StarGate: A Unified, Interactive Visualization of Software Projects

Michael Ogawa* Kwan-Liu Ma†

Visualization & Interface Design Innovation (VIDi) Research Group
University of California, Davis

ABSTRACT

With the success of open source software projects, such as Apache and Mozilla, comes the opportunity to study the development process. In this paper, we present StarGate: a novel system for visualizing software projects. Whereas previous software project visualizations concentrated mainly on the source code changes, we literally place the developers in the center of our design. Developers are grouped visually into clusters corresponding to the areas of the file repository they work on the most. Connections are drawn between people who communicate via email. The changes to the repository are also displayed. With StarGate, it is easy to look beyond the source code and see trends in developer activity. The system can be used by anyone interested in the project, but it especially benefits project managers, project novices and software engineering researchers.

Keywords: Software Visualization, Information Visualization, Social Networks

Index Terms: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Systems development; H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Graphical User Interfaces (GUI)

1 INTRODUCTION

In software projects with many contributors, it is essential to have a version control system to keep track of the code and allow for simultaneous editing. It is the job of the system to manage files and handle the reconciliation of changes. Version control systems in use today include CVS and Subversion. The activity occurring in software repositories is a challenge to visualize because there are many people making many changes to many files. In other words, the repository keeps track of many variables, such as which files were edited, when they were edited, who edited them, and what changes were made. The status quo tool for navigating a repository is a text-based browser, such as ViewVC [1]. Though these are useful to see details about the repository, they cannot see the "forest" of the project through the "trees" of files.

In a large, open source software project, developers are often distributed around the world. In order to communicate amongst themselves, the developers often maintain a shared email list. We can use data from this mailing list (an archive of which is usually made publicly available in open source projects) to infer a social network between developers.

Our goal is to use information visualization techniques to assist the user in understanding the complex interactions between developers and software repositories. We also want to aid in understanding the evolution of the software project over time. To this end we have developed StarGate: a system which visualizes the code repository and social network of developers associated with a software project in one integrated representation.

Our system benefits different users in distinct ways:

Project Novices People just starting to work on an existing project will want to know who is working on specific areas of the code in order to ask them questions. Focusing questions towards a specific group of developers should yield a more thorough answer and bother less people.

Project Managers Those in charge of overseeing software projects naturally want an overview in order to understand the complex interactions between software and people. Our system can show them the contributions made by specific people and how the developers are allocated among the files in the repository.

Software Engineering Researchers Researchers in academia are interested in finding the relationships between project organization and the source code. For example, they may ask the question: "Do two authors who communicate with each other tend to work on the same areas of source code?" Our application can help them visualize the answers to this and other questions.

In designing our system, we have combined several information visualization techniques in a unique and visually pleasing way. We use the center of a radial space-filling hierarchy to display a node-link network representation, rather than just empty space. The network is not only inside the hierarchy, but also linked to it through the node positions. In doing so, our application shifts the focus of software repository visualizations from the typical file- and source code-centric view to an author-centric view. We offer a variety of views of the data: from overviews of the relations between people and the evolution of the project, to details of the changes made to program files. StarGate incorporates these views in an aesthetic and compact fashion.

2 RELATED WORK

Our visualization of software projects relates to software repositories and developer social networks. In this section we discuss previous work in these areas.

2.1 Software Repository Visualizations

For the most part, software repository visualization has focused on the files and source code rather than the developers. A good reference for the progress in software visualization can be found in Storey et al.'s survey paper [18]. In this section we will discuss visualizations which are not covered in [18] or are particularly relevant to our application.

A more recent application, Augur [12], uses the SeeSoft [11] paradigm of line-by-line source code visualization and adds visualizations of developer activity. These additional visualizations are

*e-mail: msogawa@ucdavis.edu
†e-mail: ma@cs.ucdavis.edu

IEEE Pacific Visualisation Symposium 2008
4 - 7 March, Kyoto, Japan
978-1-4244-1966-1/08/$25.00 ©2008 IEEE

located in separate inset windows and linked to the main visualization. Our application seeks to unify the different information in one view.

Visual Insights' ADVIZOR, described in [10], offers more views of the software change process. These include matrix, cityscape and network views. The user can display multiple views at a time, referred to as "perspectives." However, it lacks a *unified* view that links the information in the same space.

CVSscan [20] uses a visualization similar to History Flow [19] to depict the evolution of code. CVSgrab [2] visualizes the evolution of software files.

ViewVC [1] is "a browser interface for CVS and Subversion version control repositories." It allows for textual exploration of the directories, changes to files, and developer logs. There is even a tool for highlighting the differences between file versions. Unfortunately there are no visualizations of the repository activities.

The Cenqua Fisheye[1] is a commercial software tool for visually interacting with software repositories. Like ViewVC, it lets users easily navigate the directories and files. The interface is rich with textual information, however the graphical displays are limited to line charts and histograms. They are able to display, for each file, the line count over time and the edit volume over time. The Cenqua Fisheye gives detailed information at the file and source code level, but not an overview of the entire project, such as which authors are interested in which areas of the repository.

Burch et al. visualized the Mozilla software project in [5]. Their approach, however, did not take developers into account, but visualized the binary association rules between files with visual constructs such as pixelmaps and parallel coordinates.

The GEVOL system, described in [9] visualizes graphs inherent in a software system. These structures include inheritance graphs, call graphs, and control flow graphs. They display a series of time steps in the life of the system. Their specialized layout algorithm preserves the mental map of the system so that comparisons between timesteps are easier for the user.

The Software Development Bloom [13] should be considered author-centric because its main axis is used for authors. However, it does not provide detailed information about the *activities* of the authors. Bloom sorts the contributors by their activity type (code vs. comment) and presents them as wedges in a circular layout. It gives information on the amount of recent code/comment activity of people, but not what they coded/commented.

File repository visualization has mostly been file- and source code-centric. Our approach is author-centric.

2.2 Software Social Networks

Social networks are currently a hot topic in visualization. Our work specifically relates to software social networks.

Bird et al. [3] extracted and analyzed the email network of the Apache project. They found that the network has small-world properties and obeys a power law. Ogawa et al. [16] subsequently created an evolving network visualization of the same data. They found links between the communication activity on the mailing list and development activity in the source code.

While StarGate also includes a social network visualization, it is complemented by and linked to the software repository structure.

2.3 Visual Design

StarGate's visual paradigm was inspired by the work of Livnat et al [14]. Their VisAlert application places a node-link network representation in the center of a table of events wrapped into the form of a ring. We improve upon their design in several aspects.

First, we strongly link the positioning of the central information to the surrounding rings. That is, the author nodes in the center

[1] http://www.cenqua.com/fisheye

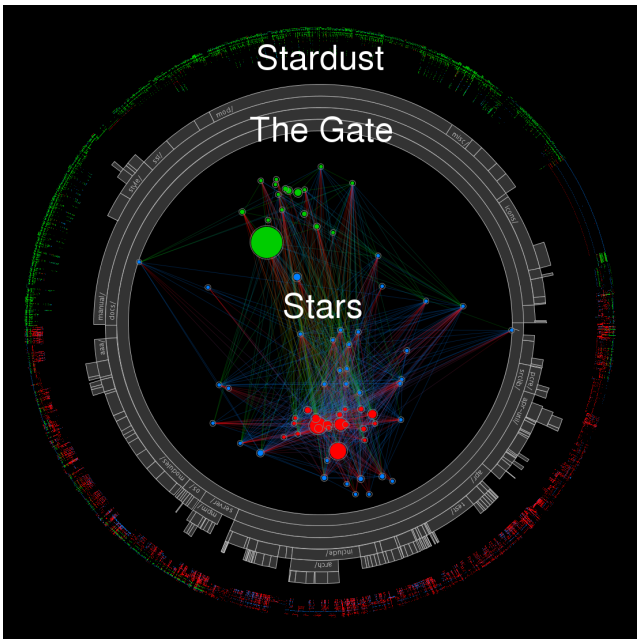


Figure 1: A typical view of StarGate.

circle are pulled towards the areas of the file repository that they tend to work on. This positioning is more useful than, say, a force-directed layout because we can group programmers by code expertise. Then, when the mailing list connections are drawn between programmers, we can see the communication patterns between groups. Second, the information in the ring surrounding the central network of authors is hierarchical rather than tabular. Third, we introduce an outer layer of file information beyond the inner hierarchical rings.

3 Data Sources

The system is designed to work with the following data: the software repository and the project mailing list archive.

3.1 Version Control Repository

In large open source software projects, it is necessary to have a version control system. This system keeps track of revisions and allows for the simultaneous editing of files. For testing our application, we use the data from public CVS and Subversion servers. This data includes the directory hierarchy and file modification records. Each file modification record includes who edited the file, when it was edited, and how many lines were added or removed.

3.2 Project Email List Archive

Since the software projects we are interested in are open source, the developers are distributed throughout the world. In order to communicate with each other, the project members utilize an email list, which can be examined through a publicly available online archive. The data we use in this paper was gathered from the Apache and PostgreSQL projects, as described in [3]. The dataset is a list of all emails containing sender, receiver and time information. Emails which are replies are considered to be connections between sender and receiver. Thus the mailing list can be thought of as a network of project participants.

4 A Tour of StarGate

In this section we will guide you through StarGate's various components.

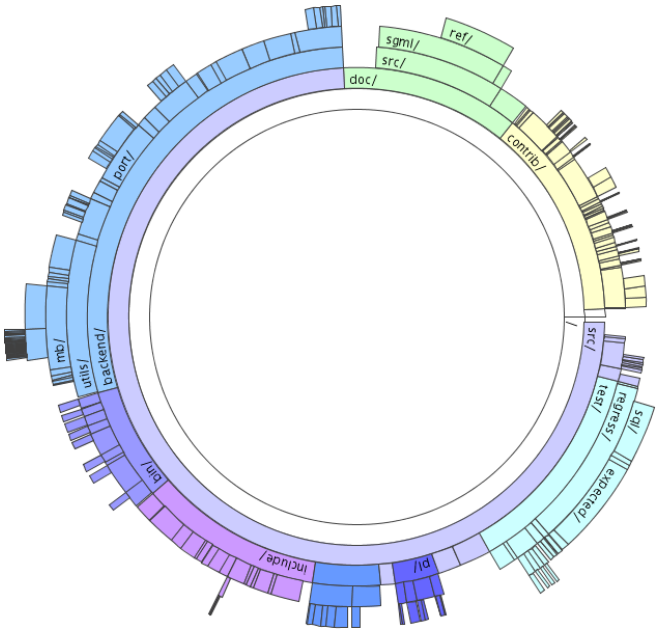
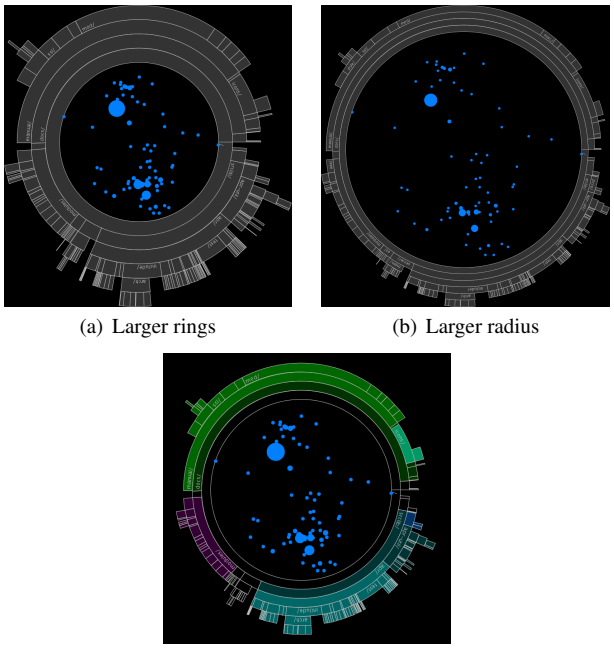


Figure 2: The Gate component. This is the directory hierarchy of the PostgreSQL version control repository. The documents directory is colored green (at the top) and the source code directory is colored in various shades of blue.



(a) Larger rings (b) Larger radius

(c) Colored rings

Figure 3: Varied Gate visual parameters.

4.1 The Gate (The Ring of Directories)

The Gate forms the base of the visualization. To visualize the software repository directory structure, we use a space-filling radial hierarchy. This visual representation was first proposed by [8] and refined by [17] and [22], among others. We chose this representation because, as demonstrated later, it allows us to pack more coherent information both inside and outside.

The inner-most ring of the Gate represents the root of the directory structure. From here, the levels of the hierarchy grow outwards from the center as concentric ring slices (Figure 2). The size of the slices is proportional to the number of files within that directory, including the subdirectories. Thus a large directory's ring slice will span more radii than a small directory's.

If there is enough room in a ring slice, the directory name appears inside of it. For smaller directories, hovering over them with the mouse will display the directory name in a popup.

The user can change the size, shape and color of the Gate (Figure 3). The size of the ring thickness and center radius can be increased or decreased with slider interfaces. Directory ring slices can be colored by the user, with the color percolating through the subdirectories. The directories in the Gate can also be used to select developers who have made contributions to the files contained within.

4.2 The Stars (Developers in the Center)

In the center of the Gate, the developers are represented as colored circles, or "stars." The size of each star is proportional to the number of file modifications the developer has performed. Stars are positioned according the areas of the repository the developers have worked on. For example, if a developer has worked mostly in the directories appearing on the right-hand side of the Gate, then they will be positioned to the right of center, towards the right-hand side of the Gate. This use of positioning allows the observer to see, at a glance, the developers concentrating on specific areas of the repository. The spatial grouping of developers working on the same areas of the repository naturally leads to clustering. The user can easily see, for example, those who have focused their work on the documentation.

4.2.1 Details of Star Positions

The position of a developer's star is determined by which files he or she has modified and where those files are located in the repository hierarchy. The star is "pulled" towards areas of the repository in which there is more development activity. The placement algorithm is similar to [15]'s anchored maps method, but differs in that we weight the anchored connections and only draw links within the social network. To calculate the appropriate placement of the star in regards to this notion, we use the *centroid* of the surrounding modified files. Let p be the developer whose star we want to place. Let F_p be the set of files that p has modified. Also, define the function $position(f)$ to return the coordinate on the surrounding ring which file f is assigned to. Then the equation for the position of developer p is

$$\text{centroid}_p = \frac{\sum_{f \in F_p} position(f)}{\|F_p\|}.$$

We consider that each file may not be equally important to the person who modifies them. For this reason, we weight the "pull" of each file by the number of times it was modified (committed) by a person. This introduces the function $changed(f,p)$, which returns the number of times file f was modified by person p.

$$\text{weighted-centroid}_p = \frac{\sum_{f \in F_p} [position(f) \cdot changed(f,p)]}{\sum_{f \in F_p} changed(f,p)}$$

Note that each file position is on a circle, so the weighted centroid of any set of files cannot be outside the circle. A problem may arise, discussed further in section 6, with the misinterpretation of a star's position for an unusual file modification pattern.

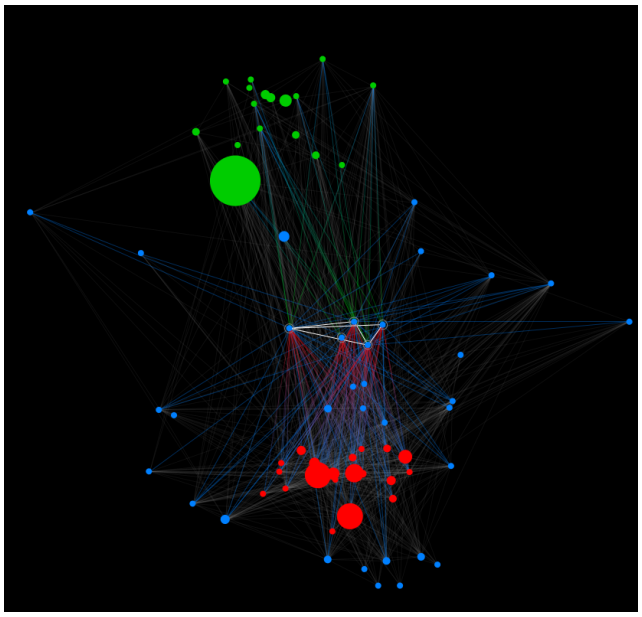


Figure 4: The developer stars, sized in proportion to their project contributions. The top cluster (green) is the documenters. The bottom cluster (red) is the core developers. The blue stars in the middle have been selected. Their network connections are colored.

4.2.2 Star Interaction

Brushing over the each star with the cursor displays the developer name next to their star in a popup. More robust interaction can be accomplished as follows.

Initially, all stars are the same color (e.g. blue in the figures). They are uniformly colored (rather than, say, randomly assigned a color) because we expect the user to use color as an analysis tool. The user can choose the color for each star by right-clicking and selecting "Color" from the popup menu. The user may also assign a color to a group of stars by selecting "Color Group" from the menu.

Left-clicking on a star selects it. Multiple stars may be selected by dragging a box around the desired ones (i.e. rubber banding). Selected stars are differentiated from unselected stars by the following properties:

- They are distinguished by a visible halo.
- The developer name is displayed beside them (optional).
- Their connections are shown in the social network as constellations (Section 4.2.3).
- Their file edit history is shown as stardust (Section 4.3).
- During time animation, they leave a visible trail (Section 4.5).

With many stars visible in the center, it may be difficult to find a specific person. We provide a text field for the user to enter a name or part of a name. All stars containing the text entered by the user are then selected. The user can also select stars based on files. The user can enter the name of a file in another text field and all developers who have modified that file (or those files, if the text matches more than one file) are selected.

4.2.3 Constellations: Developer Connections

The center area is also used to visualize the social network of developers. Initially, there are no visible edges between the stars. When

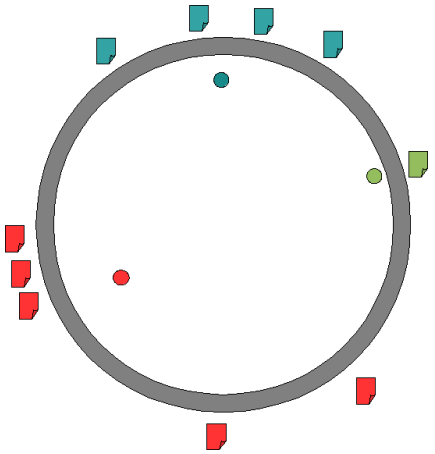


Figure 5: Positioning the stars. The positions are determined by the weighted centroid of the files the developer has modified. More modifications to a file means more weight is given to a file.

the user selects any star, the star's neighbor edges are displayed and colored according to star color. Additionally, when multiple stars are selected, opaque edges between the selected developers who are also connected in the social network are drawn (Figure 4).

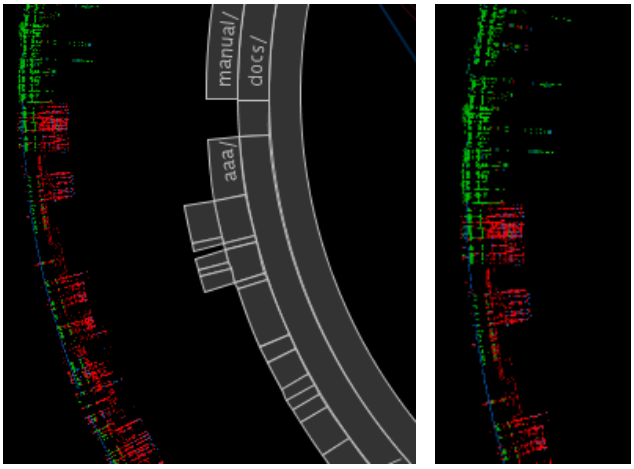
The entire social network can also be viewed via a slider that controls the opacity of the edges. Initially the opacity value is zero (completely transparent), but the user can increase it to show all the connections at once.

Other social network visualizations, such as [10], have used a force-directed algorithm to layout the nodes. Our method has several advantages over the force-directed approach:

- Our placement is deterministic. That is, the positions of the stars for a given dataset remain the same every time the program is run. Force-directed methods start with a random placement and refine the layout from there.
- It is easy to find a group of developers. Simply examine the stars in proximity to the area of interest in the repository. Since a force-directed layout changes every time it is run, the resulting star positions will very likely not be linked to the repository.
- We are able to show both development and communication relationships between people, instead of only the communication relationships. The positions of the stars imply the developement relationships: clusters of stars work on the same general areas of the repository. The edges between the stars show the communication relations: an edge means two people have corresponded through the email list. A force-directed layout only clusters based on the communication network.
- Since it is not an iterative process, our weighted positioning runs faster than a force-directed layout.

4.3 The Stardust (The Outer Ring of Edit History)

The outer ring of stardust (Figure 6) represents the files which selected authors have modified. Each file's edit history is placed along a thin line extending from its place in the directory hierarchy. It can be thought of as a timeline, with the earliest events towards the center and later events towards the outside. Edits are represented as dots along the timeline. So edits occurring at the beginning of the project will be on the inner side of the ring and later edits will be towards the outside. The color of the dot is the same as the star color of the developer who modified it. So an early developer will have



(a) Stardust (b) Detail

Figure 6: The stardust. Each line represents a file. Each colored dot on the line represents a change to the file. Dot color corresponds to the developer that made that change. Time flows radially outward.

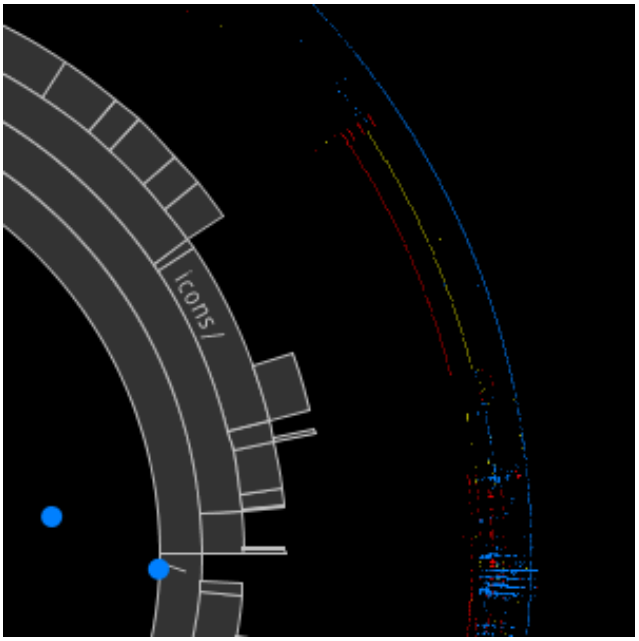


Figure 7: Three bursts of longitudinal activity in the `/docs/icons/` directory. This indicates that the files in the directory were all modified at about the same time. In chronological order, they were made by Roy T. Fielding (red), Martin Kraemer (yellow) and Jean-Jacques Clar (blue).

dots towards the inside and a later developer will have dots towards the outside. Initially, the stardust is not visible. When authors are selected via the stars, the files they have modified are visible as stardust.

The first impression the stardust gives the user is the *amount* of files the developer has modified. A developer with very little activity will not have much stardust to show for it, while a busy developer will have a lot of stardust. The second piece of information gleaned from the stardust is *when* the developer modified the files. One interesting pattern emerges when developers have bursts of activity, shown as a circular arc through the files (Figure 7). Another pattern emerges when developers take "vacations" from the project, showing lots of activity at one time, then a break with no modifications, then more activity later. This is seen as bursts of color interrupted by grey.

4.4 File Details

Once the user has selected stars and examined the related stardust, she or he may want to view details about the files in the repository. Clicking on one of the stardust spokes will pop up a file details dialog window (not shown). In this window there is a table containing the file modification history. The history is a list of modification events. Each event includes a developer name, date, number of lines added and number of lines removed.

4.5 Time Travel

The project timeframe does not necessarily have to be from the very beginning of the project to the most current time in the dataset. With the time slider, located at the bottom of the window, the user can change the latest time in order to see project evolution. Changing the time affects the stardust (only file modifications up to the chosen time are shown) and the positions of the stars (because future developer modifications are not included in the calculations).

With the press of a button, the time can be incremented automatically as an animation. The user will see the stars move around as the developers work on different areas of the source code repository (Figure 8). The stardust will increase as the number of file changes increase.

StarGate also has the option to track the movement of stars during an animation, which we call *star trails*. At each time step, the position of a selected star is recorded in a list. A line is displayed connecting all of the star's previous positions along with the star's current position, much like the tail of a comet (Figure 9). In this way the user can see a developer's history within the repository.

5 Case Study: The Apache Project

We now present a case study using data pertaining to the Apache project. This data was collected by the authors of [3] and details of the process can be found in their paper. The Apache project dataset contains 72 developers, 188 directories and 4,019 files. The data spans eight years: from July 3, 1996 to November 14, 2004. Note that the project actually started in 1995—one year before the earliest time in our dataset.

The project management is a meritocracy, meaning that people who have contributed useful code are allowed to have more freedom to contribute. Developers are classified as part of the *Project Management Committee*—the group with the most authority—or as *committers*.

We start with an examination of the Gate, showing the repository hierarchy (Figure 10). It is divided into two major sections: `/docs/` at the top and `/srclib/` at the bottom. With this in mind, we can also see that the stars in the center are divided roughly into two major clusters. We color the top cluster green and the bottom cluster red for clarity in all images of Figure 11. Figure 11(a) has the core developer cluster highlighted. We can see that their communications network is quite dense; it appears that everyone in the group talks to everyone else. Contrast this with Figure 11(b) where only the documenters at the top are highlighted. Their network is sparse, meaning that documenters tend to not talk to each other. But interestingly, when we highlight both clusters in Figure 11(c), we can see that the documenters talk with the core developers a good deal.

Examining the star trails in Figure 9, we found that some core developers (shown in red) started their work on the project in the documents directory; then they moved down to the source code area. The documenters (shown in green), on the other hand, stayed within the documents area. The star trails also showed that many

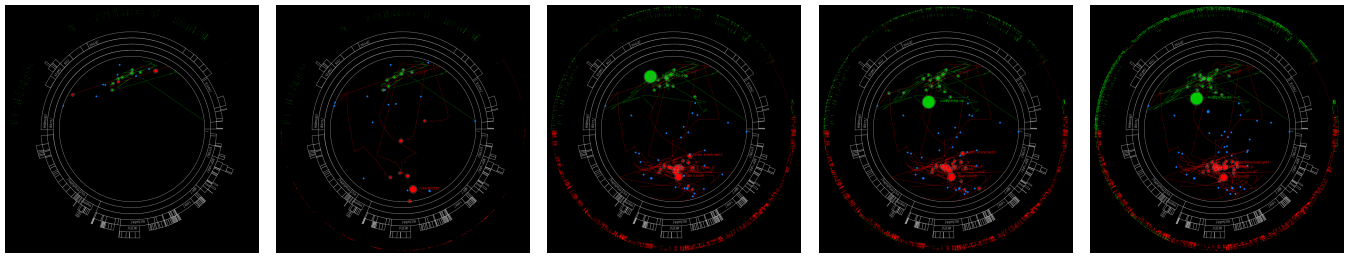


Figure 8: Evolution of the Apache project. Documenters are colored green and core developers are colored red. Other developers are colored blue. We can tell by the red trails that some core developers start as documenters but not vice versa.

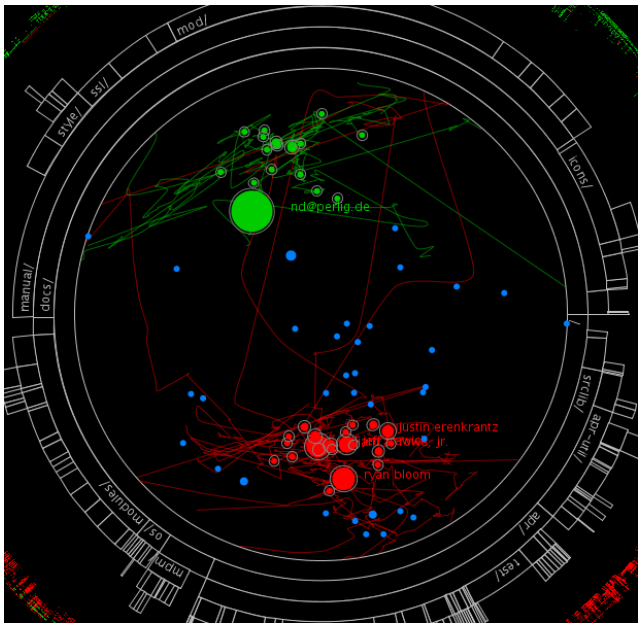


Figure 9: Star trails in the Apache project. While some core developers (in red) started at the top and moved down, the documenters (in green) stayed more or less in the same area.

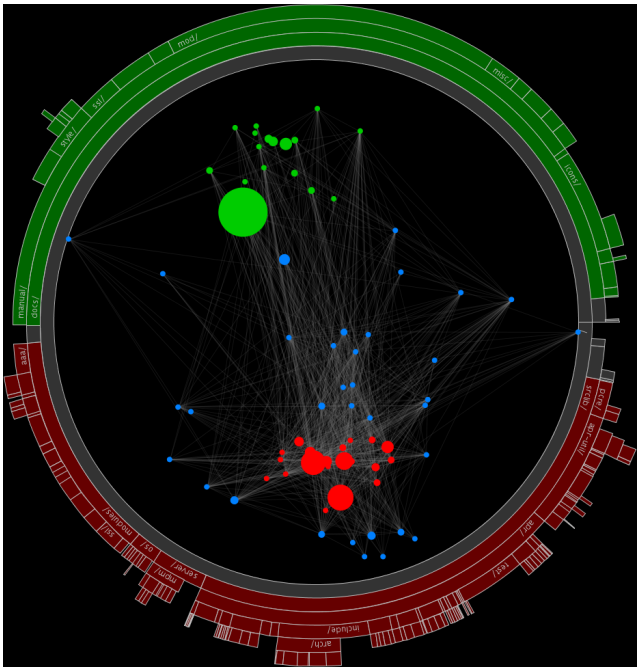


Figure 10: Overview of the Apache project at the latest timestep. The documentation directory and documenters have been colored green. The source code directory and core developers have been colored red.

core developers started work in the modules section of the source code before moving to the center. Therefore, we think that programming a module may be a good way to get "in" with the core developers.

In exploring the dataset, we have found that Ralf S. Engelschall has done a lot of development on the SSL module in May of 2001, but did not communicate with anyone else (Figure 12(b)). His work in May was also the only work he did on that particular area of the repository.

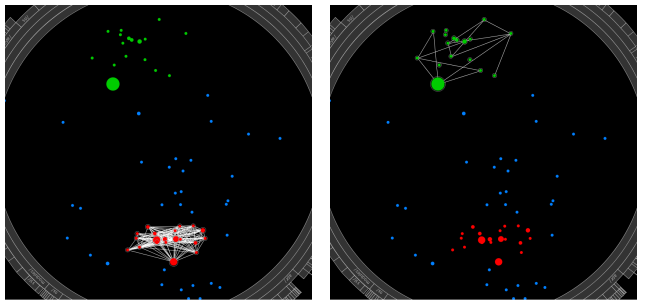
In previous sections we have stated that StarGate is useful for project novices, managers and software engineering researchers. We will now consider how each of them may use StarGate.

- Novices can see which developers are working in their area of interest, so as to contact them with queries. For example, one who wishes to work on a module can ask for help from developers in the lower-left quadrant.
- Project managers can, by selecting a star, tell how much work that person has been doing and in what area, all the way down to the file modification details. They may also make discoveries about communications, such as the two we made above.
- Software engineering researchers will find the animation and star trails helpful in visualizing the evolving project and developers' tendencies. They can see the amount of repository changes grow over time and correlate them with the communications network.

5.1 Comparison with PostgreSQL

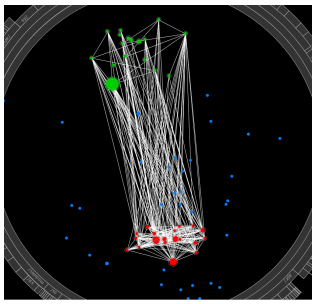
Our second software dataset is of the PostgreSQL project. There are 28 developers working with 4108 files in 277 directories. We can see in Figure 13 that there is no clear distinction between documenter and core developer stars, as the only cluster exists in the center. By selecting people who have worked on the `/docs/` directory, then comparing with the people who have worked on the `/src/` directory, we confirm that most people work a good deal on both. This is perhaps due to the smaller size of the development team and the esoteric knowledge needed to document a database management system.

We also find that there is a fair amount of communication occurring between developers on the "outside" of the core group. In Figure 14, we have selected people (colored yellow) close to the edge of the center circle, away from the core group.



(a) Core developers' dense network (bottom group, in red)

(b) Documenters' sparse network (top group, in green)



(c) Both groups

Figure 11: Clusters of Apache contributors and their communications networks. (a) The core developers form a dense communications network. (b) The documenters form a sparse network. (c) The documenters talk to the core developers, but not amonst themselves.

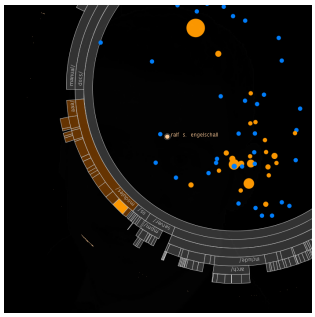
6 Discussion and Future Work

As described in Section 4.2, the positions of the stars is determined by the order of the Gate rings. The hierarchy is set by the repository, but sibling directories may be ordered in any way. That poses the question: Will a bad ordering of sibling directories in the Gate lead to bad star positions? It is theoretically possible to determine the best ordering of directories in order to achieve the best possible distribution of stars. However, there are some problems with this: 1) The meaning of the "best" distribution is in question. Is the goal to have the most separation of stars, or the tightest clustering? 2) Is it worth the time it takes to run an energy-minimizing-type algorithm on the directories? The case study in Section 5 seems to suggest that the stars are generally placed in positions adequate for making visual inferences. 3) By changing the ordering of directories, a programmer familiar with the structure may become confused.

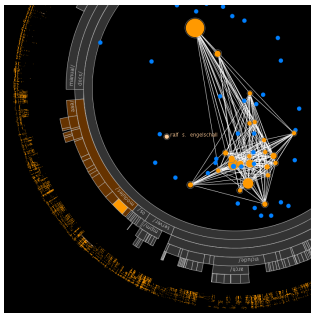
The StarGate design works well as an overview, but lacks the seamless integration of file details. How should the user transition from seeing the project overview to viewing details, such as the source code and file modification history? We may be able to add a fourth visual layer of information beyond the stardust, wherein the details of a selected file or files are displayed. The information within the layer cannot, however, precisely convey 100% of the data in the file modification events, since a good deal of text is needed. Therefore, a visual abstraction of the file history could be possible.

The cluttered graph labelling problem is the bane of many information visualizations and we are no exception. The solution offered by Wong et al. in [21], which is wrapping the labels around the node, is compelling. However, it does not solve the problem of labelling stars that are extremely close to each other.

For the communications network representation, though the user has the ability to interactively select the point in time and even animate through time steps, it only shows one point in time in the network. What we would like is to have a representation which



(a) The activity of Ralf S. Engelschall



(b) All SSL developers

Figure 12: Comparison of Ralf S. Engelschall (left) to other developers who worked on the SSL module (right). Note that Ralf did not communicate using the mailing list.

presents the user with a sense of the history of communication between developers. Time-varying network representations have been discussed in previous work like [4], [6], [7] and [16]. However, none of these representations are suitable for integration into StarGate.

Ultimately, we would like to use more data from the repository in our application; especially the source code itself. This should lead to a robust application for practical use in development. The goal is for developers to use the improved StarGate on a daily basis, complementing or even replacing command-line systems like CVS and Subversion.

One aspect of software repositories we have not addressed is the changes in directory structure over time. We have chosen instead to show the most up-to-date version of the repository. This is because, as the directories move, so too would the stars, leading to a misinterpretation of the star trails during a time-lapse animation. We have therefore dropped support for directory structure changes in favor of the more important developer evolution.

7 Conclusion

We have presented StarGate: a system for visualizing software development. StarGate is a novel way to visualize a software repository because it focuses on the authors rather than the source code. It provides an overview of a software project for developers, managers and researchers. We have presented a case study of our system in use, visualizing the Apache web server project and comparing it to the PostgreSQL project. Our system was able to discover development patterns, such as core developers starting out as documenters and periodic, longitudinal modifications of file groups.

A visual paradigm like StarGate's, with a network of actors surrounded by the artefacts they work on, is not limited to software engineering applications. One could visualize a collaboration of music artists in the center of their song catalogs. Or participants in an online forum, organized by its hierarchy, responding to one another in the center. The possiblities for visualizing complex network and hierarchical data using StarGate are compelling.

Acknowledgements

This work was sponsored in part by the U.S. National Science Foundation and the U.S. Department of Energy's SciDAC program. The authors wish to thank Premkumar Devanbu, Christian Bird and Alex Gourley for providing the software project datasets and their valuable feedback.

References

[1] Viewvc. http://www.viewvc.org/.

[2] Visual code navigator, http://www.win.tue.nl/ lvoinea/vcn.html.

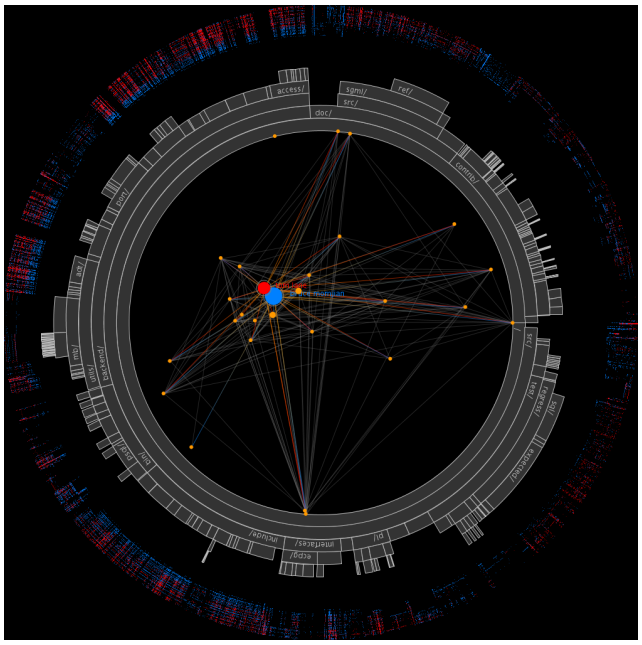


Figure 13: The PostgreSQL project. The current timestep is Feb. 6, 2006. The top two contributors (blue and red) are selected. There is no clear distinction between documenters and core developers.

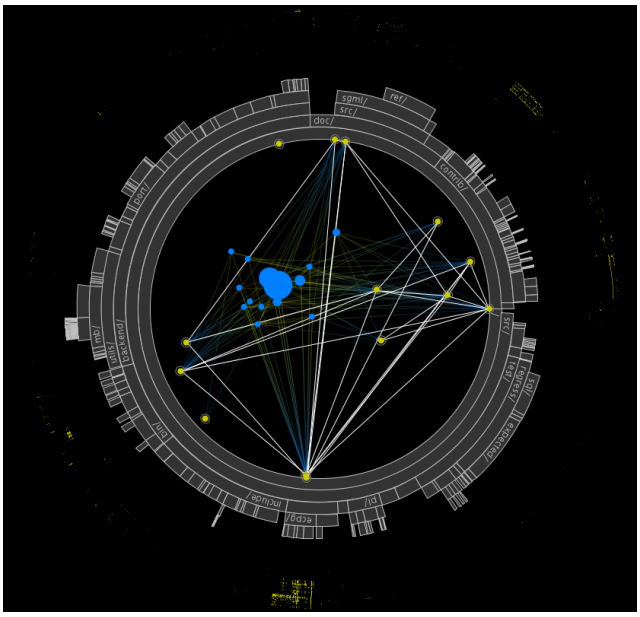


Figure 14: PostgreSQL. Selecting the developers on the periphery (i.e. not the core), we see that there is a fair amount of communication happening between them.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143. ACM Press, 2006.

[4] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.

[5] M. Burch, S. Diehl, and P. Weißgerber. Visual data mining in software archives. In *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, pages 37–46. ACM Press, 2005.

[6] C. Chen and S. Morris. Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. *InfoVis*, 00:9, 2003.

[7] E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S. K. Card. Visualizing the evolution of web ecologies. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 400–407. ACM Press/Addison-Wesley Publishing Co., 1998.

[8] M. Chuah. Dynamic aggregation with circular visual designs. In *Proceedings IEEE Symposium on Information Visualization 1998*, pages 35–43, 1998.

[9] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler. A system for graph-based visualization of the evolution of software. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, pages 77–ff. ACM Press, 2003.

[10] S. G. Eick, T. L. Graves, A. F. Karr, A. Mockus, and P. Schuster. Visualizing software changes. *IEEE Trans. Softw. Eng.*, 28(4):396–412, 2002.

[11] S. G. Eick, J. L. Steffen, and J. Eric E. Sumner. Seesoft-a tool for visualizing line oriented software statistics. *IEEE Trans. Softw. Eng.*, 18(11):957–968, 1992.

[12] J. Froehlich and P. Dourish. Unifying artifacts and activities in a visual tool for distributed software development teams. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 387–396. IEEE Computer Society, 2004.

[13] B. Kerr, L.-T. Cheng, and T. Sweeney. Growing bloom: design of a visualization of project evolution. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 93–98. ACM Press, 2006.

[14] Y. Livnat, J. Agutter, S. Moon, and S. Foresti. Visual correlation for situational awareness. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 13. IEEE Computer Society, 2005.

[15] K. Misue. Drawing bipartite graphs as anchored maps. In *APVIS '06: Proceedings of the Asia Pacific symposium on Information visualisation*, pages 169–177. Australian Computer Society, Inc., 2006.

[16] M. Ogawa, K.-L. Ma, P. Devanbu, C. Bird, and A. Gourley. Visualizing social interaction in open source software projects. In *APVIS'07: Proceeding of the 2007 Asia-Pacific Symposium on Visualisation*, pages 25–32. IEEE Computer Society, 2007.

[17] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, page 57. IEEE Computer Society, 2000.

[18] M.-A. D. Storey, D. Čubranić, and D. M. German. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, pages 193–202. ACM Press, 2005.

[19] F. B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582. ACM Press, 2004.

[20] L. Voinea, A. Telea, and J. J. van Wijk. Cvsscan: visualization of code evolution. In *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, pages 47–56. ACM Press, 2005.

[21] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote, and J. Thomas. Dynamic visualization of graphs with extended labels. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 10. IEEE Computer Society, 2005.

[22] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 77. IEEE Computer Society, 2002.