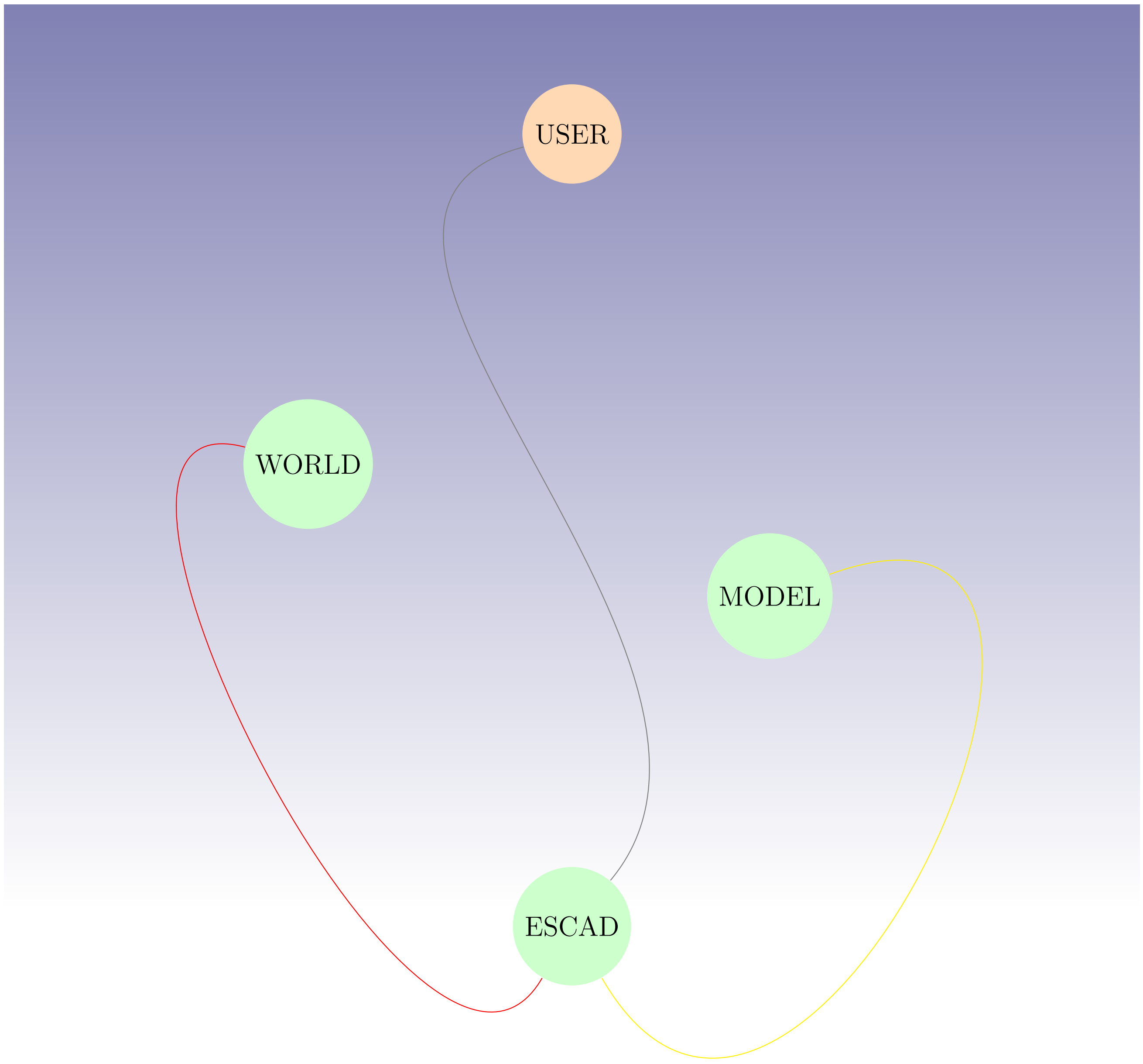
ESCAD MANUAL - 17th October 2020

by: Markus Kollmar



Alles hängt mit allem zusammen. Wir haben nur nicht das ganze Bild...

Contents

1 Motivation

History tells us about separation. Separation between different professions was and is common. People have different interests and different knowledge. So this seems natural. Is this a problem? No. And yes. Nowadays science goes into a direction where interdisciplinarity seems more useful. The bounds of our disciplines are drawn by human but it not need to reflect the reality in nature. Real problems are mostly system-problems. Systems are combined of different disciplines. But do the (software) tools support this fact? Some may but many do not. This seems not a big problem in many cases. But when it comes to documentation or knowledge transfer tasks, this can be a big problem. The tools often do not interact with each other, and if so they often feel not integrated well. This problem increases when different manufacturers have tools which interact not or very bad.

Escad strictly wants to be open source, use open interfaces and provide a wide variety of different domain tools. And if there is (yet) not the thing you need, you can customize Escad with common-lisp code.

1.1 Philosophy

Living in todays world is getting more and more complex. There are laws made by human which you have to obey. Additionally mother nature has their ever-lasting rules, overwriting all human rules, and which we should obey to keep the envrionment healthy for future mankind. Thus people may have the need to get information about this complex system and tools to make this understandable. Escad is really not good in many things. It is no unversial software. In fact it is really not yet a graph analysis tool (however one could update that functionality). Escad is a worker: get something out of your graph model, make PDF, 3D models, music or other documents which is boring to create by hand.

1.2 State

Currently escad is in development, but only usable for experts in some areas.

1.3 Plans

1. Increase domain functionality with practic use.

2. Create good and clear documentation with examples.

2 Installation

Currently there are no preconfigured packages for convenient installation of escad. However installation should not be to difficult, since escad-development tries to minimize not shipped dependencies.

2.1 Linux or other unix-like systems

1. Install a common lisp system (tested with CLISP).
2. Copy escad directories.
3. Done.

2.2 Windows

This is currently not tested, but may be possible. Take similar steps as described in the previous section.

3 Theory

To understand a software-system it is often easier to understand the theory behind. In escad this is quite simple, since it is practical use of graphs (in informatically or mathematically means). Graphs consist only of symbols and relations (edges). Whenever needed you can add classification info (taxonomy) or functionality (expansion).

3.1 Symbols

In many papers symbols are also called node. However escad has the aim to model things into software. Thus a node is a *variable* for something in our thoughts. A symbol can represent a house, a number, a theory, a joke or whatever you want. This shows the great flexibility of escad.

3.2 Relations

Relation or in computer science called edge creates relations between symbols.

3.3 Taxonomy

With taxonomy you can create a for the model. This is a domain classification which allows you to describe your symbols and relations in more detail. Depending of the taxonomy there may result different (graphical) output or behaviour (functionality).

3.4 Expansion

Expansions, nowadays it may be called *apps*, are programms who can do various things with your graph. This ranges from graphic output to new generated graph-elements. There are many possibilities and you can even write your own expansion(s). Those programms live in the escad environment and can use the provided feature, even of other expansions. However currently there is only a limited set of expansion, but this could increase in time. Feel free to write a expansion for your domain specific needs.

4 User stories

Here we assume it is easier to get what escad can do for you, by telling the user stories we think in which escad is currently usefull. This can increase in the future.

4.1 Input

Because interoperability is important in todays heterogenic software world, there should be a way to get graphs from other software.

4.1.1 graphviz dot

Graphviz is powerful in graphically drawing of graphs. You can import those graphs with .dot extension. However only basic functionality of dot is currently supported.

4.2 Create things

In case you can not do some things in escad yet or other tools may do better, you can export your graph.

4.2.1 graphviz dot

Graphviz is powerful in graphically layout and drawing of graphs.

4.2.2 SVG

SVG is a vector graphic-format viewable through most modern webbrowsers.

4.3 Output

4.3.1 Reporting

Reporting here means to get some basic statistically information about the graph. Additionally some expansions may allow you to dive deeper in the graph and provide some semantic informations too.

4.3.2 Mindmap

This creates mindmaps in SVG format. Very usefull to view in a browser and to make your graph visible.

4.3.3 Flow

A flow is a graph wich models a process. You can use escad to analyse a flow or to create a process a user should go through (e.g. a question+answer quiz).

4.3.4 Creating (technical) documentation

Documentation in the daily practise is often ignored, or - not really better - done worse. You may ask why. Maybe one answer is, people use existing tools which not support very well heterogenic documentation situations.

5 Reference

5.1 Interface

5.1.1 escad via commandline

5.1.2 escad via TCP-Socket

5.1.3 escad via REST-API

5.1.4 escad as library

5.1.5 escad via browser-client

5.2 Functionality by domain

5.2.1 ESCAD

5.3 Functionality by concept

5.3.1 Import

5.3.2 Report

5.3.3 Export

5.4 Expansion programming

6 Questions

Sometimes questions lead to a fast recognition of a problem or help to understand things better. So lets start asking...

6.1 Development

Can i help develop escad? Of course. Just contact the developer in github.

6.2 Usage

Is there a difference in the power of the different escad interfaces? Yes. The most powerful is currently the command line interface. New commands appear first there. However this is no rule. The aim is to keep the functionality equal across the interfaces in a later step.

Why you have choosen common-lisp as the language for escad? That is a good question. Why not javascript or another more popular language? Well may i ask why not lisp? In fact lisp has a very easy synthax, looks very nice in source code ;-) and for many tasks you need not more than three nested parentheses.

6.3 Other

Is there support for other languages as english? No not currently. If you want contribute feel welcome.

Index

Bibliography