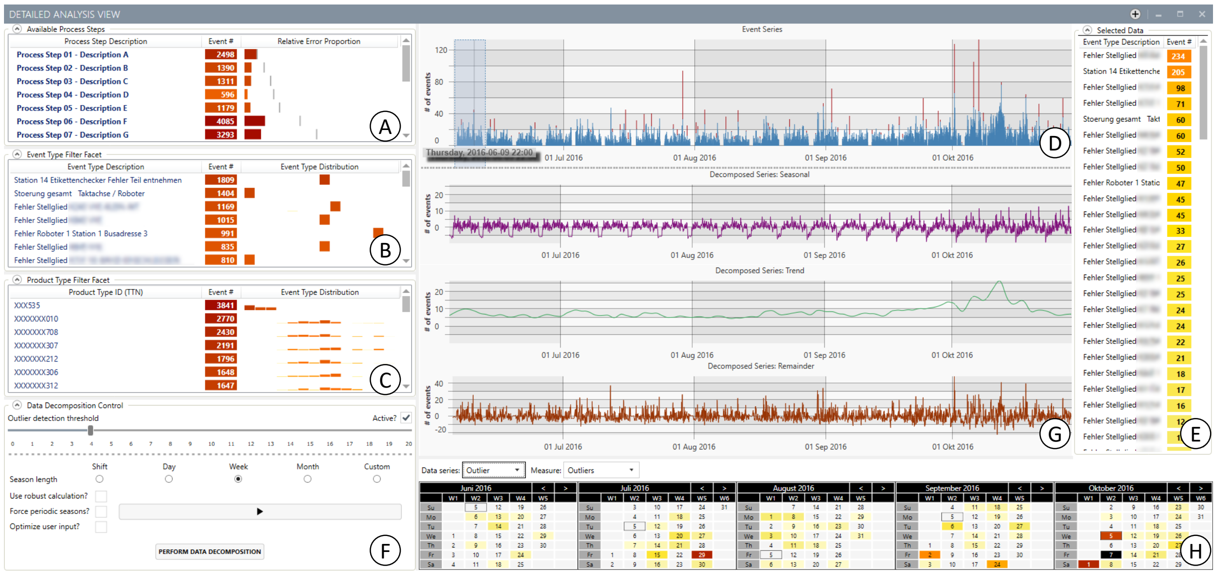
# Visual Analytics for Decomposing Temporal Event Series of Production Lines

Dominik Herr[1,2], Fabian Beck[3], and Thomas Ertl[1]
[1]Institute for Visualization and Interactive Systems[*]
[2]Graduate School of Excellence advanced Manufacturing Engineering
University of Stuttgart, Stuttgart, Germany
[3]paluno, University of Duisburg-Essen, Germany[†]



Available Process Steps

| Process Step Description | Event # | Relative Error Proportion |
|---|---|---|
| Process Step 01 - Description A | 2498 | |
| Process Step 02 - Description B | 1390 | |
| Process Step 03 - Description C | 1311 | |
| Process Step 04 - Description D | 596 | |
| Process Step 05 - Description E | 1179 | |
| Process Step 06 - Description F | 4085 | |
| Process Step 07 - Description G | 3293 | |

Event Type Filter Facet

| Event Type Description | Event # | Event Type Distribution |
|---|---|---|
| Station 14 Etikettenchecker Fehler Teil entnehmen | 1809 | |
| Stoerung gesamt Taktachse / Roboter | 1404 | |
| Fehler Stellglied [redacted] | 1169 | |
| Fehler Stellglied [redacted] | 1015 | |
| Fehler Roboter 1 Station 1 Busadresse 3 | 991 | |
| Fehler Stellglied [redacted] | 835 | |
| Fehler Stellglied [redacted] | 810 | |

Product Type Filter Facet

| Product Type ID (TTN) | Event # | Event Type Distribution |
|---|---|---|
| XXX535 | 3841 | |
| XXXXXXX010 | 2770 | |
| XXXXXXX708 | 2430 | |
| XXXXXXX307 | 2191 | |
| XXXXXXX212 | 1796 | |
| XXXXXXX306 | 1648 | |
| XXXXXXX312 | 1647 | |

Data Decomposition Control
Outlier detection threshold — Active? ✓
Season length: Shift ○ Day ○ [✓ Week] Month ○ Custom ○
Use robust calculation? ☐
Force periodic seasons? ☐
Optimize user input? ☐
PERFORM DATA DECOMPOSITION

Data series: Outlier — Measure: Outliers

Selected Data

| Event Type Description | Event # |
|---|---|
| Fehler Stellglied [redacted] | 234 |
| Station 14 Etikettenche | 205 |
| Fehler Stellglied [redacted] | 98 |
| Fehler Stellglied [redacted] | 71 |
| Stoerung gesamt Takt | 60 |
| Fehler Stellglied [redacted] | 60 |
| Fehler Stellglied [redacted] | 52 |
| Fehler Stellglied [redacted] | 50 |
| Fehler Roboter 1 Statio | 47 |
| Fehler Stellglied [redacted] | 45 |
| Fehler Stellglied [redacted] | 45 |
| Fehler Stellglied [redacted] | 33 |
| Fehler Stellglied [redacted] | 27 |
| Fehler Stellglied [redacted] | 26 |
| Fehler Stellglied [redacted] | 25 |
| Fehler Stellglied [redacted] | 25 |
| Fehler Stellglied [redacted] | 24 |
| Fehler Stellglied [redacted] | 24 |
| Fehler Stellglied [redacted] | 22 |
| Fehler Stellglied [redacted] | 21 |
| Fehler Stellglied [redacted] | 18 |
| Fehler Stellglied [redacted] | 17 |
| Fehler Stellglied [redacted] | 16 |
| Fehler Stellglied [redacted] | 12 |
| Fehler Stellglied [redacted] | 1 |

Figure 1: The facet views Ⓐ - Ⓒ on the left provide a first overview of the reported events' distribution along the production line. The filtered events' temporal distribution and optionally its outliers are presented as a stacked bar chart Ⓓ. Through brushing, analysts can get more information about the reported events during the selected timespan Ⓔ. The event series can be decomposed with STL to find recurring events and trends. The decomposition parameters are set in the *Data Decomposition Control* Ⓕ and its results shown as three line charts Ⓖ. The *Calendar Plot* Ⓗ provides information about outlier or value distributions of the data.

## Abstract

The temporal analysis of events in a production line helps manufacturing experts get a better understanding of the line's performance and provides ideas for improvement. Especially the identification of recurring error patterns is important, because these patterns can be an indicator of systematic production issues. We present a visual analytics approach to analyze event reports of a production line. Reported events are shown as a time series plot that can be decomposed into a trend, seasonal, and remainder component by applying Seasonal Trend decomposition using Loess (STL). To find specific event patterns, the data is filtered based on aspects such as the event description or the processed product. Identified temporal patterns can be extracted from the original event series and compared visually with each other. In addition to predefined settings, experts can define a subseries of the event series and the period length of STL's seasonal component through an automatically optimized brushing of the undecomposed plot. We developed the approach together with an industry partner. To evaluate our approach, we conducted two pair analytics sessions with our industry partner's experts. We demonstrate use cases from these sessions that showcase our approach's analytical potential. Moreover, we present general expert feedback that we collected through semi-structured interviews after the pair analytics sessions.

**Keywords:** Manufacturing, Industry 4.0, spatio-temporal data, correlation analysis

[*]{firstname.lastname}@vis.uni-stuttgart.de
[†]Fabian.Beck@paluno.uni-due.de

## 1 Introduction

With advances in automation of production lines and digitalization, many manual labor activities in manufacturing plants have been replaced by computer-controlled machinery [11]. These machines do not only increase productivity, but also for the first time, make the production process fully transparent with respect to throughput, production quality, and anomalies. They generate a large amount of temporal data, such as process parameters and reports of events. As the collected data is multivariate and high-dimensional, it is often unclear, which subsets of the data should be focused on to improve a factory's productivity. Currently, most of the data is used to generate daily overview reports for technical managers or to reflect on recent anomalies during technical meetings of technicians. Although this analysis of recent events is suited to spot currently relevant problems, it is neglecting hidden, but as severe long-term issues that correlate with a trend or have a repeating pattern.

Our goal is enable production experts to find and understand long-term issues of production assembly lines. To achieve this, we present a visual analytics approach to iteratively decompose event logs to find recurring event patterns. Our system presents original and decomposed event series as time series plots and in a calendar view (see Fig. 1).

Experts analyze subset by subset of events: flexible faceted browsing [31] enriched with visual summaries offers an overview and quick filtering based on involved process steps Ⓐ, event types Ⓑ, and product types Ⓒ. The filtered events' temporal distribution, as well as the data's outliers, are shown in a stacked bar chart Ⓓ. Through brushing, experts can inspect temporal subsets of the events Ⓔ. The system further supports decomposing the event series into a trend, a seasonal, and a remainder signal by applying *Seasonal Trend decomposition using locally weighted regression (loess)* (STL) [10] Ⓕ. The results of the decomposition are then visualized as time series plots Ⓖ and can be viewed in a calendar view Ⓗ. Analysts can choose from predefined settings or by interactively selecting the recurring pattern in the visualization to set the necessary parameters. Since an exact selection is difficult, an evolutionary algorithm supports experts in optimizing the result. Relevant analysis results can be stored as decompositions of data subsets. An additional view facilitates a later extension and comparison of the results. In this way, we support an iterative analysis process, where experts collect findings and incrementally refine their understanding of the long-term issues of the production line.

This paper studies a novel application of visual analytics in the manufacturing domain. It combines established techniques from visualization, optimization, and interaction into a computer-supported, integrated analysis approach. The system contributes

- visually enriched facet browsing to filter different aspects of the event series,
- seasonal trend decomposition with STL that is combined with an implicit parameter setup optimized using an evolutionary algorithm, and
- an iterative analysis explaining temporal patterns for different subsets and varying parameters.

We developed our approach in collaboration with an industry partner who produces small to mid-sized electric motors. Experts from our industry collaboration partner evaluated the approach together with us in several pair analytics sessions. We present use cases from these sessions and general expert feedback collected through semi-structured interviews. The results show that our approach facilitates manufacturing experts to gain relevant and previously unknown insights into systematic issues of a production line.

## 2 Related Work

Our approach represents event frequencies over time and is based on standard visualizations for time-dependent data [1]. It uses a *juxtaposition* (or *small multiples*) and *superposition* (or *overlay*) for visual comparison [12] of the decomposed time series. Related approaches are visual analytics systems for the manufacturing domain and of event sequences.

### 2.1 Visual Analytics in Manufacturing

Most visualization and visual analytics approaches that target the manufacturing domain focus on the optimization of simulations and production schedules.

Rohrer [23] argues that visualization helps domain experts to get a better understanding of manufacturing simulations, for example, by visually representing paths that operators take between machines. Also, visualization enables an interactive communication of results between a simulation software and its users. Another example is the work of Wörner and Ertl [29] who present a production simulation framework with an integrated visual analytics approach.

To optimize production schedules, Klöpper et al. [17] present a system that generates a set of possible production schedules that can be iteratively reduced based on aspects that experts deemed to be currently most important. *LiveGantt* [15] helps experts to explore Gantt charts of large concurrent schedules. Users can interact with the schedule and get visual feedback about their changes' effects.

*ViDX* [30] analyzes a production line's performance based on product tracking data with the goal to better understand effects of machine problems. It extends a Marey's graph to visualize products moving through a production line. Outliers are visually emphasized by aggregating products with similar process times. Further, the approach provides real-time tracking of a production line's performance. The visualization of individual products and their processing times improves the understanding of a line's performance and helps to understand the effects of problems in a production line.

In contrast to ViDX, our approach focuses on the detection of systematic issues in a production line based on event reports filed by machine tools with the goal of finding recurring issues over an extended period of time.

### 2.2 Visual Event Series Analysis

Analyzing temporal patterns of events is also relevant outside the manufacturing domain. For instance, there exist event visualization tools focusing on security issues [14,19], meteorological and oceanographic events [4], or historic events manifested in documents and media [2,18,20]. Like in our approach, they visualize events over time, but do not allow for decomposing the time series interactively into trends and seasonal components.

There exist only few other visual analytics approaches that build on such a decomposition of time series. Bögl et al. [7] provide interactive visual guidance for selecting appropriate parameters of ARIMA and seasonal ARIMA models, which decompose a time series similarly to STL. Whereas we use multiple decompositions to reflect different types of events, their goal is to fit a time series with a single model. Follow-up work [5,6] adds predictive analysis features to their approach. Chae et al. [9] apply STL on Twitter data to filter out any seasonal and trend effects to visualize unusual events. They assume that the outliers contained in the data without trend or seasonal series indicate abnormal events that could be of interest. Maciejewski et al. [21] use STL to forecast hotspots of geo-located events. These works are rather interested in prediction and outlier detection than explaining the time series to the analyst by applying a decomposition.

Some information visualization techniques of time series particularly highlight seasonal pattern, for instance, spiral plots of the time axis [8,28]. We did not use such radial diagrams, however, because they are harder to label and more difficult to juxtapose like necessary in our scenario. Alternatively, the time series can be split by season and plotted overlaid in 2D, juxtaposed in 3D, or encoded in color in a calendar grid (or any other 2D grid) [27]. This procedure scales even to large time series when color-coding the values of the series in a pixel grid [16]. Splitting the series by different seasonal lengths provides several blocks of plots, which themselves can be juxtaposed [24]; it becomes difficult, however, to see trends and seasonal patterns because the time series is not explicitly decomposed. Cycle plots [22] use a form of dimensional stacking to compare data points from different seasons on a linear axis; this is limited to a single decomposition with a single season length at a time, but we want to support the analysis of multiple decompositions with different season lengths. We work with decomposed straight-line time axes and use a color-coded calendar grid [27] as an additional view for better showing weekly and monthly patterns.

## 3 Event Reports of Production Lines

We developed our visual analytics approach in close collaboration with an industry partner. This section presents the specific scenario

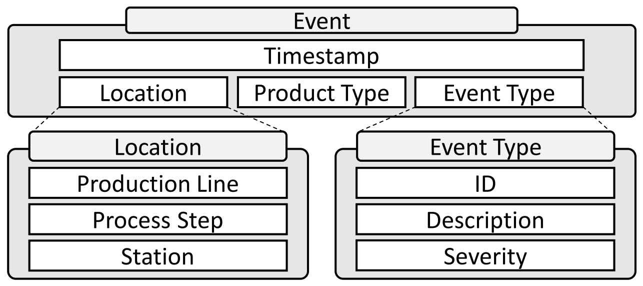


Figure 2: Data model of our industry partner. Each event comprises a timestamp, the location of the machine that reported the event, the product type that was processed, and an event type.

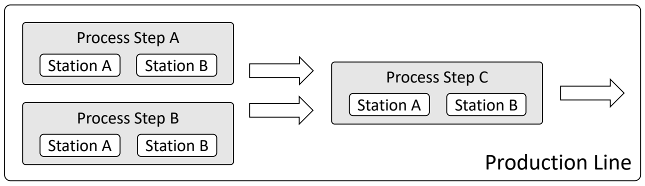


Figure 3: Excerpt of a typical production line structure that comprises several process steps, each composed of several stations that perform the production tasks.

of the production line we study. We derived requirements for a visual analytics approach to find recurring machine tool problems in a production line and introduce the data that is available for analysis.

### 3.1 Specific Scenario of our Industry Partner

Our industry partner produces small to mid-sized electric motors. The production line we study produces six types of motors with 84 variants overall and comprises 14 process steps. Amongst others, the production line records data of its stations' process times and what product type is being produced. Currently, the data is used to monitor the processing time and deviations are investigated at the shop floor. The data analysis is done on two levels: in a daily meeting, all technicians discuss recently occurring problems. The analysis of specific problems is usually conducted by looking at the process times and occasionally at the reported events during a specific time period, which is usually one day. Further, the head of the production line irregularly analyzes the process times of various stations. Such an analysis may include the data of the past hours up to a few weeks. If the head of the production line finds any anomalies, he consults the technicians at the shop floor to get further details about the identified problem and assesses solution strategies. Our goal is to support the head of the production line in finding events that recur over an extended time period.

### 3.2 Available Data

Our industry partner provided us with real production data, which contains events that are automatically reported by machine tools. Each report comprises a timestamp, the location where the event occurred, the product type that was processed during the event, and the event type (see Fig. 2). The location is composed of the production line, which is separated into several process steps that contain stations. Fig. 3 shows an excerpt of a typical production line. The event type is detailed by an ID, a human-readable description, and a severity level (information, error, etc.). During our study only considered event reports that have the severity level *error*, as these have the highest impact on productivity.

### 3.3 Requirements

We had several meetings with domain experts of our industry partner to understand what insights they hope to find in their data. Currently, analyses are executed on short time intervals and issues of the past day. We decided to support the analysis of problems that occur regularly over an extended time period to find previously unnoticed event patterns and identified the following requirements:

Requirement 1: **Overview & Faceted Information** ($R_1$)

Present an overview of the available events with respect to the process step, event type, and product type. Further, provide interactive data filters and visualize the events' temporal distribution.

Requirement 2: **Pattern & Outlier Identification** ($R_2$)

Help analysts to semi-automatically find seasonal patterns, trends, and outliers of the reported events.

Requirement 3: **Extract & Compare** ($R_3$)

Facilitate analysts to extract analysis results, visually compare them, and interactively extend them.

## 4 Approach

Our concept comprises three parts: first, analysts get an overview of the data and filter it according to their needs with facet views. Second, they analyze the filtered data regarding its temporal aspect. Third, analysts can extract and compare findings (see Fig. 4).

### 4.1 Data Subset Configuration through Faceted Search

Initially, the analysts choose a time period. The following analysis is conducted in the *Detailed Analysis View* (Fig. 1). The *Facet Filter Panel* on the left uses faceted browsing [31] to filter the events based on their process step, product type, and event type. All facets have a common color scheme for event counts, which ranges from a light yellow via orange and red to black. In most cases, the facets also provide an overview of the events' distribution along the production line's process steps. The distribution is visualized as a sparkline visualization [26] that presents the production line's process steps as bars, with descriptions available via tooltips. The height of a bar encodes the number of events of the step normalized per row.

**Process steps.** The *Process Step Facet* (see Fig. 1 Ⓐ) lists all available steps in their processing order. Each item provides the ID of the step, its description, and its event count. In addition, it shows an event share visualization similar to a Pareto chart [25], which is commonly used in the quality management domain. The event share is visualized as a bar, where the width of the bar represents the step's share relative to the total event count. Further, a line indicates the cumulative event share of the current and all previous process steps. Our industry collaboration partner's experts explained that such an information is an important aspect for prioritizing analysis tasks.

**Event types across process steps.** The data can also be filtered based on specific event types (see Fig. 1 Ⓑ). Each row corresponds to one type and comprises its ID, description, occurrence count (which is also the sorting criterion of the list), and the event distribution sparkline. Analysts get a quick overview about the event distribution along the production line.

**Product types across process steps.** Analysts can filter the data with respect to the products that were being produced when events occurred (see Fig. 1 Ⓒ). The *product type filter facet* is similar to the facet explained above, but instead of using event descriptions, the facet provides information about the events' distribution depending on the produced product. Each product is represented through a unique product number. The experts from our industry partner told us that this number is readable by analysts who are familiar with the production line. This way, it is possible to quickly see similar event distributions of different products.

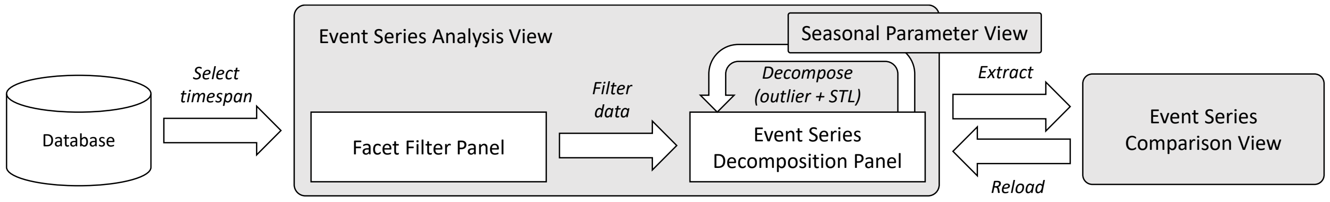

Database
Select timespan
Event Series Analysis View
Facet Filter Panel
Filter data
Seasonal Parameter View
Decompose (outlier + STL)
Event Series Decomposition Panel
Extract
Reload
Event Series Comparison View

Figure 4: At first, the users need to select, which timespan to analyze and what event type should be used. The data used in each analysis is set through facet views, which also provide an overview of the various data aspects through sparkline visualizations. After filtering the data, the users can inspect the aggregated event series in a line plot, which can be decomposed into a trend, seasonal and remainder component. In this view, analysts can iteratively extract outliers, trends, and seasonal patterns.

In case entries from multiple facets are selected, the data needs to meet at least one selection from each facet. The faceted views meet parts of requirement *R1 (Overview and faceted information)*.

## 4.2 Temporal Analysis using Event Series Decomposition

The analysis to find temporal event patterns is conducted in the *Event Series Decomposition Panel* (see Fig. 1 Ⓓ). For the temporal analysis, the filtered data are aggregated by the hour in which they occurred. Initially, this panel consists of four plots.

All series in the *Event Series Decomposition Panel* have a common x-axis that represents the loaded time frame. The y-axes represent the number of events at a point in time. It adapts to the data shown in the plot and not the global maximum to use as much space of the plot as possible. We do not optimize the diagrams' aspect ratio individually as suggested in literature [13] as there are large differences in the characteristics of the time series and optimization would result in considerably varying scales of the y-axis. Instead, we keep the heights of the diagrams constant in the juxtaposed plots to ease the visual comparison. We use the SciChart WPF Framework[1] to generate all plots in the *Event Series Decomposition Panel*.

The *event series plot*, shown at the top of the panel, is always available and shows the filtered data, as well as (optionally) the data's outliers, in a stacked bar chart. At last, the panel provides three (initially empty) line plots that, once the analysts choose to decompose the series, provide the trend, seasonal, and remainder components of the series. To provide a better comparability between the seasonal and trend series, their y-axes have a shared min-max range. The trend, seasonal, and remainder series are described in more detail later in this Section. The *event series plot* meets the temporal aspect of requirement *R1 (Overview and faceted information)*.

**Outlier configuration and extraction.** Analysts can optionally inspect and extract outliers from the event series plot. This is useful, because outliers may indicate unexpected events that should be further investigated. In addition, the extraction of outliers before the event series' decomposition improves the results, as STL may miss some outliers and include them in the seasonal component.

In case the outlier detection is enabled, we replace parts of the *event series plot* with red bars proportional to the outlier part of the series. To decide if and to what extend a point is an outlier, we use the data point's standard score, which describes how many standard deviations a data point deviates from the data series' mean value. We define a data point as an outlier, if its standard score is higher than *x*, which is four by default. However, the outlier threshold can be changed through a slider control, because it depends on the dataset and the analysts' notion what an outlier is.

Removing the entire outlier would likely cause another outlier, because no events during one hour are unlikely (except nothing is produced). Therefore, we calculate a compensated value for every outlier. First, we interpolate linearly between the previous and next

[1]https://www.scichart.com

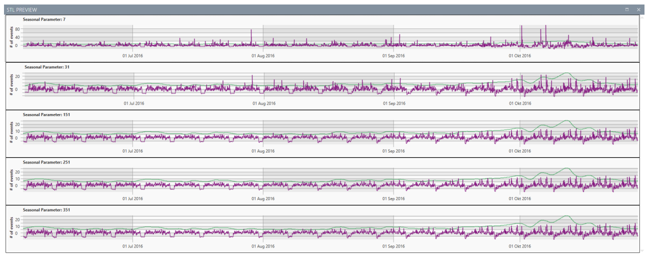


Figure 5: Exemplary depiction of a choice between various seasonal parameters used by STL. The first two options (with values of 7 and 51) still contain much noise, while the remaining three choices (with values of 151, 251, and 351) do not differ noticeably.

data point and then add or subtract *x* standard deviations to the interpolated value to move it towards the data point's actual value.

**Iterative Analysis of Trends and Recurring Behaviors.** After filtering the data and optionally extracting outliers, analysts can decompose the *event series plot* using Seasonal Trend Decomposition using Loess (STL) [10], which decomposes an event series into a trend, seasonal, and remainder component. The trend represents long-time effects in a time series. The seasonal component represents the recurring effects during the series. These are of the most interest, because recurring events may indicate systematic problems in the production line. With the visualization of the trend and seasonal component contribute towards requirement *R2 (Pattern & outlier identification)*. The remainder component represents the difference between the initial event series and the extracted trend and seasonal components. Thus, it consists of noise that is present in the data and non-extracted outliers. Further, it may contain non-extracted seasons that have a shorter length than the current season (longer seasons are extracted partly into the trend component).

Analysts need to provide three arguments to run STL: the time series represents the dataset used for the decomposition. In most cases, this is the entire series shown in the event series plot, but the users may manually select parts of the series (see Section 4.3). The seasonal period length defines the number of data points per season (e.g., 24 to analyze daily seasons). The strength of the seasonal smoother defines how strongly the seasonal component should be smoothened. A high value will lead to seasons with very low variations over time, while a low value allows high variation, which may also include noise. The strength of the smoothener cannot be predetermined, as the analysts needs to decide, how much variation is allowed in the seasonal component. As the analysts are usually no data scientists, we do not expose the other parameters and approximate them automatically (see Cleveland et al. [10]).

We provide a predefined set of seasonal period lengths to detect daily, weekly, or monthly patterns, but the period length can also be set manually (see Section 5 for details). Once the analysts decided on the event series and the period length, we provide them with a preview of the seasonal and trend component for different values for the smoothing parameter. Fig. 5 depicts an exemplary choice for

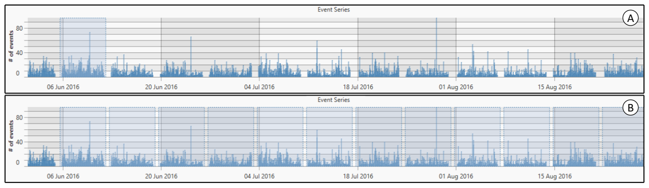


Figure 6: The users can define their custom event series by first selecting a start and endpoint of the first season occurrence (Ⓐ). Then, they define a gap (which can be zero) between the seasons' occurrences. To get a feedback of the selected pattern, the all seasons that fit in the event series are also highlighted (Ⓑ).

several seasonal parameters. The first and partially the second choice still contain much noise, and the remaining proposals show similar results, so an expert may choose the third parameter. We determined the proposed values empirically and deemed them appropriate for the decomposition of event series in a production line. However, this does not mean that they cannot be applied for other datasets. This contributes to requirement *R2 (Pattern & outlier identification)*.

It is important to understand a few fundamental concepts of STL to make sense of its results (we refer to Cleveland et al. [10] for technical details). The seasonal component is extracted in two steps. First, the input series is split into subseries that all have the input seasonal length. Then, the subseries' elements are aligned and smoothed (e.g., the *i*-th element from subseries *A* is only smoothed with the *i*-th element from subseries *B*, *C*, etc.). At last, the seasonal component is extracted. The seasonal smoothing parameter defines, how many of the neighboring elements should be considered during the smoothing operation and how strongly they affect the result. The trend component is extracted using all data points of the series in their temporal order. The remainder component represents all the data that was not extracted into the trend or seasonal component.

The resulting trend, seasonal, and remainder components are shown in their respective plots in the *Event Series Decomposition Panel*. In case the decomposition yields a season or trend that the analysts deem interesting and plausible, the current analysis state can be stored in the *Event Series Comparison View* (see Section 4.4).

**Calendar Plot.** To further supplement the analysis of the event series, we provide a calendar below the plots (see Fig. 1 Ⓗ). The analysts can switch the data source by through a combo box. If the data source is the event series, the analysts can further choose to show the regular data or the outliers. We used the same color scheme as in the *Facet Panel* (which ranges from a light yellow via orange and red to black). Initially, the calendar is grouped monthly and every entry represents one day, much like the calendar plot used by van Wijk and van Selow [27]. The analysts can change the time granularity of the calendar, so that the calendar's cells can represent hours, days, months, etc. If the calendar cells have a coarser granularity than the plot views (days, months, years, etc.), we calculate the average outliers/events per hour for the color mapping.

### 4.3 Inspection and Filtering of User-Selected Data

In addition to the explicit filter facet, the analysts can retrieve information about event types by directly interacting with the event series plot. They can either inspect reported events at a certain time (e.g., one specific hour) or during a time interval (from 8 am to 8 pm). A specific point in time can be selected by right-clicking. An extended time frame is selected by brushing over the interesting part of the *event series plot*. The selection is highlighted with a light blue background (see Fig. 6Ⓐ). Then, the selected event reports can be retrieved by right-clicking the highlighted area.

The retrieved event types are listed to the right of the event series (see Fig. 1 Ⓔ). The listed event types' selection is coupled with the event type filter facet, so selecting an event type in one filter will also select it in the other. The context dependent event type list provides a secondary filter option and therefore supports requirement *R1 (Overview & faceted information)*.

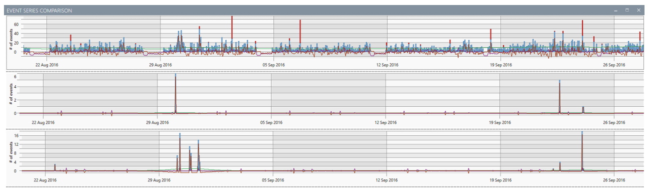


Figure 7: The event series comparison view presents stored event series by visualizing its event series and (if extracted) their outliers as a stacked bar chart, which is overdrawn by line charts of the decomposed event series. This example shows the original event series as well as two subseries of different event types.

### 4.4 Event Series Comparison

Analysts can store insights gained during the analysis process by storing the current filter settings and decomposition results in the *Event Series Comparison View* (see Fig. 7). The series are visualized as a superimposed representation. To perceive the original data as well as its decomposed series, the stacked bar chart containing the event series and outliers is overlaid by the line charts of the decomposed event series. The analysis configuration can also be restored to the *Detailed Analysis View* to continue the analysis. Fig. 7 presents an exemplary comparison that contains the original data series, as well as two series filtered by two different event types. The series' events occur at very similar points in time, which may indicate that the events may be related.

## 5 Selection and Optimization of Parameters

If the predefined STL seasonal periods are insufficient, analysts can define a custom pattern. First, they need to select the pattern's first occurrence analogous to Section 4.3. Further, analysts can indicate a timespan to skip during the decomposition by dragging a replica from the highlighted pattern to the next occurrence. We highlight the user-defined event series across the *event series plot* to make the resulting series more comprehensible (see Fig. 6 Ⓑ.

If the event series comprises a high number of data points, analysts may have problems to set the exact values they desire. Thus, we developed an optimization approach based on an evolutionary algorithm that automatically improves the users' input parameters.

Similar to genetic evolutionary algorithms, our implementation comprises the following six steps, where step (0) is executed just once and steps (1)–(5) repeat *n* times:

**0. Initialization.** We create an initial set of 30 parameter configurations. The parameter values are randomly initialized with a uniform distribution around the users' input values. The starting point can vary by up to ±24 hours, whilst the period and gap length can vary by up to ±48 hours.

**1. Fitness Evaluation.** To evaluate each parameter configuration, we first decompose the event series based on the configuration's parameters and then extract its seasonal component ($cs$). Our goal is to search for a seasonal series that will be as uniform as possible when split into its seasonal periods. To do so, we split $cs$ into its seasonal subseries $cs_i$. Then, we build an average subseries $cs_{avg}$ of the selected and the following two subseries, assuming that the searched pattern will be most visible close to the first pattern occurrence. Then, we calculate the average variation between the first three subseries $cs_i$ and $cs_{avg}$ and build the average of the variation series' values ($cs_{var}$). We define the fitness $f$ as

$$f(cs) = \frac{cs_{var}}{\underbrace{(max(cs) - min(cs))^2}_{\text{Range of the series' values}}} .$$

It should be noted that a lower fitness value is better.

2. **Keep Best Configurations (Elitism).** To save the best results, we transfer the 10 % best configurations to the next generation.
3. **Discard Unfit Configurations.** The lowest scoring 50 % of the configurations are removed.
4. **Recombination.** New parameter configurations are created by combining two still existing "parent" parameter configurations $A$ and $B$. The new configurations' values $v$ are based on a weighted average between the parents' values $v_A$ and $v_B$. The parameter $\alpha$ is randomly picked based on a uniform distribution:
$$v = \alpha \cdot v_A + (1 - \alpha) \cdot v_B, \text{ where } \alpha \in [0, 1]$$
5. **Mutation.** At last, the new configurations' parameter values have a slight change ($p = 0.1$) to be randomly changed within the variation explained in the initialization step. Values that are out of bounds (e.g., a start date earlier than the first data point's date) are assigned the outmost allowed value.

We update the user-defined selection whenever a better result was found. Due to step (5), which broadens the pool of available parameter configurations, an evolutionary algorithm is more robust regarding local optima than simpler optimization approaches (such as hill climbing).

**Results** The optimization approach was not part of the pair analytics sessions presented in Section 6, as the domain experts had no experience with our approach and we wanted to evaluate the general approach first. In the following, we present and discuss an exemplary result achieved with our input optimization approach.

As a proof of concept, we evaluated, if our evolutionary algorithm optimization is able to recommend a weekly pattern for the unfiltered event series. The unfiltered event series provides only one trivial insight, but it is clearly visible that the seasonal component reflects the weekends during which usually no events were reported. Fig. 1 Ⓓ & Ⓖ show, what the decomposition should look like.

We hypothesized, that the evolutionary algorithm recommends that the season and the season gap should add up to seven days and that the period length should be between five and seven days.

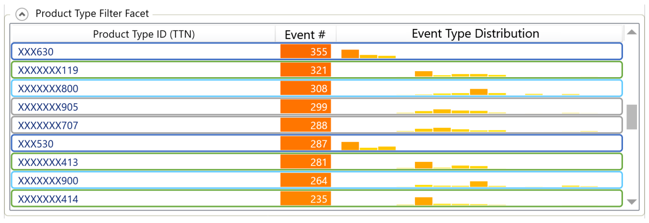
We ran multiple test runs and manually set the period length to be six days and the gap between the seasons to one day to cover our assumed results, but also allows results outside of our assumption (e.g., four days without a gap). Our results show, that a seven days period without a gap achieves the best result (f = $6.66 \cdot 10^{-5}$), whereas a period length of five or eight days without a gap resulted in the worst results (f = $1.71 - 1.74 \cdot 10^{-4}$). We also noticed, that a seasonal length of six days with one day gap resulted in suboptimal results (f = $1.18 \cdot 10^{-4}$). We assume that the variation on Sundays is still a better trade-off than ignoring Sundays and only considering the variation of the remaining six days.

Overall, it can be said that our optimization approach is promising, but there is still room for improvement regarding the handling of the data during the gaps between the seasonal patterns.

## 6 Evaluation

We evaluated our approach in two pair analytics sessions [3] with experts from our industry partner. In pair analytics, one or more domain experts work together with a visual analytics expert. The domain experts contribute their expertise and experience to focus the analysis on interesting data, whilst the visual analytics expert is operating the visual analysis tool.

We prepared the pair analytics session by searching for potentially interesting patterns by ourselves, which we used as a starting point during the pair analytics sessions. In the following, we will first showcase three use cases we derived during the pair analytics sessions to demonstrate, how our approach contributes to the analysis or event patterns in a production line. All of the use cases are based on real production data that spanned roughly six months. Only the visual analytics expert is one of the co-authors of this paper. We worked together with two domain experts: the first expert is responsible for the data acquisition and to make the data available to workers on the shop floor of the factory. He was also involved in the development process of our approach. The second expert is head of a production line and was consulted only for the evaluation of our approach. Each of our pair analytics sessions lasted approximately 60 minutes. Furthermore, we present general feedback we collected from the domain experts after the pair analytics sessions. As explained in Section 3.2, we limited the data to events that have the severity level *error*, as these events are most important. Therefore, we will refer to events as errors for the remainder of this section.



| Product Type ID (TTN) | Event # | Event Type Distribution |
|---|---|---|
| XXX630 | 355 | |
| XXXXXXX119 | 321 | |
| XXXXXXX800 | 308 | |
| XXXXXXX905 | 299 | |
| XXXXXXX707 | 288 | |
| XXX530 | 287 | |
| XXXXXXX413 | 281 | |
| XXXXXXX900 | 264 | |
| XXXXXXX414 | 235 | |

Figure 8: The distribution of the errors per product type along the production line shows several groups of error distribution (highlighted in different colors). Our domain experts already assumed such profiles. Further, they noticed that these profiles do not always correlate to the different groups of produced products (e.g., electric motors with and without a power train extension).

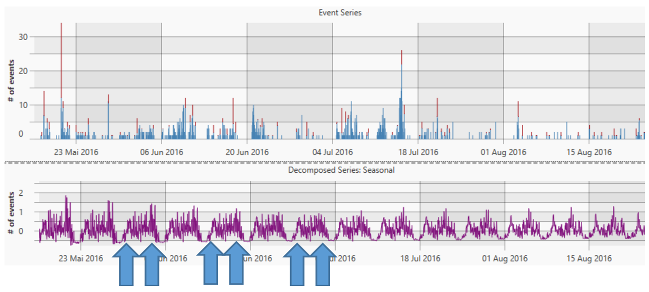
### Use Case 1: Error distribution along a production line

Before we started with the first detailed analysis, one expert mentioned that the error distribution shown in the error code facet differs from what he expected. There are, generally speaking, four different types of electric motors produced in the production line. He expected that the four types would have different error distributions when compared to each other and that variations of the motor types would have similar errors and error distributions. However, the data only partly backs up these assumptions. There are similarities in the error distributions, but the motor type is not always the same. Fig. 8 shows an excerpt from the product type facet, wherein the different error distributions are highlighted. The similar distribution is partially explainable, because even if the motor types are different, they still share parts of their production plan. However, some of the similarities were not explainable this way and further investigations in cooperation with workers on the shop floor level are required to assess other reasons.

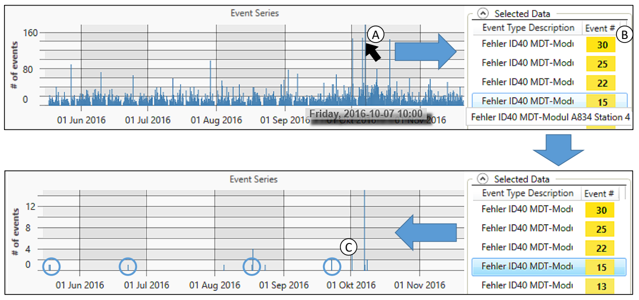
### Use Case 2: Analysis of Recurring Error Patterns

When we continued, the domain expert was interested in the most often occurring error code. This error was at the *label checker station* that has to take the part off the work piece carrier to process it. We decomposed the series with a period length of one week to check for error patterns, as the line's production schedule usually repeats every week. Fig. 9a shows the error series and the resulting seasonal plot. The seasonal pattern shows two characteristics: first, the pattern starts with a short period of time, where no errors occurred. This is expectable, as the line usually does not produce anything during Sundays and therefore there cannot be any errors during that time. However, the pattern disappears towards the end of the series, as the production line runs more often on Sundays.

Second, the number of reported errors is remarkably higher towards the beginning and the end of the week compared to other days. The head of the production line hypothesized that this finding may be caused by the quality of the imported parts used during the production. He explained that the line usually uses parts from a main supplier, but towards the end of the week, these parts often run out. In that case, they switch to parts from another supplier, whose parts have a higher quality variance than the main supplier's.



(a) The data shown in the event series plot (top) is decomposed to search for weekly seasonal patterns. There are no errors on Sundays (flat line before the arrow pairs) and the amount of errors is increasing at the beginning and end of the week. The arrows indicate at what times the overall number of errors increases.



| Event Type Description | Event # |
|---|---|
| Fehler ID40 MDT-Modı | 30 |
| Fehler ID40 MDT-Modı | 25 |
| Fehler ID40 MDT-Modı | 22 |
| Fehler ID40 MDT-Modı | 15 |
| Fehler ID40 MDT-Modı | 13 |

(b) The inspection of one of the error series' outliers reveals that the same component caused issues on various stations along the production line within an hour. When filtering for individual errors (highlighted entry in the tooltip) it becomes apparent that the errors repeat approximately every month (highlighted through circles).

Figure 9: Results of use case 2 (a) and use case 3 (b)

Although the parts' quality is mostly within the allowed margin of error, the stations that process these parts have a higher likelihood to encounter problems when processing these parts. This hypothesis was supported when the expert accessed the logistic department's inbound delivery list, which our tool cannot access. He further mentioned that they were aware of this issue before. However, they were not able to prove that this is a regularly recurring issue, because they were not able to back this hypothesis with data using their previous analysis methods. Therefore, this finding is helpful because it can be used to argue for an improvement of the robustness of stations that need to process the mentioned supplied parts.

### Use Case 3: Analysis of Outliers for Pattern Analysis

The error series of the *Event Series Decomposition Panel* shows a remarkable outlier in the second half of the data that we used during the evaluation (see Figure 9b Ⓐ). We selected the error distribution during the largest outlier's point in time by selecting it. Almost all errors were related to the *ID-40 module*, which is a sensor that reads the ID of the work piece holders to provide tracking within the production line (Fig. 9b Ⓑ).

Usually this error indicates a broken sensor, but in this case, the error was reported from various stations at the same time. One expert explained that such an error distribution may indicate a problem with the bus system that connects the sensors to the IT infrastructure. We filtered some of the errors one after another and noted that the errors repeat approximately every month (see Fig. 9b Ⓒ).

The head of the production line stated that this is an interesting and unexpected finding that they were not aware of before. As a result, they will cover both possible scenarios. First, the technicians will be informed to watch such sensor errors more closely. Second, the finding is forwarded to the responsible department, as it is not possible to fix it by improving the production line, but other lines in the factory may be affected by this problem as well. Some time after the analysis we were told that parts of the identified issues were caused by a bug in one of the machine tool's programs, which was resolved.

### General Expert Feedback

After the pair analytics sessions, we asked the participating experts about feedback regarding the general approach and the different components through a semi-structured interview. We asked the experts about advantages and drawbacks of the currently available views and if they could imagine any enhancements that would help them with their work. They first remarked that our approach results in a powerful tool that must be used by an engineer in a supervising position, as shop floor technicians have neither time nor necessary expertise to analyze such error patterns. They also mentioned that this is not a problem, as experts such as production line heads can use our approach and forward the gained insights to the technicians at the show floor level.

The experts summarized that the *Event Series Comparison View* is especially useful to see what data was already analyzed in the past. They further explained that the iterative analysis approach on the overview and detailed analysis level is helpful. It enables experts to either pursue a specific error until its occurrence is completely understood (varying STL configurations) or to analyze the most important errors separately (varying filter configurations). The overview of the analyses provided in both views is important because the analysis runs are often interrupted, e.g., because talking to specialists is required to solve an issue. The experts further inquired that the *Event Series Comparison View* should be extended to contain information about the shown series', such as the used filters.

The experts also found the *Facet Panel* useful. They mentioned that the product type facet is very helpful to gain more insight about the errors' distributions. The process step and event type facets are also useful, but unlike the product part facet, they cannot be used for a free exploration of the data. Instead, the analysts must already have a specific analysis goal that requires a selective investigation.

The experts stated that the extraction of outliers and seasonal trends are both of importance. An outlier extraction can be used to detect special events that may require special attention. The seasonal series extraction enables to form hypotheses about systematic errors that are backed by the available error reports. Furthermore, the experts emphasized that engineers are usually not experts in statistical analysis and thus require an easy to use approach to decompose the temporal error series. They understood that our approach exposes only the required parameters towards the user. However, the results are still sometimes hard to interpret, so they asked for more indications that help them to configure the analysis. One of the experts imagined that the system could suggest decomposition configurations automatically without the need to set these parameters manually. An expert would still have to evaluate which suggestions are useful, but it would make their analysis process easier. At last, the experts rated the *Calendar Plot* to be useful, because it gives a different view at the time series and also provides information about other time granularities, such as a daily or monthly level of detail.

## 7 Discussion, Future Work, and Lessons Learned

During the evaluation, the domain experts suggested that the *Analysis Summary View* should contain more information about already performed analyses. This information could comprise used filters, when was the analysis opened last, and others. In addition, we intend to support the comparison of detailed analyses to provide feedback about similarities across multiple analysis runs.

In use case 1, the experts got first insights through the sparkline visualization in the product variant and event type facet views. However, they might have found similar event behaviors of different products by chance, as the compared product variants must be both visible in the list to be comparable. This issue is caused by our current sorting by total event count. We plan to extend the *Facet Panel* so that the process steps can be filtered to those of special interest. Further, their order could follow other criteria, such as the similarity of the events' distribution.

In our approach, we use STL, because it returns the series' trend and seasonal component independently. Further, STL requires only three input arguments (the time series, the seasonal period length, and the seasonal smoother's strength). However, it still takes some time to understand the approach and the domain experts are not experts in data analytics. Therefore, we plan to try other decomposition algorithms and evaluate their suitability. Algorithms that are, like STL, based on an auto-regression integrated moving average (ARIMA) model seem to be best suited. The most important evaluation criteria will be comprehensibility and steerability.

We also want to test to automatically propose interesting seasonal intervals. One possibility would be to apply the optimization approach to the whole event series plot without the restrictions of a user-defined input.

Further, it may be hard to understand how an evolutionary algorithm compares various parameter configurations. Thus, we plan to test other optimization algorithms and enable the user to better influence the evaluation criteria of a "good" parameter configuration.

During the meetings with our industry partner, we realized how important it is to provide domain experts with visualization techniques that they are already familiar with. The experts first search for facts they already know to test the new approach's credibility. If they would have to learn a new visualization metaphor first, the experts may lose interest in before understanding it. Using facets, stacked bar charts, and line plots provides this low barrier.

Further, the data of our industry partner is dynamic and evolves over time; for example, the amount of collected data increases due to more sensors or newly defined event types. Also, its quality improves through refined event types and patterns may disappear because they are solved. This needs to be taken into consideration when analyzing recurring patterns.

## 8 Conclusion

In this paper, we presented a visual analytics approach to support manufacturing domain experts in analyzing temporal event patterns in production lines. Our concept was developed in cooperation with an industry partner that produces electric motors. The approach comprises two views that support the events' temporal pattern analysis. The *Detail Analysis View* combines facets to filter data with Seasonal Trend decomposition using Loess (STL) to iteratively extract event patterns, which may indicate systematic problems in a production line. The *Event Series Comparison View* provides an overview of the extracted patterns and enables analysts to compare them visually. We evaluated our approach through three use cases that were worked out in two pair analytics sessions with experts from our industry collaboration partner to demonstrate our approach's usefulness in the manufacturing domain. Additionally, we presented expert feedback and possible future extensions for our approach.

## Acknowledgments

We thank our industry collaboration partner for the provided feedback during the development and the evaluation of the presented approach.

## References

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented data*. Springer, 2011.

[2] P. André, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and M. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. In *Proc. of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST, pp. 101–110. ACM, 2007.

[3] R. Arias-Hernandez, L. T. Kaastra, T. M. Green, and B. Fisher. Pair analytics: Capturing reasoning processes in collaborative visual analytics. In *44th Hawaii International Conference on System Sciences*, pp. 1–10. IEEE, Piscataway, N. J., 2011.

[4] K. Beard, H. Deese, and N. R. Pettigrew. A framework for visualization and exploration of events. *Information Visualization*, 7(2):133–151, 2008.

[5] M. Bögl, W. Aigner, P. Filzmoser, T. Gschwandtner, T. Lammarsch, S. Miksch, and A. Rind. Visual analytics methods to guide diagnostics for time series model predictions. In *Proc. of the IEEE VIS 2014 Workshop Visualization for Predictive Analytics*, VPA, 2014.

[6] M. Bögl, W. Aigner, P. Filzmoser, T. Gschwandtner, T. Lammarsch, S. Miksch, and A. Rind. Integrating predictions in time series model selection. In *Proc. of the EuroVis Workshop on Visual Analytic*, EuroVA, pp. 73–77. The Eurographics Assoc., 2015.

[7] M. Bögl, W. Aigner, P. Filzmoser, T. Lammarsch, S. Miksch, and A. Rind. Visual analytics for model selection in time series analysis. *IEEE Transact. on Visualization and Computer Graphics*, 19(12):2237–2246, 2013.

[8] J. V. Carlis and J. A. Konstan. Interactive visualization of serial periodic data. In *Proc. of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST, pp. 29–38. ACM, 1998.

[9] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *Proc. of the 2012 IEEE Conference on Visual Analytics Science and Technology*, VAST, pp. 143–152. IEEE, 2012.

[10] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. STL: A seasonal-trend decomposition procedure based on Loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[11] C. B. Frey and M. A. Osborne. The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 2016.

[12] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[13] J. Heer and M. Agrawala. Multi-scale banking to 45 degrees. *IEEE Transact. on Visualization and Computer Graphics*, 12(5):701–708, 2006.

[14] C. Humphries, N. Prigent, C. Bidan, and F. Majorczyk. ELVIS: Extensible log visualization. In *Proc. of the Tenth Workshop on Visualization for Cyber Security*, VizSec, pp. 9–16. ACM, 2013.

[15] J. Jo, J. Huh, J. Park, B. Kim, and J. Seo. Livegantt: Interactively visualizing a large manufacturing schedule. *IEEE Transact. on Visualization and Computer Graphics*, 20(12):2329–2338, 2014.

[16] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *Proc. of the 6th Conference on Visualization*, InfoVis, p. 279. IEEE, 1995.

[17] B. Klopper, S. Honiden, J.-P. Pater, and W. Dangelmaier. Decision making in adaptive manufacturing systems: Multi-objective scheduling and user interface. In *IEEE Symposium On Computational Intelligence In Control And Automation*, pp. 123–130, 2011.

[18] M. Krstajic, E. Bertini, and D. A. Keim. CloudLines: Compact display of event episodes in multiple time-series. *IEEE Transact. on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[19] J. Landstorfer, I. Herrmann, J.-E. Stange, M. Dörk, and R. Wettach. Weaving a carpet from log entries: A network security visualization built with co-creation. In *Proc. of the 2014 IEEE Conference on Visual Analytics Science and Technology*, VAST, pp. 73–82. IEEE, 2014.

[20] D. Luo, J. Yang, M. Krstajic, W. Ribarsky, and D. Keim. Eventriver: Visually exploring text collections with temporal references. *IEEE Transact. on Visualization and Computer Graphics*, 18(1):93–105, 2012.

[21] R. Maciejewski, R. Hafen, S. Rudolph, S. G. Larew, M. A. Mitchell, W. S. Cleveland, and D. S. Ebert. Forecasting hotspots—a predictive analytics approach. *IEEE Transact. on Visualization and Computer Graphics*, 17(4):440–453, 2011.

[22] N. B. Robbins. Introduction to Cycle Plots. Visual Business Intelli-

gence Newsletter, 2008.

[23] M. W. Rohrer. Seeing is believing: The importance of visualization in manufacturing simulation. In *Proc. of the 32nd Conference on Winter Simulation*, WSC, pp. 1211–1216. Society for Computer Simulation International, 2000.

[24] M. H. Shimabukuro, E. F. Flores, M. C. F. de Oliveira, and H. Levkowitz. Coordinated views to assist exploration of spatio-temporal data: A case study. In *Proc. of the Second International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pp. 107–117. IEEE, 2004.

[25] N. R. Tague. Seven basic quality tools. *The Quality Toolbox. Milwaukee, Wisconsin: American Society for Quality*, p. 15, 2004.

[26] E. R. Tufte. *Beautiful Evidence*. Graphics Press, 1st ed., 2006.

[27] J. J. van Wijk and E. R. van Selow. Cluster and calendar based visualization of time series data. In *IEEE Symposium on Information Visualization*, pp. 4–9, 1999.

[28] M. Weber, M. Alexa, and W. Müller. Visualizing time-series on spirals. In *Proc. of the IEEE Symposium on Information Visualization 2001*, vol. 1 of *InfoVis*, pp. 7–14. IEEE, 2001.

[29] M. Wörner and T. Ertl. Visual analysis of advanced manufacturing simulations. In *EuroVA 2011: International Workshop on Visual Analytics*. The Eurographics Assoc., 2011.

[30] P. Xu, H. Mei, L. Ren, and W. Chen. ViDX: Visual diagnostics of assembly line performance in smart factories. *IEEE Transact. on Visualization and Computer Graphics*, 2017 (to appear).

[31] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI, pp. 401–408. ACM, 2003.