Files\\Literature\\Bergel et al. - 2014 - A Domain-Specific Language for Visualizing Softwar - § 2 references coded [0.39% Coverage]

Reference 1 - 0.25% Coverage

Roassal3 is the visualization engine used by the implementation of the domain-specific language described above.

Reference 2 - 0.14% Coverage

The GRAPH language is an internal DSL or a fluent API
built in Pharo.

Files\\Literature\\Bostock and Heer - 2009 - Protovis A Graphical Toolkit for Visualization - § 2 references coded [0.22% Coverage]

Reference 1 - 0.14% Coverage

Protovis, an embedded domain-specific
language [19] for constructing visualizations by composing simple graphical marks such as bars, lines and labels.

Reference 2 - 0.08% Coverage

Protovis is implemented in JavaScript, with rendering support for HTML 5 canvas, SVG, and Flash.

Files\\Literature\\Bostock et al. - 2011 - D³ Data-Driven Documents - § 2 references coded [0.18% Coverage]

Reference 1 - 0.12% Coverage

JavaScript-based selections provide flexibility on top of CSS, as styles can be computed dynamically in response to user events or changing data.

Reference 2 - 0.06% Coverage

D3 provides a declarative framework for mapping data to visual elements.

Files\\Literature\\Heer and Bostock - 2010 - Declarative Language Design for Interactive Visual - § 2 references coded [0.36% Coverage]

Reference 1 - 0.13% Coverage

We describe the design of the Protovis specification language and its implementation within an object-oriented, statically-typed programming language (Java).

Reference 2 - 0.23% Coverage

Our prior work on Protovis [1]—an embedded domain-specific language (DSL) for web-based visualization in JavaScript — has demonstrated that a declarative language can simplify visualization specification while supporting a high-degree of expressiveness and customization.

Files\\Literature\\Ledur et al. - 2017 - A High-Level DSL for Geospatial Visualizations wit - § 3 references coded [0.47% Coverage]

Reference 1 - 0.23% Coverage

With the goal of simplifying the creation of visualiza-
tions for large-scale geospatial data, we propose an external DSL that provides a high-level specification language called GMaVis

Reference 2 - 0.13% Coverage

Our compiler recognizes the DSL source code and generates geospatial data visualizations application code.

Reference 3 - 0.11% Coverage

The second phase generates the final visualization using HTML/Javascript and Google Maps API.

Files\\Literature\\Li et al. - 2018 - ECharts A declarative framework for rapid constru - § 2 references coded [0.08% Coverage]

Reference 1 - 0.02% Coverage

Declarative languages

Reference 2 - 0.05% Coverage

The declarative option is a hierarchical JSON object

Files\\Literature\\Liu et al. - 2021 - Boba Authoring and Visualizing Multiverse Analyse - § 2 references coded [0.26% Coverage]

Reference 1 - 0.25% Coverage

We design a domain-specific language (DSL) to aid the authoring of multiverse analyses. The DSL formally models an analysis decision space, providing critical structure that the visual analysis system later leverages. With the DSL, users annotate the source code of their analysis to indicate decision points and alternatives, and provide additional information for procedural dependencies between decisions.

Reference 2 - 0.01% Coverage

integrated DSL

Files\\Literature\\Logre and Déry-Pinna - 2018 - MDE in Support of Visualization Systems Design a - § 4 references coded [0.30% Coverage]

Reference 1 - 0.05% Coverage

We provide a Domain Specific Language (DSL) to design the system

Reference 2 - 0.04% Coverage

to model variability is to use a Feature Model (FM)

Reference 3 - 0.09% Coverage

Each characterized widget is expressed as a variability-free feature model (FM) in the Familiar language

Reference 4 - 0.12% Coverage

Using the provided XText12 concrete syntax, one can instantiate conform models, as illustrated by the running example in the following code.

Files\\Literature\\Logre et al. - 2014 - Sensor Data Visualisation A Composition-Based App - § 3 references coded [0.62% Coverage]

Reference 1 - 0.32% Coverage

a meta-model that tackles the first challenge identified in Sec. 2: “How one can design a monitoring dashboard according to her very own needs?”. To address this challenge, the key idea is to tame the complexity of dashboard design using a dedicated meta-model

Reference 2 - 0.14% Coverage

each column is translated into a tool-ready representation of each descriptor, using the Familiar language

Reference 3 - 0.17% Coverage

The resulting feature model, based on a simplified version of the AmCharts
that only contains 12 widgets is depicted as a feature diagram

Files\\Literature\\Morgan et al. - 2017 - VizDSL Towards a Graphical Visualisation Language - § 1 reference coded [0.39% Coverage]

Reference 1 - 0.39% Coverage

VizDSL is based on IFML and Agile Visualisation. We have chosen IFML because (1) it is an accepted standard by the OMG and well established through its predecessor WebML, (2) it provides a modelling languages that sits between a UI and the code, and (3) it can be extended through its specification written in UML.

Files\\Literature\\Rojas et al. - 2020 - Cities-Board A Framework to Automate the Developm - § 4 references coded [1.14% Coverage]

Reference 1 - 0.49% Coverage

his article introduces cities-board, a framework to automate the development of smart cities dashboards based on model-driven engineering. Cities-board proposes a graphic domain-specific language (DSL) that allows the creation of dashboard models with concepts that are closer to city authorities. Cities-board transforms these dashboards models to functional code artifacts by using modelto-model (M2M) and model-to-text (M2T) transformations.

Reference 2 - 0.46% Coverage

a framework to automate
the development of smart cities dashboards. Cities-board is based on model-driven engineering (MDE) [16] and its main contribution is a graphic domain-specific language (DSL) that allows the creation of smart cities' dashboards models. Such models conform to a cities-board (CDB) metamodel, which abstracts the cities dashboards domain using concepts closer to end users (e.g., city authorities).

Reference 3 - 0.08% Coverage

We use the eclipse modeling framework (EMF)7 to implement these components.

Reference 4 - 0.10% Coverage

Cities-board includes a graphic DSL based on the CDB metamodel, which is developed using Sirius.

Files\\Literature\\Satyanarayan and Heer - 2014 - Authoring Narrative Visualizations with Ellipsis - § 1 reference coded [0.23% Coverage]

Reference 1 - 0.23% Coverage

a JavaScript-based domain-specific language (DSL) provides a runtime for interactive narratives, and a web-based interface enables direct-manipulation authoring of scenes, annotations and user interactions.

Files\\Literature\\Satyanarayan et al. - 2017 - Vega-Lite A Grammar of Interactive Graphics - § 1 reference coded [0.19% Coverage]

Reference 1 - 0.19% Coverage

Drawing from Vega, VegaLite uses a portable JSON (JavaScript Object Notation) syntax that permits generation from a variety of programming languages. VegaLite specifications are compiled to full Vega specifications, hence the expressive gamut of Vega-Lite is a strict subset of that of Vega.

Files\\Literature\\Smeltzer and Erwig - 2018 - A domain-specific language for exploratory data vi - § 2 references coded [0.56% Coverage]

Reference 1 - 0.22% Coverage

our DSL is embedded in Purescript3, which is similar to Haskell, but has the additional feature of making it easy to produce browser-based graphics. Readers familiar with Haskell should have no problem understanding the definitions contained in this work.

Reference 2 - 0.34% Coverage

Before we can integrate variability into visualizations, we need a systematic way of structuring variation. To that end we make use of the choice calculus [7], which is a formal model for variation based on named choices. Each choice consists of a list of alternatives, and its name, called a dimension, synchronizes the selection of a specific alternative with the selection in other choices of the same name.

Files\\Literature\\Smeltzer et al. - 2014 - A transformational approach to data visualization - § 2 references coded [0.34% Coverage]

Reference 1 - 0.15% Coverage

As a possible solution, we present a Haskell-embedded domain-
specific language (DSL) designed to provide a concise but expressive way to construct, compose, and transform data visualizations.

Reference 2 - 0.19% Coverage

In addition to making some higher-level abstractions available than those provided by D3 and Protovis, our work is distinguished from this through the use of functional programming idioms and type classes, particularly the use of functors and monads.

Files\\Literature\\Sun et al. - 2021 - TRANSIT-GYM A Simulation and Evaluation Engine fo - § 3 references coded [0.34% Coverage]

Reference 1 - 0.10% Coverage

A novel DSML that allows intuitive specification and variation of transit scenarios.

Reference 2 - 0.09% Coverage

We use metamodeling by developing a DSML that is specific to transit simulations.

Reference 3 - 0.15% Coverage

SUMO is used to conduct microscopic traffic simulation, wherein vehicles, public transport and persons are modelled explicitly.

Files\\Literature\\Teng et al. - 2021 - Sketch2Vis Generating Data Visualizations from Ha - § 2 references coded [0.32% Coverage]

Reference 1 - 0.27% Coverage

We propose a solution to the training data set problem
that applies (1) a domain-specific language (DSL) and its grammar to generate source code for data visualizations randomly

Reference 2 - 0.05% Coverage

Our DSL uses the XML-based syntax

Files\\Literature\\Vázquez-Ingelmo et al. - 2018 - Domain engineering for generating dashboards to an - § 2 references coded [0.73% Coverage]

Reference 1 - 0.42% Coverage

In this case, a domain specific language (DSL) has been implemented with XML technology [20] to provide a structured file to the code generator so it can easily extract the features and inject them into the code templates (with Jinja2

Reference 2 - 0.31% Coverage

his code generator is implemented in Python, and it is in charge of processing the XML configuration files and inject the functionalities through the code templates logic.