
109        braces = false;
110    }
111    if (char in parens){
112        parens = true;
113    }else{
114        parens = false;
115    }
116
117    switch( getStyleOfPartAtPos( stored.obj.parts, flushedPosition + ↵
        ↵streamOffset ) ){
118    case 'fn':
119        token_name = (braces || parens) ? 'fn-b' : 'fn';
120        break;
121    case 'var':
122        token_name = (braces || parens) ? 'var-b' : 'var';
123        break;
124    case 'plain':
125        token_name = (braces) ? 'cmd' : undefined;
126        break;
127    default:
128        token_name = undefined;
129
130    }
131
132    stream.next();
133    return token_name;
134    }
135
136    token_name = stream.skipToEnd();
137    return token_name;
138    }
139    };
140 });
141
142 module.exports = exports = CodeMirror;

```

7.3.6. Optimizations and Foreign Function Interface

Listing 12: Rust “where is” program to locate dependencies (demonstration purposes `whereis.rs`).

```

0 // @author Raúl Nozal
1 // @license GNU GPLv3
2
3 // The idea it is to implement this behavior, but being BROWSER and ↵
4   ↳BROWSER_RUNNER
5 // dynamically obtained.
6
7 // #!/usr/bin/env bash
8 // [[ -z "$WEBBROWSER" ]] && WEBBROWSER=conkeror
9 // [[ -z "$WEBBROWSER_RUNNER" ]] && WEBBROWSER_RUNNER=xulrunner
10
11 // for cmd in zenity cgrep wmcctl $WEBBROWSER $WEBBROWSER_RUNNER;
12 // do
13 //     which $cmd &> /dev/null
14 //     [[ $? -ne 0 ]] && echo "Needs: $cmd" && exit 1
15 // done
16
17 #![feature(process)]
18 #![feature(env)]
19 #![allow(unused_variables)]
20 #![feature(old_io,io)]
21 #![feature(core, collections)]
22
23 use std::process::Command;
24
25 const GUESSER: &'static str = "which";
26
27 pub extern fn check_dependency(st: &str) -> bool {
28     let status = Command::new(GUESSER).
29         arg(st).
30         stdout(std::process::Stdio::null()).
31         stderr(std::process::Stdio::null()).
32         status().
33         unwrap_or_else(|e| {
34             panic!("Error executing `{}`: {}", GUESSER, e);
35         });
36
37     status.success()
38 }
39
40
41 use std::old_io;
42 use std::env;
43
44
45 // http://doc.rust-lang.org/1.0.0-beta/
46 // https://doc.rust-lang.org/book/
47 // https://doc.rust-lang.org/reference.html

```

```

48 #[allow(deprecated)]
49 pub extern fn main() {
50
51     let env_keys = ["WEBBROWSER", "WEBBROWSER_RUNNER"];
52     let default_values = ["conkeror", "xulrunner"];
53
54     let mut programs_user: Vec<String> = Vec::new();
55     // the order is important between this line and the next one
56     let mut programs: Vec<&str> = Vec::new();
57
58
59
60     for i in 0..2 {
61
62         let var = env::var( env_keys[i] );
63         match var {
64             Ok(val) => {
65                 programs_user.push( val )
66                 // needed to create this vec of String
67             },
68             Err(e) =>    programs.push( default_values[i])
69         }
70     }
71
72     for progr in programs_user.iter() {
73         programs.push(progr);
74     }
75
76     programs.push_all( &["zenity", "cgrep", "wmctrl"] );
77
78     let mut heading_shown = false;
79     let mut serr = old_io::stderr();
80
81     for program in programs {
82
83         if ! check_dependency(&program) {
84
85             if ! heading_shown {
86                 heading_shown = true;
87
88                 let bytes_w = serr.write( String::from_str("Programs needed: ↵
89                                     ↵").into_bytes().as_slice() );
89
90                 bytes_w.ok().expect("failed to write to STDERR (0).");
91
92             }
93             let mut prog = String::from_str(program);
94             prog.push_str(" ");
95             serr.write_all( prog.into_bytes().as_slice() ).
96                 ok().expect("failed to write to STDERR (1)");
97

```

```

98     }
99   }
100
101   if heading_shown {
102     serr.write_all( String::from_str("\n").into_bytes().as_slice() ).
103     ok().expect("failed to write to STDERR (2)");
104   }
105
106   drop(serr);
107
108   std::env::set_exit_status( if heading_shown { 1i32 } else { 0 } )
109 }

```

Listing 13: JavaScript Jeff Greenberg's Duff's Device (`duffs-device.js`).

```

0  /*
1   * Credits to Jeff Greenberg, Tom Duff,
2   * Functions exposed independently,
3   * just to be shown. Benchmarks and tests apart.
4   * arr as parameter to be in one func, but not
5   * a real case.
6   */
7  function duffsDevice(arr){
8     var sum = 0;
9     var a = arr;
10    var l = a.length;
11    var i = l % 8;
12    while(i){
13      sum+=a[--i].index;
14    }
15    i = Math.floor(l / 8);
16    while(i--){
17      sum+=a[--l].index;
18      sum+=a[--l].index;
19      sum+=a[--l].index;
20      sum+=a[--l].index;
21      sum+=a[--l].index;
22      sum+=a[--l].index;
23      sum+=a[--l].index;
24      sum+=a[--l].index;
25    }
26    return sum;
27  }
28
29  /*
30   * Adaptation to its functional form
31   * @author Jeff Greenberg (imperative)
32   */
33  function duffsDeviceFn(arr, fn){

```

```

34     var sum = 0;
35     var a = arr;
36     var l = a.length;
37     var i = l % 8;
38     while(i){
39         fn(a[--i]);
40     }
41     i = Math.floor(l / 8);
42     while(i--){
43         fn(a[--l]);
44         fn(a[--l]);
45         fn(a[--l]);
46         fn(a[--l]);
47         fn(a[--l]);
48         fn(a[--l]);
49         fn(a[--l]);
50         fn(a[--l]);
51     }
52     return sum;
53 }

```

Listing 14: *JavaScript Nozal's Loop* implementation (nozals-device.js).

```

0  /*
1   * @author Raúl Nozal
2   * Functions exposed independently,
3   * just to be shown. Benchmarks and tests apart.
4   * Imperative, to be used inside the code
5   * used with /yasnipet/ as a scheme
6   * arr as parameter to be in one func, but
7   * not a real case (imperative)
8   * Speedup (S) of 1.19
9   */
10 function nozalsLoop(arr){
11     var sum = 0;
12     var a = arr;
13     var l = a.length;
14     var lmod = l >> 2;
15     var li = lmod << 2;
16     var i, k;
17     for (i=0, k=0; i<lmod; i++) {
18         var a0 = a[k++],
19             a1 = a[k++],
20             a2 = a[k++],
21             a3 = a[k++];
22         sum += a0.index + a1.index + a2.index + a3.index;
23     }
24     var rest = l - li ;
25     switch(rest){

```

```
26     case 3: sum += a[li++].index;
27     case 2: sum += a[li++].index;
28     case 1: sum += a[li++].index;
29     }
30     return sum;
31 }
32
33 /*
34  * Functional form. Speedup (S) of 1.95
35  * @author Raúl Nozal
36  */
37 function nozalsLoopFn(arr, fn){
38     var a = arr;
39     var l = a.length;
40     var lmod = l >> 2;
41     var li = lmod << 2;
42     var i, k;
43     for (i=0, k=0; i<lmod; i++) {
44         var a0 = a[k++],
45             a1 = a[k++],
46             a2 = a[k++],
47             a3 = a[k++];
48         fn(a0); fn(a1); fn(a2); fn(a3);
49     }
50     var rest = l - li;
51     switch(rest){
52     case 3: fn(a[li++]);
53     case 2: fn(a[li++]);
54     case 1: fn(a[li++]);
55     }
56 }
```
