Files\\Literature\\Bergel et al. - 2014 - A Domain-Specific Language for Visualizing Softwar - § 3 references coded [0.60% Coverage]

Reference 1 - 0.36% Coverage

The key difference between GRAPH and traditional graph description languages is the relation between visual elements and dimensions and the application to visualize.

Reference 2 - 0.07% Coverage

small and consistent language

Reference 3 - 0.17% Coverage

Produced visualization may be seen either in the Pharo window or in a web browser.

Files\\Literature\\Bostock and Heer - 2009 - Protovis A Graphical Toolkit for Visualization - § 3 references coded [0.27% Coverage]

Reference 1 - 0.10% Coverage

enables concise visualization definitions with a large expressive range and a minimum of intervening abstractions.

Reference 2 - 0.06% Coverage

Although high-level abstractions allow concise specifications,

Reference 3 - 0.11% Coverage

they may appear magical if the user does not fully understand how the specification translates to the resulting visualization.

Files\\Literature\\Bostock et al. - 2011 - D³ Data-Driven Documents - § 1 reference coded [0.28% Coverage]

Reference 1 - 0.28% Coverage

With our prior work on Protovis [2, 13] we have instead advocated for declarative, domain specific languages (DSLs) for visualization design. By decoupling specification from execution details, declarative systems allow language users to focus on the specifics of their application domain, while freeing language developers to optimize processing.

Files\\Literature\\Heer and Bostock - 2010 - Declarative Language Design for Interactive Visual - § 3 references coded [1.18% Coverage]

Reference 1 - 0.14% Coverage

separating specification from execution, declarative languages can simplify development, enable unobtrusive optimization, and support retargeting across platforms.

Reference 2 - 0.30% Coverage

Declarative languages often simplify programming tasks by requiring that a developer specify what the results of a computation should be rather than how the results should be computed. The separation of specification from execution allows language users to focus on the specifics of their application domain, while freeing language developers to optimize processing.

Reference 3 - 0.73% Coverage

Moreover, contemporary visualization design tools must address a number of new technical challenges. Not least among these is the increasing heterogeneity of commodity hardware and interactive devices. Visualization tools should ideally support interfaces ranging from traditional desktop applications, to browser-based web clients, to multi-touch mobile devices. Furthermore, visualization tools should

effectively capitalize on hardware trends such as multi-core computing and specialized graphics hardware. While point designs exist for each of these areas, the field currently lacks a consistent approach to visualization design and deployment across heterogeneous platforms. To address these issues, we argue for a break with current component model architectures and instead advocate the design of declarative, domain-specific languages for interactive visualization.

Files\\Literature\\Ledur et al. - 2017 - A High-Level DSL for Geospatial Visualizations wit - § 5 references coded [0.68% Coverage]

Reference 1 - 0.20% Coverage

Our goal is to improve the geospatial visualization creation experience by reducing the effort required for pre-processing data and visualization implementation.

Reference 2 - 0.16% Coverage

Therefore, we implemented a DSL to abstract complexities and let users focus on gaining quick insights in the geospatial data domain.

Reference 3 - 0.06% Coverage

abstracting all geospatial visualization steps

Reference 4 - 0.03% Coverage

coding development time

Reference 5 - 0.22% Coverage

Moreover, even not considering the fact that GMaVis avoids the implementation or use of external software to process data, it improves the developing time on geospatial visualization

Files\\Literature\\Li et al. - 2018 - ECharts A declarative framework for rapid constru - § 2 references coded [0.98% Coverage]

Reference 1 - 0.30% Coverage

Declarative languages such as D3.js (Bostock et al., 2011) and Vega (Satyanarayan et al., 2016) are popular tools for building visualizations. With the encapsulation of underlying data transformations and control flow exposed to users, these declarative languages allow users to focus on the visual design.

Reference 2 - 0.68% Coverage

Easy-to-use. There are some difficulties for users to learn the
visual representations if a declarative language is employed. It is desirable to allow users to focus on the design of the visualization rather than on the use of some tools. Rich built-in interactions. Efficient data exploration and analysis demand a wealth of configurable interactions. ECharts designs and implements rich built-in interactions that are attached to each chart type, minimizing the requirement of customization of user. High performance. By introducing a streaming system architecture and incremental rendering mode, high performance is achieved with ECharts, even when handling millions of data points.

Files\\Literature\\Liu et al. - 2021 - Boba Authoring and Visualizing Multiverse Analyse - § 2 references coded [0.27% Coverage]

Reference 1 - 0.13% Coverage

Rather than managing myriad analysis versions in parallel, the Boba DSL allows users to specify the shared portion of the analysis code only once, alongside local variations defining alternative analysis decisions.

Reference 2 - 0.14% Coverage

The Boba Visualizer facilitates assessment of the output of all analysis paths. We support a workflow

where users view the results, refine the analysis based on model quality, and commit the final choices to making inference.

Files\\Literature\\Logre and Déry-Pinna - 2018 - MDE in Support of Visualization Systems Design a - § 2 references coded [0.22% Coverage]

Reference 1 - 0.15% Coverage

While the abstract syntax captures the structure of the domain in a meta-model, a concrete syntax allows the design of conform models through textual or graphical representations.

Reference 2 - 0.07% Coverage

provide to each user the relevant support to ease their task in the diagnosis process.

Files\\Literature\\Logre et al. - 2014 - Sensor Data Visualisation A Composition-Based App - § 2 references coded [1.35% Coverage]

Reference 1 - 1.15% Coverage

This meta-model allows a user to focus on the way she wants to compose her data, and does not require implementation knowledge. It focuses on the different visualisation concerns one can apply to a given datasets, and is not bound to any concrete library implementation. Thus, at this level of abstraction, the user is completely free to work. The binding with existing visualisation libraries, as well as the introduction of new libraries with respect to this meta-model corresponds to C2 (see Sec. 4). Restricting the domain to its essence, a designers works according to three dimensions while designing a dashboard: (i) the data involved in the dashboard according to her monitoring needs (i.e., “What am I visualising”), (ii) the different visualisation concerns applied to these data (i.e., “How do I visualise it?”) and finally (iii) the spatial and temporal layout of the dashboard (i.e., “Where and when do I visualise it?”).

Reference 2 - 0.19% Coverage

As the meta-model has been defined to meet business requirements of dashboard designers, it seems reusable for sensor data visualization in other contexts.

Files\\Literature\\Morgan et al. - 2017 - VizDSL Towards a Graphical Visualisation Language - § 5 references coded [0.99% Coverage]

Reference 1 - 0.41% Coverage

Domain specific
languages
(DSLs) play a crucial role in MDE and represent languages for a specific purpose that are highly abstract and easy to use. In this paper we propose a new language VizDSL for creating interactive visualisations complex data and information structures for enterprise systems interoperability.

Reference 3 - 0.10% Coverage

benefits of visualising the semantics of data using a graphical notation.

Reference 4 - 0.20% Coverage

interactive visualisation can be implemented quickly using the VizDSL language without writing code which makes it easier to design for non-programmers.

Reference 5 - 0.26% Coverage

Model-driven engineering (MDE) is used to mitigate the design and engineering challenges posed by integration and interoperability in the context of Big Data and resulting complex software systems.

Files\\Literature\\Rojas et al. - 2020 - Cities-Board A Framework to Automate the Developm - § 2 references coded [0.27% Coverage]

Reference 1 - 0.17% Coverage

This framework is composed of a graphic DSL that enables the creation, manipulation, and transformation of dashboard models in a closer language to end users

Reference 2 - 0.10% Coverage

This language is defined by the CDB metamodel, which abstracts the cities dashboards domain.

Files\\Literature\\Satyanarayan and Heer - 2014 - Authoring Narrative Visualizations with Ellipsis - § 3 references coded [0.50% Coverage]

Reference 1 - 0.05% Coverage

enabling storytelling without programming

Reference 2 - 0.32% Coverage

Our model of visualization coordination decouples narra-
tive structure from visualizations, such that the two can be developed in a more independent fashion. Although narrative and visualization are tightly integrated in a user's experience, a decoupled implementation can facilitate authoring.

Reference 3 - 0.13% Coverage

As we demonstrate, our model allows authors to incorporate independent, pre-existing visualizations into their stories.

Files\\Literature\\Satyanarayan et al. - 2017 - Vega-Lite A Grammar of Interactive Graphics - § 2 references coded [0.12% Coverage]

Reference 1 - 0.06% Coverage

High-level languages can also enable search and inference over the space of visualizations.

Reference 2 - 0.06% Coverage

facilitate programmatic generation and retargeting of interactive visualizations

Files\\Literature\\Smeltzer and Erwig - 2018 - A domain-specific language for exploratory data vi - § 2 references coded [0.53% Coverage]

Reference 1 - 0.29% Coverage

The DSL groups functionality in to different layers. At the bottom-most layer, users can work directly with the core data types to customize every detail and make intricate extensions. Higher layers of the DSL provide increasingly abstracted combinator functions for cases when the details are less important than quickly building charts.

Reference 2 - 0.24% Coverage

First, it allows the composition of visualizations into larger visualizations. This could be spatial composition or something more sophisticated such as stacking chart elements. Second, it allows visualizations to be transformed between Cartesian and polar coordinate systems.

Files\\Literature\\Smeltzer et al. - 2014 - A transformational approach to data visualization - § 2 references coded [0.24% Coverage]

Reference 1 - 0.08% Coverage

problems such as data handling and rendering can be handled by the host language and treated separately.

Reference 2 - 0.16% Coverage

Additionally, the use of Haskell as a host language gives us access to its rich type class hierarchy which, in turn, can provide powerful operations using a style that is already familiar to many Haskell

programmers.

Files\\Literature\\Sun et al. - 2021 - TRANSIT-GYM A Simulation and Evaluation Engine fo - § 3 references coded [0.65% Coverage]

Reference 1 - 0.20% Coverage

As mentioned before, the scenario construction is performed through the use of DSML developed on the textual metamodeling framework TextX and used in simulation configuration.

Reference 2 - 0.23% Coverage

The DSML interpreter provides an automatic way for parameter flexibility on vehicle type, scenario generation, and block and vehicle to trip mapping constrains checker in customizable SUMO simulation.

Reference 3 - 0.22% Coverage

enables intuitive and rapid specification of scenario variations in transit demand, route networks, and vehicle trip assignments as well as simulating them on integrated cloud backend.

Files\\Literature\\Teng et al. - 2021 - Sketch2Vis Generating Data Visualizations from Ha - § 2 references coded [0.28% Coverage]

Reference 1 - 0.18% Coverage

A key advantage of using a DSL is that it makes the training process independent of the target visualization library.

Reference 2 - 0.10% Coverage

the language design can be tailored to facilitate faster learning.

Files\\Literature\\Vázquez-Ingelmo et al. - 2018 - Domain engineering for generating dashboards to an - § 1 reference coded [0.09% Coverage]

Reference 1 - 0.09% Coverage

management of customization within this domain.