筑波大学大学院博士課程

システム情報工学研究科修士論文

大規模活動情報の視覚的分析支援ツールの開発

矢崎 聖也

(コンピュータサイエンス専攻)

指導教員　三末 和男

2012 年 3 月

概要

本論文では、大量の活動データを対比やドリルダウンしつつ分析するためのツール Series at a Glance (SaaG) について述べる。活動情報の持つデータや分析の切り口は対象データや分析者によって異なるが、SaaG によって活動情報を視覚的に一望しながらも多角的に分析することが可能である。これを実現するために、我々は活動情報を並べ、かつ個々の時間的変化を表すための視覚的表現を実装することで俯瞰や対比を実現した。その上でズーミングやフィルタリングといった操作を実装し、ドリルダウンを伴う分析を支援した。また得られた知見を記録し知見の蓄積を支援する機能の開発も行った。SaaG を用いて実際のプロジェクトのデータを分析し、本ツールの有用性を示した。

目 次

図目次

第1章　はじめに

近年、OSS (Open Source Software) の開発プロジェクトなどの知的活動組織において、bugzilla や redmine を初めとしたプロジェクト管理ツールを用いてタスクの情報を管理、記録することが広く行われるようになってきている。これらの情報はチケットと呼ばれ、チケット 1 つはタスク 1 つの経過の記録である。プロジェクトのメンバーはタスクの発生・進捗・終了を、チケットの作成・更新・終了として記録する。

ノウハウを得たり自分や同僚達の過去や現状を知るためには、今までのチケットを俯瞰や分析することによる、データに基づいた観察が有用である。例えばプロジェクト内の仕事が平穏無事に進んでいるか紆余屈折を経ているかの様子やスピード感、タスクの配分の現状などの観察結果 (知見) は、自他の行動を振り返ったり今後の行動を考える役に立つ。今までの自分のタスクの進捗データに基づいた客観視を行っている ”The errors of TeX” [29] や Mockus らの第三者視点での分析 [17] など古くから近年に至るまでタスクの経過を見ることによる活動の分析は行われていることである。

しかし、プロジェクト管理ツール上でこういった知見を得るためには数千、数万ものチケットを見て回る必要があり、多大な時間と労力を要する。また既存の活動分析支援ツールは、特定の指標に基づいた分析を支援するツールであり、予測のつかない知見を探索する用途には適用しづらい。

我々は、多角的な分析を実現するために、数万件のチケットの時間的変遷の俯瞰、フィルタリング、対比を可能とするツール SaaG (Series at a Glance, 図 1) を開発した。形状で時間変化を表す視覚的表現をカテゴリごとに固めて配置することで、時間的変遷の俯瞰や対比を視覚的に可能とした。この視覚的表現を操作することや、視覚表現の一部への拡大 (ドリルダウン) やフィルタリングをすることによって、特徴的なチケットや活動の法則性を突き止めることが可能である。

1.1　この論文の構成

本論文ではまずチケットデータというものについて説明、例示、一般化、データの出元の説明をし、それらに対する分析において必要とされることが何かを考察する (2. チケットデータ)。その上で、活動の分析という観点と、チケットデータの一般形である多次元時系列のデータを視覚的に見せるという観点それぞれでの既存の取り組みを示す (3. 関連研究)。そして、本研究において提案・実装したチケットデータの視覚的表現について説明する (4. SaaG の視覚的表現)。その上で、この視覚的表現を用いて、提案・実装するツールである SaaG の各機

能やその機能に至るまでの背景や過程を述べる (5. SaaG の開発)。また、このようにして実装された SaaG を実データに適用することで知見が得られることをケーススタディによって示す (6. ケーススタディ)。そして、最後にこのような視覚的なチケットデータの俯瞰・分析ツールにおける今後の課題を述べる (7. 今後の課題)。

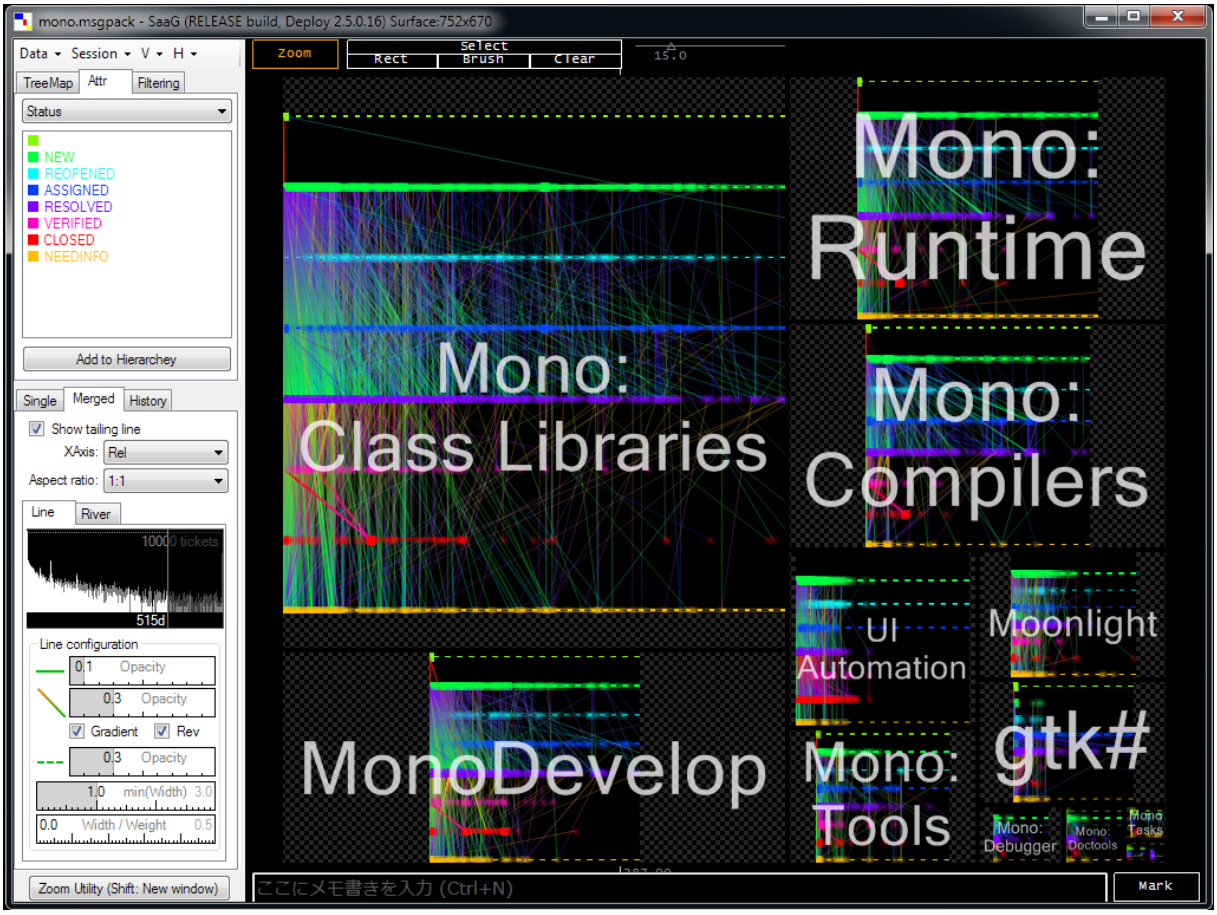

図 1.1: SaaG (Series at a Glance) の画面全体のスクリーンショット。19734 件のチケットデータの量と時間変化を、カテゴリごとにまとめつつ提示している状態の図である。

第2章　チケットデータ

2.1　チケットデータとは

本研究では活動情報の一種であり、多くの知的生産プロジェクトにおいて用いられているチケットのデータを分析作業の対象とする。

2.1.1　チケットの概要

チケットはタスク一つ一つと対応し、そのタスクの現状や履歴の記録である。各チケットは状態 (Status) や担当者 (Assigned to)、サブプロジェクト (Product) などの属性を持ち、チケットに対応するタスクの進捗がある都度にプロジェクト管理ツールを介して属性値が更新される。例えば Status は「新規」「担当者割り当て済み」「終了」といった値を取り、タスクの状態変化に応じて値が変わる。

また、属性や属性のとりうる値、それらの意味はプロジェクトや組織によってさまざまである。

2.1.2　チケットの例

ソフトウエア開発プロジェクトにおけるチケットの実例を一つ示す[1]。このチケットは、表 2.1 に示す属性値を初期状態に持っており、表 2.2 に示す更新履歴を持っている。この例では、バグが担当者に割り当てられてから、解決と再開を繰り返しており、その過程でいくつかの属性値が変更されている。

表 2.1: 属性の初期値の例

属性名	属性値
Assignee	mono-bugs@lists.ximian.com
Status	NEW
Resolution	(なし)
Severity	Critical
Priority	P5 - None
Product	Mono: Runtime
Found in Version	2.0.x

[1]例は https://bugzilla.novell.com/show_bug.cgi?id=489019 の一部を引用

表 2.2: 変更履歴の例 (部分)

イベント日時	属性名	新しい属性値
2009/03/26 06:14:04	Assignee	gonzalo@novell.com
	Severity	Normal
2009/03/27 20:08:07	Status	RESOLVED
	Resolution	FIXED
2009/03/30 08:09:39	Status	REOPENED
	Resolution	(なし)
08:10:20	Status	NEEDINFO
2009/03/30 17:08:53	Status	RESOLVED
	Resolution	FIXED
2009/04/01 10:05:53	Status	REOPENED
	Resolution	FIXED
10:06:43	Status	NEEDINFO
15:15:05	Status	REOPENED
2009/04/01 15:15:38	Proprity	P2 - High
	Severity	Major

2.1.3 チケットデータの一般化

本論文中で取り上げているデータ例はバグ・タスク管理ツールのチケットデータであるが、本研究では進捗を表すデータ一般をチケットとして整理しており、それを対象としている。

たとえば企業が営業によって獲得した”案件”であれば、その案件のフェーズ、予算の規模、取引先、担当部署や部門・担当者などを属性として持つチケットとして整理することで、案件の進捗データを本研究の対象とするチケットデータに帰着させることができるであろう。

2.1.4 チケットデータの収集

本研究に用いるチケットデータは、mono project [2] や redmine [3] などのプロジェクトで用いられている、bugzilla [4] や といったプロジェクト管理ツールの持つチケットデータを収集したものを用いた。なお、redmine はプロジェクト管理ツールである redmine によって redmine の開発プロジェクト自体を管理している。

redmine や bugzilla は、チケット一つの現在の状態を表 2.1 のように属性ごとの属性値として持ち、かつチケットの更新によって属性値がどう変わってきたのかを表 2.2 のようにして持っている。チケットそれぞれについて上記の情報を収集し、この二つをあわせることでチケットの作成時から現時点までのデータを得ることが出来る。

管理ツールである bugzilla や redmine は web API による機械可読なデータ出力機能を備えているが、上記のプロジェクトではその機能が一般向けに有効化されていない問題や、一部の属性値しかデータを得られないという API 仕様上の問題があった。そこで、我々の研究で

[2] http://www.mono-project.com/
[3] http://www.redmine.org/
[4] http://www.bugzilla.org/

は管理ツールの人間向けの HTML ページをクローリングし、得られた HTML を解釈し、チケットデータを抽出したものをデータとして用いた。

なお、redmine や bugzilla に限らず、任意数の属性を持つバージョンの連なりに落とせるのであれば、他のシステムからのデータを用いることが考えられる。組織としての手順のルールを持つ企業活動を例に取れば、一つ一つの営業案件もチケットとして扱えうる。たとえば営業案件であれば、取引先や担当者名、交渉の段階 (提案段階、受注済み、引き渡し済み、等々) 交渉している取引物の額面や量、といったものを属性として表すことで、交渉内容の変遷 (たとえば提示する額が下がった) や交渉の進捗状態といったことをチケットデータとして扱える。

2.2 チケットデータの分析

どのような切り口から分析するべきかを事前に知らない状況において、プロジェクトにおける人の活動を観察するということがある。たとえば、内情に詳しくないプロジェクトについて調査を行うとき、多くのプロジェクトからプラクティスを発見しようとするとき、普段とは違う角度から自分たちを振り返りたいとき、といった場合を想定する。このような場合には、「担当する人によって切り分け、抱えている作業量を見る」「時期によって切り分け、作業量とその作業の重要度を見る」「取り組んでいる作業の分類によって切り分け、各作業が手戻りや停滞あるかどうか見る」といった様々な切り口で見ることが考えられるが、分析前にどの切り口から見るのかを定めることは難しい。そこで、様々な切り口を試行錯誤しつつ、「xx という分類の作業では手戻りが多い」「yy 月に決まって重要度の低い仕事ばかりしている」といった具体的な知見を得て、さらにはどのような切り口がそのプロジェクトの分析に有用なのかを知る、ということが必要となる。

このように本研究は、大量の活動情報に対して、分析者が具体的な知見のみならず、興味深い切り口を見つけること自体の支援を目的とする。

そこで本研究では、タスクの認知や割り当て、検収や差し戻しといった人間の行動や意思表明の記録であるチケットデータを対象とする。最終成果物の変化などではなく、人間がいつどのような行動や意思表明をしたのかという情報を読み取れることによって、人間の活動に対する知見を得るためである。

また、本研究では実プロジェクトに存在する、最大数万件に及ぶスケールのチケットデータを対象とする。このスケールのデータの全貌を把握した上で、ユーザーの興味に応じてデータのサブセットを見ることも可能にすることを意図する。データの全貌の把握はそもそもどのサブセットを見るのか、全体像だけを見るのかの決心するためである。また、どのような基準で分類した場合のサブセットに着目するかを判断するための、分割作業やサブセット同士の対比も必要であると考える。俯瞰と分割・対比がある上で、必要に応じてサブセットのみの表示 (ズーミングやフィルタリング) が必要となる。

加えて、チケットの持つ属性や値はプロジェクト毎に異なっており、かつ分析者はどのような知見が存在するかを事前に確信しているとは限らない。そのため、チケットデータの分析ツールは特定の属性や尺度に依存せず、かつさまざまな切り口からの分析を支援する必要が

ある。また、分析過程で得られた知見を別の切り口からも探ることや、特徴のあるチケット (群) を詳細に調べること (ドリルダウン) も可能であることが望ましい。加えて、「活動」は時間的なものであるため、チケットの時間変化や時間長さを見せることも重要である。

第3章　関連研究

SaaG は、ソフトウエア開発プロジェクトの活動情報の俯瞰や分析を大量の時間変化するデータの可視化によって支援する。本研究の関連研究等をここに挙げる。

3.1　ソフトウエア開発プロジェクトの活動分析

ソフトウエアの開発プロジェクトの活動に関する情報を可視化し、そこから知見を得ることを支援するための研究は多く行われてきている [25]。例えば、SeeSoft [3] や Augur [12] などは、ソフトウエアの開発状況の変化を閲覧するために、レポジトリから得られるソースコードに対する変更履歴を見ることで開発作業の経過を分析するためのものであり、大量のファイルに対する何回もの変化の俯瞰や、ディレクトリ構造を同時に見せディレクトリ間の対比などが可能である点に強みがある。

しかし、タスクの認知や割り当て、検収といった人間の行動や意思表明の記録であるチケットデータの分析を目的としている本研究では成果物の変化だけを見る手法は適用しがたい。

MDSViews [11] は、CVS(レポジトリ) と Bugzilla(プロジェクト管理ツール) から得られる情報に多次元尺度構成法と呼ばれる手法を適用することで、プロジェクト内の要素同士の関連性を網図によって視覚的に表現するものであり、プロジェクト内の要素の関係性を見たり、影響の大きな要素といったものを見いだす点に有用性がある。

だが、大量の活動情報に対して、分析者が興味深い切り口を見つけて知見を得ることを目的する本研究には、データを簡略化したり活動の情報の一側面のみを見せたりするアプローチは適さない。

Social Health Overview (SHO) [9] は、Bugzilla から得られた個々のチケットとその属性を点と色として表現として見ることで、プロジェクトの活動状態の健全性を評価するためのツールであり、プロジェクトの健全性評価という目的に特化し、ぱっと見によって全体や部分ごとの健全性が見える点に強みがある。SHO は、プロジェクト管理ツールから得られる最大 3500 件程度のチケットの情報を視覚的に表現することで、全体の活動の様子を視覚的に表現する点と、表現に対して操作を行うことで多角的に分析する点で本研究と類似している。

Software evolution storylines [20] や code_swarm [19] は開発プロジェクトのコミュニティの変遷を見ることを目的とし、プロジェクトに対する個々人の貢献度や貢献する部分の時間変化を出しており、プロジェクトの活動を視覚的に見ることを広く一般に普及させる点に力を置いている。人の行動の時間変化を見るという点では類似しているが、あくまで人の出入りを見ることに特化している点や数百人程度を対象とする点が、複数の属性を持つ数万件の進

捗データの切り口を模索する本研究とは趣旨を異にする。

しかし、特定の属性や尺度を前提としている点や、やはり分析の切り口を限定している点、それに画面の縦や横解像度よりも多くのチケットを対象としていない点に発展の余地がある。

3.2 時系列データや多次元データの視覚的表現を用いた分析ツール

時間変化するデータ (時系列データ) や多次元の値を持つデータが大量にある時に、それらをまとめて可視化することによって大量のデータに対する理解を助ける研究も多く行われてきており、Aigner らの調査 [1] において時系列データのどの側面を表現するのかについて整理が行われている。また、Extreme Visualization [23] では、単なる表でない方法でより視覚的に大量のデータを表現することによる分析支援についての研究が示されている。

複数の属性値を持つ大量のデータについての傾向を提示する研究としては、上記以外にも Chen らによる dendrogram を用いた研究 [7] や Muelder らによる研究 [18]、ActiviTree [27]、WireVis [6] なども挙げられ、これらは複数属性値を持つデータの俯瞰をサポートしているが時間変化も視覚的に表現するものではない。

ガン検査の結果についての大量のデータをクラスタリングして並べて表示している Chen らの研究 [7] は、大量の多次元データを並べて、かつクラスタリングによってデータ群同士の対比を可能とする点は本研究と類似しているが、データの時間位置や長さというものを対象データが持つ本研究とは前提が異なる。

クラスタ上でやりとりされた大量の MPI メッセージの傾向を可視化している Muelder らの研究 [18] は、大量のデータの属性値のみならず時間位置や長さも表現し、さらにデータの並びを変えることで傾向を見いだす点が似ている。しかし、データの視覚的表現同士が重なってしまいうる点が分析や操作上の弱点である。

また一つのエンティティ(チケット、人など) が時間変化し、そのエンティティ多数を提示する研究としては、Mielog [26]、StackedGraph [5]、稲川らの研究 [30] などがも挙げられるが、本研究では、途中で変化するデータの表現と、任意の属性による対比の操作が必要である。

このように、時間変化するデータの可視化についてデータの固まり、データの時間変化の双方を表現する視覚的表現の上に対比やドリルダウンの操作を実現する点が本研究の特色である。

第4章　SaaG の視覚的表現

数万件のチケットの俯瞰と対比のための視覚的表現を設計・実装した。この視覚的表現は、チケットの時間変化をガントチャートや折れ線状の形状として微小な領域内で表す表現、あるいはチケットの集合の時間変化を折れ線の重なりや積み上げグラフ状に表す表現を、チケットやチケットの集合ごとに割り当てた矩形領域に埋め込むものである。これによって、大量のチケットの時間変化や量を限られた面積の画面にて表し、またカテゴリ間の対比を可能とする。

今まで困難であったチケット群の俯瞰と対比を可能にする点や、一部へのズーミングが可能である点が SaaG の視覚的表現の特長である。これによって、まず全体を見て、そこから興味深い部分に着目するという分析の流れを可能とする。

4.1　個々のチケットの属性値の時間変化の表現

SaaG の視覚的表現では、まずチケット一つの時間変化を形状によって表す。そのために、Gantt 表現と Line 表現の二種類の表現を設計した。

4.1.1　Gantt 表現

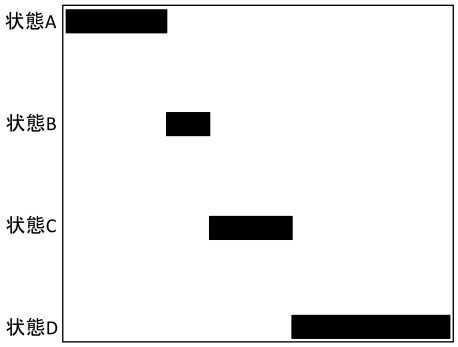

図 4.1: 一般的なガントチャート。計画されたタスクの開始から終了予定までの時間を横軸、各時間の取る状態を縦軸とし、ある状態をある時間に取っていることを塗りつぶしによって表現する。

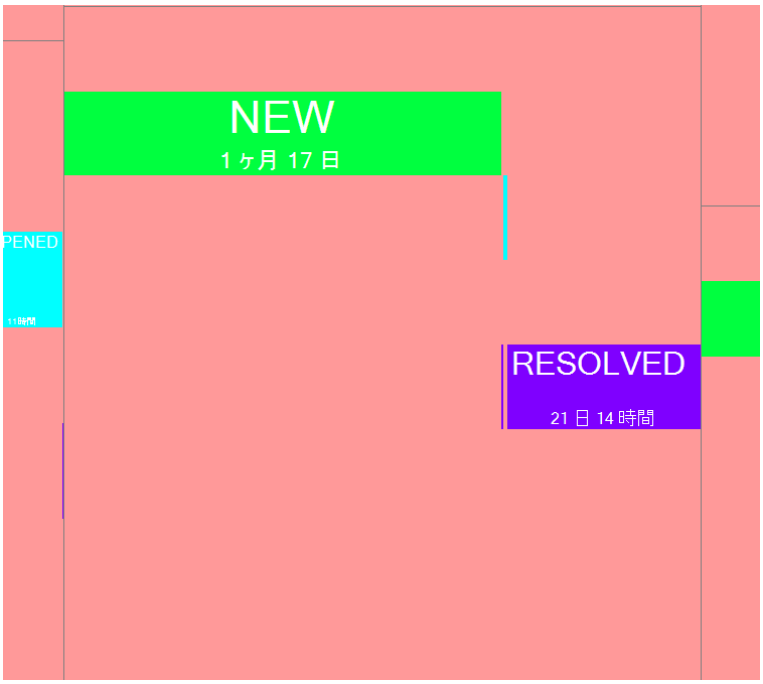

図 4.2: SaaG の Gantt 表現によって一つのチケットを表現した図。縦軸は Status 属性値、横軸は開始からの経過時間

Gantt 表現は、進捗状況の表現に古くから用いられているガントチャート [13] を基本とした表現である (図 4.1 に例示)。これは、ユーザーによって選択された属性の各値を縦軸、チケット開始から最終状態までの時間を横軸に取り、チケットがある値を取っている期間を横向きの棒によって表す表現である (図 4.2)。横軸については左端が必ずチケットの開始、右端がチケットの最終の状態になった時点に対応する。

ビューの設計としては、縦横軸は上記の通りであり、また Gantt の背景色は (後述する理由で) チケットの最終属性値に対応する色相の色である。そして、チケットはある時点ではどれか一つの属性値を取っていることを用いて、その属性値の縦軸の範囲と、その時点の横軸の領域を、属性値に対応する色で塗りつぶす。これによって、縦方向に重なり無く、かつ左端 (チケット始端) から右端 (終端) まで隙間のない棒状の領域の連なりが得られる。これと背景色とが Gantt 表現である。また、SaaG では、図の読み取りの補助のために Gannt 表現の棒の中に、属性値のラベルと、棒の示す時間長さのラベルを描き込んである。これは、一般的なガントチャートでは棒の名前と時間長さがラベルで描かれていることを踏まえて親しみやすさのために設計しているのと、図の読み取りをしやすくするという二つの意図がある。ビュー全体では最大数万ものチケットを 1 画面に描画するためこれらのラベルはつぶれてしまいノイズになるため描画しないが、ビューの一部へと拡大しフォントのサイズが閾値以上になった場合に描画するように設計した。

図 4.3: 時間長さが短く、横方向がつぶれてしまう棒を広げた Gantt 表現の図。 図 4.2 と比べて Reopened (水色) の棒の存在をはっきりと認識できる。

この図を見ることでチケットの属性値のたどってきた変化を読み取ることができる。また図では属性値に対応した色を付けているが、形状のみからも属性値と時間を読み取ることが出来るため、大まかな形状さえ分かれば変化の流れを掴める。ゲシュタルト要因により、形状の類似性の認知や、珍しい形状の発見は人間が直感的に得意とすることであるため、後述するように大きさの限られた Gantt 表現を多数のチケット分並べる際に、重要となる性質である。

なお、図の背景色がチケットの最終状態の色相の色であるのは、選択属性の値が最後の値になって以降にチケットの更新がない場合に、最後の状態を表す棒の幅が 0 になってしまい見えなくなるという現象が多々発生するという問題に対処するためである。

ガントチャートによる表現の弱点として、チケットがある属性値をごく短時間だけ取った場合の表現がつぶれてしまいやすい事があげられる。チケットの場合、数日や数ヶ月におよぶ待ち状態から、作業中における数時間単位での状況の変化、さらには訂正や追記による秒単位の更新に至るまで、様々なタイムスケールでの状態変化がある。このため、たとえば新規の状態で 3 時間、作業中の状態で 15 分間であった場合、作業中の幅がチケット全体の中で小さくなってしまい、見落としてしまいやすくなる。これに対処するために、SaaG では横方向の幅が一定以上小さくなる棒は一定幅に必ず広げることをオプションで可能にした (図 4.3)。これによって、時間長さの短い状態の見落としを防ぐことを可能とした。ただし棒を横方向に広げてしまうことで、時間方向の読み取りの精度が低下する点や、一般的なガントチャートとしての理解のしやすさが損なわれてしまう点によるトレードオフがある。

4.1.2 Line 表現

Line 表現 (図 4.4) は、Gantt 表現と縦横軸を同じくしつつも、状態と状態の間を線分で結んだ図である。

Gantt 表現が縦方向に離散的な表現であるのに対して、Line 表現はチケット開始 (左端) から最終状態 (右端) まで一つの折れ線で滑らかに繋がっている。このことは、後述するように表現を多数のチケット分並べる際に、一つのチケットが一つの折れ線としてまとまって見えるという性質が得られるというメリットをもたらすことで、一つ一つの表現が小面積・低解像度になっても連続の前知覚的要因によってチケット一つ一つの固まりを認知しやすくする効果を意図している。

ビューの設計としては、縦横軸と背景色は、Gantt 表現と同じく属性値とチケット開始からの時間、最終属性値の色である。そして、チケット内の各ログごとに、そのログの時刻に対応する横位置、そのログ時点での属性値に対応する座標を制御点とする。その上で、最初のログから最後のログの制御点まで順に線分で結ぶ。線分の色は制御点の所での属性値に対応する色であり、線分両端の色が異なる場合は線形グラデーションによって彩色する。

折れ線として繋がっていることは、Gantt 表現で問題となった短時間での状態変化の読みやすさにも貢献する。短時間での状態変化は線分の急な傾きとして、Gantt 表現のように例外的な処理を行うことを必要とせずに、表現することが折れ線であることによって可能になっている。ただし、急な変化が連続して起こることで、線分と線分が重なってしまうという弱点

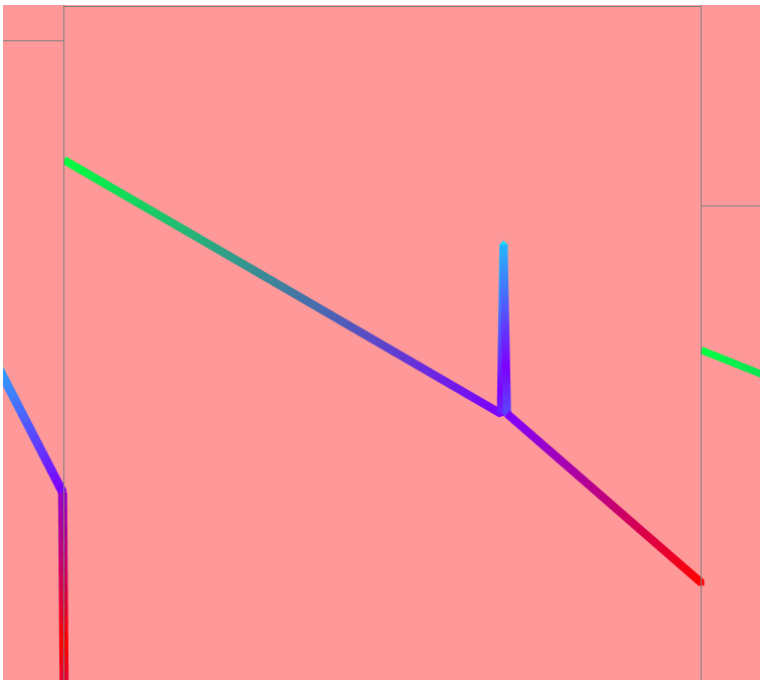

図 4.4: 図 4.2 を Line 表現に切り替えた図

がある。線分には描画上の太さがあるために、図 4.4 のように上向きと下向きの急峻な線分が隣接すると重なりが生じてしまい、読み取りの精度に支障をきたす。

Gantt 表現の場合、属性値ごとに上下の位置が分かれているために原理的に上記のような重なりの問題は起こらず、この点は Gantt と Line 双方に得意不得意のある点である。SaaG では、ユーザーが Gantt と Line のどちらも選べるようになっている。

なお、Line 表現の場合、最後の線分の向かう先を見ることで最終状態がわかるため、Gantt 表現と異なり形状から最終状態も見て取ることが出来る。SaaG では Gantt と随時切り替えて使える関係上、背景色を一貫して最終状態の表現に用いているが、Line 表現は背景色の存在が必須ではなく、このことは後述の Lines 表現の実現に貢献している。

4.2　複数のチケットの時間変化の視覚的表現

Gantt や Line 表現に加え、複数のチケットの時間変化をまとめて表すための Lines 表現と River 表現の二種類の表現も設計した。これらの表現は数百や数千個のチケットの時間変化の仕方を概観し、領域間で対比するためのものである。

4.2.1　Lines 表現

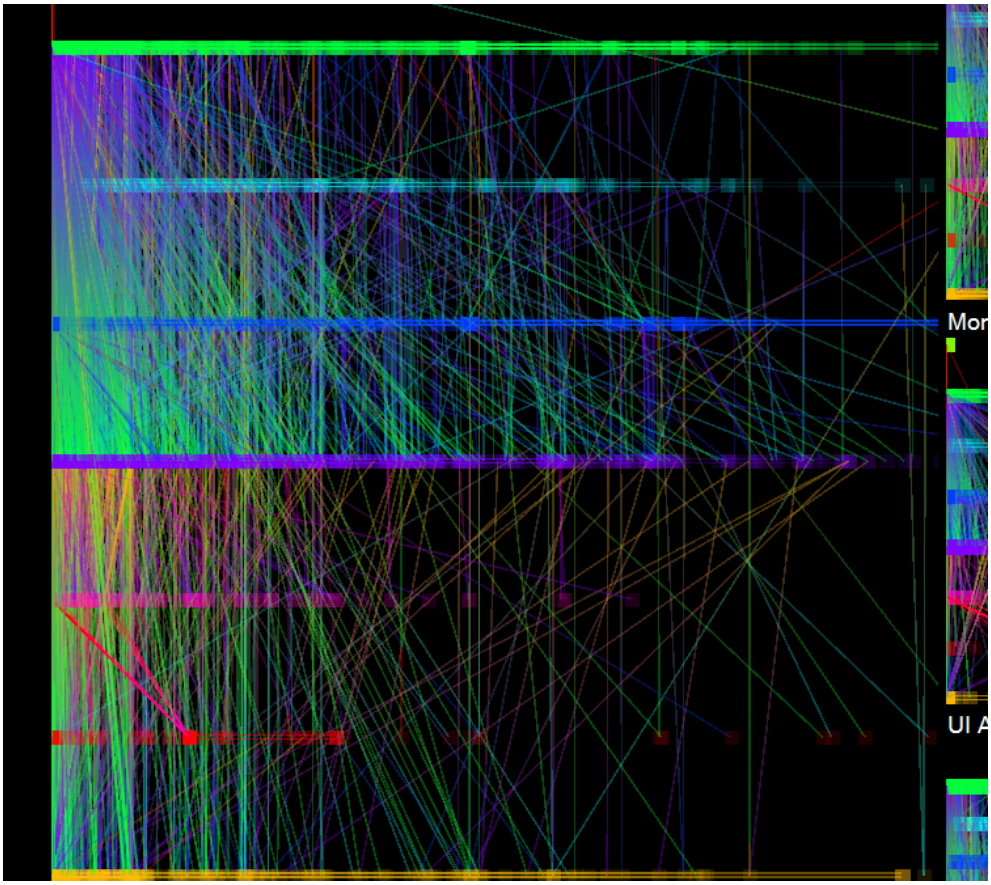

図 4.5: Lines 表現によって約 8000 チケットの変化を表現した図。縦軸は 図 4.4 などの図と同じく Status 属性。

Lines 表現 (図 4.5) は、属性値を縦軸、チケット開始からの経過時間を横軸にとり、チケット一つが折れ線一つとして表現されるものである。多数の Line 表現を半透明にして重ね合わせた図として読むことが出来る図になっており、Line 表現、ひいては Gantt 表現の延長線上の間隔で読める図を意図している。

Lines 表現の縦軸は Gantt や Line 表現と同じく属性値、横軸は左端がチケット開始、右端はチケット開始から一定の時間が経った時点を表す。右端のデフォルトはチケットデータの中で最長のチケットの長さである。Lines 表現の内容はチケット数だけの折れ線を半透明に重ねたものであり、個々の折れ線は Line 表現と同等である。ただし、後述するように折れ線を構成する線分のグラデーション方向が逆になっている。

また、データの取得時点において閉じていないチケットについては、描画領域の右端まで点線を延ばしている。これは、閉じられていないチケットはデータの取得時点においてタスクとして存在していることを表現するためである。閉じられていないチケットの折れ線がチケットの最終更新時点にて終わってしまうことで最終更新時点でタスクが減っているように見えてしまうことを防ぐために、このような設計としている。なお、破線と、同じ縦位置 (同じ属性値) の実線とが重なって識別しづらくなることを防ぐために、破線と実線は重ならないように縦位置をわずかにずらして描画している。

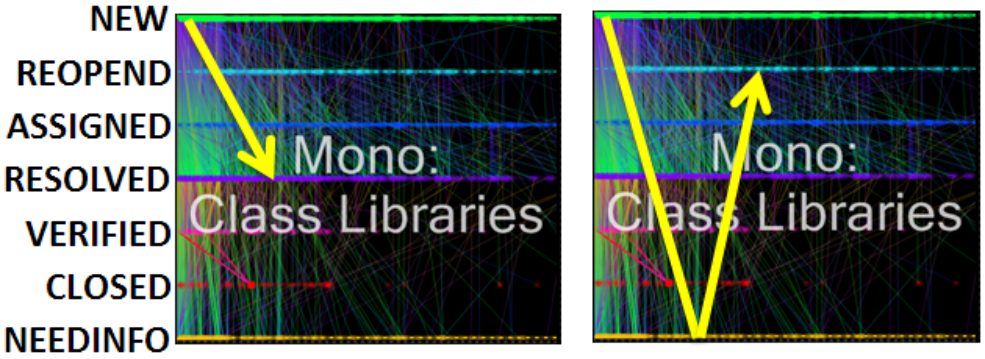

図 4.6: 図 4.5 からの読み取り例

Lines 表現によって、例えば素早く終了したチケットは右下がり (図 4.6 左) 、紆余屈折を経ているチケットは V や W 字形 (図 4.6 右) 、という風に形状によって時間的経過を読み取れる。

Lines 表現は多数のチケットを折れ線で重ねて描画するため、線が重なって混雑することがある。これに対処するために、線分に逆向きのグラデーションを施してある。このグラデーションは、線分の向かう先の属性に対応する色を線分の始点の色、線分の元の属性の色を終点の色とするグラデーションである。このグラデーションによって、同じ属性間の線分が視覚的に一まとまりに見える効果を得ている。かつ線分がどちらへ向かっているのか、どこから来たのか、を始点や終点を見るだけでも読み取れるようにしている。

グラデーションを施さなかった場合 (図 4.7)、たとえば NEW (緑) から RESOLVED (紫) へ向かっている線分の中に、NEW と RESOLVED の間にある ASSIGNED (青) へ向かっている線分が紛れているのか否か、といった読み取りが困難になる問題が起こる。グラデーションを順方向にした場合 (図 4.8) は読み取りに支障があるわけではない。しかし、逆方向にする (図 4.5) ことで、線分の始点を見るだけで向かっている先が、終点を見るだけで線分の元が

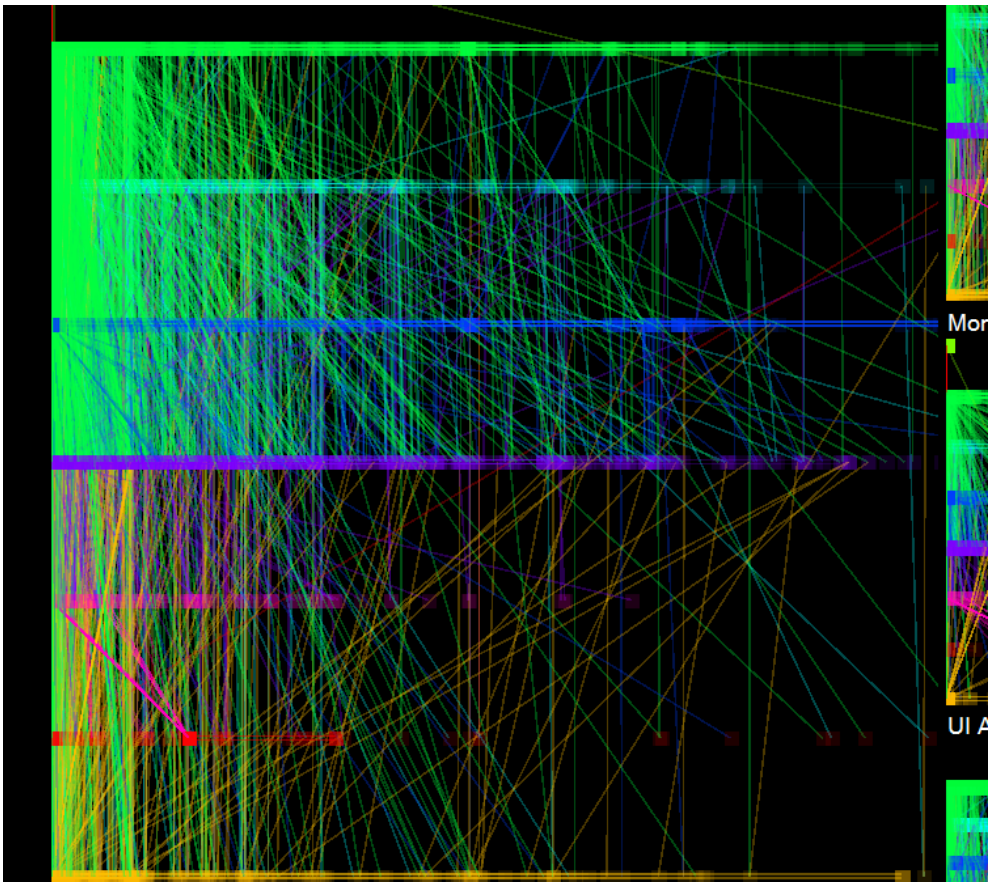

図 4.7: 図 4.5 のグラデーションを施さなかった場合の図

分かる。このことは、たとえば「初期に RESOLVED (紫) へ来ている大量のチケットは、どの値から変化しているのだろうか」といった読み取りを線分をたどるまでもなく高速に行える。こういった利点を得るために逆方向のグラデーションを施してある。

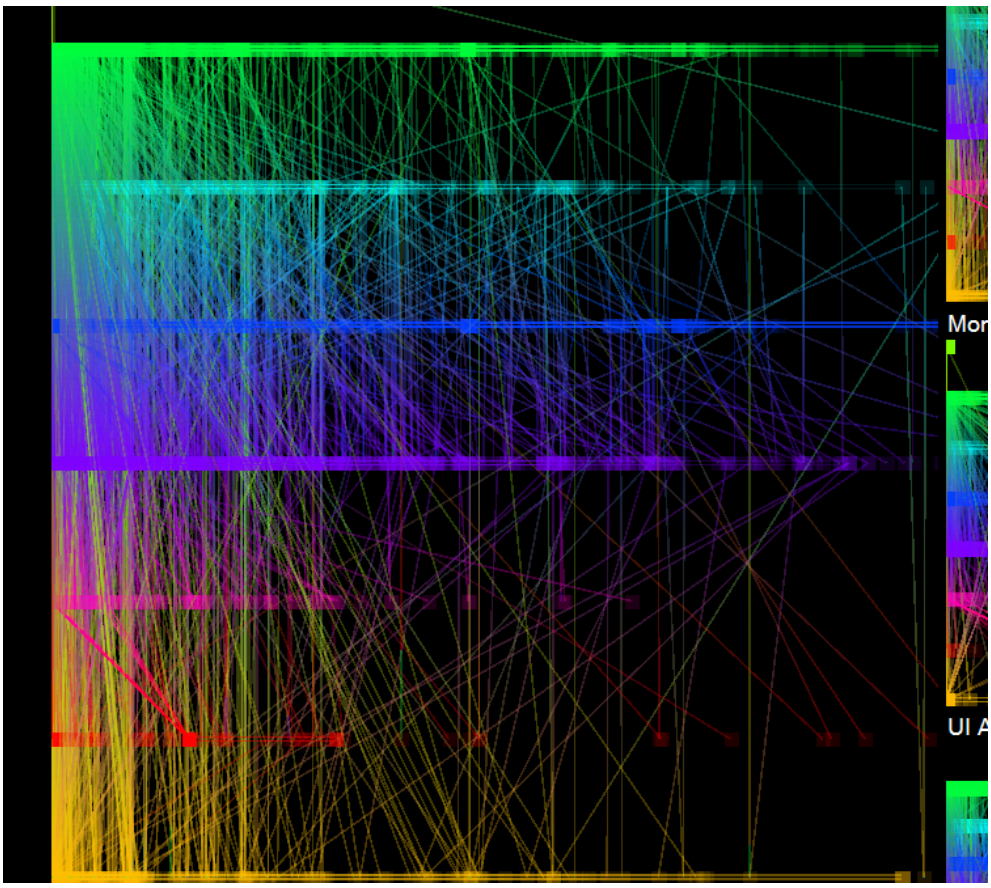

図 4.8: 図 4.5 のグラデーションを逆向きにしなかった (順方向にした) 場合の図

4.2.2 River 表現

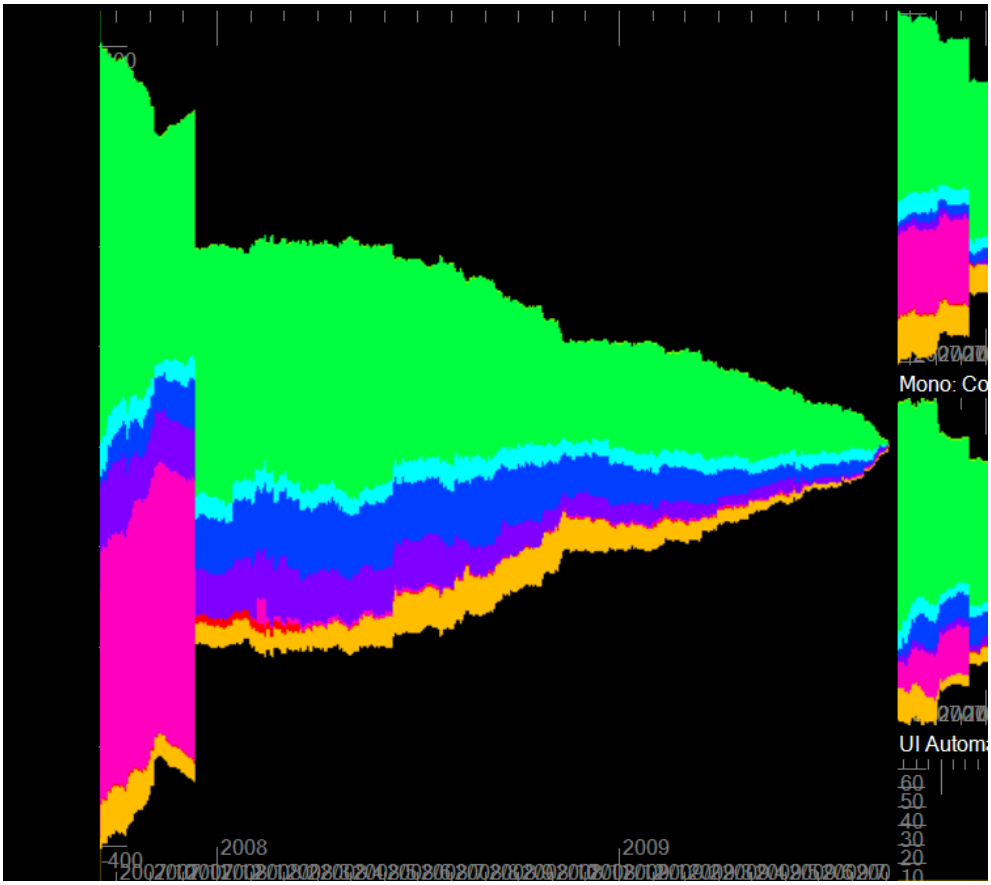

図 4.9: River 表現によって 図 4.5 と同じチケット群を表現した図

River 表現 (図 4.9) は、これまでに述べた視覚的表現ではチケット開始からの相対時間しか表現できない点を補うための表現である。ThemeRiver [16] の視覚的表現手法を元にした表現であり、横軸が日時、縦軸がチケット数になっている。

River 表現は縦軸が中央を 0 とするチケット数の軸であり、横軸は時刻 (年、月、日、時...) である。各時刻におけるチケット数が高さであり、属性値ごとに積み上げられた図になっている。

ThemeRiver の表現ではデータにおける時間方向の分解能が数十から百程度の刻みだけであり、時間と時間の間は滑らかな曲線で繋ぐことで補完している。それに対して本手法では、データの持つ時間の分解能が細かく、かつ短時間でのまとまった更新がしばしば起こることから、それを表現するために画面上の 1 px ごとにチケット数を計算し描画しており、曲線による補完は行っていない。これによって急峻な変化を表現することを可能としている。

4.2.3 Lines や River 表現の横軸

ここまでの説明では Lines 表現の横軸は Line 表現と同じくチケット開始からの時間、River 表現の横軸は絶対時間であった。しかし原理的には 2 種類の表現と時間軸の種類は直交していることを鑑み、SaaG では横軸を絶対日時にした Lines 表現 (図 4.10) や横軸を相対時間にした River 表現 (図 4.11) もオプションで可能としている。Lines 表現は線分の数 (属性値から属性値への変化の数) が、River 表現では River の縦幅 (ある時点での属性値ごとのチケット数) が読み取りやすく、その逆は困難であるというトレードオフがあるが、時間軸自体はどちらも取り得るものであることを示している。

図 4.10: Lines 表現の時間軸を絶対日時にした図

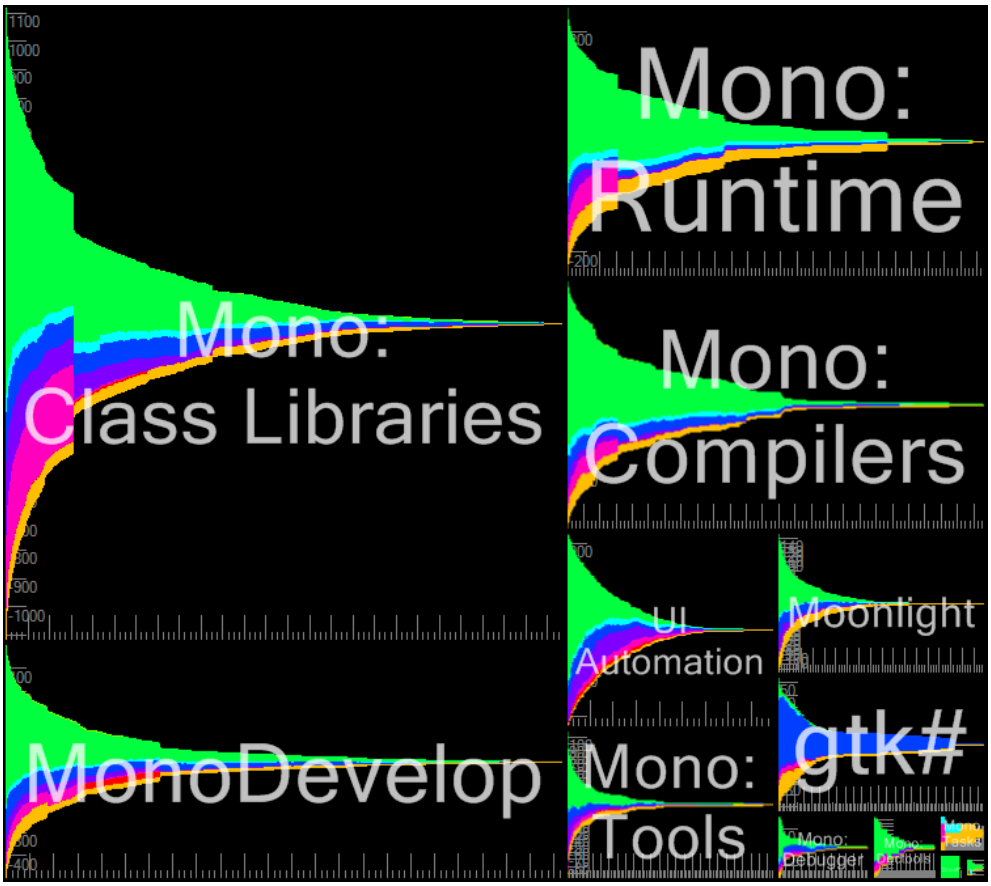

図 4.11: River 表現の時間軸を相対日時にした図

4.3 Treemap を用いた、チケットの量の表現と対比支援

SaaG では、チケットごとに矩形の画面領域を割り当て、領域内にこれまでに述べた Line、Gantt、Lines、River 表現のいずれかを埋め込む。また、個々のチケットの大きさを作業の量 (チケット更新回数) に比例させることで、量的側面を表現する。矩形の隙間の無い入れ子によって階層構造を表す Treemap [21] の一種を用いることで、チケットをカテゴリ毎の矩形領域に分け (図 4.12 3)、領域間の大きさや中身の対比を可能とする。Treemap の矩形の大きさによって作業量を表すことは [2] などで行われており、作業量を大きさで表すことの直感的さも示されている。Treemap による量の把握と対比に、Gantt、Line、Lines、River 表現を埋め込むことによる進捗の表現のシナジー効果を得る点に本研究の視覚的表現手法の新しさがある。なお、カテゴリ分けに用いる属性をどれにするかや、どの表現を埋め込むのかはユーザーが随時選択、切り替え可能である。

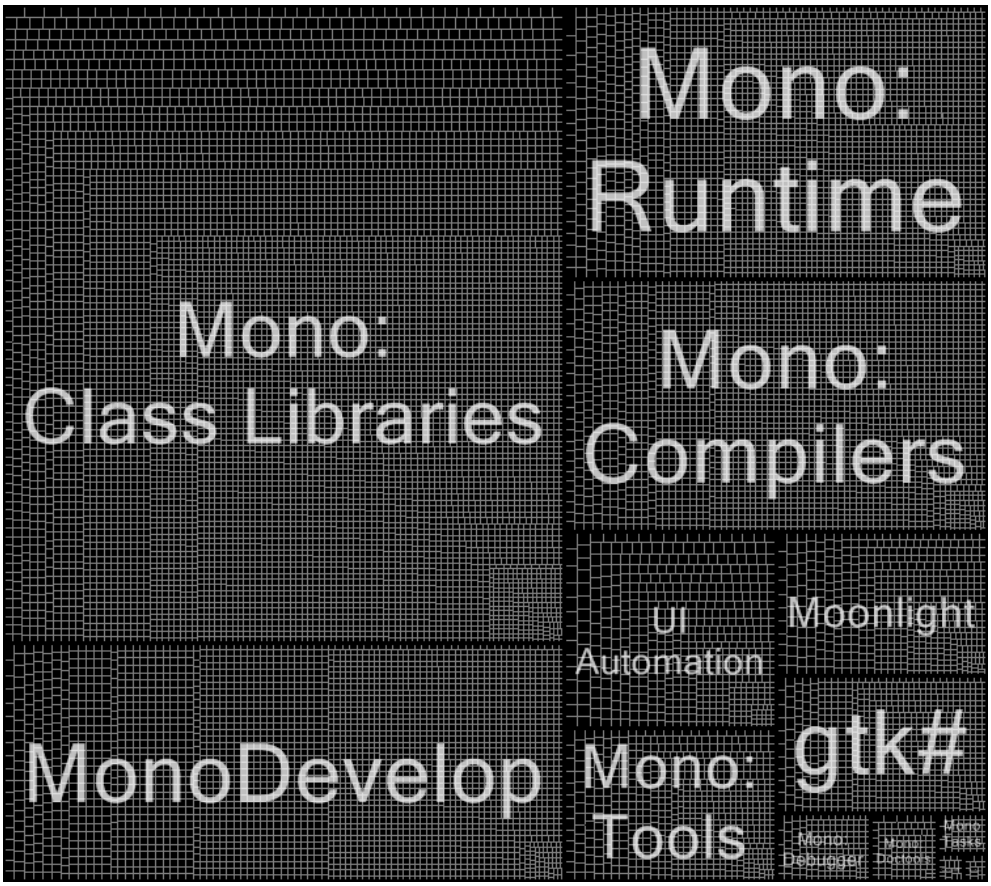

図 4.12: Treemap によって、mono project のチケットを Product 属性値ごとにカテゴリ分けし、個々のチケットの領域を割り当てた図。752x670 px 四方の領域に 19734 チケットの領域が区分けされており、また同じ Product 属性値を持つチケットが一カ所に固まっている。

4.3.1 木構造の構築

Treemap それ自体は木構造を表現する手法である。SaaG では、チケットそれぞれを葉ノード、ユーザーが指定された属性によって、属性値を同じくするチケットごとに束ねることで得られたチケット集合を中間ノード、全チケットをルートノードとする木構造を生成する。なお、中間ノードとはルート要素より下位であり、かつツリー最下位の要素 (葉ノード) より上位のノード全ての呼称である。この際、ユーザーは任意数の属性を順序を付けて指定することができ、最初の属性の属性値が木構造の 1 階層目、次の属性値が 2 階層目、という風に対応づけられる。このようにすることでチケットを葉とする木構造が得られる。

こうして得られた木構造をレイアウトし、出来上がった Treemap の葉ノードの領域に Gantt 表現や Line 表現を埋め込む。なお、Lines 表現や River 表現のようにチケットの集合を対象とする視覚的表現を用いる場合、最も末端の中間ノード (葉ノードの直接の親) の領域をそれら

の表現の描画領域とし、中間ノード内の葉ノードに対応するチケットをチケット集合の元とする。

4.3.2 Treemap と埋め込む表現のレイアウト

Treemap のレイアウトには Benderson らによる squarified layout 手法 [4] を適用することで、できるだけ正方形に近い矩形になるようにしている。これによって、空間の無駄のなさを維持しつつも出来るだけ正方形に近い領域としてカテゴリを表すようにレイアウトしている 図 4.13 。これは、限られたディスプレイの領域内に大量のチケットを表現し、かつ領域内に埋め込まれた表現の対比をサポートするための工夫である。

図 4.13: mono project のデータを Product 属性値でカテゴリ分けし Lines 表現で表現した図。領域内一杯を使い切るように表現を埋めている。

また、Treemap の領域内に埋め込む表現の縦横比が必ず 1:1 になるように、左右ないしは上下に余白を設けることもユーザー設定によって可能とした。縦横比を 1:1 に揃えることで、領域間の形状を見比べる際に縦横比の違いによって惑わされることを防ぐ意図である。1:1 に揃えるために余白を設けることは空間の無駄を作ることになるが、squarified layout 手法によって領域の縦横比を正方形に近づけていることで空間の無駄を減らしている。ただし多少の領域の無駄は存在するため、1:1 にすることで比較しやすくするか、余白を作らないようにするか、のトレードオフはユーザーが選択可能にした。

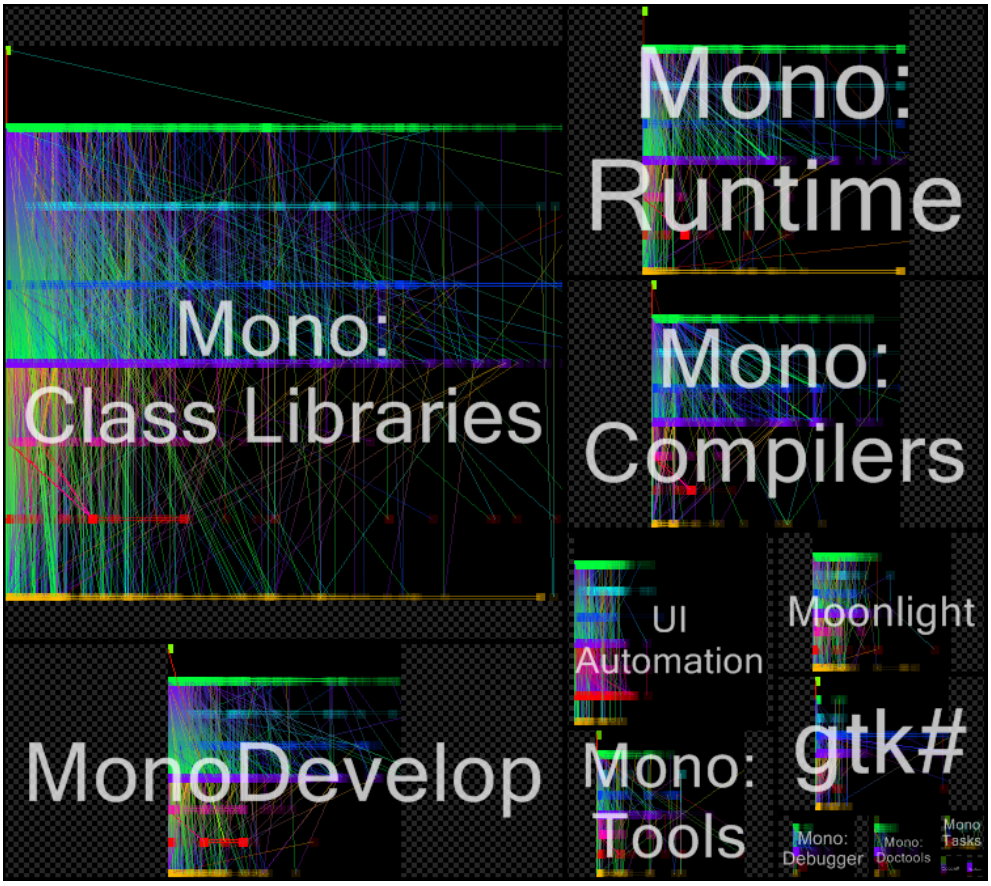

図 4.14: 領域内の視覚表現の縦横比が 1:1 になるように余白を設けている点以外は 図 4.13 と同じ状態の図

第5章　SaaGの開発

チケットデータの分析を支援するために、我々は SaaG (Series at a Glance) というツールを設計、開発した。

5.1　チケットデータの分析アプローチ

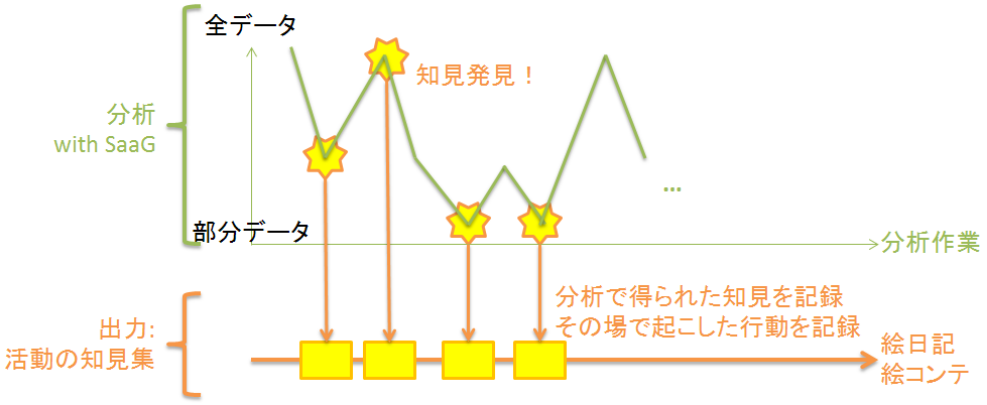

図 5.1: SaaG による分析のプロセスの概念図

活動に関する知見を得るために、我々は、チケットの量的側面と時間的経過の側面とを、人やサブプロジェクトといったカテゴリごとに対比できる形で俯瞰することを支援する。その上で不必要なチケットを除外し、興味深いチケットに着目し、また元の俯瞰に戻る、というプロセスの実現をめざす (Shneiderman のマントラ [22])、 図 5.1 上部)。関連研究の章にて後述するように、チケットの数量 と属性値の時間変化を視覚的に見せるということ自体がそもそも今までに取り組まれて来ていなかった問題である。このように大量の進捗データが人間に見えることと、その上でカテゴリ分けやフィルタの動的な切り替え (試行錯誤) が可能であることは先述の分析プロセスを行う上で必要不可欠なことであり、本研究ではこれを実現する。

加えて、分析プロセスで得られた知見が忘れられ散逸することを防ぐために、紙面にまとめて知見を記録することも支援する (図 5.1 下部)。

5.2 SaaG の入力データ

SaaG はチケットデータを入力として受け取る。チケットデータの下位レベルのフォーマットは JSON か MessagePack [1]、または MessagePack をさらに gzip 圧縮した形式である。SaaG のチケットデータでは、チケットのバージョン一つが連想配列として表され、連想配列のキー (文字列) が属性名、値 (文字列) が属性値を表す。バージョンを表す連想配列には "_At" という特別なキーに、W3CDTF [2]形式での日時文字列が保持され、これはそのバージョンのタイムスタンプである。チケットのバージョンを表す連想配列の配列がチケットと見なされる。配列内の順序は、"_At" の値 (タイムスタンプ) の順である。チケットデータは、チケットを表す配列の配列である。この配列の順序には意味がない。SaaG はこのようなチケットデータをファイル、ないしは http ストリームから読み取る。

5.2.1 チケットデータの俯瞰

我々はチケットデータの俯瞰には以下の二側面をともに表現することが必要であると考える。

1. チケットの量
2. チケットの属性値の時間変化

量の表現は作業がどれほど有るか、無いかを知るために、時間変化は作業がどのように進んでいるのかを知るために必要である。この二つが分かることで、例えば「新規から担当者割当済みになった後進捗が無い (時間変化の側面) チケットが大量にある (量的側面)」といった知見が得られる。

5.2.2 チケット (群) 同士の対比

上記のように量や時間変化による知見が得られた場合、それが何に起因するものであるかが重要である。そのために、担当者やサブプロジェクトといった属性によってカテゴライズし対比することも支援する。

5.2.3 一部のチケットへの着目

上記のような知見が得られたならば、そのチケット (群) だけを表示し、より多くの情報量を得ることが必要とされる。そのために、ビューの一部へのズーミングや、条件によるフィルタリングを実現する。

[1] http://msgpack.org/
[2] http://www.w3.org/TR/NOTE-datetime

5.3　表現の切り替え

これまでに述べた視覚的表現を用いることで、大量のチケットの時間的推移の俯瞰や、カテゴリ間の対比が可能になる。ユーザーは視覚的表現に対して下記の切り替えを行うことで切り口を変える事ができる。データの読み込み後であれば、任意のタイミングに、以下にある切り替えのうち任意のものを行うことが可能である。

5.3.1　表示する属性の切り替え

図 5.2: 属性を選択するためのドロップダウン。ばらつきの度合いが大きい順になっており、属性名の色 (赤〜黄色) もばらつきの度合いに比例した色相になっている。

Gantt、Line、Lines、River 表現において着目する属性をどれにするのかを、チケットデータの持つ全属性の中から一つ選択できる。チケットデータの持つ属性はドロップダウンによって提示されており、個々の属性の属性名には色が付いており、属性値の数も示されている (図 5.2)。ここで、ドロップダウンにおける属性の並び順や属性名の色は属性の値のばらつきの度合い (下記の方法によって計算) によって決定されている。ばらつきの度合いの大きい属性ほど上に表示され、また赤に近い色相となる。ばらつきの度合いの小さい属性は下位に表示され黄色に近い色相となる。

属性の値のばらつきの度合いは、属性の持つ全ての属性値のチケットデータにおける出現回数を数え上げた上で、数え上げた出現回数の情報エントロピーを計算し、最後に属性値の数で割った値である。このばらつき度合いの値は 0 より大きい 1 以下の値となり、全ての属性値がデータ中に均一に出現する場合に 1 を、ある特定の属性値しかデータ中に現れない場合に最小値を取る。

このようにして算出されたばらつき度合いは、属性の選択 UI や、後述の階層分けに用いる属性の選択 UI に用いられている。ばらつき度合いの値を見ることでその属性の持つ情報量が

分かるため、エントロピーが高くも低くもない属性を用いれば、大きなカテゴリから小さなカテゴリまでが存在する Treemap や、偏りのある Gantt、Line、Lines、River 表現を得ることを見込めるため、サイズの差や偏りの様子として情報を読み取ることを期待できる。また、エントロピーが大きい属性を選んだ場合、同程度のサイズのカテゴリが得られるため、同程度の量での対比が可能であることが期待できる。それに対してエントロピーの低い属性であると、単一カテゴリがほとんどを占める Treemap や一つの値しか見られない表現が得られてしまい情報量が少なくなるが、そういった属性はリストの下位に黄色に近い薄い色で表示されている。

5.3.2 カテゴリ分け操作

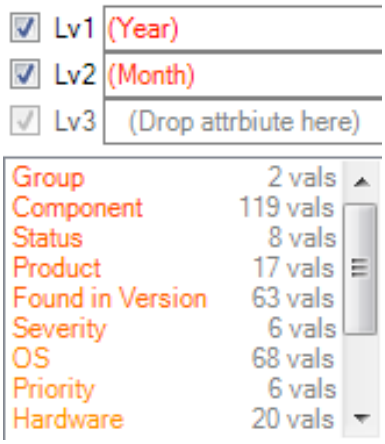

図 5.3: カテゴリ分け (木構造の構築) に用いる UI。下部の属性を選択することで、3 つまでの属性値をカテゴリ分けに用いられる。

Treemap で提示する木構造を構築するために用いる属性を選択できる。原理的には何階層の木構造でも構築しうるが、SaaG では 3 階層 (3 属性) までを選択可能にした。これは 3 以上の階層が実際に用いられることが希であったことと、Treemap が細分化されることによる視認性の低下が著しいためである。

木構造の構築に用いる属性は、先述の着目属性の選択ドロップダウン (図 5.2) と同様の属性のばらつき度合いに基づくリストから属性を選択する UI となっている (図 5.3)。上部にある 3 つの領域は上から順に木構造の階層分けに用いるために選択されている属性を意味する。3 つの領域それぞれにある属性の横のチェックボックスを解除すると、その領域に選択されている属性による階層分けが行われなくなる。下部の属性のリストから上部にある 3 つの領域へ属性をドラッグ＆ドロップすることで属性を選択することができる。また、上部にある 3 つの領域の間でも属性のドラッグ＆ドロップが可能であり、この場合は両者で属性の入れ替えが行われる。なお、下部の属性のリストにある属性をクリックすると、上部の 3 つの領域のうち、最下位にある属性がクリックされた属性に置き換わる。

原理的には、先述の着目属性の選択ドロップダウン (図 5.2) と同種のものを 3 つ並べることも可能であり、SaaG の開発初期の実装ではそのようになっていた。しかし、そのような実装では、階層の入れ替え操作のためにドロップダウン 2 つを選び直す操作 (計 4 クリック) が面倒であるという感想や、3 つのうち 2 つに同じ属性を選んでしまいユーザーが混乱するという結果があった。

そこで次のバージョンでは、属性をドラッグ＆ドロップによって選択する UI にすることで、階層の入れ替えを容易にし、またすでにドロップして「持ち去った」属性はリストからは除外することで同じ属性の複数回選択を違和感なく防ぐことを可能とした。

このように初期バージョンの問題を解決したところ、新たに、様々な属性による切り分けを試行する際のドラッグ＆ドロップの繰り返しが面倒である、ある階層の切り分けをする場合としない場合の見比べのためのドラッグ＆ドロップが同じく面倒である、といったフィードバックが得られた。

そこで最終的なバージョンでは、リスト上の属性をクリックすることでの切り替えも可能とし、また属性の横にチェックボックスを設けることによってある階層の切り分けをする場合としない場合の見比べを可能とすることで、試行錯誤時における上記のような負荷を軽減するようにしたことで、良好なフィードバックを得られた。

5.3.3 表現の種類の切り替え

チケットの進捗の表現法を、Gantt、Line、Lines、River 表現のどれか、ないしは何も無いから選べるようにした。4 つの表現のいずれかを選んだ場合、先述の方法で選択した属性について、表現が描画される。各々の表現の詳細については前記の視覚的表現の説明を参照のこと。

着目する属性がまだ選択されていない場合や、いずれの表現も適用しないことを選んだ場合、Treemap のみが描画され、Treemap の矩形の中には何も描画されず黒一色となる。この状態は、着目する属性が決まっておらず表現を描画できない場合や Treemap についての説明のためのものであり、ツールの使い方のインストラクション以外で積極的に用いることは想定されていない。

5.3.4 時間軸の切り替えや調整

先に述べた 4 種類の表現は全て横軸に時間軸を取るが、この時間軸については絶対時間 (年月日) と相対時間 (チケット開始を 0 とする時間) の二通りが考えられ、4 種類の表現と 2 種類の時間軸は理論的には直交するものと考えることが出来る。しかし、Gantt と Line 表現については、1 チケット分の微細な領域においてさらに絶対時間を横軸に取ると、チケットの表現される横幅が著しく小さくなり現実的でない。そのため、Lines と River 表現について、絶対時間と相対時間の双方を取ることを可能とした。

また、相対時間を取る場合、Lines や River で表示する相対時間の範囲を 図 5.4 のような UI にて指定することを可能にした。これは、例外的なチケットだけが極端に長いことで他のチ

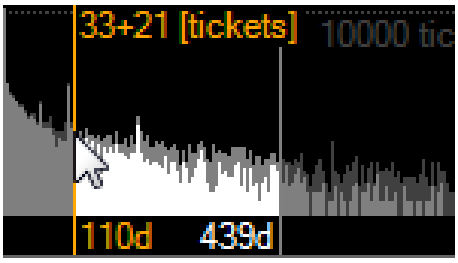

図 5.4: 時間長さの提示・選択 UI。チケットの時間長さがヒストグラムで示されており、ドラッグすることで時間長さの範囲を指定することもできる。

ケットがつぶれてしまうというデータの特性の問題に対処する用途と、「開始から長く時間が経った後の経過だけに着目したい」といった分析ニーズに対応する用途の双方を想定したものである。 図 5.4 に示した UI はヒストグラムになっており、縦軸がチケット数の 10 の対数、横軸がチケット長さになっている。また白い実線のヒストグラムの上に灰色の線のヒストグラムが積み上がっており、それぞれがデータ収集時点において完結しているチケットと、完結していないチケットを表している。ヒストグラムの上でドラッグすることで Lines や River の横軸の相対時間の範囲を指定することが可能であり、たとえば「チケットが半数に減った後にどのような経過をたどったのかだけを子細に観察する」といった操作が可能となっている。

なお、ヒストグラムの縦軸が対数になっているのは、実データにおいては短いチケットの数とそうでないチケットの数に指数的な開きがあり、対数的な軸でないとグラフがつぶれるためである。著者が収集した mono project や redmine のデータではチケットの時間長さは 図 5.4 のように対数分布に近いため、縦軸を対数軸にすることで自然なヒストグラムが得られている。

5.3.5 Gantt や Line 表現の背景色の設定

Gantt や Line 表現の背景はチケットの最終値に基づいた色相で塗られているが、彩度や明度を調整可能にしている。もし背景を前景と同じ彩度・明度で塗った場合、同じ色相の棒や線分が見えなくなりその値の存在感が無くなるという問題や、ビュー全体に対する背景色の影響が強くなり、棒や線によって表されている属性値がビュー全体の見た目に影響しづらくなるといった問題がある。

このような理由があるため、前景の棒や線分によりも背景色は薄い (彩度が低い) か暗い (明度が低い) 状態になる必要がある。しかし、背景色による情報の読み取りを行うためには薄すぎず、暗すぎないことが必要である。このとき、適切な値は表示機器やユーザーの視覚の特性によって異なる。そのため、背景色の彩度や明度を随時調整可能にした。

このことに加え、背景色による情報の表現を行わず、背景色を黒か白にする機能も設けた。これは、チケットの変化だけに着目したく、最終値には興味がない、という時に、チケットの変化の表現 (前景) を最大限読みやすくするためのものである。

5.3.6　縦横比 1:1 固定か非固定かの切り替え

視覚的表現において説明したように、Lines や River 表現を Treemap に埋め込む際に縦横比を 1:1 に固定する (図 4.14) か、縦横比率が一定にならない代わりに領域全体を使うか (図 4.13) を選択可能としている。初期の実装では 1:1 に固定する状態のみであったが、縦横比の不一致による形の比較がしづらくなることを承知の上で出来るだけ大きく描画するニーズもあったために切り替え可能にした。

5.4　ズーミング

俯瞰と対比した後に特定のチケット群に着目するために、ユーザーはズーミング操作を行うことができる。

ズーミングは特定の領域にズームする操作であり、これによって Treemap 上で一箇所に固まっているチケットに着目することが可能である。ユーザーは気になる領域をドラッグによって選択することでズームすることができ、この操作は何段階も可能である。またシングルクリックによってズーム操作一回分だけ元に戻る。これによって、一部に着目し、また元に戻る、という操作を素早く行うことが出来る。

5.5　フィルタリング

特定の側面で見た際に、ユーザーは今は興味を感じないチケットがあることが分かった時や、あるいは特に興味を感じるチケットがあることが分かった時には、フィルタリング操作を行うことが出来る。

たとえば、Status が Closed であるチケットには今は興味がない、あるいは担当が特定のチームであるチケットに今は特に興味がある、といった時にフィルタリング機能を用いることを想定している。

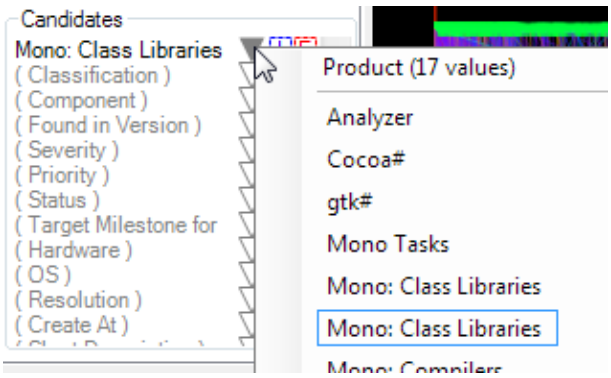

図 5.5: 属性値指定によるフィルタリング

フィルタリングは特定の属性値にマッチするチケットを除外、あるいはマッチするチケットのみを表示する機能である。これによって例えば「終了したチケットを隠す」「特定の担当者のチケットだけを表示する」といった操作が可能である。フィルタリングの条件に用いる属性値はドロップダウンから選択する (図 5.5) ことが可能である。属性値の選択状態は、属性ごとに保持される。選択した属性値を、後述のフィルタ条件に加えることが可能である。

属性値の選択は、ドロップダウンの他にも、ビュー上のチケットやチケット集合のクリックによっても可能である。Gantt や Line 表現のチケットをクリックした場合、全ての属性について、そのチケットの最後の属性値が選択状態になる。Lines や River 表現をクリックした場合、チケット集合のカテゴライズ条件 (Product ごとに Lines を作っているなら Product 属性) に用いられている属性について、その属性の属性値が選択状態になる。ドロップダウンの場合一つの属性の属性値の選択状態が変わるが、チケットやチケット集合のクリックでは全属性や複数属性の属性値の選択状態が変わる。

SaaG のフィルタリング機能は、入力として (属性名, 属性値) の組を 1 つ以上受け取り、これを OR 結合した述語をフィルタに用いる条件として用いる。例えば

$$(\text{“}Status\text{”}, \text{“}CLOSED\text{”}), (\text{“}Status\text{”}, \text{“}RESOLVED\text{”}), (\text{“}Assignedto\text{”}, \text{“}nobody\text{”})$$

という入力は

$$Status \text{ が } CLOSED \text{ か } RESOLVED、\text{あるいは } Assignedto \text{ が } nobody$$

であるチケットにマッチする、という意味として解釈する。

5.5.1 Facet Browsing との関係性

SaaG のフィルタリング手法に関連するものに Facet Browsing [28] がある。これは、多属性データのそれぞれの属性について、どの属性値を持つデータを検索にヒットさせるのかを選ぶことによるフィルタリング・インタフェースである。Facet Browsing を用いたツール[3]では、検索条件に用いる属性値をまず指定し、それによって絞られた限られた件数のデータがリストや表にて表示されるインタフェースが用いられている。すなわち、データを提示するビューは絞り込み前の大量のデータを提示することを前提としていない。これに対して SaaG は、まず全データをビュー上に表示し、ビュー上のチケットを選択することでフィルタ条件の指定、フィルタリングをするインタフェースである。このように、データを絞り込んでからそれを全て表示するという一般的な Facet Browsing に対し、データを全て表示してから絞り込む SaaG という違いが存在する。これは、どのような観点で分析すべきかを、データを見る前ではなく見た後に考えるという SaaG の用途に起因する違いである。

[3] iTunes http://www.apple.com/itunes など

5.6 個別チケットの閲覧

Gantt や Line 表現を用いている場合、チケットをダブルクリックすることでウインドウが開き、その場でチケットを閲覧することができる。チケットの閲覧画面は web ブラウザーになっているため、その場でチケットの更新などのアクションを起こすことも可能である。

5.7　操作ログの記録、ノートテイキング

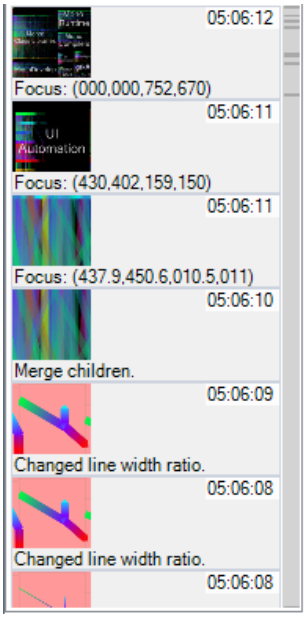

図 5.6: 操作履歴と入力されたノートの一覧

SaaG 上での全ての操作は記録されており、ファイルに保存したり undo/redo する際の目印に用いることができる。操作のログはその時点でのスクリーンショットと共に全て記録され、操作中に見返したり (図 5.6)、undo/redo に用いたり、ファイルに永続化して後日見返す際にロードすることが可能である。

また、ノート記入欄にて文字列を入力することで、最新の操作ログにノートを付加することができる。分析中に得られた発見や今後取ると決めた行動を記録可能にすることで、詳細と俯瞰の往復や切り口の試行錯誤の過程で知見を忘れてしまいうるという、記憶力の限界を補うことを意図したものである。

5.8　絵コンテ・絵日記出力

分析を行ったことで得られた知見や、知見の得られた流れを後から見返すことは、知見の共有や復習の際に行われることである。しかし、例えば平均 5 秒おきに操作があった場合、操作ログは 720 ログ/時にもおよび、この大量のログを見直すことには困難を伴う。そこで、ノートのあった瞬間の画面、操作ログ、カテゴリ分けや選択属性などの状態、時間を絵コンテや絵日記風に印刷する機能 (図 5.7) を実現した。これによって、分析によって得られた一連の知見を紙や画像に集約することが可能になり、これらを集めアーカイブすることやまとめて見直すことを支援した。

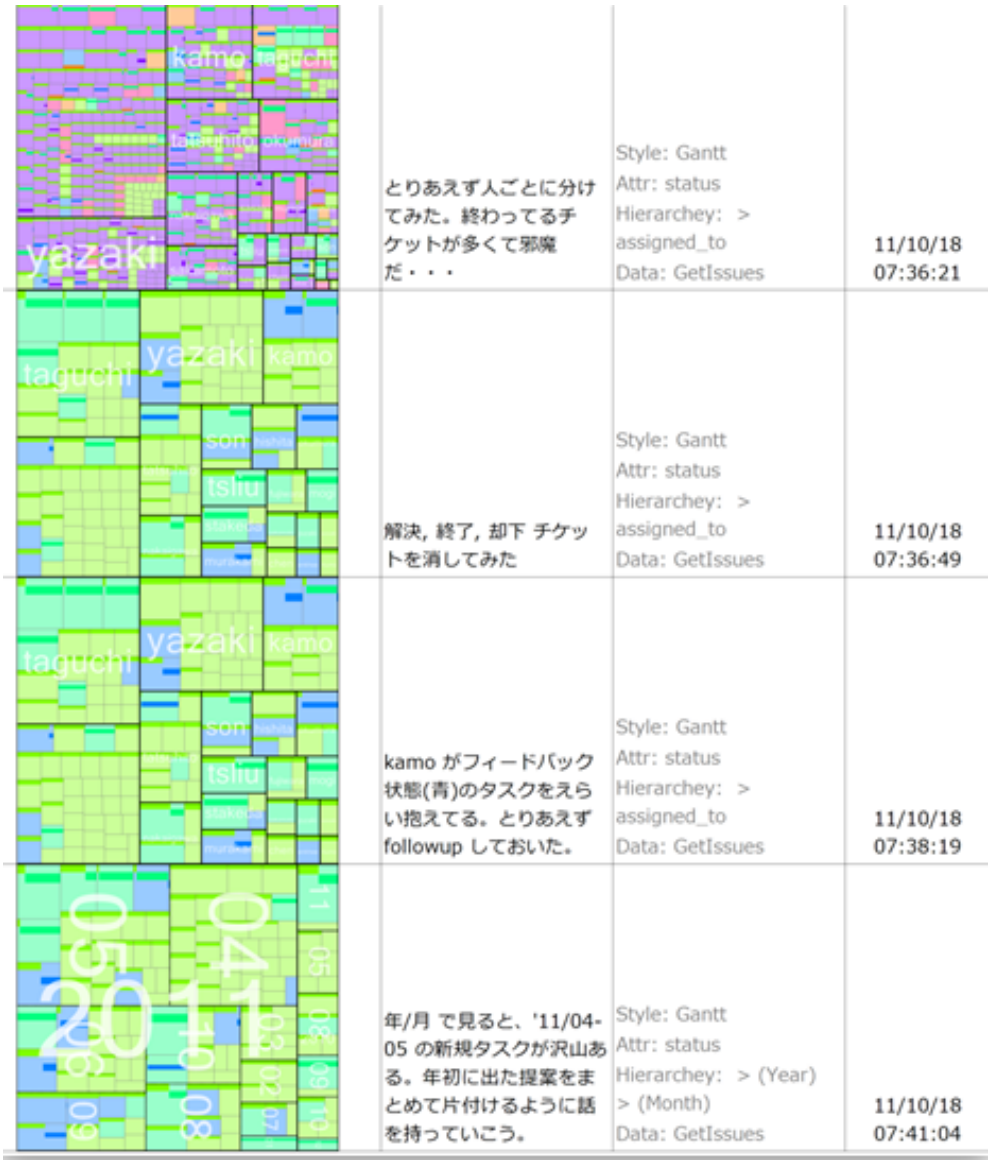

図 5.7: 出力された絵コンテ。左側がスクリーンショット、右側がその時点でのカテゴリ分けや埋め込まれた視覚的表現の設定、それにユーザーの入力したノートの内容。このような絵コンテを画像に保存したり、プリンタを用いて紙に印刷する機能を実装した。

第6章　ケーススタディ

SaaG を用いて実際のデータの分析例を示し、本ツールによって実プロジェクトに対する知見が得られたことを示す。

6.1　プロジェクト観察者による分析

OSS プロジェクトである mono project (19734 チケット, 2007/9 から 2009/9 までのデータ) のデータに対して、大規模なプロジェクトの内部がどのようになっているのかを観察し、既存プロジェクトの実情を調査することを目的として著者が分析を行った。mono project は C# などのコンパイラ、CLI (C# のコンパイル結果であるプログラムの実行環境) の実装、大量のライブラリ実装、などを持つ大きなプロジェクトである。

6.1.1　プロジェクト観察者による分析作業

Mono project は大きなプロジェクトであるが、その内部はいくつかのサブチームに分かれておりそれは Product 属性で表されている。プロジェクトがどのようなサブチームによって成り立っているのかを見ることでプロジェクトの内情を知ることが出来うると考え、Product 属性ごとに分けた Lines 表現で表示した (図 6.1)。Class Libraries について緑から紫になる左右に短い直線や黄色から出入りする線 (図 4.6 で例示) がある。前者は NEW(緑) から短時間で RESOLVED(紫) に、後者は NEEDINFO(黄) になってまた進行中に戻っていることが分かる。ここから、素早く一回の更新で片付けられているタスクや一旦情報不足で止まった後に作業再開しているタスクの存在を読み取れる。また MonoDevelop や Runtime も似た傾向であることが視覚的に分かる。それに対し UI Automation 等の小さめの Prodcut は形状が異なる。また先述の情報不足で止まるタスクの数は大きめのプロジェクトの方が多い、といった差が見える。

小さい Product に着目すると、水色や赤が比較的多く紫が少ない、かつチケットが右方向にあまり伸びていない。このことから、Reopned(水色) や Closed(赤) を多用する代わりに Resolved(紫) を用いず、また所要時間は短いと分かる。ここから Resolve の多いプロジェクトのような作業者以外からの解決 (Resolved) 判定を受けておらず作業のぶり返し (Reopen) が多いことや、作業のペースが速いことを推測できる。この性質は、若い Product の傾向であろうと考え、River 表現を適用する (図 6.2) と、確かに UI Automations は新しい Product であるが、Tools や Moonlight はまた異なる展開を見せていることが分かる。

図 6.1: mono project の各 Product の Status 属性を Lines 表現で見た場合の図

この新しめの Product である UI Automatins についてドリルダウンするために、UI Automations にズームし Line 表現にした (図 6.4)。

Line 表現で見ることで、大きなチケット以外は右下がりであり、背景色には赤が多く、上から下へ直線的なパターンが多いことが分かる。NEW(上) から終了値 (下) への順調なパターンが多いことや、Closed(赤) の量が多いことから、Reopened がより多めだったとはいえ進捗は順調ではあることが推測できる。また、大きなタスクに限っては順調でなく苦闘していることがうかがえる。

また Gantt 表現を適用する (図 6.4) と、大き目のチケットは大半を緑ですごし、断続的に紫や赤になっていることから、断続的に Verified(緑) や Closed(赤) になっている、問題の再発を繰り返している様子が伺える。

6.1.2 分析者の得た知見

分析によって、以下の知見が得られた。

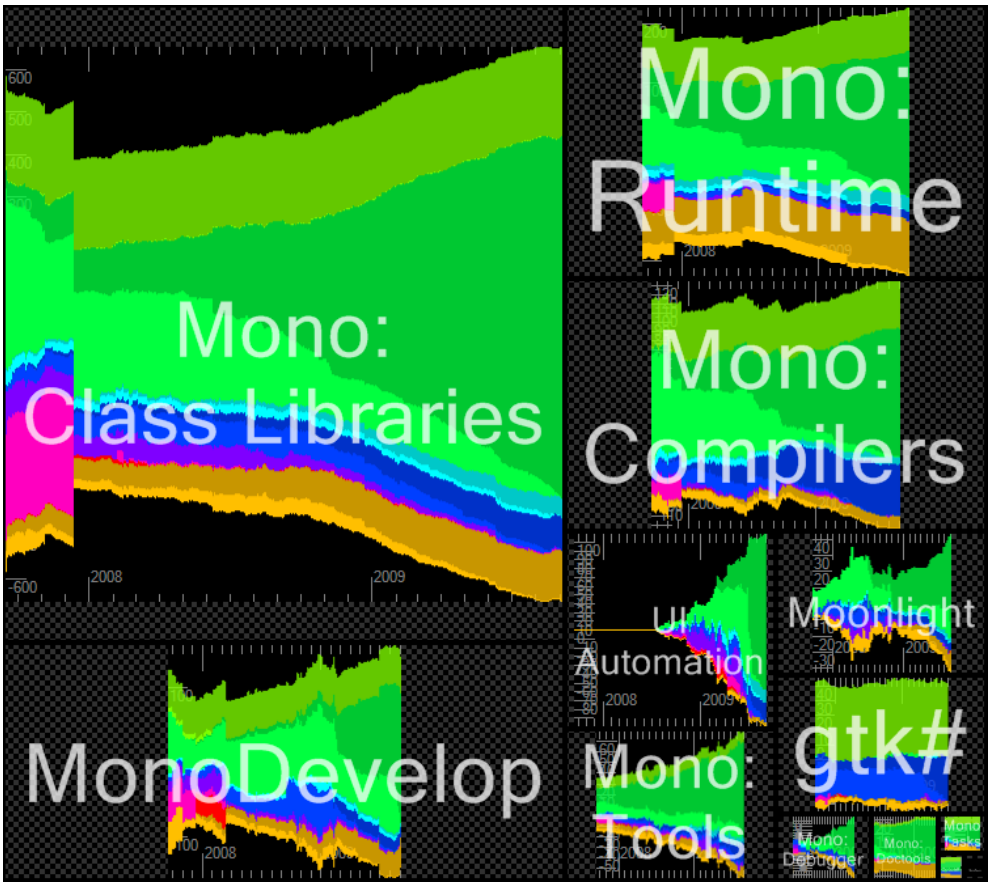

図 6.2: mono project の各 Product の Status 属性を River 表現で見た場合の図

1. Mono Class Libraries や UI Automation をはじめとする Product では、多くの作業が短時間のうちに、1 更新で Resolve されている
2. 作業量の多い Product では情報不足に陥るタスク数が多い
3. 作業量が小さく年季が浅い Product では作業のペースが速い
4. 作業量の小さい Product では作業依頼者や責任者による解決判定がされることが少なく、手戻りが多い

プロジェクトにおける活動の傾向が見いだせているだけでなく、その傾向がどのような背景の場合に言えることなのかを特定できている。

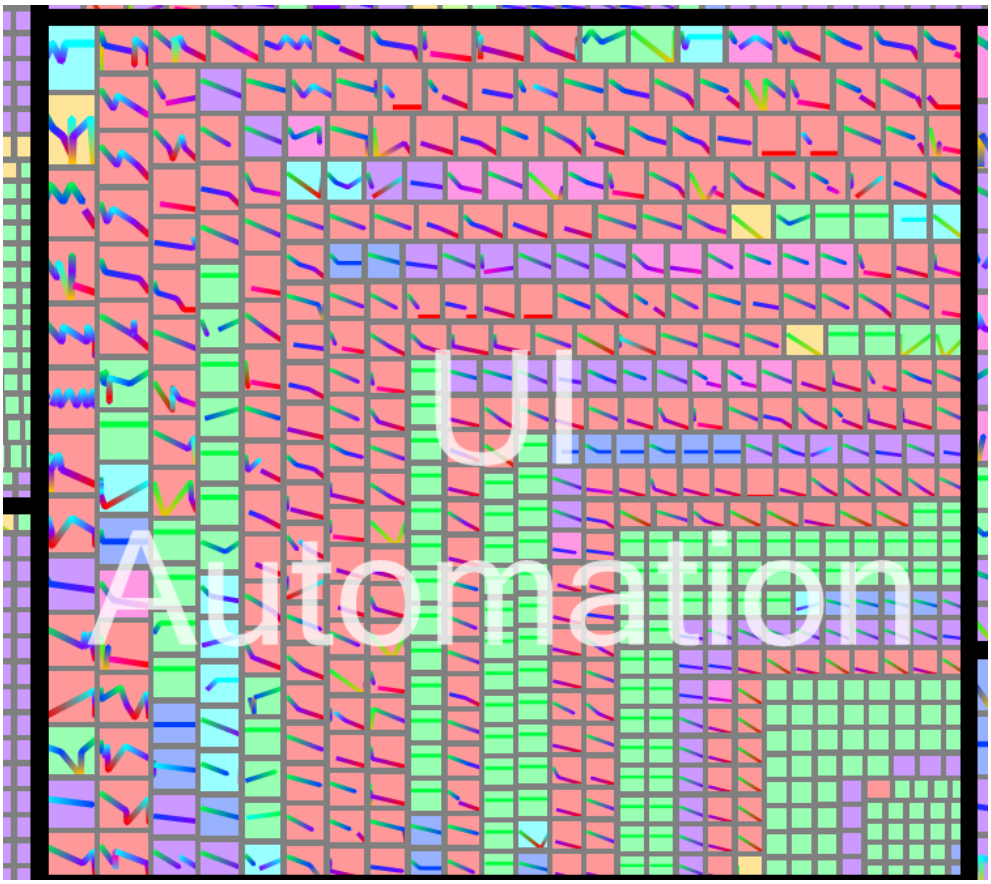

図 6.3: mono project の UI Automation という Product の各チケットの Status 属性を Line 表現で見た場合の図

6.2　プロジェクト管理者による分析

プロジェクトの管理者による分析も実施した。分析者は計算機管理やイントラ向けのサービス提供を行う 23 人ほどチームのリーダー (以下、リーダー) である。このチームはプロジェクト管理ツールによるチケット管理を 1 年半ほど行っているため、分析者はチケットを扱った経験がある。また、分析者は事前に視覚的表現それぞれの縦横軸と色、Treemap の意味、それに視覚的表現の切り替え可能な部分の切り替え方の説明を受けている。しかし分析の目的やゴールなどは与えず、分析者自身が考えるように依頼した。ケーススタディには特に制限時間を設けなかったが、30 分未満で完了した。またケーススタディ完了後に分析の感想を尋ねた。

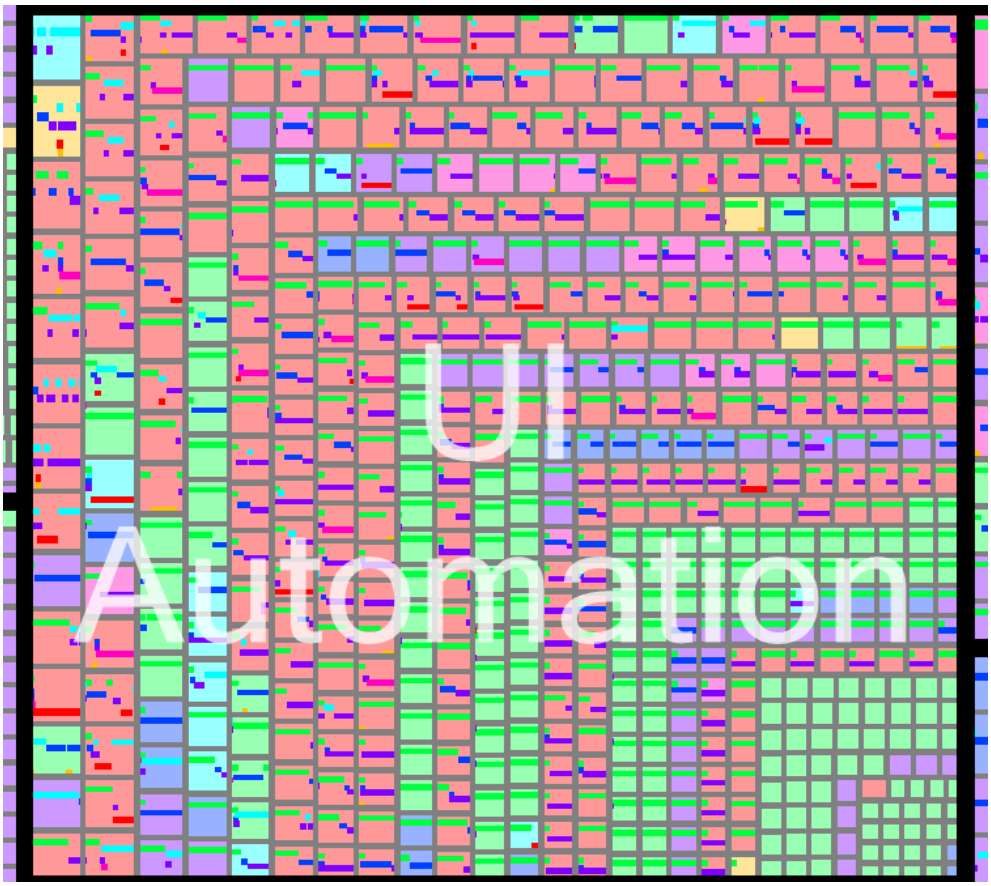

図 6.4: mono project の UI Automation という Product の各チケットの Status 属性を Gantt 表現で見た場合の図

6.2.1 管理者による分析作業

まずリーダーは Tracker 属性 (サブチームに対応) によって階層化を試みた (図 6.5)。

しかし、このチームでは Tracker 属性がきちんと扱われていないため、この属性を介した分析があまり有効に機能しないことを見出した。

そこで、リーダーは人別にカテゴライズし、また終了、却下、解決 Status であるチケットをフィルタリングすることで現在進行中のタスクを人別に俯瞰した (図 6.6)。これによって、自分がタスクを抱えていることを認識した。またサブチームのリーダー (以下、サブリーダー) もチケットを多く持っており、懸念を感じ、リーダーはそのことを記録した。

また、リーダーは大きなチケット (更新の多いチケット) が妙に多いことに気づき、それらの詳細を見たところ既に終わったタスクや報告作業が忘れられたチケットであったと理解し、その場でチケットを閉じたり、報告作業を催促した。

図 6.5: チームの各サブチームの Status 属性を Gantt 表現で見た場合の図

6.2.2 分析者の得た知見、取った行動

分析によって、リーダーは以下のような知見を得た。

1. 自分自身がタスクを抱えてしまっている
2. サブチーム内の仕事の割り当てが滞っている

また、リーダーは以下の行動を分析中におこなった。

1. 報告作業を忘れていたタスクを処理
2. 閉じ忘れていたチケットを閉じる
3. OB に割り当てられたままのタスクの処理

問題のあるチケットの存在を認知し、さらにそれらがどの様な分け方 (人、サブチーム、報告待ち状態) で見いだせる物であるか、どう処理するのが良いものか、を短時間の内に見いだしていることが伺える。

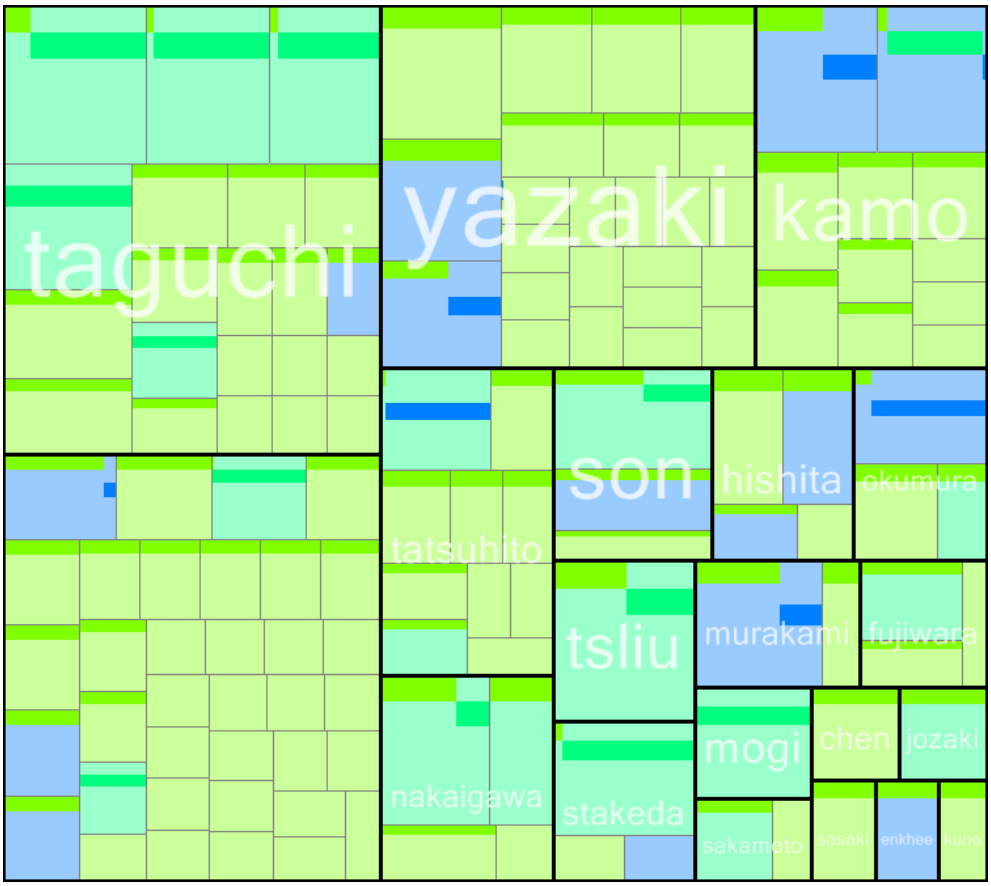

図 6.6: チームの各担当者の Status 属性を Gantt 表現で見た場合の図

6.2.3 分析者の感想

リーダーは、従来のチケット管理システムでは見落としてしまっていた事柄を発見でき有意義であると述べた。また今までは行動 a, b, c といったことをするために、専用の時間を数時間以上設けてチケットを整理する作業をしていたが、SaaG では素早く同等のこれらの成果が得られたとも述べた。これは量や時間変化を俯瞰する機能が従来のツールよりも有用であったことを示しているといえる。絵コンテ風の出力については便利であるとのことであったが、分析中のノートテイキング機能についてはわずらわしいとのことであり今後の課題であると考えられる。

6.3 分析における機能の活用

ケーススタディを通して、分析に使えそうな属性やフィルタリング条件を見つける、特異なチケット (群) を見つけその特色を理解する、別の見方からその裏づけを得る、といった発

見が実際にあった。

これらの知見を得る過程ではここまでに述べた視覚的表現それぞれが使われており、また属性の切り替えやズーミング、フィルタリングといった操作が活用されていた。また操作による切り口の試行錯誤では全データの俯瞰が判断に用いられていた。分析中の記録については記録操作の煩わしさの軽減が課題である。

分析を通して、限られた時間内にて従来のツールの機能では困難のある知見発見が実現されており、知見の獲得やプロジェクトの改善に繋がっていた。

第7章　今後の課題

7.1　配色の工夫

SaaG の視覚的表現では属性の変化を形によって表しているが、表現と実際の属性値の対応付けは色に頼っている。この色の配色は現在では色相環上に等間隔に割り当てるか、あるいは手作業で指定された色を用いている。この色と属性値の対応付けについては、適切な配色を自動で割り当てることが考えられ、単純な解決策としてはたとえば ColorBrewer [15] などが提供している決まったパレットを用いることが考えられる。またさらなる発展として、たとえば作業が止まっている値は赤くするといった意味的な側面と、図上で頻繁に隣接する値は離れた色にするといった図としての判読性の側面双方を考慮して最適な色を求めるといった方向が考えられうる。

属性値と色の対応付けの問題に加えて、図上での色の混合方法についても発展が考えられる。SaaG の表現では、Lines における線分の重なりは R, G, B それぞれの半透明合成によって合成されているが、この方法では反対色同士が混色した際の読み取りが困難になる、知覚的に正しい中間色でないために認知しづらくなる、といった問題が考えられる。SaaG の Lines では縦軸の位置によって色に制約があること、グラデーションする線という文脈が与えられていること、あらゆる属性から任意の属性へランダムに状態変化するといったデータは実用上希であること、といった特性があるが、より一般的なデータや用途に応用する際には問題になりうると考えられる。その際には、色相を維持した色の合成 [8] を用いる、色と色の合成を行わずモザイク様の文様によって表す [14] といった工夫が考えられるであろう。また半透明による重なり表現の精神学上の効果 [10][24] を考慮する工夫を行うことは、SaaG が行っている領域の上にラベルなどの付加情報を表示する事の可読性に貢献するであろう。

7.2　評価手段の確立

SaaG の支援する分析プロセスは、分析の切り口自体を試行錯誤するものである。そのため、たとえば「放置されているチケットが何個有るか」「もっとも負担の大きな担当者はだれか」といったクエリそれ自体を発見できたかどうか、の定量的な評価を行う手法の確立もまた追求の余地がある。そのためには、発話的思考法の利用といった測定手段の導入や、実プロジェクトの管理者を多く集めるといった大規模な実験を行いデータを得るフィールドスタディを行った上で、被験者が得た発見の価値や、その発見が得られるまでの被験者にとっての労力の大きさを定量的に評価する何らかの基準の確立が求められるであろう。

第8章　まとめ

本研究ではチケットの時間変化の視覚表現を属性値ごとに敷き詰めることで、大量のチケットの量と時間変化の俯瞰や対比を可能とする視覚表現を開発し、それを用いてツール SaaG を開発した。今回はチケットデータを対象としたが、企業の内部情報といったデータであっても、時間変化する多属性データでさえあれば視覚表現による分析対象にできうると考えられる。

活動の記録をネットワーク上で常時共有し蓄積することが一般化してきている現在、そこから知見を見出すための支援が必要であると考えられる。本研究の活動情報の俯瞰や試行錯誤の支援の手法は、今後のデータ分析手法の発展の手がかりになると思われる。

謝辞

本研究を行った 2 年間、三末和男准教授には日々の研究活動において多大なご指導を頂きました。充実した内容の研究を行うことができ、また無事に論文執筆ができたのは先生のご指導のおかげです。本当に有り難うございました。

田中二郎教授には研究生活において多大なアドバイスを頂きました。心から感謝しています。高橋伸准教授、志築文太郎講師、Simona Vasilache 助教には研究発表等の場において有意義な意見を頂き、研究を進める上で大変参考になりました。有り難うございました。インタラクティブプログラミング研究室の皆様には公私共に大変お世話になりました。学生生活最後の 2 年間をインタラクティブプログラミング研究室の仲間と共に過ごすことができ、実に嬉しく思います。

最後に、学生生活を送る上で多大な援助をして下さった家族には大変感謝しています。家族の支えなしにはこれまで充実した生活を送ることはできなかったと思います、本当に有り難うございました.

参考文献

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE transactions on visualization and computer graphics*, 14(1):47–60, 2008.

[2] M. Baker and S. Eick. Space-filling software visualization. *Journal of Visual Languages and Computing*, 6(2):119–133, 1995.

[3] T. Ball and S. Eick. Software visualization in the large. *Computer*, 29(4):33–43, Apr. 1996.

[4] B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *AcM Transactions on Graphics (TOG)*, 21(4):833–854, 2002.

[5] L. Byron and M. Wattenberg. Stacked graphs–geometry & aesthetics. *IEEE transactions on visualization and computer graphics*, 14(6):1245–52.

[6] R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, J. Yang, E. Suma, C. Ziemkiewicz, D. Kern, and A. Sudjianto. WireVis: Visualization of Categorical, Time-Varying Data From Financial Transactions. *2007 IEEE Symposium on Visual Analytics Science and Technology*, pp. 155–162, Oct. 2007.

[7] J. Chen, A. M. MacEachren, and D. J. Peuquet. Constructing overview+detail dendrogram-matrix views. *IEEE transactions on visualization and computer graphics*, 15(6):889–896, 2009.

[8] J. Chuang, D. Weiskopf, and T. Möller. Hue-preserving color blending. *IEEE transactions on visualization and computer graphics*, 15(6):1275–82.

[9] J. Ellis, S. Wahid, C. Danis, and W. Kellogg. Task and social visualization in software development: evaluation of a prototype. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 577–586. ACM, 2007.

[10] F. Faul and V. r. Ekroll. Psychophysical model of chromatic perceptual transparency based on substractive color mixture. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 19(6):1084–95, June 2002.

[11] M. Fischer and H. Gall. MDS-Views: Visualizing problem report data of large scale software using multidimensional scaling. In *Proceedings of the international workshop on Evolution of large-scale industrial software applications (ELISA)*, pp. 110–121, 2003.

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

[13] H. L. Gantt. *Work, wages, and profits*. Management in History No 41. Hive Pub. Co., 1916.

[14] H. Hagh-Shenas, S. Kim, V. Interrante, and C. Healey. Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. *IEEE transactions on visualization and computer graphics*, 13(6):1270–7, 2006.

[15] M. Harrower and C. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *Cartographic Journal, The*, 40(1):27–37, 2003.

[16] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: Visualizing theme changes over time. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pp. 115–123. IEEE, 2000.

[17] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, July 2002.

[18] C. Muelder, F. Gygi, and K.-L. Ma. Visual analysis of inter-process communication for large-scale parallel computing. *IEEE transactions on visualization and computer graphics*, 15(6):1129–1136, 2009.

[19] M. Ogawa and K. Ma. Code_Swarm: a Design Study in Organic Software Visualization. *IEEE transactions on visualization and computer graphics*, 15(6):1097–1104, 2009.

[20] M. Ogawa and K. Ma. Software evolution storylines. In *Proceedings of the 5th international symposium on Software visualization*, pp. 35–42. ACM, 2010.

[21] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, Jan. 1992.

[22] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336–343. IEEE, 1996.

[23] B. Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 3–12. ACM, 2008.

[24] M. Singh and D. Hoffman. Part boundaries alter the perception of transparency. *Psychological Science*, 9(5):370, 1998.

[25] M. Storey, D. Čubranić, and D. German. On the use of visualization to support awareness of human activities in software development: a survey and a framework. In *Proceedings of the 2005 ACM symposium on Software visualization*, Vol. 1, pp. 193–202. ACM, 2005.

[26] T. Takada and H. Koike. Mielog: A highly interactive visual log browser using information visualization and statistical analysis. In *Proceedings of LISA XVI Sixteenth Systems Administration Conference*, pp. 133–144, 2002.

[27] K. Vrotsou, J. Johansson, and M. Cooper. Activitree: Interactive visual exploration of sequences in event-based data using graph similarity. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):945–952, 2009.

[28] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pp. 401–408, New York, NY, USA, 2003. ACM.

[29] ドナルド・E. クヌース. 文芸的プログラミング. アスキー・メディアワークス, 1994.

[30] 牛田貢平. 列車運行実績データの可視化. 交通・物流部門大会講演論文集, 2009(18):347–350, 2009.